Parametric Synthesis of Human Animation

Amr Adel Hassan Ahmed

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey

Centre for Vision, Speech and Signal Processing
School of Electronics and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

April 2004

© Amr Adel Hassan Ahmed 2004
Summary

Virtual humans are the computer representation of real humans in virtual worlds. Human representation in virtual worlds is vital as people are central elements of the real and virtual worlds. Animating virtual humans remains a challenging task due to the large number of degrees of freedom in the human skeleton, the ability of human observers to detect unnatural movements, and the complexity of realistic appearance and behaviour. The difficulty of the problem has resulted in research into many sub-problems of human animation synthesis.

In this thesis, we focus on the motion control problem. The objective is to synthesise realistic human motion with improved control over the resulting animation. Motion capture of real actors provides a source of realistic animation sequences. Many existing motion editing techniques require an expert or trained animator to deal with complex models or to define the desired motion by manipulating low-level variables. A few high-level editing techniques have been recently introduced. However, their common drawbacks include the need of relatively large database of animation sequences and the high computation complexity.

This thesis introduces a high-level parametric approach for synthesis of realistic human animation based on synchronised blending of existing motion clips. The proposed approach allows intuitive control of the desired motions while generating realistic animation based on a small motion captured database of basic motion clips. A framework has been developed and utilised to realise interesting and challenging tasks such as parametric synthesis of animation sequences along arbitrary paths and over uneven terrains. The extension for control of multiple parameters simultaneously is also presented. Other contributions have been made to allow the synthesis of long animation sequences from short animation clips which reduces the input data requirements and costs.

The presented approach, based on motion blending, is computationally efficient with linear computation complexity with length of the desired animation sequence. It also requires a relatively small database of basic motion clips.

Visual inspection is the ultimate evaluation of animation. However, a number of quantitative measures are introduced and adopted in this thesis to provide consistent and systematic evaluation of the resulting animation. Results demonstrate realistic synthesis of novel motion sequences using intuitive parameters such as speed, direction, and slope.

Key words: Human Animation, Animation Synthesis, Motion Capture, Character Animation, Blending, Interpolation, Virtual Human, Virtual People, 3D Graphics, Wavelet Analysis.

Email: A.Ahmed@surrey.ac.uk
WWW: http://www.ee.surrey.ac.uk/Personal/A.Ahmed/
Acknowledgements

It is my pleasure and honor to dedicate this thesis to my Dad, Mum, my wife, daughter, and my son.

I would like to thank my parents for their continuous support, encouraging, and concern, throughout my life in general and remotely during the PhD period. There is no words that can give justice in thanking parents. Hence, I can not say anymore!

Thanks to my wife, daughter, and son for their patient all the way through my PhD period. I know it was not an easy period. A special Thank You for my wife for her efforts in taking the children's responsibilities and releasing me to concentrate in my study.

Thanks to my supervisors; Dr Farzin Mokhtarian and Dr Adrian Hilton, for their support, follow-up, and constructive comments all the way through, especially during the writing-up of this thesis.

Thanks to all colleagues in CVSSP and in the 3D group. Thanks to Dr James Smith, Dr Jonathan Starck, Dr Joel Mitchelson, Dr Luis Molina-Tanco, and Dr Gordon Colin for the cooperation and various support and discussions. And thanks to Dr Eng-Jon and Dr Jonathan Starck for reading various parts of my thesis and for their constructive feedback in the early stages of writing-up. Thanks to Dr Joel Mitchelson for his help with the motion capture system. Also, thanks to Robert King and Nabeen Ajmal who volunteered for the motion capture sessions.

A special ‘Thank You’ for Dr Djelloul Mahboub for all his efforts, continuous encouragement, and extreme support, both scientifically and morally, which is continuing up to the last minute of printing this thesis. Also thanks to Eng. Amr Ibrahim for his remote support during my PhD period.

I would also like to thank all the friends and families who have been providing their help and support for me and my family during our stay in Guildford. The list is too long to be included. However, we are really very grateful for everyone of you.

Finally, this PhD project was fully funded by the Egyptian Government through the Suez Canal University, where I have been working as an Assistant Lecturer. I would like to thank Prof. Ahmed El-Nekhaily, the Dean of the Faculty of Industrial Education, all my colleagues over there and everyone contributed in the nomination process. Also, I would like to thank everyone contributed in the administration and all the other aspects related to this scholarship.
## Contents

1 Introduction ................................................. 1
   1.1 Motivation .................................................. 3
   1.2 Publications and Summary of Contributions .......... 4
   1.3 Thesis Structure and Overview ......................... 6

2 Human Motion Animation: A Literature Review .............. 9
   2.1 Computer Graphics, Animation, and Simulation .......... 9
   2.2 Virtual Worlds and Virtual People ....................... 10
   2.3 Biomechanics of Human Motion ........................... 11
   2.4 Computer Animation of Human Motion .................... 14
       2.4.1 Traditional Methods ................................. 14
           2.4.1.1 Manual and Key-Frames Animation .......... 14
           2.4.1.2 Kinematics Animation ........................ 16
       2.4.2 Dynamics Animation .................................. 18
       2.4.3 Motion Capture Animation .......................... 20
   2.5 Motion Editing ............................................. 21
       2.5.1 Signal Processing ..................................... 22
       2.5.2 Constraint-based editing and Optimisation ........ 23
       2.5.3 Statistical Analysis .................................. 24
       2.5.4 Transition Graphs ..................................... 25
       2.5.5 Interpolation and Blending ......................... 27
   2.6 Motion Retargeting ........................................ 30
   2.7 Animation Sequences/Streams .............................. 30
       2.7.1 Motion Annotation .................................... 31
       2.7.2 Motion Cyclification ................................. 32
   2.8 This Thesis on the Research Map .......................... 33
   2.9 Summary ....................................................... 34
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Aim and Motivation</td>
<td>36</td>
</tr>
<tr>
<td>3.2</td>
<td>Problem Definition</td>
<td>36</td>
</tr>
<tr>
<td>3.3</td>
<td>Data Representation</td>
<td>39</td>
</tr>
<tr>
<td>3.4</td>
<td>Proposed Parametric Approach</td>
<td>41</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Synchronised Motion Blending</td>
<td>42</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Solving the Inverse Interpolation Problem</td>
<td>51</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Motion Annotation</td>
<td>53</td>
</tr>
<tr>
<td>3.4.4</td>
<td>Motion Synchronisation</td>
<td>56</td>
</tr>
<tr>
<td>3.5</td>
<td>Results and Evaluation</td>
<td>59</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Quantitative Evaluation</td>
<td>60</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Walk on a level floor</td>
<td>62</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Run</td>
<td>65</td>
</tr>
<tr>
<td>3.5.4</td>
<td>Walk on a sloped floor</td>
<td>65</td>
</tr>
<tr>
<td>3.5.5</td>
<td>Jump</td>
<td>68</td>
</tr>
<tr>
<td>3.5.6</td>
<td>Summary of Synthesis Results</td>
<td>68</td>
</tr>
<tr>
<td>3.5.7</td>
<td>Automatic Gait Analysis</td>
<td>74</td>
</tr>
<tr>
<td>3.6</td>
<td>Applications</td>
<td>77</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Encapsulated and Layered Animation database</td>
<td>77</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Scaled Avatars</td>
<td>81</td>
</tr>
<tr>
<td>3.7</td>
<td>Summary</td>
<td>83</td>
</tr>
<tr>
<td>4.1</td>
<td>Aim and Motivation</td>
<td>86</td>
</tr>
<tr>
<td>4.2</td>
<td>Extended Problem Definition</td>
<td>86</td>
</tr>
<tr>
<td>4.3</td>
<td>Continuous Parametric Synthesis</td>
<td>87</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Extending Input Motion Clips</td>
<td>88</td>
</tr>
<tr>
<td>4.3.1.1</td>
<td>Cyclification guided by part of an additional cycle</td>
<td>90</td>
</tr>
<tr>
<td>4.3.1.2</td>
<td>Self Cyclification</td>
<td>92</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Continuous Parameterisation</td>
<td>97</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Improved Root Trajectory Blending</td>
<td>99</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Synchronised Mirror</td>
<td>109</td>
</tr>
</tbody>
</table>
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B Wavelet and Direct Blending</td>
<td>183</td>
</tr>
<tr>
<td>B.1 Direct Blending</td>
<td>183</td>
</tr>
<tr>
<td>B.2 Wavelet Blending</td>
<td>184</td>
</tr>
<tr>
<td>B.3 Summary</td>
<td>186</td>
</tr>
<tr>
<td>C Qualitative Evaluation</td>
<td>187</td>
</tr>
<tr>
<td>C.1 Methodology</td>
<td>187</td>
</tr>
<tr>
<td>C.2 Results</td>
<td>190</td>
</tr>
<tr>
<td>C.3 Discussion</td>
<td>191</td>
</tr>
<tr>
<td>D Comparisons</td>
<td>193</td>
</tr>
</tbody>
</table>
List of Figures

1.1 Principle branches of ‘Virtual Humans’ research .......................... 2

2.1 Common methods in Human Animation research .......................... 15

3.1 Block diagram of the Parametric Animation approach .................. 43

3.2 Procedure of adding new motion clips to the animation database ..... 44

3.3 Parametric Synthesis Procedure ............................................ 45

3.4 Real motion capture straight-path walking (in blue) compared with its corresponding synthesised motion (in red) from curved-path walking motions .................................................. 48

3.5 Real motion capture straight-path walking (in blue) compared with its corresponding synthesised motion (in red) from Sloped-floor walking motions .................................................. 48

3.6 Key-event times for both Real and synthesised motions (from curved-path clips) for a straight-line walking movement. .................................................. 49

3.7 Key-event times for both Real and synthesised motions (from Sloped-floor clips) for a straight-line walking movement. .................................................. 49

3.8 Variations of Key-event times in real human movement. ............... 50

3.9 Key-event times for both synthesised and ground-truth motions (from a procedural method) .......................................................... 51

3.10 Example of the function $P = g(X)$: Walking Speed ‘V’ as a function of blending factor ‘X’ .......................................................... 53

3.11 Example of the function $X = f(P)$: Blending factor ‘X’ as a function of Walking Speed ‘V’ .......................................................... 54

3.12 Automatic detection of gait timing ............................................. 56

3.13 Markers setup on the performer for a Motion Capture session .......... 59

3.14 Synthesised ‘Walk’ motion on level floor with desired speed given by user. Original clips are ‘SlowWalk’ (left in blue) with speed of 73.74 cm/sec and ‘FastWalk’ (right in green) with speed of 150.6 cm/sec. 63
List of Figures

3.15 Output speed vs. Input speed of Walk examples. ..................... 64
3.16 Speed profile of the Root joint for both input clips and a sample output clip. ......................................................... 64
3.17 Evaluation of result of controlling the speed of 'walk' motions on level floor. The two ends of each curve represent the measure value of input motions. ................................. 65
3.18 Samples of generating 'Run' motion with a specified speeds given by the user. Original clips are 'SlowRun' (right in blue) with speed of 863.5 cm/sec and 'FastRun' (left in green) with speed of 1452.7 cm/sec. .. 66
3.19 Output speed vs. Input speed of Run examples. ..................... 67
3.20 Evaluation of result of controlling the speed of 'run' motions. The two ends of each curve represent the measure value of input motions. ................................. 67
3.21 Sample of generating motions on specified slopes (up or down hill). Original clips (not shown in figure) 'UpSlopeWalk' with slope of 12.5° and 'DownSlopeWalk' with slope of 12.6°. .................................................. 69
3.22 Output slope vs. Input slope of Walk on sloped floor examples. ... 70
3.23 Evaluation of result of controlling the slope of the floor. The two ends of each curve represent the measure value of input motions. ................................. 70
3.24 Jumping example where user controls the facing direction at the end of the jump. Original clips are 'JumpInPlace' (right in blue) 'JumpTurn' (left in green) with turn-angle of $-91.8^\circ$. .................................................. 71
3.25 Output speed vs. Input speed of Walk on sloped floor examples. ... 72
3.26 Evaluation of result of controlling the jumping turn angle. The two ends of each curve represent the measure value of input motions. ................................. 72
3.27 Jerk measure calculated without the final time re-warping stage. .... 73
3.28 Automatically detected key-event times (in magenta) for normal gait motions versus the corresponding manually detected ones (in blue). .................................. 75
3.29 Automatically detected key-event times (in magenta) for Special/Uppnormalgait motions versus the corresponding manually detected ones (in blue). .................................. 76
3.30 Automatically detected key-event times (in magenta) for 'kick' motion versus the corresponding manually detected ones (in blue). .................................. 76
3.31 The Layered Animation Database diagram .................................. 78
3.32 Examples of enriching animation database contents .................. 80
3.33 Example of parametric synthesis of animation for scaled characters. 'A' is the original size character, 'B' is the scaled character (with original animation data), 'C' is the scaled character (with modified animation data) .................................. 81
List of Figures

4.1 Example of ‘Mirrored Half-Cycle’ motion category (Walk) .................. 93
4.2 Self Cyclification algorithm for ‘Mirrored Half-cycle’ motion category 94
4.3 This diagram explains the self-cyclification algorithm for ‘Mirrored Half-cycle’ category .......................................................... 95
4.4 Example of ‘Reversed Half-cycle’ motion category (Kicking) ............ 95
4.5 Self Cyclification algorithm for ‘Reversed Half-cycle’ motion category 96
4.6 Continuous parameterisation ............................................................. 99
4.7 Absolute vs. Relative Root Blending. The figure shows the root-positions trajectory, projected on the floor plane, for random values of blending factor. .................................................. 101
4.8 Explanation of the Root Blending problem inherited from Vector Analysis ..................................................................................... 104
4.9 The Synthesised root trajectory (middle) is suddenly flipped to the opposite direction after the difference in direction of input motions exceeds 180°. .................................................... 105
4.10 Explanation of the Root Blending problem when the difference in input directions exceeds 180°. The root trajectory flips when moving from frame j (left) to frame i (right) with i > j and \( \theta_j < 180° < \theta_i \) .... 105
4.11 The synthesised root trajectory (middle) has no sudden flipping any more after using the improved blending technique. .................. 108
4.12 Blending original (left) and ordinary mirrored clip (right) produces invalid motion (middle) as the mirrored clip is out-of-phase with the original clip. ................................................. 110
4.13 Synchronised Mirror algorithm, ‘SyncMirror’. ............................... 111
4.14 This diagram explains the synchronised mirror algorithm, ‘SyncMirror’ of figure 4.13 .............................................................. 112
4.15 Motion directions extracted from trajectory of root positions for different walking motions (ranges from straight-line walking [bottom] to curved-path walking [top]) ................................................. 114
4.16 Motion directions extracted from root rotations about vertical axis. The example is for walking motions (ranges from straight-line walking [bottom] to curved-path walking [top]) ............. 116
4.17 Relationships between motion directions and corresponding blending factors at each frame .................................................. 117
4.18 Approximated Relationships between motion directions and correspond- ing blending factors at each frame .................................. 118
4.19 Input clips for the PathControl module. One cycle of (from left to right) StraightPath, CurvedPath (CW), SynchronisedMirror CurvedPath (CCW) motions. .............................. 118
4.20 The desired arbitrary path drawn by the user on the developed *Path Control Pad* .................................................. 119

4.21 Algorithm for Parametric Synthesis for motion on arbitrary path ................................................................. 120

4.22 The output generated from the user-input arbitrary path drawn in figure 4.20. The synthesised motion path is shown in *green* while the desired path drawn by the user is shown in *yellow* .................................................. 121

4.23 Relationship between floor slope and the corresponding blending factor .......................................................... 124

4.24 Input clips for the *TerrainControl* module. *UpSlope* cycle (Left) and *DownSlope* (Right) ........................................ 124

4.25 Algorithm for Parametric Synthesis of motion over *arbitrary uneven terrain* .................................................. 125

4.26 Input motions on the parameters space (Irregular Grid) ................................................................. 126

4.27 Regulated grid of input motions ................................................................. 127

4.28 Pseudo Code for Generating Parameterisation data for Multi-parameters Synthesis .................................................. 128

4.29 Plot of the 1st Parameterisation function for controlling *speed* over uneven terrain .................................................. 129

4.30 Plot of the 2nd Parameterisation function for controlling *speed* over uneven terrain ........................................ 130

4.31 Plot of the 1st Parameterisation function for controlling the *hitting position* of a *kicking* action ........................................ 130

4.32 Plot of the 2nd Parameterisation function for controlling the *hitting position* of a *kicking* action ........................................ 131

4.33 Input clips for the *PathControl* module. One cycle of (from left to right) *StraightPath*, *CurvedPath* (CW), *SynchronisedMirror CurvedPath* (CCW), and *TurnBack* motions ........................................ 133

4.34 Example of following an arbitrary path drawn by the user. The drawn path is shown in *yellow* while the synthesised path is shown in *green* ........................................ 134

4.35 Example of following an arbitrary path representing the handwritten name *Amr*. The drawn path is shown in *yellow* while the synthesised path is shown in *green* ........................................ 135

4.36 Example of following an arbitrary path representing the handwritten word *ACM*. The drawn path is shown in *yellow* while the synthesised path is shown in *green* ........................................ 135

4.37 Quantitative evaluation of 'PathControl' examples ........................................ 136

4.38 Input clips for the *TerrainControl* module. *UpSlope* cycle (Left) and *DownSlope* (Right) ........................................ 137
4.39 Example of synthesising motion (path shown in green) on arbitrary uneven terrain provided by the user (shown in yellow). .............................................................. 138

4.40 Another example of synthesising motion (path shown in green) on arbitrary uneven terrain provided by the user (shown in yellow). .............................................................. 138

4.41 Quantitative evaluation of 'TerrainControl' examples. ....................................................................................................................... 139

4.42 Input clips for the SpeedTerrain Control module. ....................................................................................................................... 140

4.43 Example of synthesising motion with arbitrary speeds (path shown in green) over arbitrary uneven terrain provided by the user (shown in yellow). ....................................................................................................................... 141

4.44 Another example of synthesising motion with arbitrary speeds (path shown in green) over arbitrary uneven terrain provided by the user (shown in yellow). ....................................................................................................................... 141

4.45 Quantitative evaluation of 'SpeedTerrainControl' examples. ....................................................................................................................... 142

4.46 Original (Left) and cyclified (Right) 'Walk' motion. ....................................................................................................................... 143

4.47 Quantitative evaluation of original and cyclified 'Walk' cycles (1 cycle each). ....................................................................................................................... 144

4.48 Effect of concatenation on both original and cyclified 'Walk' cycles. ....................................................................................................................... 145

4.49 Original (Top) and cyclified (Bottom) 'Walk' motion on slope floor. ....................................................................................................................... 146

4.50 Quantitative evaluation of original and cyclified 'Walk' cycles on slope floor (1 cycle each). ....................................................................................................................... 146

4.51 Effect of concatenation on both original and cyclified 'Walk' cycles on slope floor. ....................................................................................................................... 147

4.52 Original (Left) and cyclified (Right) 'Kick' motion (Initial and final postures of each motion). ....................................................................................................................... 147

4.53 Quantitative evaluation of original and cyclified 'Kick' motions. ....................................................................................................................... 148

4.54 Effect of concatenation on both original and cyclified 'Kick' cycles. ....................................................................................................................... 148

4.55 Initial posture (middle), final Original posture (Left) and final cyclified posture (Right) 'Jump' motion. ....................................................................................................................... 149

4.56 Quantitative evaluation of original and cyclified 'Jump' motions. ....................................................................................................................... 149

4.57 Effect of concatenation on both original and cyclified 'Jump' cycles. ....................................................................................................................... 150

4.58 Compare original (middle), Mirrored (Left), and Synchronised Mirrored (Right) motion clips. It can be noticed that the mirrored motion (Left) starts with the right-foot while the original (middle) and Synchronised Mirrored (Right) motions start with the left-foot. ....................................................................................................................... 151

4.59 Blending original (left) and Synchronised mirror clip (right) produces a valid motion (middle) as the mirrored clip is in-phase with the original clip. ....................................................................................................................... 151
4.60 Quantitative evaluation of 'Synchronised Mirrored' clip. ............. 152
4.61 The synthesised root trajectory (middle) is suddenly flip to the opposite
direction after the difference in direction of input motions exceeds 180°.
Input motions are StraightWalk (right) and TurnBack clips (left). ... 153
4.62 The synthesised root trajectory (middle) has no sudden flipping any
more after using the improved blending technique. Input motions are
StraightWalk (right) and TurnBack clips (left). ......................... 153
4.63 Comparing the change of direction between initial and final frames, of
each sample motion clip, with and without the proposed improvement. 154
4.64 The improvement in blending results in noticeable reduction in errors as
shown by the quantitative measures. ................................. 155
4.65 Comparison of the storage requirements. ................................. 155
4.66 Example for integrating the developed modules for longer sequence of
real scene. .............................................................. 156
4.67 Example for integrating the developed modules for longer sequence of
real scene. .............................................................. 157

A.1 Wavelet reconstruction/synthesis. ................................. 171
A.2 Wavelet decomposition/analysis. ................................. 172
A.3 Frequency-contents analysis of different human motions. .... 174
A.4 Example of compression of 'Walk' animation. Original (right), com-
pressed with 69% (middle), and compressed with 87% (left) clips. ... 177
A.5 Quantitative Evaluation of compression of the 'Walk' example. .... 178
A.6 Example of compression of 'Run' animation. Original (right), com-
pressed with 69% (middle), and compressed with 77% (left) clips. ... 178
A.7 Quantitative Evaluation of compression of the 'Run' example. .... 179
A.8 Compression of BalletDance animation. Original (right) and compressed
with 79% (left) clips. .................................................. 179
A.9 Previous quantitative measures, applied on 'Walk' example, are shown
for compatibility with other modules. ................................. 180
A.10 Previous quantitative measures, applied on 'Run' example, are shown
for compatibility with other modules. ................................. 180

C.1 The Evaluation form used by the participants for qualitative evaluation. 189
C.2 The rating distribution for both input and output animation sequences. 190
Chapter 1

Introduction

There is an increasing demand on virtual reality applications especially with the development of computer hardware, software, modelling and animation techniques. Virtual worlds that simulate parts of our real world have a lot of useful applications such as scientific simulation, education, training, games and entertainment. To complete the picture, the virtual world needs to include the representation of human beings as a major and effective part of the real world. Virtual humans or virtual people are the computer models of the real humans in the virtual worlds. These models should be as realistic as possible in their shape, appearance, motion, behaviour and interaction with each other as well as with other objects of the virtual world.

Achieving realistic animation is not an easy task as it may appear in the first instance. It consists of four major stages:

1. Realistic modelling of the human body shape.
2. Realistic motion control which is concerned about how the model will move.
3. Realistic appearance of body, skin deformations, hair, and clothes animation.
4. Behaviour animation that is concerned about how the model will behave and react to the surrounding environment.

Each stage is sufficiently complex to be treated separately. Research in ‘Virtual Humans’ can be classified as shown in figure 1.1.
Chapter 1. Introduction

Research in "Virtual Human"

- Modeling
- Animation
  - Motion Control
  - Appearance & Deformation
  - Behaviour

Figure 1.1: Principle branches of 'Virtual Humans' research

The main concern of this research falls in the 'Motion Control' category, which aims to produce a realistic motion animation using the human skeleton representation. There are many methods for generating animation. They can be categorised into, motion synthesis methods, captured motion data methods, or hybrids of both of them. Motion synthesis methods include defining the joint angles, interpolation between postures, using kinematics or dynamics techniques and incorporating a thorough knowledge about the motion to generate its animation. In the captured motion animation, the motion can be captured directly from video or from capturing devices with sensors on the subject. Choosing the animation method depends on the application that will use it as each method has its own advantages and disadvantages. For example, in the motion synthesis methods, most of the joints can be controlled by a small number of parameters like position and orientation of the end effectors. This small number of controlling parameters reduces the required amount of knowledge that the animator should know about the motion. On the other hand, maintaining the human motion details, which identify human motion from other motions, is not easy.

The main advantage of captured motion animation is its realism as it corresponds to real human motion. Its implementation is also often faster than the manual animation synthesis methods. The main disadvantages of captured motion animation are its limitation to be adapted for motion changes as well as the changes in properties of the used virtual human model. These problems are usually known as the motion editing and motion retargeting problems respectively.
1.1 Motivation

Our research is motivated by the fact that although some techniques can produce realistic animation for certain requirements, editing or modifying these existing animations to meet a slightly different requirements is still a challenging task. Various synthesis and editing techniques have been developed in previous research. However, the synthesis process is usually controlled by low-level parameters. The lack of high-level parametric control has been one of the obstacles for new users to produce animation.

This research aims to provide an intuitive high-level parametric control over the synthesis of realistic human animation. This parametric approach is to satisfy the user-defined high-level motion parameters while maintaining the quality and realism of the generated animation and preserving the motion constraints and properties. The realism of the generated motion comes from the fact that the motion captured data produces the movement of a real person with its intrinsic dynamics and unique characteristics (provided that the motion capture devices have sufficient accuracy).

The parametric approach allows both novice and expert users/animators to define their desired motion using intuitive parameters. It is also important to provide this advantage with a computationally efficient system. Hence, a compromise between intuitive control, realistic animation, and efficient computation cost is another motivation. It is worth noting that different applications have different requirements (including the level of realism of the generated animation). This varies from simple tutorial or presentation, games, up to 3D broadcasting and film production, with the latest often having the highest requirement of realism.

The presented parametric approach is useful in many applications such as games, virtual education and training, simulation, navigation, and many others. It can also accelerate the learning curve and productivity of novice and existing animators.

In the rest of this chapter, the principal contributions are summarised in section 1.2 with the publications produced within the thesis work. Then, the thesis structure and overview is given in section 1.3.
Chapter 1. Introduction

1.2 Publications and Summary of Contributions

The principal contribution of this thesis is the proposed high-level parametric approach for synthesis of realistic human animation. The approach allows an intuitive high-level parametric definition of the desired motions while generating realistic animation based on a small motion capture database of short basic clips. The framework is developed and utilised to realise interesting and challenging tasks which demonstrate its usability and usefulness.

The principal novel contributions can be summarised as follows:


- High-level parametric approach for synthesis of long animation sequences according to arbitrarily time-varying parameters provided by the user (Chapter 4). This extended framework has been utilised to achieve interesting animation tasks such as controlling motion synthesis along arbitrary path and over uneven terrain. The proposed framework has less data requirements and efficient computation cost with linear complexity with the length of the desired animation sequence.

- The approach is extended to allow multi-parameters synthesis with enhanced parameterisation technique that reduces the number of parameterisation functions required. It also works on the given irregular grid of input motions, on the parameters space, which avoids the pre-processing of modifying inputs to form a regular grid on the parameters space as discussed in section 4.6.

Other contributions have been introduced at different stages of the synthesis process in order to achieve the parametric synthesis of long animation sequences. These novel contributions are:

- Novel motion cyclification techniques [5] to ensure smooth concatenation of motion cycles to construct a long sequence (section 4.3.1).
• A novel animation processing technique for enriching the basic animation database with synchronised mirrored clips. The resulting clips are automatically annotated for processing within the parametric approach (section 4.3.4).

• Improved Root – Trajectory Blending technique that overcomes the identified limitations of existing blending such as the reduced translation magnitude and the sudden flip of the blended root trajectory when the difference in direction between input motions exceeds 180° (section 4.3.3).

In addition to providing intuitive and high-level parametric animation synthesis, the proposed approach employs simple, fast and efficient synchronised blending algorithm. This provides a computationally efficient solution with the realism of the motion capture animation.

The following papers have been produced throughout the research period:


1.3 Thesis Structure and Overview

The thesis consists of 5 chapters, including this introductory chapter. The novel contributions of our research are presented in chapter 3, chapter 4, and appendix A. Chapter 2 presents a literature review of the research in human animation while chapter 5 summarises and concludes the thesis along with suggestions for future work. Following is an overview of the thesis and its various chapters.

The next chapter presents a literature review on human motion animation. It starts by giving a general overview of computer graphics, animation, virtual world, and virtual people. Then, presents a survey on research in computer animation of human motion. The survey summarises the most common animation techniques and focus on previous work that is related to the presented work.

The initial parametric framework and methodology are described in chapter 3. This initial framework focuses on the parametric synthesis of short animation clips [4] with a specific value of motion parameter given by the user. The core idea of solving the inverse-interpolation problem is presented and the results are shown and evaluated towards the end of the chapter with the introduced and adopted quantitative measures. Samples of candidate applications for this initial framework are presented at the end of the chapter.

Chapter 4 shows how the initial framework has been extended, with novel contributions, to parametrically synthesise longer animation sequences controlled by high-level parameters. This includes pre-processing for extending the input motions and ensuring a smooth concatenations [5]. Practical animation tasks, such as generating motion along arbitrary path and over arbitrary uneven terrain, have been achieved. The framework has been extended for multi-parameters synthesis where more than one motion parameter can be controlled simultaneously. Results are presented at the end of the chapter with both qualitative and quantitative evaluation, using the introduced and adopted quantitative measures.

In appendix A, an adaptive compression technique is described [1]. It shows how wavelet analysis is employed, based on the analysis of human motion, to achieve high
compression ratios without destroying the animation realism.

The presented work is then summarised and concluded in chapter 5 with ideas and suggestions for future work. Chapter 5 also includes a discussion of the advantages, limitations of the proposed approach as well as the assumptions made, and their justification.

A summary of comparisons with some relevant approaches is depicted in appendix D while the details and the discussion of our qualitative evaluation experiments are presented in appendix C.

Finally, a CD-ROM is enclosed with this thesis which contains video clips of both the resulting animation sequences and the input animation clips.
Chapter 2

Human Motion Animation: A Literature Review

A literature review on human motion animation is presented in this section. The subject of human animation represents a huge research area and this review is limited to the work most relevant to this thesis. The chapter starts with an overview of computer graphics and animation in section 2.1, then virtual humans and their applications are covered in section 2.2. The review on computer animation of human motion and the various related techniques used for animation are then presented in the rest of the chapter. Finally the chapter concludes with a summary of the literature reviewed and identifies the open areas of research that are addressed in this thesis.

2.1 Computer Graphics, Animation, and Simulation

Many human activities can be described as time-dependent processes, which can be animated using computer graphics and animation techniques. Depending on the activity and what we know about it, the animation can be based on either observation data (either for the sake of animation only or to analyse and understand this activity) or the mathematical model of the activity which employs physical laws (if the model and its physical properties are available).
Although the terms of *computer animation* and *computer simulation* are closely related and often used interchangeably, there is some distinction between them. *Computer animation* is the process of generating, and displaying, successive postures using the computer. Creating the postures is a creative and interactive process where the animator uses his imagination and skills with the aid of the computer graphics and animation system capabilities. Advances in computer graphics and animation techniques have improved the capabilities of these animation systems to automate many tasks and also to incorporate some knowledge about the different motions. *Computer simulation* can be considered as a special case of computer animation which mostly represents or animates a real world phenomenon based on its mathematical, physical and dynamical properties. Simulation helps in studying the behaviour of real phenomenon as it can be run in real, compressed or expanded time which may be expensive or impossible to do otherwise [46].

In this thesis, our focus is on the computer animation of human motion. The importance of animating human motion is explained in the next section.

### 2.2 Virtual Worlds and Virtual People

Computer animation technology provides us with methods and tools for modelling many objects from our real world. With the development of computer modelling techniques, virtual worlds become easier to generate and much more realistic. The virtual world is a computer representation of the real world (or part of it) in which the user can create, control, and interact with modelled objects. This ranges from 2D to 3D virtual worlds and extends to real-time and shared environments [25]. As the real humans play a major role in the real world, existence of their representation in the virtual world is essential to complete the picture. Virtual humans or virtual people are the computer representation of real human beings in the virtual worlds. There are many different applications in different fields that need virtual humans [71] such as:

- Engineering design: Simulation-based design and virtual prototyping of products (such as ergonomic design).
2.3 Biomechanics of Human Motion

• Bio-medical simulation: Simulating human biological systems helps in better understanding of human behaviour and reactions. Surgical training is one of the important applications in this area.

• Virtual conferencing: Virtual representation of the conference participants reduces the transmission bandwidth requirements and leads to more efficient teleconferencing.

• Education and training: Virtual humans can be used as an instructor, presenter or interactive assistant.

• Military: Simulation of the battlefield and studying the mutual effects of the environment and the individual participants.

• Games: Increasing realism and interactivity.

• Digital broadcasting and film production: Including visual effects and mixing real and virtual objects.

The rest of this chapter reviews the most relevant work to this thesis in the area of computer animation of human motion.

2.3 Biomechanics of Human Motion

Animation of human motion is not an easy task because the human body is one of the most complex mechanisms. Before going through the existing techniques of animation of human motion, we provide a brief review of the human motion itself. Analysis of human movement attracted many researchers from different fields and for different applications. The most common and related field, to animation, is the field of biomechanics.

Biomechanics utilises the engineering mechanics principles for the study and analysis of the human movements. It is an interdisciplinary field combining engineering mechanics, biology and medicine. This section is not intended to give an exhaustive
biomechanical overview of the human motion. For more information, other references are recommended [72, 103].

Due to the complexity of the human movements, and in order to improve the animation realism, animation algorithms utilise the biomechanics knowledge of these movements. There are many different types of human movements that may have different characteristics. In this thesis we focus on the cyclic movements which represent a major sector of our activities. Common examples of these cyclic movements include walking and running.

Walking and running are among the most common human activities in our daily life. Hence, such common activities need to be animated in order to populate the virtual world with virtual people. Being so common means that we are so familiar with these types of movements, and we can easily detect unnatural motions. This increases the demand for more realistic animation which require more knowledge about the motion in order to simulate it.

Walking and running are defined as means of locomotion that use the two legs, in turns, to provide support and forward motion [100]. In a normal walk, at least one foot is in contact with the floor at any time. However, in running, there is a period where both feet are on the air.

The walking cycle (or gait cycle), is the period between two successive initial foot-floor contact of the same foot [100]. During the gait cycle, different phases and sub-phases can be recognised. The main phases of the gait are the stance phase and the swing phase.

The stance phase (also called the support phase) occupies the time interval when the foot, of the supporting leg, is in contact with the floor. This extends from the time of the first contact, usually by the heel, to the last contact, usually by the toe. During this period, the contact with the floor is achieved by different parts of the foot. Initially, the heel touches the floor, then the foot becomes flat over the floor, and finally only the toe is in contact with the floor. The overlap between the stance phase of both feet results in a double support period where both feet are in contact with the floor (although the contact is achieved by different parts of each foot). The double support period is one
of the differences between walking and running motions. In running motion, there is no double support period. Instead there is a period where both feet are simultaneously not in contact with the floor.

The *swing* phase starts when the toe, of the swinging leg, leaves the floor and ends when the heel of the same foot touches the floor again. The swing phase is usually shorter than the stance phase. In the normal walk, the swing phase usually occupies about 40% of the cycle time while the stance phase occupies about 60% of the cycle time.

Each of the main phases of the gait contains a few important events of the motion. The stance phase has a sequence of events like *heel-contact*, *foot-flat*, *mid-stance*, *heel-off*, and *toe-off*. The swing phase has the sequence of *toe-off*, *mid-swing*, and *heel-contact*. More details of these events and the gait phases can be found in [100].

Biomechanics provides different aspects of knowledge about the human movement. This includes kinematic and dynamic knowledge of the analysed motion. The type and level of biomechanical knowledge that is employed in animation differs from one animation technique to another. This also depends on the implementation of the animation technique and the level of realism or quality of the required animation.

Kinematics animation techniques [12, 15, 16, 14, 65, 21] utilise the kinematics description of the human movement without taking into account the forces that produced this motion. On the other hand, dynamic simulation techniques [11, 102, 47, 104, 28, 98, 30, 31, 26] incorporate the dynamics knowledge of the movement and utilise the known mechanical laws to produce and govern the simulation.

Gait analysis provides knowledge about the different important events. This knowledge is important for animation as most of these events represent motion constraints that should be preserved in animation. Preserving motion constraints helps to improve the animation realism.

More details on the different animation techniques are the focus of the next section.
2.4 Computer Animation of Human Motion

The virtual human is one of the most complex models in computer animation and virtual reality. Animation of such complex models is a challenging process. To simplify this process, research is divided into two main branches [70] (See figure 1.1 in chapter 1); motion control, and appearance and deformations. In the first branch, motion control, motion animation is generated using a skeleton which is a simplified representation of the virtual human kinematic structure. The skeleton representation consists of a set of hierarchical rigid links connected by joints. This simplified model allows the animator to generate and modify the animation more efficiently. Once the required motion animation is generated for the skeleton, the surface model can be animated and rendered to synthesise a realistic appearance of the virtual human. This is the second branch of the research which studies the hair, skin and clothes deformation modelling. In order to achieve believable and realistic virtual human animation, both branches need to cooperate and to be applied together. Their separation in research is mainly due to the complexity of their individual computation and modelling.

There are various methods for generating and editing animation. Figure 2.1 depicts a summary of the common human animation methods reviewed in this chapter. The most common methods deal with the virtual human as an articulated figure and use its skeleton for generating the motion. In the following subsections, an overview of the most common methods is presented.

2.4.1 Traditional Methods

2.4.1.1 Manual and Key-Frames Animation

Historically, animation has been a manual process in which the animator should produce the poses of the avatar (or avatars) for the whole animation sequence [99]. This is a very time consuming task especially for a complex figure and becomes more difficult with the existence of many figures in the scene. As it is difficult to create, it is difficult to modify because any single change may require all the poses in a sequence to be altered which means almost recreation.
2.4. Computer Animation of Human Motion

Common Methods of Human Animation

- Traditional Methods
  - Manual and Key-frames
  - Kinematics
- Dynamics Animation
- Motion Capture Animation
- Motion Editing Methods
  - Signal Processing
  - Constraint-based Editing and Optimisation
  - Statistical Analysis
  - Transition Graphs (Sampling Motion Database)
  - Interpolation and Blending

Figure 2.1: Common methods in Human Animation research
Key-framing [24] is an enhancement to the manual method to make life easier for the animator and utilise their skills. Traditionally, a principal animator provides a few essential frames (known as the key-frames) and junior animators produce the rest of frames between key-frames. This is enhanced by utilising computer to interpolate key-frames and generate the intermediate frames for the whole animation period. This is the traditional and popular way of generating animation. To modify the animation, the animator should edit the key-frames only which is more efficient than manually animating every frame.

In both of the above methods, the quality of the resulting animation depends heavily on the experience and knowledge of the animator about the motion that should be animated.

2.4.1.2 Kinematics Animation

In general, Kinematics is the science of studying the motion of a body (or system of bodies) without considering its mass and the forces acting on it [99]. From the point of view of human motion, kinematics is the study of the positions, angles, speeds, and accelerations (i.e. all geometric and time-related properties) of the human body joints and/or segments (assuming they are rigid bodies) during their motion. Kinematic systems are mainly based on studies of human motion in biomechanics and biology.

In this type of animation, the animator can define some of the positions and/or angles of the skeleton joints and segments and the rest of these positions/angles could be computed by the kinematics algorithms. Depending on what variables are known and what variables are required to be calculated to achieve the desired animation, there are two types of kinematic systems in human animation: Forward (Direct) kinematics and Inverse kinematics. In forward kinematics, the animator specifies the angles of the skeleton joints. As the skeleton representation of the virtual human is a hierarchical model, positions of the joints/segments can be calculated by accumulating all the transformation resulting from the chain of joint angles acting on a body part.

However, in most practical situations (like reaching a certain position), the animator is often times more concerned about positioning the end-effectors (i.e. hands and
2.4. Computer Animation of Human Motion

feet) than specifying the joint angles explicitly. In *inverse kinematics* (IK), knowing
the positions of end-effectors, the joint angles are calculated to produce the required
transformations to achieve these positions. It is sometimes referred to as the 'goal-
directed' method [99].

IK has been utilised in controlling the animation of various human movements such
as sitting down, grasping and locomotion. Badler et. al. [12] introduced the solution
of multiple constraints using an iterative algorithm to make a synthetic actor sit down
on a chair. In [15], Boulic et. al. proposed a global human walking model based on
a kinematic approach that tried to preserve the dynamic characteristics. Later on,
they proposed a method for combining forward and inverse kinematics to control the
motion and its modification [16, 14]. The automatic grasping algorithm developed
by Mas-Sanso et. al. [65] also uses forward and inverse kinematics to control the
hand through the grasping action of synthetic actors. Zaho and Badler [106] used non-
linear programming techniques for solving the inverse kinematics problem in the *Jack* \(^1\) system. Tolani and Badler [93] developed real-time inverse kinematics for human arm
animation. Other real-time inverse kinematics techniques for the anthropomorphic
limbs are also proposed by Tolani et. al. [94].

As inverse kinematics is a well known problem in animation as well as robotics, much
research has been done to enhance the solution methods. The most common methods
of solving IK are the Jacobian and non-linear programming techniques.

The IK problem becomes more difficult when the number of joints increases as it be-
comes an under-constraint problem for which there are many solutions. Therefore,
some other constraints should be added to the system to find a suitable solution such
as joint limits and external physical constraints.

Unlike key-frame animation, the quality of the resulting motion from kinematics ani-
mation is based on the IK model and solver instead of the animator's skills.

\(^1\) *Jack* is a human figure animation system developed at the University of Pennsylvania
2.4.2 Dynamics Animation

In kinematics animation, motion is animated without defining the forces and torques that cause that motion. This can result in physically unrealistic animation due to unnatural postures. For greater realism, the physical laws governing the human motion can be incorporated within the model. This is what the dynamics animation does.

Like kinematics animation, dynamics has two main types: Forward (Direct) dynamics and Inverse dynamics. Forward dynamics requires the forces and torques to be known. Knowing the initial conditions and the constraints, the system will apply these forces and torques to the masses representing the objects or human parts to simulate their motion. Inverse dynamics is used when the motion of the end-effectors is known but forces and torques that generate that motion need to be calculated.

Dynamic simulation is normally achieved in five main stages [11]. The first stage is modelling the objects to be simulated and their physical properties. In the second stage, the dynamic equations of motion are formulated. The third stage solves these dynamic equations to find the accelerations from the given forces and torques. Knowing the accelerations, speeds and locations can be obtained by integrating these accelerations. Finally, the animation of the body parts is performed using the calculated locations (which serve as a kinematics description). In the first stage of object modelling, the human body parts (as well as the environment objects) are usually represented as rigid bodies. The motion is simulated by applying forces and torques on those rigid bodies governed by the physics and classical mechanical laws. This is the reason for this type of animation to frequently referred to as a dynamic simulation instead of animation.

There are many formulations for the dynamic equations of motion and there has been much work to find computationally efficient solution methods. Pina et. al. [74] have classified the solution techniques to two main categories: Solving by integration [102, 47] and solving by non-linear optimisation [104, 28]. For real-time response, different approaches have been developed to simplify the dynamic models. Some approaches deal with some joints or degrees of freedom (DOF) dynamically and with others kinematically. Other approaches incorporate parallelisation techniques to speed up the solution. More details are discussed in [74].
2.4. Computer Animation of Human Motion

Most of the work in this category is based around control and optimisation techniques. For example, physics-based controllers are designed, composed and integrated in order to provide high-level control on simulated characters [98, 47, 30]. Integrated controllers allow the character to perform motor tasks while being able to autonomously react to different situations within its environment [31]. In other work, optimal control mechanisms, mixed with inverse kinematics, are used to produce periodic gaits [26]. As a physics-based technique, the optimisation formula is still to be determined for each motion and more importantly, how it is controlled by the user-defined parameters.

On a recent trial for bringing interactivity and artistic touch to physics-based animation, a hybrid of control techniques is used to provide an interactive control for physically-based animation [58]. However, the system needs a lot of effort in designing and learning the appropriate interfaces for physical animation which results in a steep learning curve. Moreover, more investigations and enhancements are needed for the technique to be applied to 3D human figures.

Dynamic simulation is not only used to synthesis motion animation from scratch, but can also be used for constraint validation. In such systems [53] animation is generated kinematically but validated using inverse dynamics (e.g. to maintain the balance and comfort situations of body). Adding such dynamic constraints to kinematics animation provides a compromise between realism and ease of control for animation. It provides the animator with an easier interface for design (not using forces and torques) but with more realistic results than kinematics alone. Unfortunately, this method is not applicable to all dynamic constraints. One of the disadvantages is the added computation cost of using two stages (kinematics and dynamics) [70].

Compared with pure kinematics animation, dynamic animation (or simulation) results in subjectively realistic motion with greater ability to interact with other objects in the environment (e.g. collision detection and avoidance). However, dynamic techniques have common disadvantages that include the high computational cost needed for solving the motion equations and the difficulties of controlling the animation as the animator usually does not think in terms of forces and torques [74, 11].
2.4.3 Motion Capture Animation

In motion capture animation, motion is captured from a real person performing the required motion. Recorded data can be in the form of positional and/or orientation data. Motion data is then applied to a computer model, usually a skeleton hierarchy, representing the real performer.

Motion capture is based on the use of different sensors technologies to track and record 3D positions of the moving human joints. The most popular capturing systems are based on mechanical, magnetic, or optical sensors. Mechanical systems, the oldest capturing systems, are used to capture motion of different body parts. These systems introduce great restrictions and inconvenience to the performer's motion due to the need of attaching mechanical armatures and encoders to the body. In magnetic systems, magnetic sensors, that can detect positions and orientations, are attached to the performer's body. However, many wires are required to collect data from the sensors which introduces inconvenience for the performer. Magnetic systems are also affected by metals in the capturing volume.

The most commonly used capturing technologies nowadays are the optical systems. Passive optical systems require reflective markers (balls) to be attached to the performer. Multiple cameras, equipped with light sources, are used to detect the 3D position of those reflective markers. This type of capturing systems provides the most freedom of movement to the performer because no wiring is required. The typical limitation of the passive optical system is the tracking of occluded and hidden markers.

Active optical systems overcomes the tracking problem by using active markers. The markers attached to the performer are transmitting the light in a synchronised order and multiple detectors are fixed around the capturing volume. However, the wiring problem appears again.

Optical sensors require a clear line of sight between transmitters and detectors and can be affected by reflective surfaces. A common issue among almost all technologies is the limited capture volume.

With the rapid and continuous improvement of motion capture techniques and equip-
2.5. Motion Editing

ment, more accurate motion captured data can be acquired. Additionally, despite its limitations, motion capture becomes popular and widely used for generating realistic animation. It should be noticed that there are some situations where motion capture has no reasonable alternative such as capturing motion style and personality of the performer. It is also the main technique for producing realistic animation driven by a real performer in real time. Advanced and expensive motion capture systems are used in the film industry for mixing real and virtual scenes and actors as in the movie “Titanic”. It can also be used in generating animated films and games as well as special effects in video-clip songs.

When played back on the same character, the captured motion looks realistic. However, problems arise when any variation of the motion (or the character) is needed. As the captured motion is mainly a set of data with no explicit model, uncarefully modifying any part of this data is likely to damage part or the entire motion. This damage can range from loosing some motion properties and dynamics to completely destroying the core motion itself. The straight forward solution is to repeat the capturing procedure to acquire the specific desired motion. This loses the benefit of the existing data and the cost of motion capture is increased again and again until the appropriate motion is recorded. The alternative is to find a way to modify the captured animation. Modifying the existing data can be motivated by the interest of new motion for the same character (which is referred to as the motion editing, section 2.5), or by the interest of adapting the same motion to other characters with different dimensions and geometry (which is known as motion retargeting, section 2.6). The motion modification has attracted many research as discussed in the following sections.

2.5 Motion Editing

There is an increasing interest in developing tools for modifying motion animation, regardless of how this animation has been created, to benefit from any existing animation clips. A particular area of interest for modification is motion captured data. Motion capture is an expensive and time consuming process although it has been widely used in film and game production as it provides a source of highly realistic detailed move-
ments. Motion editing has became an area of interest in human animation field and is a challenging task.

In this section, an overview is given of the principal research work in motion editing. Motion editing approaches can be grouped according to their methodology and aims. In one group, established time and frequency domain signal processing techniques have been applied to the motion signals in order to modify and edit the motion [8, 22, 34, 96, 105]. Multi-resolution analysis has also been used for the same purpose [48, 60, 91, 4]. Another group has employed optimisation techniques to edit motion while preserving constraints which have been identified in advance [104, 28, 35].

In another group, the captured data is used to build a statistical model which is then used in motion synthesis [75, 76, 68, 61, 59, 17]. Recent research has investigated building transition graphs from database of movement sequences. New motions are synthesised by searching the built graphs for the most relevant frames according to some defined criteria [9, 55].

Interpolation and blending of motion data have been utilised to produce variations of existing motions [101, 77, 79, 54]. Our work is related to this research in terms of the core methodology of using interpolation and blending. However, this methodology is extended and employed in many different ways and hence, the presented work is jointly related to other categories as well in terms of functionality and achievements as discussed in the corresponding chapters in the thesis.

The following sections introduce the key work in motion editing and modification classified according to the methodology of modifying the data.

2.5.1 Signal Processing

As the captured motion data is a set of time-varying signals, many techniques from signal processing have been utilised and applied to motion editing and modification tasks. Witkin and Popovic [105] introduced motion warping (stretching or shrinking a signal in time) for editing captured motion. They reported that the key advantage is the ability to be integrated with existing key-framing tools. Their motion warping has
2.5. Motion Editing

some inherited limitations from the traditional key-framing. For example, satisfying the geometric constraints must be achieved by additional effort. Moreover, the technique does not incorporate any knowledge about the motion. So, realistic and physically correct results are not guaranteed.

In [22], Bruderlin and Williams presented a simple library of signal processing techniques for motion editing. Pyramid filters were used for multi-resolution motion filtering, time warping was used as a method for synchronising motions and wave-shaping was presented as a simple and effective method of producing some effects on different degrees of freedom. Motion displacement mapping was introduced as a tool for modifying basic motions through a standard key-framing interface. However, interpolation is still required between the modified key-frames to update the in-between frames. Recently, Gleicher [34] has extended the displacement mapping technique to edit the motion path of an existing motion clip (more details in section 4.4). Also, within the context of dealing with the motion signals, Sun et. al. [90] combined the use of the sagittal plan angles with a walk model to synthesis walking on uneven terrain and arbitrary path.

Multi-resolution analysis has been also applied to motion data [48, 60] in order to edit the different frequency bands of the motion signals separately. Sun [91] utilised wavelet analysis and its multi-resolution properties to model bipedal locomotion. The decomposed motion curves could be edited or blended at any resolution level independently. We extended this work in [4] with the long-term aim of multi-resolution parametric synthesis for animation and emotion (more details in chapter 3 and A).

A common objective on this category is modifying individual motion clips. Our work is different in that instead of editing one motion clip, we synthesise the desired motion based on a set of motion clips. This provides additional information for correctly animating some segments (such as the feet on curved pathes or slopes).

2.5.2 Constraint-based editing and Optimisation

A principle problem of editing existing data is the risk of violating the motion constraints such as joint limits, self intersection or intersection with scene objects. Some
techniques have been developed for editing motion while preserving its constraints based on optimising pre-defined cost functions. Witkin et al [104] introduced the 'spacetime constraints' technique where motion synthesis is considered as a constrained optimisation problem. The optimisation problem is solved simultaneously for the whole animation sequence instead of individual frames. This results in a high computational complexity and reduction of the interactivity with the environment during the animation.

To improve interactivity, Cohen [28] proposed an improved spacetime constraints method called 'spacetime windows' in which solutions are found for sub-periods of the animation instead of the whole animation sequence. In order to improve the performance, Gleicher [35] suggests ignoring some constraints in order to reduce the computational complexity, although this will impact quality of the resulting animation.

Liu et al. [63] introduced a method for generating relatively complex realistic animation based on simple input animation. The input animation is analysed and the required motion is generating through enforcing linear and angular momentum constraints. As reported, the method is more suited for highly dynamic motions and low-energy motions such as walking and reaching are not achievable.

Spacetime constraints technique has also been utilised in other related applications such as creating motion transitions [78] and retargetting existing motions to different characters [36](section 2.6).

Common drawbacks of this approach include the computational complexity and the need for explicitly specify constraints a priori. Spacetime analysis requires the whole animation sequence, or parts of it in the case of spacetime window, to be available beforehand. This prohibits the use of spacetime constraints technique in online or real-time applications. The approach is suitable for motion editing in offline applications and as a post-processing tool for satisfying user-specified motion constraints.

2.5.3 Statistical Analysis

Statistical approaches are based on learning a statistical model from existing motion data. This model is then used to synthesise various motion sequences with the same
2.5. Motion Editing

statistical characteristics. The type and size of the motion data used in the learning phase depends on the intended application of the built model. In some cases, the data is limited to a specific context such as interaction between people [50]. In other cases, the data is not classified and can be a mix of different categories of motions as in [67]. Most of the existing research used the 3D animation data. However, in some cases, 2D animation data are used for simplicity [75].

Brand et. al. [18] employed hidden markov models (HMM) to model both the basic motions and their styles from existing data. In order to separate basic motion from style, the data is abstracted which can affect the realism of the output as it may discard details of the input motions. Lee et. al. [59] preprocess extended unlabelled sequences of motion data, related to a specific application, for flexible control of avatar. Their statistical model consists of two layers (clustered data, Markov process) to benefit from two different forms of data representation. Molina-Tanco [68, 67] also uses a two layers framework for the realistic synthesis of in-between motions from motion captured database. The first layer clusters the data and simplifies the resulting graph while the second layer samples the original data to obtain the relevant frames. This approach preserves motion details because it uses the original motion data, without abstraction, for synthesis.

Statistical approaches require a relatively large amount of data in order to learn the statistical properties of the motions and for building the required models. This leads to relatively high computation cost, due to the large amount of data to be processed or searched. This cost increases with the complexity of the search criteria and cost functions prohibiting real-time interactive application. In contrast, our work requires much less input data and employs simple but effective parametric blending technique which is computationally efficient.

2.5.4 Transition Graphs

In this approach, motion is synthesised by sampling animation database, which is driven by navigating a pre-constructed transition graph, according to a pre-defined search criteria. Arikan et. al. [9] constructed transition graphs from an annotated motion
capture database and considered the synthesis process as a randomised search in a
hierarchy of graphs. They have reported that the system can not handle motions on
non-uniform surfaces and to guarantee physically correct motions, more post-processing
is still to be incorporated. Also, the manual annotation of input clips is still a time
consuming task. The automatic annotation of given data is addressed in our work [5]
(section 3.4.3).

Recently, Arikan et. al. [10] introduced an improved search method based on dynamic
programming. To enhance the quality and efficiency of annotation of motion clips, a
tool based on iterative support vector machine is used. Similarly, Lucas et. al. [55]
introduced a path editing method based on motion graphs, a directed graph that or­
ganises and links the original motion clips and the possible transitions between them.
The approach is extended in [37] for assembling animation clips that are extracted from
a collection of motion data based on finding common postures. The preprocessing, in­
cluding finding common postures and modifying clips around these common postures,
is performed at authoring time to improve the synthesis performance.

Instead of building graphs based on frames from motion clips, another approach is
building motion trees from motion models and the operations defined on these models
[43]. However, details of how the motion models will satisfy the desired motion is
not discussed. In a similar approach, instead of searching graphs, Lamouret et. al.
[57] searches the animation database for the best-fit clips. The selected best-fit clips
are adapted and smoothly joined together to form the desired motion. The work is
presented on a simple figure (Luxo: A table lamp figure created by Pixor). More
advanced techniques are required to adapt motions of complex articulated structures
such as human figures.

Graph, tree, and database search techniques share some drawbacks. First, determin­
ing the search criteria or the cost functions can be difficult and critically affect the
results (as mentioned in [55]). Although snapping short segments from different mo­
tions gives more flexibility, it introduces difficulties in smoothly and seamlessly joining
those segments together. Most of the approaches are based on the similarity of static
poses and rarely maintain other properties (e.g. velocity). To overcome this limitation,
2.5. Motion Editing

The cost functions should include some terms representing these properties as discussed in [67]. Also, depending on the search algorithm, the performance can be non-linearly proportional to the length of the required motion (examples of various performances are presented in [55]). These drawbacks are already absent in our approach as we do not incorporate search operations.

Another limitation is the integrity of the original dataset. For example, as reported in [9], the system can not generate turning left motion unless there is some left turns in the original dataset. This issue is tackled in our work as depicted in sections 4.3.4 and 4.4 where we were able to generate left and right turning given either left or right turn only.

2.5.5 Interpolation and Blending

Interpolation is one of the earliest techniques used in computer animation. In its basic form, the tedious and time consuming process of creating the intermediate frames from key-frames is automated by employing interpolation. This technique is still effectively in use in most of today’s animation productions.

Some of the earliest work in this field is introduced by Ko and Badler [52]. Their generalisation method uses rotoscoped data to generate anthropometry of walking with arbitrary step length. They used an interpolation approach and assumed a linear relationship for different step lengths, with the blending factor, which is a simplification of the real case as discussed in chapter 3. Wiley and Hahn [101] applied the linear interpolation technique on pre-stored motion data to generate new motions. For faster synthesis, the pre-stored data clips are resampled to a uniform time scale. However, this does not guarantee the synchronisation of key-events in the motions and consequently does not guarantee a realistic and physically correct motion. Unuma et. al. [96, 97] used interpolation of the Fourier coefficients (of joints trajectories) of periodic motions to generate motions with emotion, motion transitions, and exaggerations of existing motions. However, they reported that the transitions are not fully invertable. For

---

2The rotoscope is a device that project a pre-recorded footage onto a drawing board where the animator can trace the projected images to produce animation.
example, the transition from walking to running could be achieved while the transition from running to walking does not look natural. Also, there is no guarantee that the resulting motion will be realistic or preserve motion constraints. In [88, 89], editing and manipulation is performed using a set of primitive motion operators (including blending, warping and cyclification) which were applied on non-uniform B-splines representation of the motion curves. It was reported in [88, 89] that using B-spline fitting may lead to violation of joint limits, and kinematics constraints.

The use of ‘Radial-Basis functions’ is introduced by Rose et. al. [77, 79, 85] accompanied by motion correction using inverse kinematics. However, the high-level motion parameters and the inverse interpolation problem are not discussed thoroughly. In related work, Guo et al. [45, 44] introduced a high-level human locomotion model based on parametric frame space interpolation. As their work was based on keyframing animation (not motion capture), locomotion cycles are assumed to be perfectly periodic, that is cycle boundaries are identical. This rarely exists in real human motion. Matching cycle boundaries for smooth repetition or concatenation is one of the problems facing the generation of long sequences as discussed in [5], section 2.7.2 and section 4.3.1. Golam et. al. [38] extended the application of frame space interpolation to cyclic as well as acyclic motions by automatically establishing correspondences and imposing constraints on transition curves. In their analysis, the upper and lower body halves are decoupled and processed independently which introduces a phase difference that requires correction. Moreover, satisfying user-defined high-level parameters (such as specified speed or slope) was not discussed.

In [40], the notion of motion models is introduced. Motion models encapsulate pre-compiled knowledge of basic motions and enable techniques to produce variations on these motions according to given parameters. The motion models notion is extended in [41, 42] for real-time animation using clip operators and dynamic motion trees. This is analogous to motion modifiers in the work presented in chapter 3. However, it is not clear how the motion model will satisfy the required parameters in the generated motion. Also, pre-processing of motion clips for generation of long sequences are not discussed (see sections 2.7 and 4.3.1 for more details). One of the drawbacks is the required effort needed to build new motion models. In a sense, our work provides a
2.5. Motion Editing

level of automation at many stages of this process.

In a relatively recent research, Park et. al. [73] presented a locomotion blending based on the radial-basis function interpolation introduced in [77]. They aimed to control motion style, speed and path simultaneously. It is worth noting that the term *style* in their work refers to the motion type such as 'walk' and 'run'. We share the assumption of constant speed during the motion cycle. However, for curved paths, we deal with the varying individual motion direction at each frame while they assume a constant turning angle at each frame which is not the common case in real human movement. Their assumption is related to the approximation of the motion path by fitting a circular arc to the root trajectory. Following an uneven terrain is achieved through a retargetting process [81] rather than blending. In terms of preprocessing the input clips, motions are manually annotated to label the key-event times. There is no discussion about generating long animation sequences from short animation clips and how the cyclification and root blending problems can be solved. In addition to our proposed automatic gait analysis (section 3.4.3), the above issues are identified and tackled in our work as discussed in sections 4.3.1 and 4.3.3.

Kovar et. al. [54] is considered the most recent work related to our research, which appears while writing up this thesis. Their introduced registration curves are data structures that encapsulate input motions and their constraints. Registration curves are constructed automatically based on detecting common postures using a distance function introduced in [55]. However, the intuitive parametric control of animation synthesis has not been discussed. More details of the commonly identified problems and comparison of the proposed solutions are discussed in the relevant sections (sections 2.7.1, 3.4.3 and 4.3.3).

Many of the above techniques share a common objective, which is to edit some properties of individual clips (or group of clips) in order to generate some variations of the original motion. In chapter 3, we present a novel approach for intuitive parametric synthesis of animation clips from motion data. Our work is different to previous approaches as we are concerned with parametric generation of continuous animation sequences with limited data requirements and efficient computation as presented in
2.6 Motion Retargetting

Although most of the interest is focused on generating variations of motion for the same character, there has been some efforts in modifying and reusing the existing data for other characters. Motion retargetting is the term used to describe the process of adapting an existing motion, created or captured for a specific character, to another character that has different dimensions and/or geometry [36].

Gleicher [36] introduced an optimisation technique to retarget existing animation to a new human figure which is identical in structure but different in dimension from the original one. Geometric constraints are treated by optimising a simple objective function. The simplified objective function discards some physical constraints in order to reduce computational cost. This may result in unrealistic poses.

A real-time retargetting of motion capture animation to various animated characters is introduced by Shin et al. [81]. The notion of dynamic importance is introduced to emphasise the importance of motion constraints that to be preserved. The defined constraints are preserved by the IK solver in real-time.

In [80], it has been suggested that motion analysis can be used for finding biomechanical information from the existing data. This biomechanical information can then be utilised in order to retarget the motion with better realism.

2.7 Animation Sequences/Streams

Most of the research in animation has concentrated on generating short clips. However, due to the increasing demand for animation techniques that generate long sequences, generating long sequences or continuous animation streams is becoming the focus of animation research. As our work is also concerned with generating longer and/or continuous animation sequences (as described in chapter 4), a brief review of the stages, difficulties and proposed solutions are described in this section.
“Motion Annotation” is one of the common tasks for synthesis of both animation clips and sequences. But “Motion Cyclification” is an example of challenging tasks specific to generating long animation sequences, especially from short clips.

Annotating motion is reviewed in section 2.7.1 while our proposed solution is discussed in section 3.4.3. The motion cyclification problem is also reviewed in section 2.7.2 and our novel cyclification techniques [5] are presented in section 4.3.1.

2.7.1 Motion Annotation

The first problem which is common to much work in this field is the annotation of input data in order to identify motion constraints and their timing (which will be referred to as the key-events timing or ‘KETs’). These events need to be synchronised among the various input clips in order to preserve the motion constraints and properties. In the majority of existing work, the annotation task is carried out manually. There have been some trials to help the user to identify the events, such as the use of a tool based on an iterative support vector machine which is introduced in [10]. Gleicher et. al. [37] introduced to annotate the input motion clips together while building a motion graph. Annotation is based on detecting the common poses, guided by the user, within the given motion clips. Kovar et. al. [54] extended this approach by building a strictly increasing spline curve passing through a grid constructed from the distance function introduced in [55]. The aim is to automatically synchronise and time-warp the input motions. One drawback of this approach is the complexity of constructing the time-warping curve. Another limitation is that in some cases such as reaching, up and down, or moving on a sloped floor, up and down, the corresponding logical parts of the motion may have the most dissimilar postures. In such cases, the input motions are incorrectly synchronised and blending can result in unrealistic animation. In our work, we have developed a simple gait analysis algorithm as a step towards automating the annotation stage as discussed in section 3.4.3. To the best of our knowledge, there is no stable method of automating this task up till now.
Chapter 2. Human Motion Animation: A Literature Review

2.7.2 Motion Cyclification

The second stage is the seamless joining or transition between consecutive motion clips. The problem is the artifacts resulting from unmatched start and end frames of the two consecutive motion clips. Within the main focus of our research on cyclic motions (which represent a major sector of human movements), the artifacts result from the difference between postures at boundaries of the given motion cycle, known as the cyclification problem [84]. This is usually the case in reality as human motion is not perfectly periodic.

Cyclification problem has received only limited investigation. Within the context of motion captured animation and editing, Sudarsky et. al. [88] introduced a cyclification operator that works on perfectly periodic cycles. In [38], Golam et. al. used blending to cyclify their unit cycle, which consisted of two or more actual motion cycles.

The cyclification problem has also been addressed within motion transition research. Rose et. al. [78] presented a cyclification method based on fitting a least squares cyclic B-spline to a modified motion curve. Gleicher et al. [37] used a similar approach based on finding a common pose between similar frames from different motion clips. Then the motion clips are modified using displacement maps to start and/or end with the common pose.

The work of Silva et. al. [84] is the most related one to our work in the cyclification problem. Their curve cyclification algorithm is based on using a windowing technique referred to as ‘LCT’ (Lapped Cosine Transform) to accomplish time warping whilst preserving frequency contents.

Our proposed cyclification techniques [5] benefit from some observed characteristics of motion cycles for different families of motions and employ these observations with simple animation processing algorithms to match the cycle boundaries. More detailed discussion about the cyclification problem, previous research and our proposed cyclification techniques are presented in chapter 4 (section 4.3.1).
2.8 This Thesis on the Research Map

In the previous sections, we reviewed the most related work on human animation and motion editing. We have also tried to comment on each category of research and its distinction from our work. However, we believe that it is worth clarifying the presented work on the research map of human animation.

Starting from the broad view of the field, and referring to figure 1.1 from chapter 1, the work presented in this thesis belongs to the "motion control" stage under the area of "Animation" of virtual humans. The purpose of the work is to provide a high-level parametric framework for the synthesis of human animation through editing and modifying existing data. So, it falls under the motion editing techniques.

In terms of the principal methodology, we aimed to employ effective and computationally efficient techniques. Furthermore, we aimed to avoid building complicated physical or statistical models of the required motions. The reason is to avoid unnecessary complexities and to make the system unrestricted to specific motion type (which the model is built for). In this regard, we adopted the interpolation and blending techniques which we have extended to serve our purpose as discussed in chapters 3 and 4.

In our approach we have been looking to simplify the motion editing and synthesis process, reducing pre-processing, computation costs, and the required data size (chapters 3, 4, and A). Our technique is working directly on the given data without building any statistical model or transition graphs.

The objective of most of the previous work is either editing some properties of individual clips to generate some variations of the original motion or to port the individual original clip to other characters with different dimensions and geometry. Our work is different as we are concerned with generating continuous and longer animation (as long as the user input exists. Chapter 4). Moreover, instead of editing individual clips, we synthesis new motion from a group of existing clips through our developed parametric blending technique.

Generating long animation sequences still has various problems at different stages. We have tackled most of the problems at the different stages and presented novel contri-
butions in order to generate smooth and realistic animation sequences which is derived by intuitive high-level parameters defined by the user in a natural way (Chapter 4).

2.9 Summary

Due to the increasing demand for synthesis of human animation for different applications, many techniques has been developed. The main purpose of this chapter was to give an overview of the key research in the human animation area which is related to the presented work and share the same interests. This includes advantages of existing human animation techniques and limitations that motivated our research.

Previous techniques of creating animation (such as traditional and dynamics methods, sections 2.4.1 and 2.4.2) suffer from limitations such as being labor intensive, time-consuming, computationally complex, and require low-level control parameters which are not intuitive to the animator.

Motion capture animation (section 2.4.3) has proved to produce the most realistic animation. However, editing the captured data is still an active research topic due to its difficulty. The focus of existing research is to correctly edit animation data to synthesis realistic animation. However, intuitive control over the synthesis of realistic human animation is still one of the challenging tasks.

Most previous research in human animation was focused on synthesis of short animation clips. Recently, in the last one or two years, generating long animation sequences has received the focus of research. Generating long animation sequences has some added challenges as discussed in section 2.7 (such as motion annotation and cyclifications) which we addressed in our work (chapter 4).

Our initial framework and the core methodology for our parametric approach in human animation synthesis is presented in the next chapter. The extension of that framework for parametric synthesis of long animation sequences is then discussed in chapter 4.
Chapter 3

Parametric Synthesis of Animation Clips

This chapter presents the initial framework for our parametric animation synthesis system. The focus is on synthesis of animation clips corresponding to given values of high-level parameters (e.g. slope or speed). This initial framework is extended in chapter 4 for arbitrary and continuous parametric synthesis of longer animation sequences. However, this chapter describes our motivations, defines the problems, and shows the core ideas of our proposed solution which works as the building blocks for the extended framework.

In section 3.1, the aim and motivation of carrying out this research is presented. Next, the problem definition is stated in section 3.2 followed by our proposed solution and methodology in section 3.4. Results and their evaluation are presented in section 3.5. Examples of candidate applications are discussed in section 3.6. This includes a framework of enriching simple animation database (section 3.6.1) and parametric synthesis of motion for different scaled characters (section 3.6.2). Finally, the principal contributions are emphasised in the chapter's summary in section 3.7.
3.1 Aim and Motivation

The main aim of this research is to develop a high-level parametric approach for controlling the synthesis of realistic human animation. This parametric approach is potentially useful in many applications such as games, film production, and 3D broadcasting. It is important to compromise between intuitive control, preserving motion realism, and providing computationally efficient solution.

This research was originally motivated by the fact that although realistic animation can be produced using motion capture data, these data are difficult to edit or modify after being captured. One of the main difficulties is caused by the nature of captured data, where there is no model or mathematical formula relating the data together. Hence, there are no high-level parameters one can use to edit the motion. The synthesis of an animation clip that is slightly different from an existing motion capture clip, may require repeating the capturing process. On the other hand, editing motion -especially captured data- can results in the violation of some motion constraints as well as affecting its realism and quality.

Many of the editing techniques mentioned in section 2.5 require an expert or trained user to deal with complex models and define the desired motion by manipulating low-level variables. Our aim is to allow both novice and expert animators to intuitively define their desired motions using the natural high-level motion parameters. The system will then utilise these high-level parameters to edit the existing animation data and generate the required motion.

Another motivation is to find a compromise between the parametric control, realism and quality of generated animation, and costs of generating these animations (e.g. computation cost, time and effort, and the overall financial cost).

3.2 Problem Definition

The focus of this thesis is on the parametric synthesis of human animation clips and sequences. There are common problems which are the focus of this chapter. Specific
3.2. Problem Definition

Problems related to generating animation sequences are investigated in chapter 4. Key problems include:

- Annotating input motions to determine the timing of the motion constraints and important events.
- Motion synchronisation for aligning the important events.
- Parameterisation and solving the inverse interpolation problem.
- Correcting and matching cycle boundaries for continuous and smooth transitions and concatenation of motions for generating longer sequences.
- Arbitrary and continuous parametric synthesis.

In this section, we define the problem of synthesising animation clips by editing and modifying existing motions to satisfy given parameters. In chapter 4, this problem is extended to include the synthesis of long animation sequences. We focus on modification through interpolation and blending of two or more motion clips, for the reasons discussed in section 3.1.

Suppose we have two motion clips $M_1$ and $M_2$ with different values of a high-level parameter (such as speed or slope) $P_1$ and $P_2$ respectively. The problem, in its simplest form, is how to synthesise a new motion $M_3$ that has a specific value $P_3$ of the high-level parameter (in chapter 4, this will include a continuous input of the parameter value as a function of time).

The problem can be formulated as:

$$M_3(X_3) = B_x(M_1, M_2, X_3)$$  \hspace{1cm} (3.1)$$

where $B_x$ represents a blending function between $M_1$ and $M_2$ as defined in section 3.4.1. $X_3$ is the blending factor such that $M_3$ will have the value $P_3$ of the high-level parameter.

The formulation above represents the normal blending operation where the blending factor, $X_3$ in this case, is the controlling parameter. However, $X_3$ is not a representative parameter of the motion, especially the high-level parameters. Moreover, the animator
has to carry out many trial and error sessions to find the best value of $X_3$ that generates the new motion $M_3$ with parameter value of $P_3$ or the nearest value.

One of our goals in this research is to provide the animator with a parametric motion blending tool in which the desired motion can be defined by intuitive parameter instead of the blending factor. This means that we need equation 3.1 to be in the form:

$$M_3(P_3) = B_p(M_1, M_2, P_3)$$  \hspace{1cm} (3.2)

$P_3$, the high-level parameter is more convenient to the user but on the other hand, it is not defined in the blending operation space. Hence, to compromise between the two formulas, the relationship between blending factor $X_3$ and high-level parameter $P_3$ is required. This relationship can be in the form of $X_3 = f(P_3)$ and $0 \leq X_3 \leq 1$, where $f$ is a monotonic function. In such a form, animator can define the required motion by a high-level parameter $P_3$, the corresponding blending factor $X_3$ is calculated and used in blending the given motions. We tackled this problem in [4] and the solution is discussed in section 3.4.2.

A related problem is that the blending function in equation 3.2 can generate incorrect motion. This is due to the different timing of motion constraints and important events which we will refer to as the ‘key – event times’. For example, if we have two walking motions with different speeds, the event of the left, heel touching the floor will occur at different times in each of the given motions. Hence, the blend may produce animation that violates motion constraints. To avoid violation of constraints, input motions should be synchronised such that corresponding important events occur simultaneously. This problem is summarised in the following formulation:

Given two motions $M_1$ and $M_2$ with list of key-event times $[T^1]$ and $[T^2]$ respectively, we need a function $S$ such that $\tilde{M}_2 = S(M_2)$ is synchronised with $M_1$ (i.e. its list of key-event times $[\tilde{T}^2]$ is equal to $[T^1]$). Motion synchronisation is addressed in section 3.4.4.

In an associated problem, in order to synchronise input motions, both motion properties and key-event times need to be determined beforehand. This problem is known as the ‘motion annotation’ problem. Its difficulty is due to varying properties and important events across different motions. It is common that input motions are classi-
3.3. Data Representation

It is essential, when designing a motion editing system, to understand what the numbers contained in the given motion data represent. This also includes the representation of the character model (representation of the virtual human) itself. In this section we describe the character and motion representations we have used in this thesis. Our policy is to use the most common representations for compatibility with existing data and other animation tools.

A character is commonly represented by a skeleton structure; a simplified representation of the virtual human. The skeleton structure representation consists of a set of hierarchical rigid segments (bones) connected by joints. In editing animation, one of the requirements is to preserve the lengths of those rigid segments. Another requirement is that those segments must stay connected at the joints. Therefore, only translation and rotation transformations are applicable for moving the skeleton. Other transformations, such as scaling, will change the length of bones and may result in detaching them from each others.

Usually, the pose of a character at a specific time is defined by the global position and orientation of one of the joints, known as the root joint, and the relative orientation of the subsequent joints throughout the hierarchy. The root joint can be any joint in the hierarchy although the pelvis joint, the body centre, is a common choice.

Motion capture data (MOCAP) can be obtained from various sources as discussed in chapter 2 and is selected as the data source for this research. The reason is that MOCAP data provides a source of realistic animation as it is measuring a real human movements. However, the proposed techniques could be applied on data from almost any other source.
Chapter 3. Parametric Synthesis of Animation Clips

Motion capture systems, especially optical ones, measure a set of points on the body surface in a global coordinates. This allows for producing the pose of the character at each frame. This form of data is not suitable for editing as it can violate the assumption of rigid and connected segments of the skeleton. Hence, most of the animation formats use the absolute position and orientation only for the root joint, and the relative orientations of the rest of the joints.

Representing position is a simple and straightforward task. However, representing orientation is a difficult one. Thorough discussion of various representations of 3D orientation is beyond the scope of this thesis. However, in this section, we briefly summarise the commonly used representations. For more details, see [82, 33, 83, 39, 51].

Crucially, there is no ideal representation for all tasks. Each representation has its own \textit{pros} and \textit{cons} and can be more suitable for certain tasks than others [39]. A basic representation that is often used in graphics libraries is the \textit{rotation matrix}. A \textit{rotation matrix} is an orthonormal matrix that expresses 3D rotations in terms of 9 values (3x3 matrix). However, it is not convenient for editing because changing one value can lead to a matrix that is no longer represents a rotation (i.e. becomes non-orthonormal matrix).

In the \textit{Axis — Angle} representation, 3D orientation is represented by one rotation angle about one axis. It needs 4 values, angle and 3D vector representing the rotation axis. However, using a unit vector, this representation can be compacted into 3 values (sometimes referred to as the rotation vector). Although this is a more intuitive representation, it is difficult to interpolate between two rotations [51].

An alternative is the \textit{quaternion} representation which uses 4 values to express 3D orientation. It should be noted that only the unit-magnitude quaternion represents a rotation. As with the rotation matrix, changing individual values can violate this condition. Moreover, the numbers themselves are not intuitive. Despite this limitation, quaternions are widely used as they have some advantages. These include the well-defined operations (including interpolation) which preserve the unit-magnitude [83].

\textit{Exponential maps} representation has been recently introduced [39] which uses 3 values for representing 3D orientation. It shares similar advantages and disadvantages
of Euler angles (discussed below). However, it is easier to dynamically change the parameterisation to avoid the singularities.

Euler angles have been the most common representation for a long time. It is a compact representation as it uses only 3 values to represent 3D orientation. Orientation is represented by 3 successive rotations about specified axes. Unlike rotation matrix and quaternion, any 3 values in Euler angles representation represent a rotation. However, it should be noted that those 3 values are not independent and their interpolation is difficult. Euler angles also have other limitations such as singularities and gymbal—lock [99, 83]. Regardless of its limitations, it is commonly used in motion editing. This is because of the assumption that effect of those limitations is small with small difference in rotations between frames.

To summarise, any 3 degrees-of-freedom representation of rotation will suffer from singularity. The parameters are often used together with associated constraints.

Amongst the available representations, Euler angles are selected in this thesis for its popularity, compactness, and for compatibility with the common animation file formats. The representation is also more intuitive than quaternion. To overcome the difficulties of interpolation, the actual interpolation is internally implemented through axis/angle or quaternion.

The motion data at any specific frame $F$ (a specific point in time) is represented as:

$$M^F = \{P^F, R^F_1, R^F_2, ..., R^F_J\};$$

where $F = 1, ..., N$; $N =$ number of frames;

$P^F$ is the positional data of the root joint at frame $F$;

$R^F_i$ is the orientational data (Euler angles) of joint $i$ at frame $F$;

The rest of this chapter presents our parametric approach for synthesis of animation clips. Various problems at different stages are identified and tackled.

### 3.4 Proposed Parametric Approach

The proposed parametric approach, for synthesis of animation clips, is introduced in this section. In this approach, two main stages can be identified as depicted in figure 3.1;
adding new motion clips into an animation database, and parametric synthesis of new clips for given parameters. As discussed in the problem definition, section 3.2, there is a number of problems to be solved at different stages as shown in the rest of this section.

Adding new motion clips to the animation database involves motion annotation (section 3.4.3), synchronisation (section 3.4.4), and determination of the mapping function $f$ from user parameter to blending factor such that $X_i = f_i(P_i)$ (section 3.4.2). This stage is summarised in the algorithm in figure 3.2 as well as in the top of the diagram in figure 3.1.

Parametric synthesis of motion clips according to user-specified parameter involves automatic generation of the corresponding blending factor (section 3.4.2), motion synchronisation (section 3.4.4), and motion blending (section 3.4.1) as depicted at the bottom of the diagram in figure 3.1. Provided that the appropriate motion clips $M_1$ and $M_2$, their key-events timing $[T^1]$ and $[T^2]$, and parameterisation relationship (mapping function) are available in the animation database (as described in the addition procedure in figure 3.2), the procedure described in figure 3.3 can be used for parametric synthesis.

Firstly, the blending factor generator uses the pre-determined function $f_i$ to automatically obtain the corresponding blending factor $X_{iu}$ for a given user-specified parameter value $P_{iu}$. Next, the two motions involved in the synthesis, $M_1$ and $M_2$ are synchronised and blended with the determined blending factor $X_{iu}$.

The following sub-sections discuss the different stages of our proposed approach.

3.4.1 Synchronised Motion Blending

At this stage, synchronised blending stage (figure 3.1), it is assumed that we have annotated motions as described in section 3.4.3. Synchronised motion blending is used to generate a new motion that satisfies a given parameter value $P_{iu}$ from the user.

For a given parameter value $P_{iu}$, the corresponding blending factor is obtained from the formula $X_{iu} = f(P_{iu})$ as discussed in section 3.4.2.
3.4. Proposed Parametric Approach

Figure 3.1: Block diagram of the Parametric Animation approach
Chapter 3. Parametric Synthesis of Animation Clips

Input: Two motion clips $M_1$ and $M_2$

Output:  
* Annotated motion clips; $M_1$ and $M_2$ with their list of key-event times $[T^1]$ and $[T^2]$ respectively  
* Determined relationships between blending factors $X_i$ and parameters $P_i$

Procedure:  
1. Annotate given motions to determine their key-event times.  
   
   $[T^1] = \text{Annotate}(M_1)$;  
   $[T^2] = \text{Annotate}(M_2)$;  
2. Add annotated motions to the animation database  
3. Synchronise the annotated motions.  
   Assuming that $M_1$ is selected as the reference motion:  
   
   $\bar{M}_2 = \text{SYNC}(M_2, [T^1])$;  
   where $\text{SYNC}$ function is as defined in section 3.4.4  
4. Determine relationship between blending factors $X_i$ and parameters $P_i$. i.e. determine $X_i = f(P_i)$;

Figure 3.2: Procedure of adding new motion clips to the animation database
3.4. Proposed Parametric Approach

**Input:**
* Desired parameter value from user $P_{iu}$
* Two motion clips $M_1$ and $M_2$
* key-event times $[T^1]$ and $[T^2]$ for the given motions
* Parameterisation relationship $f_i$

**Output:**
* New motion clip $M_3$ satisfying the user-specified parameter
* New key-event times $[T^3]$ for the new motion $M_3$

**Procedure:**
1. Appropriate blending factor $X_{iu}$ is generated automatically for the given parameter $P_{iu}$ using the pre-determined parameterisation relationship $f_i$:
   $$X_{iu} = f_i(P_{iu});$$
2. Synchronised blending technique (discussed in section 3.4.1) is applied:
   $$M_3 = SyncBlend(M1, M2, X_{iu});$$

**Note:**
* $SYNC$ function is defined in section 3.4.4
* $SyncBlend$ function is defined in section 3.4.1

Figure 3.3: Parametric Synthesis Procedure
Chapter 3. Parametric Synthesis of Animation Clips

The synchronised blending function is written as:

\[ M_3 = \text{SyncBlend}(M_1, M_2, X_{iu}) \]  \hspace{1cm} (3.3)

where \( X_{iu} = f(P_{iu}) \); is the blend factor corresponding to a user-specified parameter \( P_{iu} \).

Synchronisation and blending (SyncBlend) are implemented in the following steps (assuming that \( M_1 \) is selected as a reference motion):

a. Synchronise input motions.
   \[ \tilde{M}_2 = \text{SYNC}(M_2, [T^1]) \]

b. Blend synchronised motions.
   \[ \tilde{M}_3 = \text{BLEND}(M_1, \tilde{M}_2, X_i) \]

c. Estimate the new motion timings
   \[ [T^3] = \text{BLENDKETS}([T^1], [T^2], X_{1}) \]

d. Re-synchronise generated motion to its estimated timing.
   \[ M_3 = \text{SYNC}(\tilde{M}_3, [T^3]) \]

The \textit{SYNC} function is defined in section 3.4.4.

It is important to note that the generated motion \( \tilde{M}_3 \) is still synchronised with the reference motion \( M_1 \). To produce the generated motion in its own timing, another synchronisation process is required as shown in step ‘d’ above. Key-event timing of the generated motion is estimated from input motions as shown in step ‘c’.

For the \textit{BLEND} function, equation 3.1 from section 3.2 can be re-written as:

\[ M_3 = \text{BLEND}(M_1, \tilde{M}_2, X_{iu}) \]  \hspace{1cm} (3.4)

where \( X_{iu} \) is the blending factor corresponding to user-specified parameter \( P_{iu} \). \textit{BLEND} represents the blending function between \( M_1 \) and \( \tilde{M}_2 \). In our case, a linear blending function has been found practical and simple enough to achieve our aim of synthesising motion that satisfy motion parameter value given by the user.

As mentioned in section 3.3, animation clips are represented by positional and orientational data. Implementation of the \textit{BLEND} function will have different forms for position and orientation degrees of freedom, based on our selection of orientation representation.
For positional data, as discussed in section 3.4.4, 3D data can be dealt with as independent time-varying signals. Hence, the implementation of \( BLEND \) for positional data, \( BLENDP \) can be written as:

\[
\begin{align*}
\tilde{M}_3(P_d^p) &= BLENDP(M_1(P_d^p), \tilde{M}_2(P_d^p), X_{iu}) \\
\tilde{M}_3(P_d^p) &= X_{iu}M_1(P_d^p) + (1 - X_{iu})M_2(P_d^p)
\end{align*}
\]

where \( d = 1, \ldots, 3 \) is the index of positional DOFs.

The orientation data needs more attention because degrees of freedom for each joint are dependent on each other. \( BLENDR \), the implementation of \( BLEND \) for orientational data, takes the form:

\[
\begin{align*}
\tilde{M}_3(R_d^p) &= BLENDR(M_1(R_d^p), \tilde{M}_2(R_d^p), X_{iu}) \\
\tilde{M}_3(R_d^p) &= \text{InterpolateRot}(M_1(R_d^p), M_2(R_d^p), X_{iu});
\end{align*}
\]

where \( i = 1, \ldots, j \) is the joint number.

Applying the formulas 3.5 and 3.6 will generate a motion \( \tilde{M}_3 \) which is synchronised with input motions (more specifically, synchronised with the reference motion \( M_1 \)). This generated motion should be re-warped in time to reflect its own key-event timing and satisfy the given parameter. Key-event times of the generated motion can be calculated from the key-event times of the input motions as follows:

\[
[T^3] = BLENDKETS([T^1], [T^2])
\]

\[
i.e.
T^3_j = X_{iu}T^1_j + (1 - X_{iu})T^2_j
\]

for \( j=1,\ldots,k \); where \( k \) is the number of key-events.

Using the estimated key-events timing \( [T^3] \), \( \tilde{M}_3 \) can be re-warped by applying the \( SYNC \) function as:

\[
M_3 = SYNC(\tilde{M}_3, [T^3])
\]

The result is the motion \( M_3 \) that satisfies the user-specified parameter \( P_{iu} \).

Estimation of the key-event times of the generated motion (Equation 3.7) assumes that the key-event times scale linearly between the two input motions. To justify this assumption, a few comparisons have been made between real clips and their corresponding
synthesised motions (with the same value of the parameter). Examples include generating a level-floor straight-line walking sequence at a specified speed (obtained from the real motion capture sample) with synthesised clips from curved-path walking, and from sloped-floor walking. The comparison between captured and generated motions are presented in figures 3.4 and 3.5 respectively. The key-event times for those examples are plotted in figures 3.6 and 3.7 respectively.

Figure 3.4: Real motion capture straight-path walking (in blue) compared with its corresponding synthesised motion (in red) from curved-path walking motions

Figure 3.5: Real motion capture straight-path walking (in blue) compared with its corresponding synthesised motion (in red) from Sloped-floor walking motions

The charts suggest that the key-events timing of the synthesised clip do not exactly
Figure 3.6: Key-event times for both Real and synthesised motions (from curved-path clips) for a straight-line walking movement.

Figure 3.7: Key-event times for both Real and synthesised motions (from Sloped-floor clips) for a straight-line walking movement.
match the corresponding values from the motion capture clip, especially for the higher key-event numbers (KET3 to KET5). One of the reasons of that difference (at higher key-event numbers and especially for the last key-event timing) is that the synthesised motion is limited by the input motions in the sense that, its total number of frames per cycle will be in the range defined by the input motions. For example, the captured motion of straight-path walking has 37 frames/cycle while both curved-path walking inputs have only 35 frames/cycle. Hence, the synthesised straight-path motion has only 35 frames/cycle (unlike the motion capture one).

Another important reason is the difference in motion style, personality, and the variation of natural human movement where the same movement is not perfectly re-produced (even by the same person within the same capturing session). Due to the nature of human movement, a motion with the same value of a parameter (such as speed) can be achieved with different timing of key-events.

The variation of key-event timing of real data is depicted in figure 3.8 for various speeds, styles and personalities. This graph shows the high level of variability in key-event times for real data. To model this variability would require a large amount of data.

This limited analysis indicates that a linear approximation to key-event time interpolation is reasonable.

![Figure 3.8: Variations of Key-event times in real human movement.](image)

Since our system is not restricted to motion capture data (i.e. it is applicable on
animation data created by various techniques), and for more systematic comparison, we carried out a comparison with data created by a procedural method [19, 21, 20].

In this experiment, two walking clips with different speeds (Slow: 85.48 cm/sec and Fast: 229.8 cm/sec) are generated and used as input clips. Other walking clips with different speeds, in-between the slow and fast sequences that are used as inputs, are generated to be the ground truth for comparison. Our system is used to synthesise walking clips with speeds corresponding to the speeds of the ground-truth clips. The key-event times of both synthesised and the ground-truth clips are presented in figure 3.9.

The results shown in Figure 3.9 demonstrates similar variation in key-event times for procedural and our proposed approach. The behaviour of the estimated key-events timing is similar to the behaviour of key-events timing of the ground-truth clips.

![Figure 3.9: Key-event times for both synthesised and ground-truth motions (from a procedural method)](image)

3.4.2 Solving the Inverse Interpolation Problem

The blending process is controlled by the interpolation weights $X_t$. For the parametric synthesis of motion with a specified parameter $P_i$ (e.g. speed, slope,...etc.), it is desirable to control the blending process through the parameter $P_i$. Using the traditional
trial and error approach can lead to a function $g$ such as $P_i = g(X_i)$. However, what is required is a function $f$ such as $X_i = f(P_i)$ (i.e. $f = g^{-1}$). This problem is called the inverse interpolation problem [90] and is rarely addressed in literature.

The relationship between the given parameter $P_i$ and interpolation weight $X_i$ is often assumed to be linear [52, 101]. This is a simplification of the real case where a linear relationship is not always a valid or accurate assumption. For example, using a blending factor of 0.5 does not always mean that the generated motion will have a parameter value exactly half-way between the two parameter values of original motions. The animator ends up performing many adjustments until the required motion is generated.

There is more than one factor that can affect the shape of this relationship. This includes the parameter itself, its calculation method, and the animation processing involved during the synthesis process.

To release the animator from the burden of trial and error sessions, a suitable formulation for the function $f$ is required. Crucially, this formulation needs to be determined automatically [4]. The function $f(P_i)$ should reflect the actual relationship, or its best fit, between the parameter $P_i$ and blending factor $X_i$.

In this section we automate the determination of the function $f$ by simulating the forward process of animation synthesis [4] for given values of blending factor $X$ and calculating the resulting parameter values $P$ as follows:

\[
\text{for } X_i = 0 : 1, \text{ Step } = 0.1 \\
M_i(X_i) = \text{SyncBlend}(M_1, M_2, X_i); \\
P_i^{X_i} = \text{CalcParam}(M_i); \\
\text{end;}
\]

where the function $\text{SyncBlend}$ is implemented as discussed in section 3.4.1.

This gives an analytic representation for $P = g(X)$. To enable synthesis of a motion clip with a user specified parameter value $P_i$, we then evaluate an analytic form for the inverse process $X = f(P)$ where $f = g^{-1}$. The function $f$ can be determined by approximating the data $(P_i, X_i)$, generated from the above simulation, by an algebraic function. Figures 3.10 and 3.11 depict an example of $P = g(X)$ and $X = f(P)$ respectively for human walking where $P$ represents the walking speed.
Using the above automatic simulation simplifies the problem of determining the function $f$. As soon as the function $f$ is determined, it can be used to generate corresponding blending factor $X$ for a desired parameter value $P$.

![Walking Speed (V) vs. Blending Factor (X)](image)

Figure 3.10: Example of the function $P = g(X)$: Walking Speed ‘$V$’ as a function of blending factor ‘$X$’

### 3.4.3 Motion Annotation

As in previous work in literature (e.g. [101, 77, 79]), the motion samples need to be annotated before being used. Annotation is required to identify the motion constraints and their timing. In the rest of the thesis, timing of important events within a motion will be referred to as the ‘Key – Event’ times (KETs).

The constraints and their timing are part of the motion properties. Hence, identifying key-events vary from one motion to another. As mentioned in section 2.7.1, there are few trials to address the problem of automatic annotation [10, 37, 13]. However, in general, there is no robust solution for this problem until now.
Initially, we annotated the input motions manually. Later, a simple framework was developed [5] for automatic gait analysis of cyclic movements. It worth noting that gait motions represent the majority of cyclic movements. Given a motion clip \( M \), the procedure below is used to automatically detect the key-event times (including cycle boundaries) as follows:

1. Calculate distance between the left and right foot at each frame of the given motion:
   \[
   D_i = ||X_{LF} - X_{RF}||_i
   \]
   where \( X_{LF} \) and \( X_{RF} \) are the positions of the Left and Right foot respectively.
2. Scan the distance vector $D$ for the local minimums and maximums (See example in figure 3.12):

\[
N = 1; \\
\text{for } i = 2 : \text{NumOfFrames} \\
KET(N) = \text{FindCriticalPoint}(D(i : \text{end})); \\
\text{Update } i; \\
N = N + 1; \\
\text{end;}
\]

3. From the list of local maximums and minimums, the key event times can be detected as follows (also see figure 3.12):

- The first local maximum is corresponding to the first foot touching the floor which abbreviated by $KET_1$.
- The first local minimum is corresponding to the mid-swing of the second foot, abbreviated by $KET_2$.
- The second local maximum is corresponding to the second foot touching the floor which abbreviated by $KET_3$.
- The second local minimum is corresponding to the mid-swing of the 1st foot abbreviated by $KET_4$
- The 2nd local maximum is corresponding to the first foot touching the floor again, which abbreviated by $KET_5$.

The above detection procedure has been successfully applied to many gait motions with normal and abnormal cycles like limp, slope, and corrupted cycles as shown in the results in section 3.5.7.

One of the main advantages of this technique is its simplicity. The distance is calculated between only two joints (the feet as end-effectors) instead of involving the whole skeleton as in the matching pose technique [37]. Another advantage is that it detects the gait timing of each motion separately regardless of other existing motions. Hence, inserting
Chapter 3. Parametric Synthesis of Animation Clips

Distance between feet
(Normal Walk Cycle, 33 frames/cycle)

Figure 3.12: Automatic detection of gait timing

or deleting a motion clip does not require updating existing motions or their detected key-events. The limitation is that it is dedicated to gait analysis. However, gait motions represent the majority of cyclic movement. Moreover, as mentioned before, constraints and their timing are part of the motion properties which vary from one motion to another.

3.4.4 Motion Synchronisation

Applying the blending function on the given motion directly may produce results that violate motion constraints. The simple stretching or compression, in time, of one motion to unify the total time duration does not guarantee the synchronisation of key-events. To eliminate or minimise violation of motion constraints, input motions should be synchronised together in order to align their key-events.

Given two motions $M_1$ and $M_2$ with a list of corresponding key-event times $[T^1]$ and $[T^2]$ respectively, we need to obtain a motion $\hat{M}_2$ which is synchronised with $M_1$ (i.e. its key-event times $[\hat{T}^2]$ is equal to $[T^1]$). $\hat{M}_2$ can be calculated from:

$$\hat{M}_2 = \text{SYNC}(M_2, [\hat{T}^2])$$  \hspace{1cm} (3.9)
where $SYNC$ is the synchronisation function, described later in this section, and $[\hat{t}^2]$ is
the list of aligned timing for all frames in $M_2$.

Before using the above formula, the list of aligned timing $[\hat{t}^2]$ should be determined. This can be achieved using time-warping. Time-warping can be implemented using various formulations. For example, Witken et. al [105] used cardinal spline while Sun [91] used a linear warping function and reported satisfactory results in practice. We adopted the linear warping function from [91] as follows.

Given two lists of key-event times $[T^1]$ and $[T^2]$ for motions $M_1$ and $M_2$ respectively, for each frame $j$ at time $t^1_j \in [T^1_i, T^1_{i+1}]$, the corresponding frame in $M_2$ is the frame at time $t^2_{ij}$ such that:

$$
\hat{t}^2_{ij} = T^2_i + (t^1_j - T^1_i) \ast \frac{(T^2_{i+1} - T^2_i)}{(T^1_{i+1} - T^1_i)}
$$

(3.10)

for $i = 1, \ldots, K - 1$, where $K$ is the number of key-events and $j = 1, \ldots, N_i$, where $N_i$ is the number of frames between two consecutive key-event times $[T^1_i, T^1_{i+1}]$.

Then, the aligned list of timing needed for equation 3.9 is $[\hat{t}^2] = [\hat{t}^2_{11}, \hat{t}^2_{12}, \ldots, \hat{t}^2_{(K-1)(N_{K-1})}]$. This mapping ensures an exact match for key-event times such that $[\hat{t}^2] = [T^1]$. Intermediate frames are at interpolated times $[\hat{t}^2]$.

Based on the selected data representation as discussed in section 3.3, two different implementations of the function $SYNC$ have been developed for the two existing types of data. The implementation of the synchronising function $SYNC$ for positional data $SYNC_P$ is:

$$
\tilde{M}_2 (P^F_d) = SYNC_P (M_2 (P^F_d), [\hat{t}^2])
$$

(3.11)

$$
\tilde{M}_2 (P^F_d) = \begin{cases}
    M_2 (P^F_d) & \text{if } \hat{t}^2_F \text{ is integer} \\
    K_1 + (\hat{t}^2_F - a_1) \frac{(K_2 - K_1)}{(a_2 - a_1)} & \text{if } \hat{t}^2_F \text{ is real}
\end{cases}
$$
Chapter 3. Parametric Synthesis of Animation Clips

where \( d \) = 1, ..., 3; is index of positional DOFs
\( F = 1, ..., N; \) \( N \) is the number of frames in the reference motion \( M_1 \)
\( a_1 = \text{floor}(\hat{t}_F); \)
\( a_2 = \text{ceil}(\hat{t}_F); \)
\( K_1 = M_2(P_d^{a_1}); \)
\( K_2 = M_2(P_d^{a_2}); \)

The synchronisation function for orientation data \( SYNC_R \) is as follows:

\[
M_2(R_i^F) = SYNC_R(M_2(R_i^F), [\hat{t}_i^2])
\]

\[
\hat{M}_2(R_i^F) = \begin{cases} 
M_2(R_i^F) & \text{if } \hat{t}_F^i \text{ is integer} \\
\text{InterpolateRot}(R_i^{a_1}, R_i^{a_2}, \alpha) & \text{if } \hat{t}_F^i \text{ is real}
\end{cases}
\]

where \( i = 1, ..., J; \) \( J \) is the number of joints
\( a_1 = \text{floor}(\hat{t}_F^i); \)
\( a_2 = \text{ceil}(\hat{t}_F^i); \)
\( \alpha = \hat{t}_F^i - a_1; \) \( \alpha \) is the interpolation weight;

It should be noted that the ‘InterpolateRot’ function implements quaternion interpolation. It is also worth mentioning that for practical use, the orientation data should be internally represented by quaternions. This reduces the overhead of converting from Euler angles to quaternion, and back, every time for blending. However, we preferred to keep the Euler angle representation for the research and experimental work only. This is because Euler angles representation is more intuitive than quaternion representation.

By applying the equations 3.10 to 3.12 we should have a new motion \( \tilde{M}_2 \) which is synchronised with the reference motion \( M_1 \). So, motions are now ready for blending. However, to achieve the parametric blending, the relation between motion parameter and blending factor should be determined beforehand. This problem is discussed in section 3.4.2 with our proposed solution.

For simplicity, the notion \( SYNC \) is used for the synchronisation function which includes both position and orientation synchronisation functions, \( SYNC_P \) and \( SYNC_R \) respectively.
3.5 Results and Evaluation

This section presents results produced using the initial framework discussed in the previous sections. Results have been generated from both synthetic motions, from the LifeForms package [87], and real human motion capture data, obtained either from our lab or from the SantaMonica motion capture database.

In our experiments, an optical system (CODA system [69]), with active markers and 4 detector boxes, is used. Figure 3.13 shows the markers setup on the performer for a motion capture session. The CODA system has been driven through the FiLMBOX animation package [49] for real-time monitoring and displaying of the captured data. The captured data is converted from positions into the hierarchical skeleton representation (position and orientation of the root and relative orientation of other joints) through Markers – Actor – Character mappings specified within the FiLMBOX package. The output is exported in BioVision format (BVH), which is widely used animation format.

Figure 3.13: Markers setup on the performer for a Motion Capture session.

The parametric synthesis approach has been applied successfully on different motions and motion parameters such as walking and running on a level floor as well as on a sloped floor, jumping, and kicking as shown in following subsections.

---

1LifeForms is an animation software from Credo Interactive Inc.
2SantaMonica motion capture database is available from Stanford University at http://www.stanford.edu/class/cs348c/BVH/SantaMonicaStudios/.
3CODAmotion is a real-time optical motion capture system from Charnwood Dynamics Ltd.
4FiLMBOX is a registered trademark of Kaydara Inc.
Results are also accompanied with evaluation of their quality. Evaluating animation, especially its quality, is not an easy task. It is worth mentioning that the visual inspection and evaluation is ultimately the key criteria for assessing the animation quality. This is because the main impact of animation is what the user will actually see and perceive. The problem with visual evaluation is that it is a subjective measure. Hence, it may not provide a consistent evaluation as it is subject to human experience and other temporal modes. Unlike some other fields (e.g. image and signal processing), there is no standard or unified criteria for evaluating animation quality. Having no ground-truth for comparison also increases the difficulty of the evaluation process. To provide some consistency in evaluation, we combined both qualitative and quantitative measures in order to evaluate the results.

Qualitative evaluation is performed by visually inspecting the results and compare their quality with the given original motion clips. Visual inspection aims to spot violation of motion constraints, sudden movements, and evaluate the naturalness and realism of the motion. Our qualitative evaluation experiments are explained in appendix C.

Quantitative evaluation is achieved through incorporating quantitative measures such as 'Jerk' and 'Feet-Floor Contact' measures. The definition of these measures and the resulting evaluation is discussed in the following subsection.

3.5.1 Quantitative Evaluation

For quantitative evaluation, despite the lack of ground-truth and standard quantitative measures, a few individual quantitative measures have been introduced in the last few years [67, 1]. In this section, we introduce the 'Feet – Floor Contact Error' as a quantitative measure of animation quality. We also adopted the ‘Jerk’ measure from [67] as a general measure of smoothness in the joints movements. Both measures are defined and explained below.

The jerk measure is defined as the derivative of the joint’s acceleration. That is for a joint number \( n \), given the joint positions vector \( \vec{P}^t \), the jerk at time \( t_{i-1} \) is calculated as:

\[
\vec{j}_{t_{i-1}}^n = \frac{\vec{P}_{t_{i-1}}^n - 3\vec{P}_{t_{i-2}}^n + 3\vec{P}_{t_{i-3}}^n - \vec{P}_{t_{i-3}}^n}{\Delta^3}
\] (3.13)
where: \( i = 3, ..., F; \) where \( F \) is the total number of frames.
\( n = 1, ..., N; \) where \( N \) is the number of joints in the skeleton.
\( \Delta = t_i - t_{i-1}; \) is the frame time of the given animation (i.e. sampling period of the captured data).

The jerk measure is designed to detect sudden changes in the joint movements. It is also considered as a general measure that is independent of the motion type. However, even with low values of jerk, animation can still suffer from other artifacts which result from violation of motion constraints. One of the important constraints is the 'feet-floor contact' which is common in many gait motions (e.g. walk, run,..., etc.).

The failure of the foot to keep in touch with the floor at certain times results in noticeable artifacts such as feet sliding, penetrating, and/or flying over the floor. To identify these errors, we introduced the 'Feet-Floor Contact Error' as a quantitative quality measure. The idea is to compare the actual foot position with the position that it should be in at certain periods. An example of these periods are the periods of supporting leg in walking and running motions. During these support periods, the supporting foot should be planted into its position. Any shift from this position is calculated and saved over the whole animation period as follows:

\[
\overline{S_i} = \overline{F_i} - \overline{C_i}
\]

where:
\( \overline{S_i} \) is the calculated shift in the foot position from its correct position at frame \( i \).
\( \overline{F_i} \) is the actual foot position at frame \( i \).
\( \overline{C_i} \) is the correct/expected foot position at frame \( i \).

Both maximum and Root – Mean – Square (RMS) values of the measures are used for evaluation. The maximum values, relative to its corresponding value of the original animation, indicate the maximum error that may have been introduced in the generated animation. While the RMS values give indication of the variation of errors over the animation period.

The RMS value of the Feet – Floor Contact Error measure, referred to as 'F' in the
rest of the thesis, is calculated as follows:

\[ F = \sqrt{\frac{\sum_{i=1}^{N} S_i^2}{N}} \]  

(3.15)

where \( N \) is the number of frames in the animation clip.

similarly, the \textit{Jerk} measure \( J \) is taken as:

\[ J = \sqrt{\frac{\sum_{i=1}^{N} J_i^2}{N}} \]  

(3.16)

The measures defined in equations 3.13, 3.14, 3.15 and 3.16 above are used in the rest of this section for evaluating both the resulting and the original animation clips and to compare results with original animation.

The remainder of this section presents samples of the generated clips that satisfy the desired values of input motion parameters (e.g. speed and slope) within an error of 2% of the desired parameter value.

3.5.2 Walk on a level floor

Figure 3.14 depicts the results of the proposed approach in generating a ‘walk’ motion on a level floor with a user specified speed. The relationship between the blending factor and the desired motion parameter is extracted, as described in section 3.4.2. Then, this relationship is utilised to provide the proper value of blending factor which used to synthesis the motion that satisfy the given motion parameter.

The generated motions satisfy the requested speed with 1.86% maximum error as shown in figure 3.15. This is equivalent to a maximum speed error of 2.26 cm/sec. An example of the speed profile of the root joint for both input clips and a sample output clip is shown in figure 3.16.

The quantitative measures, depicted in figures 3.17 also shows that the quality of the resulting motions are quiet close to the quality of the input motions. It is worth noting
3.5. Results and Evaluation

Figure 3.14: Synthesised ‘Walk’ motion on level floor with desired speed given by user. Original clips are ‘SlowWalk’ (left in blue) with speed of 73.74 \( \text{cm/sec} \) and ‘FastWalk’ (right in green) with speed of 150.6 \( \text{cm/sec} \).
Chapter 3. Parametric Synthesis of Animation Clips

Figure 3.15: Output speed vs. Input speed of Walk examples.

Figure 3.16: Speed profile of the Root joint for both input clips and a sample output clip.
that the measures given by equations 3.13, 3.14, 3.15 and 3.16 represent errors and unwanted properties in the animation (unless sudden changes are part of the motion properties which is not commonly the case). Hence, the less the value of the quantitative measure, the better the animation quality.

Figure 3.17: Evaluation of result of controlling the speed of ‘walk’ motions on level floor. The two ends of each curve represent the measure value of input motions.

Figure 3.18 shows similar examples for a ‘run’ motion. The generated motions satisfy the requested speed with 1.4% maximum error as depicted in figure 3.19.

Figure 3.20 shows the quantitative measures which indicate that the quality of the resulting motions are close to the quality of the input motions.

3.5.4 Walk on a sloped floor

In another example, motions on defined slopes, expressed in terms of the slope angle in degrees, have been successfully synthesised with error in the range of $-0.0492^\circ$ to $0.06269^\circ$. This is equivalent to a range of $-0.4972\%$ to $1.1\%$ of the desired slope
Figure 3.18: Samples of generating 'Run' motion with a specified speeds given by the user. Original clips are 'Slow Run' (right in blue) with speed of 863.5 cm/sec and 'Fast Run' (left in green) with speed of 1452.7 cm/sec.
3.5. Results and Evaluation

Figure 3.19: Output speed vs. Input speed of Run examples.

Figure 3.20: Evaluation of result of controlling the speed of 'run' motions. The two ends of each curve represent the measure value of input motions.
requested by the user. Figure 3.21 depicts some examples of motions on specified slopes while figure 3.22 shows the input-output relationship for the slope parameter.

The quality of the resulting motions is close to the input motion clips. The error measures even drop below their corresponding minimum values of the input motions in some cases as shown in figures 3.23. This drop is attributed to the fact that the input motions are not moving in the same direction as in the walk on level floor example. At some frames during the motion, moving in different directions will have components in opposite directions, up and down in this case, which can cancel each other (depending on the blending factor). This cancellation effect can reduce the feet-floor contact errors and may also reduce the sudden movements at certain frames and specific blending factors as shown in figure 3.23.

3.5.5 Jump

The last example, depicted in figure 3.24, shows a 'jump' motion where the user can intuitively control the direction to which the avatar should be facing at the end of the generated jump motion. This is achieved by controlling the turn angle between initial and final positions of the jump. Given two jump motions, jump in place and jump with turn, the parameterisation function is determined using the same procedure described in section 3.4.2. Then, with a specified turn angle selected by the user, the system generates the motion according to the user's input.

The error in the parameter value has been found to be within the range of $-0.0339^\circ$ to $0.0298^\circ$ which is equivalent to a range of $-0.1365\%$ to $0.24041\%$ of the desired angle given by the user. The input-output relationship is depicted in figure 3.25. Also the quality of the resulting motions is close to the input motion clips as indicated by the quantitative measures shown in figure 3.26.

3.5.6 Summary of Synthesis Results

From the evaluations discussed in the previous subsections of results, it can be seen that the proposed parametric synthesis approach is able to successfully synthesise motions
3.5. Results and Evaluation

Figure 3.21: Sample of generating motions on specified slopes (up or down hill). Original clips (not shown in figure) 'UpSlopeWalk' with slope of 12.5° and 'DownSlopeWalk' with slope of 12.6°.
Chapter 3. Parametric Synthesis of Animation Clips

Figure 3.22: Output slope vs. Input slope of Walk on sloped floor examples.

Figure 3.23: Evaluation of result of controlling the slope of the floor. The two ends of each curve represent the measure value of input motions.
3.5. Results and Evaluation

Jump with turn-angle of $-29^\circ$  
Jump with turn-angle of $-61^\circ$

Figure 3.24: Jumping example where user controls the facing direction at the end of the jump. Original clips are ‘JumpInPlace’ (right in blue) ‘JumpTurn’ (left in green) with turn-angle of $-91.8^\circ$. 
Figure 3.25: Output speed vs. Input speed of Walk on sloped floor examples.

Figure 3.26: Evaluation of result of controlling the jumping turn angle. The two ends of each curve represent the measure value of input motions.
that satisfy arbitrary values of intuitive motion parameters which are specified by the user. The maximum error in the parameter values of the generated motions has been found to be less than 2% of the required values. More importantly, the quality of synthesised motions are close to the quality of input motion clips as indicated by the qualitative and quantitative evaluations.

It can be noticed that the shape of the jerk curve varies between the different examples. However, it can be concluded that the jerk values of generated motions are within the corresponding maximum values of input motions. This is our main concern in evaluating results which show that the quality of synthesised motions is close to inputs.

The dips in the jerk curves, of some of the examples, can be attributed to the rounding or quantisation error during the estimation of the new Key-event timing (Equation 3.7), as frame numbers are always of integer values. Since the jerk calculation depends on the time, which is calculated from the number of frames and the frame time, the error in frame numbers can either stretch or compress the cycle’s time period and hence reduce or increase the speed, acceleration and consequently, affect the jerk values.

To justify that, the jerk values of the same motions, of figure 3.17, have been calculated before the final time re-warping stage as shown in figure 3.27.a As depicted in the figure, the jerk values are behaving smoothly between the jerk values of the two input motions. The same experiment has been repeated for the motions of figure 3.23 as shown in figure 3.27.b which shows the same conclusion.

![Figure 3.27: Jerk measure calculated without the final time re-warping stage.](image)

The changes in the jerk values, due to the final time re-warping stage, are still within the range of the corresponding values of the input motions. Also, there are no noticeable visual artefacts in the resulting animation. However, it is worth investigating the
rounding error, that is causing these changes, and ways of overcoming it as discussed in the future work section 5.4.

Finally, it is important to mention that the key-event times, which represent the timing of the motion constraints, play a crucial role in the synchronised blending process as discussed in section 3.4.4. The next subsection shows the results of the proposed technique for automatic gait analysis which is used to automatically identify the key-event times.

### 3.5.7 Automatic Gait Analysis

Key-event times, which represent the timing of the motion constraints, play a crucial role in the synchronised blending process as discussed in section 3.4.4. Manual determination of these key-event times has been the common approach in previous research [77, 79, 55]. As a step forward on automating this stage, a simple -but effective- technique, presented in section 3.4.3, has been developed for automatic analysis of gait cycle. The proposed automatic gait analysis approach has been successfully applied on many gait motions on level and slope floors as well as in a straight and curved paths. Figure 3.28 depicts a comparison between the automatically detected key-event times and their corresponding manually detected ones. The maximum $\text{RMS}$ error in the automatic detection is found to be within 1.67 frames. The $\text{maximum}$ error in most of the cases is between 0 to 2 frames. In some extreme cases, the maximum error becomes 3 frames. However, with a closer look, it has been found that either the motion contains extra noise or the manually detected key-event times were not accurate enough.

The automatic gait analysis technique has been successful also with special/abnormal gait motions such as walking with limp as well as some corrupted walk cycles as shown in figure 3.29 with a maximum $\text{RMS}$ error of 1.4 frames ($\text{maximum}$ error of 2 frames). It worth mentioning that the technique has been able to detect the key-event times from a long sequence, even when the first frame is not one of the key-events, as shown in 'LimpWalk' example in figure 3.29. This has been achieved by defining a threshold (to avoid small local maxima and minima) as well as arbitrarily define the advanced foot (left or right).
3.5. Results and Evaluation

Figure 3.28: Automatically detected key-event times (in magenta) for normal gait motions versus the corresponding manually detected ones (in blue).
Figure 3.29: Automatically detected key—event times (in magenta) for Special/Upnormalgait motions versus the corresponding manually detected ones (in blue).

Although the technique was mainly developed for gait motions, it has been tested on motions with similar pattern such as Kick. The result is depicted in figure 3.30. In this example, it can be noted that the proposed technique is flexible in terms of the number of key-events that have to be determined, 3 in the kick case instead of 5 in the walk. This flexibility is promising for utilising the approach for similar motions such as reaching, waving.

Figure 3.30: Automatically detected key—event times (in magenta) for Kick’ motion versus the corresponding manually detected ones (in blue).

The next section shows some candidate applications where the proposed framework can be utilised.
3.6 Applications

This section introduces examples of some candidate applications where employing our parametric approach can be useful. The encapsulated and layered animation database presented in section 3.6.1 shows how the parametric approach can fit with other animation processing techniques for enriching a basic animation database. Section 3.6.2 describes the parametric synthesis of animation clips for scaled characters; a simple retargetting example. This helps animator to synthesis animation clips for different scaled characters with the same parametric approach.

3.6.1 Encapsulated and Layered Animation database

An example of a candidate application for the proposed parametric approach is the animation databases. Our parametric system can serve as an engine for enriching an animation database.

The layered animation database framework is depicted in figure 3.31. At the bottom layer \( L_r \), only real animation clips exists. The middle layer \( L_m \) contains modifiers that generate new motion clips based on one or more real animation clips from \( L_r \). Modifiers can be cascaded so that the output of one modifier can be used as an input for another to generate a desired clip. The top layer \( L_v \) represents the virtual database that is available to an animator. It includes real and modified animation clips.

The contents of the top layer \( L_v \) can be represented in terms of other clips and modifiers using a grammar like:
Chapter 3. Parametric Synthesis of Animation Clips

For the end-user, layers of the database are transparent and only the top layer $L_v$ is what the user interacting with. Hence, modifiers should be selected, designed, and implemented to be as simple and computationally efficient as possible. However, the exact action of the modifier can be controlled by some parameters or attributes.

Modifiers, situated in the middle layer $L_m$, can be classified according to the number of animation clips they are applied to. Hence, there are uni-modifiers, bi-modifiers, and multi-modifiers as explained below. The work presented in this chapter is classified...
into the bi-modifiers type. In chapter 4, examples of multi-modifiers are presented.

The uni-modifiers are mainly basic animation processing techniques. They are applied on individual animation clips. An example of this type of modifiers is the "Mirror" modifier. The most common form of mirroring is on the sagittal plane; the plane dividing the human body vertically into two symmetrical halves, right and left. Using only this form of mirroring, we can double the number of existing clips as shown in figure 3.32. Another example of uni-modifiers is rotating the global path of existing motion. This modifier can be controlled by the required rotation angle. Hence, from one clip, many other clips can be generated by rotating the given clip as shown in figure 3.32.

The bi-modifiers are using two input motions to generate a new one. Examples include speed, slope and path curvature control as depicted in figure 3.32. These modifiers are using the proposed parametric approach as discussed in section 3.4. They are controlled by high-level motion parameters according to the user needs. From two given motions, these modifiers can produce varieties of motions which can be impractical to capture. Moreover, they are generated on demand and need not to be saved permanently, if they are not required in the future, which reduces the storage requirements. Examples of multi-modifiers are discussed in chapter 4 as well as the continuous controlled modifiers.

The layered database is expandable, therefore user can expand the database through any of the following methods:

- Cascading one or more of existing modifiers and clips to generate new clips (as shown in the discussed grammar-like representation above).
- Defining new modifiers.
- Inserting more original clips.

Before leaving this section, it worth pointing out that some basic animation processing may produce interesting clips but introduce other problems. A typical example is the 'Mirror' modifier that can produce an interesting clip but on the other hand, it results in out-of-phase motion if used, with its original clip, as inputs to a bi-modifier. This problem is tackled in chapter 4 by introducing the 'Synchronised Mirror' technique.
Figure 3.32: Examples of enriching animation database contents
3.6. Applications

3.6.2 Scaled Avatars

Editing existing motion data is not limited to generating new motions. Editing can be used for other purposes such as generating or porting the same motion for other characters with different dimensions. This is known as the motion retargetting problem. Although this is not the focus of this research, it is useful to show how combining the proposed technique with the motion retargetting can be helpful [4]. Only scaled characters are considered in this section.

![Diagram showing different scaled characters]

Feet-floor contact violation

Feet skidding

Figure 3.33: Example of parametric synthesis of animation for scaled characters. ‘A’ is the original size character, ‘B’ is the scaled character (with original animation data), ‘C’ is the scaled character (with modified animation data).

What we need to do is to generalise the parametric synthesis approach to be used for characters that are scaled versions of the original character. Applying the data from original character on a scaled character will have the following problems (see figure 3.33):
1. Violation of feet-floor contact motion constraint. Due to scaling, feet will either penetrate the floor or fly above the floor.

2. Feet skidding due to the incorrect translation of the root joint.

The first problem, feet-floor contact, can be easily corrected by correcting the vertical translation of the root joint as follows:

\[ Y_s^F = \text{Offset} + Y_g^F; \quad (3.17) \]

where:

\( \text{Offset} = (S - 1) \times Y_g^1 \)

\( S = \text{Scale} \) (ratio between the two characters)

\( Y_g^1 = \text{Root vertical position at 1st frame of original animation} \)

\( F=1,...N; \quad N=\text{Number of frames}; \)

For the second problem, feet skidding, the translational degrees of freedom for the root joint need to be updated using the following formula:

\[ P_d^F = S \times L \times (F - 1) + P_d^0; \quad (3.18) \]

where:

\( S = \text{Scale} \)

\[ L = \frac{(P_s^F - P_s^P)}{(F_e - F_s)}; \]

\( F=2,...N; \quad N=\text{Number of frames}; \)

\( F_s, F_e \) are the starting and ending frame numbers respectively

For discussing the parametric synthesis approach for scaled characters, let us take the walking speed as an example of a parameter. Assume that we have a motion with speed \( V_g \) generated for the original character. If motion data applied directly to a character scaled by scale \( S \), the scaled character motion will have a speed \( V_s \) such as:

\[ V_s = S \times V_g \]
So, the procedure for synthesis of motion with speed $V_s$ for the scaled character is as follow:

1. Calculate speed $V_g$ for original character which corresponds to the required speed $V_s$ as:

   $$ V_g = \frac{V_s}{S}; $$

2. Synthesis motion (as discussed in section 3.4), using original character’s data, that satisfies the calculated speed $V_g$.


4. Use the generated and updated motion data to drive the scaled character.

As a result, the scaled character will be moving realistically with the desired speed $V_s$ as shown in figure 3.33 (the character denoted by ‘C’). The above procedure can help animators in generating basic multiple characters animation.

### 3.7 Summary

In this chapter, a high-level parametric synthesis approach has been introduced for generation of motion with desired motion parameter (e.g. speed and slope). The proposed approach is based on synchronised blending of motions with an automatically-obtained blending factor corresponding to the desired motion parameter value.

The introduced approach provides animators with more intuitive control over the animation synthesis process. It allows defining the desired motion by its intuitive parameters such as speed and slope, while the system synthesises realistic animation satisfying the desired parameter value.

Within the proposed approach, novel contributions presented in this chapter are:

- Solving the inverse interpolation problem to release user from trial and errors sessions [4]. The introduced solution is simple and take into account the real case of non-linear relationships between blending factor and given parameters (section 3.4.2).
Chapter 3. Parametric Synthesis of Animation Clips

- Automatic gait analysis [5] as a step forward on automating the annotation task (section 3.4.3).

- Applications for the proposed parametric approach; enriching the contents of animation database (section 3.6) and parametric animation synthesis for scaled characters (section 3.6.2).

- Results and evaluation are presented in section 3.5. For consistent evaluation, a number of quantitative measures has been introduced and adopted in order to quantitatively evaluate the resulting animation.

The next chapter extends the presented work for more practical situations where longer animation sequences are required and the given motion parameters are varying over time.
Chapter 4

Parametric Synthesis of Animation Sequences

Automatic generation of realistic long animation sequences has attracted the attention of researchers for the last few years [55, 59, 9, 90]. In this chapter, the initial framework discussed in chapter 3 is extended to allow for using this parametric synthesis approach to generate continuous animation sequences.

Section 4.1 restates the aim and motivation of this research. The extended problem of generating animation sequences is defined in section 4.2. Then the continuous parametric synthesis is presented in section 4.3 which covers the continuous parameterisation and introduces novel techniques for pre-processing input clips in order to be suitable for generating motion sequences [5]. Additional contributions such as handling the root blending and producing synchronised mirrored clips are also introduced.

Section 4.4 presents a practical case of employing the proposed parametric approach in synthesis of motion along arbitrary paths. It includes identifying the parameter and its representation, parameterisation, and synthesis. Section 4.5 presents another practical case for parametric synthesis of motion over arbitrary uneven terrain.

The extension of the approach to cover multi-parameters synthesis is discussed in section 4.6 where the enhanced parameterisation technique is presented.
Finally, before summarising in section 4.8, the results are shown in section 4.7 with our evaluation of the generated animation.

4.1 Aim and Motivation

The principal aim of this research is to provide animators with an intuitive control over the animation synthesis while preserving the level of realism from original motion clips as much as possible. It is also aimed at achieving a compromise between the parametric control, realism and quality of generated animation, and costs of generating these animations (e.g. Computational cost, time and effort, and the overall financial costs).

Our research is motivated by the realism achieved by motion capture data, limitations of editing and re-using these captured data, the animator’s need for intuitive parametric control, and the increasing demand of generating long and continuous human animation sequences.

In chapter 3, the parametric approach for synthesis of human animation clips is discussed. The focus of that chapter was on generating short animation clips according to a constant user-specified motion parameter.

This chapter extends the focus to the more practical demand of parametric synthesis of human animation sequences. This includes synthesis of continuous and long animation sequences according to user-specified time-varying parameters and simultaneous multi-parameters synthesis.

First, the extended problem is defined in the next section. Then, our approach and contributions are presented in the subsequent sections.

4.2 Extended Problem Definition

Generating animation sequences through editing short animation clips is one of the central topics in computer animation research. It is not a straightforward extension of the process of generating new short clips from existing data as discussed in this chapter.
In addition to the problems of motion annotation, synchronisation, and inverse interpolation, defined in section 3.2, there are some other issues to be addressed in order to generate continuous animation sequences from short clips as described below. Our focus in this research is the animation of cyclic motion which represents a major sector of human motions and daily activities.

In chapter 3, the user-input was assumed to be a constant parameter value $P_{iu}$. Practically, the user will need to generate motion according to arbitrary parameter values. For long or continuous animation sequences, the user input is considered as time-varying parameter values $P_{iu}(t)$. Hence, equation 3.3 can be written as:

$$M_3(t) = \text{SyncBlend}(M_1(t), M_2(t), X_{iu}(t))$$  \hspace{1cm} (4.1)$$

where $X_{iu}(t) = f(P_{iu}(t))$; is the blend factor corresponding to a user-specified parameter $P_{iu}(t)$.

For longer animation sequences, the user-input is expected to be continuous or at least longer than the length of original short clips. Hence, the relation $X_{iu}(t) = f(P_{iu}(t))$ should be defined for $0 \leq t \leq L_s$, where $L_s$ is the length of desired animation sequence, although input clips exist only for $0 \leq t \leq L_n$, where $L_n$ is the length of input clips. This continuous parameterisation is discussed in section 4.3.2.

Also, input clips need to be smoothly and realistically extended to cover the period $0 \leq t \leq L_s$. This problem is tackled as discussed in section 4.3.1 [5].

The following section describes the parametric synthesis of continuous animation sequences from short animation clips. It identifies the main difficulties and shows our proposed solution.

4.3 Continuous Parametric Synthesis

In this section, the focus is on the parametric synthesis of continuous animation sequences based on short animation clips (single cycle each). The parametric approach proposed in chapter 3, with constant user-input $P_{iu}$, is extended to the continuous case with time-varying user input $P_{iu}(t)$.
To use equation 4.1, the function $f$ that maps user-input $P_{iu}(t)$ to blending factor $X_{iu}(t)$ is to be identified such that:

$$X_{iu}(t) = f(P_{iu}(t)); \quad \text{for} \quad 0 \leq t \leq L_s$$

where $L_s$ is the length of the required animation sequence.

However, the input motion clips usually have limited lengths $L_n < L_s$. Section 4.3.2 describes the continuous parameterisation so that the function $f$ is defined over the period of the required animation sequence.

The next section presents our novel techniques for extending the input animation clips.

### 4.3.1 Extending Input Motion Clips

Although motion captured animation can produce realistic animation, due to some limitations, certain motion sequences cannot be completely captured. Especially for cyclic movements such as walk and run. Motion captured animation can be used to build a database of basic motions which can be used for character animation. The challenge is to find suitable tools and techniques to edit, modify, and re-use these data.

Cyclic motions represent a major sector of human movements. This includes many of our daily activities such as walking, running, and waving. Some other motions, that have similar pattern, can be considered as cyclic motions especially when they are repeated (e.g. kicking, jumping and reaching). A trivial method for generating longer sequence of cyclic movement animation would be to concatenate motion cycle sequences.

Concatenating cycles is a trivial task only when the available cycles are complete and perfect (i.e. the postures, at the start and the end of the cycle, are perfectly matched). However, this is rarely the case in human motions. Due to the nature of human movement, human cyclic movements are usually not perfectly cyclic.

Matching the cycle boundaries, the process known as the *cyclicification* [84], is one of the essential stages in generating a stream of animation of cyclic human movement. If
4.3. Continuous Parametric Synthesis

the motion cycle boundaries do not match, which is usually the case in the real human movements, the generated sequence will possess noticeable artefacts and flickers at these boundaries regardless of the realism of the available motion cycle. In this section, we first review the previous research that addresses this problem. Then present our novel techniques [5] for matching cycle boundaries (cyclification).

The cyclification problem has received only limited investigation. Within the context of motion captured animation and editing, Sudarsky et. al. [88] introduced a cyclification operator. They represent motion curves as splines. Given a periodic curve, the cyclification operator is the one that generates a new spline which has the same values at the cycle period and its multiples. If the given curve is not accurately periodic, another process is needed such as blending. In [38], Golam et. al. also used blending to cyclify their unit cycle, which consisted of two or more actual motion cycles. In their analysis, the upper and lower body halves are decoupled and processed independently. This introduces a phase difference, between the upper and lower body movement, that needs correction. The above techniques have in common the assumption that there are at least two complete motion cycles available. Although they are similar to our first approach, the difference is that we do not require a second complete cycle to be available, only a small part of it either at the beginning or end. Also, our unit cycle is the only complete cycle available and body parts are not decoupled.

The cyclification problem has also been addressed within motion transition research. Rose et. al. [78] presented a cyclification method based on fitting a least squares cyclic B-spline to a modified motion curve. The modified motion curve is constructed to minimise the difference of position, speed, and acceleration between the cycle boundaries. However, using B-spline fitting may lead to violation of joint limits, and kinematic constraints as mentioned in [88]. These constraints need to be corrected later using IK algorithms and enforcing the joint limits. Gleicher et al. [37] used a similar approach based on finding a common pose between similar frames from different motion clips. The motion clips are then modified using displacement maps to start and/or end with the common pose. This results in violating some motion constraints at the modified parts. They concentrated on correcting the foot-plant constraints only and employed a correction algorithm from [56]. In our techniques, sections 4.3.1.1 and 4.3.1.2, the
original motions are synchronised before blending to ensure that timing and important events are aligned. This automatically minimises the violation of the motion constraints to a great extent (provided that they are correctly defined in the original motions).

The work of Silva et. al. [84] is most relevant to our work in the cyclification problem. Their curve cyclification algorithm is based on using a windowing technique referred to as 'LCT' (Lapped Cosine Transform) to accomplish time warping whilst preserving the frequency content. When applying this algorithm on the motion of articulated figures, some considerations should be taken into account. First, the large number of segments and connecting joints in the articulated figure hierarchy and the multiple degrees of freedom (DOFs) at each joint result in large amounts of data that should be processed. Second, and more importantly, synchronisation between body parts in the human motion and dependencies between different joints, or groups of joints, should be preserved. Also, in their algorithm, selecting the window size is critical as it should be equal to the cycle period. If it is larger or smaller than the cycle period, it will result in discontinuities and/or noise.

In the rest of this section, our proposed techniques are presented. These techniques employ simple and fast animation processing techniques such as blend, mirror, and reverse of motion cycle parts. The first technique, described in section 4.3.1.1, shows the cyclification guided by part of the additional motion cycle and assumes that there is enough data (i.e. motion frames) to correct the cycle. The other techniques described in section 4.3.1.2 are based on the availability of only one motion cycle. In such cases, where only one motion cycle is provided, the techniques benefit from cycle characteristics in order to achieve the cyclification.

4.3.1.1 Cyclification guided by part of an additional cycle

The first and simplest technique treats the cycle using a part of another existing cycle by employing a synchronised blending technique. Our motivation in this technique is that in many cases, when we capture data, we usually capture more than one cycle of the cyclic motion (depending on the capturing volume), before or after the cycle we are focusing on. Due to some measurement noise and missing data, captured movement
data may be slightly corrupted. The extra data, or some parts of it, can be utilised carefully to modify and match cycle boundaries. It can also be used in correcting some other corrupted motion constraint (e.g. feet-floor touch/lock) near the edges, which could avoid repeating the motion capture process.

In our system, the unit cycle consists of only one actual motion cycle. So, any other part of a cycle, preceding or following our cycle, is considered as additional data that can be utilised in this technique. It is worth mentioning that our technique does not require a complete additional cycle. Just part of it (down to only one segment of the cycle time, where the segment is the time between two consecutive key-events) is enough. The technique works as follows:

A motion cycle $M_1[1 : m]$, and an additional part of another cycle $M_2^p[1 : n]$ are used, where $m$ and $n$ are the number of segments of $M_1$ and $M_2^p$ respectively and $m > n$. They are divided into segments by the key event times; the times of important events and postures during the motion which imply motion constraints. The following procedure is applied to match the cycle boundaries:

1. From $M_1$, extract $M_1^f$ the corresponding part for $M_2^p$:
   
   
   $M_1^f = M_1[m - n + 1 : m]$; if $M_2^p$ is preceding $M_1$
   
   Or
   
   $M_1^f = M_1[1 : n]$; if $M_2^p$ is following $M_1$

2. Synchronise and blend the corresponding segments:
   
   $M_1^f = \text{SyncBlend}(M_1^f, M_2^p, X)$
   
   where $X$ is the blending vector such as
   
   $X[1 : \text{length}(M_1^f)] = [1 : \frac{1}{\text{length}(M_1^f)} : 0]$

3. Construct the corrected cycle:
   
   $CM_1 = \text{Append}(M_1[1 : m - n + 1], M_1^f)$; if $M_2^p$ is preceding $M_1$
   
   If $pMC$ is part of the following cycle, then:
   
   $CM_1 = \text{Append}(M_1^f, M_1[n : m])$; if $M_2^p$ is following $M_1$

The result from applying the above procedure is a motion cycle that has similar start
and end postures. This ensures that we can smoothly concatenate this cycle to itself, repeatedly, to construct a longer animation sequence.

The above cyclification technique assumes that we have enough data to modify our cycle. It also assumes that the extra data is directly preceding or following the complete cycle of data we have. However, if this extra data is not available (e.g., we do not have access to the raw captured data, the data is corrupted or not preceding/following our cycle) this approach may fail. This is where the other proposed techniques can help as shown in the next section.

4.3.1.2 Self Cyclification

The self cyclification techniques work on a single cycle only and assume that there is no extra data available. These techniques utilise animation processing techniques to manipulate different parts of the given cycle in order to achieve similar start and end frames. As there is no additional animation data available in this case, the techniques try to benefit from the cycle characteristics of human movement. In this section, our techniques are described on two major families of cyclic human movements as presented in the following subsections.

"Mirrored Half-cycle" category

The observed characteristic of the motion cycle of this category is that the second half of the cycle has a similar pattern to the mirror of the first half of the same cycle. Typical examples of this category of motion are 'Walk' and 'Run' (see figure 4.1 for 'Walk' example).

Based on these characteristics, simple and fast animation processing techniques, such as 'Mirror' and 'Synchronised Blending' (presented in section 3.4.1), are utilised to modify the cycle in order to match its boundaries for seamless appending or transition. Figure 4.2 presents the proposed cyclification algorithm for this category of motion.

The diagram in figure 4.3 explains the algorithm on a walking cycle starting with the left foot touching the floor. Given a motion cycle \( M \), its mirror \( M_r \) is calculated and
4.3. Continuous Parametric Synthesis

Figure 4.1: Example of 'Mirrored Half-Cycle' motion category (Walk)

the two halves (of both motions) are extracted. The first half of the corrected cycle ‘$M_1$’ is constructed by blending between the second and first halves of the mirrored cycle and the original cycles respectively. Similarly, the second half of the corrected cycle ‘$M_2$’ is constructed by blending between the second and first halves of the original cycle and its mirror respectively. Then both corrected halves are appended together to construct the corrected cycle ‘$M_{cm}$’ with matched boundaries.

"Reversed Half-cycle" Category

In this category, it has been observed that the second half of the cycle is similar to the reverse of the first half of the same cycle. ‘Kick’ and ‘jump-turn’ are some examples of this category of motions (see figure 4.4 for kicking example).

Knowing these characteristics, a similar approach is developed that applies ‘Reverse’ and ‘Synchronised Blending’ techniques to update and match the cycle boundaries. The cyclification is described by the algorithm shown in figure 4.5.

The synchronised blending first aligns the corresponding motion segments, divided by the key-event times, together using time-warping (as discussed in section 3.4.4) to maintain motion constraints. Next, different parts of the given cycle, and its reverse, are manipulated together to achieve matched cycle boundaries.
Chapter 4. Parametric Synthesis of Animation Sequences

<table>
<thead>
<tr>
<th>Input:</th>
<th>Output:</th>
</tr>
</thead>
<tbody>
<tr>
<td>* One motion cycle ‘M’.</td>
<td>Corrected motion cycle with its boundaries matched together ‘M_{cm}'.</td>
</tr>
<tr>
<td>* The corresponding Key Event Times ‘KETS’.</td>
<td></td>
</tr>
<tr>
<td>where CycleKET = Number of KETs/Cycle</td>
<td></td>
</tr>
<tr>
<td>Procedure: 1. Get the mirror of the provided cycle (around its forward direction).</td>
<td></td>
</tr>
<tr>
<td>[ M_r = \text{Mirror}(M); ] [ rKETS = \text{Mirror}(KETS); ]</td>
<td></td>
</tr>
<tr>
<td>2. Define the different parts of the given cycle (and its mirror) that is going to be manipulated:</td>
<td></td>
</tr>
<tr>
<td>- 1^{st} half of ( M )</td>
<td></td>
</tr>
<tr>
<td>[ A = M[KETS(1) : KETS(\text{CycleKET}/2)]; ]</td>
<td></td>
</tr>
<tr>
<td>- 2^{nd} half of ( M )</td>
<td></td>
</tr>
<tr>
<td>[ B = M[KETS(\text{CycleKET}/2) : KETS(\text{CycleKET})]; ]</td>
<td></td>
</tr>
<tr>
<td>3. Construct the corrected 1st half 'M1' as follows:</td>
<td></td>
</tr>
<tr>
<td>- Prepare blending vector so that ( X = 1 \rightarrow 0 ) over ( B_r )</td>
<td></td>
</tr>
<tr>
<td>[ X[1 : \text{length}(B_r)] = [1 : \frac{1}{\text{length}(B_r)} : 0]; ]</td>
<td></td>
</tr>
<tr>
<td>- SyncBlend; starting by ( B_r ) &amp; ending with ( A )</td>
<td></td>
</tr>
<tr>
<td>( M_1 = \text{SyncBlend}(B_r, A, X); )</td>
<td></td>
</tr>
<tr>
<td>4. Construct the corrected 2^{nd} half 'M2' as follows:</td>
<td></td>
</tr>
<tr>
<td>- Prepare blending vector so that ( X = 1 \rightarrow 0 ) over ( B )</td>
<td></td>
</tr>
<tr>
<td>[ X[1 : \text{length}(B)] = [1 : \frac{1}{\text{length}(B)} : 0]; ]</td>
<td></td>
</tr>
<tr>
<td>- SyncBlend; starting by ( B ) &amp; ending with ( A_r )</td>
<td></td>
</tr>
<tr>
<td>( M_2 = \text{SyncBlend}(B, A_r, X); )</td>
<td></td>
</tr>
<tr>
<td>5. Construct the corrected cycle by concatenating both halves ( M_1 ) and ( M_2 ).</td>
<td></td>
</tr>
<tr>
<td>( M_{cm} = \text{Append}(M_1, M_2); )</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.2: Self Cyclification algorithm for ‘Mirrored Half-cycle’ motion category
4.3. Continuous Parametric Synthesis

$L$: The Left foot in front and touched the floor

$R$: The Right foot in front and touched the floor

$L_r$: The same as $L$ but for the mirrored motion cycle.

$R_r$: The same as $R$ but for the mirrored motion cycle.

Figure 4.3: This diagram explains the self-cyclification algorithm for 'Mirrored Half-cycle' category

Figure 4.4: Example of 'Reversed Half-cycle' motion category (Kicking)
**Input:**
* One motion cycle 'M'
* The corresponding Key Event Times 'KETS'

**Output:**
Corrected motion cycle with its boundaries matched together 'M_c'

**Procedure:**
1. Get the 1st half of the provided cycle.
   \[ A = M[1 : KETS(\text{CycleKET}/2)]; \]
   where \( \text{CycleKET} \) = No. of Key Event Times per Cycle.
2. Get the 2nd half of the provided cycle.
   \[ B = M[KETS(\text{CycleKET}/2) : KETS(\text{CycleKET})]; \]
3. Get the reverse of 1st half
   \[ A_r = \text{Reverse}(A, \text{Root}); \]
4. Prepare the blending factor.
   \[ X[1 : \text{length}(B)] = [1 : \frac{1}{\text{length}(B)} : 0]; \]
   \( (X = 1, 1 - \frac{1}{\text{length}(B)}, ..., 0 \text{ over the original 2nd half } B) \)
5. Blend original 2nd half & aligned, reversed 1st half.
   \[ B_c = \text{SyncBlend}(B, A_r, X); \]
6. Concatenate both halves; Original 1st half, and matched 2nd half
   \[ M_c = \text{Append}(A, B_c); \]

Figure 4.5: Self Cyclification algorithm for 'Reversed Half-cycle' motion category
As mentioned in previous research, in dealing with the articulated figure, the root joint which has 6 DOFs usually needs special attention as it represents the overall translation of the whole hierarchy. Depending on many factors, including the motion type and the required cyclification, the root joint information can either receive the same processing as the rest of the joints or not. This option is implemented in the ‘Reverse’ module as indicated by ‘Root’ in step 3 of the algorithm. For example, in ‘Kick’ motion, the cyclification is required to maintain the same root position and orientation at both ends of the cycle. To achieve that, the root joint is processed normally with the rest of the joints. However, this condition is not required in the ‘Jump-Turn’ as the initial and final root positions and orientations are expected to be different. So, the root joint just follows the original cycle.

Summary of Cyclification techniques

The main advantages of our proposed cyclification techniques are that they are applied directly to the animation data (joints DOFs) without fitting (e.g. spline) or any time-frequency transforms, which reduces the processing time. Instead, these techniques benefit from some observed characteristics of motion cycles for different families of motions and employ these observations with simple animation processing algorithms to match the cycle boundaries. The proposed techniques preserve many aspects of animation such as joints limits and constraints because they are guided by original motions and observing their constraints (through key events).

4.3.2 Continuous Parameterisation

In section 3.4.2, the user-input (i.e. the motion parameter), was assumed to be a constant value $P_{iu}$. For continuous synthesis, the user-input may have time-varying values $P_{iu}(t)$. It is required to find out the function $f(P_{iu}(t))$ that provides the corresponding blending factor $X(t)$ for $0 \leq t \leq L_s$, where $L_s$ is the length of desired animation sequence.

As a start, let us identify the function $f$ over the period $0 \leq t \leq L_n$, where $L_n$ is the length of input clips. Some parameters are calculated over the entire cycle such
Chapter 4. Parametric Synthesis of Animation Sequences

as speed, slope, jump-turn angle, and kick position. Other parameters are calculated instantaneously at each frame such as the direction of motion (section 4.4). Hence, there are two cases to be distinguished based on how the motion parameter is calculated.

For the first case, where the motion parameter is calculated over the entire cycle, the function \( f \) that maps input parameter \( P \) to blending factor \( X \) is to be determined as discussed in section 3.4.2. Then, at each time \( t_i \), the required blending factor is calculated using the determined function as follows:

\[
X(t_i) = f(P(t_i)); \quad \text{for } 0 \leq t_i \leq L_n
\]

Now, this should be extended to the period \( 0 < t < L_s \). Using the same function \( f \), we can determine the blending factor values for the entire period of the animation sequence as follows:

\[
X(t - t_{c_{j-1}}) = f(P(t - t_{c_{j-1}})); \quad \text{for } 0 \leq t - t_{c_{j-1}} \leq L_n
\]

where \( t_{c_{j-1}} \) is the end time of cycle \( j - 1 \) (i.e. the previous cycle).

Figure 4.6 shows an example of this mapping. The continuity of the blending factor values across cycle boundaries is maintained as \( \lim_{t_i \to L_n} X(t_{c_{j-1}} + t_i) = \lim_{t_i \to 0} X(t_{c_j} + t_i) \).

However, to achieve smooth and realistic concatenation of generated cycles, the input cycles should be cyclified as discussed in section 4.3.1.

In the second case, the parameters are calculated at each frame in the animation sequence. Therefore, each individual frame in the sequence will have its own blending factor for the given parameter values. This adds an extra factor which is the time (or frame number). Hence, the mapping function \( f \) becomes a group of functions \( f_{F_i} \) for \( 1 \leq i \leq L_n \).

To determine this group of functions \( f_{F_i} \), the same procedure discussed in section 3.4.2 is repeated. The main difference is that for each individual value of blending factor, the parameter value is calculated at each frame as follows:
4.3. Continuous Parametric Synthesis

\[ X = f(P(t)) \]

\[
\lim_{t_i \to L_n} X(t_{cj} + t_i) = \lim_{t_i \to 0} X(t_{cj} + t_i)
\]

Figure 4.6: Continuous parameterisation

\[
P_i = \rho(\text{SyncBlend}(M_1, M_2, X_i)); \quad \text{for } 1 \leq j \leq \text{NumberOfFrames}
\]

where \( \rho \) is the function that calculate the parameter value for the resulting motion clip.

This parameterisation results in a new curve (instead of a single value) for each single value of the blending factor. Then the calculated data is rearranged so that for each individual frame, we have a curve mapping the parameter and the blending factor values. These curves are approximated and their approximation functions represent the required group of functions \( f_{p_i} \) that can be used in the synthesis process. A practical example of this type of parameterisation is depicted in the path control module described in section 4.4

4.3.3 Improved Root Trajectory Blending

It is important to handle the root joint carefully because it influences the whole skeleton. In this section, the limitations of the existing ordinary blending of root trajectory are identified. We also introduce enhancements to the blending process in order to overcome the identified limitations.

The identified limitations of existing root trajectory blending are summarised into the following three points (details are discussed throughout the section):
• Blending of global root positions introduces incorrect and unrealistic root position trajectory especially with time-varying blending factors.

• The magnitude of the blended root translation is affected by the increase of the difference in input motion directions. This introduces artifacts such as the foot sliding on the floor.

• The sudden flipping of the root trajectory when the difference in input motion directions changes around the 180° (i.e. from $\theta < 180^\circ$ to $\theta > 180^\circ$ or vise-versa). The result is unrealistic motion due to the incorrect root trajectory.

Each of the above identified points is discussed in details through the remainder of this section along with our proposed enhancements.

**Incremental root translation**

The direct blend of the absolute root positions can produce unrealistic artefacts in the resulting motion. To explain this let us have a look at a simple example. If we are given two walking motions with slow and fast speeds $V_s$ and $V_f$ respectively. The objective is then to generate a motion that starts with speed $V_s$ and end up with speed $V_f$ which are the two extreme speeds of the given motions. This means starting with a blending factor $X = 1$ and ending with $X = 0$ which are again the extreme values of the blending factor corresponding to the extreme speeds of the given motions. Let us assume that the starting root positions $P_{s0}$ and $P_{f0}$, of the slow and fast motions respectively, are aligned to start at the same absolute position. When blending the absolute root positions, the result will start at the absolute position of $P_{s0}$ and end at the absolute position of $P_{fn}$ where $n$ is the number of frames involved in blending. Intuitively, this may seem correct. However, with a closer look, we note that for this motion to be realistic, the speed should have exceeded the fastest input speed $V_f$ at certain points during the animation. As the blending factor ranges only between 0 and 1, the resulting motion must have speed $V$ such that $V_s \leq V \leq V_f$. Hence, the result is incorrect root movements which causes the feet to slide on the floor due to the incorrect speed of the root.
The sliding effect discussed in the above example can be attributed to the fact that the blend of absolute root positions does not incrementally build on the previous generated frames. Another example of such problem is also emphasised in figure 4.7 where the root trajectory, projected on the floor plan, is shown for straight and curved input motions as well as the output (in blue) of blending with random values of blending factor. We can see how the output jumps between the input trajectories. Although it obeys the provided blending factor $X$, it does not take into account the absolute location of the last generated frame.

![Figure 4.7: Absolute vs. Relative Root Blending. The figure shows the root-positions trajectory, projected on the floor plane, for random values of blending factor.](image)
To avoid this problem, we need to take into account the relations between frames and build upon previous root positions. At each frame, the incremental root translation, between current and previous frames, of input motions are blended and the resulting translation is added to the absolute root position of last generated frame. So, the generated position is built on top of the previous frame’s position and not just an absolute positions. The improvement of using the relative root blend is depicted (in red) in figure 4.7.

**Scaled Root translation**

The incremental root trajectory blend performs well with motions in the same directions. However, there are situations where the input motions are not moving in the same direction. An example would be the straight and curved path motions (figure 4.7). As the difference in direction increases, the distortion to the generated root positions increases and results in greater distortion to the animation. The motion can easily violate its constraints, resulting in feet sliding on the floor. Also the motion parameters, especially those which depend on root translation (such as speed), will be affected. For example, if we blend between a straight and curved path motions with similar speeds, the resulting motion is likely to have less speed than the originals. This is due to the fact that the blended root translation are reduced with the increase of difference in directions between original motions. This is not too surprising as it is one of the inherited properties of vector analysis as shown in figure 4.8. In figure 4.8, vectors $\mathbf{A}$ and $\mathbf{B}$ represent the change in the root position of the first and second input motions respectively. The blending result is represented by vector $\mathbf{C}'$ which depicts the equation:

$$
\mathbf{C}' = x \mathbf{A} + (1 - x) \mathbf{B}
$$

(4.4)

where $x$ is the blending factor.

From the figure, it can be noted that with the increase of the angle $\theta$ between the original vectors $\mathbf{A}$ and $\mathbf{B}$, the magnitude of the resulting vector $\mathbf{C}'$ decreases. This means a reduction of the distance travelled by the root which in turn reduces the resulting speed and produce the unwanted feet sliding. In order to overcome this
4.3. Continuous Parametric Synthesis

In order to maintain the distance travelled by the resulting root and the correct direction and orientation, we divided the process into two steps:

1. Apply the vector analysis to obtain the correct direction represented by the unit vector $\vec{u}$.

   \[ \vec{C} = x * \vec{A} + (1 - x) * \vec{B} \]

   \[ \vec{u} = \frac{\vec{C}}{|\vec{C}|} \]  \hspace{1cm} (4.5)

2. Use the magnitude of the original vectors to maintain the magnitude $G$ of the resulting vector $\vec{C}_s$.

   \[ G = |\vec{C}_s| = x * |\vec{A}| + (1 - x) * |\vec{B}| \]  \hspace{1cm} (4.6)

Then, the incremental root translation is represented by the vector $\vec{C}_s$ such that:

\[ \vec{C}_s = G \cdot \vec{u} \]  \hspace{1cm} (4.7)

where $G$ and $\vec{u}$ are defined in equations 4.6 and 4.5.

**Improved Incremental Root Trajectory Blend**

The scaled translation of the root is enough as long as the difference in input motion directions is less than $180^\circ$. When this difference in direction passes the limit of $180^\circ$, the resulting direction of blended root trajectory suddenly flips over to the opposite direction as shown in the example of figure 4.9.

The sudden flip in root position trajectory is inherited from vector analysis as depicted in figure 4.10. The resulting vector is always in the side of the smallest angle between the two input vectors. As the smallest angle changes, which occurs just around the $180^\circ$, the resulting vector suddenly changes to the opposite direction.
Figure 4.8: Explanation of the Root Blending problem inherited from Vector Analysis.
4.3. Continuous Parametric Synthesis

Figure 4.9: The synthesised root trajectory (middle) is suddenly flipped to the opposite direction after the difference in direction of input motions exceeds 180°.

Figure 4.10: Explanation of the Root Blending problem when the difference in input directions exceeds 180°. The root trajectory flips when moving from frame $j$ (left) to frame $i$ (right) with $i > j$ and $\theta_j < 180^\circ < \theta_i$. 
In addition, the sudden flip in root-position trajectory is also attributed to the fact that a single orientation can be achieved by two different rotations (i.e., two different representations for the same orientation). Rotation of a 100° will lead to the same orientation of a rotation by −260°.

To overcome the problems explained above, we introduce the incremental blend of both trajectories. The idea of the incremental blend is to build the resulting root trajectories by blending the incremental changes in input root trajectories. This incremental approach allows us to prevent the sudden flipping, regardless of the global root position and orientation of the input motions.

The incremental blending of the root orientation is achieved on the following steps:

- The initial root orientation, at frame 1, is calculated by direct blending of root orientation of input motions (using quaternion interpolation).

\[ R_1 = \alpha R_{M_1}^1 + (1 - \alpha) R_{M_2}^1 \]

where \( \alpha \) is the blending factor.

- For each frame \( i \), calculate the accumulated root orientation \( R_i \) as follows:

\[ R_i = R_{i-1} + \Delta R_i \]

where \( \Delta R_i = \alpha \Delta R_{M_1}^i + (1 - \alpha) \Delta R_{M_2}^i \)

and \( \Delta R_{M_j}^i = R_{M_j}^{i-1} - R_{M_j}^i \), for \( j = \{1, 2\} \)

and \( \alpha \) is the blending factor.

It is worth mentioning that the operations on orientation, especially interpolation, are all done using quaternion.

In a similar procedure, the incremental blending of the root translation is achieved on the following steps:

- The initial root positions, at frames 1 and 2, are calculated by direct blending of root positions of input motions such that:
4.3. Continuous Parametric Synthesis

\[ P_i = \alpha P_{M1}^i + (1 - \alpha) P_{M2}^i \quad \text{for } i = \{1, 2\} \]

where \( \alpha \) is the blending factor.

- For each frame \( i > 2 \),
  
  - Calculate the magnitude of blended incremental root translation.
    \[ \Delta T_i = \alpha \Delta T_{1i}^i + (1 - \alpha) \Delta T_{2i}^i \]
    where \( \Delta T_j^i = \sqrt{||X_j^i - X_j^{i-1}||^2} \); 
    for \( j = \{1, 2\} \)
  
  - Calculate the change in direction of root position trajectory for each input motion:
    \[ \Delta \theta_j^i = \theta_j^{i-1} - \theta_j^i \]
    \[ \Delta \theta_2^i = \theta_2^{i-1} - \theta_2^i \]
    where \( \theta_j^i = \tan^{-1}\left( \frac{z_j^i - z_j^{i-1}}{x_j^i - x_j^{i-1}} \right) \) and \( j = \{1, 2\} \)
  
  - Check for sudden change in direction that may results from different representations of same orientation (e.g. \( \pm180^\circ \)) and compensate it:
    \[ \text{if } (\Delta \theta_j^i > 180^\circ) \quad \Delta \theta_j^i = \Delta \theta_j^i - 360^\circ \]
    \[ \text{if } (\Delta \theta_j^i < -180^\circ) \quad \Delta \theta_j^i = \Delta \theta_j^i + 360^\circ \]
    for \( j = \{1, 2\} \)
  
  - Calculate the blended incremental change in direction of the root trajectory:
    \[ \Delta \theta^i = \alpha \Delta \theta_1^i + (1 - \alpha) \Delta \theta_2^i \]
  
  - Calculate the global direction of the incremental root translation, relative to the global direction at last frame \( i - 1 \):
    \[ \theta^i = \theta^{i-1} + \Delta \theta^i \]
  
  - Calculate the global root position vector \( \vec{P}^i \) by accumulating the incremental translation vector \( \Delta \vec{P}^i \) to the global position vector \( \vec{P}^{i-1} \) of the last frame \( i - 1 \) as follows:
    \[ \vec{P}^i = \vec{P}^{i-1} + \Delta \vec{P}^i(\Delta T^i, \theta^i) \]

Figure 4.11 presents the result of applying the improved root trajectory blending on the example of figure 4.9. We can see that the improved blending technique has
Figure 4.11: The synthesised root trajectory (middle) has no sudden flipping any more after using the improved blending technique.

successfully avoided the problem of sudden flip of the root trajectory. Results and evaluation are presented in section 4.7.6.

Before leaving this section, it is worth noting that to the best of our knowledge, the only related work to this section is Kovar et al [54] which was developed in parallel with ours. Their proposed solution is to use a 2D transformation in the XZ-plane, the floor plane, to match orientation of inputs at each frame. The transformed frames are blended to produce the resulting frame. Also, the 2D transformations are blended to produce the 2D transformation required to return the resulting frame to its global direction and position. It is similar to our technique. However, it can be noticed that the rotation part of the 2D transformation can be done in both directions, clockwise and counter-clockwise. There is no criteria for deciding which rotation direction will produce the appropriate results in different situations. If we select the rotation to be within the smallest (or largest) angle between the directions of input motions, we will end up with the same problem. This is because the smallest (or largest) angle will change suddenly around the 180° of difference in input directions. In our solution,
4.3. Continuous Parametric Synthesis

blending the changes of orientation of input motions reduces the chance of facing a large difference in direction between inputs. Also, the history of one frame provides a clue for the change of direction of each input motion and allows for preventing this sudden flip in root trajectory. In terms of storage, Kovar et. al. [54] store the 2D transformation between every pair of corresponding frames for every pair of motions. This requires a storage of order \( O(\frac{m!}{2}(m - 1)n) \) where \( m \) is the number of motion clips and \( n \) is the number of corresponding frame pairs. In our case, the storage requirements are reduced to order \( O(mn) \) as we can save only the changes of direction for each individual motion. This also simplifies the process of inserting new clips to the database.

4.3.4 Synchronised Mirror

The most common forms of mirroring in existing human animation packages [87, 49] is the mirroring on the sagittal plane defined at the first frame; the plane dividing the human body vertically into two symmetrical halves, right and left. Using this form of mirror can double the number of clips in an animation database. However, one principal problem is that the mirrored clips are out of phase with their original clips. For example, if the original clip is a ‘walk’ in a clockwise curved path, its mirror will be a ‘walk’ in a counter-clockwise curved path. However, the problem in this example is that if the original clip starts with the left foot in front, its mirror will start with the right foot in front (instead of the left foot). This means that the key events are out of phase and blending will produce invalid motion as shown in figure 4.12.

In this section, the ‘Mirror’ modifier (mentioned in section 3.6.1) is extended to overcome this limitation and utilise existing motion clips. Our novel ‘SyncMirror’ modifier achieves the goal of mirroring while producing motion that is in-phase with the original motion.

Given a curved-path motion cycle, say clockwise curvature \( M_{cw} \), its synchronised mirror cycle \( M_{ccw} \) can be obtained by applying the algorithm in figure 4.13, which is also explained by the diagram in figure 4.14. Instead of mirroring the given cycle only (as the common mirror modifier), we mirror a sequence of two cycles constructed from the given cycle. Having two cycles allows for selecting the appropriate start of our desired
Figure 4.12: Blending original (left) and ordinary mirrored clip (right) produces invalid motion (middle) as the mirrored clip is out-of-phase with the original clip.

mirrored cycle so that it is in-phase with the original cycle. This desired mirrored cycle is then translated and rotated to align with original cycle as shown in figure 4.14. Results and evaluation are shown is section 4.7.5.

The above algorithm assumes that the given cycle is perfect (i.e. cycle boundaries are matched) so a smooth and realistic sequence of two cycles can be automatically constructed. If the given cycle is not perfect, we can apply the cyclification techniques presented in section 4.3.1.

4.4 Path Control

In this section, parametric control of motion along an arbitrary path is presented. The method is based on the proposed parametric synthesis framework and depicts a practical example of its application. The objective is to provide the user with an intuitive parametric control for synthesis of motion that follows a given arbitrary path. Motion path and direction are among the important properties of animation. There is a common interest in many animation applications (such as virtual navigation and games) for editing path and direction of motion.
### 4.4. Path Control

**Input:**
- One motion cycle ‘\(M_{cw}\)’ on curved-path (Say, Clock-Wise).
- List of key-event times \([T_{M_{cw}}]\).

(Cycle is assumed to have matched boundaries. If not, cyclify as discussed in section 4.3.1).

**Output:**
- Mirrored and Synchronised motion cycle ‘\(M_{ccw}\)’

**Procedure:**
1. Concatenate \(M_{cw}\) to its self
   \[ M = \text{Append}(M_{cw}, M_{cw}); \]
2. Mirror the 2 cycles clip
   \[ M_r = \text{Mirror}(M); \]
3. Get the synchronised 1 cycle from ‘\(M_r\)’
   \[ M_s = M_r[K_i : K_{i+n}]; \]
   where \(K_i\) is the Key-event that in-phase with \(T_{M_{cw}}(0)\)
   and \(n\) is one less the number of key-events per cycle.
4. Translate and rotate \(M_s\) to align with \(M_{cw}\)
   \[ M_{ccw} = TR * M_s; \]
   where \(TR\) is the appropriate transformation matrix.

Figure 4.13: Synchronised Mirror algorithm, ‘\(Sync\text{Mirror}\)’.

The path control problem in particular is addressed in some recent research in human animation. Sun et. al. [90] combined pre-recorded data, of the sagittal plane angles, with a walking model to synthesise walking on uneven terrain and along an arbitrary path. This approach is limited to the synthesis of walking motion and is computationally expensive. Variations in style are limited to a few low-level parameters of the motion model. Gleicher [34] introduced a method for editing the motion path of an existing motion clip based on an extended displacement mapping technique. Our work is different in that instead of editing individual motion clips, we synthesise the desired motion based on two, or more, given motion clips. In Gleicher [34], the issue of highly-curved paths and sharp turns are not discussed. Lucas et. al. [55] introduced another path editing method based on a structure referred to as motion graph; a directed graph that organises and links the original motion clips and the possible
Figure 4.14: This diagram explains the synchronised mirror algorithm, ‘SyncMirror‘ of figure 4.13

- Left foot has just touched the floor
- Right foot has just touched the floor
4.4. Path Control

transitions between them. Our work requires much less input data (hundred frames) and employs parametric blending technique which is computationally efficient.

Our technique is applied directly on the given data without building an intermediate statistical model or transition graphs. As search operations are not employed in our technique, problems related to selecting the proper criteria or cost function (such as some unwanted set of frames in the middle of motion sequence [55]) are not exist. This also makes the computation complexity of our approach linearly related to the length of desired animation sequence.

The rest of this section presents our parametric control of motion path. Section 4.4.1 discusses the relation between motion path and direction and our selected representation of motion direction. Then the parameterisation process is explained in section 4.4.2. The parametric synthesis is presented in section 4.5.2, while the results and their evaluation are presented in section 4.7.1.

4.4.1 Motion Path and Direction

Motion path and motion direction are two closely related motion properties. Motion direction at a particular time instant or frame, in its simplest definition, is the direction of the tangent to the motion path. However, motion path is not explicitly specified in the animation data. In animation data, root positions are usually moving around the motion path. Hence, the motion path is sometimes regarded as an abstract of the root positions curve [34]. This makes the trajectory of root positions the first candidate for defining the motion direction and path. However, due to the bipedal nature of human movement, this trajectory of root positions is not smooth and this affects the motion directions as well (see figure 4.15). Consequently, root trajectory is not suitable for our parameterisation process.

The next candidate from the animation data is the root rotation around the vertical axis. It has been found that it is much smoother than the root trajectory. Although it does not represent the motion direction directly, with some analysis, the relationship can be defined and used to obtain the motion direction. For example, in walking and running, as the supporting leg changes, the root orientation swings around the main
Chapter 4. Parametric Synthesis of Animation Sequences

Figure 4.15: Motion directions extracted from trajectory of root positions for different walking motions (ranges from straight-line walking [bottom] to curved-path walking [top])

direction of motion. Even for a straight path, the root orientation is not constant as it swings with the swinging leg. Moreover, for a curved path, the root orientation represents both the curved path and the swinging due to nature of the motion as depicted in equation 4.8.

\[
\Phi_R = \phi_d + \phi_s
\]  

(4.8)

where \( \Phi_R \) is the root orientation, \( \phi_d \) is the motion direction, and \( \phi_s \) is the swinging due to the nature of motion.

Based on the above analysis and given straight and curved path motions of the same type, the swinging part \( \phi_s \) resulting from the nature of motion can be eliminated to obtain the root orientation that is only related to the motion direction \( \phi_d \). Figure
4.4. Path Control

4.16.a shows the root rotation about the vertical axis from straight-path (bottom) to curved-path walking (top) clips. Figure 4.16.b shows the rectified root rotation after subtracting the swinging part. It is worth mentioning that the small hump on some curves are attributed to the given curved-path motion, where some excessive rotation has been produced within the motion itself.

The next step is the parameterisation of input clips in order to provide parametric control over arbitrary paths as discussed in the next section.

4.4.2 Parameterisation

After defining our representation for motion direction, we can now start the parameterisation process. As the synthesis process involves blending between the given motion clips, the main task of the parameterisation process is to find the mapping function that represents the relationship between the motion direction parameter $\phi$ and its corresponding blending factor $X$ that can produce the motion in the specified direction along the specified path (i.e. finding the function $f$ such that $X = f(\phi)$).

However, the orientation of the character is not constant and is not changing with constant rate during the given motion cycles, which is also the case for the desired animation. As we would like to give user control over the path at any frame or time instant, we need to define the mapping function for each of the given frames. This is an example of the instantaneous parameter case discussed in section 4.3.2 where the mapping function $f$ becomes a group of functions $f_{Fi}$ for $1 \leq i \leq L_s$, where $L_s$ is the length of the desired animation sequence.

The length of desired animation sequence can widely vary from a few cycles to hundreds and thousands of motion cycles while the available input motions are usually limited to short motion clips, in our case clips of one cycle each.

As discussed in section 4.3.2 the parameterisation is performed over the length of the input clips $L_n$ and then used repeatedly for synthesis of the animation sequence as shown in equation 4.2.

The parameterisation is performed for different values of the blending factor $X$ at
a. Root rotations about the vertical axis (consists of swiging and path changes)

b. Rectified root rotations after subtracting swinging part

Figure 4.16: Motion directions extracted from root rotations about vertical axis. The example is for walking motions (ranges from straight-line walking [bottom] to curved-path walking [top])
each frame. Sample relationships are shown in figure 4.17 for walking motion. These relationships can be approximated by algebraic functions as shown in figure 4.18.

![Relationship between motion direction and blending factor at each frame](image)

Figure 4.17: Relationships between motion directions and corresponding blending factors at each frame

The next sub-section describes the parametric synthesis of motion that follows user-input arbitrary path.

### 4.4.3 Parametric Synthesis

Given *straight*-path and *curved*-path motion clips and their labelled key-event times, these motions can be carefully synchronised together (as described in section 3.4.4) to maintain the motion constraints. Then, the parameterisation process described in section 4.4.2 is applied on the input motions so that the group of parameterisation functions $f_{F_i}$ can be determined.

Given a desired path drawn by the user as shown in figure 4.20, the list of global motion direction at each frame $[\phi^l_C]$ is extracted where $1 \leq l \leq L_s$ and $L_s$ is the
Chapter 4. Parametric Synthesis of Animation Sequences

Approximating relationship between motion direction and blending factor

Figure 4.18: Approximated Relationships between motion directions and corresponding blending factors at each frame

Figure 4.19: Input clips for the PathControl module. One cycle of (from left to right) StraightPath, CurvedPath (CW), SynchronisedMirror CurvedPath (CCW) motions.
length of desired animation sequence. Then the desired motion is synthesised using the algorithm in figure 4.21.

For synthesis, at each frame $F_i$ within a cycle (i.e. $1 \leq i \leq L_n$; where $L_n$ is the length of input clips), local direction $\phi_{L_i}^{F_i}$ is calculated and used as the desired local parameter for which the corresponding blending factor $X_{F_i}$ is calculated. Then, the obtained blending factor at each frame is passed to the synchronised blending module (section 3.4.1) to synthesise the actual animation data from the given motion clips. A sample output is presented in figure 4.22.

To generate motion along an arbitrary path that includes turning in both directions, clockwise and anti-clockwise, we require motion clips which have turnings in both directions. Within our research, we developed a technique for automatically generating motion clip in curved-path in one direction from a given motion clip with a curved-path
Input:
* Straight-line path motion cycle ‘Ms’, annotated with its list of key-event times \([TM_s]\).
* Clock-Wise curved-path motion cycle ‘Mcw’, annotated with its list of key-event times \([TM_{cw}]\).
* Counter-Clock-Wise curved-path motion cycle ‘Mccw’, annotated with its list of key-event times \([TM_{ccw}]\).
* Desired path directions \([\phi_G^l]\), where \(l = 1 \rightarrow L_s; L_s\) = the length of desired sequence.
* Parameterisation functions \(f_{R_l}\).

Output:
* ‘Mp’, animation sequence following the desired arbitrary path.

Procedure:
1. For each frame \(F_i\) at cycle \(C_j+1\), the local direction \(\phi_L^{F_i}\) is calculated as follows:
\[
\phi_L^{F_i} = \phi_G^{F_i} - \phi_G^{F_{C_j}};
\]
where \(i = 1, \ldots, L_n\); \(L_n\) = Length of input cycle.
\(j = 0, \ldots, N\); \(N\) = Number of cycles in the desired sequence.
\(\phi_G^{F_i}\) = Global desired direction at frame \(F_i\).
\(\phi_G^{F_{C_j}}\) = Global direction at end of previous cycle \(C_j\).

2. Calculate corresponding blending factor:
\[
X(F_i) = f_{R_l}(\phi_L^{F_i});
\]

3. Generate motion using the synchronised blend module as discussed in section 3.4.1
\(M_{p_j} = SyncBlend(M_{s}, M_{\text{curve}}, [X]);\)
where \(M_{\text{curve}} = M_{cw}\) or \(M_{ccw}\) depending of the sign of \(\phi_L^{F_i}\).

4. Accumulate the generated sequence:
\(M_p = Append(M_p, M_{p_j});\)

Figure 4.21: Algorithm for Parametric Synthesis for motion on arbitrary path
Figure 4.22: The output generated from the user-input arbitrary path drawn in figure 4.20. The synthesised motion path is shown in *green* while the desired path drawn by the user is shown in *yellow*.

in the opposite direction, as presented in section 4.3.4. Our developed *synchronised mirror* technique, given only one curved path clip, a similar clip with a mirrored curve path is generated automatically (see figure 4.19. more details are presented in section 4.3.4). The generated motion clip is already annotated and its key-event times are determined.

The algorithm presented in figure 4.21 assumes that the given motion cycles are prepared for smooth concatenation (i.e. each cycle has matched boundaries). If it is not the case, as is common on the real human movements, input cycles need to be cyclified [5] beforehand as discussed in section 4.3.1. The input cycles can be also annotated automatically using the automatic gait analysis algorithm presented in section 3.4.3.

The algorithm is extendable to allow special cases which may not be realistically achievable using the slightly curved-path motions as shown in the results in section 4.7. For example, sharp turns such as *turning around* motions have been achieved using an
extra motion clip specifically for turning back.

4.5 Terrain Control

In virtual environments, like in the real world, the floor is not restricted to level ground. Uneven terrain is the more general case which is likely to be required in applications such as navigation in virtual environment, animated films and 3D broadcasting. Synthesising motions on uneven terrain is another example of the challenging tasks in human animation.

In this section, parametric control of motion on arbitrary uneven terrain is presented. The method is based on the proposed parametric synthesis framework and depicts another practical example of its application. The objective is to provide the user with a high-level parametric control for synthesis of motion on arbitrary uneven terrain.

The problem of generating motion on uneven terrain has been addressed in some previous research in human animation. Kinematic and dynamics techniques have been the common methods for generating motion on uneven terrain [95, 62]. Sun et. al. [90] combined a walking model and pre-recorded data of the sagittal plane angles to generate walking motion on uneven terrain and along an arbitrary path. These approaches have common limitations. They are limited to synthesis of the modelled motions only, walking motion in these cases. They are also computationally expensive. Variations in style are limited to a few low-level parameters of the motion model. Park et. al. [73] introduced a high-level control for following an uneven terrain, which is based on a retargeting process introduced in [81].

In our system, following uneven terrain is achieved using the same technique of parametric synchronised blending. The parametric approach presented in this thesis is different as it is applied directly to the given data and requires no model of the motion. It is also computationally efficient and flexible for variations in style based on the available motion captured data.

The remainder of this section presents our parametric control of motion over an arbitrary uneven terrain. Slope calculation and the parameterisation process are presented
in section 4.5.1. The parametric synthesis is then presented in section 4.5.2.

4.5.1 Parameterisation

Before identifying the mapping function, the method of calculating the motion parameter (slope in this case) should be identified. Following uneven terrain can be achieved by continuously changing the slope of a virtual floor to match the given terrain. The virtual floor is the floor that the input motion clips have been captured on.

Calculating the slope of the floor is more straightforward than the motion direction. Ideally, it can be obtained by simple measurements of the capturing platform setup. However, when we get the animation clips it is more likely that such information may not be available. Hence, the calculations should be based on the available animation data itself. We can select a specific joint in the skeleton such as the left or right foot and calculate the slope $S$ from its position at the start and end of the cycle as follows:

$$S = \tan^{-1}\left(\frac{Y_F - Y_0}{X_F - X_0}\right)$$

where $Y_0$ and $X_0$ are the vertical and horizontal position of the foot at the starting frame. Similarly, $Y_F$ and $X_F$ are the vertical and horizontal position of the foot at the final frame of the cycle.

To find the mapping function $X = f(S)$, where $X$ is the blending factor, the parameterisation procedure, discussed in sections 3.4.2 and 4.3.2, is performed over the length of input clips $L_m$. Sample relationship is shown in figure 4.23 for walking motion on up and down sloped floors. The approximated relationship is also shown in the figure.

The obtained mapping function, representing the relationship in figure 4.23, is used for the parametric synthesis as discussed in the following section.

4.5.2 Parametric Synthesis

Given two motions on different sloped floors, labelled with their key-event times, these motions are synchronised together (as described in section 3.4.4) to maintain the motion
Chapter 4. Parametric Synthesis of Animation Sequences

Figure 4.23: Relationship between floor slope and the corresponding blending factor.

Constraints. Input motions are moving up and down over a sloped floor as shown in figure 4.24.

Figure 4.24: Input clips for the TerrainControl module. UpSlope cycle (Left) and DownSlope (Right).

The desired terrain drawn by the user is provided in terms of floor slopes $[\theta_i]$ where $1 \leq i \leq L_s$ and $L_s$ is the length of desired animation sequence. For each slope $\theta_i$ at frame $i$, the corresponding blending factor $X_i$ is calculated using the mapping function $X_i = f(\theta_i)$ obtained from the parameterisation procedure of section 4.5.1. Then the desired motion is synthesised according to the calculated blending factors. The synthesis
4.5. Terrain Control

The procedure is depicted in the algorithm of figure 4.25.

| Input: | * A motion cycle ‘$M_U$’ on Up sloped floor, annotated with its list of key-event times [$T^{M_U}$].  
| * A motion cycle ‘$M_D$’ on Down sloped floor, annotated with its list of key-event times [$T^{M_D}$].  
| * Floor slopes of the desired uneven terrain [$\theta_i$], where $1 \leq l \leq L_s$; $L_s =$ the length of desired sequence.  
| * Parameterisation function $f$ such that $X_i = f(\theta_i)$. |

| Output: | * ‘$M_T$’, animation sequence of motion over the given arbitrary uneven terrain.  |

| Procedure: | 1. For each slope $\theta_i$ at frame $i$, calculate the corresponding blending factor: $X_i = f(\theta_i)$;  
| 3. Generate motion using the synchronised blend module as discussed in section 3.4.1 $M_T = \text{SyncBlend}(M_U, M_D, [X])$; |

Figure 4.25: Algorithm for Parametric Synthesis of motion over arbitrary uneven terrain.

The algorithm presented in figure 4.25 assumes that the given motion cycles are prepared for smooth concatenation (i.e. each cycle has matched boundaries). If it is not the case, as is common on the real human movements, input cycles need to be cyclified [5] beforehand as discussed in section 4.3.1. The input cycles can be also annotated automatically using the automatic gait analysis algorithm presented in section 3.4.3.

Examples of the results achieved using this module is shown in section 4.7.2 along with both qualitative and quantitative evaluation of their quality.

The next section presents the extension of the presented approach for multi-parameters synthesis where more than one motion parameters can be controlled simultaneously.
4.6 Multi-Parameters Synthesis

This section shows the extension of the proposed parametric approach for cases of more than one input parameter at a time. The parametric blending approach can be used repeatedly, in a nested manner, to synthesise motions that simultaneously satisfy more than one parameter at a time. For simplicity of discussion, the case of two simultaneous parameters is considered but extension for higher dimensions are discussed later in the section.

One of the difficulties of multi-parameters synthesis is that the parameters controlled by input motion clips are usually not independent. This means that blending input motions to achieve a specific value of one parameter results in changes of values of other parameters as well. Figure 4.26 depicts a real example of the parameters space for simultaneous control of speed and slope motion parameters.

A solution could be modifying the input motion clips, so that they are uniformly distributed on a regular grid over the parameters space (as shown in magenta in fig-
4.6. Multi-Parameters Synthesis

This would align adjacent pairs of input clips so that the variations, within each pair, is only in one parameter. This solution is not convenient due to the amount of pre-processing required to re-sample the input clips on the regular grid (see figure 4.27). This pre-processing includes repeated parameterisation and synthesis processes to produce the required motions on the grid. Moreover, the complexity of achieving the regular grid increases with the number of parameters.

Due to these drawbacks, we use the provided input motions directly without such modifications. The input motion samples are on an irregular grid in the parameters space as shown (in dashed green) in figure 4.27. To determine the parameterisation functions, the parameterisation technique presented in section 3.4.2 is extended such that different values of blending factors are explored, in a nested manner, to take into account the dependency of the parameters as discussed below.

Assuming that we have $k$ motion clips $M_1, ..., M_k$ with $n$ parameters $P_1, ..., P_n$ each. There will be $n$ different blending factors $X_{P_1}, ..., X_{P_n}$ for motion synthesis through
blending. The process of generating the parameterisation data is presented as pseudo code in figure 4.28.

```
for X_{P_1} = 0:0.1:1  // m intervals
  for X_{P_2} = 0:0.1:1
    for X_{P_n} = 0:0.1:1
      Do Blend operations using X_{P_1}, ..., X_{P_n}
      Calculate parameters P_1, ..., P_n of generated motion
    end;
  end;
end;
```

Figure 4.28: Pseudo Code for Generating Parameterisation data for Multi-parameters Synthesis

This results in a matrix $\Psi$ of size $2n \times m$ containing the parameterisation data where $m$ is the number of parameterisation intervals, which is heuristically selected to be 11 in our case (i.e. $X = 0, 0.1, ..., 1$) as shown in the pseudo code above.

To determine the parameterisation functions, for each blending factor $X_{P_i}$, a submatrix $\psi$ of size $(n + 1) \times m$ is selected from the full data matrix $\Psi$ such as:

$$
\psi(X_{P_i}) = \begin{bmatrix}
\Psi(1 : m, 1) & \Psi(1 : m, n + 1 : 2n)
\end{bmatrix} \quad (4.9)
$$

The first column in $\psi$ contains the possible values of the blending factor $X_{P_i}$ and the rest $n$ columns contain the possible values of parameters $P_1, P_2, ..., P_n$.

Using the $\psi(X_{P_i})$ matrices, multiple parameters regression is carried out to determine the parameterisation functions $f_{P_i}$ such that:
4.6. Multi-Parameters Synthesis

\[ X_{P_1} = f_{P_1}(P_1, P_2, ..., P_n) \]
\[ X_{P_2} = f_{P_2}(P_1, P_2, ..., P_n) \]
\[ .... = \ldots \ldots \ldots \ldots \ldots \ldots \]
\[ X_{P_n} = f_{P_n}(P_1, P_2, ..., P_n) \] (4.10)

Figures 4.29 and 4.30 show examples of the above parameterisation functions for the case of simultaneous control of speed over uneven terrain (i.e. variable slope). A similar example is depicted in figures 4.31 and 4.32 for the case of controlling the hitting position of a kicking action.

![Plot of the 1st Parameterisation function for controlling speed over uneven terrain.](image)

Our approach, presented in figure 4.28 and equations 4.10 reduces the parameterisation requirements to ‘n’ parameterisation functions, one for each parameter, where \( n \) is the number of parameters. Moreover, all parameterisation functions are determined once beforehand and then used for every user-input. As a comparison, multi-dimensional
Chapter 4. Parametric Synthesis of Animation Sequences

Figure 4.30: Plot of the 2nd Parameterisation function for controlling speed over uneven terrain.

Figure 4.31: Plot of the 1st Parameterisation function for controlling the hitting position of a kicking action.
4.6. Multi-Parameters Synthesis

\[ X_2 = f_2 (X, Y) \]

Figure 4.32: Plot of the 2\textsuperscript{nd} Parameterisation function for controlling the hitting position of a kicking action.

interpolation using radial—basis functions (RBF), utilised by Rose et. al. [77], requires \( k \) parameterisation functions, where \( k \) is the number of motion clips, which is always larger than the number of parameters \( (k > n) \).

In addition to reducing the parameterisation requirements, the proposed approach avoids the need for modifying the input motion clips to produce a regular grid on the parameters space.

In our experiments, the desired user-input parameters are satisfied with quiet high accuracy of 97\% in speed and 99\% in slope. Results have been found to be as realistic as the input motions. Qualitative and quantitative evaluations are presented in section 4.7.3.

Extension of the discussed technique, for higher dimensions, is achievable by adding more parameters as new dimensions in the parameters space. The extension of the parameterisation procedure is achieved by adding a new nesting level, for each additional parameter, to the procedure in figure 4.28 which results in an extra parameterisation function in equations 4.10 for the added parameter.

The presented technique shares a common drawback with other interpolation techniques
The drawback is the increase of the number of required motion clips with the increase of required motion parameters. This results in a requirement of $2^N$ motion clips of the same movement with different (extreme) parameter values, where $N$ is the number of parameters. To reduce the database size, we developed an adaptive compression technique [1] for human animation based on wavelet analysis as presented in appendix A.

4.7 Results and Evaluation

In this section, sample results of the extended framework are presented along with their evaluation. Sections 4.7.1, 4.7.2, and 4.7.3 present the results of the main modules developed with the extended framework which are PathControl, TerrainControl, Multi Parameters synthesis. It is worth noting that these modules assume that the input clips have perfect cycles (i.e. cycle boundaries are perfectly matched) which is not the common case in real human movements. Unmatched cycle boundaries is one of the extra problems in generating long animation sequences from short input clips as discussed in section 4.2 and 4.3.1.

The identified problems has been tackled in various sections. Sections 4.7.4, 4.7.5 and 4.7.6 show the results and evaluation of the other enhancements within the synthesis process to allow the synthesis of long animation sequences from short clips. This includes cyclification, synchronised mirror and the improved incremental root-trajectory blend.

The last example, section 4.7.7, demonstrates an example of integrating synthesised sequences from different presented modules for constructing a longer sequence of a real scene.

The evaluation is done both qualitatively and quantitatively as discussed in section 3.5. Qualitative evaluation is performed by visual inspection of the synthesised sequences as well as the input clips. The quantitative evaluation uses the quantitative measures introduced and defined in section 3.5.
4.7. Results and Evaluation

4.7.1 Path Control

Synthesising motions on arbitrary paths is one of the interesting tasks in animation. It is useful in many applications such as games, navigation in virtual environment, animated film production and 3D broadcasting.

In section 4.4, the proposed extended parametric synthesis framework has been utilised to allow an intuitive control of the motion path. In this sub-section, examples from the results of the path — control module are presented, evaluated and compared with the original input clips. The typical input clips for the path-control module are Straight-path, clockwise curved-path, and counter-clockwise curved-path motions as depicted in figure 4.33 (left and middle). An additional input clip, figure 4.33 (right), is added for more flexibility of including TurnBack actions within the path. The total number of input frames is 1G 7 frame which represents one cycle of each of the input clips except the TurnBack clip which has 2 cycles.

Figure 4.33: Input clips for the PathControl module. One cycle of (from left, to right) StraightPath, CurvedPath (CW), SynchronisedMirror CurvedPath (CCW), and TurnBack motions.

Figure 4.34 depicts a synthesised motion along an arbitrary path drawn by the user (shown in yellow). The synthesised animation sequence consists of 1807 frames which is more than 10 times the total number of input frames. We can see that the path of synthesised motion, shown in green, closely follows the desired arbitrary path drawn by
Chapter 4. Parametric Synthesis of Animation Sequences

the user. Visual inspection (experiments details presented in appendix C) also shows that the synthesised motion is as realistic as the input clips. The quantitative evaluation indicates that the measured quality (represented by the Jerk and Feet–Floor Contact Error measures) of the synthesised motion is close to the input clips as discussed below (see figure 4.37).

Figure 4.34: Example of following an arbitrary path drawn by the user. The drawn path is shown in yellow while the synthesised path is shown in green.

Synthesis of longer paths are shown in figures 4.35 and 4.36. Figure 4.35 shows a motion that follows an arbitrary path representing the handwritten name ‘Amr’. The synthesised sequence in this example consists of 2404 frames which is more than 14 times the total number of input frames. Similarly, figure 4.36 depicts a motion along the arbitrary path representing the handwritten word ‘ACM’\(^1\). It contains 2796 frames which is more than 16 times the total number of input frames. The animation sequences of the above examples are available in the enclosed CD-ROM along with other examples.

The quantitative evaluation (using the quantitative measures defined in section 3.5) is shown in figure 4.37. The maximum values of the measures show the maximum errors

\(^1\)ACM is abbreviation for Association for Computing Machinery
Figure 4.35: Example of following an arbitrary path representing the handwritten name *Amr*. The drawn path is shown in *yellow* while the synthesised path is shown in *green*.

Figure 4.36: Example of following an arbitrary path representing the handwritten word *ACM*. The drawn path is shown in *yellow* while the synthesised path is shown in *green*. 
that occur in the synthesised sequences as well as the input clips. While the RMS values show the variation of the error for both synthesised sequences and input clips.

The maximum jerk values, figure 4.37.a, show that the quality of synthesised sequences is close to the input clips. We can see that the maximum jerk values of synthesised sequences remain within the corresponding values of input clips. The RMS jerk values, figure 4.37.c, also show similar behaviour.

The maximum and RMS values of the feet – floor contact error measure are shown in figures 4.37.b and 4.37.d respectively. The RMS values are within the corresponding values of input clips. However, the maximum value of some synthesised sequences are slightly above the corresponding maximum value of the input clips. This marginal extra error can be attributed to sudden change in direction, required by the user at certain frames, which is beyond the limits of the input clips.

To conclude, it has been shown that the parametric path control module has been successful in generating animation sequences for motions along arbitrary paths drawn by the user. The paths of synthesised sequences are close to the paths defined by the user. Moreover, both qualitative and quantitative evaluations showed that the quality of the synthesised sequences is close to the quality of input clips. The presented

![Figure 4.37: Quantitative evaluation of ‘PathControl’ examples.](image)
synthesised sequences are at least 10 times longer than the total length of all input clips added together. This demonstrates our approach of parametric synthesis of long sequences from short motion clips.

### 4.7.2 Terrain Control

Synthesising motions on uneven terrain is another example of the challenging tasks in human animation. For applications such as navigation in virtual environment, animated films and 3D broadcasting, the floor is not restricted to level ground. Uneven terrain is more likely to be required.

Section 4.5 presented our TerrainControl module where the proposed extended parametric synthesis framework has been utilised to allow an intuitive control of the motion on arbitrary uneven terrain. Examples from the results of the TerrainControl module are presented, evaluated and compared with the original input clips in this section.

The input clips for the TerrainControl module are UpSlope and DownSlope motions as depicted in figure 4.38. The total number of input frames is less than 75 frames which represents a cycle of each of the input clips.

![Figure 4.38: Input clips for the TerrainControl module. UpSlope cycle (Left) and DownSlope (Right).](image)

Figures 4.39 and 4.40 show examples of motion on arbitrary uneven terrain provided by the user. It can be seen that the synthesised motions closely follow the given terrains. It is worth noting that the generated path (shown in green) may not look as smooth as the drawn path (shown in yellow). This is because the viewing software actually plots the projection of the root position on the floor but adds the height of the lowest point
in the avatar at each frame. This could be the supporting foot, if going up-hill, or the swinging foot (towards the end of swinging phase), if going down-hill.

The above animation sequences are presented in the enclosed CD-ROM along with other examples. Visual inspection (experiments details presented in appendix C) indicates that the synthesised sequences are as realistic as the input clips.

Figure 4.39: Example of synthesising motion (path shown in green) on arbitrary uneven terrain provided by the user (shown in yellow).

Figure 4.40: Another example of synthesising motion (path shown in green) on arbitrary uneven terrain provided by the user (shown in yellow).

Quantitative evaluation is performed by calculating the Jerk and Feet-Floor Contact Error measures as depicted in figure 4.41. The maximum values of both measures for the synthesised sequences, figures 4.41.a, 4.41.b are just slightly above its corresponding maximum value for the input clips. However, the RMS values of both measures (figures 4.41.c, 4.41.d) are still within the range of their corresponding values of the input clips.
4.7. Results and Evaluation

In section 4.6, the proposed parametric synthesis approach is extended again to incorporate more than one input parameter simultaneously. This has been demonstrated by intuitively controlling speed of motion over uneven terrain simultaneously.

This section presents examples of the synthesised sequences with arbitrarily varying speed over arbitrary uneven terrain. The evaluation of the results is also discussed later on this section.

The input clips for this module are *FastDownSlope*, *SlowDownSlope*, *FastUpSlope* and *SlowUpSlope* motions as depicted in figure 4.42. The total number of input frames is 135 frames which represents a cycle of each of the input clips.

Figures 4.43 and 4.44 show examples of motions with arbitrary speeds over arbitrary uneven terrains both provided by the user. It can be seen that the synthesised motions are closely following the given terrain. It is worth noting that the generated path (shown in green) may not look as smooth as the drawn path (shown in yellow). This is because the viewing software actually plots the projection of the root position on the floor but adds the height of the lowest point in the avatar at each frame. This could be the
supporting foot, if going up-hill, or the swinging foot (towards the end of swinging phase), if going down-hill.

The above animation sequences are presented in the enclosed CD-ROM along with other examples, including the control of the hitting position of a kicking action. Visual inspection (experiments details presented in appendix C) indicates that the synthesised sequences are as realistic as the input clips.

For quantitative evaluation, the *Jerk* and *Feet—Floor Contact Error* measures are calculated for synthesised sequences as well as for the input clips as depicted in figure 4.45. The *maximum* values of both measures for the synthesised sequences, figures 4.45.a and 4.45.b, are around the corresponding maximum value for the input clips. However, the *RMS* values of both measures (figures 4.45.c, 4.45.d) are still within the range of their corresponding values of the input clips.
4.7. Results and Evaluation

Figure 4.43: Example of synthesising motion with arbitrary speeds (path shown in green) over arbitrary uneven terrain provided by the user (shown in yellow).

Figure 4.44: Another example of synthesising motion with arbitrary speeds (path shown in green) over arbitrary uneven terrain provided by the user (shown in yellow).
4.7.4 Cyclification

Generating long animation sequences from short input clips assumes that the input clips have perfect cycles (i.e. cycle boundaries are perfectly matched) which is not the common case in real human movements. Unmatched cycle boundaries is one of the extra problems in generating long animation sequences from short input clips as discussed in section 4.3.1.

In this section, results as well as evaluation of the developed cyclification techniques are presented for a variety of motions. Section 4.7.4.1 presents the results and evaluation for cyclification of ‘Walk’ cycle on a level floor while section 4.7.4.2 presents the results and evaluation for ‘Walk’ cycle on a sloped floor (up/down hill). Sections 4.7.4.3 and 4.7.4.4 shows the cyclification results and evaluation for ‘Kick’ and ‘Jump’ respectively.

In the quantitative evaluation, both maximum and RMS values of the quantitative measures are calculated for both original and cyclified clips. The evaluation also shows the effect of cyclification on sequences of 2, 4, and 8 concatenated motion cycles and compares it with its equivalent sequence of input clips (with the same length). This emphasises the problem of unmatched cycle boundaries when concatenating cycles in
order to construct longer animation sequence. It also depicts the improvement achieved by the developed cyclification techniques.

4.7.4.1 Walk on level floor

In figure 4.46, an example of a 'Walk' motion on level floor is depicted. Both the original motion and its cyclified version are shown in the figure. The artifacts, due to the unmatched cycle boundaries, may be difficult to recognise from static figures. However, the quantitative evaluation measures, shown in figure 4.47, indicate the improvement of the quality of the motion cycle.

![Figure 4.46: Original (Left) and cyclified (Right) 'Walk' motion.](image)

The important benefit of cyclification is emphasised in figure 4.48. In figure 4.48, the quantitative measures for motion sequences of 2, 4, and 8 cycles of both original and cyclified motions are depicted. They show that the measures are more stable for the cyclified motion with the increase of number of concatenated cycles. In the case of an original cycle, with unmatched boundaries, although the errors stabilise later, they jump with the first concatenation especially the jerk measure.

It is worth noting that the constant value of the measure along different lengths of sequences indicates that it is not resulting from the unmatched cycle boundaries. The
error could be within the cycle itself and this is the reason for being the same regardless of the concatenation. The feet – floor contact error measure of some motion examples, of figure 4.48b, is an example of this case. However, it should be noticed also that, in some cases, the value itself has improved by cyclification over the original cycle.

4.7.4.2 Walk on slope floor

A similar example of ‘Walk’ motion, but on a sloped floor, is shown in figure 4.49. Again, both the original motion and its cyclified version are shown in the figure.

The quantitative evaluation measures, shown in figure 4.50, indicate the improvement of the quality of the motion cycle. The stability of motion quality of cyclified sequence is emphasised in figure 4.51. In figure 4.51, the quantitative measures for motion sequences of 2, 4, and 8 cycles of both original and cyclified motions are shown. It is worth noting that the constant value of the measure along different lengths of sequences indicates that it is not resulting from the unmatched cycles boundaries. It is within the cycle itself and this is the reason for being the same regardless of the concatenation.
4.7. Results and Evaluation

4.7.4.3 Kick

Cyclified techniques have been applied to other motions such as ‘Kick’. Figure 4.52 shows original and cyclified ‘Kick’ motions. The maximum values of the quantitative measures, shown in figure 4.53.a and 4.53.b, indicate that there is no improvement of the quality of the motion cycle. However, the evaluation after concatenating 2 or more cycles shows the improvement of motion quality as depicted in figure 4.54. It is worth noting that the cyclification techniques are developed for the later case, when 2 or more cycles are concatenated. Improving the quality of individual cycles is an added value but not the main objective.

4.7.4.4 Jump

‘Jump’ motion is also cyclified by the developed cyclification techniques. Figure 4.55 shows original and cyclified Jump motions.
Chapter 4. Parametric Synthesis of Animation Sequences

Figure 4.49: Original (Top) and cyclified (Bottom) ‘Walk’ motion on slope floor.

a. Maximum Jerk  
b. Maximum Feet – Floor Contact Error

c. RMS Jerk  
d. RMS Feet – Floor Contact Error

Figure 4.50: Quantitative evaluation of original and cyclified ‘Walk’ cycles on slope floor (1 cycle each).
4.7. Results and Evaluation

Figure 4.51: Effect of concatenation on both original and cyclified 'Walk' cycles on slope floor.

Figure 4.52: Original (Left) and cyclified (Right) 'Kick' motion (Initial and final postures of each motion).
Chapter 4. Parametric Synthesis of Animation Sequences

Figure 4.53: Quantitative evaluation of original and cyclified ‘Kick’ motions.

Figure 4.54: Effect of concatenation on both original and cyclified ‘Kick’ cycles.
The quantitative measures, shown in figures 4.56 and 4.57, indicate that the quality of motion has improved by cyclification especially after concatenating 2 or more cycles which is the main aim of the developed techniques.

Figure 4.55: Initial posture (middle), final Original posture(Left) and final cyclified posture(Right) ‘Jump’ motion.

Figure 4.56: Quantitative evaluation of original and cyclified ‘Jump’ motions.

4.7.5 Synchronised Mirror

The Synchronised Mirror algorithm has been introduced in section 4.3.4 as an example of using simple animation processing techniques to enrich the existing animation database [2]. A typical example of using this algorithm has been demonstrated in the
Chapter 4. Parametric Synthesis of Animation Sequences

PathControl module, section 4.4, where only one curved-path clip is given and the other one is synthesised automatically by the developed Synchronised Mirror algorithm. This example is depicted in figure 4.58. It can be seen that although the ordinary Mirror can produce a realistic motion on the opposite curvature, the resulting motion timing is out of phase. However, the timing of the result of our SyncMirror algorithm is in phase with the original clip. Moreover, it is automatically annotated and ready for use in the parametric blending framework.

The synchronisation of the mirrored clip with the original clip is crucial for successful blending as discussed in section 4.3.4 and shown in figure 4.12. Figure 4.59 shows the same example of figure 4.12, however, the mirrored clip (right) has been synthesised using the introduced SyncMirror modifier. It can be seen that the mirrored clip in figure 4.59 starts with the left foot in front (as the original clip does). As the original and mirrored clip (of figure 4.59) are in-phase, the result of blending them together is a valid human motion, as shown in the middle of figure 4.59, which was not achievable using the result of the ordinary Mirror modifier (see figure 4.12).

Figure 4.60 shows the quantitative evaluation of the Synchronised Mirrored as well as the original clip. The maximum values of the Jerk and Feet – Floor Contact Error...
Figure 4.58: Compare original (middle), Mirrored (Left), and Synchronised Mirrored (Right) motion clips. It can be noticed that the mirrored motion (Left) starts with the right-foot while the original (middle) and Synchronised Mirrored (Right) motions start with the left-foot.

Figure 4.59: Blending original (left) and Synchronised mirror clip (right) produces a valid motion (middle) as the mirrored clip is in-phase with the original clip.
measures are almost the same for both original and Synchronised Mirrored clips. The quantitative measures indicate that the quality of the Synchronised Mirrored clip is almost the same as the original clip.

Figure 4.60: Quantitative evaluation of 'Synchronised Mirrored' clip.

4.7.6 Improved Incremental Root Trajectory Blend

As discussed in section 4.3.3, blending the root joint trajectory needs careful handling as it influences the position and orientation of the whole skeleton. In section 4.3.3, we identified few limitations of the ordinary blending that limit the blending process, especially when the difference in motion direction increases between input motions. Our improved root-trajectory blending technique is also presented in section 4.3.3.

In this section, in addition to the example given in section 4.3.3, another example of the limitation is given in figure 4.61. The improvement achieved by applying our improved blending technique is demonstrated in figure 4.62. Comparing the two figures, figures 4.61 and 4.62, we can see that the improved blending technique has been successful in avoiding the sudden root-flip.

By visual inspection of figures 4.61 and 4.62, we can spot the limitation and the improvement introduced by the proposed technique. This is also supported by figure 4.63 which plots the change in motion direction between the initial and final frame of the motion in both cases of figures 4.61 and 4.62. The opposite difference directions, between the two versions of blending, are attributed to the fact that blending without the presented improvements results in sudden flip of the root to opposite direction.

Although we are able to spot the limitation and the improvement by visual inspection,
4.7. Results and Evaluation

Figure 4.61: The synthesised root trajectory (middle) is suddenly flip to the opposite direction after the difference in direction of input motions exceeds 180°. Input motions are StraightWalk (right) and TurnBack clips (left).

Figure 4.62: The synthesised root trajectory (middle) has no sudden flipping any more after using the improved blending technique. Input motions are StraightWalk (right) and TurnBack clips (left).
Chapter 4. Parametric Synthesis of Animation Sequences

Figure 4.63: Comparing the change of direction between initial and final frames, of each sample motion clip, with and without the proposed improvement.

quantitative evaluation is presented below for confirmation and compatibility with the rest of the thesis. Figure 4.64 depicts both the maximum and RMS values of the quantitative measures, the jerk and Feet – Floor Error. The quantitative measures shows the noticeable reduction of errors as a result of applying our improved blending technique.

In terms of storage, with comparison with Kovar et. al. [54], Kovar et. al. store the 2D transformation between every pair of corresponding frames for every pair of motions. This requires a storage of order $O(N^2(m - 1)n)$ where $m$ is the number of motion clips and $n$ is the number of corresponding frame pairs. In our case, the storage requirements are reduced to order of $O(mn)$ as depicted in figure 4.65.

4.7.7 Integrated Example

This section demonstrates an example of integrating results of more than one module. It shows the construction of longer animation sequence for a given scene.

First, the scene is analysed and the required motion is planned (manually). Then the planned motion is divided into smaller tasks such as motion on arbitrary path on level floor and motion on uneven terrain. Each task is achieved by the appropriate module
4.7 Results and Evaluation

Figure 4.64: The improvement in blending results in noticeable reduction in errors as shown by the quantitative measures.

Figure 4.65: Comparison of the storage requirements.
as discussed throughout this chapter. The important point is to keep track of the global location, orientation, and motion parameters at the end of each task. These information serve as initial conditions for the following task.

Figure 4.66 demonstrates an example of this process. The motion is decomposed into two tasks. The first task is to follow a path on a level-floor to a certain point. Then, seamlessly, move on a variable slope-floor to go up-hill.

Figure 4.66: Example for integrating the developed modules for longer sequence of real scene.

The first task is achieved by the PathControl module as described in section 4.4. The second task is achieved by the TerrainControl module (or more generally, using the SpeedTerrain control module. Especially if we need to change the speed as well). In the second task, motion starts from the end point of the previous task with the same speed and slope achieved. The motions generated from the two tasks are seamlessly joined by smooth transition. Figure 4.67 shows the constructed animation sequence rendered in the virtual scene.
4.8 Summary

In this chapter, the parametric synthesis approach introduced in chapter 3 has been extended for generating long and continuous animation sequences through editing short clips. This results in a simple -but effective- technique for achieving the following interesting tasks:

- **Path Control**: Intuitively synthesis motions on arbitrarily paths selected/drawn by the user (section 4.4).

- **Terrain Control**: Parametric synthesis of motions on uneven terrains provided by the user. (section 4.5).

- **Multi-Parameters Synthesis with enhanced parameterisation**: This reduces the number of required parameterisation functions. It allows the user to control more than one motion parameter simultaneously. It also works on the given *irregular* grid of input motions, on the parameters space, which avoids the pre-processing of modifying inputs to form a regular grid on the parameters space as discussed in section 4.6.
The additional problems of generating animation sequences are identified and tackled with the following novel contributions:

- **Motion cyclification** techniques [5] that employ simple animation processing algorithms and based on observation of characteristics of different categories of cyclic motions (section 4.3.1).

- **Improved root-trajectory Blending** technique which incrementally blend root trajectories relative to previous frames and maintain the proper magnitude of movement distance. It also avoids the sudden flipping of the root trajectory when difference in direction of input motions exceeds 180° (section 4.3.3).

- **Syncronised Mirror** technique for enriching the basic animation database with synchronised mirrored clips which is automatically annotated and ready for processing within the parametric approach. (section 4.3.4).

The proposed approach is extendable in the way discussed in section 4.6. However, the number of required animation clips can increase rapidly with the number of parameters which is a common limitation for interpolation techniques [101, 38]. This can result in a large animation database. Appendix A presents an additional module that is developed within our research [1] for adaptive compression of human animation data.

The next chapter provides a conclusion for the work presented throughout the thesis and discussion of its advantages, limitations and assumptions. It also shows our suggestions for possible future work for extending and enhancing the proposed animation system.
Chapter 5

Conclusions

In this chapter, the work presented through the thesis is summarised and concluded. The main goal in this research was to provide an intuitive high-level parametric control over the synthesis of human animation while maintaining animation realism through reuse of realistic input clips. This parametric approach is potentially useful in many applications such as games, film production, and 3D broadcasting.

In the following sections, the presented work is summarised in section 5.1 and the principal contributions of this research are emphasised in section 5.2. The advantages and limitations of the presented approach are discussed in section 5.3 along with a discussion of the assumptions made and their justification. Finally, the suggested future work is discussed in section 5.4.

5.1 Summary

This research was originally motivated by the fact that although realistic animation can be acquired using motion capture systems, the acquired data are difficult to edit or modify. Synthesis of an animation clip that is slightly different from the captured one, requires repeating the capturing process. Alternatively, editing motion, especially for captured data, may result in violating motion constraints and/or affecting its realism and quality.
Many existing motion editing techniques require an expert or trained animator to deal with complex models and/or define the desired motion by manipulating low-level variables. A few solutions have been introduced recently to provide high-level editing of animation data. These solutions are based on transition graphs or statistical models. Common drawbacks of these techniques include the need of relatively large database and the high computation complexity, which is non-linearly increases with the length of the desired animation sequence.

Our aim is to allow both novice and expert users to use the natural way of thinking and define their desired motions using the intuitive high-level motion parameters. The system will use the given high-level parameters to generate the required realistic motion. The proposed solution aimed to compromise between intuitive parametric control, quality of generated animation, and costs of generating these animations, together with limiting the amount of data required.

The parametric approach is flexible for integration with other higher-level layers such as path or motion planning and scripts which will provide the motion parameters to be synthesised (instead of the user/animator). This makes it useful in application areas such as autonomous agents and controlling virtual agents in virtual environments.

5.2 Contributions

The principal contribution of this research is the proposed high-level parametric approach for synthesis of realistic human animation based on synchronised blending of existing motion clips. The approach allows an intuitive and high-level parametric definition of the desired motions while generating realistic animation based on a small motion captured database of basic motion clips. The framework has been developed and utilised to realise interesting and challenging tasks which demonstrate its usability and usefulness.

Principal novel contributions can be summarised as follows:

• **Intuitive high-level parametric approach for synthesis of long animation sequences** according to arbitrarily time-varying parameters provided by the user (Chapter 4). This extended framework has been utilised to achieve interesting animation tasks with less data requirements and linear computation complexity with the length of the desired sequence. The following animation tasks have been successfully achieved:

  - **Path Control**: Intuitively synthesis motions on arbitrarily paths selected or drawn by the user (section 4.4).
  
  - **Terrain Control**: Parametric synthesis of motions on uneven terrains provided by the user. (section 4.5).

• The approach is extended to allow for **multi-parameters synthesis** with enhanced parameterisation technique that reduces the number of required parameterisation functions. It also works on the given *irregular* grid of input motions, on the parameters space, which avoids the pre-processing of modifying inputs to form a regular grid on parameters space as discussed in section 4.6.

In order to achieve the above contributions, especially for the long animation sequences, other contributions have been introduced at different stages of the synthesis process. These novel contributions are:

• **Automatic gait analysis** technique [5] that is used for automatic detection of key-event times, which are crucial for synchronising input motions together. This technique improves the automation of the proposed parametric framework so that user does not have to select the motion parts and label them manually. The developed technique is able, in most cases, to select the relevant parts of a given motion sequence that satisfies the user preferences such as the advanced foot at start of walking cycle, number of cycles, and number of key-event times, as discussed in section 3.4.3. Hence, the input to the system is not necessarily an exact motion cycle as the system will detect the cycle from longer sequence.
Chapter 5. Conclusions

- Novel motion cyclification techniques [5] that employ simple animation processing algorithms and based on observation of characteristics for different categories of cyclic motions (section 4.3.1).

- Novel animation processing technique (Synchronised Mirror technique) for enriching the basic animation database with synchronised mirrored clips which are automatically annotated and ready for processing within the parametric approach. (section 4.3.4).

- Improved root-trajectory Blending technique which incrementally blend the changes in root trajectories of input motions and maintain the proper magnitude of movement distance. It also avoids the sudden flipping of the root trajectory when difference in direction of input motions exceeds 180° (section 4.3.3).

Although there is a lot of interest in research in motion editing, the quantitative evaluation of the results has received little attention. In addition to the visual evaluation, which is the ultimate evaluation criteria in animation, we introduced and adopted quantitative measures to systematically and consistently evaluate our results and the efficiency of different developed modules. These measures are defined and used in sections 3.5, 4.7, and A.3 for quantitative evaluation of the achieved results. The quantitative measures have indicated that the quality of the generated motions is close to the quality of the original clips.

The proposed framework can also serve as a test bed for further research in human animation. It provides an easy way of generating various motions. Enriching or extending the existing animation database reduces the limitations of data availability. This alleviates the problem of obtaining new data which is expensive and time-consuming task.

To conclude, the presented parametric synthesis approach has been successfully applied for synthesis of short clips and long animation sequences. It has been utilised to achieve interesting and practical animation tasks such as controlling motion along arbitrary path and over arbitrary uneven terrain. The extension for multi-parameters synthesis has been also introduced and successfully utilised for controlling the speed of motion.
over arbitrary uneven terrain. Both qualitative and quantitative evaluations have indicated that the quality of the generated motions is close to the quality of the original motion clips.

The size of the animation data required for the presented parametric synthesis approach is relatively small. It requires few animation clips (one cycle each) with a total number of frames within hundreds frames. The presented framework computationally efficient, with computation complexity which is linearly related to the length of desired animation sequence.

5.3 Discussion

In this section the advantages and limitations of the proposed approach are discussed as well as the assumptions made. Along the thesis, the contributions at different stages of the system has been presented and compared with the relevant work. This has also been summarised in tables D.1 to D.5 (Appendix D) and discussed throughout this section.

A rapid and intuitively controlled parametric synthesis method for human animation sequences has been presented in this thesis. The proposed approach has reduced data requirements, reduced parameterisation functions, automatic determination of the mapping function, and efficient and linear computational complexity. Other contributions at different stages have been introduced such as the improved root-trajectories blend, synchronised mirror, and automatic cyclification techniques. Those contributions allow for more utilisation of the existing animation clips (which reduces data requirements), automatic synthesis of longer animation sequences from few basic short clips.

The presented method is useful in many applications. However, different applications have different requirements (including the level of realism of the generated animation). This varies from simple tutorial or presentation, games, up to 3D broadcasting and film production, with the latest often having the highest requirement of realism.

The approach presented in this thesis, with its current implementation and input data, is more suitable for games and presentations. For film production, a few developments
are required on top of the current approach, such as constraints enforcements (using IK as an example) as well as using more realistic input data.

The focus of the developed technique is on the cyclic movements (e.g. walking and running) and motions with similar pattern (e.g. kicking and reaching). For generating long sequences, the input short clips are extended (on demand) by repeating the motion cycle (assuming its boundaries are matched) to achieve the length of the required sequence. This may result in the lack of natural variations of human movement from one motion cycle to another. The style, mode and personality of the resulting motion are controlled by the set of input motion clips. Producing animation with different style and personality is a complete topic of research by itself and needs more investigation as planned in the future work.

In this research, it has been assumed that the input motions are structurally similar. This means that corresponding important events (or motion constraints) are occurring in the same order. Blending motions of dissimilar structure (e.g. walking and sitting-down or laying on the floor) does not guarantee a valid resulting motion. A simple example is depicted in figure 4.12. Although the two input motions are both walking motion, the key-event times are out-of-phase. Blending between those two motions produces an invalid motion as shown in the figure. For this particular category, where one motion is the mirror of the other one, a synchronised mirror technique has been developed within our research to overcome this limitation. However, the general case is still a common limitation for all existing interpolation and blending techniques.

Within the synchronised blending process, and in the estimation of the key-event times of the new motion, an assumption has been made that the key-event times scale linearly between the input motion clips. The limited analysis discussed in section 3.4.2 shows the high variability of key-event timing and that the assumption of linear approximation is reasonable. A large amount of data will be required for critical analysis of the key-event timing variations and for modelling this variation if possible.

Also, in estimating the key-event times of the synthesised motion (through interpolation of key event times of the input motion), there is a rounding error as frame numbers are always of integer values. Although this rounding error is slightly affecting the accuracy
5.4 Future work

of key-event timing (and values of some quantitative measures such as jerk), no visible artefacts have been noticed due to this error.

In the next section, suggestions for future research, to overcome the identified limitations and improve the capabilities of proposed system, are discussed.

5.4 Future work

In recent years, there has been reasonable increasing research in the motion editing with the aim of improving control and realism of the resulting animation and automating the common and time-consuming tasks. However, animating virtual humans remains a challenging task due to the complexity of the human skeleton hierarchy, the large number of degrees of freedom, the ability of human observers in detecting unnatural movements, and the complexity of the realistic appearance and behaviour. The complexity of the problem has resulted in research into many sub-problems of human animation synthesis. Currently, there is no single technique that provides a unified solution for all applications. Thus, integrating two or more techniques may produce better results in some cases.

Regarding the work proposed in this thesis, there are extensions that can be developed in the future. Ideas and directions for future work can be summarised as follows:

- Extending the approach to include variations of style, mode, and personality in the synthesised motions.

- Integrating with other approaches such as [67] to extend the existing animation database by providing transitional clips which may not be possible to realistically generate within the presented system such as transitions from lying down to walk or vise-versa.

- The presented system can be used as an engine that generate animation according to high-level parameters provided from other modules such as motion and path planning and behaviour controller.
Chapter 5. Conclusions

- Automatic classification of input motion to detect the motion type (e.g. walk, run, jump,...,etc.). Identifying the motion type is a pre-requisite for the detection of its constraints and hence the key-event times that are used for synchronising the input motions. Automatic classification or recognition of human motion shares a common interest with other research fields such as computer vision, surveillance, and human recognition. At the moment, our system can deal with motions of same categories in terms of matching the key-event times, parameterisation, and parametric synthesis of new motions. For future work, we aim to be able to identify the different classes of motions.

- Critical analysis of the variation of key-event timing and methods of modelling this variation.

- Investigating avoiding the rounding error in the key-event timing estimation.

In terms of development and implementation, there are some ideas for improving the integration between the developed modules:

- Implementing a suitable constraint-enforcement algorithm as a post-processing layer.

- Developing a task manager that can handle motion planning and control the various modules of the system in order to achieve the desired animation sequence based on a given virtual scene.

- Proper setup and management of the animation database and its layers and improving the user interaction with the database.

- Developing a module for importing and rendering virtual environments (including the resulting animation).

Finally, the proposed framework has been successfully implemented and achieved interesting and challenging animation tasks such as intuitive control of motion along an arbitrary path and on uneven terrain with variable speeds. The framework can be
extended in several ways and the above ideas give some examples of the possible extensions. It is also flexible for integration with other modules such as path planning or behaviour control.
In this appendix, we introduce an adaptive motion compression technique using the discrete wavelet transform [1]. Wavelet compression techniques are already used in image and video compression. However, to the best of our knowledge, the compression of human animation data has not been investigated.

One of the common problems of interpolation and blending techniques, for animation synthesis, is the increase of the size of the animation database required to characterise variations of movements. As the capabilities of these techniques are based on the available motion clips, the more variations in the resulting motion, the more motion clips will be required.

We addressed the compression of human animation as a solution for reducing the animation database size. It can be useful in many areas such as motion editing, blending and online games. In such situations, compressing the animation data, with minimum visual effect, can reduce the animation database size, reduce transmission load and help in the real-time performance with reduced cost.

Based on the analysis of human animation data, and its frequency content, the wavelet analysis is utilised to achieve high compression ratios (up to 86%) with small effect on the visual quality of the animation.

A brief introduction to wavelet transform, its properties and utilisation in compression are discussed in section A.1. The section also describes previous related research in
Appendix A. Adaptive Compression

using wavelets analysis in animation and different related fields and applications. The
proposed adaptive compression technique is described in section A.2. Then, the results
and evaluation are shown in section A.3. Finally, a summary of the chapter is given in
section A.4.

A.1 Wavelets and Compression

Wavelet transform provides an alternative representation of the signal. It has some
useful properties which include the localisation in time and frequency, the ability of
handling non-periodic and non-stationary signals [7], and the easy extendability to
higher dimensions. Wavelet transform is also adaptable, as wavelets could be designed
to match specific applications.

There are also two properties that makes wavelet transform a successful candidates
for compression. Firstly, the rapid decrease in the number of wavelet coefficients with
the resolution level. Going towards a lower resolution level will be accompanied with
decrease of the number of wavelet coefficients used to represent the signal. The second
useful property of the wavelet transform is the efficient numerical calculation. Wavelet
transform coefficients can be efficiently calculated using the filter bank algorithm which
incorporates only multiplication and addition operations. In most cases, the calculation
time is linearly related to the length of the given signal [7]. The computation complexity
of the fast wavelet transform is $O(n)$ compared to $O(n \log_2(n))$ for the Fast Fourier
Transform [92], where $n$ is the signal length.

One of the important features of wavelet transform is the 'Multiresolution Analysis'
(MRA) property. MRA means that the signal can be analysed at different frequencies
with different resolutions. Following is a brief presentation of the MRA property of
the wavelet transform. For more details we suggest the reader to refer to appendix ??
and [6, 7, 23, 27, 66].

In the wavelet transform, the signal $S$ at a resolution level $j$ can be written as:

$$S_j(t) = \sum_k C_{j-1,k} \varphi_{j-1,k}(t) + \sum_k D_{j-1,k} \psi_{j-1,k}(t)$$  \hspace{1cm} (A.1)
A.1. Wavelets and Compression

where: $\varphi_{j-1,k}(t)$ are called the scaling functions.

$\psi_{j-1,k}(t)$ are called the wavelet functions.

$C_{j-1,k}$ are the scaling functions coefficients at resolution level $j-1$.

$D_{j-1,k}$ are the wavelet functions coefficients at resolution level $j-1$.

$j$ is the resolution level.

$k$ is the index of the scaling function within a certain resolution level.

Using the nested space concept, the signal $S$ can be represented as follows:

$$S_j(t) = \sum_k C_{0,k} \varphi_{0,k}(t) + \sum_{m=0}^{j-1} \sum_k D_{m,k} \psi_{m,k}(t) \quad (A.2)$$

The sequence $C_0, D_0, D_1, ..., D_{j-1}$ is called the wavelet representation of the signal $S$. $C_0, D_0, D_1, ..., D_{j-1}$ are called the wavelet transform coefficients. In matrix form:

$$S_j = C_j = [P_j|Q_j][C_{j-1}] \quad (A.3)$$

This expression is called the synthesis expression, which is used to reconstruct or reproduce (or synthesise) the higher resolution level $j$ from its lower resolution level $j-1$ as shown in figure A.1. $P$ and $Q$ are called the synthesis matrices. Figure A.2 depicts the analysis (or decomposition) process of the signal which can be formulated as:

$$C_{j-1} = A_j C_j$$

and

$$D_{j-1} = B_j C_j \quad (A.4)$$

where

$$[A_j \ B_j] = [P_j|Q_j]^{-1}$$

Figure A.1: Wavelet reconstruction/synthesis.
Wavelet analysis has been used in computer graphics applications such as multiresolution curve editing [32], image processing and compression [86]. However, there is little previous research in applying wavelet analysis to human motion animation.

Lin et al. [64] employed wavelets to accelerate the space-time interpolation of their physically-based keyframe animation. On the other hand, Sun [91] used the wavelets to model bipedal locomotion. Wavelets have been used also for smoothing kinematic motion data in the biomechanics studies [8]. Multiresolution analysis using different techniques has been explored in [22, 48, 60]. However, to the best of our knowledge, the compression issue has not been investigated.

In [4], we introduced a parametric motion blending approach using wavelet interpolation and blending. Analysis of the motion curves and its frequency contents has shown that we can utilise the wavelet transform to achieve a reasonable level of data compression without destroying the quality of the generated animation [1], as shown in sections A.2 and A.3.

The following section presents the proposed adaptive compression technique that utilises the wavelet transform and its useful properties to compress animation data.

A.2 Our Adaptive Compression

The wavelet compression technique is based on using the minimum number of wavelet transform coefficients to reproduce or synthesise a signal that is as close as possible to the original signal. In the simplest form, this can be done simply by discarding the detail coefficients $D_{j-1}$ then $D_{j-2}$ and so on according to the required compression ratio or acceptable error tolerance. This assumes that the signal contents are more
A.2. Our Adaptive Compression

concentrated in the coarse level (i.e. low-frequency contents). Bruderlin et. al. [8] reported that the waving and knocking motions contain mainly low-frequency contents. We have extended the analysis to some other human motions such as walking, running, dancing, and fighting as well. It has been found that this assumption is valid in many basic human motions. Discarding some of the small details coefficients is found not to disturb the motion realism as explained in the following subsection and shown in the results, section A.3.

A.2.1 Human Animation data

Human animation data is usually represented by a 3D translation of the root of the skeleton hierarchy plus 3D rotation of each degree of freedom (DOF) of the hierarchy joints, including the root joint. So, each motion curve is a signal representing the rotation about one of the 3D axis.

Figures A.3 shows the frequency contents for some samples of different human motions. It can be seen that most of the signal contents are concentrated in the low-frequency range. This supports the assumption of compression by discarding some detail coefficients.

However, the behaviour of the motion curves or signals is not necessarily similar. It varies from one DOF to the next and also differs from one motion to another. According to that, using a constant compression ratio for all DOFs signals (i.e. discarding the same number of coefficients) may introduce a large amount of errors and the resulting motion may appear unrealistic (or even appear incorrect in higher compression ratios).

Our proposed technique is an adaptive compression for human animation data as discussed in the following subsection.

A.2.2 Compression

Our approach is to use an adaptive compression, which means that each signal can be compressed with different compression ratio that maintains a limited error. So, according to the behaviour of each signal, some signals may have high compression
Appendix A. Adaptive Compression

Figure A.3: Frequency-content's analysis of different human motions.

ratios, and others may have small compression ratios. However, the overall compression ratio is based on the overall number of wavelet coefficients that are used to reconstruct the motion. Hence, the overall compression ratio $R$ is defined as:

$$R = \frac{C_d}{C_a}$$

where $C_d$ is the number of discarded wavelet coefficients and $C_a$ is the total number of wavelet coefficients.

The advantage of using this adaptive compression is that it limits the individual signals error, which provides better results in terms of less visual effects with reasonable overall compression ratios.

The wavelet analysis is divided into two phases. The analysis or decomposition phase and the synthesis or reconstruction phase. In the analysis phase, the motion signal
A.2. Our Adaptive Compression

is decomposed into its wavelet representation $C_0, D_0, D_1, \ldots, D_{j-1}$. In our implementation, cubic B-Spline wavelets are used as recommended in the computer graphics literature [32].

In the synthesis phase, the signal is reconstructed from its wavelet representation. For perfect reconstruction of the original signal, the full wavelet transform coefficients should be used. It is worth noting that the full number of wavelet transform coefficient is equal to the signal length. For compression, assuming that we have $m$ wavelet coefficients, we need to reproduce the signal using $n$ coefficients where $n < m$.

The number of used coefficients $n$ can be determined either explicitly, by discarding a certain number of details-level coefficients, or using some selection criteria. For example, we can discard the coefficients so that the total error is within some specified value. Given a predefined error value $E$, it is required to find the minimum number of wavelet coefficients $n$ that represents the given signal with error less than (or equal to) $E$ as depicted in the equation below:

$$\sum_{i=1}^{k} (C_i^x)^2 \leq E^2$$  \hspace{1cm} (A.5)

where: $C_i^x$ are the discarded coefficients.

$E$ is the predefined error tolerance.

$k = m - n$; is the number of discarded coefficients.

This can be implemented in different ways. One way is to sort the coefficients in a descending order (based on their magnitude) and then discard the least significant coefficients such that their sum is less than or equal to the specified error. However, for large motion clips, sorting the coefficients will slow down the process, keeping in mind that we need to keep the original order of coefficients for correct synthesis.

Another more efficient way is to use a binary search algorithm [86] to find the coefficients to be discarded without the requirement to sort all the coefficients beforehand. Our adaptive technique adopts the second option of compression with the binary search algorithm.
The next section presents the results achieved using the proposed adaptive compression technique. It also shows the evaluation with the introduced quantitative measures.

A.3 Results and Evaluation

As mentioned in the results sections of previous chapters (3.5 and 4.7), the evaluation of how realistic the animation is, is not an easy process. It is mainly qualitative evaluation. Thus, the visual evaluation is the main criteria as it represents what the user will actually perceive. However, for more consistent evaluation, some quantitative measures are introduced to determine the quality of the compressed motions and to measure the effect of compression.

For the work in this chapter, the advantage is that we have a ground truth reference to compare with, which is the original motion. Hence, the straight-forward evaluation is to compare the compressed motion with respect to the original (un-compressed) motion. For this purpose, we introduced the '3D - Shift' in positions as a quantitative measure.

The idea of the '3D - Shift' measure is to calculate the difference in the 3D position, for each joint in the skeleton hierarchy, at each frame. This measure indicates how the joints of the compressed motion are shifted from their positions in the original motion. It can be formulated as:

\[ \varepsilon_f = \sum_{j=1}^{N} |\Gamma_{j,f} - \Gamma_{j,f}^o| \]  

(A.6)

where: 
\( \varepsilon_f \) is the total 3D shift in positions at frame \( f \).  
\( \Gamma_{j,f} \) is the position of joint \( j \) at frame \( f \) of the compressed motion.  
\( \Gamma_{j,f}^o \) is the position of joint \( j \) at frame \( f \) of the original motion.  
\( N \) is the number of joints in the skeleton hierarchy.

The measure is calculated at each frame. But for getting an indication of the error variation along the whole animation, the \( RMS \) value of the calculated measures is taken as:
A.3. Results and Evaluation

\[
\epsilon = \sqrt{\frac{1}{F} \sum_{i=1}^{F} \frac{e_i^2}{F}}
\]

(A.7)

where \( F \) is the total number of frames in the given animation clip.

The maximum value is also calculated and has been found that it has the same behaviour as shown in the examples below.

Figure A.4 shows two examples of compressed ‘Walk’ animation along with their original animation. The compression ratios are 69% and 87% respectively. The quantitative evaluation uses the 3D–Shift measure introduced in equation A.7 as shown in figure A.5. Figure A.5 indicates that for a compression of up to 86%, the accumulated 3D–Shift in positions of all the skeleton joints is less than 7 inches (max of 12 inches). More importantly, the visual inspection shows that the compressed motion is close enough to its original motion.

![Figure A.4: Example of compression of 'Walk' animation. Original (right), compressed with 69% (middle), and compressed with 87% (left) clips.](image)

A similar example for ‘Run’ motion is shown in figure A.6. Also, two compressed exam-
Appendix A. Adaptive Compression

Figure A.5: Quantitative Evaluation of compression of the ‘Walk’ example.

Examples with two different compression ratios, 69% and 77%, are shown with the original animation. The quantitative evaluation using the $3D-Shift$ measure is depicted in figure A.7. It can be seen that with up to 70% of compression, the accumulated $3D-Shift$ in positions of all the skeleton joints is less than 13 inches (max of 20 inches). More importantly, visually, the compressed motion is close to its original motion.

Figure A.6: Example of compression of ‘Run’ animation. Original (right), compressed with 69% (middle), and compressed with 77% (left) clips.

Another interesting example is depicted in figure A.8 for a BalletDance animation clip. The figure shows the original and a compressed version of the animation with 79%.
A.3. Results and Evaluation

Figure A.7: Quantitative Evaluation of compression of the 'Run' example.

c. RMS of total $3D - Shift$

b. Maximum of total $3D - Shift$

 compression ratio. Quantitatively, the 79% compressed motion results in a maximum total $3D - Shift$, for all joint positions, of 11.9 inches ($RMS$ of 5.6 inches). The visual inspection showed that the compressed motion is close enough to the original to the extent that it was difficult to distinguish between them.

Figure A.8: Compression of BalletDance animation. Original (right) and compressed with 79% (left) clips.

For compatibility with the evaluation criteria we have used along previous chapters, the jerk and feet–floor contact error measures are calculated for the presented examples as depicted in figures A.9 and A.10. It can be seen that within the visually accepted compression range (up to 86%), these quantitative measures are close to the values of the original animation. This also explains the difficulty of distinguishing between
Figure A.9: Previous quantitative measures, applied on ‘Walk’ example, are shown for compatibility with other modules.

Figure A.10: Previous quantitative measures, applied on ‘Run’ example, are shown for compatibility with other modules.
original and compressed animations in many cases.

The next section summaries the contribution introduced in this chapter.

**A.4 Summary**

In this appendix, our adaptive motion compression technique [1] is introduced. The proposed technique utilises the discrete wavelet transform for compression. Based on the analysis of human animation data, and its frequency contents, the wavelet analysis has been utilised to achieve high compression ratios, up to 86%. The compressed animation is visually close to the original animation. This visual evaluation is also supported by the quantitative evaluation. For quantitative evaluation, we introduced the 3D – Shift in joints positions as a measure of how far the compressed animation will shift from its original positions.
Appendix B

Wavelet and Direct Blending

In this appendix, both wavelet and direct blending are analytically compared for the purpose of synthesis of human animation. The advantages and disadvantages of each are presented at the end of this appendix.

B.1 Direct Blending

Motion blending is an operation that employs the interpolation between two, or more, motion data in order to produce a new motion that is related somehow to the input motions. For two motions $M_1$ and $M_2$, the simplest form of the blending operation is the direct blending which employs the linear interpolation, or weighted sum, between the two motion curves using a formula like:

$$M = \alpha M_1 + (1 - \alpha)M_2$$

where: $M$ is the new motion generated by blending.  
$\alpha$ is the blending factor.  
$M_1, M_2$ are the given two input motions.

The similarity between the synthesised motion and the original motions is controlled by the value of the blending factor $\alpha$ in the above formula.
The above general formula can be re-written for each degree of freedom as a time-varying signal as follows:

\[ S(t) = \alpha S^1(t) + (1 - \alpha)S^2(t) \]  

(B.1)

where: \( S(t) \) is the new motion signal generated by blending, for specific degree of freedom. 
\( \alpha \) is the blending factor. 
\( S^1(t), S^2(t) \) are the given motion signals for the same degree of freedom.

B.2 Wavelet Blending

In wavelet transform representation, the signal \( S(t) \) is represented by the scaling and wavelet functions and their coefficients such as:

\[ S_j(t) = \sum_k C_{0,k} \varphi_{0,k}(t) + \sum_{m=0}^{j-1} \sum_k D_{m,k} \psi_{m,k}(t) \]  

(B.2)

where: \( \varphi_{m,k}(t) \) are the scaling functions at resolution level \( m \). 
\( \psi_{m,k}(t) \) are the wavelet functions at resolution level \( m \). 
\( C_{0,k} \) are the scaling functions coefficients at resolution level ‘0’, the coarsest resolution level. 
\( D_{m,k} \) are the wavelet functions coefficients at resolution level \( m \). 
\( j \) is the number of resolution levels. 
\( k \) is the index of the scaling function within a certain resolution level.

Hence, given two motions \( M_1 \) and \( M_2 \), the signals, of a certain degree of freedom, from the given two motions can be written as:

\[ S_j^1(t) = \sum_k C_{0,k}^1 \varphi_{0,k}(t) + \sum_{m=0}^{j-1} \sum_k D_{m,k}^1 \psi_{m,k}(t) \]  

(B.2)

\[ S_j^2(t) = \sum_k C_{0,k}^2 \varphi_{0,k}(t) + \sum_{m=0}^{j-1} \sum_k D_{m,k}^2 \psi_{m,k}(t) \]  

(B.3)
The wavelet blending is formulated as:

\[ S_j(t) = \sum_k [\alpha_0 C_{0,k}^1 + (1 - \alpha_0) C_{0,k}^2 \phi_{0,k}(t)] + \sum_{m=0}^{j-1} \sum_k [\alpha_m D_{m,k}^1 + (1 - \alpha_m) D_{m,k}^2 \psi_{m,k}(t)] \]

\[ S_j(t) = \sum_k \alpha_0 C_{0,k}^1 \phi_{0,k}(t) + \sum_k (1 - \alpha_0) C_{0,k}^2 \phi_{0,k}(t) + \sum_{m=0}^{j-1} \sum_k \alpha_m D_{m,k}^1 \psi_{m,k}(t) \]

\[ + \sum_{m=0}^{j-1} \sum_k (1 - \alpha_m) D_{m,k}^2 \psi_{m,k}(t) \]  

(B.4)

As we can see, the wavelet blending offers the flexibility of blending the various resolution levels \( m \) of the signal with different blending factors \( \alpha \). For the special case of using the same blending factor for all resolution levels, i.e., \( \alpha_0 = \alpha_m = \alpha \) for \( m = 0, 1, \ldots, j \), equation B.4 can be re-written as follows:

\[ S_j(t) = \alpha \sum_k C_{0,k}^1 \phi_{0,k}(t) + \sum_{m=0}^{j-1} \sum_k D_{m,k}^1 \psi_{m,k}(t) \]

\[ + (1 - \alpha) \sum_k C_{0,k}^2 \phi_{0,k}(t) \]

\[ + \sum_{m=0}^{j-1} \sum_k D_{m,k}^2 \psi_{m,k}(t) \]  

(B.5)

Substituting from equations B.2 and B.3, equation B.5 becomes:

\[ S_j(t) = \alpha S_j^1(t) + (1 - \alpha) S_j^2(t) \]  

(B.6)

By comparing equations B.6 and B.1, with the assumption that \( 'j' \) is the highest resolution level that is equivalent to the original signal (i.e. \( S_j^1(t) = S^n(t) \) for \( n = 1, 2 \)), we can see that the wavelet blending produces the same results as the direct blending in this special case.

According to the above discussion, in the special case of using the same blending factor for all resolution levels, the wavelet blending is equivalent to the direct blending (linear interpolation in the time domain). However, its computation cost is more expensive than the direct blending due to the overhead of forward and backward transformation. The table below presented a brief comparison between the direct and wavelet blending.
Appendix B. Wavelet and Direct Blending

<table>
<thead>
<tr>
<th></th>
<th>Direct Blending</th>
<th>Wavelet Blending</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>Time</td>
<td>Time and Frequency</td>
</tr>
<tr>
<td>MRA</td>
<td>N/A</td>
<td>Multiresolution Analysis available</td>
</tr>
<tr>
<td>Single Blending Factor</td>
<td>Yes</td>
<td>Yes a</td>
</tr>
<tr>
<td>Multi-Blending Factors</td>
<td>N/A</td>
<td>Yes. Up to the number of resolution levels</td>
</tr>
<tr>
<td>Compression</td>
<td>N/A</td>
<td>High compression ratios, up to 86% as discussed in chapter A.</td>
</tr>
<tr>
<td>Performance</td>
<td>Faster</td>
<td>Slower due to the overhead of transformation forward and backward. However, applying parallel processing techniques is expected to improve the performance [29].</td>
</tr>
<tr>
<td>Others</td>
<td></td>
<td>Watermarking, for copyright protection, and other applications that benefit from the multiresolution property.</td>
</tr>
</tbody>
</table>

aIn the special case of single blending factor, wavelet blending is equivalent to the direct blending in terms of the results.

Table B.1: Table of comparison between Direct and Wavelet Blending.

B.3 Summary

The discussion presented in this appendix shows that, in the special case of using the same blending factor for all resolution levels, the wavelet blending is equivalent to the direct blending (linear interpolation in the time domain). However, its computation cost is more expensive than the direct blending due to the overhead of transformation forward and backward. A brief comparison is given in table B.1.
Appendix C

Qualitative Evaluation

Qualitative evaluation of animation is the ultimate evaluation criteria of its quality. This is because it represents how the viewer of the animation perceives it. In this appendix, our qualitative evaluation experiment is described and results are presented. The appendix is concluded by a discussion of some considerations regarding the qualitative evaluation and the conducted experiment.

C.1 Methodology

The main aim of this experiment was to subjectively evaluate the quality of the synthesised animation sequences (outputs), relative to the quality of the real input clips.

For this purpose, the set of animation sequences presented for evaluation consists of a mixture of both real clips and synthesised sequences. The order of playing the animation sequences was randomly selected. The animation sequences are also selected to have various lengths. The mixture, random order, and various lengths of animation sequences are selected to make the evaluation process blind for the user and avoid the user’s bias to either real or synthesised sequences.

For evaluation, a couple of subjective measures have been defined for the users to evaluate. These measures are defined so that different users will be measuring roughly
on the same criteria. However, users have also been asked to record their personal overall judgment or rating for the quality of each animation sequence.

The subjective measures include the smoothness of the animation sequences and how much it suffers from artefacts or flickers. The definition of these measures is stated at the bottom of the evaluation form given to the user (figure C.1), and is introduced to each user before starting the evaluation session. By flickers, we intended to spot the relatively large amplitude and noticeable sudden movements, of some parts of the body, that occurs in a short time period. These sudden movements usually look odd from the natural motion. By smoothness, we intended to spot the relatively small amplitude movements that could be occurring for longer time periods.

Users have conducted the evaluation session individually. There were no training for the users beforehand. However, a brief explanation of the defined subjective measures, the objective of the experiment, and familiarisation with the experiment's setup is introduced to the users just before starting the evaluation session. During the evaluation session, there were no monitoring or guidance from us. The experiment setup allows the user to replay any animation sequence for any number of times. This has been found to be useful, especially for evaluating the short sequences.

A group of 15 users from within our research centre has kindly participated in this experiment. About the third of the participants is doing research in 3D graphics and/or animation related to the human body and motion. Some of the other participants like to play games and/or watch movies, including the animated movies (where human animation are heavily involved).

Participant's details, including name and e-mail, were absolutely optional. A copy of the evaluation form, designed and used in this experiment, is shown in figure C.1.

The next section presents the processing of the collected data and the results of the experiment.
### Visual Evaluation Form

<table>
<thead>
<tr>
<th>Clip#</th>
<th>Flickers</th>
<th>Smoothness</th>
<th>Realistic/Human</th>
<th>Overall Rating</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Flickers:** Any sudden changes in the movement (1=very flickering, 5=Almost no flickers)

**Smooth:** Unnoisy movements (1=very noisy, 5=very smooth)

**Realistic:** How much do you feel it is natural human movement

**Overall Rating:** Your overall evaluation/judgement

**Rating code:**

- 1 Very Poor
- 2 Poor
- 3 Fair/Normal
- 4 Good
- 5 Very Good

---

Figure C.1: The Evaluation form used by the participants for qualitative evaluation.
C.2 Results

As mentioned in the previous section, we had 15 participants in the experiment who conducted the evaluation session individually. The form collected from each user after the session and data entered on a spreadsheet for processing.

Within the data processing stage, we first identify the real and synthesised sequences using a pre-prepared mapping table. The data collected from each form is analysed by averaging the rating of each measure, including the user’s overall rating, for both real and synthetic sequences. The distributions of the rating of both real (inputs) and synthetic (outputs) sequences are plotted and compared as shown in figure C.2.

![Graphs showing rating distributions for overall, realism, flickering, and smoothness ratings.]

Figure C.2: The rating distribution for both input and output animation sequences.

As depicted in figure C.2, input and output sequences have a similar distribution of the rating. This indicates that the quality of the synthesised sequences, as perceived by the users, is close to the quality of the input real clips used in the synthesis process.
C.3 Discussion

In this section, we discuss some considerations of the conducted qualitative evaluation experiment. This includes the participant’s experience, number of participants, and the data processing and analysis.

User experience: Usually, expert users (experienced animators) should be conducting the evaluation. However, in our case, the main objective is to find out how people will perceive the animation. They are not necessarily expert people, as viewers of animation are usually unskilled viewers. This is the case in many applications such as games and animated movies. The skilled users, such as animators, are a relatively small group who are involved in the production. It is worth mentioning that the main evaluation is by comparing the quality of the real and synthetic animation sequences. Hence, the evaluation of the absolute quality (if possible) of the synthetic sequences is not the goal in this experiment.

Number of participants: 15 users from within our research centre have kindly been participated in our experiment. It would have been better to involve more people. However, due to the time and cost limitations of the PhD project, the number of participants has been limited to some extent.

Data processing and analysis: In terms of processing and analysis the collected data, a simple averaging technique has been utilised. More advanced analysis techniques may be applied for better evaluation but the improvement is worthy when more participants are involved.
Appendix D

Comparisons

Along the thesis, the contributions at different stages of the system has been compared with the relevant work. This has been summarised in tables D.1 to D.5 below.
### Table D.1: Table of comparison between the proposed approach and the most relevant work (continued in the next page)

<table>
<thead>
<tr>
<th>Proposed Approach</th>
<th>Typical</th>
<th>Physical Parameters [76, 77, 78]</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-house motion capture data</td>
<td>Pre-defined criteria.</td>
<td>Function (RBF) interpolation [79, 77, 79]</td>
</tr>
<tr>
<td></td>
<td>Linear interpolation</td>
<td>Linear interpolation [79, 77, 79]</td>
</tr>
<tr>
<td></td>
<td>Searching transition graphs [9, 10, 65]</td>
<td>Searching transition graphs [9, 10, 65]</td>
</tr>
<tr>
<td></td>
<td>Ahmed et al. [14, 12, 3]</td>
<td>Ahmed et al. [14, 12, 3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rose et al. [77, 77, 77, 77, 77]</td>
</tr>
</tbody>
</table>

**Data Source**

- In-house motion capture data (300-000 frames in [69])
- Total input of only 300 frames
- Preserved results are using a pre-defined model
- Linear interpolation
- Searching transition graphs
- Ahmed et al. [14, 12, 3]
- Rose et al. [77, 77, 77, 77, 77]

**Implementation**

- Pre-defined criteria. (RBF) Interpolation [79, 77, 79]
- Linear interpolation [79, 77, 79]
- Searching transition graphs [9, 10, 65]
- Ahmed et al. [14, 12, 3]
- Rose et al. [77, 77, 77, 77, 77]
<table>
<thead>
<tr>
<th>References</th>
<th>Interpolation/Blending</th>
<th>Statistical and Transition Graphs</th>
<th>Proposed Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rose et. al [77, 79], Park et. al [73] Kovar et. al [54]</td>
<td>Molina-Tango et. al [68], Arik et. al [9, 10], Kovar et. al [55]</td>
<td>Ahmed et. al [4, 5, 2, 3]</td>
<td></td>
</tr>
<tr>
<td><strong>Parameterisation</strong></td>
<td>In [77, 54], the blinding weights are used, assuming a linear relationship between the parameter and the blending weights. In [79] parameterisation is simulated indirectly by iteratively inserting pseudo clips that satisfy the given parameters if the error exceeds a defined threshold. This requires recalculation of the whole RBF arguments to reflect the new setup. The RBF interpolation requires 'T' mapping functions regardless of the number of controlled parameters, where 'T' is the number of input clips.</td>
<td>Search process is carried out at each frame to obtain the best matching posture for the required sequence.</td>
<td>Automatic parameterisation by solving the inverse-interpolation problem to obtain the mapping function(s).</td>
</tr>
<tr>
<td><strong>Single Parameter</strong></td>
<td>YES.</td>
<td>YES.</td>
<td>YES.</td>
</tr>
</tbody>
</table>

Table D.2: Continue of comparison between the proposed approach and the most relevant work (*continued in the next page*).
Table D.3: Outline of comparison between the proposed approach and the most relevant work (continued on the next page).

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Genes</strong></td>
<td>Required synthetic set of genes in the statistical model, number of frames in the motion, and I is the number of clusters in the database.</td>
</tr>
<tr>
<td><strong>Comparisons</strong></td>
<td>Table complexity of $O(T^2)$, where $T$ is the number of sequences in the database.</td>
</tr>
<tr>
<td><strong>Trains</strong></td>
<td>Process each frame which increases exponential complexity due to the search process.</td>
</tr>
<tr>
<td><strong>YES</strong></td>
<td>$O(T^2)$, where $T$ is the number of clusters in the statistical model.</td>
</tr>
<tr>
<td><strong>NO</strong></td>
<td>$O(T^2)$ except in [54].</td>
</tr>
<tr>
<td><strong>References</strong></td>
<td>Proposed Approach</td>
</tr>
<tr>
<td></td>
<td>Proposed Approach</td>
</tr>
<tr>
<td>References</td>
<td>Interpolation/Blending</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td></td>
<td>Rose et. al [77, 79], Park et. al [73], Kovar et. al [54]</td>
</tr>
<tr>
<td>Short Clips Synthesis</td>
<td>YES.</td>
</tr>
<tr>
<td>Subjective Evaluation</td>
<td>Not availablec.</td>
</tr>
<tr>
<td>Quantitative Evaluation</td>
<td>Not discussed</td>
</tr>
</tbody>
</table>

Table D.4: Continue of comparison between the proposed approach and the most relevant work (continued in the next page).

---

aDifferences with our parameterisation and assumptions are discussed in chapters 2 and 4.

bIt is worth mentioning that most of the work in this comparison has been developed in parallel with ours. However, our core idea of the parameteric synthesis has been developed and published earlier in [4].

cThe quality of the resulting animation depends heavily on the quality of the input animations which are different in each research or technique (no standard or unified inputs that can be used as a reference for comparison). There are also differences in the human models, the quality of the 3D meshed models, and the rendering quality for representing the results. These factors are not the same in all research methods which affects the evaluation.
Table D.5: Continue of comparison between the proposed approach and the most relevant work.

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>can handle dissimilar motions</td>
<td>Input motions are assumed to be structurally similar. The main focus is on the cycle (or motion) with similar patterns.</td>
</tr>
<tr>
<td>more suitable for large number of parameters and input chips</td>
<td>Focus is on the cycle (or motion) where the unnecessary parameters are assumed to be zero.</td>
</tr>
<tr>
<td>applicable for both cyclic and acyclic models</td>
<td>Input motions are assumed to be structurally similar. The main focus is on the cycle (or motion) with similar patterns.</td>
</tr>
<tr>
<td>more convenient for relatively small number of parameters (1-4) as it avoids the unnecessary variables</td>
<td>Due to a loose selection criteria, some input motions were required to be structurally similar.</td>
</tr>
</tbody>
</table>

Proposed Approach

- Statistical and correlation
- Interpolation/blending

References
Bibliography


