Locomotion for Crowd Animation

by

Martin Pražák

A dissertation submitted to the University of Dublin, Trinity College
in fulfillment of the requirements for the degree of
Doctor of Philosophy

May 2012

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and that unless stated, it is entirely my own work.
I agree that the library may lend or copy the thesis upon request. This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgement.

Martin Pražák
17th May 2012

Summary

Real-time computer animation is an essential part of modern computer games and virtual reality applications. While rendering provides the main part of what can be described as "visual experience", it is the movement of the characters that gives the final impression of realism. Unfortunately, realistic human animation has proven to be a very hard challenge.

Some fields of computer graphics have a compact and precise mathematical description of the underlying principles. Rendering, for example, has the rendering equation, and each realistic rendering technique provides its approximate solution. Due to its highly complex nature, character animation is not one of these fields. That is one of the reasons why even single character animation still provides significant research challenges. The challenges posed by a crowd simulator, required to populate a virtual world, are even larger. This is not only because of the large number of simultaneously displayed characters, which necessitate the use of level-of-detail approaches, but also the requirement of reactive behaviour, which can be provided only by a complex multi-level planning module.

In this thesis, we address the problem of human animation for crowds as a component of a crowd simulator.

To ensure that we start with realistic animation data, we record our data using a camera-based passive optical motion capture system. We provide a detailed description of our camera setup, our human body model and our refined pipeline, which together allow for robust and precise human motion reconstruction and its usage in a real-time system.

While the captured motion data provide an accurate representation of human motion, their direct usability in a real-time system is limited to a simple playback of the original clip on a human model with body proportions corresponding to that of the original actor. To overcome this limitation, a data-driven animation synthesis method has to be incorporated to create novel animations based on the source data. The solution described in this thesis uses a parametric data-driven locomotion synthesis model, with particular focus on motion synthesis for crowds. This description includes motion preprocessing, periodisation, parametric motion blending structure and its integration with a high-level behaviour module.

Even though a compact and general mathematical representation of human motion is problematic, humans show an impressive, yet intuitive, understanding of human motion. Taking into account that a resulting animation is almost exclusively presented to a human observer, an effective way to determine the properties of an animated motion is to explore how it is perceived by the human visual system. To this end, we conducted several perceptual experiments with a view to devising metrics which would be directly applicable to a crowd animation system. Specifically, we present a method to compare human locomotions in a perceptually correct manner; establish the minimal number of characteristic animations required for a group of characters to appear varied; and determine the perceptual impact of two common animation artifacts caused by motion editing – animation timewarping and footsliding.

Acknowledgements

Unfortunately, the acknowledgments section is not long enough to mention all the people who helped me on this long journey. Therefore, to begin with, I want to thank all my friends, flatmates, family, colleagues and fellow students, and everybody else who I met on the way. Even if I don't mention your name here, I want you to know that I am grateful.

On a personal note, I would like to thank Giulia for the very unreasonable amount of support, shelter, plans, love and pasta – I would not have been able to go this far without you. Then my flatmates, Bartosz, Suule and Princess, with all their ideas, stories and distractions – thank you for all the sweets you didn't eat. Finally, I can't forget to mention Lucka for her occasional invasions into my life, and, of course, the unconditional support of Juli and Lisa - thug je che!

In the academic world, I was fortunate to have Prof. Carol O'Sullivan as my supervisor. Her invaluable advice and support guided me through the labyrinth of research, and allowed me to explore concepts and ideas I would never have explored by myself. My examiners, Dr John Dingliana and Dr Ronan Boulic, provided a very professional assessment of my work during and after my viva, and helped to refine this dissertation into its final form. Dr Ladislav Kavan and Dr Daniel Sýkora showed me the way in their respective fields, which form the very basis of a large part of my work. And finally, I couldn't forget to mention Dr Rachel McDonnell, Dr Ludovic Hoyet, Colin Fowler, Jiang Zhou, Tom van Eyck, Dr Cathy Ennis, Dr Micheal Larkin and all the other members of GV2, as without their help this thesis wouldn't have been possible.

Even though many people helped me to eliminate as many errors and omissions as possible, the ones that may remain are, of course, entirely my own responsibility.

Publications

Related Publications

1. Prazak, M., McDonnell, R., Kavan, L., & O'Sullivan, C. (2008). Towards a perceptual metric for comparing human motion. In *Poster proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 13–14)

2. Prazak, M., McDonnell, R., Kavan, L., & O'Sullivan, C. (2009). A perception-based metric for comparing human locomotion. In *Proceedings of the 9th Irish Workshop on Computer Graphics* (pp. 75–80)

3. Ruttle, J., Manzke, M., Prazak, M., & Dahyot, R. (2009). Synchronized real-time multi-sensor motion capture system. In *ACM SIGGRAPH Asia 2009 Sketches & Posters* (pp. 16–19)

4. Prazak, M., Kavan, L., McDonnell, R., Dobbyn, S., & O'Sullivan, C. (2010a). Moving crowds: A linear animation system for crowd simulation. In *Poster Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*

5. Prazak, M., McDonnell, R., & O'Sullivan, C. (2010b). Perceptual Evaluation of Human Animation Timewarping. In *ACM SIGGRAPH Asia 2010 Sketches* (pp. 30:1–30:2)

6. Prazak, M., Hoyet, L., & O'Sullivan, C. (2011). Perceptual evaluation of footskate cleanup. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 287–294)

7. Prazak, M. & O'Sullivan, C. (2011). Perceiving human motion variety. In *Proceedings of the symposium on Applied Perception in Graphics and Visualization*

Other Publications

1. Pouli, T., Prazak, M., Zemcik, P., Gutierrez, D., & Reinhard, E. (2010). Rendering fur directly into images. *Computers & Graphics*, 34(5), 612–620

Contents

List of Figures

1 Introduction

Through the influence of the entertainment industry, virtual reality and artificially created environments became a part of our daily lives. With the help of modern computer graphics, artists can express their visions in virtual worlds with limitless potential. Advances in rendering methods allow these worlds to become extremely believable; particularly so in movies, where even very complex models of physical behaviour can be implemented, and every little aspect can be defined in advance and fully controlled by the artist. But even recent games and virtual reality applications, operating under very limiting constraints of responsiveness and high framerates, can provide a real-time immersive experience very close to that of the film world.

However, to complete the illusion of a living world, impressive visual scenes are not enough – they need to be inhabited by virtual humans, which in turn have to be moving and interacting, both with the environment and among themselves. For certain purposes, we need many humans – from dozens to simulate a tourist group, through thousands to create a protest march or a virtual army, up to millions if we want to recreate whole cities.

The movie industry offers many interesting examples of large crowds. Older movies, such as The Last Emperor (1987) or Stargate (1994) would employ thousands of actors to create their crowd scenes. However, the same effect and much more can be achieved using computer graphics (CG), with significant savings in effort, finance and manpower. CG can create a large variety of crowd scenarios – from highly-stylised crowds (I, Robot, 2004; Antz, 1998), through fantasy-like humanoid characters (Lord of the Rings Trilogy, 2001-2003; the Chronicles of Narnia, 2005), up to very realistic crowds in a stylised environment (Inception, 2010) or during natural disasters ("2012", 2009).

Certain virtual applications and computer games require large crowds as well. The Fifa Soccer series (1995 - 2011) demonstrates a progress in virtual cheering crowds – from simple repetitive textures in Fifa 1995 to very realistic crowds in Fifa 2012; although the background nature of these characters does not require any significant interactions or complex behaviours. Many strategy games employ large armies of characters (e.g., Age of Empires series (I - III)), but the bird's eye point of view and

Figure 1.1.: Metropolis crowd simulation system.

small size of individual characters simplifies the task significantly. Several recent games employ crowds in the most natural and challenging 1st (or close 3rd) point of view (e.g., Assassins Creed I and II). While extremely impressive considering the level of crowd interaction and real-time framerates on common hardware, these games also demonstrate the shortcomings of current state-of-the-art crowd techniques, such as the lack of responsiveness and animation artifacts.

The main topic of this thesis is to address the animation aspects of crowd simulations, creating an entire city of realistically moving characters in real times. Naturally, in a city simulation, most of these characters will be walking, hence the focus on the locomotion of humanoid figures. Our system should be capable of providing an artifact-free animation for each individual character based on example animation data (with a very low computational cost), while providing an efficient interface to the behaviour module and the necessary animation variety.

1.1. Motivation

As the field of data-driven character animation matures, its methods provide increasing levels of animation realism for single characters. Several distinct groups of methods evolved, each with its own specific set of properties and limitations (see Section 2.5). However, a large majority of these methods either scale linearly with the number of characters, with a relatively large computational cost for every character, or do not allow the necessary animation variety required for a crowd. This is an important issue, as the number of characters in a crowd can be very high. However, not all of them will be visible at high detail all the time; an aspect described in the rendering systems as *level of detail* (LOD).

Therefore, we aim to create a data-driven crowd animation model effective enough to animate a large crowd, in a level of detail high enough to provide realistic animation, but at the same time low enough to avoid any unnecessary computation. Moreover, the computational and storage costs required for storing a new characteristic animation style should be minimised, with the style creation automatised as much as possible. Lastly, the animation should be responsive, to enable interactive crowd simulations that are influenced by the user.

The final product of each animation synthesis algorithm is presented to a user. Therefore, human perception plays an important role in animation synthesis, but the perceptual thresholds and properties of many different animation synthesis aspects remain largely unexplored. As they can provide interesting insights into the perceivable properties of human motion, and provide ways to simplify crowd animation without negatively influencing the visual aspects of the result, perceptual experiments are an interesting direction to be explored.

1.2. Scope

In this thesis, we focus on the animation aspects of human locomotion and their perceptual properties, in the context of crowd animation synthesis.

Apart from an animation module, a typical crowd system has to include high level behaviour and rendering components. However, these latter two topics are not within the scope of this thesis; we only describe the layers connecting them to our animation system. For behaviour, we assume the existence of a system that provides the position and velocity of every character for every frame, with values within a range natural for a human actor. The rendering should be performed by a state of the art hardware accelerated graphics engine, that describes the character's deformation using a set of 4x3 matrices.

Even though we use a certain amount of procedural animation editing, our model is primarily data-driven and requires a motion database as its input. These data are captured using our passive optical motion capture system, a description of which is included. While other data sources for our animation model are possible, they are not described as a part of this thesis. We assume that the data do not contain explicit constraints information (e.g., footstep timing and positions), and therefore require a constraints detection step to be performed during a preprocessing stage.

Apart from the behaviour and rendering modules, we assume that a level of detail (LOD) system is in place, which separates the visible crowd into several groups with different display and animation accuracy requirements. As such a system is a standard part of either behaviour or rendering module (or a combination of both) of a crowd simulator, this is a generalisable assumption. The main part of our work is focused on the *middle* level of detail characters, with characters large enough to be individually recognisable, but small enough for the animation not to require an excessive level of accuracy. Generally, the high LOD characters are in the foreground, usually numbering less than 100; the middle level of detail characters can number up to several thousand; and the low level of detail, with characters spanning no more than several pixels of the screen, can be animated using a very simple model.

As our perceptual experiments are designed to answer questions arising from the technical and implementation aspects of the thesis, we use only humanoid characters as the experiment stimuli. However, their representation varies depending on the type of the experiment from a simple stick-figure, through a wooden mannequin to a realistic skinned human model. The number of simultaneously displayed characters also varies depending on the experiment goal from a single character to a group of 25.

While we limit our scope mainly to crowd animation, almost all developed techniques and established perceptual properties apply to single character animations as well. For a list of limitations of our implementation and results please refer to Section 7.2.

1.3. Methodology

In a similar way to the rest of the thesis, our methodology can be separated into the implementation and technical part, and the techniques used for the design and evaluation of perceptual experiments.

1.3.1. Implementation Methodology

The implementation process was performed in cycles, each containing a design, implementation and evaluation stage. While these cycles are not explicitly reflected in the thesis, this methodology facilitates the continued building and refinement of the existing set of techniques and connected code base. The evaluation stage can be performed in several different ways – visually, if the algorithm contains easily identifiable artifacts not predicted in the design stage, using a suitable evaluation metric, or

Figure 1.2.: The Biodancer installation screenshots.

rigorously by performing a formal perceptual experiment.

The implementation uses exclusively the C++ programming language, extensively incorporating the Standard, Boost, OpenGL and Wild Magic libraries. A large number of other libraries were used for specific tasks, such as ANN (Approximate Nearest Neighbour), SDL (Simple DirectMedia Layer), wxWidgets, TinyXML, FreeImage and NaturalMotion Morpheme. With the exception of the Metropolis project, all development was done in the Linux operating system (Ubuntu, Debian), using the GCC compiler and the CMake building system. To allow the use of the run-time system inside the Metropolis project, a part of the developed code was ported to the MS Windows and Visual Studio 2005 compiler. The visualisation of the algorithms' results, both for debugging purposes and for perceptual experimentation, also includes several GLSL shaders.

1.3.2. Perceptual Experiments and Evaluation

All the perceptual experiments presented in this thesis use standard methods developed in the field of psychophysics. The procedures (tasks) used are variations of the standard *n-alternative* or *n-interval forced choice task*, where several stimuli are presented, separated either spatially or temporally, on a computer screen (please see the description included in each experiment section for more details).

The majority of the experiments use the *method of constant stimuli*, where a combination of presented stimuli are selected from a set of pre-generated examples. Each possible combination of the tested factors, as represented in the example database, is shown to the user several times, leading to a reliable measure of the responses. The significance of the results is tested using a repeated measures analysis of variance (ANOVA), with a post-hoc analysis using the Newman-Keuls comparison of means, or by performing the standard t-test. All the presented graphs show the average values for significant factors and their standard error.

For the particular purpose of accurate threshold estimation in the footskating experiments (see Section 6.3.2), we employ the *adaptive staircase method*. Its evaluation is performed by fitting a psychometric curve to the results, with the point of subjective equality (PSE) and just-noticeable difference (JND) parameters determined from the curve shape.

1.4. List of Contributions

The contributions of this thesis can be separated into three groups, which consist of three different aspects of the research.

The first group, **technical contributions**, focuses on the development of a framework for large crowd animation. The contributions include:

- *a refined motion capture pipeline* (Chapter 3), which consists of:

  - a camera setup (Section 3.1),
  - an anatomically-based human body model (Section 3.4),
  - a novel constraints detection algorithm (Section 4.3), and
  - a pipeline for converting motion capture data to a format usable in motion editing software (Section 3.5);
- *a novel data-driven animation system* (Chapter 5), aiming in particular at large crowd animations. Its parts are:
  - motion preprocessing (Sections 5.2 and 5.3), editing the source animations and making them periodic,
  - a classification scheme (Section 5.3.3), registering each input animation based on its trajectory,
  - a parametric model (Section 5.5), and related blending scheme (Section 5.6), and
  - a behaviour interface (Section 5.4), connecting the animation system with a higher level planning module.

The second group, **perceptual metrics**, provides a set of metrics and recommendations aimed in particular at data-driven crowd simulations. It is separated into four sections, each describing a set of experiments, their setup, evaluation and results:

- The *human locomotion comparison metric* (Section 6.1) provides insights into the way people differentiate between locomotions and identifies which features provide the largest amount of information to do so,
- the *crowd motion variety* experiment (Section 6.2) determines the minimal number of different characteristic animations required for a group of walking characters to appear varied,
- the *footskating perception experiments* (Section 6.3) determine the saliency of this common artifact and evaluate ways to address it in a perceptually consistent manner, and
- the *human animation timewarping* experiment (Section 6.4) shows that the effects of timewarping, one of the simplest and most common methods for animation editing, are not symmetrically distributed around zero.

While all these experiments are aimed to be used to tune our crowd animation system, their setup is independent of the implementation, making the result generalisable to any crowd / animation system.

The last group of contributions contains **implementation outputs** – code base, programs and data created during the course of this research:

- the *motion editing library*, which is the foundation of all programs, demos and experiment frameworks described in this thesis. It includes a large collection of algorithms, data structures and data importers/exporters,
- the *experimental frameworks*, each described in its respective section of Chapter 6,
- the *data converter*, a GUI application capable of converting data between our motion capture system, our modelling software and our runtime framework,
- the *motion editor*, a command-line application allowing advanced manipulations of motion data,
- the *Biodancer Installation* (Figure 1.2), an interactive installation presented in the Science Gallery during the Biorhythm exhibition,
- the *Metropolis Animation System* (Figure 1.1), a crowd animation module integrated into a full crowd simulation system, and
- the *Natural Movers Project*, of which the primary output is a consistent database of motions, containing 83 different actors (45M and 38F), 108 motions per each. The group of actors covers a varied sample of human subjects, with ages ranging from 14 to 50, weights from 41 to 102kg

and heights from 1.53 to 1.96m.

1.5. Summary of Chapters

The remainder of this thesis is structured as follows:

- **Chapter 2** provides an overview of the previous work related to the topic of this thesis.
- **Chapter 3** describes the process of motion capture, aimed in particular at the passive optical motion capture pipeline. We provide details of the camera and capturing volume setup, calibration, the developed marker setup and connected human model, and a novel method for constraint detection.
- **Chapter 4** deals with description of the most important constraint type required for locomotion synthesis, i.e., the footsteps. We analyse several previous approaches and suggest a new method, applicable to both skeletal and marker data.
- **Chapter 5** provides details of our crowd animation module. It includes the description of:
  - the motion map concept, used for determining the properties of the input motions,
  - data preprocessing step,
  - the parametric blending scheme,
  - linearised skinning method based on the properties of the blending scheme, and
  - details of the animation module integration into a full crowd simulation system.
- **Chapter 6** summarises the perceptual experiments we performed in relation to the topic of this thesis, including
  - a perceptually-based human locomotion metric, allowing to compare the characteristics of human locomotion (Section 6.1),
  - the requirements of animation variety for a crowd scene (Section 6.2),
  - the impact of footskating and footstep cleanup (the most common locomotion artifact; Section 6.3), and
  - the impact of timing changes on the levels of a motion being perceived as realistic (Section 6.4).
- Finally, we conclude in **Chapter 7** with an overview of our results, further discussion and plans for future development.

2

Background and Related Work

The area of human character animation is an important part of computer graphics, and as such it received a lot of attention from the start. The corpus of previous work is therefore very large and its full account is beyond the scope of this thesis. The following chapter contains a description of previous work directly influencing the development of ideas contained in this thesis.

A brief overview of the content of each section and its relation to the rest of the thesis can be listed as follows:

- Motion data representation plays a crucial role in storage, data handling and interpolation. In Section 2.1, we provide a brief overview of common representations, with a focus on joint-based (skeletal) methods. Throughout the thesis, we use mainly the matrix form (see Section 2.1.2), but other techniques are used for specific tasks (e.g., rotation interpolation using quaternions, axis-dependent joints using Euler angles).
- The final goal is to use skeletal data to drive a human character represented as a polygonal mesh using a process called "skinning" (see Section 2.2). This topic is also closely related to our linear mesh deformation technique to speed-up the animation generation in a parametric space (see Section 5.6).
- Motion editing techniques are necessary for both altering captured motion and creating data-driven animation models (Section 2.3). They are used in both contexts throughout this thesis (Section 4.4 and Chapter 5). As constraint-based methods are part of the motion editing field, an overview of constraints detection (closely related to Chapter 4) is provided as well.
- Motion capture serves as our main source of data. A brief summary of motion capture technologies and motion data storage is provided in Section 2.4.
- Our parametric space, described in Chapter 5, uses a data-driven parametric motion synthesis model. A summary of previous work on data-driven motion synthesis can therefore be found in Section 2.5.
- One of the main outcomes of our work is the proposal of several perceptually-based metrics

to assess motion "naturalness" and for clip comparison. A discussion of several metrics from previous work can be found in Section 2.6, followed by a brief summary of related perceptual research (Section 2.7).

2.1. Motion Data Representation

The question of motion data representation is fundamental to data-driven character animation, and as a topic it is still an area of active research.

With the exception of fully parametric representations, each method consists of a description of **keyframes** (explicit datapoints with their deformation description) and an **interpolation method** (a way to extend the keyframes' description to include other times). During motion synthesis, the animation system combines these to create a particular pose corresponding to the **animation time** parameter.

While other representations exist, the most common way of describing a model in computer graphics is using a polygonal mesh. This is essentially a piecewise-linear representation of a manifold in space. An animation of such a mesh is a function describing the position of each vertex in time. Although an explicit representation of this function can be used directly, for character animation we can exploit the fact that the human body contains an almost-rigid underlying structure – a skeleton.

This section provides an overview of methods used to describe this skeleton as a method for motion data representation. As the main focus of this thesis is on data-driven animation models, animation representation is an important issue. For the most part, our data are represented in a hierarchical format and we use matrix algebra to perform operations on them (see Appendix A). However, the storage model for our data uses quaternions to represent orientations and their algebra to perform accurate orientation interpolation. Moreover, most constraints handling is performed after the source data are converted into a world-space (Euclidean) representation, as we need to know the exact world location of each end-effector (see Section 4.3).

2.1.1. Animation Skeleton

The most natural way of describing a human skeleton is as **a hierarchical structure of rigid bodies** (bones). Any movement of a bone in the hierarchy propagates to all its children bones (e.g., the shoulder influences all bones in the arm), which is a desirable property for mimicking the real skeleton. However, this behaviour also amplifies any errors introduced at the lower levels (Arikan, 2006), thus causing error accumulation at the end nodes (called end-effectors; usually feet and hands).

Mathematically, the skeletal structure is represented as a directed graph (tree) of rigid body transformations. The root of this graph, usually located in the pelvis (close to the centre of mass of the character), is described as a full 6 degrees-of-freedom (DOF) transformation. It determines the overall position and orientation of the character. All remaining joints are then described using 3 DOF rotational transformations related to their parent (see Appendix A for more details).

The actual structure of joints used to create a human model is determined by the human skeleton, but certain aspects (like spine links, position of the root in the hierarchy, details in the hands) can be application-specific, as some applications benefit from a more simplified model than others. A common standard for skeletal topology and naming would simplify the reusability of motion data – a desirable property especially for public motion databases. The H-Anim ISO standard ("Web 3D Consortium", 2005) aims to achieve this objective by providing standardised topology, naming, rest-poses and types of joint connections in the human body.

Non-hierarchical (world-space) representations describe bones in Euclidean space relative to

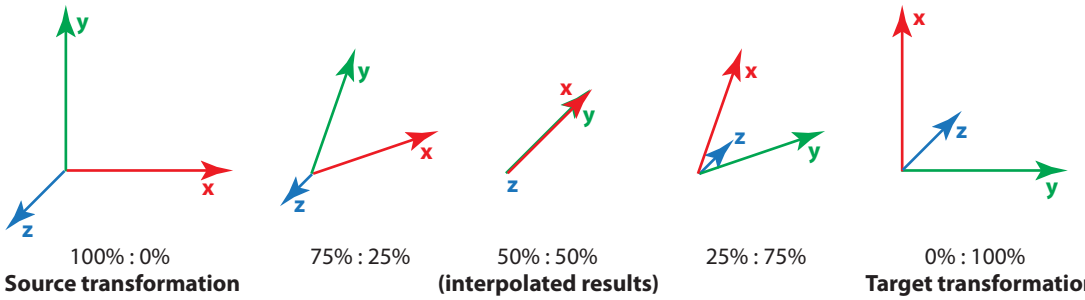

Figure 2.1.: Linear interpolation of transformation matrices (depicted as three axes of the local coordinate system) showing the violation of orthonormality principle during interpolation.

a single "world" origin. This representation lacks an implicit propagation of transformations, but it is useful for several purposes – it forms a basis for most of skinning methods (Section 2.2), and constraint-based and simulation methods (such as Inverse Kinematics (IK); see Section 2.3.3). Furthermore, by avoiding the error accumulation of hierarchical methods, it allows for better compression (Arikan, 2006; Troje, 2002). The mathematical representations of non-hierarchical skeletons are the same as for the hierarchical case, except every bone has to have both its position and orientation represented explicitly (i.e., storing 6 DOF per bone).

Non-hierarchical methods are often used only temporarily for performing certain types of operations (e.g., IK), reverting back to the hierarchical representation after this processing is finished. Nevertheless, several methods use them as the main representation – notably the pointlight walker framework of Troje (2002) and motion compression methods of Arikan (2006).

2.1.2. Bone Representation Overview

The rotational 3 DOF transformations are mathematically described as the SO^3 group with composition operation. This group has several properties that make its representation problematic – it is inherently non-linear, non-commutative and finite. This section describes previous work related to different representations of these transformations.

Rigid-body transformation matrices are the most common way of representing skeletal transformations, and as such they are used extensively in this thesis. The underlying algebra allows simple composition (using matrix multiplication), correct handling of both translations and rotations in a uniform manner, and conversion between coordinate frames. General (non-rigid) matrices can also represent projective transformations, skew and scaling, making them the most versatile representation and the method of choice for modern graphics hardware.

The main disadvantage of this representation is the lack of a simple interpolation method. A linear combination of matrices is usable for certain specialised tasks, such as linear skinning (Lewis et al., 2000), but even if both input matrices represented rigid body transformations (or rotations), the resulting matrix will generally not. A general matrix can be converted to a rigid-body transformation using singular value decomposition (SVD) or polar decomposition (Shoemake & Duff, 1992), but both are computationally expensive, might introduce numerical errors and the result is not defined for all input matrices. Another disadvantage is their inherent data redundancy – 6 effective DOF are represented using 16 numbers (or 3 DOF using 9 numbers for rotational matrices).

Each rigid body transformation matrix can be separated into a world position and 3 *axes of a local coordinate system*. One of these axes can be always determined from the other two using the vector crossproduct. Furthermore, with one of the two axes represented fully, the other one contains only

one further degree of freedom. This reduces storage requirements to 7 real numbers, with some of them requiring only limited accuracy to provide a description with relatively small error.

While not providing any advantage over transformation matrices for motion interpolation, this representation can be used for motion compression. Arikan (2006) uses such representation as a first step of a motion compression algorithm (with each rigid body transformation described as 3 points in space, thereby providing more accuracy and better compression parameters).

Euler angles decompose an orientation into three (or less) rotations around predefined axes. This representation is simple and intuitive, and allows rotations to be dealt with in a linear manner, thus allowing even complex linear manipulations such as principal and independent component analysis (Shapiro et al., 2006) and frequency-domain analysis (Unuma et al., 1995).

Unfortunately, the linear combinations of angular data do not always correspond to expected geometric results. Each orientation can be represented in an infinite number of ways and therefore provide different interpolation results when used naively (each axis angle is periodic). Moreover, the results are dependent on the order of applied rotations and certain combinations of angles lead to gimbal lock (a loss of one degree of freedom).

Unit quaternions are a 3D extension of the ability of unit complex numbers to describe 2D rotations (Shoemake, 1985). They are more effective than transformation matrices in representing orientation (4 real numbers) and their algebra provides a simple and geometrically-correct way of combining (multiplication), inverting (conjugate) and interpolating (spherical linear interpolation) orientations. These properties explain why this representation is popular in data-driven character animation, as they provide an intuitive way of performing all common operations on motion data.

Unfortunately, unit quaternions do not provide an exact way of computing a combination of more than two orientations or their mean. The can only be used to compute their approximate values, i.e., a normalised linear combination (Shoemake, 1985) and exponential maps mean (see below), respectively. Another issue with quaternions is their bipodality – each orientation can be represented in two different ways. This can be explained by their relationship with axis-angle representation, where both an axis/angle combination and its negative value represent the same orientation (see Shoemake, 1985 for a more detailed explanation). While in most cases bipodality can be handled properly, problems can arise when working with several orientations simultaneously (Park et al., 2002).

Exponential maps attempt to linearise the rotation algebra SO^3 by mapping it onto a linear subspace R^3 around a particular rotation (often zero). This operation projects the surface of a 4D sphere onto a section of a 3D hyperplane, thereby successfully capturing most of its local properties, but providing a poor approximation for non-local ones. For example, this can cause singularities around the edges of the projection, with decreasing accuracy based on distance from the projection centre(Grassia, 1998). However, it allows the use of an intuitive and commutative algebra on orientations, including their simple interpolation, integration and differentiation. The conversion to exponential maps can be performed on both quaternions (Grassia, 1998) and transformation matrices (Alexa, 2002).

The centre of mapping is a crucial parameter for the accuracy of exponential maps. For interpolation purposes, the ideal mapping point would be the weighted mean of its operands (which is also the interpolation solution). However, no such operation is explicitly defined on rotational algebras. Park et al. (2002) propose to use an optimisation step to determine the *best average* value, which is also the main idea behind Alexa's linear rotational algebra (2002). However, performing a non-linear optimisation step for every interpolation of every character's joint is computationally expensive. Park et al. (2004) propose to avoid it by using the interpolation result of the previous frame, assuming that the consecutive frames are close enough to provide sufficient accuracy. Forbes & Fiume (2005) go

even further in this direction, by computing a single projection centre for each joint, and then using it throughout the animation.

In his lecture notes, Lee (2008) compares quaternions and exponential maps to points and vectors; the former expressing the absolute value and the latter its relative displacement. Indeed, this method is used quite often. For example, constraint-based motion editing uses a displacement function, e.g., the hierarchical B-splines approach by Lee & Shin (1999), while motion statistical analysis requires a linear and commutative algebra for displacements related to an average value (Lim & Thalmann, 2002; Tournier et al., 2009; Forbes & Fiume, 2005).

The last representation in our list – **dual quaternions** – expands the 4-dimensional quaternion algebra to 8 dimensions by including the translational information and redefining the related operations (Kavan et al., 2007a). Dual quaternions can be represented either as a pair of quaternions, or as quaternions constructed of dual numbers. Unfortunately, as it is based on quaternion algebra, this representation suffers from many of the same disadvantages as unit quaternions. Moreover, for a classical hierarchical skeleton, only 3 degrees of freedom are required for most joints (with the exception of the root), making the added translational information redundant. However, dual quaternions have a practical use as a non-linear and rotationally correct alternative to standard linear skinning (Kavan et al., 2007a; see Section 2.2.2).

A group of **physically-based methods** reflect the physical structure of the human body on a lower level of abstraction than the methods mentioned above, often creating a representation of bones, muscles and other tissues. The motion is then described at the level of joint torques or muscle activations, with physically-based simulation and constrained controllers guiding the overall behaviour. However, even though often inspired by (or trained on) captured data, these techniques are not inherently data-driven, which puts them outside the scope of this thesis. For a detailed list of methods and their descriptions, please refer to Geijtenbeek et al. (2011) and van Welbergen et al. (2010).

2.2. Character Skinning

Character skinning involves a set of methods that deform a character mesh according to a skeletal animation. A broader concept of character skinning can be generally divided into three groups:

- generic mesh deformation, i.e., methods that pose the mesh based on a set of constraints,
- skinning, which is actually a specific subset of the previous group, which uses a set of rigid-body transformations as constraints to describe desired mesh deformations,
- image-based methods, which avoid real-time deformation calculations by precomputing an image-based proxy representation.

In this thesis, we use the simple linear skinning technique for most of our animated characters. An exception is the linear blending scheme of the parametric space (see Section 5.6), which uses an interpolation method at the level of skinning matrices instead of using a hierarchical skeleton representation. Mathematically, however, it closely resembles the example-based skinning methods (for more information please refer to Section 2.2.2). The following Sections introduce the related work in this field.

2.2.1. Generic Mesh Deformation

Generic mesh deformation methods use a set of constraints to describe a desired pose of a mesh. The constraint types can be quite diverse; the most common type specifies the position of a given subset of vertices (parameterised by time) or a spatial region where a subset of vertices is supposed to remain (e.g., above the ground). The deformation method ensures that the final mesh pose is plausible while

satisfying the input constraints. Character skinning represents a specific subset of these methods, and uses a set of rigid body transformations as constraints (see Section 2.2.2). The field of generic mesh deformation is vast and only loosely related to this thesis (see the cloth simulation described in Section 5.6). For this reason, in the next paragraph we provide only several references for further study.

A natural way of dividing different generic mesh deformation methods is according to the dimensionality of their control objects. Gain & Bechmann (2008) provide a survey using this approach, divided into four main sections – points (0D), curves (1D), surfaces (2D) and volumes (3D). Apart from the large group of generic methods, several subgroups received particular attention from the research community due to their suitability for a particular purpose. Laplacian mesh editing methods (Sorkine, 2005) allow the shape of a mesh to be controlled on both local and global levels, while preserving the surface details (or deforming them appropriately). Garment deformation methods (e.g., English & Bridson, 2008) behave in the opposite manner, creating or destroying local details (wrinkles) in a physically-consistent manner to compensate for overall shape change.

2.2.2. Skinning

Skinning represents a very specific group of mesh deformation methods, where the target pose is described using a skeleton. However, as most skinning methods expect their input to be defined in world space, the hierarchical skeleton is usually converted into a non-hierarchical representation relative to the world origin (see Appendix A).

The relationship of skinning methods to this thesis is twofold – first, all the characters animated using our parametric space are skinned using a linear skinning approach; and second, we introduce a novel skinning technique that exploits the properties of our animation model (see Section 5.6).

Linear blend skinning (LBS), also called skeleton subspace deformation (SSD), uses a linear combination of matrices that describe the transformation of each bone with respect to its original, or binding, pose to deform a mesh (see Appendix A for a mathematical explanation).

While several more advanced and flexible versions of this method were proposed (see below), the original simple method is currently the industry standard and the most popular method for skeleton-based mesh deformation. A detailed description together with a list of properties and drawbacks is provided by Lewis et al. (2000). The skinning data consists of a set of weights assigned to each vertex, which connect the vertex with a number of bones (often limited to 4). Each of these weights is non-negative and their sum equals one. They form a set of weights to compute a weighted average of the bone transformations, which is then used to determine the deformed position of the vertex. While very simple and fast to compute, this method has two main drawbacks: the loss of volume caused by linear combination of matrices (see Figure 2.2) and its inability to represent axis-dependent transformations (e.g., the human wrist).

Several techniques were developed to address these issues. Apart from non-linear skinning methods, which usually lead to significant increase in computational cost (see below), **linear extensions to LBS** add either more joints or more weights per vertex. The first approach was used by Mohr & Gleicher (2003), who suggested creating new joints as parameterised non-linear interpolations of existing joint transformations (e.g., using quaternions). Using the second approach, Merry et al. (2006), proposed using 4 weights per vertex and bone (one for each component of the vertex position), while Wang & Phillips (2002) used 12 weights, one for each non-trivial element of the transformation matrix. However, the additional parameters in both of these methods do not have an intuitive meaning and therefore they are not practical for manual input. For this reason, in all three cases a method

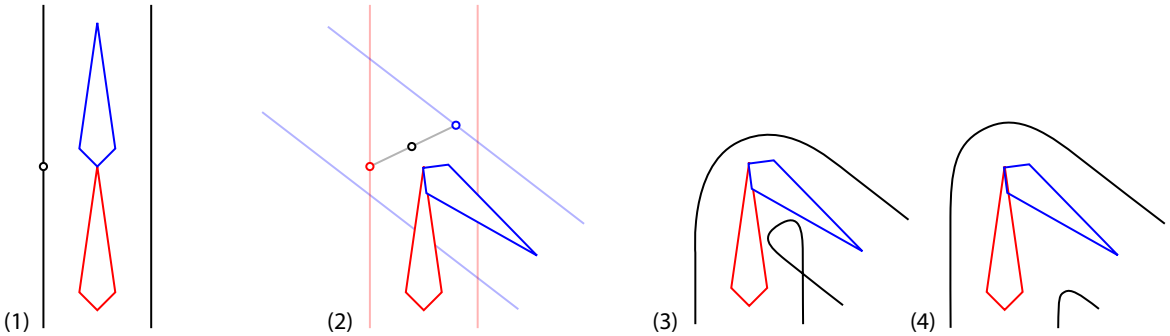

Figure 2.2.: The illustration of the artifacts of simple linear skinning. A vertex skinned with weight 0.5 to both displayed bones (1) is deformed by bone movement, with its new position as a linear combination of the deformation by the two bones (2). This can lead to loss of volume and self-intersections (3). However, a more natural solution (4) would require non-linear deformation model.

is provided to derive them automatically from an example mesh animation (with connected skeletal data).

Given a mesh animation as a source of data, skeletal animation can be described as a problem of **mesh animation compression**. From this point of view, skinning methods are decompression mechanisms with varying abilities to represent the original data. In a classical computer animation pipeline, this compression is essentially performed manually by an artist. However, a large group of methods provide automatic ways to do this. For example, Kavan et al. (2007b; 2010) provide methods based on non-linear optimisation that aim to create a non-hierarchical skeletal structure and a set of localised skinning weights that represent an animation as closely as possible. While eliminating the need to specify extra parameters manually, these algorithms are at risk of over-fitting source data, making the compression relevant only for the animation provided as input.

Non-linear skinning methods provide alternative ways of deforming a mesh based on skeletal animation data. Kavan et al. (2007a) introduce a skinning method based on transformation interpolation using the non-linear algebra of dual quaternions, thereby successfully addressing the joint-deflation problem of linear methods, without adding any additional parameters. Furthermore, they provide a way to convert this representation to simple linear skinning, at the cost of adding additional interpolation joints into the skeletal hierarchy (Kavan et al., 2009). Physically-based skinning using a finite element solver was presented by McAdams et al. (2011). Simulating the bones and attached soft tissue with collision detection, this technique provides very accurate and physically plausible results. However, as a non-linear solver, it is computationally very expensive and far from being interactive.

The last group of skinning methods – **example-based skinning** – use a set of example poses and an interpolation scheme to synthesise even very complex character deformation at a relatively small computational cost. However, to achieve plausible results, the space of deformation parameters has to be sampled appropriately, but determining the positions of new examples is not a trivial task and their creation poses a significant work overhead. Moreover, the storage costs of a large number of examples can be high.

Lewis et al. (2000) introduce the *Pose Space*, an example-based skinning method using a radial basis interpolation function. They also provide a detailed comparison with traditional linear skinning. Sloan et al. (2001) extend this approach by incorporating a cardinal basis function (a combination of linear and radial basis functions). They provide an interactive tool for building the pose space and introduce the use of pseudo-examples to reparameterise the pose space. However, both methods

suffer from problems with their interpolation schemes. Radial-basis function parameters are non-local, in that each point in the parametric space has non-zero weights for all examples. Furthermore, as parameters tend towards infinity in any direction, the result converges asymptotically towards the mean of all examples. Cardinal basis function solves this problem, at the cost of introducing negative weights with no geometric meaning.

2.2.3. Image-based Techniques

The use of polygonal models can be a limiting factor for systems with a large number of characters displayed simultaneously on the screen. Level-of-detail techniques aim to address this issue, but the models cannot be simplified any further than a particular level without severely impacting on visual quality. Image-based techniques aim to address this problem by representing the geometrical details using textures instead of the geometry, with a particular texture selected (and optionally deformed) using the skeletal pose information. They are especially effective in extreme scenarios, where the number of polygons exceeds the number of displayed pixels.

Dobbyn et al. (2005) presented a system built on impostors (or billboards), where a character is represented by a single polygon with a texture applied. By connecting the impostor generation system with the on-line renderer, the rendering style was matched exactly, creating seamless transitions between the two representation. Unfortunately, animated models require a very large number of stored animation poses. To address this, Kavan et al. (2008) proposed polypostors, which are impostors built of several textured polygons. By displacing the vertices of these polygons, it is possible to deform the displayed models and mimic their appearance after deformation without the need to explicitly store every animation frame.

2.3. Motion Editing

All data-driven animation techniques incorporate a database of motion clips. In order to be able to do more than just replay the pre-recorded motions, we need methods to alter this data in ways suitable to its representation and properties.

The large field of motion editing methods can be separated into four groups – methods dealing with motion as a multi-dimensional signal (Section 2.3.1), blending and interpolation methods (Section 2.3.2), constraint-based techniques (Section 2.3.3) and physically-based techniques (Section 2.3.4).

The motion editing field is related to this thesis in several ways. We deal with motion as a signal, when we filter it and blend in motion changes (Section 4.3). Moreover, the motion map method (Section 5.2) uses signal processing methods quite extensively. Motion blending underpins all data-driven animation approaches (including our locomotion system; see Sections 5.5 and 5.6), and our footstep handling (Section 4.4) is just a variation on the theme of constraint-based motion editing methods. Therefore, we focus in this section primarily on these aspects; for a more detailed overview, please refer to a more comprehensive survey by van Welbergen et al. (2010).

2.3.1. Motion as a Signal

A motion clip can be represented in a simple manner using a set of joint-angle functions parameterised by time, which form a multi-dimensional signal. This data can then be altered using signal processing techniques, such as blending, concatenation, multiresolution editing, timewarping or displacement mapping (Bruderlin & Williams, 1995).

Adding a smooth displacement function to the motion data allows certain properties of a motion clip to be changed while preserving its details. Witkin & Popovic (1995) use a cardinal spline to create a displacement function that changes the input motion to satisfy a set of user-defined constraints, using

IK, changes in joint angle or timing edits. The placement of the knots in their spline is based entirely on the input constraints and can therefore be quite uneven, leading to overshooting in certain regions while others are oversmoothed. Lee & Shin (1999) address this issue by implementing a multiresolution approach, with each level using only splines with evenly spaced knots. Starting from the coarsest level, each spline is fitted into the difference residual, thereby adding more details without compromising the overall curve shape. Furthermore, their spline is defined using quaternions and exponential maps, which provide a better interpolation scheme than the joint angles used by Witkin & Popovic.

Motion style describes subtle or stylistic differences between motion clips of the same type. Amaya et al. (1996) show on non-locomotion motion clips (e.g., drinking, kicking) how simple amplitude and speed changes affect the emotional context of the motion. Unuma et al. (1995) provide a method, based on frequency-domain analysis, which is capable of describing, interpolating and altering more generic styles of periodic motions. A dynamic filter can be used to emphasise a set of primary movements while minimising the influence of others (Wang et al., 2006). Although this method is particularly suitable for cartoon animation, Wang et al. also demonstrated its applicability to motion capture data.Hsu et al. (2005) use a non-linear optimisation technique to find spatio-temporal correlations between two motion clips (both periodic and non-periodic) and then construct a time-invariant representation of the motion style difference.

The motion texturing approach (Pullen & Bregler, 2002) starts with a user-defined rough animation of a small subset of a character's joints. By splitting the source animation into segments, based on the 2nd derivation zero crossing, and matching them with segments from a motion database, this approach creates a new animation by providing more details into the animated joints and synthesising the motion for the free joints.

2.3.2. Blending and Interpolation

Motion clips can also be edited by blending (interpolating) with a database of pre-recorded motion primitives. To do so, it is necessary to first establish both spatial and temporal correspondences and, based on this information, blend the clips in a way that will reduce the possibility of introducing motion artifacts.

The usual approach to **motion interpolation** blends skeleton poses (frames) by interpolating each joint of a hierarchical representation separately using an algebra of rigid body transformations (see Section 2.1). However, more advanced interpolation methods can avoid certain artifacts caused by this simple method, such as footsliding. Rose et al. (1996) allow the disassembly of the original motion into bodyparts and time fragments, based on manually-defined spacetime constraints. The blending can then by performed on each bodypart separately, with different parameters and timings, and even assembled into more complex motions using IK to enforce blended constraints. Another advanced interpolation scheme was introduced by Mukai & Kuriyama (2005). Their method first aligns the data in both the spatial and temporal domains and then creates blends using a statistical kernel-based scheme built on an artifact minimising predictor.

Hierarchical blends of poses create a highly non-linear algebra with respect to end-effector positions, which is a problem particularly for data-driven inverse kinematics. For this reason, IK blending schemes must attempt to linearise their parametric space using resampling (see Section 2.3.3).

Several methods convert the motion data to an alternative description, which allows them to perform the blending and interpolation directly in the new domain. Examples of such descriptions are parameters of principal components in Euclidean space (Troje, 2002) in geodesic space (Tournier et al., 2009; Lim & Thalmann, 2002), or in the frequency domain (Unuma et al., 1995).

The problem of **temporal alignment** for motion blending was first stressed by Rose et al. (1998) in their Verbs and Adverbs work, showing how it can help to minimise blending artifacts in motion parameterisation methods. They use manually annotated events and linear timewarping, which stretches and contracts the signal intervals to match the corresponding events in the time domain, to ensure that the blending is performed only on similar frames only. However, the explicit definition of discrete constraints can create conflicts and violations of maximal timewarping limits. Therefore, Ménardais et al. (2004) introduced an algebra capable of handling priorities, conflicting constraints and incompatible constraint sequences.

Automatic temporal alignment can be achieved using Dynamic Timewarping (DTW), a method commonly used to match two sequences of events in signal processing (Bruderlin & Williams, 1995). Unfortunately, for a set of animations that are not sufficiently similar, DTW can introduce timewarping artifacts, such as non-causality (incorrect event sequence match/order) or excessive time manipulation. Kovar & Gleicher (2003) address these issues by applying an alternative sequence matching algorithm in both space and time, limiting the maximal slope and final function shape. In contrast to the DTW algorithm, this leads to an invertible function, making the result usable as a reversible motion alignment transformation (Mukai & Kuriyama, 2005).

Hsu et al. (2005) use a spatio-temporal optimisation technique to determine correspondences between two motions in space and time in a single step, thereby allowing a closer motion match with fewer artifacts. Based on this information, they build a time-invariant difference data structure usable for motion style transfer.

2.3.3. Constraint-based Motion Editing

A constraint is an explicitly defined spatial or temporal feature that represents a desired state of the motion. This feature can be defined manually, or extracted from the motion (e.g., footsteps). A constraint based editing method changes the original motion to satisfy a set of constraints. With this broad definition, most motion editing techniques can be described as constraint based methods, as they alter the motion to give it a desired property. However, the term usually describes methods that deal explicitly with spatial features.

Constraint based methods are closely related to several parts of this thesis. Chapter 4 is primarily concerned with footstep constraints detection and enforcement. Our linearised parametric space (Chapter 5) is aimed at minimising walking motion artifacts defined using constraints. Finally, our perceptual study about footskating evaluates the impact of enforcing (or not) footstep constraints (Section 6.3).

In this Section, we describe only a small subset of constraint based methods that are directly related to this thesis. For a more comprehensive overview, please refer to Gleicher (2001).

Constraints Detection

The first step of any constraint based motion editing method is to determine the spatial and temporal information about the constraints in the original data. In practical applications, these data are often defined manually by an animator (Rose et al., 1996; NaturalMotion, 2011), because a fully generic and accurate method based solely on motion data is yet to be developed.

If a digitised description of the environment is available (or is trivial enough), a simple constraint detection method can compute the distance between the objects in the environment and the articulated model (assuming no significant noise in the input data). This computation performed at every frame can be ineffective, so Bindiganavale & Badler (1998) propose to determine a set of likely candidates using the zero crossing of the motion data second derivative (acceleration). A similar method, proposed

by Hreljac & Marshall (2000), uses local maxima of acceleration data instead. In their implementation, these points are determined using the first derivative of the heel and toe acceleration (jerk), separated into its horizontal and vertical components. They provide an evaluation of their method using force platforms.

When noise is present, higher derivatives of motion data tend to be unreliable. To achieve similar results as higher-order methods, the end-effector to object distance threshold can be combined with a threshold of end-effector velocity magnitude (Lee et al., 2002; Ménardais et al., 2004). To avoid multiple detections caused by noise in the data, Glardon et al. (2006) filter the data in a preprocessing step using a PCA based motion anticipation filter, and incorporates an adaptive thresholding scheme instead of a single fixed value.

Another group of methods builds on the analysis of the world-space end-effector transformation nullspace (i.e., the subspace that does not change under the transformation). Liu & Popović (2002) analyse the translations and provide the information about static or sliding constraints. Salvati et al. (2004) extend this method by determining the full nullspace of transformations and merging it over a window of frames. The dimensionality of the solution determines the constraint type – a 0D constraint is a rotation around a point, a 1D constraint rotates around an axis and a 3D constraint stops the whole space changing, thus providing a static constraint (please note that a 2D solution is not possible). Based on this work, Le Callennec & Boulic (2006) develop a method using only a difference transformation between two frames (only 1D and 3D constraints are detectable). They also develop a prediction filter, making the detection more robust against a noisy input.

If a set of annotated data is available, a machine learning approach towards constraint detection is possible. Ikemoto et al. (2006) uses a k-nearest-neighbour classifier trained on a set of annotated data and the world positions of leg joints in 21 consecutive frames. This classifier outputs a vector of 4 binary flags – left or right, heel or toe. The training is simplified by an interactive training tool.

Inverse Kinematics

Inverse kinematics (IK) is one of the basic tools for constraint based motion editing. For its definition, we should start with defining the opposite term – forward kinematics. Forward kinematics defines the position (and orientation) of the end-effector $\boldsymbol{p}$ as a function $\boldsymbol{f}$ of all joint states in the chain $\boldsymbol{q}$ leading to this end-effector, with the shape of the function $\boldsymbol{f}$ defined by the topology of the skeletal structure. Inverse kinematics solves the inverse problem, i.e., finding a joint configuration that leads to the desired state of the end-effector by creating an inverse function $\boldsymbol{f}^{-1}$. This is a much harder problem than forward kinematics, as the inverse function

- does not have to have a unique solution, when the problem is under-constrained. This results in a subspace of configurations, all leading to a correct solution;
- does not have to have any solution, i.e., when the problem is either over-constrained, or simply outside the space of possible solutions;
- may provide a solution with a large number of degrees of freedom (DOFs), each of them being highly non-linear.

Previous work provides several different ways of addressing this problem. For relatively simple joint chains, an **analytical solution** is possible, which defines the complete subspace of all configurations. The IKAN toolkit Tolani et al. (2000) provides such a solution for anthropomorphic limbs.

For more complex chains, a **numerical solution** is necessary. Unfortunately, numerical methods usually provide only a single solution, and although the solution subspace can have many degrees of freedom, these methods are not capable of fully exploring them. As a consequence, the results can

be very unnatural, especially if the starting pose is far from the goal position (van Welbergen et al., 2010).

The Jacobian inverse method linearises the problem around the current joint configuration (Welman, 1993). This solution ensures minimal changes to the joint rotations, but it is computationally expensive and unstable when close to the Jacobian singularities. The Cyclic Coordinate Descent (CCD) (Wang & Chen, 1991) is both faster and avoids the singularities of the previous method by changing the joint configuration one at a time. However, it can exhibit poor convergence and, by not distributing the transformations along the whole chain, it can lead to unnatural poses (Welman, 1993). To solve the latter problem, Kulpa & Multon (2005) proposed an alternative iteration mechanism.

The solution ambiguity can be reduced using a larger set of constraints, by analytically defining the plausible configurations of the IK chain. Unfortunately, this can lead to an over-constrained problem. Boulic et al. (2003) provide a system that allows constraints to be prioritised, with each priority level restricting the solution space for all lower levels to the subspace of its manifold.

Yet another group of methods based on **optimisation** poses the IK problem as a minimisation problem, which can be handled using a number of standard non-linear solvers. This approach also allows other constraints to be added into the IK computation, such as joint limits and inter-joint dependencies. The main disadvantages are based on the properties of the solver used, with the most restrictive one being the computational complexity.

A **data-driven** approach towards solving complex IK chains limits the many-dimensional solution space to poses that can be represented as a combination of examples from a database, thus providing more natural solutions. To represent the example combination in terms of end-effector position, a non-linear reparameterisation of the configuration space is required. A standard solution to this problem is to use a locally linear approximation of the inverse function by registering a larger set of interpolation results into a space parameterised by the end-effector configuration.

Wiley & Hahn (1997) use linear nearest neighbour interpolation and dense uniform sampling to create this structure. Kovar & Gleicher (2004) extend this approach to the temporal domain, using timewarping to blend different motions, to create a system that can synthesise not only single poses, but also full animations that reach these poses. Rose et al. (2001) use cardinal basis function interpolation to provide smoother parameterisation. This comes at a cost of more complex blending and a risk of unnatural poses, as cardinal basis functions do not guarantee positive weights and each pose is created using a combination of all poses in the parametric structure. Their approach does not sample the parametric space regularly, but rather uses the smoothness of the inverse function to determine the locations for new pseudo-examples, thereby improving the resulting accuracy.

Per-frame Inverse Kinematic + Filtering

The classical definition of inverse kinematics is as a stateless function, not providing any explicit consistency between frames. One possible way to extend this information to other frames is to use filtering to create smooth motion with enforced constraints. This group of techniques is called Per-frame Inverse Kinematics + Filtering (PFIK+F), a term introduced by Gleicher (2001), who also provides a comprehensive list of previous techniques with a discussion of their properties.

Choi et al. (1999) use IK to enforce end-effector configurations in constrained time intervals and filter the poses at the start and end of the interval to ensure smoothness. Many other methods use a similar approach, and utilise smoothing methods from signal processing field (Gleicher, 2001; Kovar et al., 2002b; Boulic et al., 2003). A different filtering approach involves fitting a spline to the constrained data, thus creating a smooth displacement function (Witkin & Popovic, 1995; Lee & Shin,

1999).

Optimisation Methods

Optimisation methods provide more flexibility than the previous techniques, because the solution to multiple constraints and their priorities can all be embedded into their minimisation function and solved in one step. They can also be used to solve IK for the whole animation at once, enforcing solution continuity, threfore providing both smoothing and constraints enforcement. Their disadvantages include very high computational complexity, issues with determining a single minimisation function that includes all the desired properties, and the possibility of unpredictable behaviour (van Welbergen et al., 2010). A detailed comparison between PFIK+F vs. optimisation techniques was provided by Gleicher (2001), with the conclusion favouring the former.

Gleicher (1997) provides an implementation of a spacetime optimisation method that enforces constraints at specified keyframe locations (the constraint enforcement is not guaranteed outside these keyframes). This technique was later extended to incorporate many different constraint types, and provides a comprehensive method for motion retargeting (Gleicher, 1998). A constraint based optimisation technique, capable of synthesising a motion from a set of motion clips, was introduced by Liu & Popović (2002). Their approach involves automatic constraint and transition detection, with the final result generated using a spacetime optimiser that prefers smooth and physically correct (balanced) motions.

Motion Retargeting

Motion retargeting is a specific subset of constraint-based motion editing techniques, that focuses on adapting a motion to a character with a different bone structure and/or bone sizes than the ones from which the motion was captured.

In the case of locomotion, the footstep cleanup (i.e., footstep constraints enforcement) is the most important step when retargeting a motion. Essentially, any constraint based motion editing method can provide a solution, but a targeted method can more successfully address this particular issue. Kovar et al. (2002b) provides such a method, combining successive steps of root displacement, a specific variant of analytic leg inverse kinematic, root trajectory smoothing and leg lengthening. Another specific footstep cleanup method was introduced by Glardon et al. (2006), who use an IK technique with constraint preferences.

To retarget motions of other types, cleaning up the foot motion is not sufficient. Gleicher (1998) introduces a retargeting technique based on spacetime optimisation, which incorporates several types of user-defined constraints, e.g., parameter value range (joint limits), point in a specific position, point constrained to a region (above ground, outside an object), stable point (for footstep cleanup), point-to-point distance (holding an object) or vector orientation (heading direction).

If the source and the target character have a different topology (e.g., different number of joints in a limb), a more generic method is required to adapt the motion in a plausible way. Monzani et al. (2000) proposes such a method that is applicable to humanoid characters. He uses an intermediate skeleton to standardise the motions and then a combination of IK and filtering to satisfy the constraints. An even more generic technique was proposed by Hecker et al. (2008), which was used in a game called Spore. The user has the freedom to design creatures with a wide range of bodyshapes, number of limbs (heads, claws, tails) and other radically differing properties. The animation is manually specified in a morphology independent form, which allows it to be retargeted to each specific case using a particle based inverse kinematics technique.

2.3.4. Physically-based Motion Editing

Methods based on a physical simulation form the most challenging group of motion editing techniques. The main reason is the complicated nature of the human body, together with the even more impressive control abilities of the central nervous system. A large number of methods that simulate both aspects on a different levels of complexity have been developed in recent years, though a complete list is outside the scope of this thesis. For a detailed overview of physically-based simulation and control methods, please refer to the Eurographics STAR report by van Welbergen et al. (2010).

Several methods combine a data-driven approach with physical simulation. Popović & Witkin (1999) start their motion editing technique with a captured motion clip, fit a physical character model to the motion, alter the model's parameters to generate a new motion and map the difference onto the original sequence. This approach retains the subtle properties of the original animation (which cannot be fully represented in the physical model), while providing physically based control of the overall motion. Their approach uses a simplified human model, created manually to reflect the intended motion editing task.

The constraint-based spacetime optimisation technique of Liu & Popović (2002) incorporates a physical simulation as a part of the minimisation function, specifically a momentum control and a balance predictor. This allows the full scheme to provide more physically correct results by taking into account the dynamic properties of the motion.

2.4. Capturing and Storing Motion Data

Motion capture technology provides a means to directly record an actor's motion. For this reason, it is valuable for all data-driven animation methods as a source of extremely realistic motion data. In this thesis, we use an optical motion capture system as the source of our data. However, as it is a commercial system, the software and internal procedures are mostly proprietary. For this reason, we use it only as a tool (see Chapter 3) and do not discuss its technical details.

The second part of this section deals with motion data storage. Even though there has been a significant amount of work done on motion compression, in our case it is not an issue (see Section 2.4.2) and therefore we provide only a very brief overview of previous work in this area.

2.4.1. Motion Capture

Motion Capture covers a very large set of methods capable of tracking human or animal motion in space. Its applications include surveillance and identification (biometrics), virtual/augmented reality, computer interaction using gestures and movements, medical/sport applications and accurate 3D motion recording. Methods can be classified as facial motion capture, bodypart motion capture (gestures, head movement), full body motion capture or high-resolution techniques. The latter are capable of capturing garment animation and subtle muscle motions (usually referred to as performance capture).

This section briefly describes the different technologies used in this vast field. Their large diversity is characterised by different requirements on the captured motions and the range of sensors capable of recording a particular type of motion. We will describe only the most common technologies, which produce data accurate enough for data-driven character animation. For a more complete list, please refer to Moeslund et al. (2006).

Optical motion capture based on simultaneous multiple viewpoints is most commonly used in practical applications due to its stability and robustness as well as minimal invasiveness of the equipment. *Markers*, which are the physical representation of tracked features (e.g., active

markers that emit light pulses, or back-reflecting passive markers in combination with light sources placed close to the camera positions), have optical properties that are easily recognisable in the each camera's output. Compared to markerless optical motion capture, this allows more robust tracking and consequently less noise in the resulting data. The easy identification of a marker in the camera output allows for simpler data processing and real-time usage. The main disadvantage is the requirement to place markers on each captured feature, which limits the capabilities for capturing high resolution and garment movement.

Extending the cameras with panning and tilting capabilities can lead to better coverage of the capture region with a lower number of cameras at the cost of a more complex camera calibration procedure (Kurihara et al., 2002).

Markerless motion capture methods identify tracking features automatically, either based on correspondences between multiple viewpoints, or by analysing 3D representations (optical flow or voxel based) extracted from the captured data. Because physical markers are not used, this approach is more suited to capturing garments and for performance capture. The main disadvantages include a higher noise ratio in the resulting data (caused by inaccurate detection and triangulation of features), the requirement for either a non-uniform texture (e.g., clothes with patterns that can be used for feature detection), or in the case of silhouette based methods, an accurate way to extract the moving figure's silhouette from video data.

Other motion capture technologies are not as common sources for data-driven motion synthesis as the ones mentioned previously. The optical motion capture field includes not only multiview approaches, but also a large group of single view and stereo vision methods. Applications range from surveillance to low cost motion capture used for advanced user interfaces in virtual environments (Moeslund et al., 2006). Another group of optical motion capture uses structured light to allow depth reconstruction, providing a 2.5D representation of the captured volume in the real time (e.g., Microsoft Kinect). However, the consumer level products of this kind do not provide the same level of accuracy as previously mentioned systems, while professional level systems are not broadly used due to their cost.

Non-optical technologies can utilise a broad range of electromagnetic, inertial, mechanical and ultrasound location sensors, but only electromagnetic and inertial sensors are used in practice. Electromagnetic motion capture (Bodenheimer et al., 1997; O'Brien et al., 2000) has limited range but directly yields the transformations of body segments which, together with with very low noise, makes it ideal for on-line use (Herda et al., 2000). Inertial sensors provide only acceleration data (2nd derivative of position with respect to time) which makes them unsuitable for full-body motion capture, but their low price and simple setup makes them well suited for driving on-line performance capture (Slyper & Hodgins, 2008).

2.4.2. Storing Motion Data

Motion data, as any other kind of data, need to be stored and retrieved efficiently. For large databases, the total size of this data can also pose a problem. In this section, we discuss different aspects of storing motion data that are directly related to this thesis.

The storage requirements for uncompressed mesh animations pose a significant problem – assuming a relatively standard number of 10000 vertices in a mesh model, and an animation of one minute length, recorded with 30 frames per second and stored using IEEE floating point number representation, the space required to store this animation is approximately 206 MB.

Skeletal animations provide significant data compression – for the same animation, described by a

relatively standard number of 22 bones (a skeleton without detailed hands; described by 6 numbers for the root and 3 for the rest of the joints), the required storage space is only 0.5 MB. With the usual amount of hard drive space in modern computers, this becomes an issue only for very large motion databases.

Many of the classical data compression methods can be applied to skeletal motion data. Apart from generic lossless techniques, the lossy algorithms would include Principal Component Analysis (PCA), wavelets, motion JPEG (a combination of discrete cosine transformation and entropy encoding) and subsampling. Arikan (2006) provides a detailed analysis of the properties of each of these methods and describes a new technique that combines them, which has better properties than any of its individual components. Faloutsos et al. (2007) provide an even more comprehensive list of compression methods applicable to motion data.

In all the implementation work in this thesis, we store our data uncompressed, in a binary format (created with Boost::serialization library), using quaternions for rotational representation, 3D vectors for translations and IEEE floating point numbers for their components. This allows to reduce the whole database of 83 subjects, 100+ motions for each subject, to less than 1 GB of motion data. Again, compared to the usual capabilities of modern computers, this is a very modest requirement. Therefore, the motion compression problem is outside the scope of this thesis.

However, motion file formats pose an important problem for our work – depending on the project, our pipeline consists of several commercial programs used for data processing (Vicon IQ, Autodesk 3DS MAX, NaturalMotion Morpheme, Autodesk Maya, Autodesk MotionBuilder...) and several 3rd party run-time libraries used for rendering and animation. Unfortunately, due to the lack of one generally agreed and supported file format, these programs and libraries do not cooperate well. In our work, we have implemented a set of importers and exporters for several of common formats. For their description and a brief overview of our pipeline, please refer to Section 3.5.

2.5. Data-driven Animation Methods

Data-driven animation methods use a database of existing motion data to synthesise novel animations. This synthesis is based on blending, concatenating and timewarping of the original data, or using the original data as a starting point for an optimisation process (possibly based on a physical simulation). Compared to motion editing techniques (Section 2.3), data-driven animation methods provide more comprehensive ways of generating novel motions, and often incorporate motion editing as part of the generation process.

The data-driven animation models, based on a single character but fast enough to generate motion for a crowd, are the main topic of this thesis. Specifically, our parametric space concept (see Chapter 5) is a data-driven parametric model, and the Biodancer project was based on a beat-synchronised fragment based technique.

The list of publications in this section is aimed at issues directly addressed in our work. For a more complete overview, please refer to the Eurographics state-of-the-art report by van Welbergen et al. (2010).

2.5.1. Fragment Based Methods

Fragment based methods (or graph approaches) concatenate short fragments of motions into a stream of motion data. More advanced methods can connect complex structures in a similar manner, leading to hybrid models (see Section 2.5.3).

The motion fragments are stored in a directed graph structure, with all possible clip connections

usually being precomputed. For this reason, the computational cost of motion synthesis is minimal; the most complex part is motion planning, which has to search through the graph to synthesise a motion with the required properties. The resulting motion is highly realistic, as all clip transitions are at points with high similarity. However, the synthesis system is less responsive and incapable of reaching certain configurations, as the transition can happen only in discrete intervals (Reitsma & Pollard, 2007). A way to address this drawback is to extend the set of input motion clips by their pairwise blends, thereby increasing the inter-clip similarity and providing more transition edges, at the cost of more complex motion planning (Zhao & Safonova, 2009).

The first set of fragment based motion synthesis methods starts with an **unannotated and unstructured database** of motion clips. Because there is no explicit meaning assigned to the motions, which could be used to limit the fragment search, a search through the whole graph structure is required. Early methods plan the motion using the graph structure directly, often leading to large planning times. Later techniques utilise a certain amount of precomputation, thus speeding-up the search procedure.

One of the first methods that uses *global search,* and a description of challenges and issues connected with such approaches, was provided by Lamouret & Van De Panne (1996). Using a simple lamp model and a database of motions generated using physical simulation, they show that a method based on splitting motions into fragments and searching through them to find a suitable connection clip is capable of creating plausible animations. To clean the final result, they use an IK-based motion editing technique.

Markov chains are statistical structures made of a finite number of states connected with edges that represent transition probabilities. Galata et al. (2001) use variable-length Markov chains to represent different behaviours in motion data (body contours and 3D marker trajectories). The states of their chains are described using a common alphabet obtained by vector quantisation of input motion clips, split beforehand into atomic components. Lee et al. (2002) apply a similar approach to skeletal motion data, with each frame represented as a Markov state. For storage optimisation, they present a method to prune the resulting graph. Novel motions are synthesised using graph search, sped up by higher order statistical models and cluster analysis. Kovar et al. (2002a) creates motion graphs without the explicit use of statistical techniques, making the approach simpler to implement. Arikan & Forsyth 2002 ease the problem of the costly graph search using a fast randomised algorithm with progressive refinement.

Constraints provide a simple yet powerful way of describing different requirements placed on the resulting motion. Liu & Popović (2002) provide an optimisation technique, which creates a motion by combining short motion clips, while enforcing the smoothness and physical constraints (e.g., balance, momentum).

Kim et al. (2003) create a motion graph suitable for dance and rhythmic motions. They first derive motion beat and rhythmic patterns (a sequence of motion beats) by analysing second derivatives of motion signals (a method previously used by Bindiganavale & Badler (1998)). Splitting the motion data into fragments according to these patterns then implicitly guarantees music correspondence.

To improve motion graph search performance, many later techniques incorporate a certain amount of *precomputation* or *graph optimisation.*

Lee & Lee (2004) describe the motion graph as a parameterised state-action model (with parameters such as reachability and direction) and discretize the animation state, thereby reducing the number of graph nodes. With this representation, they use reinforcement learning to rank the reachability of transition edges, eliminating all rarely-used edges. In a similar manner, Srinivasan et al. (2005)

precompute a mobility map consisting of multiple least-cost sequences from each graph node. Defining reachability, these limit the searching space, which allows greedy local searches to be performed rather than expensive global ones.

Probabilistic Roadmaps (Choi et al., 2003) represent the environment in terms of randomly placed oriented footsteps and connections between them, thereby describing the possible paths from one foothold to the next. A path through the environment is planned on this structure using a graph search algorithm. The footsteps from this path are used to extract a set of matching clips from the database, which are then connected and adapted to the desired trajectory. Sung et al. (2005) extend the clip extraction and adaptation procedure by a progressive refinement from both ends of the planned path.

If the motion clips in the source database are **annotated** with motion type, the motion graph can be assembled into a structure that explicitly represents this information. This provides additional control over the animation, which is often desirable for practical applications.

A classical example of such an approach are *Move Trees*, manually defined annotated graph structures used extensively in the animation industry (Mizuguchi et al., 2001). The animator is responsible for providing source animation clips and defining transitions between them, determining all possible motion combinations beforehand. Gleicher et al. (2003) provide an intuitive interface for authoring move trees, called *Snap-together Motion*. First, with the assistance of an automatic algorithm, the user identifies a set of common poses between different input animations. These poses are then merged together, forming a graph node, with fragments of the original animations that describe the graph edges. The result is a well-defined annotated motion graph that is directly usable for real-time animation. *Fat Graphs* (Shin & Oh, 2006) merge multiple edges connecting two common poses into a fat edge, parameterised along an user-defined joint, thereby creating a hybrid model (see Section 2.5.3).

Arikan et al. (2003) use motion annotations explicitly to specify the desired motion's parameters. These annotations are user-defined, can represent both the motion type or its style, and allow multiple annotations for a single motion clip. To ease the annotation process, they employ support vector machines, which allow a motion database to be annotated using a set of examples. The motion synthesis is then guided by the user by specifying the annotations and their combinations on a time axis. A dynamic programming algorithm searches the database for most appropriate clips and stitches them together to produce the final motion. The level of control is limited only by the type of annotations provided.

2.5.2. Parameterisation Models

Parameterisation methods are aimed at solving the inverse motion problem – from a set of high-level parameters, determine the combination of motion clips from a database that would lead to a motion with desired properties. As with forward and inverse kinematics, the inverse problem is burdened with inherent non-linearity, a large number of possible solutions for certain cases and only approximate solutions for others.

The following section describes previous work on data-driven parameterisation models. A large group of similar models, called controllers, use analytical equations to generate the animation sequence. However, this group is out of scope of this thesis.

Generic Parametric Models

Generic data-driven parametric models do not limit the type of the parameterisation used for describing the target properties of the motion. As such, their parameters can, for example, describe speed and velocity for locomotion, target location of a punching motion or a dancing style, all represented

in the same manner. A number of methods that could be classified as being in this group, but which are usually used for altering a motion sequence instead of synthesizing a new one, are described in Section 2.3.2.

Perlin et al. (1995; 1996) define source motions analytically using a grammar (as a controller). They show how, in this representation, motions can be blended (Perlin, 1995) and parameterised (Perlin & Goldberg, 1996). They also provide a scriptable interface that allows interactive responses, as demonstrated in an application for multi-user chats.

Several methods allow the synthesis of reaching or other spatially-constrained motions using an approach resembling data-driven inverse kinematics with filtering. Wiley & Hahn (1997) introduce a scheme to linearise the parameterisation of an example-driven reaching motion by resampling the parametric space. The new samples, created using linear blends between the original motions, are then registered into the parametric space, thereby providing a local linearisation of the parametric manifold. Their blending them uses nearest-neighbour linear combination of sample motions. While simple and fast, this method is not smooth in the first derivation, and possibly creates visible discontinuities. To address this issue, Rose et al. (2001) use cardinal-basis function interpolation. Although not local and possibly resulting in negative weights, their approach provides a smooth interpolation function, and is also capable of determining the best position of pseudo-samples, based on the smoothness of the blend-weight parameters.

Kovar & Gleicher (2004) return to the linear nearest-neighbour approach in their system, arguing that the necessity of interpolating all example motion creates an unnecessary computational overhead. Their approach is based on comparing motions using a match web as follows: using a metric, create a 2D map of comparison values between each pair of frames (in the same spirit as our Motion Maps – see Section 5.2); then detect the map's local minima and connect them into chains. This procedure provides a set of possible motion matches together with their timewarping function. After computing the match webs of all pairs of the input motions, they use a similar approach to Wiley & Hahn (1997) to fill the parametric space with pseudo-examples using blending, which results in a locally linearised version of the inverse function.

Mukai & Kuriyama (2005) incorporate previously described correspondence determination scheme together with the monotonically increasing timewarping function introduced by Park et al. (2004), (see below) in their statistical interpolation scheme. The interpolation of this scheme uses a kernel-based blending method built on a function prediction that minimises the resulting artifacts.

Standard statistical methods used for dimensionality reduction can also be used for parameterisation. Unfortunately, these methods extract parameters that have only mathematical meanings (e.g., direction of highest variance) and do not correspond to logical ones. Nevertheless, they can be used for compression or for easing-up an optimisation problem (Safonova et al., 2004).

Locomotion Models

Most of the parametric methods introduced above are capable of creating a parametric locomotion model with a certain amount of accuracy, even though they are not designed to solve this problem in particular. In the next few paragraphs, we will describe several methods targeted solely at locomotion generation, which makes them closely related to our parametric space model (see Chapter 5).

The main inspiration for our work is a group of locomotion methods based on trajectory analysis and parametric motion synthesis. A classic example of these approaches was presented by Park et al. (2002). Their locomotion model is described using three parameters – style (a discrete parameter based on the clip annotation, which has to be available beforehand), speed and turning angle. First,

the source motions are analysed by fitting a circle into their XZ trajectory and extracting their turning angle and speed. Using these values, each clip can be placed into the parametric space structure. At runtime, each clip's weight value is first determined using cardinal basis function interpolation (see Sloan et al. (2001) for a detailed description). All clips are then timewarped using an incremental timewarping scheme, a set of frames for blending is determined and the blended frame is computed using quaternion exponential maps (each joint is handled relative to a best average orientation determined using a non-linear optimisation step). Because the final blend does not correspond precisely to the requested trajectory, the last step of the method adapts the footsteps and root position / orientation to follow the desired trajectory precisely.

The main disadvantage of this method is the chosen parameterisation scheme (Kovar & Gleicher, 2004). While smooth in any derivation, it assigns a non-zero weight for every motion in every point of the parametric space, thus requiring an expensive many-frame blending for each frame. Moreover, it can assign a weight lower than zero, which does not have any physical meaning. Finally, when the parameter values are distant from all datapoints, the function converges towards an average of all input motions. All these issues are addressed in our solution to the locomotion generation problem (Chapter 5).

In Park et al. (2004), an extension of the Park's original parametric locomotion model addressed several issues with the original method and led into the creation of a hybrid model in the spirit of Rose et al. (1998). There were two major improvements of the parametric part. The first improvement altered the timewarping method – the original scheme permitted negative values, leading to motions that ran backwards in time. This was addressed by introducing a strictly non-negative timewarping function, different from the one used for motion blending. Second, the expensive optimisation process computing the best average for quaternion blending was replaced by the value from the previous frame. Assuming that the frame difference is small, this value provides a good approximation of the best average point. The original parametric model represents nodes of a graph, allowing to connect several different parameterisations by transition edges.

The most recent addition to this group, and also closest to our work, is the parametric model by Johansen (2009). Their technique uses gradient band interpolation, which addresses the drawbacks of previous methods, providing a non-linear, but continuous, non-negative and monotonous interpolation function. However, as the resulting weights are non-local and computed using an optimisation, both the costs of their computation and of pose interpolation are significantly higher than in our approach. Nevertheless, the overall approach, locomotion cycle analysis and their inverse kinematics techniques are closely related to our method (see Chapter 4).

An alternative parameterisation can be achieved by significantly simplifying the motion. Extending the rhytmic motion synthesis method (Kim et al., 2003), Kwon & Shin (2005) decompose animations into segments using local extrema of the COM (centre of mass) trajectory. By analysing the footstep pattern, they separate these segments into groups and use each group to create a parametric space of a particular locomotion type. Afterwards, they create a transition graph between these parameteric spaces (with structure similar to Rose et al. (1998)), with each node representing a particular locomotion type and its walking pattern, and the overall graph allowing type transitions.

A recent addition to locomotion models was presented by Lee et al. (2010). First, the frames of input motions are represented as motion states, holding a vector with pose information and its first differential. All the input states form a high-dimensional continuous motion space, i.e., a path which represents a motion sequence (both original and synthesised). The value of each motion state and its position in the motion space determines the possible actions that can be taken in the next

frame. Each action is a discrete sample from the continuous motion space, created using a linear combination of neighbouring states, with a slight bias towards example states to avoid drifting into empty regions of the motion space. By randomly picking one of the actions, it is possible to generate a plausible random motion. The control of the final motion can be achieved in two ways. First, it is possible to assign weights to the action values, thus preferring certain actions over others. This simple method unfortunately does not provide exact control and prefers only short-term goals. The second method samples the motion field and uses a reinforcement learning technique over this discrete space. While much slower than the previous method, it provides significantly more accurate control over the resulting motion.

An important aspect of a locomotion is motion style, both individual and as a motion type descriptor. Several methods aim at providing a parametric model for motion style (for style transfer as a motion editing method, please refer to Section 2.3.1). Troje (2002) introduces a generic parametric model for pointlight walkers (see Section 2.7.1 for more details). This system analyses the source motions in terms of principal component analysis and contained frequencies (related to the PCA components), allowing a surprisingly large number of motion styles to be parameterised using simple linear methods. Another method for parameterising both the locomotion and its style, called Style Machines, was introduced by Brand & Hertzmann (2000). Their hybrid method represents motions as Hidden Markov Models (HMM) and allows them to be merged using statistical methods, to provide parametric abstractions of their styles.

2.5.3. Hybrid Models

Hybrid models combine both parametric approaches and motion graphs to produce a comprehensive parameterisable model of motion. Previous sections introduced several of these models already, as a clear division between hybrid models and fragment-based / parametric ones is hard to make.

The first classical example of a hybrid model is the *Verbs and Adverbs* approach introduced by Rose et al. (1998). In this method, verbs describe the type of the motion (i.e., walking, running) and adverbs the style (i.e., slow, happy).

The approach consists of several steps. First, the user constructs a *verb graph*, i.e., he manually annotates motion clips with their type and creates possible transitions between them (several clips can have the same type). Then, he adds *adverbs* to the motions annotated with the same type. The system processes this information, creating transitions between verbs using IK and blending, and parametric spaces from adverbs. Interpolating between them is then performed using a radial basis function. While mostly manual, this method was the first of its type and served as an inspiration for many later techniques.

Several extensions of basic fragment-based motion graph techniques (Kovar et al., 2002a; Lee et al., 2002; Arikan & Forsyth, 2002; Gleicher et al., 2003) were proposed, thus making them hybrid models.

As previously mentioned, Shin & Oh (2006) introduced *Fat Graphs*, a parametric extension to the *Snap-together Motion* method by Gleicher et al. (2003). In their method, all motions originally connecting two nodes are merged (using timewarping and dimensionality reduction) into a parametric structure called *fat edge*. To allow user interaction, the parameterisation is converted into an intuitive 2D representation along a selected joint, which can be easily manipulated using a 2D controller (e.g., mouse).

In a similar manner, Heck & Gleicher (2007) create Parametric Motion Graphs. Using an automatic motion search and parameterisation described by Kovar & Gleicher (2004), they create a modified motion graph with states describing parametric structures and edges providing transitions between

them. The result of this method closely resembles the original work of Rose et al. (1998), in this case created using an automatic algorithm.

A hybrid method particularly suitable for human locomotion was introduced by Park et al. (2004). Extending their previous work on locomotion parameterisation (Park et al., 2002) and addressing several of its issues, they connect several locomotion parametric spaces into a graph structure. This allows them to transition between different types of locomotion (e.g., running, walking or crawling). Each of the parametric spaces uses the same parameters, thereby providing a level of consistency between different motion types.

Another human locomotion method, building on rhythmic motion analysis (Kim et al., 2003), was introduced by Kwon & Shin (2005). First, in a preprocessing step, they decompose the source motions into fragments by analysing the COM (centre of mass) trajectory. Then, they analyse the footstep pattern of these fragments and the corresponding ones together, thereby forming a parameterised state of a graph structure. The footstep pattern analysis also provides motion fragments that can serve as transitions between different states. The final structure represents a parametric and periodic walking pattern in each node, thereby allowing transitions between them.

The last hybrid method described in this section is *Style Machines* approach, introduced by Brand & Hertzmann (2000). In this method, the structure of the motion is represented as a Hidden Markov Model (HMM), with each state described as a Gaussian curve in the many-dimensional space of full-body poses. The method first converts each motion into this representation and then merges the states using an optimisation that minimises the state entropy and cross entropy. This process results in a single HMM describing the motion patterns extracted from all clips in the database (with their transitions), and a small number of parameters to adjust the style. In the examples provided in this paper, of particular interest is the synthesis of walking patterns (periodic motions), which leads to a circular graph with parameters thus span all styles of locomotion in the source database in a consistent manner.

2.5.4. Group and Crowd Animation

The target of crowd animation is to generate the realistic motion of a large number of characters simultaneously. For this reason, it is not feasible to define the motion properties for each particular character's motion on a level usual for single-character animation. Consequently, crowd simulators work at a higher level of abstraction, with parameters such as crowd density, overall crowd direction or a source/target (crowd generator and sink) model. The overall structure of a crowd simulator is often described using a behaviour pyramid, with levels of higher abstraction at the top and physical / display levels at the bottom (Paris & Donikian, 2009). This thesis is focused on the lower levels of this pyramid, i.e., the generation of the character animations and the deformation of the models representing them.

The following section describes methods that handle the motion of a group of characters moving simultaneously. We do not address the behaviour simulation, however, as this vast field of research is outside the scope of this thesis.

A classical example of simple trajectory planner is the *Boids* system (bird-like objects) by Reynolds (1987). This simulator can synthesise non-trivial scenarios by exploiting the emergent behaviour of relatively simple individual agents. Extending this approach, Lai et al. (2005) present group motion graphs, a motion graph-like structure generated from the results of boids simulation. It allows the synthesis of group motions based on simulation examples and improves the predictability of the results, without running an expensive simulation.

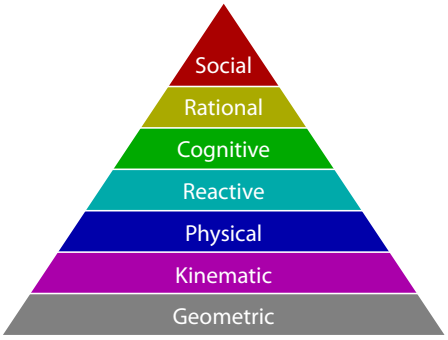

Figure 2.3.: Behaviour pyramid, illustrating the levels of abstraction used in behavioural simulation. Each level sends main data output down the pyramid, while receiving a feedback (upward direction) (Paris & Donikian, 2009).

Kwon et al. (2008) introduces a simple method of group motion editing at the trajectory level. Their technique is based on warping the trajectories of the characters with as little change as possible, while maintaining their spatio-temporal distances. They represent the trajectories of a group of characters using a spacetime mesh. 3D curves in spacetime that represent the 2D trajectories of individual characters and their temporal aspects, are connected by edges representing their spatial relationship, thus creating a 3D mesh (vertices correspond to a single character's position at a given moment). The user then defines constraints by dragging the vertices (displacing individual characters) or by imposing constraints on the shape (e.g., limiting the 2D trajectory information by obstacles, such as squeezing the characters through a narrow hole). New trajectories are then created by manipulating the mesh using Laplacian mesh deformation, which preserves the details and overall shape and structure by changing the original mesh as little as possible.

An even higher level of preprocessing is used by Yersin et al. (2009) in their *Crowd Patches* method. The whole environment is built out of precomputed blocks, with each of these blocks representing spacetime motion curves (3D – 2D trajectories and time) of all characters passing through it. Each side (i.e., 2D plane represented by a spatial edge and the time vector) of each block poses hard constraints on the entering and exiting trajectories, selected from a pool of patterns. After simulating the content of each block (accounting for character avoidance, environment and high-level behaviour), the continuity between neighbouring blocks is guaranteed by this pool of examples. This technique creates crowd simulation with an extremely low computational overhead, as practically everything in the simulation is computed beforehand and only replayed in real time, albeit with extremely limited responsiveness.

A different type of environment mapping is used by Sung et al. (2005). In a preprocessing step, they create a *probabilistic roadmap* by randomly sampling the environment to create graph nodes and creating edges between all pairs of nodes if they are reachable and correspond approximately to the length of one character's step. Using this roadmap, the motion planner first finds the shortest path from starting to target node that does not collide with any obstacles. Afterwards, this path is used to pick the motions that move the character between two consecutive nodes on the path. Starting from both the start and the end nodes, the planner iteratively picks a motion with the required properties using a greedy approach, and progresses along the path. Each clip change alters the resulting motion and its trajectory, with progressive refinement improving the result with each iteration. The iterative process stops when both forward and backward paths are similar enough, at which point they are both merged using a motion editing approach, thus providing the final motion. The planning for a

crowd is performed sequentially, with all previously handled characters being dealt with as moving obstacles (reusing the probabilistic roadmap structure).

A different kind of preprocessing approach precomputes the character appearance into the texture memory, displaying an image representation (called *impostor*) instead of the deformed geometry. This approach exchanges the computational cost of deformation and rendering of a mesh model with the storage space required to save all the possible deformations and view angles of the character. Hamill et al. (2005) provide a detailed evaluation of this approach in terms of number of poses required for impostor representation to minimise the impact on the perceived quality of the result. Dobbyn et al. (2005) introduce *Geopostors*, an extended version of impostors, that allows to seamlessly switch between the impostor and geometry representation, and enhances the realism of impostor crowd representation by adding an accurate lighting model and a colour variation scheme. To lever the storage requirements, Kavan et al. (2008) propose *Polypostors*, which render the characters using several polygons (instead of one quad required for an impostor), and allow their deformation, thus providing an approximation of the original mesh's animation.

2.6. Motion Metrics

A metric (or a distance function) is a function that defines a scalar distance between two objects in a set. This set, together with such metric, is called metric space. A composed metric can be created as a combination of several different metrics but, by definition, such a metric can also return only one scalar value. An example of a composed metric from the world of data-driven animation is a metric that compares animation frames using a combination of simpler metrics applied to particular joints.

In our case, we will focus on metrics defined on

- separate joints (or bones / rigid components),
- frames (pose information, with optional dynamic data),
- fragments (windows of frames),
- clips (longer frame sequences, usually with different lengths; the aim is often to achieve length independence),
- style (difference between two motions of similar type),
- artifacts (measuring the unnaturalness of edited or synthesised motion),
- classification (the probability that a motion is of a specific type, belongs to a specific group or has a specific property).

Metrics provide information about the motions and are therefore essential for all data-driven animation methods, as they provide a way to compare motions, measure physical properties, determine minimisation targets or the impact of physical interactions. They can be based on any property of the animation sequence, i.e., kinematic, dynamic, statistical, environment interactions or any other. There are three basic groups of metrics:

- analytic (based on purely mathematically derived properties of the motion),
- physically based (kinematic, dynamic or environment based)
- perceptual (modelling the way the motion is perceived, usually based on a perceptual experiment).

The analytical metrics are based on the underlying mathematical models that describe the motion and the environment. While often relatively easy to evaluate and consistent with the algorithmic foundations of the animation methods, they do not necessarily reflect any objective properties of the motion.

Physically-based metrics are usually harder to evaluate and require complex models and additional

information about both the character and the environment (e.g., weights of the character's bodyparts, joint limits and friction models). Moreover, they usually represent only very simplified models of the character, as the realistic simulation of all muscles, the skeletal structure and neurological principles is still beyond our reach. Nevertheless, they provide results that are significantly closer to the real world than the analytical metrics, and as such have become popular in recent years.

The last group, perceptual metrics, mimic the response of the human visual system to a particular motion property, described by the statistical evaluation of results of a perceptual experiment.

In this thesis, the analytic and perceptual metrics are represented in several places. The analytic metrics are used mainly for constraints detection (see Chapter 4) and for motion analysis for our parametric space concept (see Section 5.2). Perceptual metric form a large part of this thesis, particularly represented in Chapter 6.

2.6.1. Metric Definition

Informally, a metric, or a distance function, is a way of measuring distances between two objects (with no restriction on the object type). It is a function that takes two objects of the same type and returns a single scalar value, i.e., their distance. More formally, it is defined as a function d mapping a pair of objects from a set X to a real number

$$d: X \times X \longrightarrow \mathcal{R}$$

This function has to satisfy following four conditions (with $x, y, z \in X$):

1. **Non-negativity** – all values of a metric have to be larger than or equal to zero

$$d(x, y) \geq 0$$

2. **Identity** – a metric results in a zero value if and only if both objects are equal (implicitly defining the equality operator)

$$d(x, y) = 0 \iff x = y$$

3. **Symmetry** – the order of operands is not important, i.e., it is a commutative operation

$$d(x, y) = d(y, x)$$

4. **Triangle Inequality** – ensuring that the distance from an element x to z via an element y is at least as great as the distance from x to z directly (i.e., if the three distances are used to create a triangle, this triangle will be non-degenerate)

$$d(x, y) + d(y, z) \geq d(x, z)$$

This definition provides a way of describing different properties of the compared objects. Please note that each of the metric functions outputs exactly one real number. More complex metrics can be created using a combination of simpler ones, but according to the metric definition, they can also output only one real value.

2.6.2. Per-Frame Comparison Metrics

Per-frame metrics combine the joint values (i.e., kinematic information) of two frames, often together with the associated dynamic properties (or, generally, any per-frame information), into a metric that describes the difference between two animation frames. Out of the large number of metrics described

in previous work, this section attempts to provide an example for each of the common types.

The simplest way of representing the pose of a hierarchical model is to use the angles of its joints (i.e., Euler angles representation). Bruderlin & Williams (1995) use this representation in their Motion Signal Processing work and use it also to determine the frame differences. Unfortunately, Euler angles do not represent the space of 3D rotations well (see Section 2.1.2) and any metric using this representation will have the same issues.

A metric can be created using a (possibly weighted) sum of Euclidean distances between corresponding points placed on the character. Two common versions of this metric use a pointcloud created from a low resolution version of a character mesh (Kovar et al., 2002a) and points placed in the location of joints (Arikan, 2006). If the metric is to represent only the differences of poses independently of the environment, then the global orientation and position of the character has to be removed first from both compared frames. In the case of joint locations, the XZ translation and Y rotation can be removed by decomposing the root transformation. To align two pointclouds, Kovar et al. (2002a) use an analytical solution to the alignment problem described as a least-squares fit.

Another very common method of measuring differences between two frames is based on a weighted sum of the lengths of quaternion logarithms computed from difference quaternions of corresponding joints (Lee et al., 2002). This metric was presented in two versions – with the root translation/rotation term to account for the environment and without, making the metric only pose-dependent. To distinguish between the direction of the motion, Lee et al. also add a second term computed in a similar way out of the first differentials (velocities) of the joints. Based on the previous description, it can be seen, that the metric works in the local space of each joint, independently of the hierarchy. For this reason, correct weight values are critical for a good performance of the metric. Lee et al. (2002) proposed to use binary weights – one for important joints (shoulders, elbows, hips, knees and spine) and zero otherwise. Johnson (2002) refines these values, adjusting them based on relative joint importance. Wang & Bodenheimer (2003) provides another set of weights, evaluated by a user study. Their weight set is determined using least squares fitting on a set of hand-picked good and bad transitions from a database. They find that the overall weight of the velocity part of the metric does not seem to make any difference to the outcome; for the joint differences, their results give the highest values for hips and shoulders, small weights to knees and elbows and zero otherwise.

Forbes & Fiume (2005) use a metric based on the values of the eigenvectors determined using Principal Component Analysis (PCA). In their motion search algorithm, they convert the motion data into principal components by first subtracting the average pose, linearising the remainder using exponential maps and decomposing it using weighted PCA (the weights provide accuracy for each analysed joint; necessary for correctly compressing hierarchical data). Their metric then compares vectors composed of the values of important components (i.e., with high associated eigenvalues). This reduced metric provides an effective way of both searching a database of motions and performing the temporal alignment of clips using timewarping.

Another statistical metric was introduced by Li et al. (2002). The original motion is decomposed into small segments called motion texons using a Markov approach in combination with linear dynamic systems. The transition likelihood between these segments can then be used as a similarity measure.

A fully dynamics based metric was proposed by Liu & Popović (2002). They compute the position of three center-of-mass points computed from the character's body (lower body, upper body and arms), relative to the character's centre of support. Although they use this metric for a specific task (comparing poses during takeoff and landing in a jumping sequence), it provides a good similarity measure for any dynamic motions, while remaining very simple and reducing the number of compared

parameters. As its values are determined by abstracting the actual structure of the character, it can be used to compare motions performed by humanoid models with very different topologies.

A vector of analytically defined Boolean features can be used as a very simple, intuitive and fast way of comparing motion frames (Müller et al., 2005). Each of these features represent a property of the frame pose, such as one leg in front of the other or one foot interacting with the ground.

To determine which of these metrics best reflects both qualitative and quantitative frame differences, Van Basten & Egges (2009) performed a comprehensive study consisting of motion graph building and a perceptual experiment. They tested three metrics – local joint angles (Lee et al., 2002; with a very small weight on the velocity part), geometrical (Kovar et al., 2002a) and PCA-based (with weights determined using the eigenvalues of each component).

In the quantitative test, they first build four complexities of a motion graph, with five animations as their source data (832 frames in total), and then synthesise a 33 meters long zig-zag path, testing three different criteria. The first test is aimed at measuring footsliding, i.e., the distance the foot travels during the footplant. Their results suggest that the best metric to avoid footsliding is the geometrical one, as it works in the Euclidean space instead of local joint space. The second test evaluated path deviation as an integral of distance from the desired path. Their conclusions show that the local joint angles metric provides the best path following, as it prefers inter-animation blends. This leads to a large amount of turning transitions, while other methods result in many self-blends (under the constraint of approximately the same number of transitions per motion graph). The third test measured the on-line graph search time. Again, joint-angles provide the best results, because the other metrics tend to cluster nodes together into small circles, leading to rapid growth of the search space.

The qualitative test was designed as an on-line perceptual experiment. Using a pool of 50 motions, generated as 10-frame blends with 5 different distance levels together with 5 original animations, each participant was presented with 40 clips and was asked to grade them according to their level of realism on a scale from 1 to 10. The value of 10 frames for blends was suggested as ideal in several previous works (Mizuguchi et al., 2001; Kovar et al., 2002a; Rose et al., 1996). Their hypothesis was that there will be a strict inverse relationship between the distance level and blend realism. This hypothesis was confirmed partially, as there was an inverse relationship, but the shape of the results curve exhibited a significant plateau at the beginning and a non-linear relationship for the rest of the curve.

Another per-frame metrics test based on human perception, using the local joint metric by Lee et al. (2002), was performed by Wang & Bodenheimer (2004). In their work, they focus on determining an optimal blending length, the use of timewarping and the detection of blending points. They conclude that a good transition detection method should use a window of frame differences, that the 10-frames blend length is indeed the best compromise and works under most circumstances (while 8 is already recognisably different) and that for this short blending window, timewarping does not provide any visual advantage.

2.6.3. Clip Comparison Metrics

Comparison of animation clips can be performed by applying a per-frame comparison metric to the frames of the two animations and computing some function of the resulting values (usually an average or a sum). However, to obtain a meaningful result, the temporal alignment of the two input clips needs to be established, to ensure that only frames describing similar motion stages are compared.

Taking their inspiration from the signal processing field, Bruderlin & Williams (1995) introduce the usage of dynamic timewarping (DTW). This method allows the temporal correspondences between

two motion clips to be determined as a piecewise-linear function, linking the best matching candidate indices selected using a per-frame metric. To achieve better temporal resolution and to limit the maximum function slope (i.e., maximum time skewing), Kovar & Gleicher (2003) introduce temporal registration curves, which are further refined in their follow-up work (Kovar & Gleicher, 2004), and later combined with Weighted Principal Component Analysis to obtain better computational efficiency (Forbes & Fiume, 2005). The timewarping can be accompanied by spacewarping, thereby providing higher alignment accuracy. Hsu et al. (2005) introduce an optimisation technique to solve both parts of this warping combination simultaneously.

Instead of using the per-frame metrics, several approaches use per-bodypart features to compress the pose (or dynamic) information in an intuitive and meaningful manner. Chiu et al. (2004) create an index structure of per-frame feature vectors, composed of binary representations of analytically defined frame properties. This allows for faster retrieval of motion sequences, which can be based not only on the whole motion, but also on its fragments or on a small subset of specific features.

A variation on a similar theme was described by Onuma et al. (2008). Instead of using a feature vector on per-frame basis, they compute its values using an approximation of the total energy of each joint (with several filtering and shaping steps to ensure that the result are not biased by the noise in the motion data). To ensure independence from the actor's bodyshape, they compose values of bodypart joints together (in a similar way to Chiu et al. (2004); presented as a simple way of dimensionality reduction) and then compute the ratios between different bodyparts. They show that this method can also provide a meaningful and length independent way of distinguishing different types of motion for motion classification.

2.6.4. Classification Metrics

Classification metrics are aimed at extracting the meaning of the motion and, based on this information, to sort the motions into categories without considering the style of the motion or properties of the performer's body. Categories are defined by heuristic analysis of the typical properties of a particular type of motion, or by a set of example motions and a distance metric (selecting the category of a motion by the closest example). Obviously, for the latter case, any of the clip comparison metrics can be used, but several methods were developed specifically for this purpose.

One such method was introduced by Arikan et al. (2003), where a statistical clustering approach based on support vector machines is used to generalise and cluster a database based on a small set of example motions. Unsupervised methods create clusters of motions without the need for explicit examples, e.g., segmenting in PCA reduced space (Barbic et al., 2004) or clustering in the reduced space of bodypart energies (Onuma et al., 2008). Analytically determined Boolean feature vectors can also provide similar clustering capabilities (Müller et al., 2005).

2.6.5. Artifact Metrics

Artifact metrics are used to determine the magnitude of the impact of an effect on the naturalness of a motion. Usually, each metric attempts to measure only one specific effect (or in a specific scenario), because of the vast number and diversity of possible artifacts in the motion.

To measure the effects of artifacts introduced by a motion compression algorithm, a pose distance metric can be used. In this case, it is explicitly defined what the correct answer is, and as the number of frames should correspond, any of the clip or frame difference metrics can be used (e.g., Arikan et al. (2003) use the per-frame comparison metric introduced by Kovar et al. (2002a) to evaluate artifact levels between several compression methods).

Van Basten & Egges (2009) perform a comprehensive study to determine the properties of several

per-frame metrics on the problem of motion graph building and motion generation. Apart from the actual tested metrics, they also use two artifact metrics to determine the performance of each method. The first of these metrics measures the level of footsliding by first detecting the expected footstep using Y thresholding and then integrating the amount of XZ movement of the foot during the expected constraint. The second metric determines the amount of path deviation between the input and output of a motion synthesis algorithm by integrating the distance between the expected root position and its actual position.

Methods addressing a general notion of naturalness of motion are usually closely related to human perception, as a generic mathematical definition is still out of reach for current techniques. This paragraph provides only a very short list of these methods. For a detailed overview please refer to Section 2.7. Reitsma & Pollard (2003) focus on the perception of anomalies in ballistic motions, altering their timing, acceleration and gravity. McDonnell et al. (2007b) test the possibility of using different framerates in crowd simulations. Hodgins et al. (2010) determine the perceptual impact of several artificially induced artifacts, concluding that artifacts in facial motion are significantly more salient than artifacts in body motion. Harrison et al. (2004) test the perceived levels of limb extension in an animated motion. And finally, Ren et al. (2005) provide a comprehensive study of different automatic methods of determining the naturalness of a motion clip (based on machine learning) and compare their results with the outcome of a perceptual experiment. The conclude, that users are capable of significantly better performance than any of the tested automatic methods.

2.7. Perception of Motion

As the research on human animation algorithms matures, attention is shifting towards motion perception. This is a logical step, as results of these algorithms are almost always presented to a human user. As previously shown in many fields of computer graphics, the human visual system (HVS) can be very sensitive to certain features of the presented stimuli, while other, often mathematically significant factors, do not seem to be perceptible at all.

An ultimate test of motion synthesis algorithms is the *Animation Turing Test*, proposed by Hodgins et al. (1995). In the spirit of the original Turing test of machine intelligence, it would present an observer with a real animation and a synthesised one, and the observer should not be able to determine which is which. However, a motion synthesis algorithm capable of both synthesizing a large variety of motion, and of passing such a test, has not yet been created (Van Basten & Egges, 2009).

In the context of this thesis, motion perception plays a crucial role. We have developed several metrics based on motion perception and tested the perceptual saliency of several motion features and artifacts, on both crowds and single characters (see Chapter 6). This section introduces a small subset of previous work in the field of motion perception closely related to our experiments.

2.7.1. Pointlight Walkers

The pointlight walker is an extremely simplistic stimulus representing human motion, capable of completely separating the appearance of the character (actor) from the motion information. The concept was introduced by Johansson (1973) as an extension of their previous work on the motion of markers on a structureless background. Using the technology available at the time, they attached a set of 10 to 12 back-reflective strips on the joints of an actor and captured his motion on a VHS camera placed next to a halogen lamp. By playing the recording on a TV screen with very high contrast, they obtain a moving image containing only white dots on a black background. In this pioneering work, Johansson shows that a moving stimulus of this type is immediately recognised as a human from any

natural point of view. This fact is not changed even by removing several markers. When the motion is stopped, however, the impression of a character is lost.

Cutting & Kozlowski presented two important studies based on pointlight walkers and a dataset of six walkers (3M, 3F). In the first study (Cutting & Kozlowski, 1977) they used the original actors as participants (a group of students who knew each other well), and demonstrated that the identification of a person is possible based on these stimuli. While far from 100% accurate (the average score was 38%), their results are significantly above the chance level. The second study (Kozlowski & Cutting, 1977) was focused on the walker's gender. With the sideways walking stimuli, the participants were able to recognise the gender correctly in approximately 70% of cases. However, their dataset contained one outlier – a female actor consistently labelled as male (an effect we found in our study as well; see Section 6.1.1). They also tested several other stimuli variants. Altering the arm swing, walking speed and blocking the upper or lower part of the body resulted in reduced recognition performance; static displays led to gender recognition at the chance level.

Barclay et al. (1978) tested the impact of several other properties on a walker's gender recognition. They conclude that several aspects are required for successful results (above chance) – motion duration at least 1.6s, two step cycles, normal presentation speed (timewarping the motion leads to chance levels), crisp and small markers (diffused markers lead, again, to chance levels), normal display orientation (vertically reversed display led to recognition significantly below the chance level, reversing the recognised gender).

Other researchers focused on gender recognition with different types of motions. Dittrich (1993) presents a cross-evaluation of different actions (locomotions, instrumental motions and social interactions) under different presentation setups (markers on joints, in the middle of bones; normal and vertically reversed display). He found that locomotions provide the best gender recognition scores, but all other actions are significantly above chance level as well. Of the different setups, only the vertically reversed presentation led to a significantly lower recognition rate. Later, this work was extended to the perception of emotional body language (Dittrich et al., 1996). A pantomimic performance of several different emotions (surprise, fear, anger, disgust, grief and joy) was presented to the participants under several different setups (markers, full light; normal, upside-down). The results shown that all emotions were recognised with scores significantly higher than chance, with the full-light displays and normal orientation significantly higher than the others.

Given the simplicity of pointlight walker stimuli, it is possible to create their **generative model**. Early work on this subject was performed by Cutting (1978), who created an analytical set of equations describing the motion of each marker and explored the correlation of its parameters with gender recognition.

However, important work on this topic, applicable directly to both psychophysics and character animation, was presented by Troje (2002). He introduces a data-driven pointlight walker synthesis model based on statistical analysis of a database of motions. In contrast to the previous work, his pointlight walkers are created using motion capture technology, with each marker being described as a 3D trajectory of an articulated figure's joint. This provides the means to perform more comprehensive and accurate motion analysis than was possible with the 2D data representation.

The source database for his model contained 40 subjects (20M, 20F), each consisting of 15 virtual markers computed using the trajectories of 38 motion capture markers. Several steps are performed to extract the statistical properties of this data. First, the space of poses (animation frames) is reduced by performing principal component analysis (PCA) on the marker positions in all input frames. This provides very successful compression, with 98% of the information contained in first 4 components.

Second, the values of each component can be successfully modelled using a sinus function, with the first two components using the frequency of the motion and the other two its first harmonic. This leads to a compact 229-dimensional vector that describes all the data of a walking sequence. The third compression level is performed on the walking sequence data, leading to an average walk and 39 eigenwalkers (eigenvectors of full walking sequence).

In the first analysis (Troje, 2002), the main focus is on gender recognition, testing several classifiers in this context (e.g., height, dynamic properties, walking frequency), and the effect of viewing angle and structural information. He also tests the effect of using a stick figure (markers connected by lines) instead of unstructured pointlights. In later work (Troje, 2008), this framework is extended to allow the building of generic linear classifiers (e.g., male/female, calm/angry, happy/sad, republican/democrat) by fitting a line described in the parametric space of eigenwalkers into results of a perceptual experiment. While a linear approach cannot accurately represent all possible motion properties, the results show impressive accuracy for most real-world scenarios (as demonstrated in an on-line application).

Although not directly applicable to the synthesis of 3D character animation, due to the fact that linear analysis does not respect the rigidity of human body segments and the simplicity of the parameterisation model, this work provides an important insight into the possibility of creating very simple linear metrics and generative models for human animation.

2.7.2. Perceptual Motion Metrics

Motion metrics are an essential part of the data-driven animation field, used both for comparing motions (or their fragments) and determining their specific properties. As the results of a motion synthesis algorithm are almost exclusively presented to a human observer, it is important to relate the properties of these metrics to the properties of the human visual system.

One of the important questions for perceptual metrics is whether it is possible to separate the appearance of a character from its motion (in a similar manner to a pointlight walker), using a simple model instead of a fully realistic character. Hodgins et al. (1998) asked this question, concluding that the answer is no. In their work, they compared a polygonal model to a stick figure, both of them animated using a parameterised model of motion changes (torso movement, dynamic arm movement, additive sinusoidal noise). The participants were presented with pairs of animations and asked to answer if they were "same" or "different". For all presented cases, the results show significantly higher sensitivity to motion alteration when the result is displayed on a polygonal character than on a stick figure.

A comprehensive study, testing the influence of character model on the perception of gender in animation data, was presented by McDonnell et al. (2007a). Following the series of studies focused on this topic using pointlight walkers (Kozlowski & Cutting, 1977; Barclay et al., 1978; Troje, 2002), their stimuli combine a set of different character models (male, female, neutral mannequin and a pointlight walker) and different locomotions (male, female and synthetic neutral). The results show a strong interaction between these, with male and female motions capable of altering the perceived gender of neutral characters accordingly, and providing an ambiguous result when applied to a model with the opposite gender.

Van Basten & Egges (2009) provide a comprehensive study of several different **frame comparison metrics** and their performance when used for transition detection (using a motion graph). The qualitative part of their work tested a hypothesis that the output value of a metric for a particular transition point should be inversely proportional to the realism of the generated transition. This

hypothesis is confirmed partially, as the relation is inverse, but it is also highly non-linear with a wide plateau for small differences.

Wang & Bodenheimer (2003) adjust the parameters of a particular frame difference metric (introduced by Lee et al. (2002)) to reflect the perception of motion blending artifacts. Using a set of user labelled motion transitions, they determine a set of weights by means of a least-squares fit. They validate their result using a simple perceptual experiment aimed at determining the more natural transition (at a point detected by the original and modified metric).

Several studies aim at providing information about a specific **motion artifact** on the naturalness of the resulting motion.

Harrison et al. (2004) studies the perception of length changes of an articulated joint structure during a motion. Their report contains several experiments performed on a very simple stimulus: i.e., two black segments connected by a joint on a white background, with one end fixed while the other one is performing a circular motion. They conclude that the perceptual threshold for segment extension is 7%, while shrinking is significantly less salient, with a threshold of 20%. The duration of the alteration plays an important role, successfully masking the change under certain conditions, while the presence of a distractor does not influence the experiment outcome significantly.

Reitsma & Pollard (2003) study in detail the perception of ballistic motion and its editing. Using a perceptual experiment, they determine that errors in horizontal velocity are easier to detect than vertical and that acceleration artifacts are more salient than deceleration. Moreover, they derive an analytical equation usable for predicting the saliency of a manipulation.

A comprehensive study on artifacts that influence the animation of different parts of character's body was presented by Hodgins et al. (2010). Using a set of fully animated and acted clips, they explore the impact of the presence of facial animation on the emotional response of the experiment participants. Furthermore, they use short extract of these clips with several types of animation artifacts introduced to determine their relative saliency. Their results show that facial artifacts are always more salient, even in the presence of very significant body artifacts.

A generic metric capable of judging the naturalness of any motion clip would provide an important tool for determining the efficiency of a motion synthesis method. Ren et al. (2005) present a comprehensive study testing a large set of reinforcement learning methods on this problem (mixture of Gaussians, hidden Markov models, switching linear dynamic system and naive Bayesian approach; using a global fit or as an assembly of several bodypart classifiers). As training data, they use a diverse set of 261 natural animations and 170 altered animations, with different levels of unnaturalness introduced (IK edits, keyframe changes, noise, transitions or motion capture noise simulation). Then, they present each method with a new set of motions to test its effectiveness. Moreover, they perform a perceptual experiment on the same data, with the results handled in the very same way as the result of the automatic methods. The overall performance of each method is then presented as an ROC (receiver operating characteristics) curve, which provides the ratio between true positives and false positives, or between the correctly and incorrectly identified motion clips. They conclude that the performance of a human observer is still significantly better than any of the automatic techniques, even though the hierarchical approaches, which construct an overall classifier from several smaller per-bodypart ones, perform significantly better than the global ones.

The main topic of this thesis is the simulation of crowds. Several perceptual studies in previous work focus specifically on **crowds perception**, usually with the aim of reducing the computational complexity while maintaining its visual quality.

Impostors can be used as a simplified representation of animated characters, using textured polygons

instead of complex animated meshes. McDonnell et al. (2007b) provide guidelines for the framerate required for a realistic crowd representation, directly usable as a sampling ratio for impostor-based systems. They conclude that 16 frames per second is a rate that, when applied to distant characters, does not create any perceptually salient artifacts. However, for close-up characters, 30 frames per second are necessary. Hamill et al. (2005) use a "same or different" task, as in Hodgins et al. (1998) to determine the required resolution, angular sampling and mesh-to-impostor transition distance to provide a realistic crowd effect. Using the same strategy, McDonnell et al. (2005) study the perceptual differences between several common representations of virtual humans, including high resolution models with approx. 2000 polygons, low resolution with 800 polygons, impostors, stick figures and pointlight walkers, with a particular focus on impostors and low resolution models.

McDonnell et al. (2008) perform a set of experiments on the perception of clones in a crowd system. Their main focus is on appearance clones, but a set of experiments on motion clones is also provided. In the comparative section of their work they conclude, that the appearance is more significant than motion when detecting clones in a crowd. For motions, they show that identifying a clone solely based on motion is a very hard task, made even harder by showing the motions out-of-step.

3
Motion Capture Pipeline

This thesis focuses on a data-driven method for motion synthesis. Our main source of motion data is our motion capture system – a commercial implementation of the optical motion capture technology with passive markers by Vicon (2009). It consists of 13 high-speed near infra-red (NIR) cameras, each equipped with a NIR LED-based directed illumination unit, shaped as a circle around the camera's lens (see Figure 3.1, right). The markers are plastic (for small sizes) or rubber (for larger sizes) spheres, covered in back-reflective tape. For the purpose of full body motion capture, we use markers with a diameter of 14mm and 20mm; smaller markers are useful for detailed hand and facial capture (see Figure 3.1, left).

The following section provides detailed descriptions and reasoning behind our camera placement, marker and skeletal setup, and the calibration process. Surprisingly, even though these steps are similar for all optical motion capture technologies, we are not aware of any description of them in any available literature.

3.1. Camera Setup

Camera placement is a crucial part of the motion capture setup process. The goal is to cover the capturing space as effectively as possible, with each point viewed by multiple cameras from several different angles. This provides a good source of information for the triangulation step of data processing. For practical reasons, it is necessary to place a large number of cameras close to the ceiling of our capturing space because the room is not dedicated solely for capturing purposes. Furthermore, the cameras can be mounted horizontally or vertically, thus providing larger viewing angles for a selected dimension.

Cameras are also equipped with different lens types. Six of our cameras use standard Nikon variable focal length lenses (i.e., zoom lenses with a large focal length, see Figure 3.1), and provide better resolution at larger distances while covering a relatively small viewing angle. The seven remaining cameras are equipped with CCTV (CS-type) short fixed focal length (prime) lenses (see Figure 3.1),

Figure 3.1.: Detailed view of our motion capture hardware. From left: Reflective markers used for motion capture, photographed without and with flash to demonstrate their back-reflective optical properties; a wide-angle IR camera; a narrow-angle IR camera; detail of a long-focal-distance lens with setup rings.

which are ideal for covering large view angles at the cost of lower accuracy at larger distances. Both types also differ in aperture values, with prime lenses allowing much wider settings and consequently more light reaching the sensor (F1.4 for prime lenses and F5.6 for zoom lenses, resulting in a difference in the maximum amount of light reaching the sensor by a factor of 16).

Each camera type therefore provides a different set of properties that, together with practical aspects, need to be addressed when designing an ideal setup. Our final configuration is shown in Figure 3.2. Six narrow-angle cameras are mounted vertically on the short side of the capturing area, covering it lengthwise. Their main purpose is to provide coverage for as much space as possible with high accuracy provided by their long focal distance. Three horizontally mounted wide-angle cameras provide a similar view for the long side of the area. As they are necessarily closer to the captured subjects, their wide angle provides better spatial coverage whilst the lower accuracy at large distances does not pose any significant problems. The remaining four cameras are mounted horizontally on tripods and placed in the four corners, closer to the ground. These cameras provide the coverage for regions not accessible by the long focal-length cameras mounted higher in similar positions. This configuration has proven to provide very good coverage of the capturing region, given the constraints and limitations of our system and capturing space (see Figure 3.3).

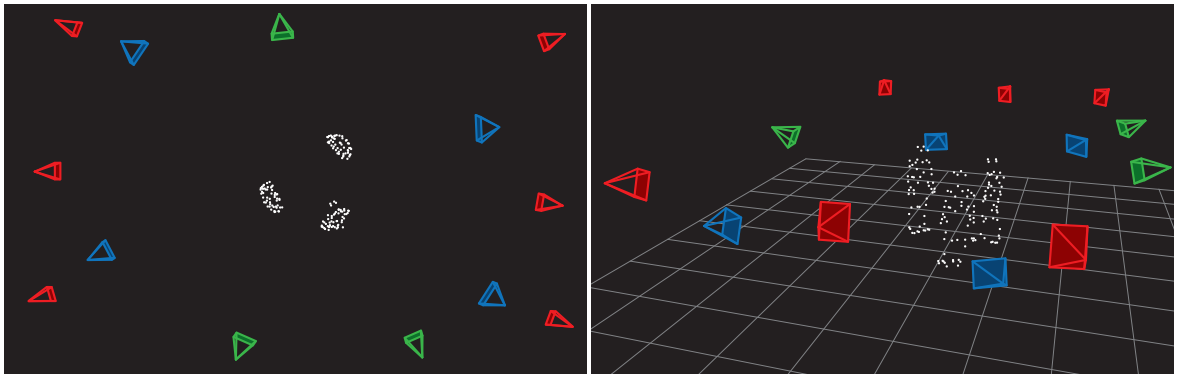

Figure 3.2.: Top (left) and perspective (right) views of our camera setup during the motion capture process with three subjects being captured. Six cameras form the long focal length set (red), accompanied by three wide-angle cameras (green) and four additional cameras closer to the ground level (blue).

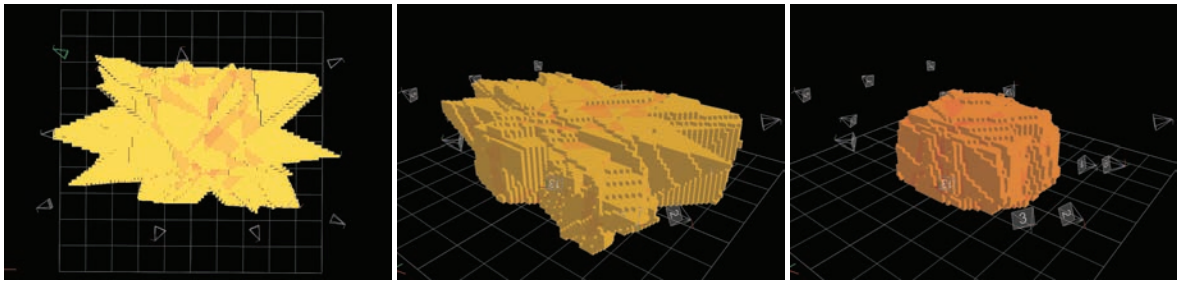

Figure 3.3.: Coverage of our camera setup. Top (left) and perspective (middle) view of the area covered by at least 6 cameras; perspective view of area covered by at least 9 cameras (right).

3.2. Calibration

After placing and pointing the cameras, it is necessary to adjust their parameters. These include lens aperture (F number), lens focus, strobe intensity, camera threshold and gain. This section describes the process of setting up these parameters, which needs to be performed every time the camera placement is changed to account for environment change. The procedure starts with each camera's strobe intensity set to maximum, fully opened aperture (lowest F number), focus set to infinity, gain to 1 and camera threshold to 20%.

The first parameter that needs to be adjusted is strobe intensity, which determines the strength of NIR light emitted by the ring surrounding each camera's lens. For most camera setups, the maximal intensity is the optimal setting, providing enough illumination for subjects approximately 1.5 to 8 metres from the camera. For shorter distances (e.g., facial motion capture), the strobe intensity has to be lowered in order to avoid ghosting, i.e., overexposed regions that are identified by the software as markers. Too low a value, however, can lead to insufficient contrast between markers and other objects in the scene, causing a large amount of noise.

After this first step, the values of focus, aperture and gain are adjusted (in this order). The first two parameters are set manually on each camera's lens by moving the corresponding rings, the third is software-based, set in the Vicon UI. A small marker is placed in the middle of the capturing area beforehand. We use 14mm markers for full-body motion capture, but a smaller marker might be necessary for setups aimed at recording facial or hand motion.

By moving the lens focusing ring, we should try to find a value that minimises the apparent size of the marker in each camera's view. This step needs to be performed on a fully opened aperture (lowest F number) as with other values, the unfocused marker would appear smaller, making the difference between the focused and unfocused image less obvious. If the marker is not visible in the camera's view, the gain value should be set to a higher value, which changes the software-based multiplier of the camera's output intensity. However, this step should be avoided, if possible, because it significantly increases noise levels in the result. When the marker appears to be in focus, the aperture ring should be set to the highest value that still allows the camera to cover the desired area. This can be tested by moving the calibration marker forwards and backwards in the camera's view. Setting the aperture to a lower level (higher F number) makes the marker appear smaller in the camera's view, thus providing more accuracy for the recording process.

The last parameter is the software threshold, used to distinguish marker data from background light. If performed correctly, the process described above should provide good calibration in most cases. However, if a camera still captures a large number of ghost markers, it might be necessary to raise the threshold value and, in some cases, repeat the whole calibration process for this camera.

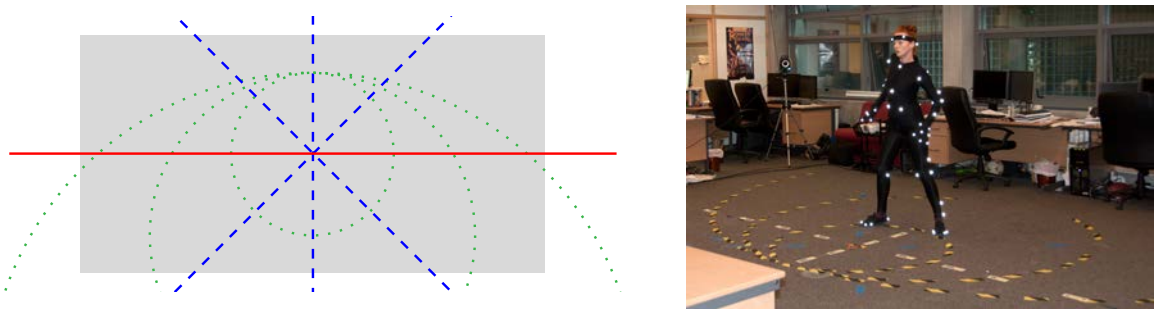

Figure 3.4.: Guiding trajectories for locomotion capture. Left – top view on the trajectory marks; the straight line marks (red), turning marks (blue) and circular marks, with 1.5m, 3m and 6m diameters (green), are all contained inside the capturing volume (gray), allowing to capture a large variety of different locomotions. Right – a photograph of the capture room with an actor.

3.3. Capturing Human Locomotion

Locomotion forms a significant part of human movement and was shown to contain a large amount of perceptual information (see Section 2.7). However, capturing locomotion in a limited space can be problematic. A possible solution could be a treadmill; but because walking on a treadmill is not natural, the subject usually needs a long time period to adjust to this type of movement (Troje, 2002). The usual construction properties of a treadmill also do not support turning motion.

Our solution uses straight lines and circular trajectories drawn on the ground of the capturing area (see Figure 3.4). The actor is instructed to walk on a line with a defined pace, with each path spanning approximately three periods of the motion. In the processing stage, it is possible to extract the walking curvature (by fitting a circle to the actor's trajectory; see Section 5.3.3), which is further used to separate the global position of the actor from the pose. Poses can then be merged into long clips using simple blending techniques, while the natural variance of motion is preserved in multiple trials of one motion type (see Section 5.3.1).

3.4. Human Skeleton Model

An unavoidable problem connected with optical motion capture technology is marker occlusion. While dense camera placement and correct setup can significantly reduce its effects, it is impossible to completely eliminate cases where a set of markers is not visible by at least two cameras.

Common motion capture software provides means to reconstruct the missing data using extrapolation (Vic, 2009; Menache, 2000). Apart from simple methods (e.g., using a linear function or a spline to fill-in the gaps in data), it is possible to use kinematic or dynamic solvers that can exploit the knowledge about model's topology. Vicon IQ software uses an iterative non-linear kinematic solver to perform this task (Vic, 2009).

However, during our experiments we found out that the provided stock markerset was too sparse, leading to artifacts with as little as two simultaneously occluded markers. Moreover, the stock skeletal model included joints that do not correspond closely to human anatomy. Based on the analysis of Menache (2000), we have developed a new markerset and skeletal model (see Figure 3.5), providing better coverage, closer correspondence to human anatomy and better results when used in conjunction with manufacturer's software.

The description of the human skeleton as an articulated hierarchy of transformations, referred to as a skeleton, is the most common way of representing human anatomy in the computer animation field (see Section 2.1.1). This representation uses a 6 degree of freedom (DOF) joint to represent the

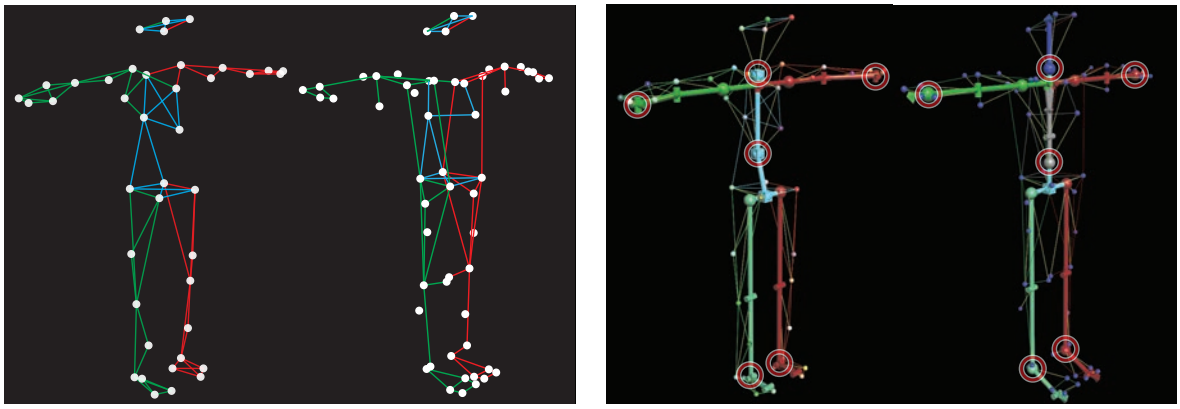

Figure 3.5.: Original Vicon markerset with 43 markers (left) compared to our markerset with 55 markers (right). The main difference lies in the denser layout with two markers describing each joint and at least one additional marker per each rigid body. Right: Differences between the original Vicon skeleton model (left) to our model (right). The main differences (circled) include 6 DOF joints in spine and neck as compared to the 3 DOF anatomically-correct version of our skeleton, 2 DOF joints on wrists and ankles compared to 3 DOF joints in our version and the different, anatomically-correct spine structure.

character root (usually placed in the pelvis area, close to the centre of gravity of the human body) and 3 DOF for the rest of the joints. This corresponds closely to the characteristics of the human skeleton, even though an actual human body is not perfectly rigid and its joints have a significantly more complex structure than a 3 DOF rotation around a point.

For motion capture purposes, this description is often extended by joint limits, allowing some joints to have only a certain DOF based on their local coordinate system. This is yet another simplification as these joints usually have a more complicated structure that cannot be described in terms of rotational DOFs (i.e., the wrist) (Menache, 2000). However, for the purposes of computer animation, this description is considered sufficient.

Our skeleton uses this form of description, but in comparison with the original model provided by the equipment manufacturer, it tracks the structure of the human body more accurately. Specifically, it does not use any 6 DOF joints (rotation + translation) with the exception of the root. The original production skeleton contained two more 6 DOF joints to represent spine and head, which allowed an easier fit to the motion data, but also produced severe non-rigidity artifacts. This property rendered the resulting motion unusable for the animation of mesh models.

The creation of the new skeleton and marker model was a tedious process, as the non-linear optimisation algorithm used for skeleton fitting gives unpredictable results. For this reason, it was necessary to use a trial-and-error refinement process, which in the end led to the new structure presented here.

3.5. Canonical Software Interface

Due to the lack of a generally agreed skeletal motion file format, interfacing different software in the motion capture pipeline can be problematic. Moreover, certain tasks (like footstep cleanup based on constraints detection) can be very tedious to perform in generic motion editing software. We address these problems using two tools implemented as a part of this thesis. The first one, the *Vicon Animation Converter*, is a GUI application allowing the Vicon data (V and VSK file formats) to be converted into more generic file formats suitable for our particular pipeline (BVH for skeletal data, CSM for marker data). Moreover, it allows the data to be previewed on a simple model, thus providing a way to identify possible motion artifacts. The second tool, the *Animation Editor*, is a command-

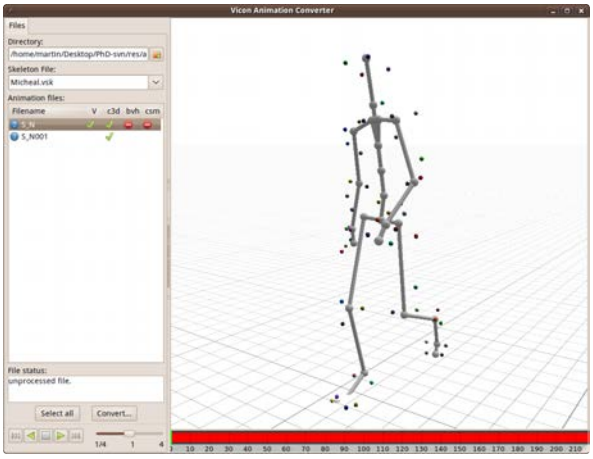

Figure 3.6.: Vicon to 3DS MAX animation converter software.

line tool allowing several common motion editing task (bodypart merging, constraints detection and enforcement, animation cutting, periodization, trajectory adjustment) to be performed, together with supporting several more file formats for both data import and export. As a command-line tool, it is fully scriptable, allowing the editing tasks to be performed in a semi-automatic manner.

Vicon Animation Converter

The export of motion data from the Vicon software into 3D Studio MAX (3DS), which is required by our processing pipeline, has proven to be a problematic task as both programs use proprietary file formats and do not support any simple way of exchanging information. The original solution required exporting marker trajectories from the Vicon software and carrying out the skeleton fitting in 3DS. This was not an ideal solution, as 3DS does not provide any means of skeleton calibration and consequently the skeleton fitting could not achieve accurate results.

The solution we developed moved the skeleton calibration and fitting procedures into the Vicon software. The processing is performed in two steps. First, a calibration motion is captured, exercising each joint of the actor's body. This motion is processed in a calibration procedure, adjusting the skeleton parameters to the captured data. Second, by fitting this skeleton to the rest of the captured motion clips, we can ensure parameter stability and guarantee a good fit to all the input motions.

The problem of data exchange was addressed by implementing a conversion tool capable of reading the proprietary Vicon ".V" file format and converting it into a broadly supported Biovision motion format. The main challenge was the reverse-engineering of the Vicon ".V" format, as its documentation is inaccurate and incomplete and it contains a large amount of data not required for skeletal data extraction. This tool has become an inherent part of our motion capture pipeline and is extensively used both in the Natural Movers project and in all other motion capture tasks for projects in our group.

Animation Editor

The animation editor tool is a more advanced command-line application for skeletal and marker data editing. It satisfies the need for extensive manual motion adjustments in generic animation software, which is both labour intensive and repetitive. Moreover, some of the supported manipulations are not possible in the canonical software (such as bodypart merging) and therefore have to be performed on the raw motion data.

The list of tasks provided by this tool contains:

Figure 3.7.: The pipeline for Metropolis project characters animated using NaturalMotion Morpheme software, illustrating the large number of different editing steps and file formats required in a typical pipeline.

- motion format conversion (loading and saving in most of the motion formats listed below),
- animation cutting (by specifying the first and the last frame of the animation),
- FPS change (resampling the original animation data; if the motion was converted to be periodic beforehand, the adjustments keep the frame continuity),
- skeleton standardisation (changing the skeletal structure by projecting its hierarchy to a hierarchy specified by a configuration file and adapting the motion using interpolation / joint merging to reflect the new structure),
- bodypart cutting and appending from a different source file (exchanging a bodypart with separately captured data – necessary for detailed hands or facial capture),
- skeleton fixing (the manipulations based on constraint detection, as described in Section 4.4.1),
- motion periodisation (see Sections 5.2 and 5.3),
- trajectory bone extraction and on-spot conversion (see Section 5.3),
- locomotion trajectory editing (curvature changes with inverse kinematics for enforcing the constraints).

This tool has proven to be of great help for simple everyday motion capture editing tasks.

Motion Data File Formats

The field of computer animation requires a large amount of work to be performed manually. This originates in the artistic background of the field – every single aspect of the final result has to be under the direct control of the artist, and often each of these aspects has its own set of specific tools, algorithms and approaches. Consequently, the field contains a large number of specialised programs aimed at specific tasks in the pipeline.

This is also the case for our animation pipeline. In a typical pipeline used in the Metropolis project (see Figure 3.7), a motion clip is:

- captured and processed in the Vicon IQ software,
- altered using our motion editing tool, creating periodic animations and filtering the constraints,
- imported into 3DS MAX, converted into 3DS native format and applied to a character,
- exported into the NaturalMotion file format and used in the Morpheme tool for building a Move Tree,
- exported into the native Morpheme Runtime format, which then gets imported into the runtime system,
- and finally the animation is extracted from the Morpheme runtime module and applied to a character.

Even though this example shows the highest number of steps taken in practice in our pipeline, it demonstrates the large number of different programs and file formats required to bring a motion clip from motion capture to the final product.

Unfortunately, there is no widely accepted motion file format that would create a common link between all these steps. Several attempts at standardisation have been made, such as the H-Anim human model description ("Web 3D Consortium", 2005) or Collada file format (Barnes, 2006), but none of them have been widely accepted by the 3D software producers (e.g., Autodesk developed

their own version of the Collada file format, which is not compatible with the original standard). For this reason, our tools support many different file formats (listed below), thereby allowing the data import/export from and to most common computer animation tools.

Skeleton File Formats

VSK (Vicon Skeleton) file format stores the information about skeletal data in a XML-compatible format. Apart from the hierarchical information and the base pose, it describes the degrees of freedom for each joint (including the rotational axes), optical motion capture marker locations and their connections to the skeletal structure.

ASF (Acclaim Skeleton File) is a plain text format that describes skeletal data. Predating the XML specification, it uses an alternative section and subsection markup, capable of describing the skeletal hierarchy and base pose, a small subset of dynamic properties (bodypart weights, centre of mass location) and per-joint axis-based degrees of freedom with simple joint limits. It can represent multiple skeletons in one file.

XSF (XML-compatible Skeleton Format) is a variation of the ASF file format, with a specification compatible with XML. Even though the XML family of languages allows the hierarchy to be implicitly defined in the tag structure, this format uses a linear list of bones, with an attribute explicitly enumerating the IDs of all the children bones.

Motion Data File Formats

C3D is a generic non-proprietary binary file format. It is separated into a header and a set of containers, allowing a large volume and wide variety of data to be specified. The structure of the file enables the reading software to parse only the data it can interpret and skip over the rest. The original definition provides a set of standard header records and a description of several types of data blocks (allowing both skeletal and marker data to be stored).

V (Vicon motion file format) is a proprietary extension of the C3D format. Keeping the general structure of the original specification, it enables the reading software to skip over the extensions and read only the standard C3D motion data. The Vicon extensions include mainly several new types of header records, and an extension of the 3D trajectory data to include the level of accuracy for each sample (determined by the number of cameras used to reconstruct its value).

AMC (Acclaim Motion Capture) file format describes the motion data, and is intended to be paired with an ASF file describing the corresponding skeletal structure. The data description uses plain text and the markup is similar to the ASF file.

XAF (XML-compatible Animation Format) is, in a similar manner to the ASF/XSF relationship, an XML-based version of the AMC file format.

Combined Skeleton and Motion Data Formats

BVH (Biovision Hierarchy) is the most commonly used motion file format. It is in plain text, with a simple structure that is very easy to parse. It consists of two sections – the first section, denoted by the HIERARCHY keyword, describes a single skeleton structure. This structure provides the position of each bone relative to its parent (with a base orientation assumed to be the identity), its parent and children, and the number and type of all its degrees of freedom. The second section, denoted by the MOTION keyword, specifies the number of frames followed by the animation data. These data are organised into frames, with each frame providing one real number for one degree of freedom specified in the skeleton section. The main drawback of this

format is the combination of enforced identity orientation for base pose and the Euler angles for orientation representation, which can lead to gimbal lock and inaccurate data interpolation.

BIP (Autodesk Biped) is a proprietary binary format introduced in the Autodesk MotionBuilder software, later included in 3D Studio MAX. The animation data stored in this format also contain separated constraints and end-effector information, allowing a simplified retargeting of the animation data. Unfortunately, this format is not documented, making it usable only inside Autodesk products and impossible to support in our tools.

FBX (FiLMBOX) is an Autodesk Exchange file format, allowing any mesh, texture, skeleton or animation data to be stored in a compact binary representation. It was intended as an exchange format, allowing data transfer from one software product to another. Unfortunately, there are several incompatible versions of this format, and the publicly available specification is now obsolete and no longer used in Autodesk software products. This format is not supported by our tools.

OgreSkeleton was developed as the file format for the real-time Ogre graphics engine. It has two versions – a compressed binary representation and an XML-compatible one, with a file converter provided. Each file can contain only one skeleton and its filename is linked to a corresponding mesh file. The animation data are structured into a set of animation clips, each consisting of tracks describing an ordered set of keyframes and their time values. The animation data for each joint are then reconstructed using nearest-neighbour interpolation of the keyframes.

Collada is an XML-compatible file format developed by the Khronos group as an universal data exchange file format (Barnes, 2006). It is designed to be capable of containing any 3D scene data, with the skeletal animation of multiple characters being just a small subset of its full capabilities. Unfortunately, the flexibility of this format makes it also very complex, and even though an open-source software development kit (SDK) is provided, its adoption by the 3D industry is lacking. Moreover, Autodesk software uses their own incompatible version of this format, making the use of the official SDK impossible.

Marker Data File Formats

CSM (Character Studio Motion capture format) is a plain text file format describing the raw motion capture data of a marker-based optical motion capture system. It enables the calibration data of the character (distances of certain joints measured in a special pose), the animation properties and the marker data, with a special markup for missing marker frames to be stored. The animation data are organised in frames, with 3 real values for each marker.

C3D/V are binary file formats capable of describing both skeletal and marker data. The marker data section of this file is organized into frames and allows missing data to be described using a special value.

MNM is a specialized file format allowing markers in a CSM file to be renamed to match the internal description of the software. It is used by Autodesk 3DS MAX and Autodesk Character Studio, allowing them to import data from sources that were not directly intended to be used with these programs.

4

Footstep Constraints

Constraints are explicitly represented, time-varying properties of a motion clip. Often, instead of describing an existing property of the clip, they represent a desired property, and serve as an input to motion editing or motion synthesis algorithms (see Section 2.3.3). In our work they are used as both – they are first detected in the input motion clips, and then enforced to remove noise and artifacts present in the original motions. This chapter deals specifically with these two problems in relation to footsteps.

In the case of locomotion, footstep constraints represent the information about a foot (or part of it) in contact with the ground. They are an important feature of a walking pattern, and provide information necessary for motion retargetting, which is performed when a recorded motion is adapted to a character with different body shape than the original actor. We have demonstrated the necessity of correct footstep constraint handling in our perceptual experiment (see Section 6.3).

In the motion data obtained from motion capture, the footstep constraints are not represented explicitly (with the exception of motion capture with force platforms (Hreljac & Marshall, 2000)). The implicit footstep information, however, is often heavily influenced by noise. There are many different noise sources influencing the optical motion capture setup:

- measurement noise (caused by camera noise, camera resolution, occlusions and camera switching),

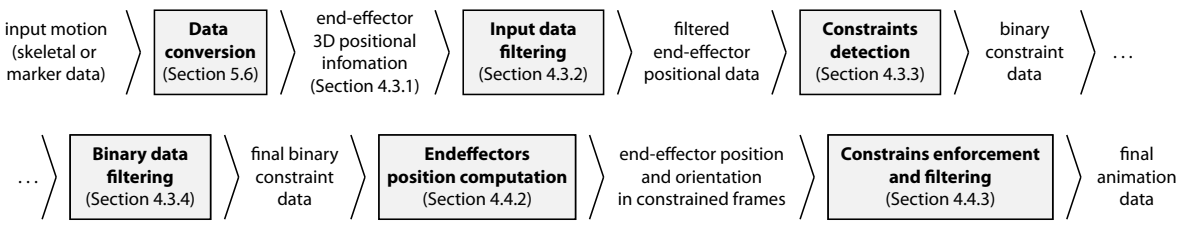

Figure 4.1.: Footstep enforcement pipeline.

Figure 4.2.: The foot bones and their representation in our optical motion capture. Left – the relative location of the foot markers and its virtual bones; middle – an actual skeletal structure of the foot, with lower leg bones (black), tarsus (purple), metatarsal bones (green) and phalanges (red); right – side view demonstrating the implicit representation of the heel in the skeletal data.

- environment noise (light pollution, uneven or non-rigid floor),
- noise based on the physical properties of an actor's body and its model (non-rigidity of bodyparts, shoe compression, dynamic skin and tissue movement),
- skeleton fitting artifacts,
- systematic noise (instrument miscalibration).

In this section, we describe a set of algorithms specifically designed for detecting and cleaning footstep constraints for motion capture data. While our algorithms can be generalised to other constraint types, they are designed under the strong assumption that a constraint in motion data is represented either by an end-effector static in space, or rotating around a point or axis. While this assumption is correct for feet during a locomotion (and most other natural motions), it does not apply generally to other end-effectors of the human body.

Our analysis of time-related properties of locomotion is restricted to normal walking, with speeds in range approximately from $0.5m/s$ to $2.5m/s$. Outside this interval, the features of locomotion change significantly and so does the footstep pattern – below $0.5m/s$, the walk changes into a shifting motion, with long periods of both feet planted on the ground and almost no pelvis movement; above $2.5m/s$, walking changes into jog or run. While both cases can be handled by our algorithms with minimal changes (different threshold values), they are not specifically discussed in this chapter.

4.1. Feet Motion Capture

The human foot is the terminal part of the leg which bears the weight of the body and allows locomotion. Its mechanical structure contains more than 26 bones and 33 joints, connected with more than 100 muscles, tendons and ligaments (Kahle et al., 1986).

From the motion point of view, there are four main segments of the human foot (in this paragraph, we will be referring to Figure 4.2, middle). Tibia and fibula (black) are the two bones of the lower leg connecting the knee with the ankle structure. The tibia is the strongest weight bearing bone of the body and also the second largest; the fibula is the smaller of the two, with an articulating function. They both form the ankle joint by their connection to the tarsus (purple), which is composed of the ankle bone, heel bone and five irregular bones that create the arches of the foot, serving as a shock absorber. Five metatarsal bones (green) form the middle of the foot and, finally, the toes are composed of phalanges (two for the big toe and three for other toes). Altogether, the foot bones and muscles form a complex articulated structure with many degrees of freedom, allowing flexibility, durability, shock absorption and detailed articulation.

For the purposes of motion capture, we simplify this structure significantly (Figure 4.2, left and right). The main reason for this simplification is that such a high level of detail is not necessary to produce believable motion, and most of our virtual characters would be wearing shoes, thus limiting

the range of possible subtle movements significantly. Therefore, we represent the lower leg using a single bone, connected by a 3 DOF ankle joint with a rigid structure representing both tarsus and metatarsal bones (with the heel not explicitly included; see Figure 4.2, right), which is further linked with a 1 DOF toe joint, connecting the metatarsal structure with a single toe bone formed by all phalanges. This simple structure does not describe many features of the foot realistically (e.g., the possible independent motion of different toes, the dynamic properties of the tarsus), but it is an approximation of the human foot suitable for our purposes.

In our optical motion capture pipeline, the movement of each foot is captured using 6 markers (see Figure 4.3). Five of these markers are fixed to the actor's shoes (a very flexible pair of canvas shoes with thin rubber base), with the last marker being attached from the outside to the ankle. We use markers with 12mm diameter (Figure 3.1, left, second largest marker), as they provide enough accuracy without interfering with the actor's movement. The redundancy of the marker information allows us to reconstruct the foot movement even when several markers are occluded simultaneously.

The marker information is transformed into skeletal data using the motion capture processing software (Vicon IQ 2.5) as a part of the full-body skeletal fitting process. As many of the features are not represented in this simple model (e.g., the flexibility of the foot arches, non-rigidity of the foot and leg bones), the final motion contains a certain amount of noise, caused inherently by the model, due to the fitting method used (see Figure 4.4) and by the optical motion capture process itself.

4.2. Anatomy of a Footstep

The footstep can be generally divided into four distinct stages, determined by the combinations of 2 different foot constraints (Johansen, 2009; see Figure 4.5). The stages can be described as:

1. heel strike (heel strikes the ground and the foot rotates around the heel; for most normal walks, this stage is very short),
2. static foot (the foot is lying on the ground with no explicit motion),
3. constrained toe (the foot is in the process of lifting with all the force focused on the toe; toe bone is constrained and ankle is rotating around the toe joint),
4. lift-up (the foot is not in contact with the ground).

From this description, we can derive two different constraints:

- ankle (heel) constraint – static in stage 2, rotational in stages 1 and 3 (in further text, we describe the constraint in stage 3 as toe-only, even though, technically, both bones are constrained),
- toe constraint – static in stages 2 and 3.

Obviously, a more anatomically correct description of the footstep would be significantly more complex. However, for the purposes of footstep cleanup, our simple model provides a suitable level of abstraction.

For constraints detection and enforcement, the most important task is to use accurate start end end times, and guarantee the constraint continuity. This ensures that no artifacts are introduced in

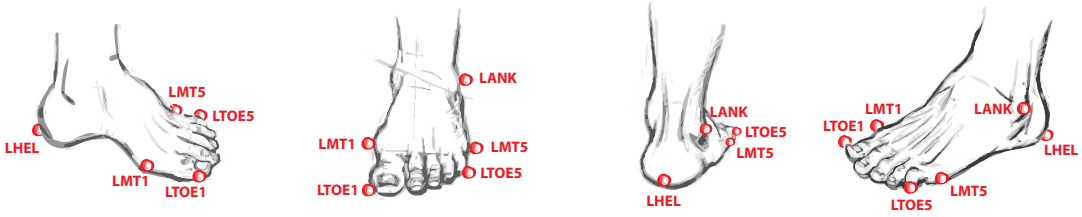

Figure 4.3.: Foot markers placement and naming in our capture pipeline.

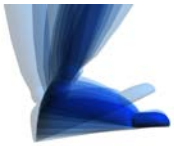

Figure 4.4.: Non-rigidity fitting artifact of skeletal motion capture data. The flexibility of human body on heel strike cannot be represented by a rigid skeleton, causing an artifact on the end-effector.

the constraint enforcing process (such as sticky or floating feet). Another important problem is to distinguish between stage 2 and 3 – if the transition point is shifted towards later, the heel sticks to the ground. As the leg is almost stretched at that moment already, enforcing such a constraint can either cause problems with knee-popping in the inverse kinematics stage (i.e., a discontinuity in the bending angle of the knee, based on non-linear relationship between the angle and foot position), or requires a more complex fix of the root position (Kovar et al., 2002b). An early termination of stage 2 can lead to a more subtle artifact which can be described as a floating heel.

4.3. Footstep Detection

The accurate detection of the time intervals during which constraints are active is a crucial step in the footstep cleanup process. Unfortunately, due to the complexity of human motion and the many different sources of both systematic and noise artifacts in the data, this is not a simple task. This section describes the tests of previous methods and introduces a new detection method, which is more robust than previous techniques.

The main requirement placed on the technique we are looking for is accuracy. Moreover, while we will demonstrate the footstep detection primarily on the skeletal data, it should be applicable to the marker data alone, thereby extending its use to raw motion capture data cleanup and fitting. This expectation mainly limits the input of the algorithm – the trajectories contain only the positional component and no bone rotation information is available. Furthermore, it requires the method to be flexible in dealing with several different artifact types – the skeletal data are influenced by additional artifact sources (e.g., skeletal fitting), but they are less likely to contain random noise present in marker data (the fitting process averages over several markers, reducing the measurement noise with Gaussian properties).

4.3.1. Input Data

The input to our constraint detection algorithm are 3D positional trajectories of the markers (joints) with respect to the world origin (global positions). In order to be able to handle both joint and marker data in a consistent manner, we do not use the orientation information present in the joint

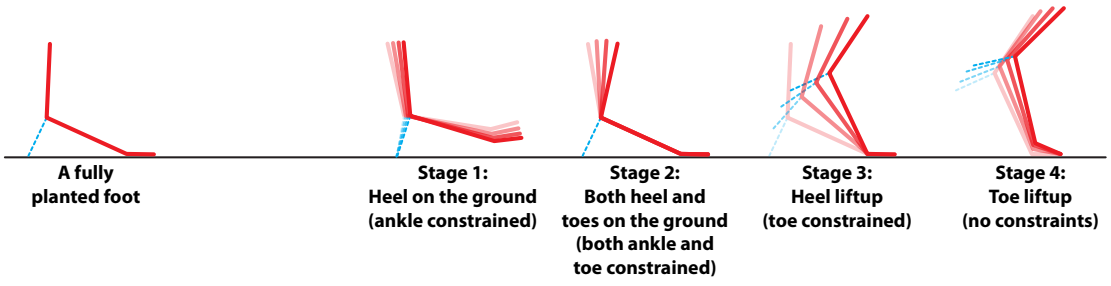

Figure 4.5.: The description of different stages of a footstep.

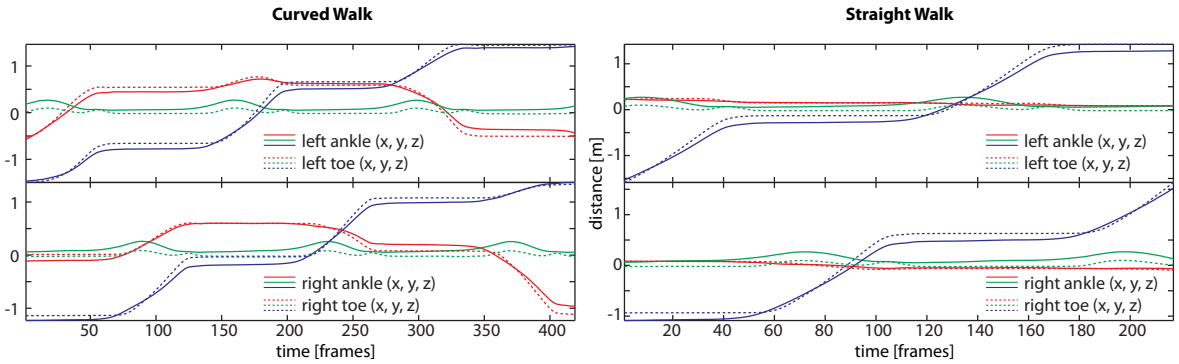

Figure 4.6.: XYZ plot of joint trajectories during locomotion. The flat regions represent a foot lying on the ground.

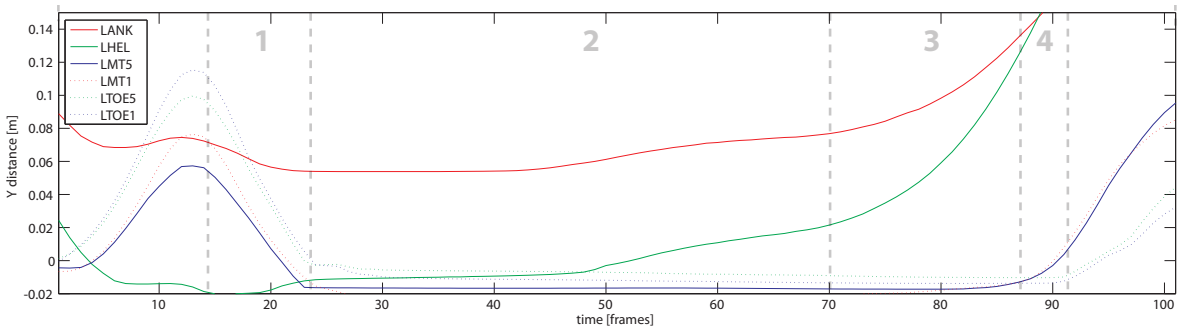

Figure 4.7.: The y (up) component of the foot marker trajectory data during a footstep. The numbered regions represent manually annotated footstep stages.

representation.

The *axis-separated trajectory data* for a straight and curved walk are presented in Figure 4.6 (only joints presented for clarity, marker trajectories are similar; see Figure 4.9). This graph shows the distinct flat regions, where a position of a joint does not change in time. However, it also shows that these regions are not precisely determined, having a smooth lead-in and lead-out transition. This poses a significant problem for constraint detection.

The problem can be also illustrated on a graph with the separate Y trajectories of the markers, manually labelled with the start and end of each footstep stage (Figure 4.7). While the transition between stage 1 and 2 is defined by an edge in several markers (LMT5 and LMT1), all the other transitions are present somewhere in a relatively smooth region.

The separation of x-z and y data (horizontal and vertical) is a common manipulation of the trajectory data, often used for footstep constraint detection purposes (Hreljac & Marshall, 2000). It is a very intuitive description, as the y data represent the distance between the marker (joint) and the constraint object (floor; assuming a flat floor surface) and the x-z part describes the forward and side trajectory (see Figure 4.8).

4.3.2. Input Data Filtering

A motion capture system is a physical measuring device and as such, its output data contain a certain amount of noise. As an electronic system with discrete outputs, a large portion of its noise levels can be modelled using the standard additive white Gaussian noise model with limited bandwidth. Therefore, the spectrum of the original signal will be flat on the logarithmic scale, up to the sampling frequency.

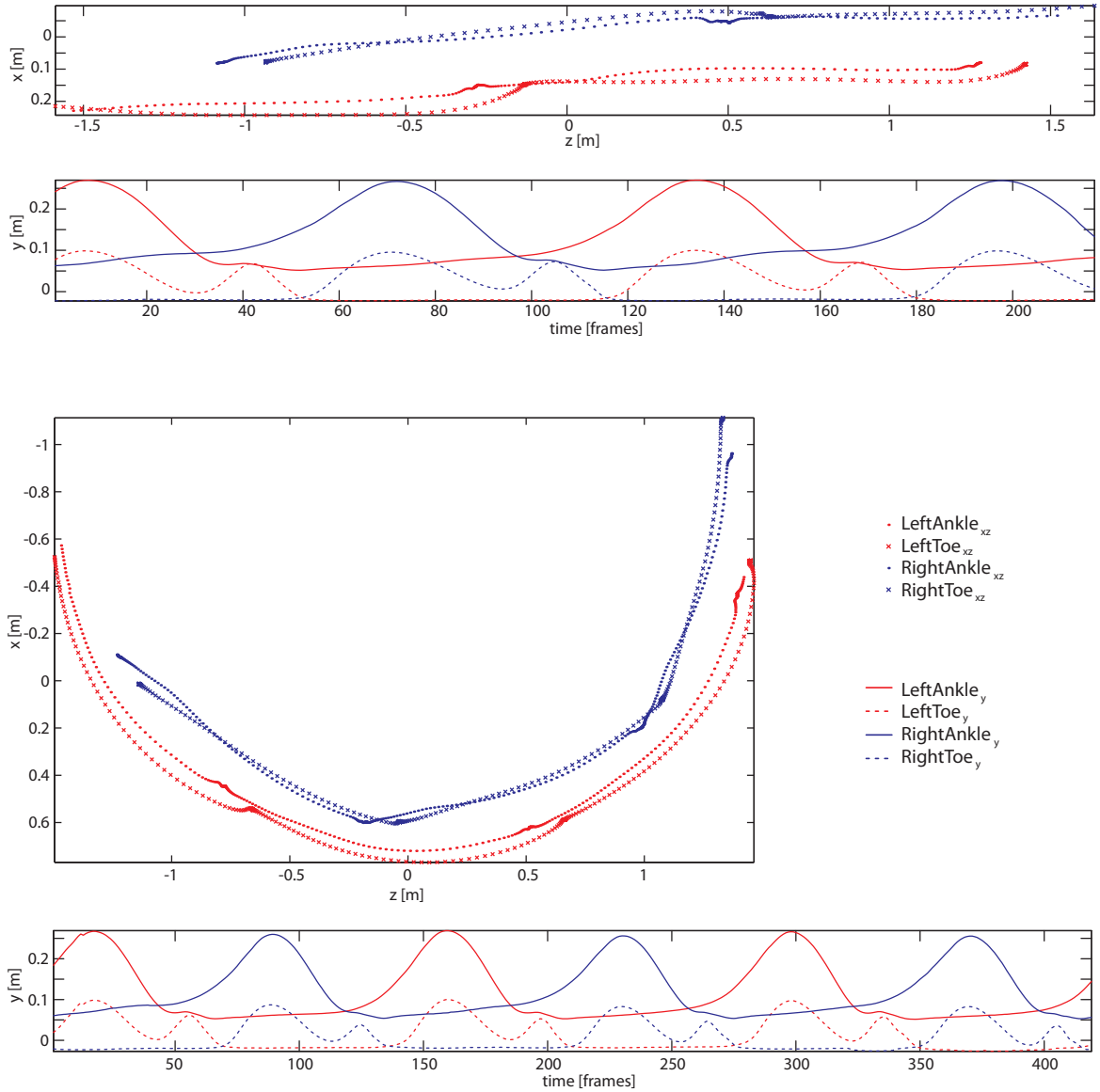

Figure 4.8.: The x-z and y axis separation of joint data for two different locomotions (top – a straight walk, bottom – a turning motion).

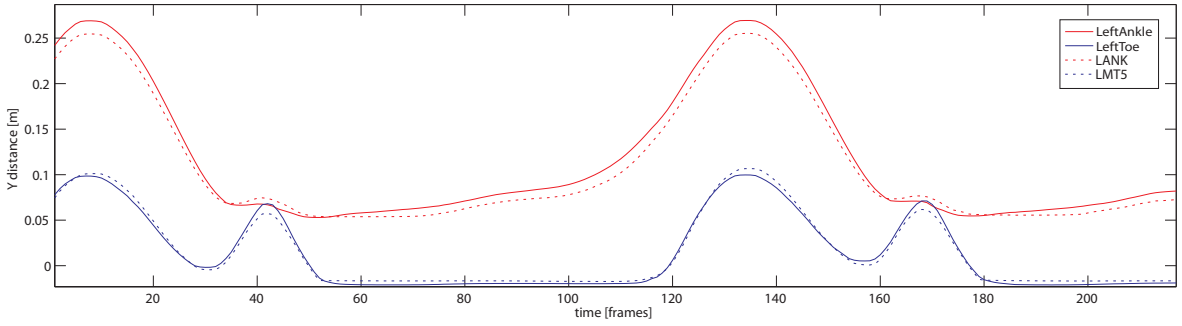

Figure 4.9.: A comparison between the y axis data of the joints and closest markers. While both descriptions show very similar trends, the marker data contain more noise. On the other hand, the averaging property of joint fitting can oversmoothen certain properties usable for constraint detection.

The differential of this signal, however, will have its noise spectrum shifted towards higher frequencies (which can be proven by converting the noise signal to the frequency domain and computing its differential there). This effect can be demonstrated on any of the trajectories captured by our motion capture system (see Figure 4.10).

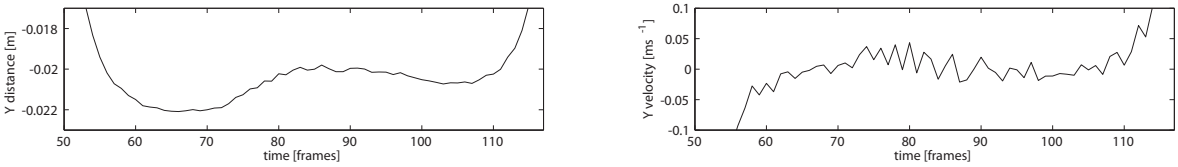

Figure 4.10.: An illustration of the input signal differentiation causing an increase in the noise levels (left – y ankle position during a footstep, right – y ankle velocity).

In Section 4.3.1, we have described the properties of the input signal, stating that a constraint is represented in the input data by a flat region. A simple method to detect this flat region is by using a threshold of the first derivative of the signal (the marker velocity). This is a standard method for constraint detection used in previous work (Lee et al., 2002; Ménardais et al., 2004). To detect the start and end of a constraint, several approaches used even higher derivatives, such as the zero crossing of the acceleration data (Bindiganavale & Badler, 1998), or the zero crossing of the first derivative of acceleration data (jerk; Hreljac & Marshall, 2000). However, with each higher derivative, the amount of noise in the input signal is significantly amplified. In our experiments, even acceleration data exhibited levels of noise that excludes them from practical use. For this reason, it is necessary to filter the input data.

Our target is to determine the start and end of a constraint. However, such information has usually a pulse nature, which moves it to a high frequency part of the signal spectrum – a naive way of signal filtering, which addresses noise reduction by removing the high frequencies, would smooth out this information. Therefore, in the following paragraphs we attempt to find a way to filter the motion data, which removes the noise while keeping the details about footstep events intact.

Gaussian Filter

The Gaussian function is the most commonly smoothing filter in signal processing. It is infinite, its pulse response is another Gaussian and it has a well known bell shape. Its two parameters are the centre (expected value) μ and the standard deviation σ:

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4.1}$$

For the case of signal filtering, the Gaussian function is centred ($\mu = 0$), leaving only the standard deviation parameter σ.

The input data of the filtering process are discrete, with each sample corresponding to the information of one animation frame. For this reason, in all the following equation we will use integer variable i to describe the sample (frame) index, instead of the continuous free variable x used in Equation 4.1:

$$i \in (1..n)$$

with n denoting the total number of input samples (frames).

The filtering process is performed using the discrete convolution of the signal function f with a smoothing kernel g created using the equation above. The resulting discrete function $(f \star g)$ is then

defined as:

$$(f \star g)(i) = \sum_{j=-\infty}^{\infty} f(i+j)\, g(i) \tag{4.2}$$

with values outside the range of the input signal usually defined as zero (or as a mirror of the original signal).

From a certain amount of input signal data, the convolution can be significantly sped-up by transforming both the input signal and the kernel into the frequency domain (using a Fast Fourier Transformation, FFT), thus converting the convolution into simple multiplication. However, this assumes that the input signal is periodic and the number of samples is a power of two (both issues can be addressed by padding).

For a simple non-Fourier implementation of this filter, it is possible to limit its kernel size to the interval $(-3\sigma...3\sigma)$, as the integral of the Gaussian function in this range covers more than 99% of its total energy. Therefore, we can define a variable s using the ceil rounding function (denoted by $\lceil\,\rceil$) as:

$$s = \lceil 3\sigma \rceil$$

and the surrounding set $\Omega'(i)$ of a sample i as:

$$\Omega'(i) = (-s \,...\, s)$$

together with a value w denoting the cardinality of this set (and the width of the convolution window):

$$w = 2s + 1 \tag{4.3}$$

However, the convolution as defined in Equation 4.2 provides smaller values around the signal edges, as all the data outside the definition range are assumed to be zero (or distorts them when the values are mirrored). Therefore, for practical reasons, we define a sample index dependent surrounding set $\Omega(i)$:

$$\Omega(i) = (max(1, i-s) - i \,...\, min(n, i+s) - i) \tag{4.4}$$

and use a normalised version of the convolution:

$$\begin{aligned} (f \star g)(i) &= \frac{1}{k(i)} \sum_{j \in \Omega(i)} f(i+j)\, g(j) \\ k(i) &= \sum_{j \in \Omega(i)} g(j) \end{aligned}$$

where $k(i)$ is the normalisation term of the convolution value.

By applying the convolution, we receive the filtered result in Figure 4.11. While addressing the noise successfully, Gaussian filtering also influences the leading and trailing edges of the constraint significantly. Moreover, its results are highly dependent on the smoothing parameter σ.

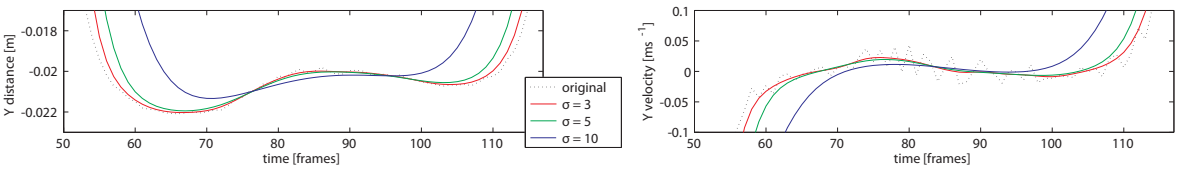

Figure 4.11.: Result of filtering the y ankle signal from Figure 4.10 using a discrete Gaussian filter.

Median Filter

The median filter is a non-linear method aimed particularly at noise removal. It replaces the current value of the discrete signal by the median of its neighbours, by selecting the median value from a window of samples centred at the current position.

The practical implementation processes the set $\Omega(i)$ surrounding a sample (frame) i (Equation 4.4) by sorting them according to their value and selecting the sample in the middle of the sorted array. The only parameter of the filter is the width of the processing window w (see Equation 4.3).

The resulting function (see Figure 4.12) is smoothed to a certain degree, with all significant peaks and changes removed. The lead-in and lead-out slopes, however, are perfectly preserved. Unfortunately, the changes in the first derivative of the function are minimal, suggesting that this filtration has to be performed on each derivative separately.

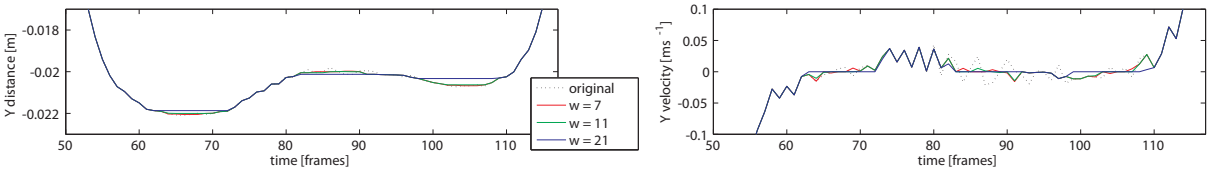

Figure 4.12.: Result of filtering the y ankle signal from Figure 4.10 using a median filter.

Bilateral Filter

Bilateral filter is a non-linear filter consisting of two Gaussian kernels, one applied to sample distances and the other to sample values. It uses a modified (normalised) version of the convolution algorithm and the combination of two kernels, both of which do not allow any processing speed-up in the frequency domain (as in the original Gaussian filter). However, compared to the Gaussian filter, it has an important property of edge preservation.

Using the notation established for the Gaussian function, the bilaterally filtered function $h(i)$ for a sample i is defined as

$$\begin{aligned} h(i) &= \frac{1}{k(i)} \sum_{j \in \Omega} f(i)\, g_s(i-j)\, g_v(f(i) - f(j)) \\ k(i) &= \sum_{j \in \Omega} g_s(i-j)\, g_v(f(i) - f(j)) \end{aligned}$$

where Ω is the surrounding of sample i (Equation 4.4), $k(i)$ is the normalisation coefficient for sample i, $g_s(x)$ is the spatial Gaussian function, $g_v(x)$ is the value smoothing Gaussian function and $f(x)$ is the input function. The final filtering function therefore has two parameters – the spatial standard deviation σ_s and the value standard deviation σ_v.

The filtered result can be found on Figure 4.13. The noise smoothing properties of this filter are similar to the Gaussian filter, but the major edges are preserved significantly better in both the original function and its first derivative. Furthermore, the impact of the two configuration parameters is relatively intuitive and the final result is not overly sensitive to their values.

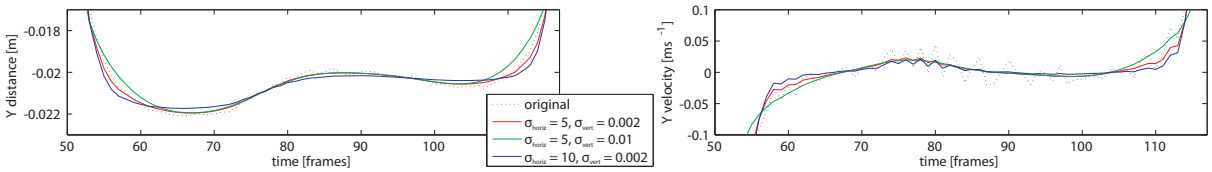

Figure 4.13.: Result of filtering the y ankle signal from Figure 4.10 using a bilateral filter.

Discussion

In previous paragraphs we have discussed the use of several classical examples of filtering methods of signal noise cleanup on motion data. The main aim is to reduce the levels of noise in the signal, which mostly influences the high frequencies of the motion, without compromising the relevant constraint cues. Unfortunately, some of these cues are contained in the high frequency information.

The properties of the described techniques can be summarised as following:

- The Gaussian filter significantly reduces noise levels, both in the original signal and in its derivatives. Unfortunately, it also negatively influences the larger edges in the data, thus interfering with important properties required for the constraint detection.
- The Median filter successfully reduces the noise levels in the original signal while preserving important signal cues, but it does not reduce the noise in higher derivatives and furthermore influences the signal in a non-linear and non-smooth manner (removing the outliers). This can introduce new artifacts to the result that can be hard to predict.
- The bilateral filter preserves the signal edges while successfully removing the noise both in the original signal and in its derivatives. It includes two configuration parameters, but the final output is not overly sensitive to its values.

Comparing these properties, we conclude that the best method for our purposes is the bilateral filter.

Our evaluation did not include several important filter types used in practice. A number of Infinite and Finite Impulse Response filters (IIR and FIR) can be used to filter discrete data. However, both types introduce a significant phase shift of the signal, interfering with constraint information. Moreover, generally, their response is similar to linear filters, represented in our short list by the Gaussian filter.

From the large group of linear filters, we have included only simple Gaussian filter. Other filters, such as the Butterworth, Chebyshev or Elliptic are used in practice (with the Butterworth filter specifically uses by the Vicon IQ motion processing software), providing different frequency and phase characteristics. However, the general property of influencing certain frequencies without the distinction between the noise and signal data is common in all these methods.

Finally, our evaluation did not include any state-based filters (e.g., Kalman filter) or more complex statistical or simulation approaches, as our target is to provide a simple and generic preprocessing technique with predictable results. However, it is possible that some of the more advanced techniques would provide better pre-filtering properties.

4.3.3. Detection Methods

The following section focuses on the properties of different footstep constraints detection methods. As discussed earlier, we are focusing on footsteps only, and the tested methods are specifically designed as such. Moreover, our methods are designed to function on both marker and joint data, thus excluding any possible orientation information that might be available for joint data.

In the following text we describe several methods from previous work, which do not require the

use of high order derivatives of the source data. The main reason for this limitation is the fact that the data from our motion capture system have proven to be relatively noisy, with higher differentials unusable for constraint detection (even when filtered in a preprocessing step). Therefore, we start with the simple positional methods, continue through first derivation thresholding and finish with a simple method of our own, based on rotation axis detection inspired by the work of Le Callennec & Boulic (2006).

Each method is described together with its characteristic properties, and with an informal analysis of suitability for locomotion foot constraint detection. Unfortunately, apart from the measurable physical quantities discussed in this chapter (e.g., well-defined threshold value, noise invariance, continuity), a direct evaluation of constrain detection methods is problematic. This is caused primarily by the fact that motion capture data do not represent the constraint values explicitly, and humans are not capable of identifying them directly. However, an indirect evaluation of footstep constraints is provided in Section 6.3.

Y Position Thresholding

Assuming a standard studio motion capture setup for simple locomotion, the environment description is trivial – a floor plane positioned at the origin, with the normal pointing up. The distance of a joint from this environment can be computed simply by taking the y axis of the position at given point of trajectory. This is essentially an implementation of the environment-distance idea described by Bindiganavale & Badler (1998).

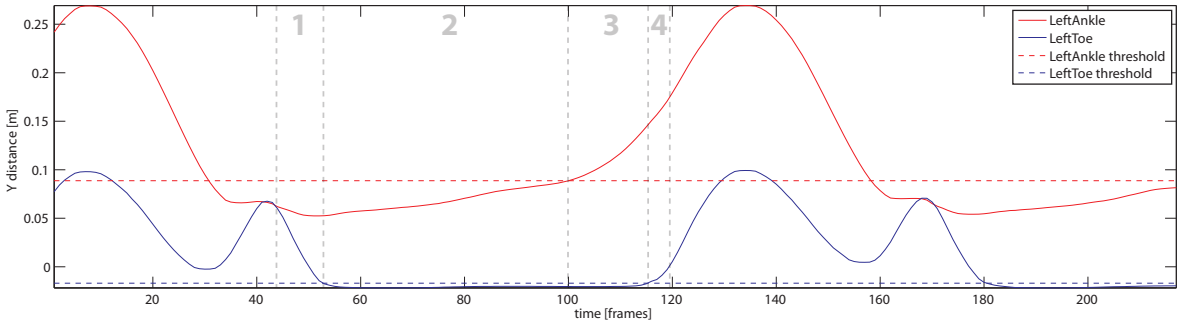

Figure 4.14.: The illustration of the y axis thresholding constraint detection approach, with manually labeled stages of the footstep in light gray. The thresholds are set up manually to represent the expected constraints well.

An illustration of this approach can be found in Figure 4.14, which also demonstrates well the problem with the threshold setup – determining a threshold value applicable to our data is not trivial and depends heavily on the input. Moreover, simple thresholding is not effective for the ankle data, as it supplies no explicit features to distinguish between different stages of the motion (a solution would require a hysteresis thresholding, with one more configuration variable). While both problems can be addressed using a machine learning approach, the derived values would apply only to one type of motion, a single actor and one particular motion capture setup (recalibration of the system would change the values slightly, offsetting the threshold).

Moreover, this method is particularly sensitive to the calibration of the floor plane and the flatness of the floor. Even slight miscalibration of the normal vector, leading to a floor tilt of only one angular degree, would lead to Y difference of 9 centimetres in a capturing volume 5 meters long; a systematic error of a similar magnitude as the threshold value. Figure 4.14 shows a slight miscalibration of a similar type – the floor Y value is shifted approximately 2 centimetres below its actual position,

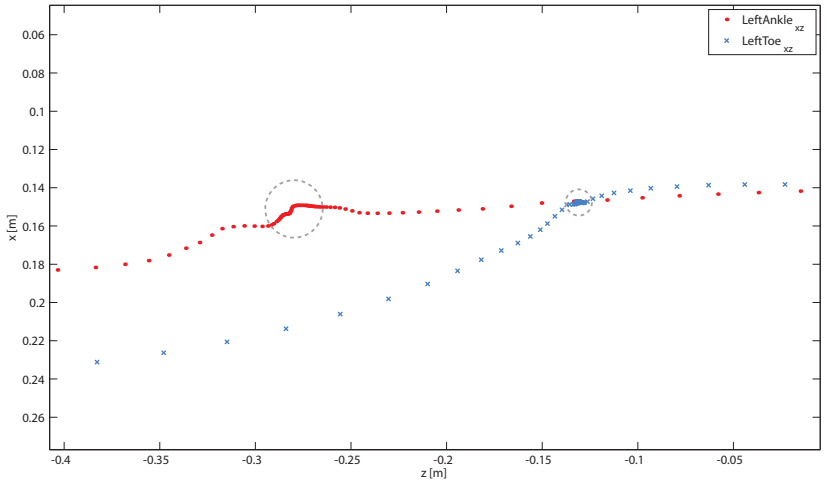

Figure 4.15.: A detailed view on a single footstep detected using XZ position thresholding. Each of the depicted points is a single sample in 120Hz motion capture data; the gray circles describe the maximal threshold diameters from the cluster's centre.

leading to values below the zero level.

XZ Position Thresholding

In the previous method, we have separated the Y data from the XYZ trajectory to allow separate processing of the distance from the floor. The remaining XZ data describe the movement parallel to the floor plane. During a foot constraint, this movement should be minimal, as the foot is in contact with the ground.

A second method, based directly on the end-effector positional data as well, can detect the centre of a cluster of samples and threshold the positional distance (see Figure 4.15).

This method performs relatively well and is significantly more generalisable (it does not depend on the floor calibration and the difference between actors is minimal). However, the heel lift-up stage is not distinguishable from the standing phase (stages 2 and 3), as the XZ movement during the lift-up is minimal. This problem could be addressed by combining with the previous method, but the number of different thresholding parameters would increase significantly, making it unpractical.

Speed Thresholding

A simple mechanism for detecting static segments of a trajectory is to determine its velocity (using the first derivative, in our case calculated from our discrete data) and then threshold its magnitude (speed). This is also a method used often in previous work (Lee et al., 2002; Ménardais et al., 2004). A graph of speed values for ankle and toe bones during a walking motion can be found in Figure 4.16.

However, computing the first derivative of the motion capture data also significantly enhances the signal noise (see Section 4.3.2). Rescaling the Y axis of the speed graph provides an insight into this problem (see Figure 4.17). Moreover, thresholding speed values for different walk speeds or styles can be a challenge, as the required threshold value can differ significantly (see Figure 4.18). Le Callennec & Boulic (2006) address this issue by implementing an adaptive threshold, but as its value is adjusted using previously determined constraints, it does not present a reliable estimate of its actual value (i.e., heavily relies on a perfect calibration and similarity between multiple footsteps).

XZ and Y Separation and Rotational Point Detection

Inspired by the work of Le Callennec & Boulic (2006), constraints can be detected from positional data by recovering a common rotational point or axis. Unfortunately, the original work was specifically aimed at detecting constraints in skeletal data, with mathematical background built on the analysis of differential transformation between two frames (both rotational and translational). We aim for a more flexible method, applicable to only positional information, which would generalise these methods to marker trajectories.

The feet contains several possible rotational axes, depending on the footstep phase (see Figure 4.5) – in the first phase, the ankle rotates around the heel point. The second phase is mostly static, with the lower leg bones rotating around the ankle bone. In the third phase, the ankle rotates around the toe point and in the fourth phase, the rotational axis is not detectable close to the foot (while the leg performs a rotational motion centred at the hip, the foot follows a parabolic trajectory, with its approximate rotational centre below the ground). From this simple analysis we can see that the stages can be recognised using the position of the rotational axis or its distance from one of the joints of the foot.

Another important observation used in this method is the fact that, ignoring the sample order, a circle fitted into a random pattern of points would have a diameter corresponding to their spread (see Figure 4.19, left). A different effect can be observed on noisy straight trajectories, where the fitted circle would have a large diameter.

To use this information for detecting footstep constraints, we can also use the fact that all rotational constraints of a foot during the walking motion have their axis approximately parallel with the ground. Following the separation introduced above, we can convert the joint (marker) trajectories into 2D by using the XZ component as the forward direction and the Y component as up. The XZ of each trajectory signal $f(i)$, separated into components as $f(i) = (f_x(i),\ 0,\ f_z(i))$, is then converted into scalar monotonous function $g(i)$ using the formula:

$$g(i) = g(i-1) + \sqrt{(f_x(i) - f_x(i-1))^2 + (f_z(i) - f_z(i-1))^2}$$

effectively differentiating the XZ component of the trajectory data, computing its magnitude and integrating it back into a function (with $g(0) = 0$). This leads to a 2D discrete function as shown on Figure 4.20.

A visual analysis of this data (Figure 4.21, left) reveals several obvious circular trajectories, with constrained frames determined by the small circle diameters. An objective analysis by fitting circles

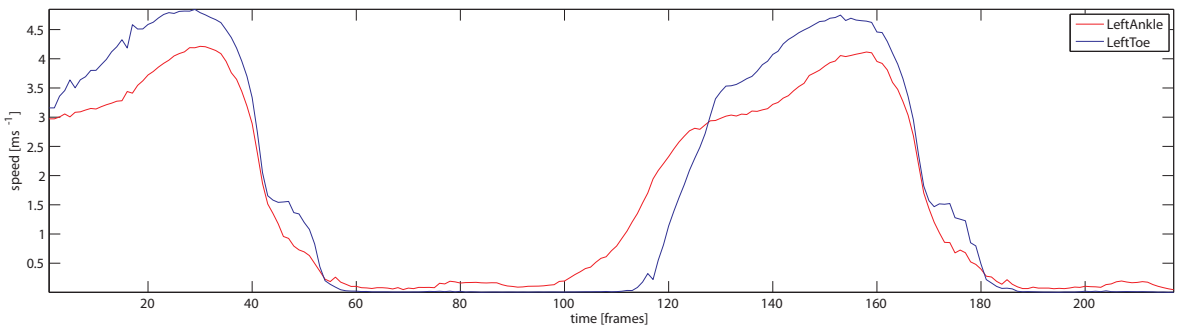

Figure 4.16.: Velocity magnitude (speed) graph of one foot during straight walking motion, with the noise reduced using the bilateral filter.

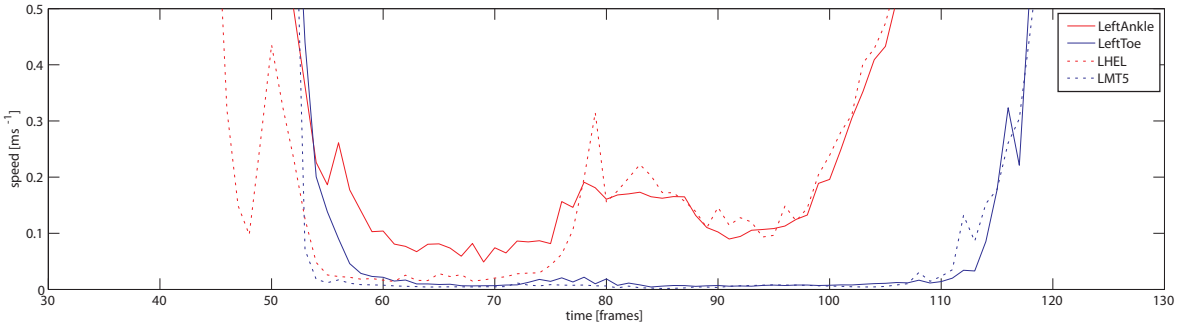

Figure 4.17.: Detail of a footstep in the speed graph from Figure 4.16. The noise in the first derivative of the trajectory data is prominent even after filtering the source data using a bilateral filter, with significantly higher levels in the marker data as compared to the joint data.

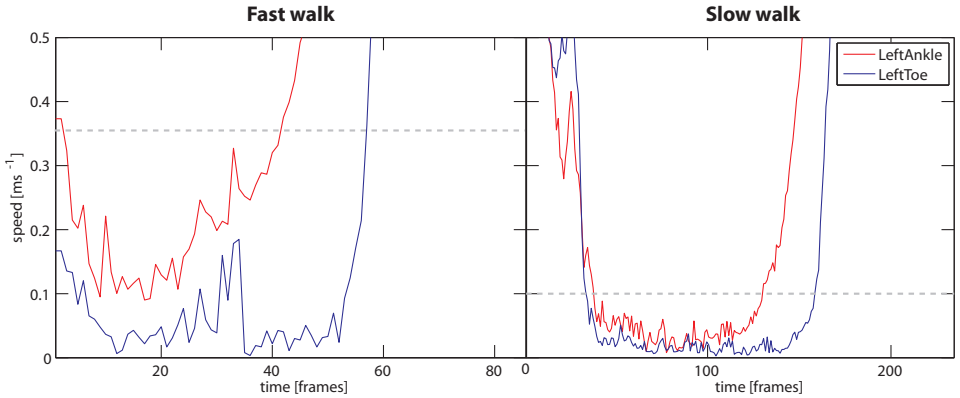

Figure 4.18.: A graph of joint speeds for different types of locomotion (unfiltered source data).

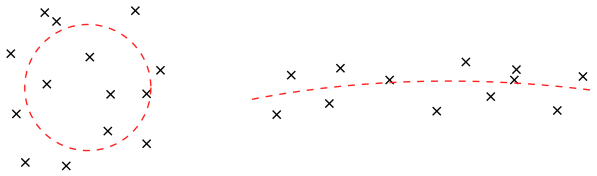

Figure 4.19.: An illustration of the circle fit into noisy data. Left – fitting a circle into noisy positional data of a static point leads to a circle dependent on the noise amplitude. Right – a circle fit into a noisy straight trajectory leads to a circle with large diameter.

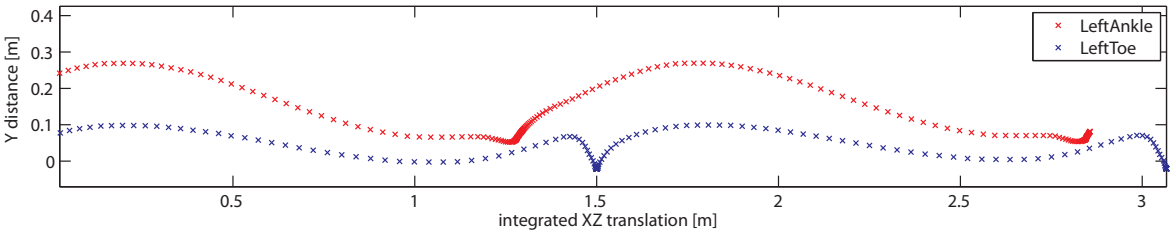

Figure 4.20.: The XZ-Y separation for circle detection. The XZ components of the trajectory data are integrated into a single scalar forward function, leading to a concise description of the end-effector height in relation to the overall travelled distance.

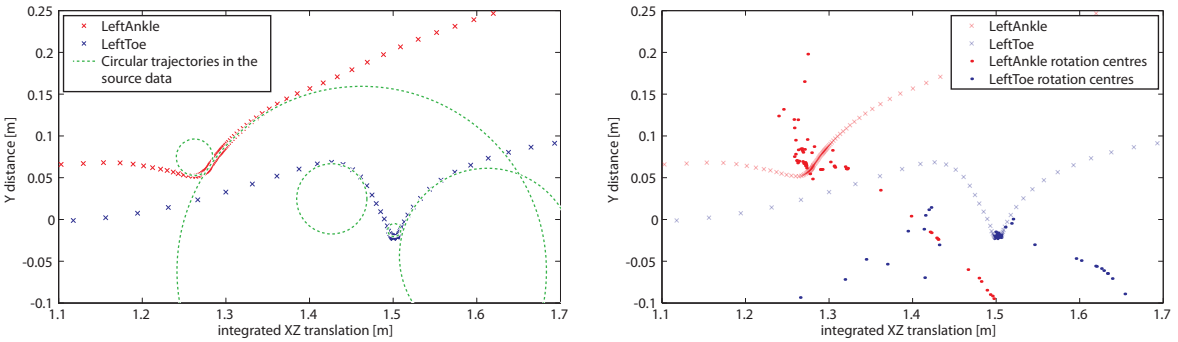

Figure 4.21.: Rotation centre analysis of the XZ-Y separated trajectory data as a constraint detection method. Left – illustration of circles present in the end-effector trajectories; right – the circle centres detected by fitting circles into windows of 7 samples.

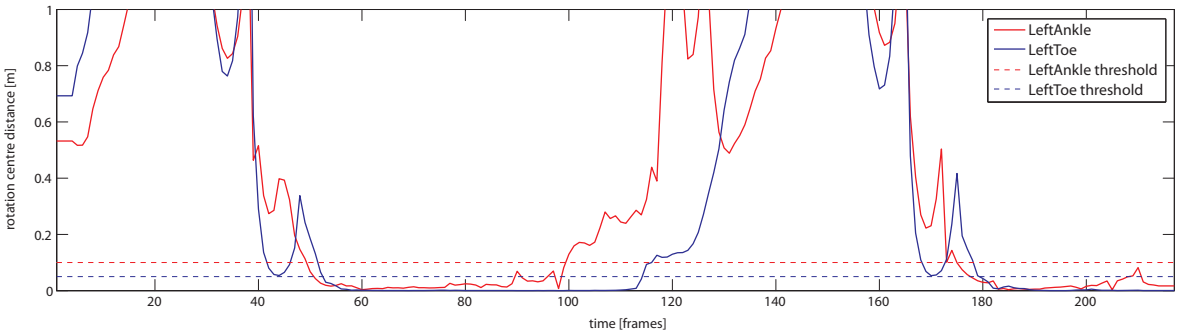

Figure 4.22.: The radii of detected circles in footstep data, based on windows of 7 samples.

into windows of data that are several frames wide (see Appendix C) roughly corresponds to the preliminary visual analysis (see Figure 4.21, right). To determine frames with small diameters, we can threshold the function of circle diameters as depending on the sample index (Figure 4.22). The thresholding values are determined intuitively – the expected maximal circle radius corresponds to the expected inaccuracy in the joint (marker) data, which is in turn determined by the size of the bodypart being analysed. For the case of ankle and toe in further examples, the threshold values are 10cm and 5cm respectively (Figure 4.22).

Apart from the threshold values, the second important parameter is the fitting window size. Using a small window leads to a large amount of noise in the data, but it also preserves the important edges (in a similar manner as filtering techniques; see Section 4.3.2). In previous work, Le Callennec

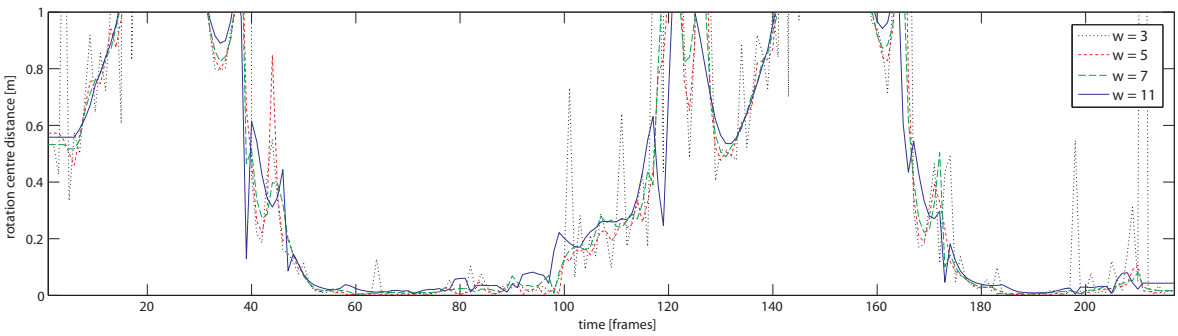

Figure 4.23.: The impact of different fitting window width on the detected circle diameters.

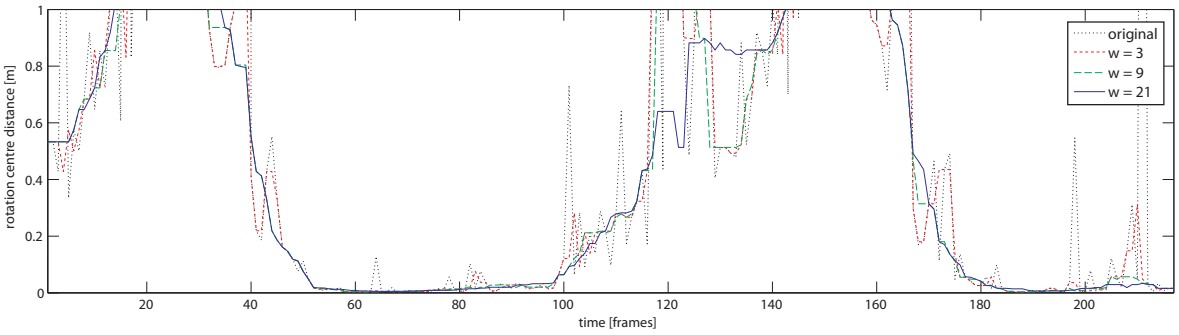

Figure 4.24.: The median filtering of the circle diameters (the source data are generated using a 3 samples wide window to simulate noisy data input).

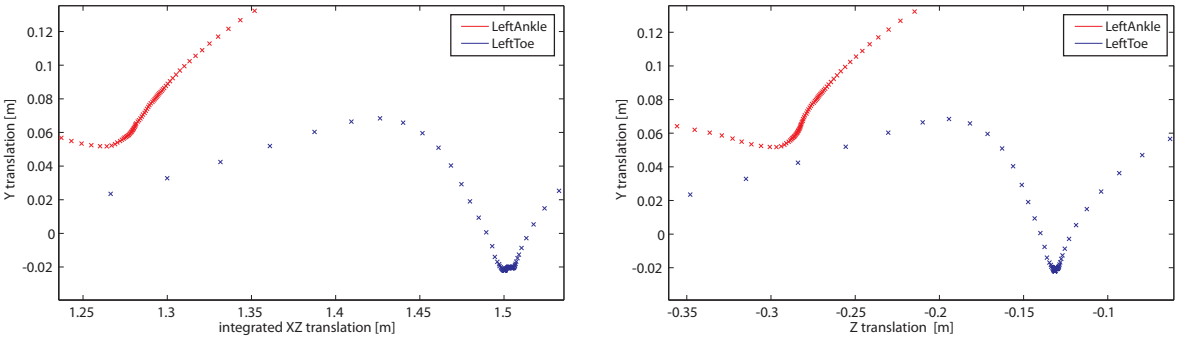

Figure 4.25.: The comparison of the xz-y separation (left) to the rotational axis detection (right). The latter method exhibits data with noise patterns closely corresponding to the preliminary assumptions.

& Boulic (2006) develop a complex noise prediction/filtering scheme to address this issue. However, their approach is limited to use only two consecutive frames of the motion. As we are using only positional data, our approach allows us to use multiple frames, thereby significantly reducing the amount of noise. However, even when using a large window, the method does not influence the significant edges significantly (see Figure 4.23 for a comparison of different fitting window sizes). In the practical implementation, the fitting frame window is always 7 samples wide.

Another important observation with our method is the fact that prefiltering the motion (using any of the methods described in Section 4.3.2) actually affect our results in a negative way, thereby introducing noise into the circle diameter function. This can be explained by the fact that the prefiltering method is not feature-aware and therefore destroys important information with the noise.

However, the result, exhibits a certain amount of spiky noise, especially for a small window diameter. A filter specifically designed to deal with this noise pattern is the median filter (see Figure 4.24), which can be as an optional post-processing step of the detection algorithm.

The main problem of our method is the XZ-Y separation, which assumes that the rotational axis is parallel to the ground plane. Moreover, the separation algorithm integrates the noise on the XZ axes, which leads to significantly larger forward motion than which was present in the original data, along with a certain level of bias in the detection results (see Figure 4.25).

Rotation Axis Method

The rotational axis detection method attempts to address the drawbacks of the technique introduced in the previous section. First, we want to be able to allow an arbitrary axis of rotation, as the rotational axes of feet are not precisely parallel with the ground, especially for turning motions. Second, the

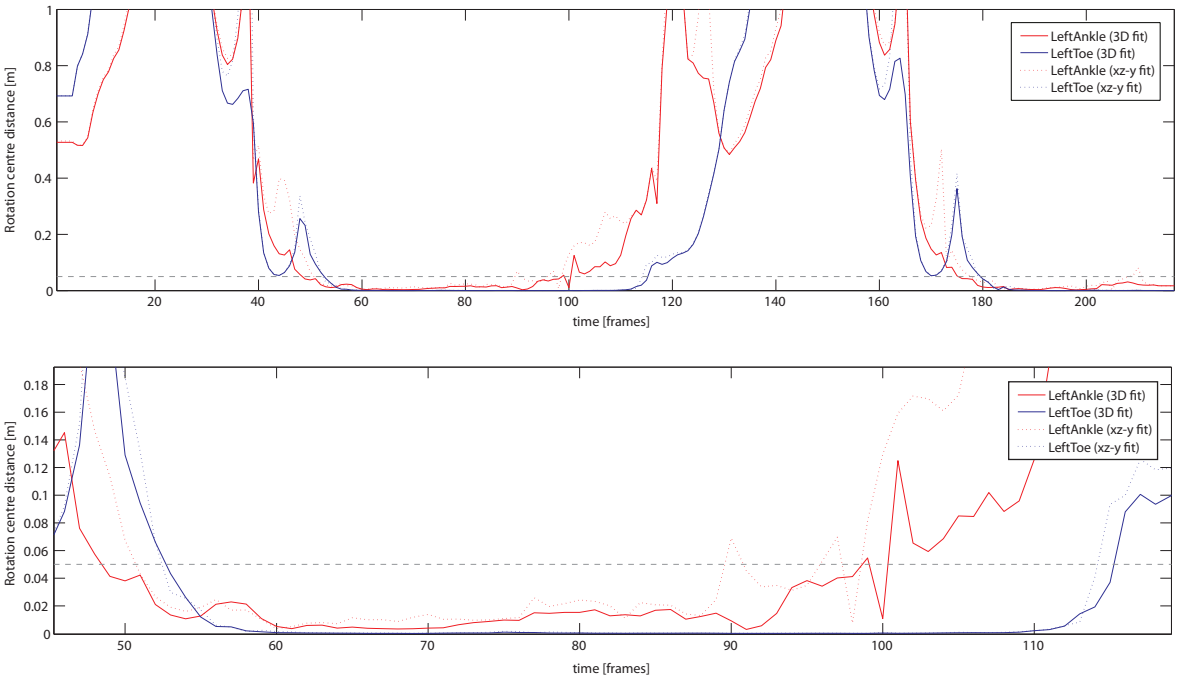

Figure 4.26.: A comparison between XZ-Y separation method and the rotation axis method. Due to the noise integration in the XZ-Y conversion and the assumption of the rotation axis parallel with the ground, the diameters detected using this method are significantly larger and contain higher levels of noise than the data generated using the rotation axis (3D fit) method. Both graphs were generated using a window of 7 frames.

XZ integration error (see Figure 4.25) is an aspect that introduces a certain amount of noise into the detection process.

As in the previous method, our inspiration comes from the work of Le Callennec & Boulic (2006) – we aim to determine the constrained frames by finding a common axis of rotation and then thresholding the maximum rotation radius. However, while the XZ-Y separation assumed that the axis of rotation is parallel with the ground plane, the rotational axis method determines the axis direction from the circular motion of the end-effector.

The algorithm consists of two steps. For every window of joint (marker) positions, we first fit a plane into the data using a standard least-squares fitting algorithm (see Appendix B). By projecting the marker positions to this plane, we obtain a 2D version of the data, which can then be used for circle fitting in the same way as the previous method.

While technically more generic than the previous method, the results are very similar (Figure 4.26 shows a comparison of the unfiltered results).

A simple extension applicable to this method, providing more data about the rotational axis change, is to determine the distance between detected rotational points (i.e., intersection between the rotational axis and the fitting plane) between each pair of frames (see Figure 4.27). This metric provides a function with distinct spikes at the point of the rotational axis transition, allowing a more robust detection of different stages of the constraints. However, a detailed analysis of this method is reserved for future work.

4.3.4. Filtering Output

The output result of each constraint detection method is a per-constraint binary discrete signal, with value 1 whenever a constraint condition is satisfied and 0 otherwise. However, independently of the

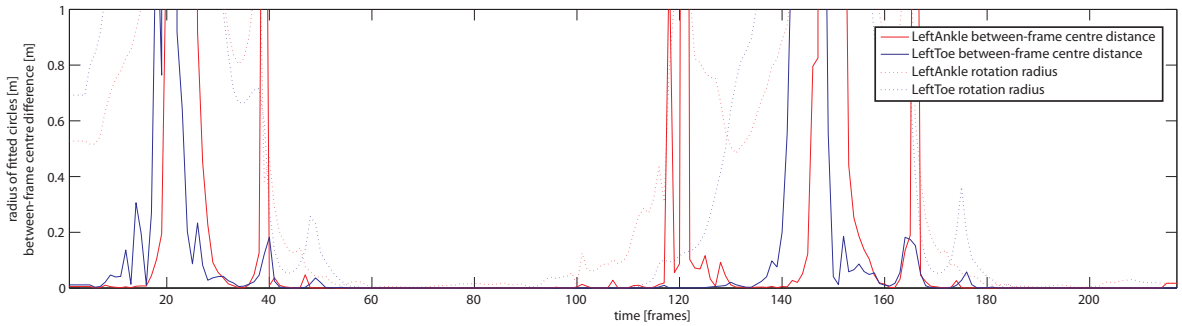

Figure 4.27.: An example of between-frame rotational centre distance function on footstep data. Notice the distinct spikes corresponding to the start and end of each footstep stage.

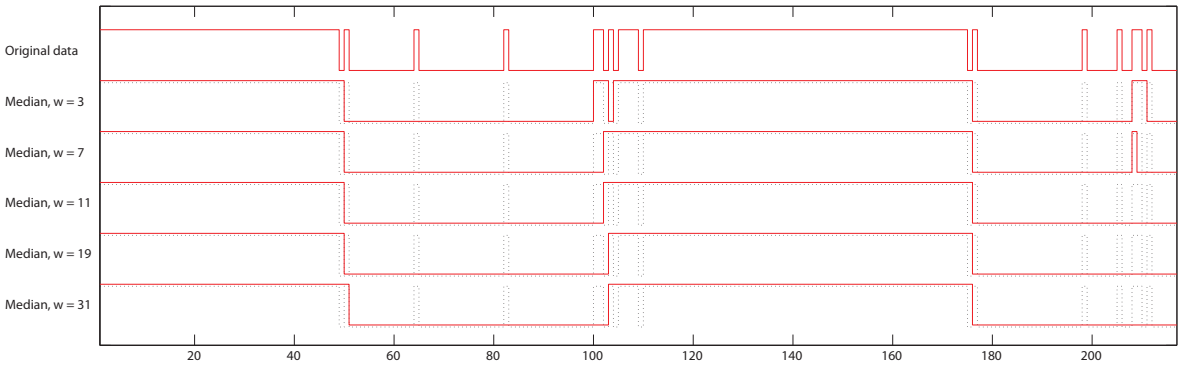

Figure 4.28.: An example of binary median filtering of the output constraint data with noise. The input data are generated using the same example data as in Figure 4.23, with a fitting width of 3 samples (generating a significantly higher levels of noise for demonstration purposes).

constraint detection method used, this signal contains a certain amount of noise, prominent especially at the beginning and end of the constrained interval. The per-frame detection methods, such as velocity thresholding or our rotation axis method introduced in Section 4.3.3, produce a specific spiky pattern (see Figure 4.28, top). Several statistical or state-based methods were used to address this problem in previous work (see Section 2.3.3), incorporating the temporal information into the per-frame constraint detection results. We aim at a simpler solution, applied as a post-processing step of the constraint detection algorithm.

The median filter is a classical non-linear noise reduction filter used in signal processing. We have already introduced its use on motion data in Section 4.3.2, where each discrete signal value was a real number. However, the nature of this filter enables its simple adaptation to binary data.

While it is possible to implement the binary version of median using a classical sorting algorithm (see Section 4.3.2), the binary nature of the data allows to simply count the number of samples with 1 and with 0 value in a window and return the one which is more common. The only parameter of this filter is the window width, but on our data we found that the result is not sensitive to its value. For practical implementation, we use a window, which is 11 samples wide (see Figure 4.28 for a comparison of different widths).

In the final step of constraint detection, we enforce the correct order of footstep stages. The necessity of this step is caused by the fact that in rare occasions, the noise accumulated in some of the source clips led to a wrong order, either missing the stage one in the footstep, or starting with stage three immediately followed by stage two (with the error state not longer than several milliseconds). To

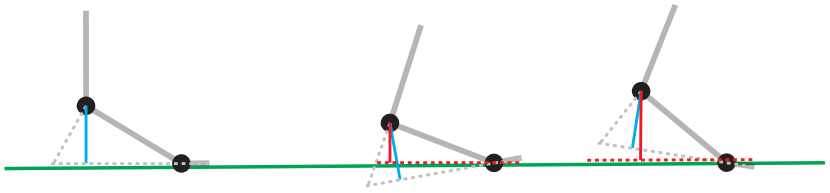

Figure 4.29.: The skeleton miscalibration artifact. A seemingly correct base pose (left) can lead to a consistent behaviour of the feet either below (middle) or above (right) ground during the footstep.

address this issue, we use a heuristic solution, which determines the stage two first and derives the other stages related to its time interval. This simple method ensures correct footstep handling in the footstep enforcement stage (see Section 4.4).

4.4. Footstep Constraints Enforcement

Given the binary function describing the expected time intervals for each constraint, a method is needed to ensure the fulfilment of the relevant constraint condition (e.g., a foot static on the ground). Moreover, as this requires an alteration of the original motion, the constraint enforcement method has to ensure that no artifacts are added into the final result.

The constraint enforcement has its primary use in two cases. First, as a post-processing step of motion capture pipeline, as no captured animation is perfect, both due to noise and motion reconstruction methods used. Second, as a step of a motion synthesis method – most data driven motion editing/synthesis methods alter the motion in a way that disrupts the original crisp constraints (e.g., by blending), and a reconstruction step is required to make the motion realistic.

In this section we provide a description of two methods used in our work. The first method aims to correct a systematic error caused by skeleton miscalibration (Section 4.4.1), which is a specific problem related to locomotion recorded using a motion capture technique with rigid body skeleton fitting, but also applies to motion retargeting. The second method describes a more generic method for footstep reconstruction, which consists of determining the idealised foot trajectories (Section 4.4.2) and altering the animation to follow them (Section 4.4.3).

Throughout this section, we will be using the classic matrix notation used in skeletal animation, which describes the local coordinate matrices of a bone in the binding pose as $\boldsymbol{R}_i$, its animated transformation as $\boldsymbol{T}_i$, the premultiplied transformation as $\boldsymbol{P}_i$, the global binding pose transformations as $\boldsymbol{A}_i$ and the global animated poses as $\boldsymbol{F}_i$, with i determining the bone index. For a detailed description of this notation and its mathematical background, please refer to Appendix A.

4.4.1. Skeleton Miscalibration Correction

A skeleton-based human animation relates the animation data to a starting pose of the skeleton, often referred to as *binding pose* (see Appendix A for more details). However, if this pose is not accurate, the resulting animation exhibits a certain amount of systematic errors. In the case of human locomotion, there are two important types of these errors. The first type of error leads to incorrect positions of the feet during a footplant, i.e., whole feet are either above or below the ground. This artifact can be addressed by translating or rotating the positional component of the root transformation. The second artifact exhibits itself only on the heels, i.e., during a footplant, the heel is consistently either above or below the ground, while the toe joint is planted correctly (see Figure 4.29). The focus of this section is aimed at the latter case.

There are several possible causes of this artifact. One is a different shape or height of the foot, when

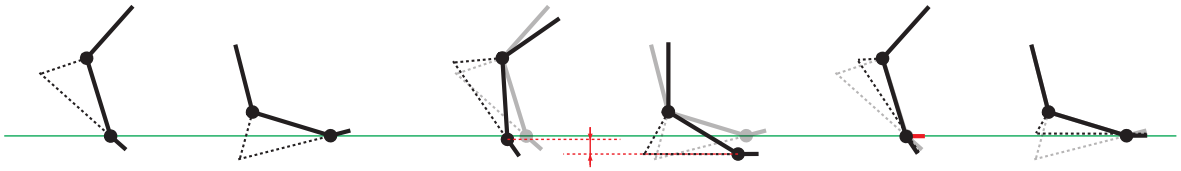

Figure 4.30.: Methods of addressing the feet miscalibration artifact. The original motion, with heels consistently below (above) the ground (left) can be altered using an IK approach, but this would lead to significant difference of feet height during the footstep (middle), which has to be addressed at the knee level. By altering the base pose, however, the artifact can be successfully addressed without any further need of pose adjustments (right).

the motion is applied to a different character. Another one is specific to the skeleton-based motion capture, where the skeleton is calibrated on a different motion or slightly inaccurate marker set than captured. The main reason for this type is the fact that the human body is not rigid and the skeleton cannot represent the dynamic data accurately. However, in both cases, the artifact exhibits itself in the same way.

With constraint time interval data available, a standard method of enforcing constraints would use a certain type of inverse kinematics (IK, e.g., see Section 4.4.3). However, for this kind of artifact, the IK method, which flattens the feet as the first step, can lead to significant foot height difference. This difference has to be compensated for at the knee level (see Figure 4.30, middle). Adjusting the skeletal base pose, though, can lead to a solution that does not alter the source animation significantly (see Figure 4.30, right).

The corrective algorithm then consists of several steps:

1. As we want to keep the overall foot direction (rotation around Y axis) intact, we extract the XZ transformation by first transforming the unit vector $\boldsymbol{x}$ using the global ankle matrix $\boldsymbol{F}_a$, leading to a transformed vector $\boldsymbol{x}_t$:

$$\boldsymbol{x}_t = \boldsymbol{F}_a \boldsymbol{x} \qquad \boldsymbol{x} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

2. From this vector, we can extract the rotation angle α:

$$\alpha = -atan2\left(\frac{x_t^{(z)}}{\sqrt{\left(x_t^{(z)}\right)^2 + \left(x_t^{(x)}\right)^2}}, \frac{x_t^{(x)}}{\sqrt{\left(x_t^{(z)}\right)^2 + \left(x_t^{(x)}\right)^2}} \right) \qquad \boldsymbol{x}_t = \begin{bmatrix} x_t^{(x)} \\ x_t^{(y)} \\ x_t^{(z)} \\ 0 \end{bmatrix}$$

3. which can be used to create a matrix $\boldsymbol{M}$ that expresses the rotation around the Y axis by angle α. This way we can extract the XZ transformation matrix $\boldsymbol{F}_a^{(xz)}$:

$$\boldsymbol{F}_a^{(xz)} = \boldsymbol{M}^{-1} \boldsymbol{F}_a$$

4. By converting the rotational part of $\boldsymbol{F}_a^{(xz)}$ to quaternion representation, we can use NLERP (normalised linear interpolation) to compute the approximate average transformation $\bar{\boldsymbol{F}}_a$ of all the footstep frames of the animation clip, with axes lying on the XZ plane (keeping the Y directional part intact).
5. Using this matrix, we can alter the binding pose and the animation frames, so that the overall

ankle transformation stays intact, thus achieving the effect illustrated on Figure 4.30 (right):

$$\begin{aligned} \boldsymbol{R}'_a &= \boldsymbol{R}_a \bar{\boldsymbol{F}}_a \\ \boldsymbol{T}'_a &= \left(\bar{\boldsymbol{F}}_a\right)^{-1} \boldsymbol{T}_a \end{aligned}$$

leading to the premultiplied matrix $\boldsymbol{P}'_a$:

$$\begin{aligned} \boldsymbol{P}'_a &= \boldsymbol{R}'_a \boldsymbol{T}'_a \\ &= \left(\boldsymbol{R}_a \bar{\boldsymbol{F}}_a\right) \left(\left(\bar{\boldsymbol{F}}_a\right)^{-1} \boldsymbol{T}_a\right) \\ &= \boldsymbol{P}_a \end{aligned}$$

and consequently

$$\boldsymbol{F}'_a = \boldsymbol{F}_a$$

For practical reasons, we use the same correction matrix for both feet, which keeps the skeletal structure symmetric. As a result of this transformation, the foot orientation of the skeleton in the binding pose then corresponds to the average orientation of the constrained feet.

4.4.2. End-effector Positions

The input data of the footstep correction algorithms described in Section 4.4.3 consist of the original animation and the full target trajectories of the feet. These trajectories should contain corrected footstep information, while being as close as possible to the original feet trajectories without introducing any discontinuity artifacts. This section focuses on determining these trajectories.

An alternative approach, often used in previous work, would use only target information of the constrained frames. After solving the poses for these frames, it would employ a filtering scheme to reduce discontinuity artifacts. However, our scheme allows a simpler and consistent data handling for all frames of the animation, not differentiating between the constrained and unconstrained frames.

A specific property of footstep handling is the fact that the constraint consists of two separate joints. The primary joint (ankle) describes the overall motion of the foot, and its representation implicitly contains the heel bone (see Figure 4.2). The secondary joint is the toe, which, as a child of the ankle joint, is directly influenced by the ankle motion. The target feet trajectory has to explicitly reflect this property.

A footstep constraint can be fully described using three points on the foot (heel, ball joint, toe) and four stages (see Figure 4.5). The behaviour of each of the three points can be described as follows:

1. The heel is in contact with the ground during footstep stages one and two, with rolling contact motion during stage one.
2. The ball (or toe) joint is constrained in stages two and three, with the ankle rotating around its position in stage three (thereby temporarily reversing the parent-child relationship, as the ankle motion should be dependent on the ball joint motion).
3. The toe tip is in contact with the ground in stages two and three, and partially in stage four.

However, in our implementation, we ignore the heel contact in stage one, as in a normal walk it is very short; and toe tip from stage four, as it does not give a significant force response to the foot.

As our thesis is primarily concerned with locomotion, we focus on *periodic animations*. The periodic conversion operation, as described in Section 5.3.1, alters the original motion data by detecting and cutting one motion period and making sure that the start and end of the resulting animation clip are connected. Consequently, the first and the last frame of this clip contains the same pose, differing only

in the global body translation and rotation, which allows the motion clips to be simply concatenated in order to generate a long animation (see below). However, the blending required to achieve the similarity between the start and end frames can introduce a certain level of footsliding, requiring a post-processing step performing footstep cleanup.

Given the first frame ($t = 0$) and last frame ($t = \tau$) of a periodic clip with period τ, we can create the periodicity transformation of the root $\boldsymbol{T}_p$ using the root local transformation matrix $\boldsymbol{T}_1(t)$:

$$\boldsymbol{T}_p = \boldsymbol{T}_1(\tau)\left(\boldsymbol{T}_1(0)\right)^{-1}$$

whose properties then allow us to convert between the first and last frame as follows:

$$\begin{aligned} \boldsymbol{T}_p\boldsymbol{T}_1(0) &= \boldsymbol{T}_1(\tau) \\ \boldsymbol{T}_p^{-1}\boldsymbol{T}_1(\tau) &= \boldsymbol{T}_1(0) \end{aligned}$$

By concatenating the clips (transformed by $(\boldsymbol{T}_p)^n$), we can create an animation with arbitrary length.

Converting the animation to world space, we can express the periodic transformation as

$$\begin{aligned} \boldsymbol{F}_p &= \boldsymbol{F}_1(\tau)\left(\boldsymbol{F}_1(0)\right)^{-1} \\ &= \boldsymbol{R}_1\boldsymbol{T}_1(\tau)\left(\boldsymbol{R}_1\boldsymbol{T}_1(0)\right)^{-1} \\ &= \boldsymbol{R}_1\boldsymbol{T}_1(\tau)\,\boldsymbol{T}_1(0)^{-1}\,\boldsymbol{R}_1^{-1} \\ &= \boldsymbol{R}_1\boldsymbol{T}_p\boldsymbol{R}_1^{-1} \end{aligned}$$

The world space periodic transformation is the same for all joints of the skeleton. Therefore, it can be used for creating periodic animations of separate bodyparts (e.g., feet) in the world space.

The first step towards determining the full end-effector trajectory is to compute the final footstep position and orientation in a particular constraint interval, for our case the double constrained phase (i.e., stage two). Assuming a certain level of noise at the beginning and the end of the constraint, the best starting value would be in its time interval centre. In the case of periodic animation, however, the start time t_{start} can actually have a higher value than t_{end}, therefore describing a constraint that wraps around the end and start of the period. The actual t_{centre} can therefore be computed using:

$$t_{center} = \frac{t_{start} + t_{end}}{2}\ mod\,\tau$$

The full footstep information involves two parts – ankle and toe. The constrained ankle transformation $\boldsymbol{F'}_{ankle}(t_{centre})$ is determined from the original $\boldsymbol{F}_{ankle}(t_{center})$ by using only the XZ translation and Y rotation from $\boldsymbol{F}_{ankle}(t_{center})$, and Y translation together with XZ rotation from $\boldsymbol{A}_{ankle}$. This process effectively plants the ankle segment of the foot on the ground. The constrained toe transformation $\boldsymbol{F}'_{toe}(t_{center})$ is determined using:

$$\boldsymbol{F}'_{toe}(t_{center}) = \boldsymbol{F}'_{ankle}(t_{center})\,\boldsymbol{R}_{toe}(t_{center})$$

which sets the $\boldsymbol{T}_{toe}$ to identity, thus making the whole foot fully planted.

To generalise the constraint information to other frames of the constraint interval (t_{frame}), we have to take into account the periodicity of the animation. This leads to a total of five possible forms of

the final $\boldsymbol{F}'_{ankle}(t_{frame})$ computation:

$$\boldsymbol{F}'_{ankle}(t_{frame}) = \begin{cases} \boldsymbol{F}'_{ankle}(t_{centre}) & t_{start} < t_{centre} < t_{end} \\ \\ \boldsymbol{F}'_{ankle}(t_{centre}) & t_{end} < t_{start} < t_{centre} \quad and \quad t_{frame} \geq t_{start} \\ \boldsymbol{F}_p^{-1}\boldsymbol{F}'_{ankle}(t_{centre}) & t_{end} < t_{start} < t_{centre} \quad and \quad t_{frame} < t_{start} \\ \\ \boldsymbol{F}_p\boldsymbol{F}'_{ankle}(t_{centre}) & t_{centre} < t_{end} < t_{start} \quad and \quad t_{frame} \geq t_{start} \\ \boldsymbol{F}'_{ankle}(t_{centre}) & t_{centre} < t_{end} < t_{start} \quad and \quad t_{frame} < t_{start} \end{cases} \tag{4.5}$$

The $\boldsymbol{F}'_{toe}$ is computed in a similar way by substituting in the previous equation. This provides all the foot trajectory information for footstep stage two.

Stage three requires the toe joint to be fully planted, while the ankle joint rotates around its position. Using a small variation of the methods described above, we decompose the original ankle motion to its XZ and Y parts, while enforcing the XZ part to be connected to the fully constrained toe joint. This provides the ankle motion for stage three directly based on the original motion, thus minimising the introduced changes.

At this point, we have determined the full trajectory for stages 2 and 3. To make sure that the final animation does not contain any non-smooth transitions, we determine the corrective transformation for the remaining frames by linearly interpolating between the end of the previous footstep and beginning of the next. The corrective transformations for the ankle bone, describing the difference between the original animation and its corrected version at the beginning and end of a constrained interval, can be determined using:

$$\boldsymbol{F}^{(c)}_{ankle}(t_{start}) = \boldsymbol{F}'_{ankle}(t_{start})\left(\boldsymbol{F}_{ankle}(t_{start})\right)^{-1}$$
$$\boldsymbol{F}^{(c)}_{ankle}(t_{end}) = \boldsymbol{F}'_{ankle}(t_{end})\left(\boldsymbol{F}_{ankle}(t_{end})\right)^{-1}$$

with the interpolated matrix $\boldsymbol{F}^{(c)}_{ankle}(t_{frame})$ computed using quaternion SLERP. For simplicity we do not provide all five versions, which can be determined using a similar approach as in Equation 4.5. The final animation frames are then computed using:

$$\boldsymbol{F}'_{ankle}(t_{frame}) = \boldsymbol{F}^{(c)}_{ankle}(t_{frame})\boldsymbol{F}_{ankle}(t_{frame})$$

To ensure no undesirable artifacts, the toe bone is interpolated in the local space (matrices $\boldsymbol{R}_{toe}$) in the same way, making the motion relative to the corrected ankle motion.

4.4.3. Footstep Constraints Solutions

Using the simple kinematic algorithm from Section 4.4.2, we have determined the position and orientation of both feet for each frame of the animation. The new trajectories are smooth and follow the original data as closely as possible while enforcing the foot constraints.

In this section, we describe three kinematic methods that change the original animations to accurately follow a new trajectory. Two of these algorithms are tested in our perceptual study aimed at footstep correction saliency (see Section 6.3). Although more complex dynamic or optimisation solutions are possible, kinematic algorithms are prevalent in previous work due to their simplicity and the possibility of combining them with simple filtering methods.

For the sake of simplicity, we avoid mathematical descriptions of each of the methods, as all the

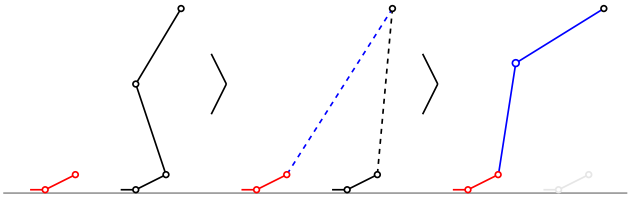

Figure 4.31.: Limb-lengthening solution for footstep constraints. The original limb pose (left) is first adjusted in the hip joint to point to the expected end-effector position (middle). Then, the limb is scaled so that its knee joint is placed in the expected position (right). This leads to constraint satisfaction without changing the knee angle, eliminating the possibility of the knee popping artifact.

concepts can be implemented using simple 3D geometric operations and transformation algebra.

Given the trajectories of both feet, several possible animation manipulations to cause the feet to follow each trajectory can be summarised as follows:

- the *limb lengthening* solution changes the leg bone lengths to reach the final position without influencing the root or knee movement. However, adjustments to hip and ankle bones are necessary.
- the *rootfix* solution primarily affects the position of the root bone to enforce constraints. Consequently, the ankle and hip joints have to be changed to reflect the new root position, but the knee can essentialy remain unchanged.
- the *inverse kinematics* solution focuses on changing the leg configuration, affecting the knee joint, and consequently the hip and ankle joints. In the case of a leg not being able to stretch to a certain distance, a correction of the root position is necessary.
- a heuristic combination of all the above.

Each of these solutions produces a certain number of artifacts, but the nature of these artifacts differ significantly (e.g., knee-popping, body pose changes, limb length changes). The following paragraphs aim to analyse their possible impacts.

The **limb-lengthening solution** takes its inspiration from a perceptual study performed by Harrison et al. (2004), who showed that, under certain conditions, even significant changes of limb length provide perceptually plausible results. This should allow the leg length to be changed significantly, while keeping the overall pose (determined mainly by the knee angle) intact.

Our proposed method first changes the orientation of the hip joint to reflect the new direction. The leg resize value can then be computed using a ratio between the original leg length and the distance between the hip joint and the constrained ankle. Rescaling both the upper and lower leg connects the leg bones with the ankle joint, which leads to an adjustment of ankle orientation to reflect the constrained target.

The **rootfix solution** attempts to adjust the root position to compensate for the end-effector displacement, while changing the leg configuration as little as possible. The following paragraphs describe two tested versions of the algorithm.

The first naive version keeps the knee configuration intact, changing only the hips and ankle orientations. The computation follows two steps: First, a circle is computed by intersecting two spheres centred at the constrained ankles, with their diameter determined by the leg lengths from the current frame. Second, the new root position is determined by finding the point on this circle which is closest to the original root position. While this solution is feasible for most joint configurations, necessitating only minimal adjustments of the root position, certain combinations of constraint positions relative to the original state can lead to severe artifacts (see Figure 4.32)

Figure 4.32.: The naive rootfix solution for footstep constraints. Left – a successfully applied correction leads to root position change proportional to the required end-effector change. Right – a case when a relatively minor end-effector correction leads to a large difference in the root position.

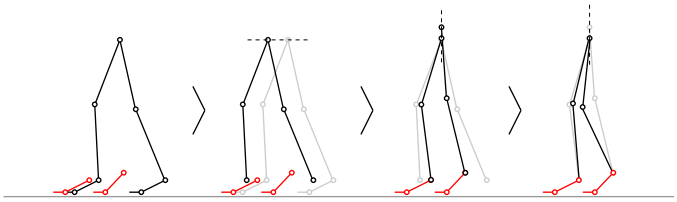

Figure 4.33.: XZ-Y separate rootfix solution. The first step determines the best fit in the XZ plane (left), while the second step determines the Y position (right).

A second option, addressing the main drawback of the previous one, computes the new root position in two steps: First, the root is shifted in the XZ plane to a point determined by the ratios between the original root and end-effector positions (see Figure 4.33). Second, the root is shifted on the Y axis only and the new leg configuration is determined using IK. While not keeping the knee configuration static, this solution limits both the artifacts and the changes in feet configuration. Unfortunately, it also changes the root motion, potentially causing artifacts that influence the whole body dynamics.

The **inverse kinematics** solution is a standard way of solving constraints problems. It determines the new leg configurations by finding the smallest changes of the leg joints that satisfy the constraints. For certain cases, when a sole change of the leg's configuration cannot provide a solution, a root change (similar to the rootfix solution) is necessary. A classical artifact of IK is the knee popping, caused by a non-linear relationship between the leg extension and joint parameters (knee). A simple solution to this problem can be to limit the maximal stretch of the leg, but that often leads to more severe changes of the root trajectory.

A **hybrid solution**, presented by Kovar et al. (2002b), uses a combination of all methods above to provide a comprehensive footstep cleanup solution. In their method, the IK solution is used to change the leg configuration, together with a rootfix when the IK change would lead to knee popping or is not able to provide a solution. As altering the root trajectory can lead to discontinuities in the motion, they filter the resulting root trajectory using a low-pass filter and, so as not to alter the leg configuration, they apply the changes using the limb lengthening solution.

After testing all the previous approaches (see Section 6.3.3 for a perceptual experiment comparing two of the solutions), we use primarily the hybrid approach, while not changing the bone lengths. Based on the results of our experiment, the lengthening solution would be an ideal choice, but it requires a slight change in the skinning algorithm (see Kavan et al. (2007a) for more details). While it is not a problem in a custom animation system, this alteration is not possible in the main 3D animation software systems (3DS MAX, Maya).

5

A Linearised Locomotion System for Crowd Animation

Animation of individual characters is a well-established topic in animation research with a broad range of methods available (see Section 2.5). However the animation of many hundreds or even thousands of simultaneously visible characters presents new challenges. First, a data-driven heterogeneous crowd requires a variety of motions, so fully automatic motion pre-processing is highly desirable. Second, the run-time component of the animation system needs to be fast enough to allow for the real-time rendering of many individually animated characters. For this reason, previous crowd models tended to employ very simplified motion models (see Section 2.5.4). In this chapter, we present a novel animation system, targeted specifically at human locomotion and medium level-of-detail (LOD) characters.

Because we are focusing on crowd animation, we assume a LOD management system is already in place, which allows the characters to be separated into three groups:

1. a small group of characters closest to the camera, animated with a highly accurate animation model (contains usually up to 20 characters),
2. medium level-of-detail characters, occupying most of the screen area in high density crowd scenarios, whose motion still requires a high level of accuracy, but its smallest details are not recognisable anymore (usually up to several thousand characters; all characters in Figure 1.1 belong to this group),
3. and the largest group of very distant characters, who are far enough away to allow extremely simple animation models (e.g., an animation with a significantly reduced number of bones, a single repeated walk pattern, or no animation at all).

A single character can change its group many times during a single simulation run. To avoid introduction of artifacts into the motion, the transition has to be smooth (e.g., linear blend between the two animation models over a number of frames), and the transition point should be randomised with hysteresis (different in and out value). Both of these conditions represent the animation equivalent of

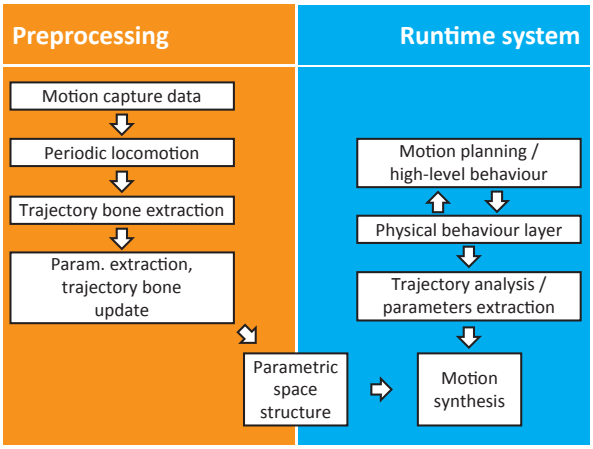

Figure 5.1.: The overview of our data-driven locomotion system, showing the division between preprocessing and runtime components, with data flow depicted using arrows.

rendering LOD transition techniques, and are aimed to prevent a sharp transition line.

Our data-driven locomotion system can be separated into components, reflected in the structure of this chapter:

- we first provide an overview of the system functionality (Section 5.1),
- then introduce the Motion Map concept, a novel way of manipulating motion comparison matrices (Section 5.2),
- next we describe the fully automatic preprocessing of the motion clips used in our system – the periodisation, trajectory bone embedding and trajectory idealisation; these steps allow the abstraction of high-level locomotion features from the animation data (Section 5.3),
- we then demonstrate the integration of our model into a behaviour pipeline, using the derived high-level locomotion features in the previous section and a PD controller as an optional interface layer (Section 5.4),
- the parametric space concept and its parameters are then described. This is the central part of our synthesis model (Section 5.5).
- Based on the properties of our parameterisation, we present a linearised skinning model, capable of combining both mesh-based animation and articulated characters (Section 5.6).
- Finally, we evaluate the model in terms of computational speed and efficiency (Section 5.7).

5.1. Overview

Our system can be characterised as a parametric locomotion synthesis method. While essentially a single-character technique, its computational efficiency and optimised integration with a higher-level planning module make it particularly suitable for crowd animation. A very similar technique, using the same basis for the parameter representation and the interpolation synthesis, was independently developed by CryTek and deployed in CryEngine 3 (as presented at the Game Development Conference, 2011).

Our system can be divided into two parts – first, the preprocessing phase, where the underlying data structure is created, and second, the runtime system, which uses the preprocessed data to generate character motion for the crowd system (Figure 5.1).

Because we aim to use motion capture data as input, each motion clip needs to be preprocessed before it can be placed in the parametric space structure (described in Section 5.5). This consists of en-

suring periodicity (Section 5.3.1), creating an auxiliary trajectory bone (Section 5.3.2) and extracting clip parameters (Section 5.3.3).

The run-time system takes the parametric space structure and motion planning data as input, analyses the planned trajectory in order to extract parametric information and uses these parameters to create the resulting motion by blending and timewarping source motion clips (Section 5.5).

Our concept is closely related to several previous methods. It is a parameterisation method based on motion blending, in principle similar to parameterised nodes in the work of Rose et al. (1998) or edges in Shin & Oh (2006) and Heck & Gleicher (2007). It allows for motion synthesis by interpolation (Wiley & Hahn, 1997; Sloan et al., 2001), but as the motion parameters after interpolation are well-defined, we do not require resampling of the parametric space. Lee et al. (2010) recently introduced a locomotion model built around a many-dimensional parameterisation of keyframes. Their approach encoded the time domain, in the form of pose samples, into the parameterisation structure, thereby implicitly allowing timewarping and multiple clip blending. Even though their method can parameterise and generate a wider variety of motions than our approach, the synthesis results do not correspond exactly to the expected outcome of the parameters, and therefore require an expensive graph planning mechanism to provide a motion with the expected properties. By explicitly representing the time axis, our approach allows more accurate motion synthesis with a very simple motion planner.

Our method builds on the work of Park et al. (2002), who introduced an interpolation based locomotion model, which was later incorporated into a more generic graph-based motion generation approach (Park et al., 2004). Our work differs from their approach in several important ways – our method includes a fully automatic procedure to build the parameterisation structure from a set of motion clips. We use a different parameterisation structure and interpolation scheme, thus allowing us to perform local blends alone (3 motions) instead of blending all motions for every combination of parameters. Moreover, it allows us to generate motions that correspond to the requested parameters accurately, without requiring any additional post-processing.

5.2. Motion Map Concept

The motion preprocessing in our system is built around the concept of a motion map. It is essentially a visual representation of a matrix of per-frame comparison values (based on a metric function), with one pixel corresponding to each pair of frames. In this way, time-dependent manipulations can be performed, which is hard to achieve otherwise.

This allows us to perform time-dependent manipulations that are hard to do otherwise.

The idea of a motion map is not entirely new. Kovar & Gleicher use the same concept in several of their works (2002a; 2003; 2004) to perform time-dependent matching of motions using registration curves. The graph approach of Lee et al. (2002) starts with a description of motions as Markov processes, with a state for each frame and a transition edge for each pair of frames, describing their similarity. A Markov process can be represented as a transition matrix, which is also a motion map. Many other examples of its usage can be found, as it provides a convenient and intuitive way of describing the difference between two animations (or between different segments of one animation).

In our work, we interpret the motion map as an image, and perform image-based signal processing operations to analyse the properties of the motion. This provides a novel perspective on motion data, allowing significant simplifications of some of the more complex operations on the motion.

In the following paragraphs, we describe several ways to generate motion maps and how we have used them in the context of our work.

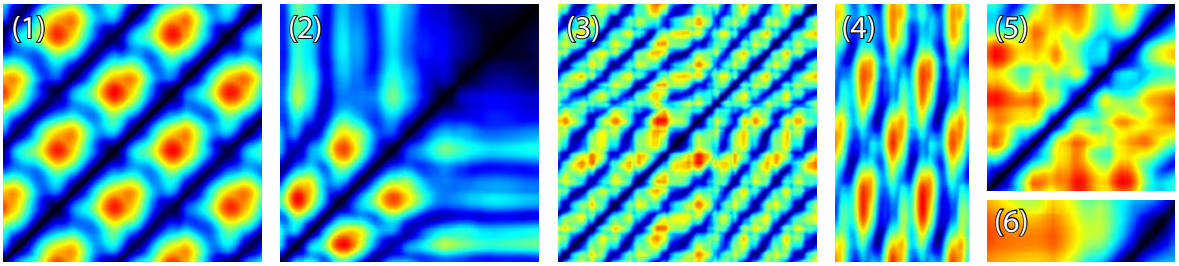

Figure 5.2.: The motion maps visualisation overview. Both horizontal and vertical axes describe animation frames (1 per pixel), with the origin in bottom left corner, while the pixel colour represents the value of two frame comparison. The maps describe: (1) a straight normal walking motion; (2) a stopping motion; (3) slow walking on a circle; (4) fast (horizontal) vs. slow (vertical) walking motion; (5) a vault; and (6) a vault (horizontal) with a normal walking motion (vertical).

5.2.1. Visualising Motion Maps

In addition to automatic path-finding for timewarping curves (Kovar & Gleicher, 2003), motion maps can provide important insights into the nature of a motion, or the comparative properties of two motions. In this section, we provide an informal description of the motion properties that can be explored using motion maps; in Section 5.2.3, we elaborate further on a subset that is usable for periodic motion analysis.

Motion maps can be used to detect popping artifacts, which are very common in motion capture data (caused by lost markers, camera switching and systematic noise). They are represented as vertical and horizontal lines with high contrast, caused by a quick change of pose in the animation (see Figure 5.2, 5 and 6). While it is possible to detect them by other means, motion maps provide a simple and automatic way of determining both the location and severity of the artifact. If both horizontal and vertical axis represent the same motion, the horizontal and vertical lines will intersect on the diagonal.

Another important property of the motion map is that, when both horizontal and vertical axes represent the same motion clip, the motion map has zero diagonal and is symmetric (see Figure 5.2, 1, 2, 3 and 5). For periodic motions, the similarities between periods create distinct diagonal lines in the motion map, which are parallel to the main diagonal (Figure 5.2, 1). Comparing two periodic motions with different periods leads to a similar effect, with the diagonal skewed in the direction corresponding to the timing difference (see Figure 5.2, 4). This effect, together with a specialised smoothing method, is used for period detection in our system (see Section 5.2.3).

If the motion is not perfectly periodic, or periods of the two motion do not match (see Figure 5.2, 3), the parallel diagonals are distorted (a phenomenon used by Kovar & Gleicher (2004) for the construction of registration curves). Our example shows how even a small sample of walking on a circular trajectory can contain relatively large variations.

A static part of the motion clip causes smearing of the image, as there is no change in the posture over a period of time (see the stopping motion in Figure 5.2, 2).

A highly dynamic motion, such as a vault (see Figure 5.2, 5), would not exhibit any of the previously mentioned properties. From our example, however, it is easy to see that the starting and ending pose of the vault was the same, and that the vault was finished with a straight step (by comparing it with a locomotion clip; see Figure 5.2, 6).

A motion map of a long sequence can provide more insights into the nature of the motion. Figure 5.7 contains a dancing motion – its regular structure is easy to identify, together with sequences of repetitive movements and similarities between different parts of the dance. Automatic motion analysis

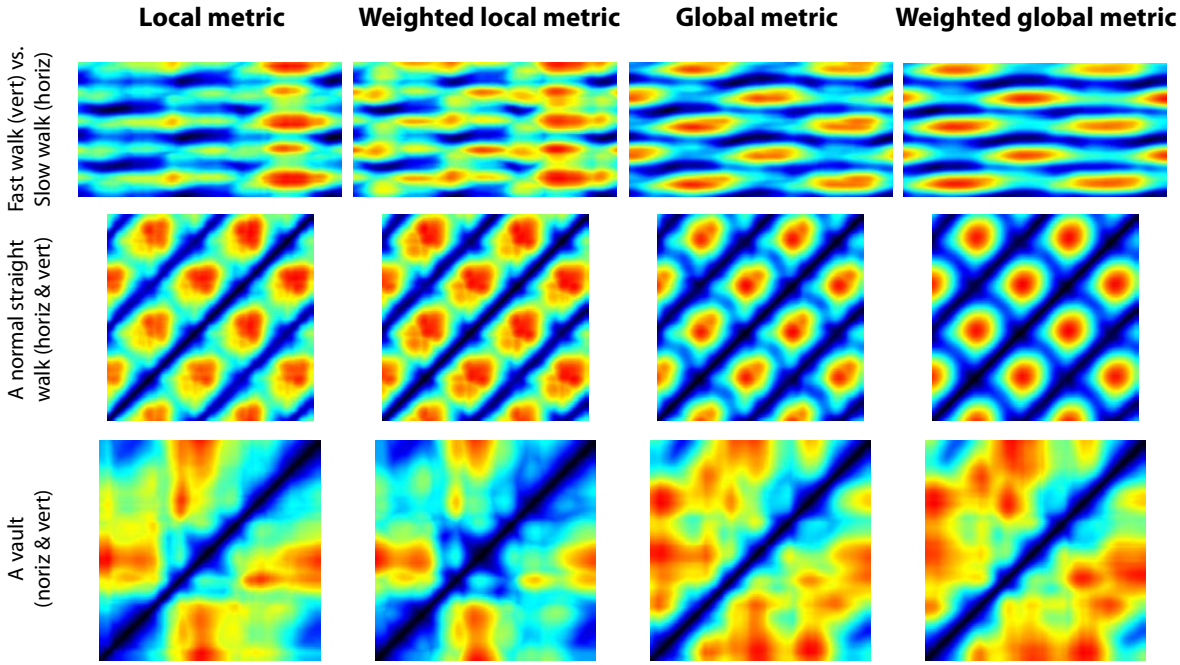

Figure 5.3.: A comparison of metric functions, showing their impact on the resulting motion map.

of this type, based on beats and rhythmic patterns, was performed by Kim et al. (2003). In our case, we used the motion maps to detect the dancing patterns as related to the beat and used them to create an interactive installation called Biodancer (see Figure 1.2).

The analysis of the properties mentioned above can be performed using relatively simple image processing algorithms. In Section 5.2.3, we use them to detect the most suitable period of a motion, as required for our parametric motion synthesis methods. However, the applications and implementations of other methods mentioned above pose an interesting avenue for future research.

5.2.2. Comparison of Metric Functions

A motion map is generated by comparing pairs of frames using a *metric function*, which strongly influences the properties of the resulting map. In this section, we evaluate the use of two metric functions – a global function, based on the world positions of points on the character's mesh (Kovar et al., 2002a) and a local function, which is determined by the differences between joint angles (Lee et al., 2002). Both functions are used in a form that does not account for the character's position in the environment.

As the temporal properties of the motion are determined by the frame sequence (Kovar et al., 2002a), we do not explicitly use the dynamic aspects of the motion, such as velocity or acceleration. Previous work suggests that the dynamic properties are not important except for judging the overall direction of the motion (Wang & Bodenheimer, 2003). Moreover, in a later processing step, we will be working with motion timewarping; the use of dynamic properties would make this impossible, as any significant timing changes would have to be reflected back into the metric computation.

We test two versions of each metric – weighted and unweighted. The weighting values for the local metric were suggested in Lee et al's original paper (2002) and a set of weights for the global metric were described by Arikan (2006).

In Figure 5.3, we use three motions to demonstrate the different properties of the four metrics.

The first row contains a comparison between a slow walk on a circular trajectory and a straight

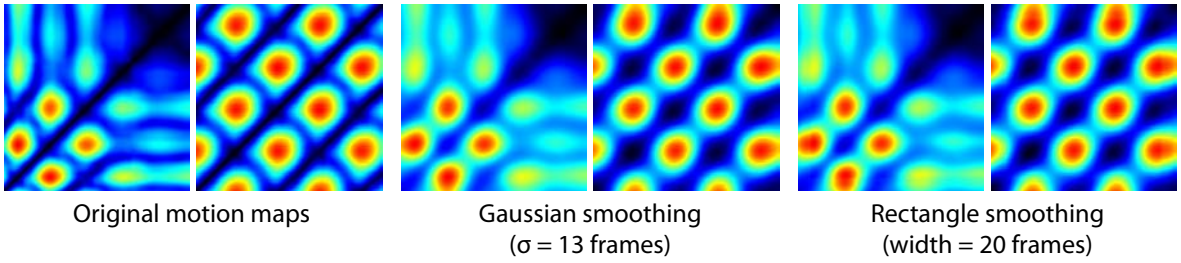

Figure 5.4.: An illustration of the effects of applying a normalised smoothing filter to a motion map.

walk with normal speed. While both clips contain locomotions performed by the same actor, the overall poses and timing differ significantly. Moreover, the actor turned his upper body slightly at the end of the straight walk clip (i.e., looked in a wrong direction), thus creating an artifact in the motion, but not changing its overall character. Arguably, this artifact was better handled by the global metric (especially the weighted version), as it did not change the overall features of the motion map. Moreover, the global metric provides stronger local features for temporal mapping between the two motions (diagonal lines), which is, again, a desired property.

The middle row of Figure 5.3 shows motion maps generated by comparing a straight walking cycle with itself. The very strong 45° diagonal lines set apart by a constant distance, show the strong periodic nature of the walk. However, the global metrics also exhibit strong antidiagonal lines, matching the forward motion of a leg in one cycle with its backward motion in the other cycle. This shows that the global metric is not capable of correctly distinguishing between the left and right cycle, which might lead to problems for certain types of motions.

The last row describes a comparison of a vaulting motion with itself, starting and ending in a similar pose. In this case, the global metric correctly recognises the significant differences between the poses during the vaulting motion, while the local metric is strongly influenced by the localised joint poses, ignoring the fact that the character is for a time upside down. Moreover, the temporal similarity between the half-step at the beginning and end of the sequence is much better described using the global metric. Therefore, in this case, the global metric performs better.

From these examples, we can conclude that the global metric introduced by Kovar et al. (2002a) provides more intuitive results. This conclusion is also supported by our perceptual evaluation of motion metrics as presented in Section 6.1.

5.2.3. Period Detection with Motion Maps

The motion maps can be considered as greyscale images, i.e., 2D matrices with one floating point value per cell, which enables us to use image processing techniques. With an intuitive analysis, we can interpret each operation as a method for comparing motions, that exhibit certain properties. Our main target is to detect suitable transition points, which can be used to cut a part of an input clip to create a periodic animation.

By considering the neighbourhood of a particular pixel, we take into account the time-domain properties of the corresponding motion fragment. The forward flow of time is described by the main diagonal (bottom-left to top-right), while the secondary diagonal represents the reversed time flow.

Simple smoothing of the motion map, using either a Gaussian smoothing function or a rectangular window, can determine the similarity between a window of frames (see Figure 5.4). While useful for noise filtering, this method smoothens both forwards and backwards in time, without taking the frame sequence into account.

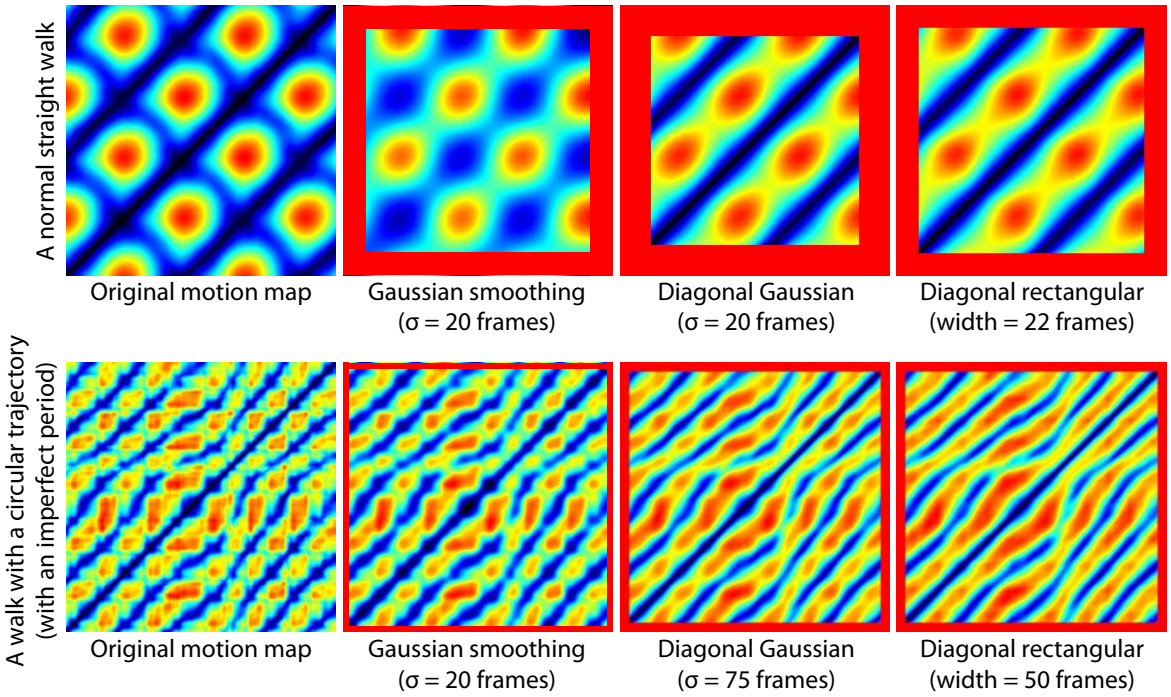

Figure 5.5.: A comparison of 2D Gaussian smoothing with the effects of the diagonal smoothing approach.

To address this issue, we can smoothen the image **diagonally** using a 1D smoothing function, resulting in a sum of comparisons of consecutive frames only (see Figure 5.5). This corresponds to the multiple consecutive frames comparison approach taken by Kovar et al. (2002a), who also present a comprehensive justification for doing so. The Gaussian filter gives more weight for the values close to the centre of the evaluation window, which is a useful property for motion blending (the 50% blend is in the middle of the blending curve, so it is the point with the highest possible blending error). The diagonal does not have to have the angle of 45°; any other angle between 0 and 90° would correspond to a certain level of timewarping. A more effective approach to detect arbitrary angles of line segments is the Hough transformation, which can be utilised for this purpose. This approach, together with local minima detection (see below), can be used for a motion graph transition search, as it essentially allows all combination of frames and different timewarping values to be compared, with each frame pair tested only once.

The methods shown on Figures 5.4 and 5.5 also differ in **boundary handling**. While the former use a normalised smoothing filter implementation (not taking into account the values outside the map), which distorts the map close to the edges, the latter simply exclude all pixels close to an edge from the filtering (which is also the approach taken in all the following examples).

For detecting a **motion period**, it is desirable to find two consecutive motion fragments of the same length with very similar corresponding frames, as they need to be blended together to reduce the possibility of artifacts (see Section 5.3.1 for more details). Therefore, it is desirable to link the blending length, described using the width of the smoothing window, with the total period length, represented as the Manhattan distance from the diagonal. This approach is illustrated in Figure 5.6. However, we need to address the handling of the boundaries, as they are even more pronounced in this case (Figure 5.6, 2 and 3). The previously used approach, excluding any blends that contain undefined values, would lead to a 50% reduction of usable pixels (Figure 5.6, 4). However, if we limit

the corrective blend (Figure 5.6, 4 shows an example of limiting the corrective blend to 50% of the period length), we can include significantly more pixels, thereby allowing us to use shorter source clips that include only one full period and a smaller fraction of another period.

The possible transition frames are described as **local minima** of the motion map (the best frames from the neighbourhood). To detect them, we can employ another image processing technique – the Laplacian filter, which results in values close to zero for pixels at local extrema.

To limit the interval of evaluated comparisons, it is sometimes desirable to exclude values above certain level. This can be simply achieved using **image thresholding**. However, the thresholding values are relative to the range of the input values and their histogram. Applying **histogram equalisation** on the motion map alters the values to make them directly related to the percentage of their occurrence in the map (see Figure 5.7). This allows the threshold to be simply set, for example, to 0.2 to eliminate 80% of the highest values.

The position dependent smoothing filter, together with histogram equalisation and image thresholding, is the combination we use for period detection from now on. It has proven to be robust enough to handle all the data in our motion capture database (more than 1700 clips of locomotion), and to detect motions where a period was not present due to a processing error.

5.3. Data Preprocessing

Motion capture as the input data source of an animation system can provide very realistic results, but the raw motions need to be extensively edited in order to fit the system's requirements (e.g., periodicity, parameterisation). The design of our system allows the raw data to be used directly, with the necessary editing being performed in a fully automatic manner.

5.3.1. Periodisation

The first preprocessing step involves converting the original motions to periodic, loopable clips. Because human locomotion is very close to being periodic, the necessary changes to the original motion are very small.

To detect the period of an input motion clip, we use the motion map concept and the period detection procedure introduced in Section 5.2.3. This provides us with the first frame and the last frame of the best period in the input motion, which are then used to crop the source animation.

When the animation clip contains exactly one period of the input locomotion, we convert the motion to its differential form – i.e., the motion of the root of the skeleton hierarchy, that originally describes the global character position and orientation in space, is changed into a difference of the orientation and position between the current and subsequent frame:

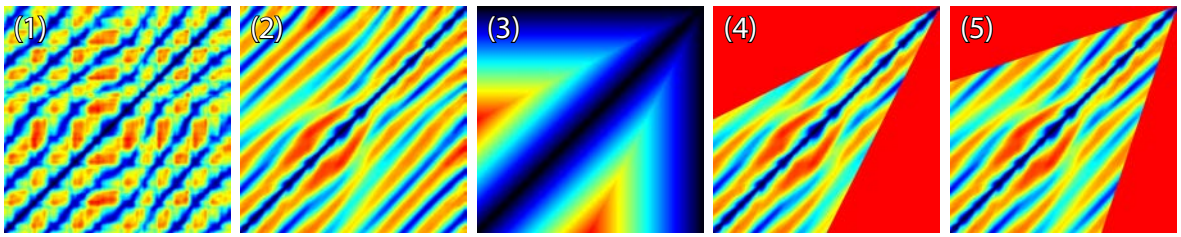

Figure 5.6.: An example of period detection using the "periodic smoothing" approach. The original motion map (1) can be smoothed using a normalised version of the periodic smoothing (2). However, the number of smoothed frames can be very low close to the edges (3 – the colour corresponds to the number of frames used for smoothing, related to the full length of the source animation). Therefore, it is desirable to exclude the values with a low number of samples (4 – full second period required, 5 – 50% of second period required).

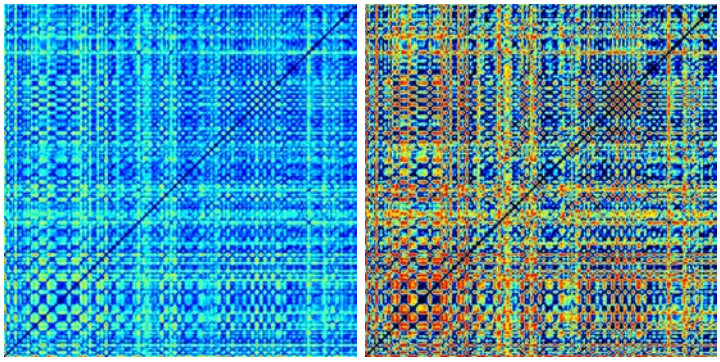

Figure 5.7.: An illustration of histogram equalisation on a motion map generated from a dancing sequence. After the equalisation is applied, the value x of each pixel directly maps to the percentage of values in the interval $[0...x)$ in the map.

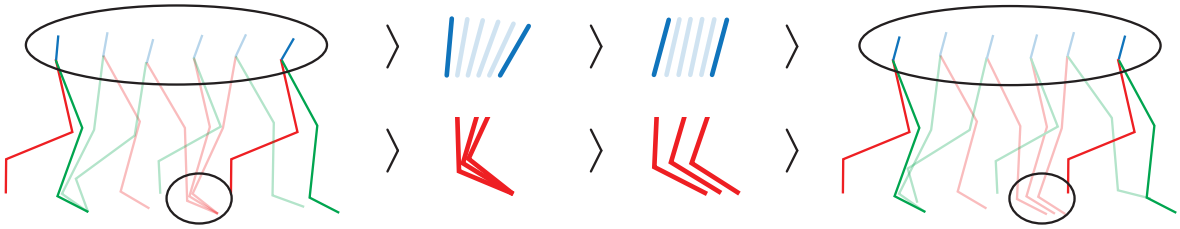

Figure 5.8.: Illustration of the footsliding artifacts as a result of period difference frame blending (left – original animation, right – blended result). If the difference frame contains a non-identity transformation for the root bone (blue), the hierarchical nature of the blending algorithm introduces a certain amount of footsliding in the middle of the animation clip.

$$\begin{aligned} dT_{root}(n) &= T_{root}(n)\,(T_{root}(n-1))^{-1} \\ T_{root}(n) &= dT_{root}(n)\,T_{root}(n-1) \end{aligned} \tag{5.1}$$

where T_{root} is the original root transformation in matrix form, dT_{root} is the transformation in differential form and n is the keyframe number. This step ensures the independence of each frame from its position in global space and allows direct blending of all data in the frames.

Then we compute a frame that contains the differences between the first and the last frame of the detected period. This frame is blended into the animation with weights ranging from -0.5 to 0.5, making the first and last frames match exactly. Finally, the motion is converted back to its original form (Equation 5.1). This processing ensures that minimal changes are made to the source animation, while extracting a precise period of the motion.

Unfortunately, even though the changes were minimised, the resulting motion can still contain certain artifacts. The most salient is footsliding, which is caused by the hierarchical nature of the blending method (see Figure 5.8). However, as the level of introduced footsliding is relatively small, it is possible to correct it using a standard inverse kinematics approach without introducing any further problems.

5.3.2. Trajectory Bone Embedding

High-level control of a character in a scene is usually provided by its trajectory, the velocity associated with each point on the trajectory, and (optionally) other data, such as motion constraints and contact

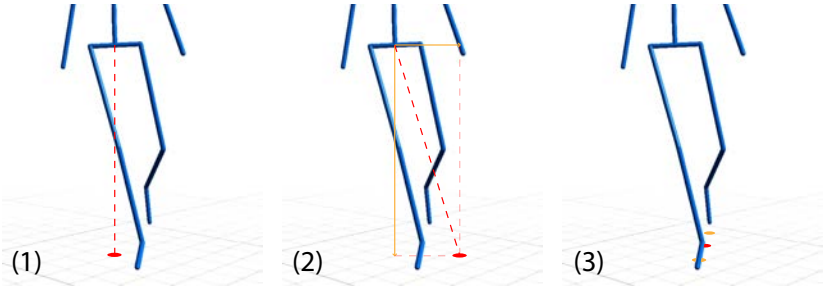

Figure 5.9.: An illustration of different trajectory extraction methods. Both the root projection method (1) and the Zero Momentum Point (2) can lead to a placement distant from the current feet position, which can lead to footsliding artifacts during blending. An average of the end-effector projections (3) provides a simple solution to this problem.

points. In order to simplify the connection of this information with the locomotion data, we introduce a *trajectory bone*, i.e., a representation of the idealised motion trajectory extracted from source motions, which is positioned on the ground under the character. The skeleton hierarchy and motion data are then updated so that the trajectory bone is placed at the root of the skeleton hierarchy, with the original root bone (usually positioned in the pelvis area) becoming its only child.

The trajectory bone movement is created in two steps. First, an approximation of the trajectory is extracted from the original motion. There are several possible approaches to achieve this – root projection, zero-momentum projection and end-effector trajectory (see below). Then, the extracted trajectory is idealised by fitting a circle into this trajectory and extracting its curvature and the motion speed.

The *root projection* approach is the simplest way of extracting the motion trajectory. By discarding the Y component of the root's motion, a flattened curve is projected onto the ground, describing the overall motion of the character. Unfortunately, this simple approach fails for fast turning motion, where the whole character is leaning sideways (a case that is relatively common in our source data). The projection point in that case is not under the feet of the character, but rather shifted left or right (see Figure 5.9), which leads to sideways footsliding when a transition from a straight walk to a turning one is needed.

Another method, inspired by physical simulation of a walking character, would use the *zero-momentum point*. This is a point, located under the feet of the character, with respect to which the reaction force of the foot/ground contact does not produce any horizontal moment (assuming a stable pose). Unfortunately, the source motion capture data are too noisy to produce a reliable estimate of this point.

The last tested option, which has proven to be the most reliable, averages the *trajectory of the end-effectors*. In the case of locomotion and a standard skeleton, the end-effectors correspond to four joints – left and right toe and heel. This leads to a very well behaved estimate of the final trajectory, which is also suitable for blending, as the character with the pose represented in respect of this trajectory leans around the point under his feet, providing an expected and natural behaviour.

5.3.3. Parameter Extraction and Trajectory Idealisation

In the next step, we extract the high level motion parameters and update the behaviour of the trajectory bone, leading to an idealised version of the trajectory. The chosen parameters should always be independent; for our pedestrian locomotion system, we have chosen tangent velocity and curvature, as they allow the intuitive description of trajectory shape and instantaneous velocity in the

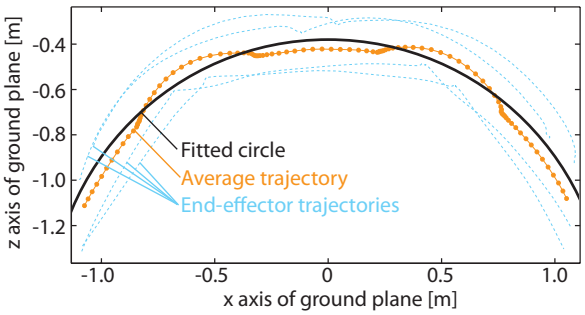

Figure 5.10.: The idealised trajectory extraction, demonstrated on a turning locomotion.

run-time system.

Instead of fitting a general curve to the trajectory (Kwon & Shin, 2005), we chose to follow the approach taken by Park et al. (2002) and fit a circle to the approximate trajectory, as its diameter can be directly translated into the curvature parameter. By projecting points on the trajectory onto the fitted circle, we can determine the average tangent velocity of the motion. Because the original method based on least-squares fitting proved not to be robust enough for our dataset (mainly for almost straight motions), our fitting method is based on creating a template trajectory from three motion periods. This involves fitting the circle to every corresponding combination of three points and computing the centre and diameter by averaging the fitted circles.

After extracting the parameterisation, the trajectory bone is updated to follow the idealised trajectory precisely and with constant speed, thus providing us with the motion abstraction we need in order to synthesise the resulting locomotion in the run-time system. If we now apply Equation 5.1 to the motion, the root (trajectory bone) differential transformation in each frame would be exactly the same. This property is used in the runtime system to create motion that is independent to global position.

Finally, we synchronise the start of every motion clip to a similar phase of locomotion. We begin with a frame selected from one of the animations on the basis of detected constraints (e.g., both feet on the ground) and find a matching frame in all animations in the set. By converting clips to the differential versions (Equation 5.1), reordering the frames to start with the desired frame by appending the preceding frames to the end of the animation, and converting the clips back to absolute format, we create a set of synchronised animations suitable for our run-time system.

5.4. Behaviour – Animation Interface

A typical single-character animation system would implement the full animation functionality by merging a higher-level behaviour/motion planning component with an animation synthesis system. However, crowd simulators often include a significantly more complex hierarchical behaviour system, with the main data flow from higher levels down. In most cases, feedback is directed up through the hierarchy, informing the higher levels about the constraints of the lower ones (e.g., reachability, blocked path, impossibility to reach the goal due to animation limits). This feedback should be limited and/or highly predictable, to avoid unnecessary restrictions on high-level behaviour models.

The animation synthesis algorithm and the low-level planning module usually addresses the lower levels of the behaviour hierarchy. In the same way, we aim to implement the physical and kinematic layers of the behaviour pyramid (see Figure 2.3 – the physical layer taking a trajectory as its input, returning the animation parameters, and the kinematic layer producing the animation frames.

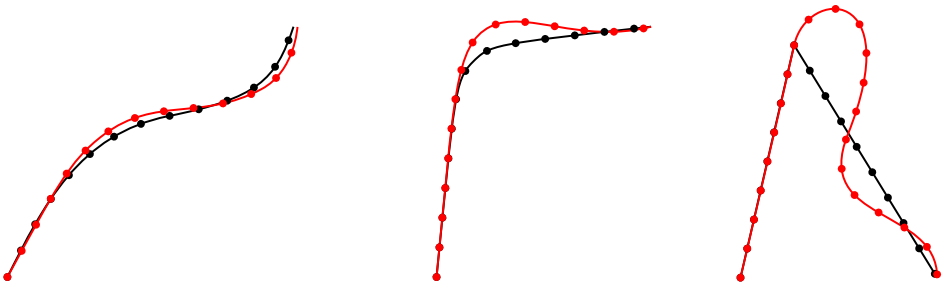

Figure 5.11.: Limited and damped PD controller (behaviour interface) path following examples, with the original trajectory depicted in black and the resulting smoothened version in red. If the path from the high level behaviour module is smooth, the generated trajectory follows it accurately (left). However, if it contains non-smooth segments (right) or overly sharp turns (middle), the PD controller follows it as closely as possible without violating the animation system limits.

A typical data-driven animation method, such as motion graphs, includes a high amount of semantic information, which serves as a base for building the working space of a planning module. Generating a specific animation in this space requires expensive optimisation methods (expensive either in terms of computational complexity or spatial requirements to store preprocessed data; see Section 2.5). Moreover, often a requested animation cannot be generated at all, but this fact cannot be easily determined before the full planning is finished.

In comparison, our system limits the planning module only by maximum speed and maximum curvature for a given speed (both described with a single polygon – the convex hull of the parametric space; see Section 5.5), and by maximum first derivatives of these parameters (i.e., how fast the parameters can change). Therefore, it is easy to evaluate if a path is plausible in a high level behaviour module, as any motion inside these intervals is synthesisable. To achieve a similar flexibility, previous methods employ expensive corrective techniques, altering the generated motion to reflect the requested parameters accurately (e.g., Park et al. (2002) use a footstep correction post-processing algorithm, Lee et al. (2010) require a graph-planning algorithm).

The input to the animation system from the higher level behaviour consists of a trajectory (orientation and position of a character in space at a particular time) and its associated instantaneous velocity. The velocity value is directly used as one of the parameters in the parametric space, whilst the curvature parameter is determined from two consecutive frames by determining the centre of rotation defined by two consecutive points and their corresponding tangent vectors on a trajectory. The resulting blending weights are determined by the position of these parameters in parametric space.

However, this approach still requires certain changes of the behaviour system (or a certain level of animation/behaviour feedback) in order to reflect the animation limits. To allow the use of a generic behaviour module, we use a damped PD controller as the behaviour system's lowest physical layer. The controller is used to follow the path generated by the higher levels as closely as possible, without violating the constraints. The limiting factors create integrative effects on the motion, while the controller's PD nature assures fast convergence towards the required trajectory (see Figure 5.11).

This method was practically implemented in the Metropolis system (see Figure 1.1), leading to highly realistic crowd animation with minimal artifacts. Its limitations are directly linked to the constraints of the animation synthesis – very sharp turns, if not explicitly represented in the source animations, cannot be generated (see Figure 5.11, right), leading to trajectory deviation. The speed outside the supported range leads to even more serious artifacts, such as characters lagging behind the expected position for speed above the maximum limit, and oscillation when the speed is too low

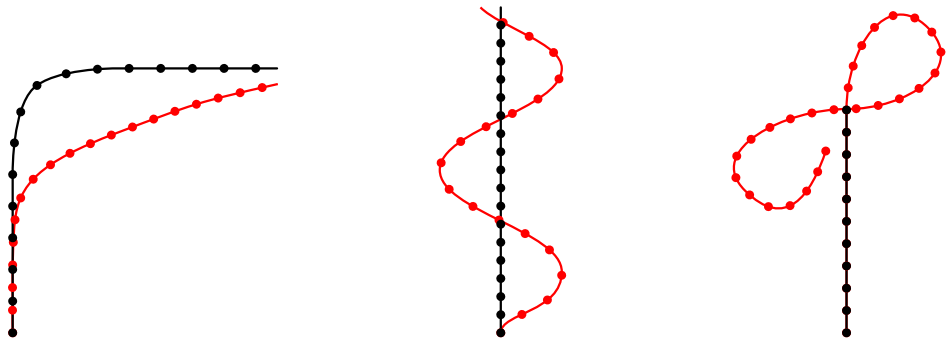

Figure 5.12.: The artifacts of the PD controller as a behaviour – animation interface. The requested speed is higher than maximum limit (left), leading to "lacking behind" artifact; lower than minimum limit (middle), with "oscillation" effect; or not moving at all (right), with "moving in circles" problem.

(see Figure 5.12).

These artifacts are the most serious drawbacks of this interface type, and the only way to address them is to extend the parametric space (please refer to Section 5.5 for its detailed description). If no additional animations are available or possible, this extension has to be implemented using data extrapolation. The most straightforward method is to generate new pseudoexamples using a motion editing method and add them to the parametric structure. However, this technique does not allow for responsive adjustments of the actual animation state.

A more responsive solution can be achieved using a motion correction method, e.g., in a similar spirit to Park et al. (2002). However, such real time adjustments tend to be computationally expensive. For extrapolation in the speed dimension, animation timewarping can be used to synthesise new animations with minimal computational cost. In Section 6.4, we provide a perceptual evaluation of such manipulation, which can also serve as a guide for time-based animation editing.

5.5. Parametric Space Concept

The core of our parametric motion synthesis model lies in the parametric space concept – a space partitioning structure allowing locomotion to be synthesised from input motion clips by motion blending and timewarping techniques. The structure itself is built from a point cloud, where every point represents the parameters of a motion clip.

A crucial step in locomotion parameterisation is the selection of suitable parameters, which are then translated to weights for the actual clip blending. Several options were explored in previous work (see Section 2.5.2), but by far the most common approach uses speed and angular velocity (Park et al., 2002, 2004; Pettré & Laumond, 2006). However, for the particular case of crowd animation, this is not the most suitable option.

In a crowd scenario, the behaviour system consists of several layers responsible for different levels of motion planning. Higher levels plan the overall trajectory shape and consequently influence mainly the side steering of the character. Lower levels implement the responsive behaviours, which uses the locomotion speed as its primary mechanism (i.e., character stopping or slowing down before an unexpected collision). This implies that the character's speed and the trajectory shape should be independent parameters of the locomotion model, allowing to fully decouple these two aspects of motion planning.

However, both *speed* (or *tangent velocity*, $[m/s]$) and *angular velocity* ($[°/s]$ or $[rad/s]$) are time-dependent parameters. Consequently, changing character's speed while keeping the angular velocity

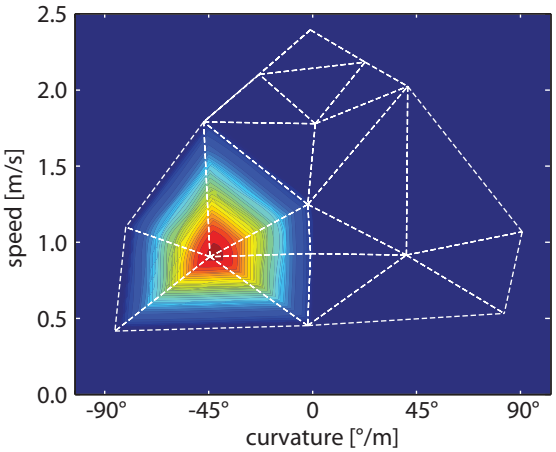

Figure 5.13.: The parametric space structure and the region of influence of one clip.

constant changes the trajectory shape (and vice versa). This is an undesirable property, as it requires readjustments of both parameters if either speed or trajectory changes.

In our system we use *speed* (or *tangent velocity*, $[m/s]$) and *trajectory curvature* ($[^\circ/m]$). Because only one parameter is time-dependent, this choice provides a more suitable and fully decoupled parametric description of local motion properties.

The idea of a parametric space is not novel – many previous works used a similar approach towards locomotion generation (see Section 2.5.2). Most of these methods employ a non-linear interpolation method, such as radial basis function (Rose et al., 1998) or cardinal basis function (Park et al., 2002, 2004; Kwon & Shin, 2005). While there are many advantages to using these methods (e.g., interpolation smoothness, extrapolation capabilities), they come with several serious problems.

First and foremost, they are non-local – for each point in the parametric space, all the source clips have to be blended with different weights (Kovar & Gleicher, 2004). This is an issue especially for highly optimised systems, as the number of clips can be relatively high (15 in the example in Figure 5.13) and blending all of them presents a significant computational overhead. The cardinal basis function can also contain negative weights, which do not have any physical meaning (thereby reversing the source clip). Finally, the non-linearity causes unpredictable results – an animation generated from a datapoint half way between the clips with speeds of $0.5m/s$ and $1.0m/s$ does not yield an animation with speed $0.75m/s$. For this reason, some of the later methods, as well as some of the earliest, employ linear K-nearest neighbour interpolation (e.g., Lee et al. (2010)).

Because our parametric space is 2-dimensional, we can partition it using Delaunay triangulation. Employing barycentric coordinates, we can then represent every point inside the convex hull of the original pointcloud as a linear combination of three original points, giving us weights to be used for motion blending (in a similar manner as Pettré & Laumond, 2006). Because any trajectory (within the speed / curvature limits) can be described in these terms, we are able to synthesise the corresponding motion precisely.

In mathematical terms, each animation sequence S_a is a triplet of curvature c_a, speed s_a and clip length l_a:

$$S_a = (c_a, s_a, l_a)$$

The c_a and s_a parameters describe the 2D position (c_a, s_a) of each source clip in the parametric space. To synthesize an interpolated clip $S_i = (c_i, s_i, l_i)$, we first determine the triangle of the parametric

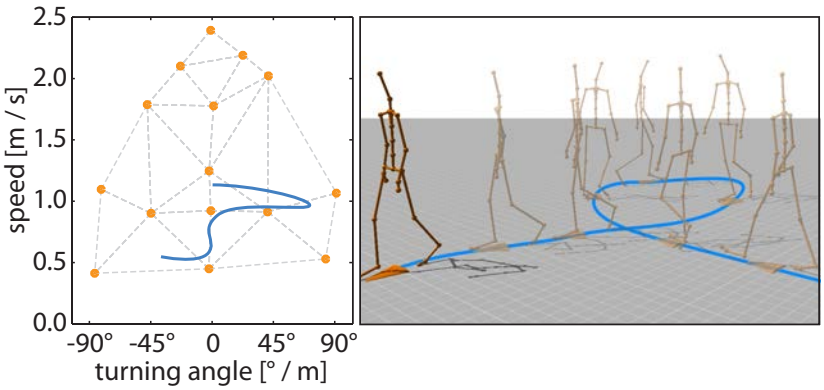

Figure 5.14.: An example motion clip (right) generated by a sequence of parameter changes inside the parametric space (left).

space containing its datapoint (c_i, s_i). The vertices of this triangle describe three motion clips S_1, S_2 and S_3; the datapoint position provides three barycentric coordinate values λ_1, λ_2 and λ_3:

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$
$$0 \leq \lambda_1, \lambda_2, \lambda_3 \leq 1$$
$$\lambda_3 = 1 - \lambda_1 - \lambda_2$$

The curvature parameter c_i is therefore computed as:

$$c_i = \lambda_1 c_1 + \lambda_2 c_2 + \lambda_3 c_3$$

and the target speed as:

$$s_i = \lambda_1 s_1 + \lambda_2 s_2 + \lambda_3 s_3$$

The barycentric coordinate values also determine the weights for motion clip interpolation (denoted in Section 5.6 as $u^{(i)}$).

However, for periodic clip synthesis, we need to ensure that the interpolation is performed on frames with corresponding poses. While more sophisticated schemes can be used (e.g., dynamic time warping (Park et al., 2002), registration curves (Kovar & Gleicher, 2003)), under the assumption that the differences between clips are not large, a simple linear timewarping scheme suffices. The advantage of this simple scheme is its clip data independence and consequently predictable behaviour. The clips are therefore retimed to have a common length l_i (leading to no change if $l_1 = l_2 = l_3$).

A naive way of computing a common length l_i would be to use the same interpolation scheme as the other clip parameters:

$$l_i = \lambda_1 l_1 + \lambda_2 l_2 + \lambda_3 l_3$$

However, the timewarping manipulation of a clip also changes the overall speed of its animation:

$$s_1' = s_1 \frac{l_1}{l_i}$$

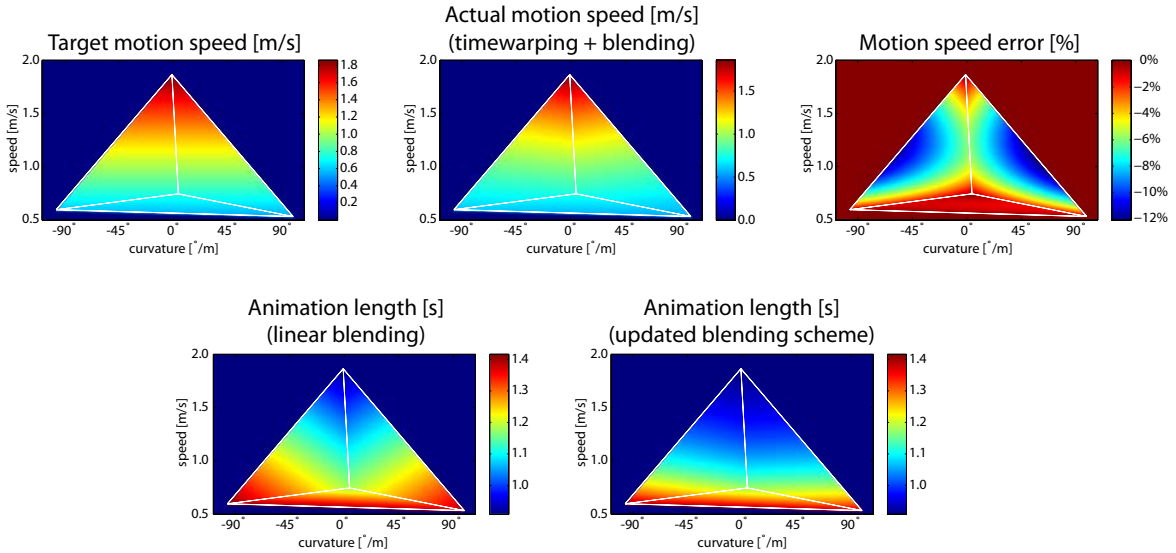

Figure 5.15.: The inaccuracies caused by the linear blending scheme (illustrated on a small parametric space with four motion clips) and the corrective step that alters the length interpolation method.

leading to the actual interpolated speed of:

$$\begin{aligned} s'_i &= \lambda_1 s'_1 + \lambda_2 s'_2 + \lambda_3 s'_3 \\ &= \lambda_1 s_1 \frac{l_1}{l_i} + \lambda_2 s_2 \frac{l_2}{l_i} + \lambda_3 s_3 \frac{l_3}{l_i} \end{aligned}$$

which describes a non-linear relation between the blending weights and the resulting speed. A standard solution, often used for linearisation of example-based inverse kinematics, would be to alter the blending scheme. However, that would also influence the curvature parameter, and a local linearisation and/or pseudoexample registration would be necessary (Lewis et al., 2000).

Our scheme, though, allows the timewarping to be changed independently by altering the length of the blended result l_i. Therefore, to achieve the speed match, we can use:

$$\begin{aligned} s_i &= s'_i \\ s_i l_i &= \lambda_1 s_1 l_1 + \lambda_2 s_2 l_2 + \lambda_3 s_3 l_3 \\ l_i &= \lambda_1 \tfrac{s_1}{s_i} l_1 + \lambda_2 \tfrac{s_2}{s_i} l_2 + \lambda_3 \tfrac{s_3}{s_i} l_3 \end{aligned}$$

which leads to the correct speed of the interpolated motion (see Figure 5.15).

The final step in the real-time process is the generation of the actual motion based on the computed position in the parametric space, performed on per-frame basis.

To implement our blending scheme described above, we first normalise the time scale of all animation clips to a common uniform time $t_u \in [0...1)$. This approach is inspired by procedural locomotion generation, where it primarily allows to link step length, locomotion speed and time (Boulic et al., 1990, 2004). The per-frame time update rule, integrating the between-frames time difference into the uniform time value t_u, can then be computed using:

$$t_u(t + \triangle t) = \left(t_u(t) + \frac{\triangle t}{l_i} \right) \; mod \; 1.0$$

where t describes integrated real time value, $\triangle t$ the time difference between previous and current

generated frame and l_i the length of the blended animation.

The frame blending can be based on the quaternion algebra interpolation scheme (normalised linear interpolation), but our model's properties allow us to use a simplified linear version of blending (see Section 5.6 for more details). The trajectory bone, which is placed at the root of the skeleton hierarchy, needs to be handled separately in order to minimise deviations from the required trajectory. In our system, the position and orientation of the trajectory bone corresponds to the position of the trajectory in the current timeframe, thus ensuring that it is followed precisely.

5.6. Linearised Animation

When animating crowds, the first bottleneck that needs to be addressed is caused by character skinning, which transfers the motion from underlying structure to the character mesh. Even though the most commonly used method, Skeletal Subspace Deformation (SSD), is fully linear and very efficient, a bottleneck results from the large number of vertices that need to be updated. With modern graphics hardware, however, this can be addressed by using the GPU to perform these computations, as its massively parallel architecture is particularly suitable for this task.

The second bottleneck related to animation arises when the number of displayed characters reaches thousands. Classical approaches, which use IK, inverse dynamics or graph search/optimisation techniques, do not scale well and cannot be used for a large portion of the crowd (with the exception of the characters closest to the camera – the highest LOD). The middle LOD is our target, and thus we do not aim to provide the most realistic and physically correct motion. Instead, our solution needs to be fast, simple and reasonably accurate, thereby allowing us to animate as many characters as possible.

In this section we describe and justify the use of linear methods for skeletal animation blending. This is an important part of our animation system, but the methods are not limited to the simple locomotion case we present. Our blending process could also be adapted to accelerate and enhance methods like Fat Graphs (Shin & Oh, 2006), Parametric Motion Graphs (Heck & Gleicher, 2007) or many other animation techniques based on motion interpolation.

Character rendering routines require skinning transformations, i.e., bone transformations from the rest pose to the animated pose, as their input. The standard way of computing these transformations is as follows: Let us assume that $\pi(i)$ denotes the parent bone of a bone with index i, and $\boldsymbol{R}_i$ is the relative transformation matrix between the bones i and $\pi(i)$ (as specified by the binding pose). Matrices $\boldsymbol{A}_i$ (which describe the binding pose in the world space) of the individual bones in the object coordinate system are thus given as:

$$\boldsymbol{A}_i = \boldsymbol{R}_1 ... \boldsymbol{R}_{\pi(i)} \boldsymbol{R}_i \tag{5.2}$$

These matrices do not change during animation and can therefore be precomputed. Their counterparts in the animated skeleton are matrices $\boldsymbol{F}_i$, which represent animated transformations of the character's bones in the world space:

$$\boldsymbol{F}_i = \boldsymbol{R}_1 \boldsymbol{T}_1 ... \boldsymbol{R}_{\pi(i)} \boldsymbol{T}_{\pi(i)} \boldsymbol{R}_i \boldsymbol{T}_i = \boldsymbol{F}_{\pi(i)} \boldsymbol{R}_i \boldsymbol{T}_i \tag{5.3}$$

where $\boldsymbol{T}_i$ are the animated bone transformations with respect to the binding pose. The final skinning matrices $\boldsymbol{C}_i$ are then given as:

$$\boldsymbol{C}_i = \boldsymbol{F}_i (\boldsymbol{A}_i)^{-1} \tag{5.4}$$

Animation systems traditionally only work with the bone transformations $\boldsymbol{T}_i$; the skinning transformations $\boldsymbol{C}_i$ are computed in the rendering stage by matrix multiplication as described above. This

Algebra / operation	+, -	*, /	fn
Conversion to $\boldsymbol{C}_i$ matrices (i.e., playing the animation			
Quaternions	129	102	0
Euler angles	123	106	0
Matrix interpolation	0	0	0
Interpolation between n transformations + conversion			
Quaternion NLERP	$4n + 128$	$4n + 107$	1
Euler angles	$3n + 120$	$3n + 106$	6
Matrix interpolation	$12n - 12$	$24n - 24$	0

Table 5.1.: The cost of generating one $\boldsymbol{C}_i$ matrix from different representations, assuming that input animations were converted to a particular representation in preprocessing step. '+, -' denotes the total number of additions, ', /' the number of multiplications and divisions, and 'fn' the number of more complex functions such as goniometrical functions and square roots.*

method ensures that bone lengths (specified by matrices $\boldsymbol{R}_i$) stay constant, as long as transformations $\boldsymbol{T}_i$ are rotations (scaling/shearing bone transformations are not usually used, and the translation component is zero with the exception of the root bone transformation $\boldsymbol{T}_1$). Unfortunately, the matrix concatenations, and especially the rotation interpolation required for animation blending, may slow the system down when the number of animated characters is large.

A natural description of algorithmic complexity is the total number of required mathematical operations, separated into groups according to their computational cost.

The common interpolation methods, i.e., Spherical Linear Interpolation of Quaternions (SLERP), Normalised Linear Interpolation of Quaternions (NLERP) and Euler angles all work in the local space of $\boldsymbol{T}_i$ transformations. This provides the advantage of keeping the bone lengths constant, but requires three matrix multiplications on top of the interpolation and conversion cost to compute the $\boldsymbol{C}_i$ skinning matrix. Each of these three methods has its own advantages and disadvantages. Quaternion SLERP is the most accurate blending method, keeping the angular velocity constant and using the shortest possible interpolation path, but it is defined only for two orientations. Quaternion NLERP has similar properties to SLERP, is defined for any number of orientations, but does not keep the angular velocity constant. Euler angles are the simplest, linear, commutative and most intuitive rotation representation, but their interpolation neither keeps the angular velocity constant, nor uses the shortest interpolation path. This can result in severe interpolation artifacts even for small angular differences, especially when angles are close to singularities.

Our method of choice is the **linear interpolation of $\boldsymbol{C}_i$ matrices** (with the $\boldsymbol{C}_i$ matrices for on-spot animations precomputed in a preprocessing stage):

$$\boldsymbol{C}_i^{(result)} = \left(\sum_j u^{(j)} \boldsymbol{C}_i^{(j)} \right) \boldsymbol{A}_1$$

with $\boldsymbol{C}_i^{(result)}$ denoting the final transformation of bone i, $u^{(j)}$ the blending weight for animation j, $\boldsymbol{C}_i^{(j)}$ the skinning tranformation of bone i in animation j and $\boldsymbol{A}_1$ the trajectory bone transformation in the current frame. Even though this approach does not satisfy the bone rigidity constraint (the transformations are interpolated in a global (world-space) coordinate system), for a small number of interpolated transformations it provides a significant improvement in overall computational cost (see Table 5.1). Note that common motion synthesis methods require only about 2-3 motions to be blended at any given time.

Even though the problem with non-rigidity of bones that arises with linear matrix interpolation

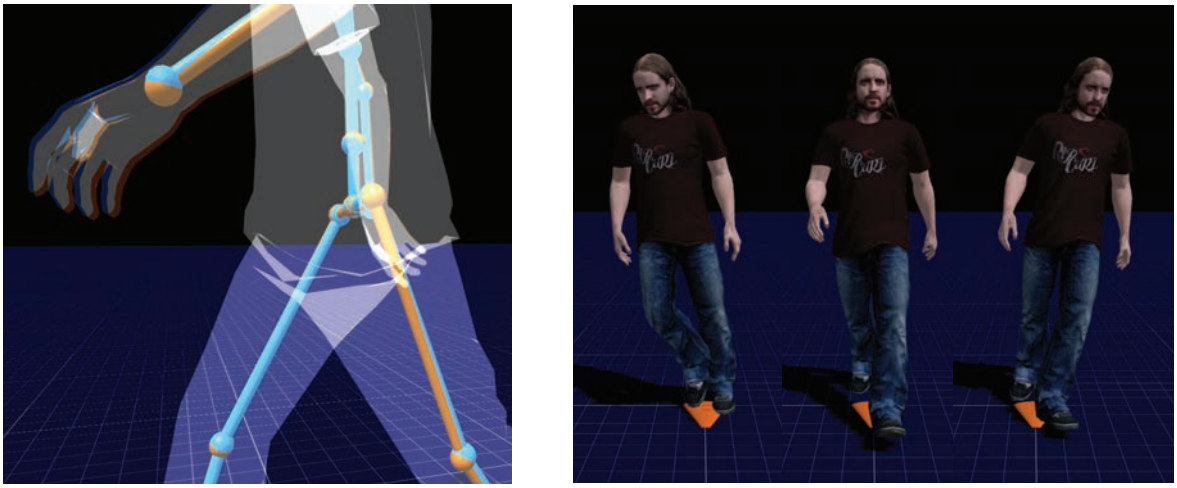

Figure 5.16.: Left: Demonstration of errors introduced by our blending method (orange – spherically interpolated motion, blue – proposed linear interpolation). Right: An illustration of the differences between motion clips with different turning parameters.

may appear to be significant, we argue that for our purposes this is not an issue. As shown by Harrison et al. (2004), the bone length changes are below perceptual limits for length differences of up to approximately 19% (for shrinking a bone), which leaves us with the possibility of blending linearly between two joints with an angular difference of up to 62 degrees. Blending between animations in frames that contain larger angular differences would be impractical, as it would lead to artifacts even in standard blending methods (e.g., foot sliding or velocity artifacts).

Another argument can be built on frame distance metrics, which calculate the difference between two frames as a scalar value (or a low dimensional vector). Distance metrics are used in many animation techniques to find a sequence of frames suitable for blending and/or transitioning between two animations. The most common approach is to use the Euclidean distance of feature vectors, with features defined either in kinematic space (i.e., $\boldsymbol{T}_i$, quaternions or quaternion logarithms) or Euclidean space (i.e., $\boldsymbol{F}_i$, object space joint positions, or positions of vertices of the driven mesh). For large angular differences between two joints, which would result in visible artifacts with our blending method, these metrics would return very large values, suggesting that the transition/blend between the animations in this frame will not lead to natural results.

To reduce artifacts when blending between very different animations, we can create new example clips by blending between two very different clips at preprocessing time (as suggested by Zhao & Safonova (2009)). Because this step is performed off-line, we can even afford to use expensive methods such as quaternion SLERP and inverse dynamics.

Apart from the issue of computational complexity, the linear combination of $\boldsymbol{C}_i$ matrices as an animation method has other interesting properties. An example, which is very useful for the locomotion engine described in this paper, is the ability to blend between two locomotion animations with different speeds and step lengths but similar relative timing (which can be assured using timewarping). In each animation, the movement of the root relative to the ground and the movement of a planted foot relative to the root, are the same except for their opposite directions. When performing the blending linearly in Euclidean space, every resulting animation will have the foot planted, independently of the blending weights (because the resulting blended relative displacements between the root and the ground, and between the planted foot and the ground, will be the same). Performing the same blend

in kinematic space and/or by non-linear methods will always result in a slight difference between these two motions, thus causing footsliding artifacts. Kinematic blending methods have to apply inverse kinematic/dynamic techniques to achieve the same result. In our method, this is achieved intrinsically, at the price of slightly non-rigid bones in the underlying skeleton.

Mesh animation describes the motion of a polygonal mesh using the trajectories of its vertices. The skinning methods, such as the most common Skeleton Subspace Deformation (SSD, linear blend skinning), also represent a form of mesh animation. The difference is that there is an underlying skeletal structure driving the skinning process, which can be non-linear, and which also introduces dependencies between the vertices of the mesh. To distinguish between this structured representation and true mesh animations, we will use the term mesh animation only for animations consisting of a set of *independent trajectories* of individual vertices (belonging to a mesh object with fixed connectivity). The only remaining structural information is the index of each vertex, identifying its place in the mesh description, and the polygons (triplets of vertex indices).

Because the subject of this thesis is crowd simulation, mesh animation is mainly interesting for providing the clothes for the crowd. The simulation of cloth in real time is very expensive, and for a large group of characters we would prefer to use blends of pre-simulated results.

Many different methods can be used for blending between two mesh animations applied to the same polygonal mesh. Gain & Bechmann (2008) provide a survey of different mesh deformation methods, which can be used to deform one animation (or the space it resides in) to match the other, thus providing the means to blend between them. Another group of methods includes those that use the polygonal information and create blends that ensure minimal deformation of the polygonal structure (English & Bridson, 2008). While all these techniques can provide very good results, they are computationally expensive. Linear blends between mesh animations are very simple and therefore appealing for a crowd system. However, a naive combination of this type of blending with skeletal animation on the underlying character mesh would lead to problems with interpenetration. We will now show that our linear animation blending method is compatible with linear blending between mesh animations.

The simplest and most common skinning method used for skeletal animation is **skeletal subspace deformation** (SSD; see Lewis et al. (2000) for a detailed description). It uses the skinning matrices $\boldsymbol{C}_i$ to transform the vertices of a mesh from their binding position $\boldsymbol{v}_n$ to a new animated position $\boldsymbol{v}'_n$ (both in the world space) using:

$$\boldsymbol{v}'_n = \sum_{i=1}^{N} w_{i,n} \boldsymbol{C}_i \boldsymbol{v}_n \tag{5.5}$$

where N is the total number of bones, $\boldsymbol{C}_i$ represents the skinning matrix corresponding to bone i and $w_{i,n}$ are the skinning weight parameters. Each $w_{i,n}$ parameter is a scalar value in the interval [0..1], all weight parameters for a particular vertex sum to 1 and generally there is a limited number of non-zero weight parameters per vertex (a common number is 4).

A frame of an animation sequence a (from a set of animations A) in our representation consists of a set of $\boldsymbol{C}_i^{(a)}$ matrices, one for each bone i. The blending equation, combining a set of animations with weights $u^{(a)}$ (each a scalar value in the interval [0..1] and summing to 1), and producing the final skinning matrix $\boldsymbol{C}_i$ would be:

$$\boldsymbol{C}_i = \sum_{a=1}^{A} u^{(a)} \boldsymbol{C}_i^{(a)}$$

The combination with skinning equation (5.5) yields the equation for the final transformed vertex

Figure 5.17.: A combination of a pre-simulated mesh animation with a skeletal animation. Pre-simulated clothes, allowed by our linearised animation system (left) provide better deformation than the skinning method (right).

position $\boldsymbol{v}'_n$:

$$\begin{aligned}\boldsymbol{v}'_n &= \sum_{i=1}^{N} w_{i,n} \sum_{a=1}^{A} u^{(a)} \boldsymbol{C}_i^{(a)} \boldsymbol{v}_n \\ &= \sum_{a=1}^{A} u^{(a)} \sum_{i=1}^{N} w_{i,n} \boldsymbol{C}_i^{(a)} \boldsymbol{v}_n \\ &= \sum_{a=1}^{A} u^{(a)} \boldsymbol{v}_n^{(a)}\end{aligned}$$

where $\boldsymbol{v}_n^{(a)}$ describes a vertex $\boldsymbol{v}_n$ transformed by a frame from animation a. Therefore, the resulting vertex position is a linear combination of meshes animated by the original skeletal animation sequences. This allows us to combine mesh animation and skeletal animation, which would not be possible with hierarchical and/or kinematic blending methods, as their non-linearities would lead to inconsistencies between mesh and skeletal results (i.e., intersections and collisions between the character and the linearly animated cloth).

We use this method to combine pre-simulated cloth animation with the skeletal animation system, giving us the means to dress our crowd characters in mesh-based clothes and to blend between their animations in the very same way as we blend between the skeletal poses.

5.7. Runtime Performance

We have tested our animation method and the locomotion system in an isolated environment to avoid including unrelated computation costs (e.g., rendering, high-level behaviour model). In each test we generated one million blended frames for a character with 22 bone segments. In the blending speed tests, we randomised the indices of the frames used for blending, while for the animation system tests, we generated a random trajectory with continuous parameter changes. The results of our experiments can be found in Figure 5.18.

The costs of **frame blending** on actual hardware correspond relatively well to our theoretical results (see Section 5.6), showing significantly longer times for traditional non-linear blending methods (Figure 5.18, middle). The difference is even more pronounced when we include the cost of conversion into skinning matrices (Figure 5.18, right). Using our method for simple blending, a speed-up factor of up to 8.3 is achieved for the computation of skinning matrices when blending between two frames and up to 7 when interpolating three animations.

The **runtime animation system** uses several operations to compute each frame (see Section 5.4).

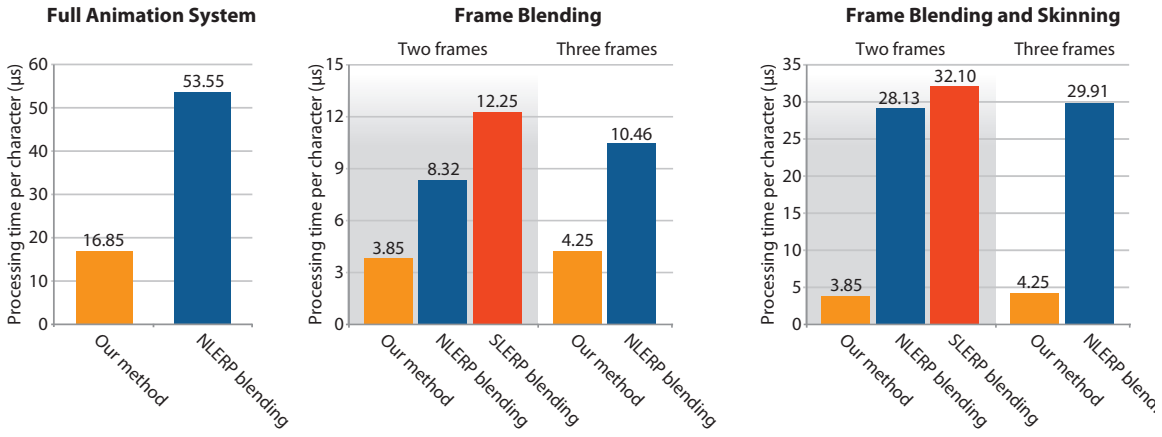

Figure 5.18.: Results of the performance tests of our blending method and the resulting animation system. Each character consists of 22 bones, which is typical for a real-time system. Results are based on one million frames of animation and expressed in terms of time needed to generate one frame of skinning animation for one character.

The final frame is generated by blending three frames belonging to animations indexed by a triangulation of parametric space. Each of these animation frames is created by blending the two keyframes of the input animation closest to the current normalised time value. This blending method avoids any jittering artifacts while still keeping the number of necessary operations low. Because of the large number of operations involved, the performance impact of the conversion into skinning matrices is lower, but our system still achieves a speed-up of 3.62 over a system using traditional quaternion blending methods (see Figure 5.18, left).

Another test demonstrated that the blending performance does not depend on data coherence. A sequential frame generation test for one character moving through space in a coherent way performed almost exactly as well as a fully randomised test generating random frames for a crowd of 1000 characters (16.849μs and 17.105μs per character, respectively). This property holds not only for our system, as a corresponding result is obtained for Normalised LERP blending (53.550μs and 53.887μs).

6

Perceptual Studies

Human perception is an area of research, that is closely related to human animation synthesis due to the fact that all synthesised motions are ultimately presented to a human observer. The corpus of previous work in this field (see Section 2.7) contains a number of perceptual studies focused on humanoid animation, leading to significant optimisations of the animation synthesis process.

The following chapter describes several perceptual studies aimed at different aspects of human locomotion synthesis. Their main aim is directed towards properties of the algorithms described in previous chapters, their extensions and optimisations. However, each experiment is designed to be generalisable to other methods of data-driven motion synthesis, making their results independently useful.

The subjects addressed in the next four sections can be summarised as follows[1]:

- Section 6.1 describes the creation of a perceptually-based locomotion metric, where results from a perceptual experiment are used to adjust and compare the properties of several metrics presented in previous work. A generic motion comparison metric supported by perceptual evidence would have broad use in data driven animation methods; in the context of this thesis, it can be used primarily for building motion maps (see Section 5.2).
- In a crowd simulator, it is important to know how many different examples of human motion are required to make the crowd look varied. In our experiment (Section 6.2), depicting a crowd scenario, we determined that the number of different characteristic animations necessary to give the illusion a varied crowd was three. This has a direct impact on the amount of memory needed to store motion data, i.e., we need to create only three parametric spaces (see Section 5.5) to animate a crowd.
- Footskating is the most common artifact in data-driven locomotion generation methods. In Section 6.3, we evaluate its saliency and the perceptual impact of its clean-up. Our findings suggest

[1] Please note that experiments in this section are not in chronological order, and were both conducted and presented as separate and generic studies, independent of the technical work presented in other chapters of this thesis.

that footsliding is indeed a perceptually salient artifact and that a simple length-based clean-up method can be more successful in removing this artifact than some of the more comprehensive ones. This closely relates to several parts of this thesis:
  - the accurate constraints detection mechanism (Chapter 4),
  - the footskate removal methods (Section 4.4) and the footsteps in our locomotion synthesis system (Chapter 5),
  - and the lengthening blending scheme, used in conjunction with our locomotion synthesis (see Section 5.6).
- Finally, in Section 6.4, we evaluate the effect of linear timewarping on the perceived naturalness of an animation. We find that speeding-up the animation has a significant impact on the perceived naturalness of the motion while even significantly slowed-down motions are still recognised as natural. This finding is important for many data-driven motion synthesis methods, can provide interesting insights into the timewarping curve generation, and also allows a simple motion extrapolation for our parametric locomotion model.

Unless explicitly stated otherwise, the animation data used in our experiments were captured using our Vicon motion capture system (see Chapter 3), with a markerset consisting of 55 markers (see Section 3.4). The motions are processed in the Vicon IQ system and converted to be periodic in our pipeline (see Section 5.3.1) and rendered in real-time using the OpenGL rendering system. The specific animation processing steps are described in detail in the *stimuli* section of each experiment.

6.1. A Human Locomotion Comparison Metric

Underpinning most data-driven methods of human animation is a metric, which allows the similarity between two motion clips and/or short segments of input motions to be determined. Many other metric types exist, that allow the animations to be classified into groups, or their specific properties to be determined (see Section 2.6). At the lowest level, though, two frames describing two character poses are compared, with an extension to short motion segments achievable by simply averaging the similarity values of corresponding frames of the input segments (thereby accounting for changes of the motion as well, which is similar to comparing derivatives of the pose changes (Kovar et al., 2002a)). For longer sequences, a timewarping technique is required to align the clip segments in the time domain (Kovar & Gleicher, 2004).

In this section, we present a metric based on the difference between two frames (poses) of human locomotion, which is directly derived from the results of a perceptual experiment. The narrow focus of the metric (human locomotion) is necessary for two reasons: first, it might be possible that metrics for different motion types would need different parameters, thereby requiring a classification to take place before the comparison. Second, as we want to base our metric directly on a perceptual experiment, for experiment design we prefer a relatively small selection of clips of one type.

The practical use of such a metric is mainly in data-driven locomotion synthesis, which is also the main theme of this thesis. With a perceptually-driven metric, we can take into account only perceptually salient features of the motion in the motion map building process (see Section 5.2). The approach we use is not built from scratch, but rather uses previously introduced metrics, adjusts their parameters, evaluates their efficiency and alters them to provide more perceptually correct results.

Evaluation and adjustments of a metric is of course not a new topic. Wang & Bodenheimer (2003) alter the parameters of the metric introduced by Lee et al. (2002) (a metric included in our basic metrics set as well) using a set of classified motion examples and a minimisation technique. They also provide a perceptual evaluation of their results. Van Basten & Egges (2009) provide an evaluation

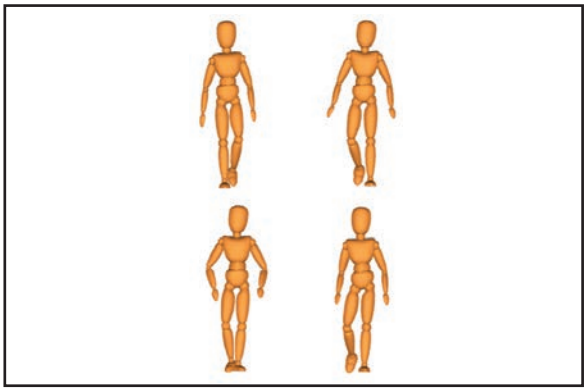

Figure 6.1.: Layout of the locomotion metric perceptual experiment. Users were asked to select two of the four shown motions, which they felt were the most similar.

of several metrics, using both mathematical methods and perceptual experiments. One conclusion of particular relevance to our work is that the shape of a perceptually-driven metric is highly non-linear. For more details about other methods please refer to Section 2.6.

The contribution presented in this section can therefore be summarised as follows:

- an approach for forming frame comparison metrics based on perceptual experiments,
- a metric created using this framework, describing the difference between locomotions of different subjects,
- an evaluation of the most common analytical frame comparison metrics and their suitability for expressing perceptual metrics, and
- an evaluation of different shaping functions (Euclidean, Manhattan and sigmoid) with respect to perceptual results.

6.1.1. Perceptual Experiment

Our motion capture database used for this experiment contained walking motions of twenty one walkers (14M and 7F). The motions were captured using a 13 camera Vicon optical system (see Chapter 3), with 43 markers placed on the body of the subject, and a capturing framerate of 100 FPS (Frames Per Second). Each walker was instructed to walk up and down along a straight line in our capture area until they felt comfortable and were walking at their natural pace. We then recorded a number of walks from each of them.

Captured data was then retargeted to a neutral wooden mannequin figure and converted into periodic clips by blending between two periods of the input motion (see Section 5.3.1). The period was detected by cross comparing the frames of the motion and detecting the minima of the resulting motion map (see Section 5.2.3). Because both periods of the motion are nearly similar, the resulting blending curve is always a straight line (see Figure 6.4). By testing several different metrics, we found that all of them performed similarly (for this particular purpose), which meant that the type of metric used in this stage was unimportant.

Twenty-nine participants (18M, 11F) took part in our perceptual experiment. All participants were naive to the purpose of the experiment and had normal or corrected to normal vision. A matrix of 4 front-facing characters was displayed using orthogonal projection, with each character's motion different from the others and randomly chosen from a set of 21 straight walking locomotions (see Figure 6.1). The orthogonal projection ensured that the characters are not translating horizontally on the screen, while it did not affect the characteristic side motion of the walks. Participants were asked

to select two of the motions that they felt were most similar, by clicking a mouse. This n-alternative forced choice design was inspired by the animation experiment by McDonnell et al. (2008). However, in contrast to their method, the number of simultaneously displayed characters was reduced from 9 to 4, significantly simplifying the task (the pair selection from a matrix of 3x3 characters proved to be too difficult).

Sixty-three trials in total were shown to each participant, each allowing 10 seconds for the selections. If the participant did not answer in the allotted time, additional set of motions was displayed. The 10 second time interval was chosen to avoid a long and detailed examination of the motion by participants, as we were more interested in their immediate reaction. The number of remaining trials was displayed on the screen between each trial, which guided participants to fixate on the screen centre. For every trial, we recorded which motions were displayed and the selected pair.

Evaluation

The recorded results were then converted to a table, where the value in each cell represented the perceived difference between the two motions indexed by row and column number; for our 21 clips, this leads to a table with a total number of 441 cells. Each cell is filled with a ratio $r(i,j)$ of the number of times the two animations i and j were **not** identified as similar to the overall number of trials where these two animations were displayed together

$$r(i,j) = \begin{cases} i \neq j & 1 - \frac{\#_{similar}(i,j)}{\#_{total}(i,j)} \\ i = j & 0 \end{cases} \tag{6.1}$$

where $\#_{similar}(i,j)$ is the count of the cases when the animation pair (i,j) was identified as similar and $\#_{total}(i,j)$ is the number of times this pair was shown together (the two values recorded during the experiment). The diagonal elements are filled with zero values, as there were no trials where a single clip would be simultaneously shown twice.

To illustrate the data obtained from each trial, we can denote the four simultaneously displayed motions as $m_1 ... m_4$, with motions m_1 and m_2 selected as similar. From the user's input, we create two similarity samples – (m_1, m_2) and (m_2, m_1), and 12 simultaneously displayed samples ((m_1, m_2), (m_1, m_3), (m_1, m_4), ... (m_4, m_1), (m_4, m_2), (m_4, m_3)). This illustration shows that the final function is *symmetric*, while the *positiveness* and *identity values* of the result are determined by equation 6.1. The last condition required for a metric definition (see Section 2.6.1) is the *triangle inequality*, which can be evaluated algorithmically by testing all possible combinations – in our data we found out that it is fulfilled for 98.6% of all possible combinations, with the remaining 1.4% considered to be noise. This analysis allows us to consider our data as a *metric function*.

Further analysis of the results showed several interesting properties of this table. To perform this analysis, we can interpret its rows/columns (because the matrix is symmetric, there is no difference between these two options) as n points in n-dimensional space (in our case $n = 21$). By applying Principal Component Analysis to this data and sorting the walks by the first component value, we obtain an ordering that makes a clear distinction between the actors' genders. In the same manner we can apply the K-means clustering algorithm with two clusters on our data, which results in two distinctive groups based on the gender of the walkers (see Figure 6.2). This shows an interesting property of the experiment results – the direction of the highest variance in the data is linked with the gender of the walkers. Even though the importance of gender in motion difference was shown in previous work (Kozlowski & Cutting, 1977; Barclay et al., 1978; Cutting, 1978; Troje, 2002; McDonnell et al., 2007a), the direct relation of gender with the direction of the highest variance in locomotion

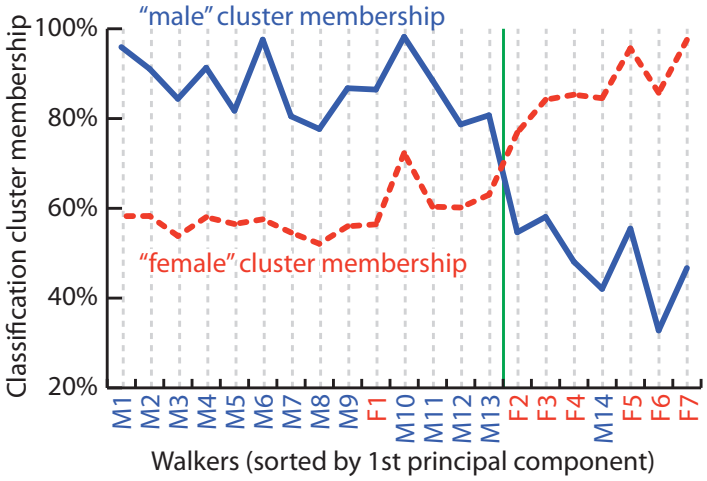

Figure 6.2.: Results of the classification performed on the data obtained in the perceptual experiment. The "male" (blue) and "female" (red) membership functions show the natural division of the motions by the gender of the actor. The order of actors on the horizontal axis is given by their projection onto first principal components of the source data. The green line shows the division line between the "male" and "female" groups.

difference recognition is, to our knowledge, a new finding.

This relation fails in two cases - female F1 and male M14 (Figure 6.2). The incorrect classification of male actor M14 can be explained by his age, as this was the youngest male actor captured. Misclassification of certain female actors is a phenomenon shown in several previous works (Kozlowski & Cutting, 1977; McDonnell et al., 2007a). Moreover, our dataset was also used in one of our previous studies (McDonnell et al., 2007a), where the same walker was consistently marked as a male as well. Therefore, we can still assume that from the perceptual point of view, this classification is correct.

A graphical representation of the values in the source table, sorted by the first principal component, is shown in Figure 6.3. The colours of the related motions appear darker, showing their lower difference and close relationship (with black on the diagonal meaning no difference), whilst light colours signify more distant motions.

6.1.2. Motion Metrics Toolkit

With the results of the perceptual experiment, we can try to define analytical metrics to reproduce the perceptual results as closely as possible. Instead of creating a metric from scratch, we choose to alter parameters of existing metrics using an optimisation technique. By comparing how the metric fits, we can also test the ability of each metric to represent the perceptual results. To reduce the number of optimisation parameters, we separate each metric into a weighted average of bodypart metrics. This step generalises the computed weights to any humanoid body structure, independent of the actual number of joints in each bodypart, and allows us to determine the perceptual saliency of the motion of each of them.

To ensure the generality of our method, the metrics have to be invariant to global position and heading of the character. In the same spirit as Kovar et al. (2002a), we discard the XZ position and Y orientation of the character, thereby effectively converting each motion to be on-the-spot.

The **primitive metrics** used in our toolkit are used on single joints of the skeletal hierarchy. Each of them is based on previous work and focuses on a different aspect of the joint pose or motion.

The *angular difference* metric measures the joint orientation distance (with respect to the parent bone) as a difference between two quaternions (Lee et al., 2002). A naive implementation would use

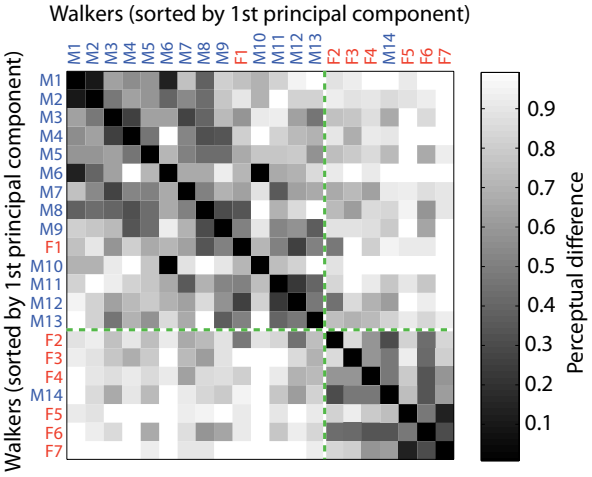

Figure 6.3.: Graphical representation of the data obtained in our perceptual experiment. A large value of the function signifies large difference between two respective motions; the black diagonal means no difference when a motion is compared to itself. The green dashed line shows the threshold between the male and female clusters from Figure 6.2.

the quaternion dot product, but a more complex function based on quaternion logarithm provides both a better function profile and a simple handling of quaternion antipodality (Shoemake, 1985)

$$d_{ang} = min\left(\left\|log\left(\boldsymbol{q}_1^{-1}\boldsymbol{q}_2\right)\right\|, \left\|log\left(\boldsymbol{q}_1^{-1}\left(-\boldsymbol{q}_2\right)\right)\right\|\right)$$

where d_{ang} is the resulting angular difference between the quaternions $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$. Please note that the quaternion logarithm is a 3D vector whose length is linearly dependent on the sinus of the rotation angle. A classical implementation of transformations in homogeneous space performs a translation and then rotation, which means that a joint affects only the *following* joints but not the configuration of the actual joint (resulting in the weights shift of the local metric as compared to the global ones; see Table 6.3).

The *global positional difference* metric is based on the world-space joint position distance (Kovar et al., 2002a; Arikan, 2006):

$$d_{pos} = \|\boldsymbol{p}_1 - \boldsymbol{p}_2\|$$

where $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ describe both joint 3D positions and d_{pos} is the resulting metric value. The on-spot motion conversion makes this metric invariant with respect to global character position and orientation (the altering of the resulting values does not have an impact on the metric performance (Kovar et al., 2002a)).

The third primitive metric, *global velocity difference* compares the magnitude of the velocity difference of a joint in the global coordinate frame (Lee et al., 2002). As in the previous case, the on-spot conversion makes this metric invariant with respect to global character motion, but also alters the results significantly – e.g., a standing foot constraint would have a zero velocity in the original motion, but after the on-spot conversion it will be moving with a negative velocity originally assigned to the root. However, for the comparison of the animation frames, this metric still provides valid results (Kovar et al., 2002a). It is the only primitive metric in our set that describes the dynamic properties explicitly – in discrete frame data, the velocity $\boldsymbol{v}(i)$ in a frame i is computed as a positional difference

between frames i and $i+1$, divided by the time difference between two frames (dt):

$$\begin{aligned} \boldsymbol{v}(i) &= \frac{\boldsymbol{p}(i+1) - \boldsymbol{p}(i)}{dt} \\ &= (\boldsymbol{p}(i+1) - \boldsymbol{p}(i))\, FPS \end{aligned}$$

where FPS is the frames per second value of the animation. The velocity difference $d_{vel}(i)$ for frame i is then expressed as:

$$\begin{aligned} d_{vel}(i) &= \|\boldsymbol{v}_1(i) - \boldsymbol{v}_2(i)\| \\ &= \|(\boldsymbol{p}_1(i+1) - \boldsymbol{p}_1(i)) - (\boldsymbol{p}_2(i+1) - \boldsymbol{p}_2(i))\|\, FPS \end{aligned}$$

All the metrics mentioned above are applicable to a single joint only. For comparing animation frames, we need to merge them to form **composite metric functions**. Lee et al. (2002) suggests to do so by simply excluding unimportant joints (e.g., leaving only elbows, shoulders, hips, knees and spine). A more general description would use weights w_i for each value of the primitive metric d_i, resulting in a composite metric function d:

$$d = \frac{\sum_i w_i d_i}{\sum_i w_i}$$

(note that a linear combination with non-negative weights always preserves all 4 conditions of a metric, as long as all combined functions are metrics).

A human body metric should also be symmetric (e.g., the left and right arm should always have the same weight). To achieve that, we define each component of our composite metric as an average of several symmetrically corresponding joints (see Table 6.2). The properties and the characteristic motion of each group can be also visualised using motion maps (see Figure 6.4).

Index	Group	Included joints
1	Head	Head
2	Clavicles	LeftClavicle, RightClavicle
3	Upper Arms	LeftUpperArm, RightUpperArm
4	Forearms	LeftForearm, RightForearm
5	Hands	LeftHand, RightHand
6	Thighs	LeftThigh, RightThigh
7	Calfs	LeftCalf, RightCalf
8	Feet	LeftFoot, RightFoot

Table 6.2.: The groups of a human body composite metric, based on body symmetry.

Different metrics can also be created by applying what we will call **shaping functions**, i.e., functions that alter the metric function profiles. A generic shaped composite metric d_s is then defined as:

$$d_s = \frac{\sum_i w_i F(d_i)}{\sum_i w_i}$$

where i determines the metric group index, d_i are the bodypart metrics and $F(a)$ is a shaping function applied to a scalar value a. In our framework, we use three shaping functions – squared Euclidean, Manhattan and sigmoid.

The *squared Euclidean* function d_{euclid} is a classical method used in least-squares fitting. Its profile

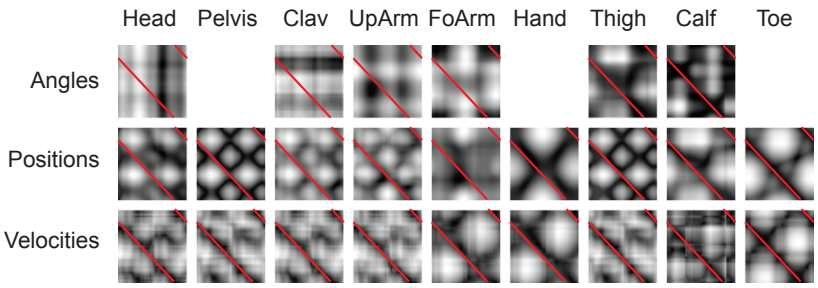

Figure 6.4.: The components of a composite metric based on two periodic animation, visualised using motion maps. The red line represents the "comparison" line – the best timewarping/blending line between two animations.

is a parabolic U-shaped function:

$$d_{euclid} = \frac{\sum_i w_i \left(d_i\right)^2}{\sum_i w_i}$$

The *Manhattan distance* d_{manh} simply sums the values of each bodypart metric:

$$d_{manh} = \frac{\sum_i w_i d_i}{\sum_i w_i}$$

while the *sigmoid function* d_{sigm} provides a logarithmically shaped profile (a logarithm does not satisfy the metric properties) and is often employed in modelling human visual responses:

$$d_{sigm} = \frac{\sum_i w_i \frac{d_i}{d_i+1}}{\sum_i w_i}$$

With all necessary components defined, we can **fit the analytic metrics** to the experiment data. With three shaping functions and three distance metrics, we create 9 composite metrics, each described as a weighed sum of a set of bodypart metrics of one type, altered by a shaping function. The base metric values are provided by a sum of per-frame values on the blending curve, normalised by the total number of frames used (see Figure 6.4). This provides a metric function independent of the animation length, while preserving the properties of the underlying primitive metric.

The fitting process alters the bodypart weights w_i to reflect the experimental results in the least-squares sense, thus minimising the term:

$$e = \sum_i \left(w_i F\left(d_i\right) - d_i^{(exp)} \right)^2$$

with the weight normalisation performed as a post-processing step.

For the final metric to conform to the metric definition (see Section 2.6.1), each of the weights has to be non-negative. For this reason, we use the *non-negative least squares iterative optimisation* technique proposed by Lawson & Hanson (1974) (as implemented in the Matlab environment).

6.1.3. Results

The results of the non-negative least squares fit for all combinations of shaping functions and primitive metrics can be found in Table 6.3. The non-negative nature of the fit makes the fitting function highly non-linear, and consequently causes many of the values to disappear completely (i.e., the best non-negative fit requires them to be set to zero).

		Sq. residual	Norm. const.	Head	Clavicles	Upper Arms	Forearms	Hands	Thighs	Calfs	Feet
Angles	euclid^2	22.62	$1.07\ 10^{-3}$	0	0	1	0	0	0	0	0
	manh	22.499	$3.27\ 10^{-3}$	0	0	1	0	0	0	0	0
	sigmoid	22.39	$1.33\ 10^{-2}$	0	0	0	0	0	1	0	0
Positions	euclid^2	(46.47)	$(8.49\ 10^{-5})$	(0)	(0)	(0.45)	(0.5)	(0.06)	(0)	(0)	(0)
	manh	19.91	$1.11\ 10^{-3}$	0	0	0.1	0.6	0.01	0	0.29	0
	sigmoid	20.29	$1.13\ 10^{-2}$	0	0	0	1	0	0	0	0
Velocities	euclid^2	123.33	$1.29\ 10^{-3}$	0	0	0	0	1	0	0	0
	manh	36.05	$5.42\ 10^{-3}$	0	0	0	0	0.87	0	0.13	0
	sigmoid	20.80	$1.87\ 10^{-2}$	0	0	0	0.66	0.19	0	0.15	0

Table 6.3.: The results of the non-negative least squares metric fitting, with non-convergent results denoted using brackets.

As we can see, overall the sigmoid shaping function matches the results of our experiment best. This corresponds to previous perceptual research (e.g., Onuma et al. (2008), who shown that the profile of a distance metric function is logarithmically shaped), and we can conclude that the logarithmic nature of the human visual system is extended also to our case of human locomotion difference.

Consistently with Troje (2002, 2008), our results also suggest that arm motion is the most important feature of the human body in judging locomotion differences. This fact is supported by all types of primitive metrics (see Table 6.3).

Finally, the best primitive metric for comparing human motion is the global positional difference. This is a logical result, as the global positions directly describe the poses of the figure. Interestingly, the dynamics properties, as described by the velocity-based metrics function, produced the worst match, hinting that the actual static poses are of greater importance when judging the characteristics of the motion than the dynamics (which is consistent with the findings of Wang & Bodenheimer (2003)).

6.1.4. Conclusions

The development of a more general metric capable of emulating human perception of motion differences seems feasible from our results. In this light, our work can be seen as a first step towards developing a well-defined mathematical metric with a desired set of perceptual properties.

6.2. Crowd Motion Variety

A crowd simulation system requires significant computational and storage resources, with a major part dedicated solely to creating the impression of variety. In the case of a data-driven crowd simulator, a certain level of animation and model re-use is inevitable. A high re-usability ratio can provide significant savings in terms of computational effort, memory footprint and artists' time. However, it can also lead to the disturbing impression of a crowd full of identical individuals, or clones, an effect that has been recently explored by McDonnell et al. (2008).

In this section, we build on this work and present a framework that enables a large range of scenarios to be created and configured to test the perception of human motion in groups and crowds. Using this framework, we conduct a perceptual study to find the number of individual motions that are needed to make the movements of a crowd of virtual humans look varied. Our results can be used to derive guidelines regarding the resources required to create, store and process captured motions. Although

the focus of this work is on the dynamic aspects of the animation (e.g., the way a character walks), our framework could also be used to study the static properties (e.g., skeletal structure variations) and the effects of a character's general appearance (e.g., 3D skinned model).

6.2.1. Experiment Framework

In this section we present our framework for evaluating the perception of crowd animation. The first step involves creating the model to be used to represent the appearance of the character. Raw captured motions can seldom be used without some preprocessing, so our second step is to process the animations while making minimal adjustments to the original motion. Finally, different crowd scenarios must be created, which involves the generation of natural and collision-free trajectories for different sized crowds of virtual characters. In all three steps, our aim is to make the framework as configurable and generic as possible.

We wished to create a parametric human model for our framework, so that it could be used to represent both the static and dynamic properties of motion, if required. However, in this section we are interested in determining the number of characteristic motions needed to create the illusion of variety, and hence we also need a way to view the dynamic properties in isolation.

The static properties of the animation are fully represented by the skeleton and its T-pose; we can use this information to construct a model to represent the appearance of the characters.

The simplest models are constructed by displaying the model's joints as dots (i.e., a point-light walker, see Section 2.7.1). The hierarchy of bones is then represented implicitly by the dynamic information of the animation. Even though the human visual system (HVS) can easily derive the hierarchy from a moving model of this type (Johansson, 1973), in practical applications in computer graphics, the skeleton is always represented explicitly. A simple representation is to connect the dots by sticks, leading to a stick-figure, while more advanced models make use of rigid-body segments and skinning techniques, thereby creating a realistic representation of the human body. Unfortunately, these representations are not perceptually equivalent, as it has been shown that motion properties are more easily perceived on realistic human models (Hodgins et al., 1998; McDonnell et al., 2007a). However, with more realistic models, confounding visual cues can be introduced, such as gender, skinning, bodyshape and texture. For that reason, we use a simple mannequin character composed of rigid segments, which has been shown to be gender neutral while still retaining the body structure, and therefore represents motion properties well (McDonnell et al., 2007a). We chose this model in order to focus on the motion properties of the animation, though for future experiments our framework could be easily extended to include a larger variety of appearance models.

While we could use a pre-modelled mannequin character to display every motion, this would require retargeting every animation to fit the skeleton of this model, potentially introducing motion artifacts. Whilst all retargeting methods aim to achieve minimal changes to the original motion, a parametric representation of the mannequin model would allow these changes to be reduced even further. Our animation framework therefore creates the mannequin model by building a simple 'box' model with parameters directly derived from bone lengths (Figure 6.5, middle). By applying several iterations of the Catmull-Clarke subdivision scheme and flattening several critical surfaces (soles of feet, palms), we create a fully parameterised mannequin model (6.5, right).

This parametric model of the character provides the required flexibility in terms of displaying both the static and dynamic properties of any motion in exactly the way they were captured.

However, in the particular case of the experiment described in this section, we are interested in examining the dynamic properties of the motion alone, and therefore wish to eliminate any static cues

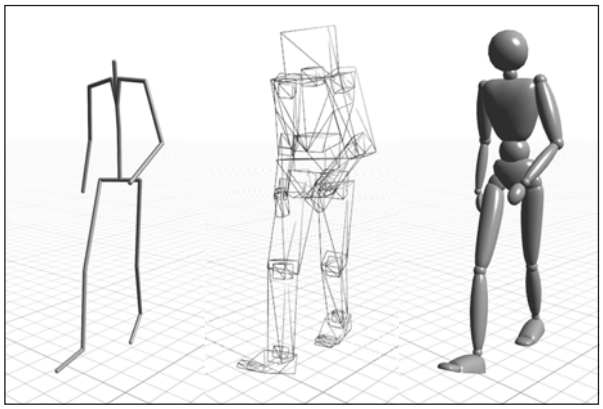

Figure 6.5.: Mannequin character building steps. A stick-figure skeleton (left) is used as a starting point to generate a parametric box-figure (middle), which is then subdivided using Catmull-Clarke subdivision to provide a smooth mannequin character (right).

that could identify an individual's body shape. For this reason, the skeletons have to be identical, so to minimise retargeting errors for a set of animations the skeleton used is the average of the source skeletons. Assuming an identical pose of all source skeletons (which is true for all motion capture systems), their only differences are in the bone lengths (i.e., the translational parts of the rigid body transformations used to describe the skeleton). To create an average skeleton for our experiment, we can therefore simply average the corresponding bone lengths.

In order to use captured animations in a real application, a certain amount of preprocessing is inevitable. In our case, we need to ensure that the character follows a specific trajectory for given length of time. Our aim is to change the animation data as little as possible, while allowing the required level of experimental control.

A simple representation of an animation performed on a flat surface is its trajectory. This is also the usual representation for solving the collision avoidance problem (see below). In the case of locomotion, we derive the trajectory by averaging the projection of the feet bones on the ground (leaving only XZ translation and Y rotation), thereby creating a *trajectory bone* (see Section 5.3.2). Using the motion map approach (see Section 5.2), we can detect one period of the motion, convert it into the periodic form (see Section 5.3.1) and by simply concatenating these periods, create an animation with arbitrary length, while retaining the properties of the original.

We complete the motion preprocessing step by adjusting the animation to fit our average skeleton, according to the adaptation described by Kovar et al. (2002b). As we create a skeleton with minimal bone-length differences for our experiment, we do not require the lengthening step for any of our source animations, though a greater range of body shapes would necessitate this step in future experiments.

Ideally, we would like to create scenarios in which animated characters are placed randomly in a given region and are assigned random trajectories, with a different scenario generated for every trial. However, without any further processing, the generated trajectories would often intersect in the corresponding animation frames, which would lead to collisions between characters (Figure 6.7, left). In order to achieve maximal randomness while avoiding disconcerting character collisions, we introduce a simple iterative least-squares optimisation scheme based on rigid-body fitting. First, we place the animations into random positions and directions in the desired area. With each trajectory represented as a set of 2D vectors $\mathbf{v}_i(t)$ (where i is the animation's index and t is the time value / index of the keyframe), we can define a vector $\mathbf{d}_{a,b}(t)$ describing the difference between two corresponding

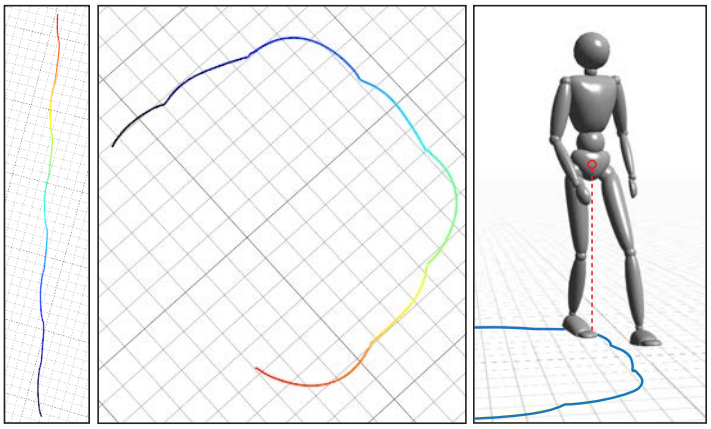

Figure 6.6.: The trajectory reconstruction and trajectory bone embedding. The 2D trajectory (left, middle) is extracted by averaging the projection of the end-effectors on the ground (right) and creating a new parent bone following this trajectory.

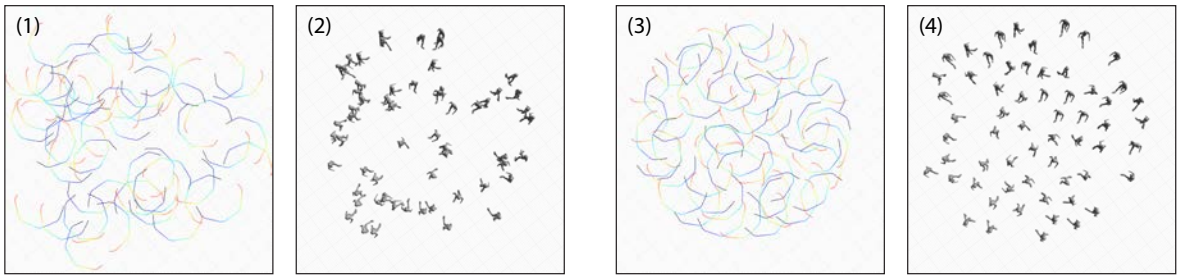

Figure 6.7.: Scenario trajectory optimisation – random trajectories result in a large number of intersections (1), causing many collisions (2). Our optimisation approach ensures a collision-free scenario (3, 4).

points on the trajectories a and b as:

$$\mathbf{d}_{a,b}(t) = \mathbf{v}_b(t) - \mathbf{v}_a(t)$$

We can also create similar displacement vectors for enforcing the trajectories into a certain area. The displacement force vector $\mathbf{f}_{a,b}(t)$ is then created from vector $\mathbf{d}_{a,b}(t)$ as:

$$\mathbf{f}_{a,b}(t) = \frac{\mathbf{d}_{a,b}(t)}{\|\mathbf{d}_{a,b}(t)\|} \left(max\left(0, f_{min} - \|\mathbf{d}_{a,b}(t)\|\right)\right)^2$$

where f_{min} is the minimal enforced distance. This equation describes a simple repulsion force of a particle system (f_{min} was set to $1.5m$ in all our experimental scenarios described below). By adding the force vector:

$$\boldsymbol{f}_a(t) = \sum_b \boldsymbol{f}_{a,b}(t)$$

to each point $\mathbf{v}_a(t)$ of the trajectory a, we create a deformed version of the trajectory data. Subsequently, we can fit the original trajectory data into the deformed trajectory using rigid-body least squares fit, thereby creating a displaced version of the original trajectory. By iteratively applying this process, we minimise the sum of all norms of vectors $\mathbf{f}_{a,b}(t)$, effectively creating a poisson distribution in every frame, leading to a set of collision-free trajectories (Figure 6.7, right).

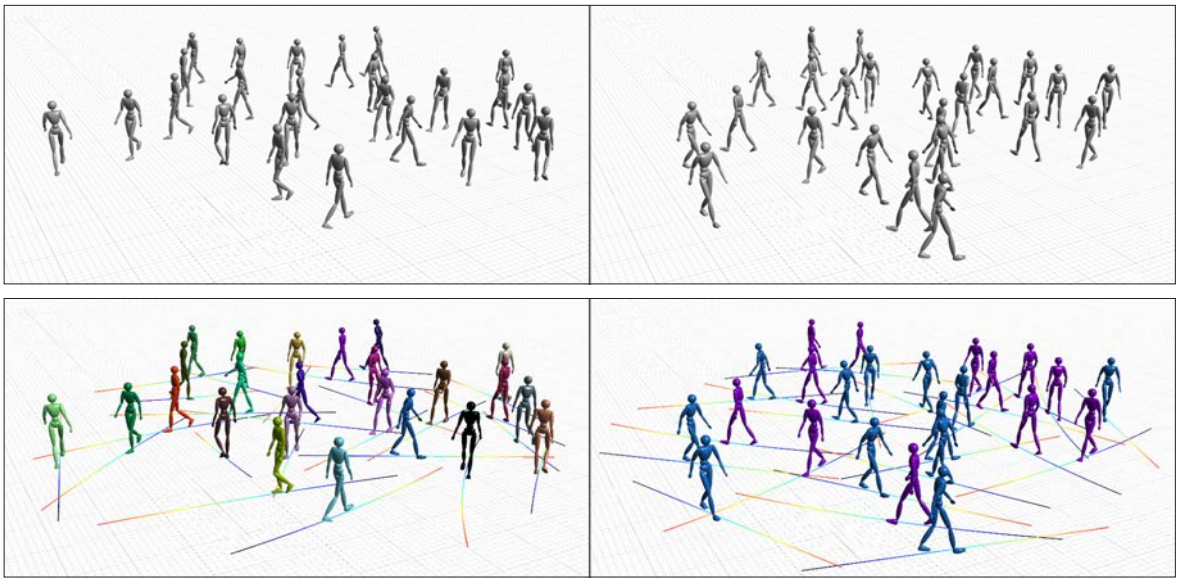

Figure 6.8.: The experiment stimuli and their creation – the stimuli as presented to the subject (top) and their creation, with trajectories and characters coloured according to their respective animations (bottom). In the depicted trial, the gold standard is shown on left.

6.2.2. Experiment Design

We used the methods described in the previous section to determine the minimum number of individual motions required to create the impression of a varied crowd of walking humans. We also explored how this number was affected by the size of the crowd and the speed of motion. Our stimuli consisted of short animations, each depicting a crowd of walking mannequins. We hypothesised that using *a higher number of individual animations will make the crowd look more varied*, up to a certain limit value. Therefore, we tested four values of factor *motion consistency*, where 100% motion consistency meant that all characters in the scene used the same animation, and the lowest value meant that four animations were each used to animate 25% of the crowd. We also tested whether *the size of the crowd will affect the perception of cloned animations*, as more characters on the screen could either mask their motion similarities or else provide more examples to aid in spotting the animation clones. Therefore, the factor *crowd size* had three values – low (8 or 9 characters), medium (15 or 16 characters) and high (24 characters). The +/- 1 is to ensure that the number of characters is divisible by the total number of animations used in the stimulus, in order to enforce an even distribution of the number of simultaneously displayed animations. Finally, we tested whether *the speed of motion will influence the perceived variety of the crowd*, i.e., whether the personal characteristics of the actors have different effects on the perception of motion differences depending on their *speed* (slow, with one period of $1.32 \pm 0.03s$; normal, with one period of $1.06 \pm 0.01s$; or fast, with one period of $0.90 \pm 0.01s$).

As the input data for our experiment stimuli, we used only *straight walks* from our motion capture database, which contains motions from 83 actors (45M and 38F), with ages ranging from 14 to 50, weights from 41 to 102kg and heights from 1.53 to 1.96m. While the framework described in Section 6.2.1 allows the use of turning locomotions, the narrow focus on straight walks allows to both significantly reduce the number of required stimuli and to explore the worst-case scenario by eliminating an important source of locomotion variation.

In order to avoid any motions with artifacts caused by an outlying body build, we selected the 24 required actors for each gender by performing Principal Component Analysis on the weight and height

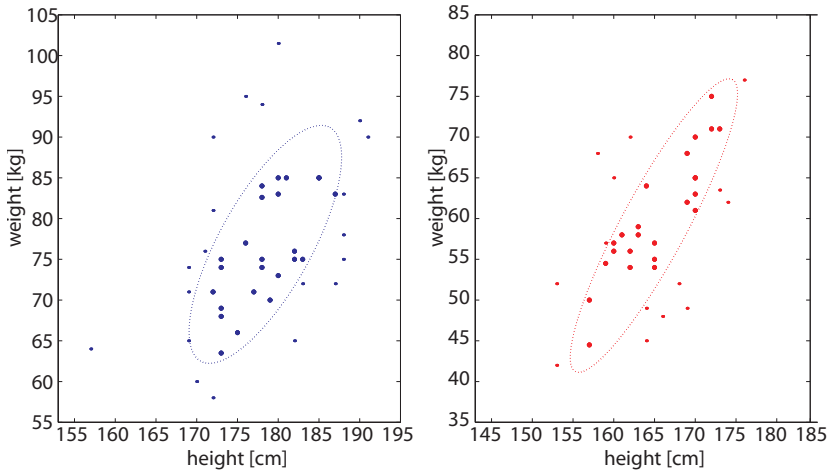

Figure 6.9.: The selection of motion clips based on actors' body shapes (left – males, right – females). The non-standard body builds were rejected (outside the ellipse), while accounting for different height and weight trends (a similar approach is used for computing BMI, and our selected subset would then correspond to participants closest to "average".

data (separately for each gender), constructed an elliptical Gaussian curve around the mean point, with directions determined by the recovered eigenvectors and standard deviations proportional to the eigenvalues, and selected the 24 actors projecting to the highest values on this curve (see Figure 6.9). This procedure provided a subset with sufficient variance, whilst avoiding any weight/height outliers.

For the input motion data, we used 3 straight walking motions from each actor (according to the *speed* factor). The motions were preprocessed by extracting one period, and looping to make the total length of each clip $5 + t_p$ seconds, where t_p was the length of one period of the motion. By selecting the start of each motion randomly from the interval $[0...t_p)$, we achieve full desynchronisation of the displayed motions.

During the experiment, the stimuli were presented on two separate 19-inch LCD screens using their native resolution of 1280x1024 pixels and refresh rate of 60Hz. On one screen, an animation was displayed where every member of the crowd was animated using an individual motion (i.e., the gold standard). On the other screen, the same animation was shown, but with cloned motions based on the *motion consistency* factor (see Figure 6.8). We counterbalanced the *position* of the gold standard across all trials, and randomised the order of trials for each participant. We also tested whether the *gender* of the actors influenced the perception of variety in the motions and therefore showed only all male or all female motions simultaneously in each of the scenarios, also counterbalanced. The scene data were generated before each trial, with the placement of trajectories first randomised and subsequently optimised to avoid any collisions (see above). The camera view was centred in the middle of the scene and oriented at an angle of 20° above the ground plane with the view angle set to 35° vertically to encompass the full scene, which was always 12m in diameter.

The scene was rendered using the OpenGL rendering system with vertical synchronisation turned on to avoid any possible flickering artifacts. The experiment itself consisted of 144 trials: 4 (*motion consistency*) × 3 (*crowd size*) × 3 (*speed*) × 2 (counterbalanced for actor *gender*) × 2 (counterbalanced for gold standard *position*). Each trial lasted 10 seconds and consisted of two repetitions of the 5 second animations. The borders of both screens started blinking 2.5 seconds before the end of each trial (at 0.5 second intervals), to indicate the end of the time limit. Following each trial, there was a 2 second delay before the next display of stimuli, during which the remaining number of trials was displayed in

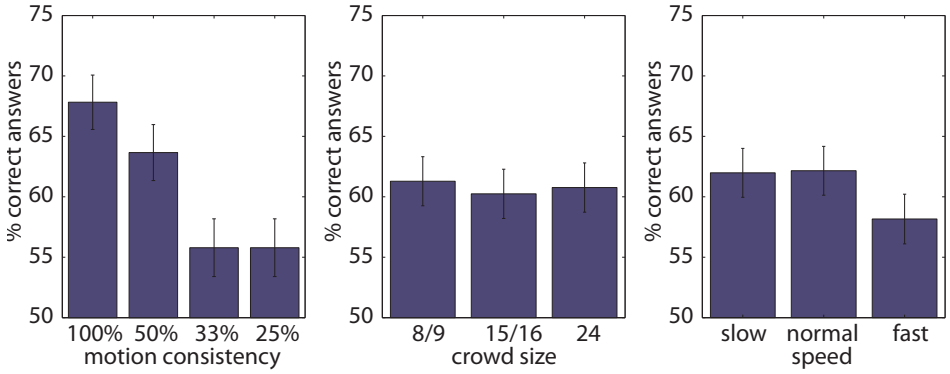

Figure 6.10.: Results from the experiment on the perception of crowd variety.

the centre of each screen.

Twelve volunteers (7M, 5F) from students and staff of our university participated in our experiment. All participants were naive to the purpose of the experiment and had normal or corrected to normal vision. They were asked to indicate *on which screen was every character walking differently* by pressing one of two clearly indicated buttons on the keyboard. We recorded both the accuracy of their answer and their response time. If a participant did not answer in the allocated time limit, the progression of trials was paused until the answer was provided. The total duration of the experiment was about 30 minutes per participant. After the experiment was finished, participants were asked to fill in a simple questionnaire, aimed at determining the discrimination method they adopted.

6.2.3. Results

First, we analysed both the percentage of correct answers and the response times recorded for all participants. As expected, we found no effect of either the gender of the actors or the position of the gold standard, so we averaged across these factors. We then performed a repeated measures ANalysis Of VAriance (ANOVA) on the remaining three factors: *motion consistency*, *crowd size* and *speed* (see Figure 6.10).

For answer accuracy, we found a main effect of *motion consistency* ($F_{3,33} = 7.6517$, $p < 0.0006$) and a three-way interaction between all three factors ($F_{12,132} = 2.1858$, $p < 0.02$). Post-hoc analysis of the main effect using Neuman-Keuls comparison of means showed that there were two distinct groupings of values, with no significant difference within these sets. Motion consistencies of 100% and 50%, i.e., where either 100% or 50% of the characters were animated using the same cloned motion, were significantly easier to detect than the more varied scenarios.

Neuman-Keuls post-hoc analysis of the three way interaction did not provide any more information (no combination of factors was significantly different), and a further study with a larger number of participants would be required to explain this effect.

Trials lasted 10 seconds with a warning sign shown at 7.5 seconds, and the average response time was 7.548s with standard error of 0.054s. The ANOVA on response times showed a main effect of *crowd size* ($F_{2,22} = 4.3230$, $p = 0.03$). Neuman-Keuls post-hoc comparisons showed response times were significantly slower when the largest number of characters was displayed. The difference of means was, however, only 0.27s.

In an informal questionnaire presented to each participant after finishing the experiment, participants reported that the discrimination method they adopted during the experiment was based on searching for pairs of similar motions. If they succeeded in finding such a pair on either of the screens,

they reported the other as the correct answer. Failing that, they resolved to a "general feeling" of the stimulus and selected the one seemingly more varied.

Our results strongly suggest that the dominant factor in the perception of crowd variety is the number of cloned motions present in the scenario. The factors of crowd size and motion speed have, at best, only a weak and unpredictable effect on the perception of motion variety in a crowd. We found that the maximum number of individual motions successfully detectable was 2, i.e., when 50% of the crowd was animated using one motion, and the other 50% also all animated using a second motion.

6.2.4. Conclusions

The main guideline we can provide from this study is that to preserve the perception of motion variety in a typical pedestrian crowd, there is no need for more than 3 different characteristic animations per gender of the displayed characters (whereas 2 motions are not enough). By setting a 10s limit for the answer, our results only indicate the participants' immediate impression of the crowd motion variety. These results are therefore probably most valid for a classical game-like scenario, but may not hold under careful scrutiny of the animations.

6.3. Footskating and Footskate Cleanup

Footskating (or footsliding) is an artifact caused by incorrect or inaccurate handling of foot constraints, such as ground contacts, during character's locomotion. As the primary topic of this thesis is locomotion synthesis, footskating and footskate clean-up is one of the main problems addressed (see Chapter 4). In this section, we evaluate the perceptual saliency of this artifact, together with the performance of some methods for its cleanup, both aspects that currently represent a gap in the available knowledge of animation perception.

In our first experiment (Section 6.3.2), we determined minimal perceivable footsliding thresholds, assuming that the observer is aware of its presence. We showed that participants can perceive even very low levels of footsliding ($<$21mm in most conditions), especially when environment cues highlight the artifacts. In the second experiment (Section 6.3.3), we introduced footsliding at levels that are clearly perceivable, and corrected them using two methods (Kovar et al.'s method (2002b) and lengthening of body segments alone, as in Harrison et al. (2004)). As each of these methods alter the animation in certain ways, they can introduce side effects and artifacts into the original motion. By asking users to compare the corrected motions with the uncorrected ones, we showed that corrected animations are always preferred to animations with footsliding, independently of the correction level required. Results also showed that participants considered animations corrected with the body segment lengthening method to be of higher quality than those edited using Kovar's method. Our results provide valuable insights for developers of games and VR applications by providing thresholds for the perception of footsliding artifacts, as well as a visually effective method for correcting this disturbing artifact.

6.3.1. Stimuli Preparation

Because our goal is to ask participants to judge the quality, naturalness and perceivable artifact levels of motions, we need to ensure that the original motions (before alterations based on the experiment objective) are of high quality, natural and artifact-free. Motion capture technology can provide such motions. For our experiments, we selected a set of motions from our database (83 actors, with a variety 100+ motion types per actor). First, based on the biometric data (using the same method as in Section 6.2), we selected 10 actors (5 male and 5 female) with normal body builds. Second, for each actor we selected a straight walk with a speed close to the actor's average walking speed. Finally, we

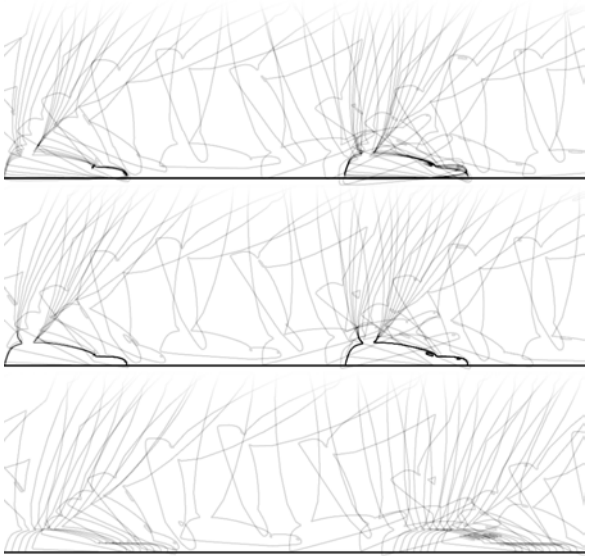

Figure 6.11.: Illustration of the footstep cleanup and parameterised footskate introduction used in our experiments. From top: original motion with imperfect footsteps; a cleaned-up version of the original animation; the same animation with a hight level of forward footslide introduced (slide of 15cm per footstep).

carefully checked that the selected motions did not contain any visible artifacts.

A classical representation of a motion clip relates motion data to the character's skeleton, which represents his/her body structure. While real-world applications would use more realistic skinned human models, which have been shown to convey more information than simpler models (Hodgins et al., 1998), one of the sources of a diversity of animation artifacts is motion retargeting, which adapts the motion to a new skeleton (e.g., a skeleton of a displayable character; Kovar et al., 2002b). In order to avoid these alterations, we chose to display the motions on virtual mannequin figures, generated to accurately reflect the morphology of the original actors (see Section 6.2.1). Such mannequins offer a more suitable representation for our purposes than stick figures, as they convey the character's spatial information, while avoiding the problem of retargeting inherent to skinned characters.

Due to the physical nature of the motion capture technology, a certain number of artifacts will always be present in the captured motion (see Figure 6.11, top), caused by measurement noise and the capturing/processing pipeline. To ensure that we started with perfect animations, we first detected the footstep constraints using the rotational-axis method (see Section 4.3) and then cleared any residual foot motion using the method of Kovar et al. (2002b). As we did not perform any retargeting, the footskate correction on captured motions caused only minimal changes (Figure 6.11, middle). Even though the detection algorithm is fully automatic, its parameters were manually checked and adjusted for each animation to ensure the resulting constraint accuracy.

6.3.2. Baseline Experiment

In the first experiment, our aim was to determine the saliency of footsliding for worst-case scenarios, when participants are aware of its presence or when it is emphasised by the properties of the environment.

In order to allow for precise control of the footsliding, we modelled its effects by altering the translational component of the root trajectory. Our model consists of two distinct components relative to a character's heading direction – the *front footskate* and the *side footskate*.

Front footskate was introduced by multiplying the forward directional movement of the root by a

speed coefficient. This resulted in speeding-up (i.e., gliding) for values above one, or slowing-down (i.e., moonwalking) for values below one (with no change when the value was exactly one). For all the motions used in this experiment, an increase or decrease of 10% of the root velocity corresponded to an average foot position change of $74.2 \pm 1.3mm$ per footstep. The amount of footsliding in mm is linearly related to the speed coefficient.

Side footskate was introduced by adding a perpendicular displacement proportional to a sinus function (with period consistent with that of the animation and phase synchronised with the midpoint of the left foot constraint). The neutral value in this case was zero, while positive values led to outwards footskating (i.e., the constrained foot was moving away from the character's root projection on the ground) and negative values to inwards footskating.

This approach ensured that only smooth changes were introduced into the original motions, while providing a controllable way of introducing footsliding artifacts similar to the ones seen in VR applications (caused by speeding up, slowing down, or blending between straight and curved locomotions).

The experiment was divided into two main blocks. In the first block (*grid environment*), we tested the worst-case scenario where a grid texture on the ground provided visual guides (Figure 6.12, top row). The grid was white on a blue background with interline distances of 10cm. The second block (*neutral environment*) used a plain blue ground without any texture (Figure 6.12, bottom row). Each block was further divided into two parts (Part 1 and Part 2). Part 1 tested frontwise footsliding with motions viewed from the side (Figure 6.12, right column), while Part 2 tested sidewise footsliding with motions viewed from the front (Figure 6.12, left column). This choice of viewport/footsliding combinations ensured that we tested the worst-case scenarios (where the footsliding direction was always perpendicular to the view direction), because we observed that the viewing direction can play an important role in perceived levels of footsliding. Motions were displayed on a 24-inch screen at 60Hz and participants were seated approximately 60cm from the screen.

To accurately determine the perceptual threshold for footsliding for each participant, we used an adaptive double staircase experiment design (Cornsweet, 1962), in a standard Yes-No form (a variant of 2-alternative forced choice) with fixed up and down steps, as described by García-Pérez (2001). We used a down/up step ratio of 0.871, converging at 52.38% point of the psychometric curve, and set the stopping condition to 20 reversals.

In each trial, participants observed one of the ten motions, randomly selected, with the introduced amount of footsliding corresponding to the current state of the staircase experiment. They were asked to determine *if the character is footsliding*, and to answer by pressing the "yes" or "no" button, clearly marked on the computer's keyboard. The description of what footsliding is was provided both formally on the instruction sheet and informally by the experimenter. There was no explicit time limit and the adaptive nature of the method used did not allow the length of the experiment to be precisely determined beforehand, but none of the participants took more than 20 minutes. Because of the complexity of the task, each participant performed at most two parts and received a book voucher in recognition of their efforts.

Nine volunteers took part in the grid environment block of this experiment (2F, 7M), with five participants for grid Part 1 and four participants for grid Part 2. Fourteen volunteered for the neutral environment block (5F, 9M), with eight participants for neutral Part 1 and six participants for neutral Part 2. For this and all subsequent experiments, all participants were naive to the purpose of the experiment and came from various educational backgrounds.

To evaluate the results, we used the Matlab psignifit toolbox (Wichmann & Hill, 2001) to fit a logistic psychometric curve into the data for each block/part, both to each participant and to the

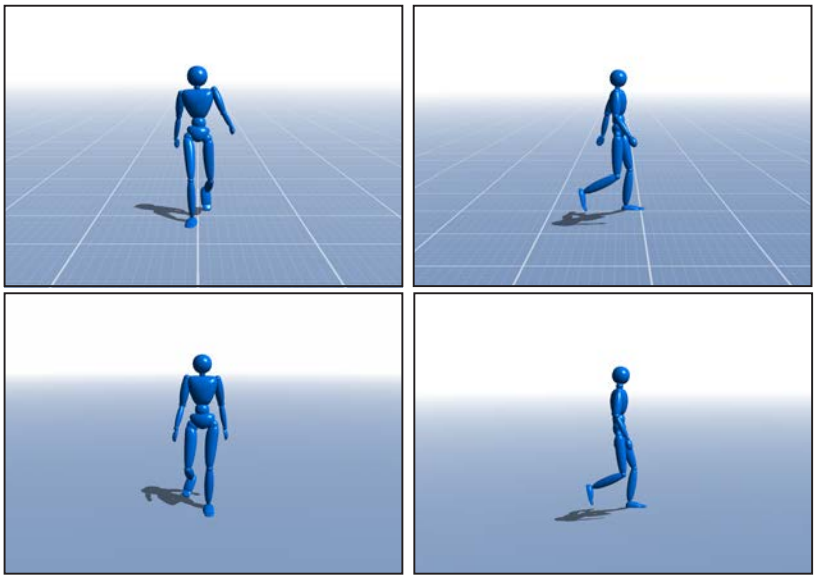

Figure 6.12.: Stimuli examples for the baseline experiment. Top row: worst-case scenario with the environment providing visual guides. Bottom row: second tested environment without visual cues.

Case	Grid environment				Neutral environment			
	Speed coef.		Footsliding amount [m]		Speed coef.		Footsliding amount [m]	
	PSE	JND	PSE	JND	PSE	JND	PSE	JND
gliding	1.028	0.010	0.021	0.007	1.115	0.034	0.084	0.027
moonwalking	0.984	0.007	-0.011	0.005	0.971	0.013	-0.021	0.009
sliding outwards	-	-	0.006	0.002	-	-	0.018	0.005
sliding inwards	-	-	-0.005	0.001	-	-	-0.014	0.005

Table 6.4.: PSE and JND for baseline experiment. Absolute values are given in metres; relative values as a ratio of the motion speed-up/slow-down.

overall merged results. A psychometric function models how the participant's response to stimuli varies depending on the variation of these stimuli. The results of the evaluation are presented in Figure 6.13, which shows both ratings per-participant and the overall psychometric curve. The overall point of subjective equality (PSE, the point where participants are equally likely to find the stimulus acceptable or otherwise) and just noticeable difference (75% JND; the minimum increase of the PSE stimulus value needed to be detectable 75% of the time) are summarised in Table 6.4. As expected, the results show that the accuracy of footsliding detection is greatly improved by the visual cues present in the grid environment. Furthermore, the levels of footsliding perceived in the grid environment (especially for the side case) suggest that the limit is approaching the display resolution of the screen. Our interpretation is that, effectively, *any footsliding, no matter how small, will be perceived on a surface with visual cues.*

However, the results are different for the neutral environment. In the case of front footsliding, the results showed that participants are more disposed to perceive footsliding caused by slowing-down rather than speeding-up, with the speed coefficient PSE of 0.971 and 1.115 respectively. These values corresponded to respective errors of approximately 21mm and 84mm. In the case of side footsliding, a lower threshold with PSE $<$ 20mm was found.

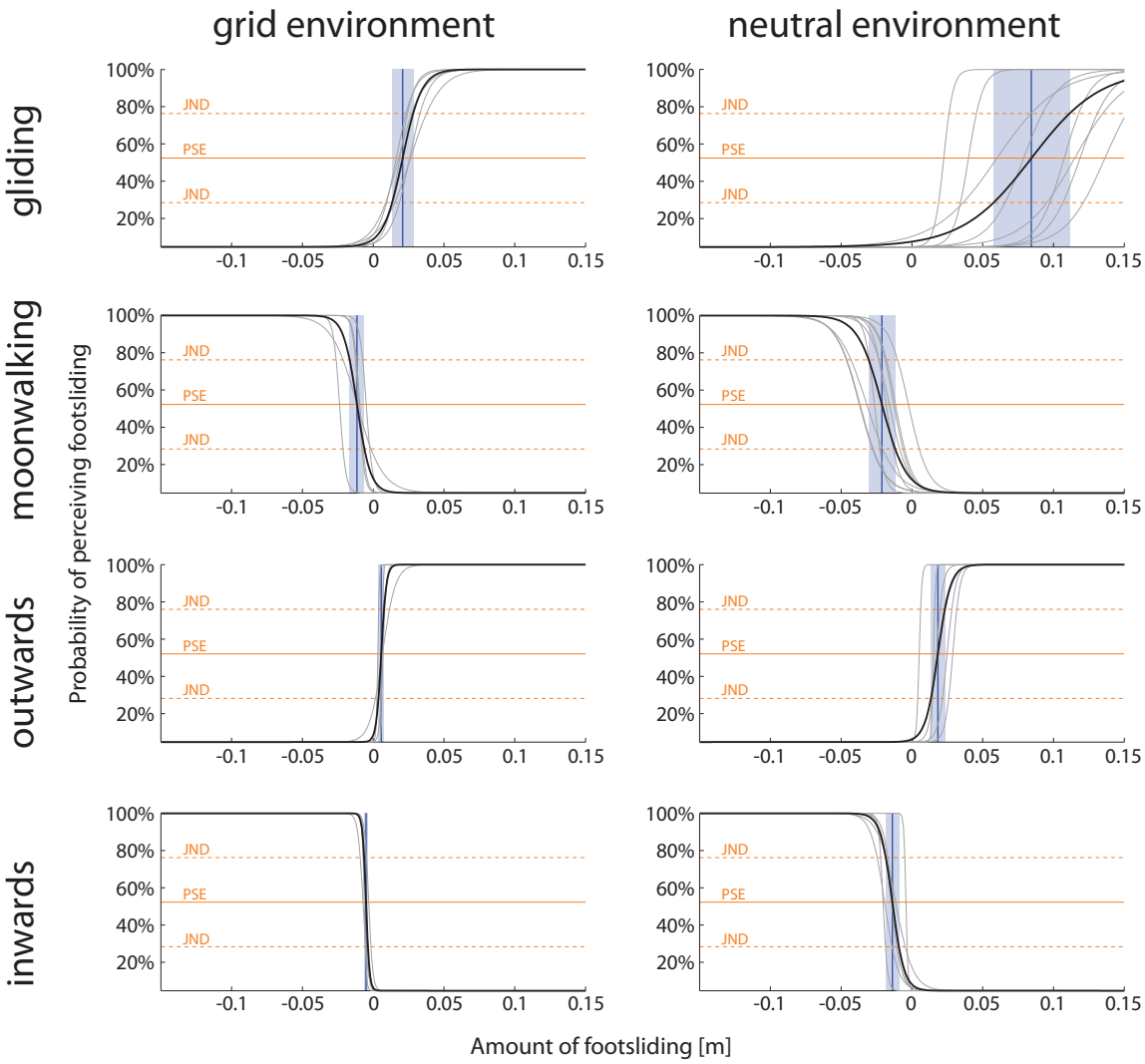

Figure 6.13.: Results of the baseline experiment. The horizontal axes represent the footsliding amount in metres; vertical axes represent the probability of identification of footsliding at a given level; original per-participant curves are depicted in thin gray; overall fitted curve in thick black.

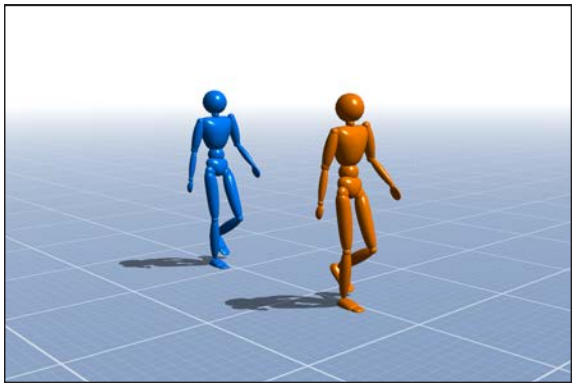

Figure 6.14.: Stimuli example for the footskate cleanup experiment.

6.3.3. Footskate Cleanup Experiment

Our previous experiment showed that the perceptual threshold for footsliding is relatively low, especially when environment cues are present to highlight its effect. However, it is still unclear whether footsliding should be corrected in all cases, or if sometimes a remedial action can actually deteriorate the quality of the motion even further. For this reason, in our second experiment we compared animations with footsliding to animations that had been corrected. We hypothesised that possible artifacts might play an important role in a participant's responses to the corrected motions, and might even lead to them preferring the original animation despite footsliding being present.

Considering the importance and frequency of footsliding artifacts, a surprisingly low number of methods have been proposed to correct them (see Section 2.3.3). For our experiment, we selected two basic methods commonly employed in real-time systems. The first and more comprehensive method was introduced by Kovar et al. (2002b). In this method, footskating is cleaned up using successive steps of a character's root displacement, root trajectory smoothing, leg inverse kinematics and leg segment lengthening. The second and simpler method employs only lengthening of the character's limbs to reach the desired end-effector position (which corresponds to the last step of Kovar's algorithm). While the first method is much more sophisticated, and therefore is expected to perform significantly better, simple limb lengthening might provide acceptable results, as hinted by work of Harrison et al. (2004), who showed that under certain conditions, limb lengthening of up to 19% might go unnoticed by the observer. In the experiment, these two methods will be compared both against each other and with uncorrected animations. In the following text, these three options will be described as *Kovar's correction* (K), *Lengthening correction* (L) and *Uncorrected* (U) respectively.

In VR applications, footsliding in the direction of the character's movement is a necessary price for responsiveness, and therefore is more common than side footsliding. For this reason, in this experiment we focused on the former aspect alone. Our results should generalise to side footskate as well, but this needs to be confirmed in future work.

Twelve participants volunteered for this experiment (3F, 9M). Seventy-two animations were presented in randomised order, with factors: 2 *directions* (gliding and moonwalking) × 3 *footsliding levels* (50%, 75%, 99%) × 3 *comparisons* (U-K, U-L, K-L) × 4 repetitions. The 3 footsliding levels, selected according to the results of the Baseline experiment, were the 50%, 75% and 99% points of the psychometric curve, which each represent an average footsliding distance of 84mm, 111mm and 196mm respectively for gliding and 21mm, 30mm and 60mm for moonwalking.

In each trial, participants viewed two characters displaying the same motion with the same level of

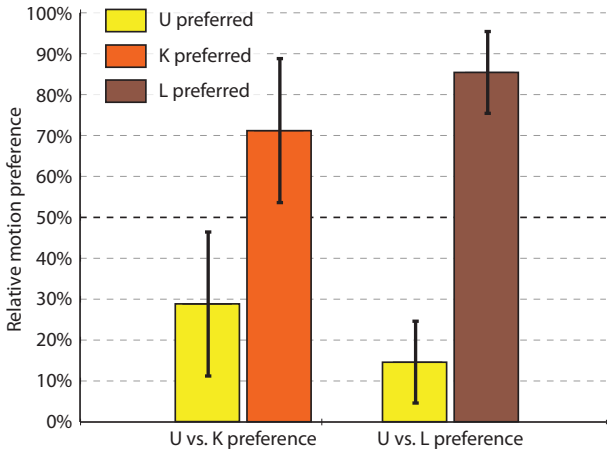

Figure 6.15.: The main effect of comparison factor on trials where corrected motions (K or L) were presented together with uncorrected ones (U).

introduced footsliding, differing only in the correction method (U, K or L). The character displayed on the left was always blue and the right one always orange, matching two coloured keys on the keyboard. In each displayed pair, a random selection was made to decide which of the characters will display which motion. Participants were instructed to select the character *displaying the most natural motion*, without any additional information about the bodypart or artifact they should focus on.

In the previous experiment, footsliding was displayed from the side view, as it gave the most information about the front sliding motion of the feet. In this experiment, however, the artifacts are present both in the feet and in the character's overall movement. Because camera angle can influence the perception of both aspects differently, we used the canonical viewpoint (Ennis et al., 2011), thereby providing as much information as possible about both the overall quality of the motions and the footsliding (Figure 6.14).

The evaluation of the results of this experiment is performed in two steps, as its design inherently contains two distinct sets of data, each of which provide separate results and require separate interpretation. The first data set includes a comparison of corrected motions displayed together with the uncorrected ones (cases U-K and U-L). The second is used to directly compare the two correction methods when displayed simultaneously (K-L).

To correct or not to correct? To evaluate participants' preferences between corrected and uncorrected motions, we extracted a subset of our data that contained the answers from trials when uncorrected motions (U) were displayed simultaneously with motions corrected using either of the correction methods (K or L).

We performed a three-way repeated measures ANOVA with within-subjects factors *comparison* (U-K and U-L), *direction* (moonwalking and gliding) and *footsliding level* (50%, 75% and 99%). In this evaluation (and in all subsequent evaluations), the post-hoc analysis was performed using the standard Newman-Keuls test for comparison of means. The results showed a main effect of comparison ($F_{1,11} = 7.376$, $p < 0.05$), where footsliding animations corrected with the lengthening method were preferred to uncorrected motions more often than animations corrected with Kovar's method (Figure 6.15). The results also showed a main effect of footsliding level ($F_{2,22} = 3.737$, $p < 0.05$), where corrected animations were more likely to be preferred with higher levels of the footsliding (Figure 6.16).

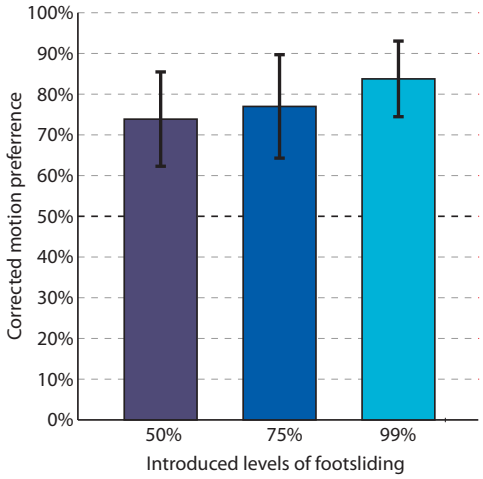

Figure 6.16.: The main effect of footsliding level factor on trials where corrected motions (K or L) were presented together with uncorrected ones (U).

The results also show that, overall, the corrected motions were considered to be of better quality in more than 70% of the trials, independent of the level of introduced footsliding. To explore this fact further, we carried out single t-tests on collapsed data to determine if there was a real preference of correction methods over non-correction (i.e., preference significantly different from 50%). The data was collapsed over direction and level factors, as we were interested in the overall preference of each correction method against non-correction. The results showed that both correction methods were preferred to non-corrections in significantly more than 50% of the cases ($t(11) = 2.946$, $p < 0.05$ for Kovar's corrections and $t(11) = 8.666$, $p < 0.00001$ for lengthening ones). From this result we can conclude, that *correcting footsliding should always be preferred over not correcting it, even if it implies changing the motion.*

Preferred correction method? The analysis above shows that corrected animations are preferred to uncorrected ones, with the lengthening method prefered on average more often than Kovar's method. The second subset of our data allows for a more direct comparison between the two correction methods when displayed side-by-side.

We performed a two-way repeated measures ANOVA with within-subjects factors *direction* and *footsliding level* on the preference of L over K. Results showed no significant effect of direction or footsliding level. We then averaged participant ratings over these two factors and carried out a single t-test to determine if one of the correction methods was significantly preferred to the other (i.e., preference significantly different from 50%). The results showed that *the lengthening of body segments method was ranked higher than Kovar's approach 72% of the time* ($t(11) = 5.693$, $p < 0.0005$, Figure 6.17).

6.3.4. Conclusions

In this section we have presented a set of experiments addressing the perceptual effects of footsliding in VR applications. Our results have shown that participants can perceive even very small levels of this artifact, with a trend towards accepting higher levels of footsliding in the direction of the character's movement. Moreover, we have shown that the ability to perceive footsliding is increased even further by visual cues in the environment, i.e., a grid texture on the ground, up to a point close to the screen resolution.

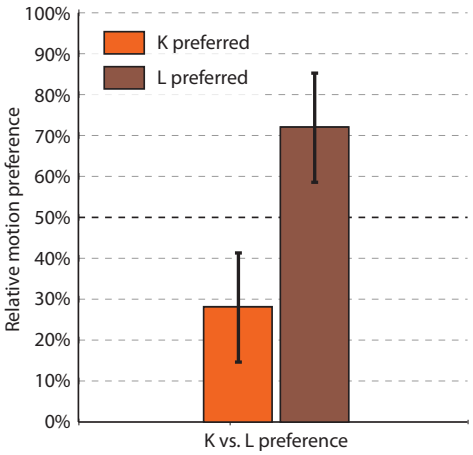

Figure 6.17.: Lengthening correction method is preferred over Kovar's when displayed simultaneously.

Out of the two tested correction methods, users preferred the simpler lengthening method. This could imply that the perceptual threshold for limb lengthening is relatively high (a finding consistent with the work of Harrison et al. (2004)), and that the artifacts introduced by Kovar's method are perceptually more salient. However, attention would also affect the user's preference, as Harrison et al. (2004) demonstrated that limb length changes of over 20% can go unnoticed only when the user's attention is not focused on the extending limb.

In our experiments, the stimulus character was a neutral mannequin figure. While a well-justified choice for our scenario, future work has to evaluate how these results would scale on more realistic characters, as they could prove to result in even less tolerance towards both footsliding and correction artifacts. Moreover, a general perceptual evaluation of limb lengthening on highly realistic humanoid characters has yet to be performed.

The viewpoint and camera angle can also play a significant role in footsliding perception. In our experiments, we have chosen two approaches – in the first experiment, we chose the worst-case scenario, by placing the camera in direction perpendicular to the introduced footsliding. While a valid choice for our goals, a different angle might prove to be more forgiving. In the second experiment, we have used the canonical viewpoint. This choice was motivated by the fact that in this experiment it is not only footsliding that we are evaluating, but also the overall motion artifacts introduced by correcting it. A different camera angle might mask either of these artifacts, shifting the user's preference towards one or another.

To conclude, future research on the perception of motion artifacts (such as footsliding) should explore features such as high quality human characters, influence of environment, camera angle, camera motion, and the effects of the user's attention.

6.4. Human Locomotion Timewarping

Understanding the perception of timing in humanoid character motion can provide insights useful for computational and storage cost optimisation, motion editing and motion extrapolation for data-driven motion synthesis methods (such as our locomotion system). In this section, we provide a perceptual evaluation of linear timewarping for human locomotion.

In our experiment, participants were shown pairs of walking motion clips, both timewarped and at their original speed, and were asked to identify the real animation. We found a statistically significant

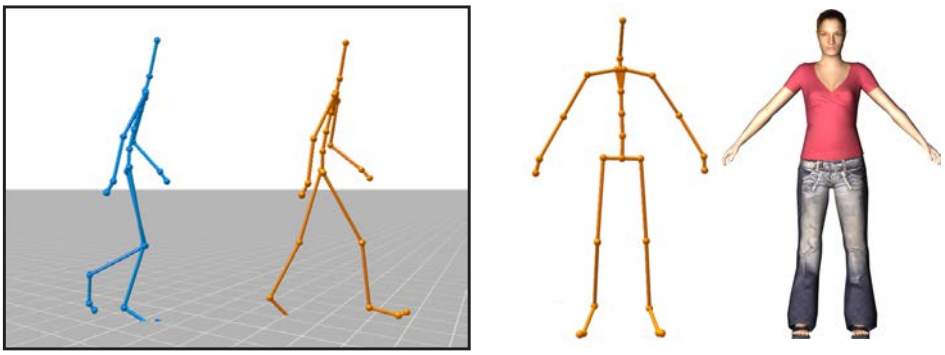

Figure 6.18.: The stimuli of the timewarping experiment. Left – screenshot from the running experiment (side view, stick figure); right – stick figure and geometrical model used in the study.

difference between speeding up and slowing down, which shows that displaying clips at higher speeds produces obvious artifacts, whereas even significant speed reductions were perceptually acceptable.

6.4.1. Experiment Design

Five motion captured clips of a walking animation served as the stimuli for our experiment. These five animation speeds covered a normal range of human walking, ranging from 0.8 m/s to 2.4 m/s with 0.4 m/s increments. For each clip, we created four other versions using timewarping to match the speed of the other clips, leading to a total of 25 clips (e.g., the 1.2 m/s motion was slowed down to 0.8 m/s and speeded up to 1.6, 2.0 and 2.4 m/s). We hypothesised that timewarping would be less noticeable if the timewarped speed is close to the original speed of the clip.

The experiment consisted of sequences depicting two animated characters side-by-side (Figure 6.18, left). Both characters (Figure 6.18, right) were either stick figures or geometric models (as a model's level of detail was found to affect perceptual sensitivity to errors in motion (Hodgins et al., 1998)), facing forwards or sideways (to test the effect of the viewpoint), with each simultaneously displayed pair using the same setup. One character's animation depicted the original motion, randomly placed on the left or right side of the screen, while the other's was timewarped to match its speed. We also tested each real animation against itself as a control case.

Sixty naive participants from the general public (64% male and 36% female) volunteered for this experiment. The instruction sheet indicated that one of the motions was a real captured motion and the other one was synthetically edited. The task was to indicate which of the two animations was the real motion by clicking the left or right mouse button. Each participant completed 100 trials in randomised order (25 motion clip combinations, 2 models, 2 viewpoints), leading to an average of 8 minutes per participant.

6.4.2. Results

Using a 3-factor repeated measures ANalysis Of VAriance (ANOVA) with factors *model type* (stick or geometric), *viewpoint* (front or side), and *speed* (speed-up or slow-down), we found a main effect of model type ($F_{1,59} = 8.09$, $p < 0.007$). Post-hoc analysis using Newman-Keuls comparison of means showed that participants were more sensitive to timewarping on the geometric models than on the stick-figure. This implies that *the effect of timewarping is more noticeable on more detailed characters*, which is consistent with previous results (Hodgins et al., 1998). No main effect of viewpoint was found, implying that *side and front views did not affect sensitivity to timewarping artifacts.*

A main effect of speed was found ($F_{1,59} = 133$, $p < 0.000001$), where participants were able to notice

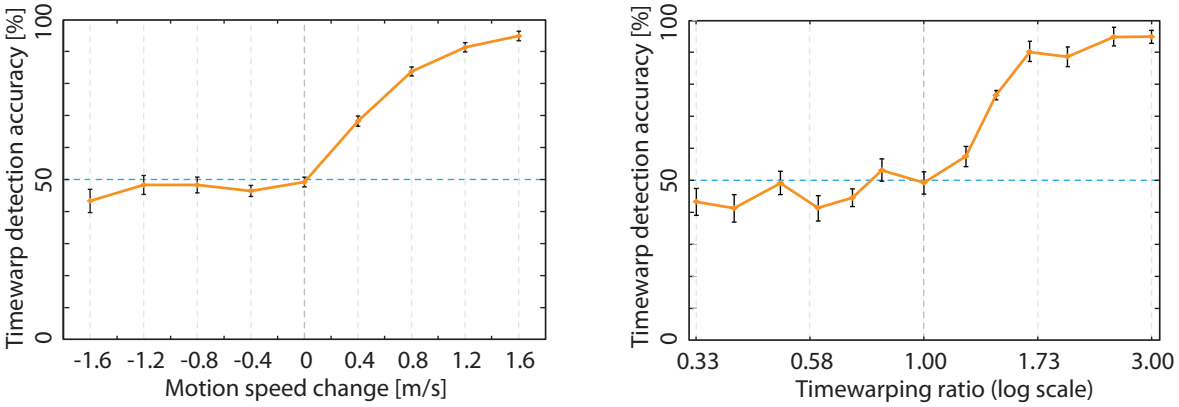

Figure 6.19.: Results of the timewarping experiment, showing that motion speeding up produces severe perceptual artifacts while even significant slow down is perceptually acceptable. The graphs show timewarping results in terms of absolute speed differenced (left) and relative timewarping ratio (right).

speeded up motions on average 80% of the time vs. 47% of the time for slowed down animations. These results were consistent across conditions. Since we used a 2AFC (2 alternative forced choice) paradigm, this suggests that participants were simply guessing when they viewed a slowed down animation beside a real one. Therefore, consistent with Reitsma & Pollard (2003), *the effect of timewarping is much more noticeable when speeding up motions than when slowing them down.*

In order to investigate the speed factor further, we averaged the accuracy of participants' ratings for each timewarp level (-1.6 to +1.6 m/s with 0.4 m/s steps) and each timewarp ratio (0.33 to 3.0). A single factor repeated measures ANOVA showed a main effect of timewarp level ($F_{8,952} = 104.4$, $p < 0.00001$). Figure 6.19 shows that, as expected, performance was at chance (i.e., 50% accuracy) for the real animation compared with itself. Furthermore, for all slowed down animations, performance was also at chance, showing that participants did not prefer the real animations more than even significantly slowed down ones. However, when the animations were speeded up, participants were much more sensitive to the different levels of timewarping – up to almost 100% for a 0.8 m/s animation speeded up to 2.4 m/s. This implies that *significantly slowed down walking animations are perceived as real, whereas even small increments in speed will be noticed.*

6.4.3. Conclusions

The results of this experiment have implications for both motion compression and data-driven parameterisation models. They suggest that, in motion compression, the underlying databases do not have to contain densely sampled low speed locomotions, as they can be reconstructed from higher speeds without producing perceptual artifacts. Similarly, the indications are that motion synthesis methods should avoid timewarping a source motion to create a faster movement. Rather, interpolation between slowed down locomotion clips would be preferable to the traditional combination of speeded up and slowed down animations. However, our conclusions are directly applicable to walking locomotion only, with speeds ranging from 0.8 m/s to 2.4 m/s, and further studies are now needed to confirm and generalise our results.

6.5. Conclusions

Our experiments provide interesting insights into problems related to the main topic of this thesis, the human locomotion synthesis. However, they are not tightly bound to out particular set of techniques, which makes them generalisable to other data-driven animation systems.

Our locomotion comparison metric directly reflects the perceptual properties of a locomotion clip, which makes it suitable for transition detection. In the context of this thesis, its most significant use is for motion map building (see Section 5.2). The crowd variety experiment shows that the total number of different characteristic motions required for a varied appearance of a crowd to is relatively low. Our result of three motion styles per gender allows significant savings in the memory requirements of a crowd animation system. Our set of footskating experiments addresses the most important artifact of human locomotion synthesis. We confirmed that it is a very salient artifact, but also recommended a simple method of its correction, which has better perceptual properties than state of the art methods. Finally, in our timewarping experiment, we found a significant asymmetry in the perception of naturalness connected with animation time manipulation. This provides a guideline for both animation editing and parameter extrapolation – the slowing down of an animation has almost no perceptual impact, whilst the speeding up will lead to artifacts.

7 Conclusions

In this chapter, we reflect upon the contributions of this thesis and discuss potential directions for future work, primarily aimed at addressing some of the limitations of the introduced techniques and experiments.

7.1. Contributions

In Section 1.4, we introduced the contributions of this thesis, which we now discuss further.

7.1.1. Technical Contributions

The primary technical contribution of this thesis is the **parametric locomotion model**. We have introduced several methods and concepts for building a parametric structure from several clips of human locomotion, such as motion maps, the trajectory bone and skinning matrix blending, which allow both building the parametric space and synthesising the motion to be fully automatic. Moreover, we have evaluated the computational complexity of this model, showing that for the purpose of animating middle level of detail characters, it is significantly more efficient than previous work.

To provide data for the locomotion model, we have also demonstrated an improved **animation pipeline**, including the motion capture system setup, an improved human body model used for kinematic data reconstruction, and a set of preprocessing tools that allow an animation to be converted into periodic form, and detect and refine/reconstruct the footstep constraint information.

All these together form a comprehensive motion capture based animation suite, providing a complete and mostly automatic solution to locomotion generation for crowds.

7.1.2. Experimental Results

The results of our perceptual experiments provide interesting insights into how human motion is perceived, which can be used to reduce and target the computational resources required for animation generation.

Our **locomotion metric** experiment compared the performance of different analytical metrics to

perceptual data. We showed that the global positional metrics reflect the characteristic locomotion properties the best and that the motion of certain bodyparts, e.g., hands and feet, is more salient than others. Based on these results, a new combined metric was created, which reflects closely the recorded data. Finally, the mthods can be used to determine the perceptual properties of different types of motion.

In a similar fashion, we have tested the level of **motion variety** required for a group of characters displayed simultaneously on the screen to appear varied. Our results show that there is no need for motion data from more than three actors, independently of other aspects of the motion, such as motion speed. This could significantly reduce the amount of manual work and data storage required to create a realistic crowd scenario.

Our **motion timewarping** experiment provided interesting insights into time manipulation of an animation clip. We have found that the perceptual impact of timewarping differs significantly between the slowing down and speeding up of motion, with the negative effects of speeding up being significantly more salient. This is an important finding, as it can impact on both existing techniques (i.e., speeding up should be avoided if possible) and provide guidelines for new systems development (which should favour motion slowing down).

Finally, the last set of experiments was aimed at **footskating** during locomotion and related **footskate cleanup**. Our perceptual experiments confirmed that footsliding is a very salient artifact which, as long as the character is close enough to the camera, should always be corrected. However, the state-of-the-art techniques can be improved upon by adopting a much simpler limb lengthening approach.

7.1.3. Implementation Outputs

During the course of this research, many implementation solutions were created, underpinning both the technical and experimental aspects of the thesis.

A primary output is the C++ motion editing library, which serves as a base for all technical demos, practical applications and perceptual frameworks. It consists of a large collection of algorithms and data structures, all developed with reusability in mind. Based on this library, a set of motion editing tools was developed, allowing semi-automatic editing of the motion data. These allow the developer to simplify and automatise the motion processing pipeline, enabling both application interoperability due to the large number of supported file formats, and a significant reduction in manual data processing. Moreover, each of the perceptual experiments comes with its own real-time framework based on the library, which is generally significantly more flexible than the limited case used for each experiment.

Apart from the technical demos, two practical applications were developed as final outputs of the development stage. The Metropolis animation system is an implementation of the locomotion parametric model described in this thesis, while the Biodancer project, presented as a part of the Biorhythm exhibition in the Science Gallery, shows primarily the state-based animation synthesis, motion maps concept and constraints detection methods.

Finally, the main implementation output of the Natural Movers project is a consistent motion capture database, containing 100+ motions captured from 83 actors. This provides a large variety of characteristic motions usable for both motion analysis and parametric motion synthesis.

7.2. Limitations and Future Work

The future work on the methods presented in this thesis is focused primarily on addressing related drawbacks and providing a generalisation of the concepts.

7.2.1. Parametric Locomotion Synthesis

While the parametric locomotion synthesis model is flexible and more efficient than previous techniques, it has several important limitations, which also provide possible directions for future work.

First, we address only locomotion on a flat surface. While sufficient for the Metropolis project, a parametric representation of slope and/or stairs would be necessary for a more advanced cityscape. Following the concepts used in our solution, these issues can be addressed either by extending the parametric space by using another dimension to describe slope, or by creating an approximate fast non-hierarchical inverse kinematic scheme built on the same assumptions as the non-hierarchical mesh deformation model.

Second, the system as presented does not include transitions between different levels of detail, which would be necessary for a practical application. This limitation is caused by the fact that the system as it stands was efficient enough to drive all characters, while at the same time accurate enough not to produce any salient artifacts. The solution to this limitation can be implemented on the behaviour – animation interface, by including the level of detail parameter and allowing an LOD transition period, with two animation systems temporarily running for both characters.

Third, a state-based animation transition would be necessary to include motions that are not parameterisable by the speed and turning angle, such as stopping and starting the walk, or scene object interactions. This can be addressed using some of the hybrid techniques from the previous work, or by exploiting the fact that all motions inside the parametric space are synchronised, which will significantly simplify the state transition detection.

7.2.2. Experiments

Our experiments specifically address several issues related to the topic of this thesis, but the general area of human animation perception offers large potential for future research. In this section, we address only directly related limitations of our approach and possible future work.

The main limitation of our experiments, with the exception of the motion timewarping perception, is the fact that our characters are represented as mannequin figures. While a valid choice (supported by previous work), which allows us to separate the appearance of the model from its motion without introducing unrelated body shape cues, perceptual performance in comparison with other models was never rigorously examined.

The **locomotion metric** experiment was aimed at evaluating human locomotion and only walks with actor's normal walking speed were used. A set of further experiments might be required to evaluate if the results generalise over different walking speeds, locomotions and to different motion types. Moreover, we have chosen to evaluate the three most common metric functions and three simple shaping functions, with a single comparison value for the whole clip, and clip alignment performed using linear timewarping. A different choice for any of these aspects might provide a more accurate match of an analytic metric with the perceptual results.

The **motion variety** experiment focused on determining the minimum requirement for animations in a crowd scene for the overall motion to appear varied. However, we have examined only the worst-case scenario, where all displayed characters were moving with approximately the same speed, using a straight walk and had the same gender. Moreover, the animations were not manipulated in any way, providing us with very clean results, but excluding interesting ways of providing motion variety. All these aspects provide a large numver of opportunities for further investigation.

The **motion timewarping** experiment determined the difference between speeding-up and slowing-down a locomotion. However, we have not found any perceptual threshold for the unrealistic manip-

ulation of motion slow-down. This can be explained by a possible response bias, i.e., in that the speeding-up was so salient, that the slowing down seemed relatively correct in comparison, independently of the manipulation magnitude. A future experiment should explore these two cases separately, mapping the overall saliency curve precisely. Moreover, we have merged the results of timewarping different original motion speeds, thus providing a generalised results. It might be the case that different original speeds allow for different levels of timewarping. Finally, the results need to be tested on different types of motion. This was already partially explored in a different context in previous work (e.g., ballistic motion (Reitsma & Pollard, 2003)), but a comprehensive study would provide very important insights into the perception of motion dynamics.

Finally, the **footskating** set of experiments established the importance of footstep cleanup and evaluated several popular methods. However, our experiment used a particular setup which, while providing the worst-case scenario, might prove to be too strict for real world applications. Moreover, we have used two methods of footstep cleanup, which do not represent a complete set of methods from previous literature. A different method selection might prove to provide a perceptually more acceptable solution, or some of the methods used might be insufficient under other conditions than those tested.

7.3. Final Remarks

In this thesis, we have tackled the problem of data-driven locomotion synthesis for crowds. Focusing on the middle level of detail, our work successfully addresses all stages of a data-driven motion synthesis model - starting from motion capture equipment and its calibration, through data processing, system implementation, and finishing with a perceptual evaluation of locomotion-related motion properties. Moreover, each step of this work is usable independently from the other parts, with a particular focus on generalisability of the experimental results. Finally, the practical implementation of the concepts described in related projects (Metropolis, Natural Movers, Biodancer, perceptual frameworks) demonstrates the applicability of our results.

Bibliography

(2009). Vicon MX System Documentation.

Alexa, M. (2002). Linear combination of transformations. *ACM Transactions on Graphics*, 21(3), 380–387.

Amaya, K., Bruderlin, A., & Calvert, T. (1996). Emotion from Motion. In *Proceedings of the conference on Graphics Interface* (pp. 222–229).

Arikan, O. (2006). Compression of motion capture databases. *ACM Transactions on Graphics*, 25(3), 890–897.

Arikan, O. & Forsyth, D. A. (2002). Interactive motion generation from examples. *ACM Transactions on Graphics*, 21(3), 483–490.

Arikan, O., Forsyth, D. A., & O'Brien, J. F. (2003). Motion synthesis from annotations. *ACM Transactions on Graphics*, 22(3), 402–408.

Barbic, J., Safonova, A., Pan, J.-Y., & Faloutsos, C. (2004). Segmenting motion capture data into distinct behaviors. *Proceedings of Graphics Interface*, (pp. 185–194).

Barclay, C. D., Cutting, J. E., & Kozlowski, L. T. (1978). Temporal and spatial factors in gait perception that influence gender recognition. *Perception And Psychophysics*, 23(2), 145–152.

Barnes, M. (2006). COLLADA. In *ACM SIGGRAPH 2006 Courses*.

Bindiganavale, R. & Badler, N. I. (1998). Motion abstraction and mapping with spatial constraints. In *Proceedings of the International Workshop on Modelling and Motion Capture Techniques for Virtual Environments* (pp. 70–82).

Bodenheimer, B., Rose, C., Rosenthal, S., & Pella, J. (1997). The process of motion capture: Dealing with the data. In *Proceedings of the Eurographics workshop on Computer animation and simulation* (pp. 3–18).

Boulic, R., Callennec, B. L., Herren, M., & Bay, H. (2003). Experimenting prioritized IK for motion editing. In *Eurographics 2003, Slides & Videos*.

Boulic, R., Thalmann, D., & Magnenat-Thalmann, N. (1990). A global human walking model with real time kinematic personification. *The Visual Computer*, 6(6), 344–358.

Boulic, R., Ulicny, B., & Thalmann, D. (2004). Versatile walk engine. *Journal of Game Development*, 1(1), 29–52.

Brand, M. & Hertzmann, A. (2000). Style machines. In *Proceedings of the conference on Computer graphics and interactive techniques (SIGGRAPH '00)* (pp. 183–192).

Bruderlin, A. & Williams, L. (1995). Motion signal processing. In *Proceedings of the conference on Computer graphics and interactive techniques (SIGGRAPH '95)* (pp. 97–104).

Chiu, C.-Y., Chao, S.-P., Wu, M.-Y., Yang, S.-N., & Lin, H.-C. (2004). Content-based retrieval for human motion data. *Journal of Visual Communication and Image Representation*, 15(3), 446–466.

Choi, K., Park, S.-H., & Ko, H.-S. (1999). Processing motion capture data to achieve positional accuracy. *Graphical Models and Image Processing*, 61(5), 260–273.

Choi, M. G., Lee, J., & Shin, S. Y. (2003). Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Transactions on Graphics*, 22(2), 182–203.

Cornsweet, T. N. (1962). The staircase-method in psychophysics. *The American Journal of Psychology*, 75(3), 485–491.

Cutting, J. E. (1978). Generation of synthetic male and female walkers through manipulation of a biomechanical invariant. *Perception*, 7(4), 393–405.

Cutting, J. E. & Kozlowski, L. T. (1977). Recognizing friends by their walk: Gait perception without familiarity cues. *Bulletin of the Psychonomic Society*, 9(5), 353–356.

Dittrich, W. H. (1993). Action categories and the perception of biological motion. *Perception*, 22(1), 15–22.

Dittrich, W. H., Troscianko, T., Lea, S. E., & Morgan, D. (1996). Perception of emotion from dynamic point-light displays represented in dance. *Perception*, 25(6), 727–738.

Dobbyn, S., Hamill, J., O'Conor, K., & O'Sullivan, C. (2005). Geopostors: a real-time geometry/impostor crowd rendering system. In *Proceedings of the symposium on Interactive 3D graphics and games* (pp. 95–102).

English, E. & Bridson, R. (2008). Animating developable surfaces using nonconforming elements. *ACM Transactions on Graphics*, 27(3), 66:1–66:5.

Ennis, C., Peters, C., & O'Sullivan, C. (2011). Perceptual Effects of Scene Context And Viewpoint for Virtual Pedestrian Crowds. *Transactions on Applied Perception*, 8(2), Article 10.

Faloutsos, C., Hodgins, J. K., & Pollard, N. S. (2007). Database Techniques with Motion Capture. In *ACM SIGGRAPH 2007 Courses*.

Forbes, K. & Fiume, E. (2005). An efficient search algorithm for motion data using weighted PCA. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 67–76).

Gain, J. & Bechmann, D. (2008). A survey of spatial deformation from a user-centered perspective. *ACM Transactions on Graphics*, 27(4), 107:1–107:21.

Galata, A., Johnson, N., & Hogg, D. (2001). Learning variable length Markov models of behaviour. *International Journal of Computer Vision and Image Understanding*, 81(3), 398–413.

García-Pérez, M. A. (2001). Yes-no staircases with fixed step sizes: psychometric properties and optimal setup. *Optometry and Vision Science*, 78(1), 56–64.

Geijtenbeek, T., Pronost, N., Egges, A., & Overmars, M. H. (2011). Interactive Character Animation using Simulated Physics. In *Eurographics State of the Art Reports* (pp. 127–149).

Glardon, P., Boulic, R., & Thalmann, D. (2006). Robust on-line adaptive footplant detection and enforcement for locomotion. *The Visual Computer: International Journal of Computer Graphics*, 22(3), 194–209.

Gleicher, M. (1997). Motion editing with spacetime constraints. In *Proceedings of the symposium on Interactive 3D graphics* (pp. 139–148).

Gleicher, M. (1998). Retargeting motion to new characters. In *Proceedings of the conference on Computer graphics and interactive techniques (SIGGRAPH '98)* (pp. 33–42).

Gleicher, M. (2001). Comparing constraint-based motion editing methods. *Graphical Models*, 63(2), 107–134.

Gleicher, M., Shin, H. J., Kovar, L., & Jepsen, A. (2003). Snap-together motion: assembling run-time animations. *ACM Transactions on Graphics*, 22(3), 181–188.

Grassia, F. S. (1998). Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3(3), 29–48.

Hamill, J., McDonnell, R., Dobbyn, S., & O'Sullivan, C. (2005). Perceptual evaluation of impostor representations for virtual humans and buildings. *Computer Graphics Forum*, 24(3), 623–633.

Harrison, J., Rensink, R. A., & Panne, M. V. D. (2004). Obscuring length changes during animated motion. *ACM Transactions on Graphics*, 23(3), 569–573.

Heck, R. & Gleicher, M. (2007). Parametric motion graphs. In *Proceedings of the symposium on Interactive 3D graphics and games* (pp. 129–136).

Hecker, C., Raabe, B., Enslow, R. W., DeWeese, J., Maynard, J., & Van Prooijen, K. (2008). Real-time motion retargeting to highly varied user-created morphologies. *ACM Transactions on Graphics*, 27(3), 27:1–27:11.

Herda, L., Fua, P. V., Plankers, R., Boulic, R., & Thalmann, D. (2000). Skeleton-based motion capture for robust reconstruction of human motion. In *Proceedings of Computer Animation* (pp. 77–83).

Hodgins, J., Jörg, S., O'Sullivan, C., Park, S. I., & Mahler, M. (2010). The saliency of anomalies in animated human characters. *Transactions on Applied Perception*, 7(4), 22:1–22:14.

Hodgins, J. K., O'Brien, J. F., & Tumblin, J. (1998). Perception of human motion with different geometric models. *IEEE Transactions on Visualization and Computer Graphics*, 4(4), 307–316.

Hodgins, J. K., Wooten, W. L., Brogan, D. C., & O'Brien, J. F. (1995). Animating human athletics. In *Proceedings of the conference on Computer graphics and interactive techniques (SIGGRAPH '95)* (pp. 71–78).

Hreljac, A. & Marshall, R. N. (2000). Algorithms to determine event timing during normal walking using kinematic data. *Journal of Biomechanics*, 33(6), 783–786.

Hsu, E., Pulli, K., & Popović, J. (2005). Style translation for human motion. *ACM Transactions on Graphics*, 24(3), 1082–1089.

Ikemoto, L., Arikan, O., & Forsyth, D. (2006). Knowing when to put your foot down. In *Proceedings of the symposium on Interactive 3D graphics and games* (pp. 49–53).

Johansen, R. S. (2009). Dynamic Walking with Semi-Procedural Animation. In *Game Developers Conference 2009*.

Johansson, G. (1973). Visual perception of biological motion and a model for its analysis. *Perception And Psychophysics*, 14(2), 201–211.

Johnson, M. P. (2002). *Exploiting quaternions to support expressive interactive character motion*. PhD thesis, Massachusetts Institute of Technology.

Kahle, W., Leonhardt, H., & Platzer, W. (1986). *Color atlas and textbook of human anatomy, volume 1: Locomotor system*.

Kavan, L., Collins, S., & O'Sullivan, C. (2009). Automatic linearization of nonlinear skinning. In *Proceedings of the symposium on Interactive 3D graphics and games* (pp. 49–56).

Kavan, L., Collins, S., Žára, J., & O'Sullivan, C. (2007a). Skinning with dual quaternions. In *Proceedings of the symposium on Interactive 3D graphics and games* (pp. 39–46).

Kavan, L., Dobbyn, S., Collins, S., Žára, J., & O'Sullivan, C. (2008). Polypostors: 2D polygonal impostors for 3D crowds. In *Proceedings of the symposium on Interactive 3D graphics and games* (pp. 149–155).

Kavan, L., McDonnell, R., Dobbyn, S., Žára, J., & O'Sullivan, C. (2007b). Skinning arbitrary deformations. In *Proceedings of the symposium on Interactive 3D graphics and games* (pp. 53–60).

Kavan, L., Sloan, P. P., & O'Sullivan, C. (2010). Fast and efficient skinning of animated meshes. *Computer Graphics Forum*, 29(2), 327–336.

Kim, T.-H., Park, S. I., & Shin, S. Y. (2003). Rhythmic-motion synthesis based on motion-beat analysis. *ACM Transactions on Graphics*, 22(3), 392–401.

Kovar, L. & Gleicher, M. (2003). Flexible automatic motion blending with registration curves. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 214–224).

Kovar, L. & Gleicher, M. (2004). Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics*, 23(3), 559–568.

Kovar, L., Gleicher, M., & Pighin, F. (2002a). Motion graphs. *ACM Transactions on Graphics*, 21(3), 473–482.

Kovar, L., Schreiner, J., & Gleicher, M. (2002b). Footskate cleanup for motion capture editing. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 97–104).

Kozlowski, L. T. & Cutting, J. E. (1977). Recognizing the sex of a walker from a dynamic point-light display. *Perception And Psychophysics*, 21(6), 575–580.

Kulpa, R. & Multon, F. (2005). Fast inverse kinematics and kinetics solver for human-like figures. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots* (pp. 38–43).

Kurihara, K., Hoshino, S., Yamane, K., & Nakamura, Y. (2002). Optical motion capture system with pan-tilt camera tracking and real time data processing. In *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 1241–1248).

Kwon, T., Lee, K. H., Lee, J., & Takahashi, S. (2008). Group motion editing. *ACM Transactions on Graphics*, 27(3), 80:1–80:8.

Kwon, T. & Shin, S. Y. (2005). Motion modeling for on-line locomotion synthesis. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 29–38).

Lai, Y.-C., Chenney, S., & Fan, S. (2005). Group motion graphs. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 281–290).

Lamouret, A. & Van De Panne, M. (1996). Motion synthesis by example. In *Proceedings of the Eurographics workshop on Computer animation and simulation* (pp. 199–212).

Lawson, C. L. & Hanson, R. J. (1974). *Solving least squares problems*. Prentice-Hall.

Le Callennec, B. & Boulic, R. (2006). Robust kinematic constraint detection for motion data. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 281–290).

Lee, J. (2008). Representing rotations and orientations in geometric computing. *IEEE Computer Graphics and Applications*, 28(2), 75–83.

Lee, J., Chai, J., Reitsma, P. S. A., Hodgins, J. K., & Pollard, N. S. (2002). Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics*, 21(3), 491–500.

Lee, J. & Lee, K. H. (2004). Precomputing avatar behavior from human motion data. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 79–87).

Lee, J. & Shin, S. Y. (1999). A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of the conference on Computer graphics and interactive techniques (SIGGRAPH '99)* (pp. 39–48).

Lee, Y., Wampler, K., & Bernstein, G. (2010). Motion fields for interactive character animation. *ACM Transactions on Graphics*, 29(6), 138:1–138:8.

Lewis, J. P., Cordner, M., & Fong, N. (2000). Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the conference on Computer graphics and interactive techniques (SIGGRAPH '00)* (pp. 165–172).

Li, Y., Wang, T., & Shum, H. Y. (2002). Motion texture: a two-level statistical model for character motion synthesis. *ACM Transactions on Graphics*, 21(3), 465–472.

Lim, I. S. & Thalmann, D. (2002). Construction of animation models out of captured data. In *Proceedings of the IEEE International Conference on Multimedia and Expo* (pp. 829–832).

Liu, C. K. & Popović, Z. (2002). Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics*, 21(3), 408–416.

McAdams, A., Zhu, Y., Selle, A., Empey, M., Tamstorf, R., Teran, J., & Sifakis, E. (2011). Efficient elasticity for character skinning with contact and collisions. *ACM Transactions on Graphics*, 30(4), 37:1–37:12.

McDonnell, R., Dobbyn, S., & O'Sullivan, C. (2005). LOD Human Representations: A Comparative Study. In *Proceedings of the International Workshop on Crowd Simulation* (pp. 101–115).

McDonnell, R., Jörg, S., Hodgins, J. K., Newell, F., & O'Sullivan, C. (2007a). Virtual shapers & movers: form and motion affect sex perception. In *Proceedings of the symposium on Applied Perception in Graphics and Visualization* (pp. 7–10).

McDonnell, R., Larkin, M., Dobbyn, S., Collins, S., & O'Sullivan, C. (2008). Clone attack! Perception of crowd variety. *ACM Transactions on Graphics*, 27(3), 26:1–26:8.

McDonnell, R., Newell, F., & O'Sullivan, C. (2007b). Smooth movers: perceptually guided human motion simulation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 259–269).

Menache, A. (2000). *Understanding motion capture for computer animation and video games*. Morgan Kaufmann Publishers Inc.

Ménardais, S., Kulpa, R., Multon, F., & Arnaldi, B. (2004). Synchronization for dynamic blending of motions. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 325–335).

Merry, B., Marais, P., & Gain, J. (2006). Animation space: A truly linear framework for character animation. *ACM Transactions on Graphics*, 25(4), 1400–1423.

Mizuguchi, M., Buchanan, J., & Calvert, T. (2001). Data driven motion transitions for interactive games. In *Eurographics Short Presentations*.

Moeslund, T. B., Hilton, A., & Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2), 90–126.

Mohr, A. & Gleicher, M. (2003). Building efficient, accurate character skins from examples. *ACM Transactions on Graphics*, 22(3), 562–568.

Monzani, J.-S., Baerlocher, P., Boulic, R., & Thalmann, D. (2000). Using an intermediate skeleton and inverse kinematics for motion retargeting. *Computer Graphics Forum*, 19(3), 11–19.

Mukai, T. & Kuriyama, S. (2005). Geostatistical motion interpolation. *ACM Transactions on Graphics*, 24(3), 1062–1070.

Müller, M., Röder, T., & Clausen, M. (2005). Efficient content-based retrieval of motion capture data. *ACM Transactions on Graphics*, 24(3), 677–685.

NaturalMotion (2011). Morpheme.

O'Brien, J. F., Bodenheimer, R., Brostow, G. J., & Hodgins, J. K. (2000). Automatic joint parameter estimation from magnetic motion capture data. In *Proceedings of Graphics Interface* (pp. 53–60).

Onuma, K., Faloutsos, C., & Hodgins, J. K. (2008). FMDistance: A fast and effective distance function for motion capture data. In *Eurographics Short Paper Proceedings* (pp. 1–4).

Paris, S. & Donikian, S. (2009). Activity-driven populace: a cognitive approach to crowd simulation. *IEEE Computer Graphics and Applications*, 29(4), 34–43.

Park, S. I., Shin, H. J., Kim, T. H., & Shin, S. Y. (2004). On-line motion blending for real-time locomotion generation. *Computer Animation and Virtual Worlds*, 15(3-4), 125–138.

Park, S. I., Shin, H. J., & Shin, S. Y. (2002). On-line locomotion generation based on motion blending. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 105–111).

Perlin, K. (1995). Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics*, 1(1), 5–15.

Perlin, K. & Goldberg, A. (1996). Improv: A system for scripting interactive actors in virtual worlds. *Computers & Graphics*, 29(3), 205–216.

Pettré, J. & Laumond, J.-P. (2006). A motion capture-based control-space approach for walking mannequins. *Computer Animation and Virtual Worlds*, 17(2), 109–126.

Popović, Z. & Witkin, A. (1999). Physically based motion transformation. In *Proceedings of the conference on Computer graphics and interactive techniques (SIGGRAPH '99)* (pp. 11–20).

Pouli, T., Prazak, M., Zemcik, P., Gutierrez, D., & Reinhard, E. (2010). Rendering fur directly into images. *Computers & Graphics*, 34(5), 612–620.

Prazak, M., Hoyet, L., & O'Sullivan, C. (2011). Perceptual evaluation of footskate cleanup. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 287–294).

Prazak, M., Kavan, L., McDonnell, R., Dobbyn, S., & O'Sullivan, C. (2010a). Moving crowds: A linear animation system for crowd simulation. In *Poster Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*.

Prazak, M., McDonnell, R., Kavan, L., & O'Sullivan, C. (2008). Towards a perceptual metric for comparing human motion. In *Poster proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 13–14).

Prazak, M., McDonnell, R., Kavan, L., & O'Sullivan, C. (2009). A perception-based metric for comparing human locomotion. In *Proceedings of the 9th Irish Workshop on Computer Graphics* (pp. 75–80).

Prazak, M., McDonnell, R., & O'Sullivan, C. (2010b). Perceptual Evaluation of Human Animation Timewarping. In *ACM SIGGRAPH Asia 2010 Sketches* (pp. 30:1–30:2).

Prazak, M. & O'Sullivan, C. (2011). Perceiving human motion variety. In *Proceedings of the symposium on Applied Perception in Graphics and Visualization*.

Pullen, K. & Bregler, C. (2002). Motion capture assisted animation: texturing and synthesis. *ACM Transactions on Graphics*, 21(3), 501–508.

Reitsma, P. S. A. & Pollard, N. S. (2003). Perceptual metrics for character animation: sensitivity to errors in ballistic motion. *ACM Transactions on Graphics*, 22(3), 537–542.

Reitsma, P. S. A. & Pollard, N. S. (2007). Evaluating motion graphs for character animation. *ACM Transactions on Graphics*, 26(4), 18:1–18:24.

Ren, L., Patrick, A., Efros, A. A., Hodgins, J. K., & Rehg, J. M. (2005). A data-driven approach to quantifying natural human motion. *ACM Transactions on Graphics*, 24(3), 1090–1097.

Reynolds, C. W. (1987). Flocks, herds, and schools. *Computers & Graphics*, 21(4), 25–34.

Rose, C., Cohen, M. F., & Bodenheimer, B. (1998). Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5), 32–40.

Rose, C., Guenter, B., Bodenheimer, B., & Cohen, M. F. (1996). Efficient generation of motion transitions using spacetime constraints. In *Proceedings of the conference on Computer graphics and interactive techniques (SIGGRAPH '96)* (pp. 147–154).

Rose, C., Sloan, P.-P. J., & Cohen, M. F. (2001). Artist-directed inverse-kinematics using radial basis function interpolation. *Computer Graphics Forum*, 20(3), 239–250.

Ruttle, J., Manzke, M., Prazak, M., & Dahyot, R. (2009). Synchronized real-time multi-sensor motion capture system. In *ACM SIGGRAPH Asia 2009 Sketches & Posters* (pp. 16–19).

Safonova, A., Hodgins, J. K., & Pollard, N. S. (2004). Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics*, 23(3), 514–521.

Salvati, M., Le Callennec, B., & Boulic, R. (2004). A generic method for geometric constraints detection. In *Eurographics Short Presentations*.

Shapiro, A., Cao, Y., & Faloutsos, P. (2006). Style components. *Proceedings of Graphics Interface*, (pp. 33–39).

Shin, H. J. & Oh, H. S. (2006). Fat graphs: Constructing an interactive character with continuous controls. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 291–298).

Shoemake, K. (1985). Animating rotation with quaternion curves. In *Proceedings of the conference on Computer graphics and interactive techniques (SIGGRAPH '85)*, volume 19 (pp. 245–254).

Shoemake, K. & Duff, T. (1992). Matrix animation and polar decomposition. In *Proceedings of the conference on Graphics Interface* (pp. 258–264).

Sloan, P.-P. J., III, C. F. R., & Cohen, M. F. (2001). Shape by example. In *Proceedings of the symposium on Interactive 3D graphics* (pp. 135–143).

Slyper, R. & Hodgins, J. K. (2008). Action capture with accelerometers. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 193–199).

Sorkine, O. (2005). Laplacian Mesh Processing. In *Eurographics State of the Art Reports* (pp. 53–70).

Srinivasan, M., Metoyer, R. A., & Mortensen, E. N. (2005). Controllable real-time locomotion using mobility maps. In *Proceedings of Graphics Interface* (pp. 51–59).

Sung, M., Kovar, L., & Gleicher, M. (2005). Fast and accurate goal-directed motion synthesis for crowds. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 291–300).

Tolani, D., Goswami, A., & Badler, N. I. (2000). Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical Models and Image Processing*, 62(5), 353–388.

Tournier, M., Wu, X., Courty, N., & Arnaud, E. (2009). Motion compression using principal geodesics analysis. *Computer Graphics Forum*, 28(2), 335–364.

Troje, N. F. (2002). Decomposing biological motion: A framework for analysis and synthesis of human gait. *Journal of Vision*, 2(5), 371–387.

Troje, N. F. (2008). Retrieving information from human movement patterns. In *Understanding Events: How Humans See, Represent, and Act on Events* (pp. 308–334).

Unuma, M., Anjyo, K., & Takeuchi, R. (1995). Fourier principles for emotion-based human figure animation. In *Proceedings of the conference on Computer graphics and interactive techniques (SIGGRAPH '95)* (pp. 91–96).

Van Basten, B. J. H. & Egges, A. (2009). Evaluating distance metrics for animation blending. In *Proceedings of the International Conference on Foundations of Digital Games* (pp. 199–206).

van Welbergen, H., van Basten, B. J. H., Egges, A., Ruttkay, Z., & Overmars, M. H. (2010). Real time animation of virtual humans: A trade-off between naturalness and control. *Computer Graphics Forum*, 29(8), 2530–2554.

Wang, J. & Bodenheimer, B. (2003). An evaluation of a cost metric for selecting transitions between motion segments. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 232–238).

Wang, J. & Bodenheimer, B. (2004). Computing the duration of motion transitions: an empirical approach. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 335–344).

Wang, J., Drucker, S. M., Agrawala, M., & Cohen, M. F. (2006). The cartoon animation filter. *ACM Transactions on Graphics*, 25(3), 1169–1173.

Wang, L. C. T. & Chen, C. C. (1991). A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 7(4), 489–499.

Wang, X. C. & Phillips, C. (2002). Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 129–138).

"Web 3D Consortium" (2005). ISO/IEC FCD 19774 - Humanoid animation (H-Anim).

Welman, C. (1993). *Inverse kinematics and geometric constraints for articulated figure manipulation.* PhD thesis, Simon Fraser University.

Wichmann, F. A. & Hill, N. J. (2001). The psychometric function: I. Fitting, sampling and goodness-of-fit. *Perception And Psychophysics*, 63(8), 1293–1313.

Wiley, D. J. & Hahn, J. K. (1997). Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Applications*, 17(6), 39–45.

Witkin, A. & Popovic, Z. (1995). Motion warping. In *Proceedings of the conference on Computer graphics and interactive techniques (SIGGRAPH '95)* (pp. 105–108).

Yersin, B., Maïm, J., Pettré, J., & Thalmann, D. (2009). Crowd patches: populating large-scale virtual environments for real-time applications. In *Proceedings of the symposium on Interactive 3D graphics and games* (pp. 207–214).

Zhao, L. & Safonova, A. (2009). Achieving good connectivity in motion graphs. *Graphical Models*, 71(4), 139–152.

A

Matrix Algebra for Skeletal Character Animation

In the field of character animation, the skeleton is by far the most common way of describing the structure of the human body. While providing a good approximation of the musculo-skeletal system, it can be described using a simple hierarchical structure of rigid body matrices, which corresponds to the algebra commonly used in computer graphics. The notation established in this appendix corresponds to the main text of the thesis, providing a more concise overview of the underlying mathematical principles.

The **skeleton** is defined as a hierarchy of transformations, or *bones*, which describe the *base pose* (or *binding pose*) of the character. The animation frames then describe deformation related to this pose (see below). The base pose can have several different forms, depending on the method or algorithm it is intended for (see Figure A.1). The motion capture software (e.g., Vicon IQ) usually uses a "T" pose, as it offers a well-defined calibration pose without severe marker occlusion. The "I" pose is preferred by modelling and animation software (e.g., BVH file format, 3D Studio MAX), as it is the most natural standing pose for a character. Finally, an "A" pose is often used for skinning purposes, as it describes a pose approximately in the middle of the range for shoulders. However, the pose of the shoulder joint is often not well defined and can vary significantly between characters.

The *hierarchy* of bones is described as a directed graph without loops (i.e., directed tree). The bones form nodes of this graph, and are indexed using i

$$1 \leq i \leq n$$

where n is the total bone count and root node has always index 1. Edges are defined using a binary

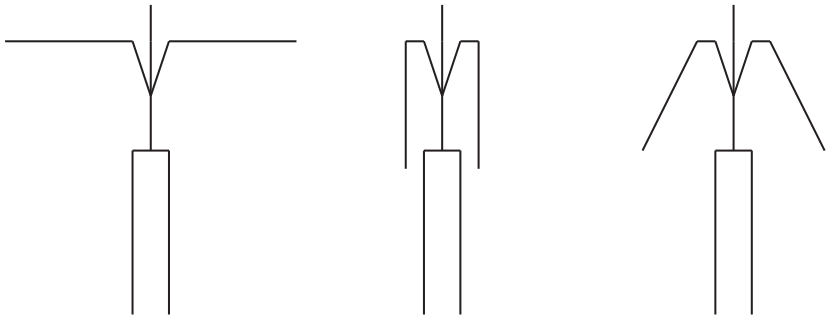

Figure A.1.: The base poses of a skeleton. From left – the "T" pose, used often in motion capture; the "I" pose, a common base pose for motion editing software; and the "A" pose, often used as a base pose for model skinning.

relation projecting each node with index i to its parent node $\pi(i)$, with

$$1 \leq \pi(i) \leq n$$
$$\pi(i) \neq i$$

Each *bone transformation* is a rigid body transformation $\mathbf{R}_i$, relative to its parent bone. The bones of a skeleton are then described as a vector

$$\boldsymbol{v}_{skel} = [\boldsymbol{R}_1, \boldsymbol{R}_2, \ ... \boldsymbol{R}_n]$$

In a similar manner, **animation keyframes** are represented as a vector of transformations $\boldsymbol{T}_i$, one for each bone of the associated skeleton

$$\boldsymbol{v}_{frame} = [\boldsymbol{T}_1, \boldsymbol{T}_2, \ ... \boldsymbol{T}_n]$$

Their purpose is to represent the change of the skeletal pose in time. This alters the binding pose transformations $\boldsymbol{R}_i$, creating animated poses $\boldsymbol{P}_i$

$$\boldsymbol{P}_i = \boldsymbol{R}_i \boldsymbol{T}_i$$

The $\boldsymbol{P}_i$ matrices are often called the *premultiplied format*, a common way of storing animation data (saving one matrix multiplication)

$$\begin{aligned} \boldsymbol{v}_{premult} &= [\boldsymbol{P}_1, \boldsymbol{P}_2, , \ ... \boldsymbol{P}_n] \\ &= [\boldsymbol{R}_1\boldsymbol{T}_1, \boldsymbol{R}_2\boldsymbol{T}_2, \ ... \boldsymbol{R}_n\boldsymbol{T}_n] \end{aligned}$$

All the description above uses matrices in **local format**, i.e., each transformation is relative to its parent, recursively. However, for display and practical use, we often need to convert this representation to the **global format**, with each transformation relative to the scene origin. This effectively discards the hierarchical nature of the transformation representation. To create a global representation $\boldsymbol{A}_i$ of each bone of the skeleton in its *binding pose*, we concatenate all the bone transformation recursively:

$$\begin{aligned} \boldsymbol{A}_i &= \boldsymbol{A}_{\pi(i)}\boldsymbol{R}_i \\ &= \boldsymbol{R}_1 ... \boldsymbol{R}_{\pi(\pi(i))}\boldsymbol{R}_{\pi(i)}\boldsymbol{R}_i \end{aligned}$$

To create a global representation $\boldsymbol{F}_i$ of each bone of the *animated* skeleton, we concatenate the transformation matrices in premultiplied format

$$\begin{aligned} \boldsymbol{F}_i &= \boldsymbol{F}_{\pi(i)}\boldsymbol{P}_i \\ &= \boldsymbol{P}_1 ... \boldsymbol{P}_{\pi(\pi(i))}\boldsymbol{P}_{\pi(i)}\boldsymbol{P}_i \\ &= \boldsymbol{F}_{\pi(i)}\boldsymbol{R}_i\boldsymbol{T}_i \\ &= \boldsymbol{R}_1\boldsymbol{T}_1 ... \boldsymbol{R}_{\pi(\pi(i))}\boldsymbol{T}_{\pi(\pi(i))}\boldsymbol{R}_{\pi(i)}\boldsymbol{T}_{\pi(i)}\boldsymbol{R}_i\boldsymbol{T}_i \end{aligned}$$

Therefore, using the premultiplied representation, it is possible to save one matrix multiplication per bone when converting to the global format.

Finally, the **skinning format** describes an animated bone transformation, relative to its respective binding position:

$$\boldsymbol{C}_i = \boldsymbol{F}_i \left(\boldsymbol{A}_i\right)^{-1}$$

This representation can be used to transform the vertices of the base model to correspond to the new skeletal pose in the skinning process. Each vertex $\boldsymbol{v}_j$, described as a homogeneous 3D vector

$$\boldsymbol{v}_j = \begin{bmatrix} v_j^{(x)} \\ v_j^{(y)} \\ v_j^{(z)} \\ 1 \end{bmatrix}$$

can be transformed into its deformed position $\boldsymbol{v}'_j$ using the equation:

$$\boldsymbol{v}'_j = \sum_{i=1}^{n} w_{i,j}\boldsymbol{C}_i\boldsymbol{v}_j$$

where $w_{i,j}$ is the skinning weight associating the vertex j with bone i. These weights are adjusted manually by the artist during the animated character creation process, and by definition have the following properties:

$$\begin{gathered} 0 \leq w_{i,j} \leq 1 \\ \textstyle\sum_{i=1}^{n} w_{i,j} = 1 \end{gathered}$$

with the number of non-zero weights for a particular vertex traditionally limited to 4.

B

Least Squares 3D Plane Fit

While simpler and more efficient approaches exist, a simple way of solving least squares plane fit is using Singular Value Decomposition (SVD). The SVD implementation is not trivial, however there are many libraries and packages providing this functionality (i.e., the BLAS library). Moreover, the use of this algorithm in this thesis is only in the preprocessing, and consequently it is not computationally critical.

The problem can be summarised as fitting a plane into a set of points:

$$\boldsymbol{v}_i = \left[v_x^{(i)}, v_y^{(i)}, v_z^{(i)}\right], \; i = (1..m)$$

while minimising the square distance of each point to the plane. The plane can be described in three ways:

- as a point $\boldsymbol{r}_0$ and a normal vector $\boldsymbol{n}$,
- as a point $\boldsymbol{r}_0$ and two non-parallel vectors $\boldsymbol{s}$ and $\boldsymbol{t}$, or
- as a normal vector $\boldsymbol{n}$ and a scalar distance c

Each point

$$\boldsymbol{r} = [r_x, r_y, r_z]$$

on the plane then must satisfy

$$\boldsymbol{n} \cdot (\boldsymbol{r} - \boldsymbol{r}_0) = \boldsymbol{n} \cdot \boldsymbol{r} + c = 0$$

or in the parametric representation

$$\boldsymbol{r} = \boldsymbol{r}_0 + a\boldsymbol{s} + b\boldsymbol{t}$$

Even though the only requirement on $\boldsymbol{s}$ and $\boldsymbol{t}$ is that they are not parallel, we will assume in the following description, without loss of generality, that they are perpendicular, creating a 2D orthonormal basis. This can be enforced by a simple cross-product manipulation of one of these vectors. Using

these terms, we can define the minimisation error function e as:

$$e\left(\boldsymbol{n}, c\right) = \sum_{i=1}^{m} \left(\boldsymbol{n} \cdot \boldsymbol{v}_i + c\right)^2$$

First, we solve the minimisation for c, thus finding an extremum of $e\left(c\right)$:

$$\begin{aligned}
\frac{e\left(\boldsymbol{n}, c\right)}{dc} &= 0 \\
2\sum_{i=1}^{m} \left(\boldsymbol{n} \cdot \boldsymbol{v}_i + c\right) &= 0 \\
\sum_{i=1}^{m} \left(n_x v_x^{(i)} + n_y v_y^{(i)} + n_z v_z^{(i)} + c\right) &= 0 \\
n_x \sum_{i=1}^{m} v_x^{(i)} + n_y \sum_{i=1}^{m} v_y^{(i)} + n_z \sum_{i=1}^{m} v_z^{(i)} + mc &= 0 \\
c &= -\left(n_x \bar{v}_x + n_y \bar{v}_y + n_z \bar{v}_z\right)
\end{aligned}$$

where

$$\bar{\boldsymbol{v}} = \left[\bar{v}_x, \bar{v}_y, \bar{v}_z\right]$$

is the centroid of the data. This signifies, that the resulting plane includes the centroid of the data, therefore we can state that in the parametric equation above:

$$\boldsymbol{r}_0 = \bar{\boldsymbol{v}}$$

and we can convert the input data to a centered version by subtracting the cetroid:

$$\boldsymbol{v}'_i = \boldsymbol{v}_i - \bar{\boldsymbol{v}}$$

Next, we define a matrix $\boldsymbol{A}$, which contains all the data:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{v}'_1 \\ \boldsymbol{v}'_2 \\ \cdots \\ \boldsymbol{v}'_m \end{bmatrix}$$

By performing the Principal Component Analysis (PCA) on these data, we obtain a set of orthogonal eigenvectors, with the first pointing the direction of highest variance, the second representing the direction of highest variance after the projection of the data into subspace defined by remaining vectors etc. The PCA can be performed using the SVD algorithm, decomposing the matrix $\boldsymbol{A}$ into three components:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}$$

where $\boldsymbol{V}$ describes the new orthonormal basis, from which we can derive the vectors of the plane equation:

$$\boldsymbol{V} = \left[\boldsymbol{s}^T, \boldsymbol{t}^T, \boldsymbol{n}^T\right]$$

By projecting the points into the coordinate system defined by $\boldsymbol{V}$, we can also obtain the coordinates of the points projected onto the plane in 2D coordinate system defined by$\boldsymbol{s}$ and $\boldsymbol{t}$:

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{V}$$

with the 2D coordinates described by first two columns of the matrix $\boldsymbol{Y}$.

C

2D Least Squares Circle Fit

The input to the 2D least squares circle fitting is a set of 2D points:

$$\boldsymbol{v}_i = \left[v_x^{(i)}, v_y^{(i)}\right], i = (1..m)$$

into which we want to fit a circle described using the equation

$$(x - c_x)^2 + (y - c_y)^2 = r^2$$

where $[x, y]$ are the coordinates of points on the circle, $\boldsymbol{c} = [c_x, c_y]$ are coordinates of the centre and r is the circle's diameter. To fit a circle into a set of points, we want to minimise the error function

$$e\left(r^2, c_x, c_y\right) = \left(\sum_{i=1}^{m} \left(v_x^{(i)} - c_x\right)^2 + \left(v_y^{(i)} - c_y\right)^2 - r^2\right)^2$$

which leads to three independent parameters. To determine the extrema of the error function, we first differentiate it by r^2:

$$\begin{aligned} \frac{\partial e\left(r^2, c_x, c_y\right)}{\partial\left(r^2\right)} &= 2\sum_{i=1}^{m}\left(\left(v_x^{(i)} - c_x\right)^2 + \left(v_y^{(i)} - c_y\right)^2 - r^2\right)(-1) \\ &= -2\sum_{i=1}^{m}\left(v_x^{(i)} - c_x\right)^2 + \left(v_y^{(i)} - c_y\right)^2 - r^2 \end{aligned}$$

with the extrema present when the first derivative is equal to zero:

$$\sum_{i=1}^{m}\left(v_x^{(i)} - c_x\right)^2 + \left(v_y^{(i)} - c_y\right)^2 - r^2 = 0$$

In a similar manner, we can differentiate by c_x:

$$\begin{aligned}
\frac{\partial e\left(r^2, c_x, c_y\right)}{\partial\left(c_x\right)} &= 2\sum_{i=1}^{m}\left(\left(v_x^{(i)}-c_x\right)^2+\left(v_y^{(i)}-c_y\right)^2-r^2\right)2\left(v_x^{(i)}-c_x\right)(-1) \\
&= -4\sum_{i=1}^{m}\left(\left(v_x^{(i)}-c_x\right)^2+\left(v_y^{(i)}-c_y\right)^2-r^2\right)\left(v_x^{(i)}-c_x\right) \\
&= -4\sum_{i=1}^{m} v_x^{(i)}\left(\left(v_x^{(i)}-c_x\right)^2+\left(v_y^{(i)}-c_y\right)^2-r^2\right)+ \\
&\quad +4c_x\sum_{i=1}^{m}\left(v_x^{(i)}-c_x\right)^2+\left(v_y^{(i)}-c_y\right)^2-r^2
\end{aligned}$$

and merging with the equation for $\partial e/\partial\left(r^2\right)$ above, we obtain:

$$\begin{aligned}
\frac{\partial e\left(r^2, c_x, c_y\right)}{\partial\left(c_x\right)} &= -4\sum_{i=1}^{m} v_x^{(i)}\left(\left(v_x^{(i)}-c_x\right)^2+\left(v_y^{(i)}-c_y\right)^2-r^2\right)+4c_x\frac{\partial e}{\partial\left(r^2\right)} \\
&= -4\sum_{i=1}^{m} v_x^{(i)}\left(\left(v_x^{(i)}-c_x\right)^2+\left(v_y^{(i)}-c_y\right)^2-r^2\right)
\end{aligned}$$

Solving for the zero result (extrema):

$$\begin{aligned}
\frac{\partial e\left(r^2, c_x, c_y\right)}{\partial\left(c_x\right)} &= 0 \\
\sum_{i=1}^{m} v_x^{(i)}\left(\left(v_x^{(i)}-c_x\right)^2+\left(v_y^{(i)}-c_y\right)^2-r^2\right) &= 0 \\
\sum_{i=1}^{m} v_x^{(i)}\left(\left(v_x^{(i)}\right)^2-2v_x^{(i)}c_x+c_x^2+\left(v_y^{(i)}\right)^2-2v_y^{(i)}c_y+c_y^2-r^2\right) &= 0 \\
\sum_{i=1}^{m}\left(v_x^{(i)}\right)^3-2c_x\sum_{i=1}^{m}\left(v_x^{(i)}\right)^2+c_x^2\sum_{i=1}^{m} v_x^{(i)}+ & \\
+\sum_{i=1}^{m} v_x^{(i)}\left(v_y^{(i)}\right)^2-2c_y\sum_{i=1}^{m} v_x^{(i)}v_y^{(i)}+c_y^2\sum_{i=1}^{m} v_x^{(i)}-r^2\sum_{i=1}^{m} v_x^{(i)} &= 0
\end{aligned}$$

To simplify the equation above, we can compute the mean $\bar{\boldsymbol{v}}$ of all points $\boldsymbol{v}_i$:

$$\bar{\boldsymbol{v}}=\frac{1}{m}\sum_{i=1}^{m}\boldsymbol{v}_i$$

and by describing each point $\boldsymbol{v}_i$ in relation to this mean:

$$\boldsymbol{v}_i'=\boldsymbol{v}_i-\bar{\boldsymbol{v}}$$

we can conclude that:

$$\begin{aligned}\sum_{i=1}^{m} \boldsymbol{v}_i' &= \sum_{i=1}^{m} \boldsymbol{v}_i - m\bar{\boldsymbol{v}} \\ &= \sum_{i=1}^{m} \boldsymbol{v}_i - m\left(\frac{1}{m}\sum_{i=1}^{m} \boldsymbol{v}_i\right) \\ &= 0\end{aligned}$$

Describing the $\boldsymbol{c}$ in the same terms:

$$\boldsymbol{c}' = \boldsymbol{c} - \bar{\boldsymbol{v}}$$

the equation above then simplifies to:

$$\begin{aligned}\sum_{i=1}^{m}\left(v_x^{(i)}\right)^3 - 2c_x\sum_{i=1}^{m}\left(v_x^{(i)}\right)^2 + c_x^2\sum_{i=1}^{m} v_x^{(i)} + \\ + \sum_{i=1}^{m} v_x^{(i)}\left(v_y^{(i)}\right)^2 - 2c_y\sum_{i=1}^{m} v_x^{(i)}v_y^{(i)} + c_y^2\sum_{i=1}^{m} v_x^{(i)} - r^2\sum_{i=1}^{m} v_x^{(i)} &= 0 \\ \sum_{i=1}^{m}\left(v_x^{'(i)}\right)^3 - 2c_x'\sum_{i=1}^{m}\left(v_x^{'(i)}\right)^2 + \sum_{i=1}^{m} v_x^{'(i)}\left(v_y^{(i)}\right)^2 - 2c_y'\sum_{i=1}^{m} v_x^{'(i)}v_y^{'(i)} &= 0\end{aligned}$$

therefore:

$$c_x'\sum_{i=1}^{m}\left(v_x^{'(i)}\right)^2 + c_y'\sum_{i=1}^{m} v_x^{'(i)}v_y^{'(i)} = \frac{1}{2}\left(\sum_{i=1}^{m}\left(v_x^{'(i)}\right)^3 + \sum_{i=1}^{m} v_x^{'(i)}\left(v_y^{'(i)}\right)^2\right)$$

Similarly for c_y:

$$c_x'\sum_{i=1}^{m} v_x^{'(i)}v_y^{'(i)} + c_y'\sum_{i=1}^{m}\left(v_y^{'(i)}\right)^2 = \frac{1}{2}\left(\sum_{i=1}^{m}\left(v_y^{'(i)}\right)^3 + \sum_{i=1}^{m}\left(v_x^{'(i)}\right)^2 v_y^{'(i)}\right)$$

Which finally leads to a linear set of equations, that can be easily solved. From the solution, we can determine the actual centre coordinates by:

$$\boldsymbol{c} = \boldsymbol{c}' + \bar{\boldsymbol{v}}$$

To solve the radius, we can expand the term for $\partial e/\partial\left(r^2\right)$:

$$\begin{aligned}\sum_{i=1}^{m}\left(v_x^{'(i)} - c_x'\right)^2 + \left(v_y^{'(i)} - c_y'\right)^2 - r^2 &= 0 \\ \sum_{i=1}^{m}\left(v_x^{'(i)}\right)^2 - 2c_x'\sum_{i=1}^{m} v_x^{'(i)} + mc_x^{'2} + \sum_{i=1}^{m}\left(v_y^{'(i)}\right)^2 - 2c_y'\sum_{i=1}^{m} v_y^{'(i)} + mc_y^{'2} - mr^2 &= 0 \\ \sum_{i=1}^{m}\left(v_x^{'(i)}\right)^2 + \sum_{i=1}^{m}\left(v_y^{'(i)}\right)^2 + mc_x^{'2} + mc_y^{'2} - mr^2 &= 0 \\ c_x^{'2} + c_y^{'2} + \frac{\sum_{i=1}^{m}\left(v_x^{'(i)}\right)^2 + \sum_{i=1}^{m}\left(v_y^{'(i)}\right)^2}{m} &= r^2\end{aligned}$$