Modelling, Tracking and Generating Human Interaction Behaviours in Video

Thesis submitted to Cardiff University in candidature for the degree of Doctor of Philosophy.

Yue Zheng

Centre of Digital Signal Processing
Cardiff University
2008

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed.....(candidate) Date 12/12/08

STATEMENT 1

This thesis is being submitted in partial fulfillment of the requirements for the degree of PhD.

Signed ..... (candidate) Date 12/12/08

STATEMENT 2

This thesis is the result of my own investigation, except where otherwise stated. Other sources are acknowledged by giving explicit reference.

Signed ..... (candidate) Date 12/12/08

STATEMENT 3

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organizations.

Signed ..... (candidate) Date 12/12/08

ABSTRACT

Intelligent virtual characters are becoming increasingly popular in entertainment, educational and simulation software. A virtual character is the creation or re-creation of a human being in an image, using computer-generated imagery. It must act and react in the environment, drawing on the disciplines of automated reasoning and planning. Creating characters with human-like behaviours that respond interactively to a real person in a video, is still a serious challenge. There are several major reasons for this. First, human motion is very complex, which makes it particularly difficult to simulate. Second, the human form is also not straightforward to design due to the large number of degrees of freedom of the motion. Third, creating novel contextual movements for virtual characters in real time is a new research area.

The research described in this thesis addresses these problems and presents novel model-based approaches to create a three dimensional (3D) virtual interactive character. In other words it can respond in a realistic and sensible manner to actions of a real person in video. To this end, a virtual character generating system is developed. The system tracks and analyses the behaviour of a real person in a video input and thereby produces a fully articulated 3D character interacting with the person in the video input. Experimental results demonstrate that the simulated behaviours are very close to those of real people.

Furthermore, in order to enhance the tracking capabilities of the algorithm, a novel technique that splits the complex motion data in an automated way has been developed. This results in an improved model of the human motion. Indeed, experimental results confirmed that the model produced, using the above technique, can provide more accurate tracking results than the model trained on all the whole data at hand.

ACKNOWLEDGEMENTS

I would like to thank my PhD supervisors, Dr Yulia Hicks, Dr Dave Marshall and Prof. Jonathon Chambers, for their ideas, suggestions and patience in helping me to complete this work. It would have been an impossible task without their advice and support.

I would like to thank my colleagues from Centre of Digital Signal Processing for helping me to collect the motion data for the experiments in this work.

Finally, I would like to thank my family for their unconditional care, support and encouragement through the whole study. I thank my boyfriend, Dr. Qiang Yu, for his unremitting support. Without their love and support, I would not be where I am now.

Acronyms

2D	Two Dimensional
3D	Three Dimensional
APF	Annealed Particle Filtering
BN	Bayesian Network
CCD	Charge Couple Device
CHMMs	Coupled Hidden Markov Models
EKF	The Extended Kalman Filter
EM	Expectation-Maximisation
EVD	Eigenvalue Decomposition
fps	frames per second
GMMs	Gaussian Mixture Models
GPLVM	Gaussian Process Latent Variable Model
GP	Gaussian Process
HMMs	Hidden Markov Models
IK	Inverse Kinematics

KF	Kalman Filter
LED	Light Emitting Diode
MCMC	Markov Chain Monte Carlo
MoCap	Motion Capture
MRF	Markov Random Field
PCA	Principal Component Analysis
PDF	Probability Density Function
PPCA	Probabilistic Principal Component Analysis
RGB	Red, Green and Blue
RJMCMC	Reversible-Jump Markov chain Monte Carlo
RPF	The Regularised Particle Filter
SA	Simulated Annealing
SGPLVM	Scaled Gaussian Process Latent Variable Model
SIR	The Sampling Importance Resampling
SIS	The Sequential Importance Sampling
SVD	Singular Value Decomposition
VLMM	Variable Length Markov Model

CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS v

ACRONYMS vi

LIST OF FIGURES xiii

LIST OF TABLES xxiii

1 INTRODUCTION 1

1.1 Thesis Overview 4

1.2 Main Contributions 6

1.3 List of Publications 6

2 INTERACTIVE BEHAVIOURS: A REVIEW 8

2.1 Modelling the Motion of the Human Body 9

2.2 Reducing the Dimensionality of the State Space 14

2.3 2D vs. 3D Tracking 16

2.4 Visual Tracking Methods in Real Video 17

2.5 Interactive Behaviour between Two People 23

2.6 Generating A Computer Graphics Character 26

2.7 Summary 28

3 VIRTUAL CHARACTER GENERATING SYSTEM OVERVIEW 29

3.1 Acquisition of 3D MoCap Data 32

3.2 Learning a Model of Human Motion for Tracking Motion of a real person in video 33

3.3 Tracking a 3D Person in Real Video 34

3.4 Learning a Model of Human Motion for Generating Interactive Behaviours 36

3.5 Generating Interactive Behaviours for a Virtual Character Responding to the Tracked Person 36

3.6 Summary 37

4 A MODEL OF HUMAN MOTION FOR A SINGLE PERSON 38

4.1 Motion Capture Systems 39

4.2 3D MoCap Data 43

4.3 A Model of Human Body Motion 45

4.3.1 Principal Component Analysis: MoCap Data Dimension Reduction 47

4.3.2 Gaussian Mixture Models 51

4.3.3 Hidden Markov Models 54

4.3.4 Training a Model of Dynamics 58

4.4 Summary 59

5 TRACKING A 3D PERSON IN A 2D REAL VIDEO 62

5.1 Introduction 62

5.2 Tracking Method – Annealed Particle Filtering (APF) 63

5.3 Tracking the Motion of a Person in Real Video 66

5.3.1 Tracking Process 67

5.3.2 Calibrating the Camera 68

5.3.3 Training Data 69

5.3.4 Subtracting Background 71

5.3.5 Model of Human Motion 72

5.3.6 Tracking Results 75

5.4 Summary 78

6 GENERATING BEHAVIORS FOR A VIRTUAL CHARACTER 80

6.1 Introduction 80

6.2 Model of Interactive Behaviour 82

6.2.1 Training Data 82

6.2.2 Dual-Input HMM Construction 83

6.3 Generating Interactive Behaviours 86

6.3.1 The Standard Viterbi Algorithm 86

6.3.2 The Windowed Viterbi Algorithm 88

6.3.3 The Trellis Structure 92

6.3.4 Estimating Output Behaviours 92

6.3.5 Trajectory Post-processing 95

6.3.6 Motion Resynthesis Summary 95

6.4 Placing Virtual Character Back into the Real Video 96

6.5 Experiments with 3D MoCap Data 98

6.5.1 Generating Algorithm 99

6.5.2 Animation Video Sequences 100

6.6 Assessing the Accuracy of the Generated Behaviour 100

6.7 Perceptual Evaluation 109

6.8 Summary 112

7 IMPROVING THE MODEL OF HUMAN MOTION 114

7.1 Introduction 114

7.2 Splitting Complex Motion Data Automatically 117

7.3 The Combined Hidden Markov Model 120

7.4 The Process of Combining HMMs 123

7.5 Experiments with 3D MoCap data 127

7.5.1 Assessing the Accuracy of Tracking Results of Shaking Hands Behaviours 127

7.5.2 Assessing the Accuracy of Tracking Results of Pushing Behaviours 128

7.5.3 Assessing the Accuracy of Tracking Results of Pulling Behaviours 134

7.6 Summary 136

8 CONCLUSION AND FUTURE RESEARCH 139

8.1 Conclusion 139

8.2 Future Research 141

A PHASESPACE MOTION DIGITIZER SYSTEM 144

B MOTIONBUILDER: ACTOR 147

BIBLIOGRAPHY 154

List of Figures

2.1 A stick-figure human body model [1]. 11

2.2 A 2D contour human body model [1]. 11

2.3 A volumetric (cylinder) human body model (viewed from two directions) [2]. 12

3.1 Overview of the virtual character generating system. 30

3.2 Human model with 30 markers (Two markers on the back are not visible). 34

3.3 (a) Original image of a person, (b) the person in the scene without background. 35

4.1 Example of 12 cameras positioned for full body motion capture. 41

4.2 Calibration object (a) and calibration wand (b) used during the calibration process. The red lights are LEDs. 43

4.3 Original captured 3D MoCap data. (a) Shaking Hands Behaviour, (b) Pulling Behaviour, (c) Pushing Behaviour. 44

4.4 Motion data. (a) No markers are occluded. (b) The markers with dark blue are occluded during the capturing process, so they need to found manually with interpolation. 45

4.5 A model of geometry of a human body. It consists of 16 segments (represented as truncated cones in the figure) connecting 20 vertices on the body (represented as red crosses in the figure). 46

4.6 The original training data of the motion for a person walking and shaking hand distribution visualised in 2D. Blue dots represent the original motion data. 49

4.7 Reduced dimension data with two biggest eigenvectors modelling the walking and shaking hand motion of the whole human body. Blue dots represent the motion data. 50

4.8 Data distribution visualised in 2D and fitted with 12 Gaussians. The red ellipses represent Gaussian and the blue dots denote the motion data. 52

4.9 Data distribution visualised in 2D and fitted with 4,8,16 and 30 Gaussians. The red ellipses represent Gaussian and the blue dots denote the motion data. 53

4.10 The Basic Structure of an Hidden Markov Model 55

4.11 GMM with connections showing HMM transition probabilities with values greater than 0.01. 59

5.1 Original background image without a person in the scene. 69

5.2 Original video image with a person present in the scene. 70

5.3 Selected video images of one person walking and shaking hands at time 2.2s, 3s and 4.83s respectively. 70

5.4 Binary image after subtracting background. 72

5.5 Selection frames from Original video sequence and binary images after subtracting background. 73

5.6 Percentage of eigenenergy vs. number of dimensions 74

5.7 Original images (first column) and the estimated 3D figures in the same view (second column). 76

5.8 Tracking result for the synthesised data (shaking hands behaviours). In the top of figure, blue trajectories are ground truth while red trajectories are tracking result. In the bottom of figure, the line shows the distance between ground truth and the tracking result. 78

5.9 Tracking result for the synthesised data (pushing behaviours). In the top of figure, blue trajectories are ground truth while red trajectories are tracking result. In the bottom of figure, the line shows the distance between ground truth and the tracking result. 79

5.10 Tracking result for the synthesised data (pulling behaviours). In the top of figure, blue trajectories are ground truth while red trajectories are tracking result. In the bottom of figure, the line shows the distance between ground truth and the tracking result. 79

6.1 Illustration of the Viterbi Trellis. S_i is the sequence of hidden states and $a_{max}(t)$ is the maximum probability at the time t. T is the number of frames. 89

6.2 Illustration of the windowed Viterbi algorithm. S_i is the sequence of hidden states and $\pi_{\frac{n+1}{2}}$ is the maximum probability at the time $t = 1$. The dashed blue box represents the length of the window T. The pink node represents the second state of the best path for the window T. The black node represents the output at the time t. 91

6.3 Initial Trellis data structure for generating motion data. Q is a set of states. 93

6.4 A representation of error Calculation for the generating motion data. $\mathbf{C}_t^j$ is the data vector for person $\mathbf{B}$ for cell j at time t, $\mathbf{b}_{input}$ is the new input signal, $\mathbf{b}_a$ is the data vector for person $\mathbf{A}$, E represents the error. 93

6.5 Match the generated motion which is represented by certain points (denote as blue points) on the body onto a 3D actor. 97

6.6 The animation sequence, the real video sequence and the background sequence are imported to Shake. 98

6.7 The process of producing the animation video. 99

6.8 Interactions with virtual character (Handshake behaviour). 101

6.9 Interactions with virtual character (Pushing behaviour). 102

6.10 Interactions with virtual character (Pulling behaviour). 103

6.11 Error of the generated behaviour (in mm). Shaking hands behaviour (Sequence 1). The error of generated behaviour using the standard Viterbi algorithm is shown in the solid blue line, the pink dash line is for using the windowed Viterbi algorithm. 104

6.12 Error of the generated behaviour (in mm). Shaking hands behaviour (Sequence 2). The error of generated behaviour using the standard Viterbi algorithm is shown in the solid blue line, the pink dash line is for using the windowed Viterbi algorithm. 105

6.13 Error of the generated behaviour (in mm). Shaking hands behaviour (Sequence 3). The error of generated behaviour using the standard Viterbi algorithm is shown in the solid blue line, the pink dash line is for using the windowed Viterbi algorithm. 105

6.14 Error of the generated behaviour (in mm). Pulling behaviour (Sequence 1). The error of generated behaviour using the standard Viterbi algorithm is shown solid in blue line, the pink dash line is for using the windowed Viterbi algorithm. 106

6.15 Error of the generated behaviour (in mm). Pulling behaviour (Sequence 2). The error of generated behaviour using the standard Viterbi algorithm is shown solid in blue line, the pink dash line is for using the windowed Viterbi algorithm. 106

6.16 Error of the generated behaviour (in mm). Pulling behaviour (Sequence 3). The error of generated behaviour using the standard Viterbi algorithm is shown solid in blue line, the pink dash line is for using the windowed Viterbi algorithm. 107

6.17 Error of the generated behaviour (in mm). Pushing behaviour (Sequence 1). The error of generated behaviour using the standard Viterbi algorithm is shown in solid blue line, the pink dash line is for using the windowed Viterbi algorithm. 107

6.18 Error of the generated behaviour (in mm). Pushing behaviour (Sequence 2). The error of generated behaviour using the standard Viterbi algorithm is shown in solid blue line, the pink dash line is for using the windowed Viterbi algorithm. 108

6.19 Error of the generated behaviour (in mm). Pushing behaviour (Sequence 3). The error of generated behaviour using the standard Viterbi algorithm is shown in solid blue line, the pink dash line is for using the windowed Viterbi algorithm. 108

6.20 Test video sequences showed only the motion of certain points on the body. 110

7.1 The general idea of the work contained in this chapter. 116

7.2 Split a complex data sequence to subsequences. 117

7.3 Error of the generated behaviour (in mm). Pushing behaviour. The error of generated behaviour using the standard Viterbi algorithm is shown in blue solid line, the pink dash line is for using the windowed Viterbi algorithm. The black dash lines show the place to split the motion. 118

7.4 An illustration of splitting process. 119

7.5 (a) Opening by flat structuring element; (b) Top-Hat transformation [3]. 120

7.6 The example structure diagram of two simple HMMs for the combined HMM are shown. a is the transition probability from sub-model 1 to sub-model 2. It can be obtained using the Baum-Welch algorithm. 121

7.7 Motion Data (walking and shaking hand) distribution visualised in 2D for normal HMM which trained on all motion data and fitted with 24 Gaussians. The red ellipsoids represent Gaussians associated with a HMM state and the blue dots denote the motion data. 122

7.8 Motion data (walking and shaking hand) distribution visualised in 2D for the combined HMM and fitted with 24 Gaussians. The red ellipsoids represent Gaussians associated with a HMM state and the blue dots denote the motion data. 123

7.9 Motion data (walking) distribution visualised in 2D and fitted with 10 Gaussians. The red ellipses represent Gaussians and the blue dots denote the motion data. 124

7.10 Transition matrix for the normal HMM (55 states). 126

7.11 Transition matrix for the combined HMM (55 states). 126

7.12 Error of the tracked behaviours (in mm). Shaking hands behaviours (Sequence 1). (a) - (e) show the error of the tracked behaviours when using each separate HMM respectively. 129

7.13 Error of the tracked behaviours (in mm). Shaking hands behaviours (Sequence 1). The error of the tracked behaviours using the normal HMM is shown in the black line, the pink line is for using the combined HMM. 130

7.14 Error of the tracked behaviours (in mm). Shaking hands behaviours (Sequence 2). The error of the tracked behaviours using the normal HMM is shown in the black line, the pink line is for using the combined HMM. 130

7.15 Error of the tracked behaviours (in mm). Shaking hands behaviours (Sequence 3). The error of the tracked behaviours using the normal HMM is shown in the black line, the pink line is for using the combined HMM. 131

7.16 Error of the tracked behaviours (in mm). Pushing behaviours (Sequence 1). (a) - (d) show the error when using each separate HMM respectively. 132

7.17 Error of the tracked behaviours (in mm). Pushing behaviours (Sequence 1). The error of the tracked behaviours using the normal HMM is shown in the black line, the pink line is for using the combined HMM. 133

7.18 Error of the tracked behaviours (in mm). Pushing behaviours (Sequence 2). The error of the tracked behaviours using the normal HMM is shown in the black line, the pink line is for using the combined HMM. 133

7.19 Error of the tracked behaviours (in mm). Pushing behaviours (Sequence 3). The error of the tracked behaviours using the normal HMM is shown in the black line, the pink line is for using the combined HMM. 134

7.20 Error of the tracked behaviours (in mm). Pulling behaviours (Sequence 1). (a) - (c) show the error when using each separate HMM respectively. 135

7.21 Error of the tracked behaviours (in mm). Pulling behaviours (Sequence 1). The error of the tracked behaviours using the normal HMM is shown in the black line, the pink line is for using the combined HMM. 136

7.22 Error of the tracked behaviours (in mm). Pulling behaviours (Sequence 2). The error of the tracked behaviours using the normal HMM is shown in the black line, the pink line is for using the combined HMM. 137

7.23 Error of the tracked behaviours (in mm). Pulling behaviours (Sequence 3). The error of the tracked behaviours using the normal HMM is shown in the black line, the pink line is for using the combined HMM. 137

A.1 PhaseSpace products layout [4]. 146

B.1 3D motion data 147

B.2 Asset Browser window 148

B.3 Match markers and actor 149

B.4 Navigator window for create actor 150

B.5 Exporting the marker set 151

B.6 Re-use the market set. 153

List of Tables

4.1 Probability transition matrix trained on walking and shaking hands MoCap data of one person. The sum values of each row is one. 60

6.1 Evaluation Results. The resulting motions are generated using the standard Viterbi algorithm. 111

6.2 Evaluation Results. The resulting motions are generated using the windowed Viterbi algorithm. 112

Chapter 1

INTRODUCTION

The main research area of this thesis is to generate human interactive behaviours for a virtual character responding to a real person in the video. The problem of generating human interactive behaviours that respond to real person is a real challenge, due to the complexity and uncertainty of human motion. Using a *priori* models of geometry and motion helps to deal with these problems, by imposing constraints on the interpretation of motion data.

In this research, there is a particular interest in developing models of human actions that not only assist in tracking human motion in real video, but are also suitable for generating interactive virtual characters, behaving in a fashion consistent with the actions of the real people in the video. The applications for such technology include areas of computer games, film production and virtual environments. However, current techniques for simulating virtual character behaviours generate a two dimensional (2D) silhouette of a virtual human. (For example, shaking hands [5]). To our best knowledge, nobody has attempted to produce interactive behaviours for fully articulated 3D virtual characters in real-time until now.

In this thesis, two novel approaches are presented for generating intelligent behaviours regarding fully articulated 3D virtual characters,

on the basis of visual analysis of the motion of a real person in a 2D video. To achieve this goal, a system named "Virtual character generating system" is developed. The system consists of learning the model of a 3D articulated human motion from motion capture (MoCap) data, tracking the 3D motion of a real person in real video, and generating a variety of complex behavioural motions for a 3D virtual character responding to the tracked person in the real video.

In recent years, MoCap technology has become the common place in computer vision and computer graphics area. Using MoCap equipment, real person behaviours can be recorded and exported as MoCap data. The latest developments in computer graphics area have included the use of low-dimensional statistical models trained on MoCap data to represent particular types of motion [6]. This allows an artist to generate new motions given only a few constraints, and to interpolate between different motions effortlessly in the low-dimensional space of the model [6, 7]. In this research, 3D MoCap data is captured using the PhaseSpace Motion Digitizer System [4] in conjunction with MotionBuilder, a commercial software package [8]. Statistical models are learnt on 3D MoCap data representing a number of interactions between two people, from which an appropriate behaviour for a virtual character can be derived. The models are based on Principal Component Analysis (PCA) and Hidden Markov Models (HMMs).

To generate an intelligent behaviour for a virtual character, the motion of a real person in video must first be analysed. In recent years, a variety of methods to extract the 3D articulated motion of a moving person in video were developed. The majority of these methods rely on a model of the human body and/or motion. Many approaches

have used the CONDENSATION algorithm or similar methods based on particle filters [9–11]. The problem with these approaches is the large number of particles required to track the motion, which grows exponentially with the dimensionality of the search space. Hence, they are not adequate for real-time applications. For a fully articulated motion of the human body these techniques entail unacceptably long processing times. For this reason, the particle filtering technique has been improved to address the problem of tracking in high dimensional space with the annealed particle filtering (APF) [12].

Two methods are then presented for generating interactive behaviours for a virtual character responding to the tracked person in the video, given a sequence of 3D poses of a person. They are the standard Viterbi algorithm [13] and the windowed Viterbi algorithm [14]. The standard Viterbi algorithm requires the full observation sequence before the processing starts, thus making real-time processing impossible. When the windowed Viterbi method is used instead, it does not require the full observation sequence before the processing starts, thus it can be used in a real-time system. Nevertheless, realistic generated interactions behaviours can still be obtained for both algorithms. Hence, the performances of the standard Viterbi algorithm and the windowed Viterbi algorithm within the virtual character generating system can be compared. When assessing the accuracy of the generated behaviours and inspecting the generated motion sequences visually, it was found that the windowed Viterbi algorithm can be used to detect sudden changes in the motion sequences.

To improve the tracking capability of the algorithm, an automatic splitting at the complex motion data is proposed. The idea of behind

such comes from the generated interactive behaviours using the windowed Viterbi algorithm. The windowed Viterbi algorithm can detect sudden changes in the complex motion sequence efficiently. By exploiting this advantage, the new method focuses on splitting the complex human motion automatically. Thereafter, it learns adaptively the model of human motion from the different split parts, and combines the separate models into one, to track the motion of a person in real video. The performance of the conventional model (a model trained on 3D MoCap data representing the complex motion) and the combined model are also compared. The analysis shows that the tracked motion using the combined model is better than the motion tracked using the traditional model.

1.1 Thesis Overview

The structure of this thesis is as follows:

- In Chapter 2, previous work in computer vision and computer graphic areas is reviewed. This includes visual tracking methods in video, human motion modelling, generating a computer graphics character, and generating interactive behaviours between two people.

- In Chapter 3, an overview of a "virtual character generating system" is given, with description of each of its major processes: 1) Acquisition of 3D MoCap data, 2) Training a model for tracking a 3D person in video, 3) Tracking motion of a 3D person in real video, 4) Training a model for generating interactive behaviours, 5) Generating human interactive behaviours for a virtual charac-

ter. The overview also describes how the following Chapters in the thesis relate to each of these processes.

- In Chapter 4, the process of data acquisition and learning dynamics model of human motion on 3D MoCap data for a single person is presented.

- In Chapter 5, the annealed particle filter (APF) for the tracking of a person's motions in real video is described. The results of applying a model to track motion of a person in real video concludes this chapter.

- In Chapter 6, the process of generating interactive behaviours for a virtual character responding to the tracked person (Chapter 5) in real video is introduced. Two methods are used for generating interactive behaviours, namely the standard Viterbi algorithm, and the windowed Viterbi algorithm which can be used in a real-time system. The experimental results are also presented, followed by perceptual evaluation.

- In Chapter 7, a novel approach is developed for finding where to split the complex motion data automatically in order to improve the model of human motion for obtaining better tracking results. The combined models which train on the different parts of the split data (for example, walking, shaking hands and pushing), and then fused together are introduced. The combined model is used for tracking motion of a person in real video.

- Chapter 8 concludes this thesis and provides directions for further research.

1.2 Main Contributions

In summary the main contributions of this thesis are:

- A novel approach for generating intelligent behaviours for fully articulated 3D virtual characters on the basis of visual analysis of the motion of a real person in ordinary 2D video using the dual-input HMM and the standard Viterbi algorithm.
- A new approach for generating interactive behaviours for virtual characters using the windowed Viterbi algorithm, capable of doing so in real-time.
- A novel method for improving the model of the human motion due to the ability to split the complex human motion automatically.

1.3 List of Publications

The research described in this thesis is published as follows:

1. **Yue Zheng**, Yulia Hicks, Darren Cosker, Dave Marshall, Juan C. Mostaza and Jonathon A. Chambers. "Virtual Friend: Tracking and Generating Natural Interactive Behaviours in Real Video", 8th International Conference on Signal Processing (ICSP 2006). Nov. 16-20, Guilin, CHINA, 2006.
2. **Zheng Y.**, Hicks Y. A., Cosker D. P., Marshall D., Chambers J. A., "Generating 3D Interactive Behaviours", Proc. of the 3rd European Conference on Visual Media Production (CVMP 2006), London, UK, 2006.
3. **Zheng Y.**, Hicks Y. A., Cosker D. P. and Marshall D., "Generating Human Interactive Behaviours using the Windowed Viterbi

Algorithm", 3rd International Conference on Computer Graphics Theory and Applications (GRAPP 2008), Jan. 22-25, Funchal, Madeira, Portugal, 2008.

4. **Zheng Y.**, Hicks Y. A., Cosker D. P. and Marshall D., "Real-time Generation of Interactive Virtual Human Behaviours", submitted to Lecture Notes in Computer Science, Springer, October, 2008.

5. **Zheng Y.**, Hicks Y. A. and Marshall D., "Improving the Model of Human Motion for Tracking the Motion of a Person in Real Video", In Preparation.

Chapter 2

INTERACTIVE BEHAVIOURS: A REVIEW

The problem of generating interactive behaviours for a 3D virtual character is a challenging task, considering the complexity of the geometry of a human body, a large number of degrees of freedom of the motion, and the uncertainty of the human motion. Using a *priori* models of geometry and motion helps to deal with these problems by imposing constraints on the interpretation of motion data.

In recent years, there has been a large amount of research in modelling and tracking 3D human motion [15–19]. There are several major reasons for this. First, 3D human motion includes more details about the object, such as the orientation of the object in the real world and different angle of view of the object. Those information will provided to the researchers to understand and analyse the human motion. Second, people in the industry are become interested in using the techniques to build 3D computer models, typically keeping the structure and throwing away the motion. Third, the real world is 3D.

In this research, the aim is to create a 3D virtual character capable of responding to actions obtained from observing a real person in video in a realistic and sensible manner. To achieve this purpose, first of all,

a model of 3D articulated human motion from MoCap data is learnt. The motion of a 3D real person in a 2D video is then tracked. Finally, interactive behaviours for a moving virtual character reacting to the motion of the tracked person in the video are generated.

In this chapter, the relevant research have been reviewed. The chapter begins by considering models of the human body geometry and the methods used for modelling dynamics of human motion in Section 2.1. Section 2.2 reviews dimensionality reduction methods of the state space. Section 2.3 presents the difference between 2D and 3D tracking. Section 2.4 investigates methods used for visual tracking in real video. Section 2.5 discusses interactive behaviours between two people. Finally, Section 2.6 reviews the work on generating a computer graphics character.

2.1 Modelling the Motion of the Human Body

As mentioned above, the objective is to generate interactive behaviours for a virtual character reacting to the motion of the tracked person in the original video footage using model-based approach. Thus, a geometry model of the human body and motion model of the human body is built. The model is used to represent the moving bodies in the video.

In recent years, there has been a large amount of research in modelling the geometry of the human body in computer graphics and computer vision areas. Conventionally, a human body is represented by a stick figure (Figure 2.1), 2D contours (Figure 2.2) or a volumetric model (Figure 2.3) [2].

The simplest representation of a human body is the stick-figure

[20–22] which is drawing to depict the general form of humans. It is based on the observation that human motion is essentially the movement of the human skeleton brought about by the attached muscles. The stick-figure models are often used to recover the 3D configuration of a moving subject according to their projected 2D image. In 2D contour models [23], the body segments can be approximated as 2D ribbons. A 2D ribbon model consists of two components: the basic human body model and the extended body model. The basic human body model outlines the structural and shape relationships between the body parts. The extended model is intended to resolve ambiguities in the interpretation process by identifying a certain pattern from the outline picture. Thus, a description of the body parts and the appropriate body joints is obtained. Volumetric models (such as elliptical cylinders [24,25], and spheres [26]) represent the shape of the human body, but require more parameters for computation. Each model can be scaled according to the height of the human body. The volumetric models can be used to model articulated and self-occluding objects such as fingers. While each model has its strengths, there is not a single model that is perfect for every possible use.

Elliptical cylinders are commonly used in modelling the human body. For example, Hogg's [24] WALKER model is represented by elliptical cylinders. In his model, the human body is described by 14 cylinders. Each cylinder is controlled by two parameters: the size of the major and minor axes of the cross-sectional ellipse with respect to the scale defined by the embedded coordinate system. Rehg and Kanade [27] tracked two self-occluded fingers with cylinders in 2D finger images. Goncalves *et al.* [28] addressed the problem of estimat-

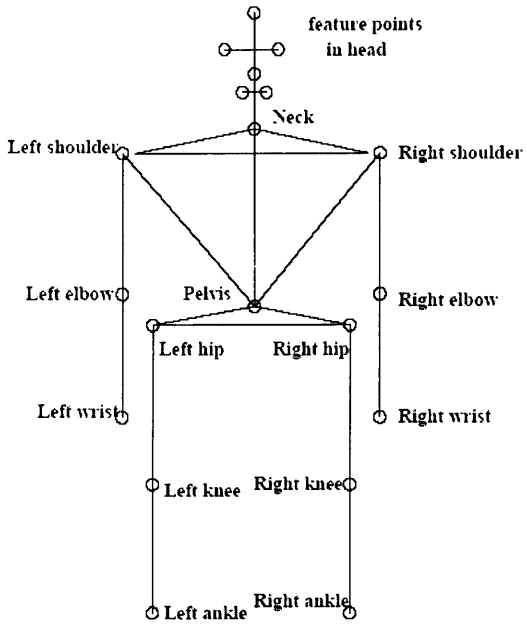


Figure 2.1. A stick-figure human body model [1].

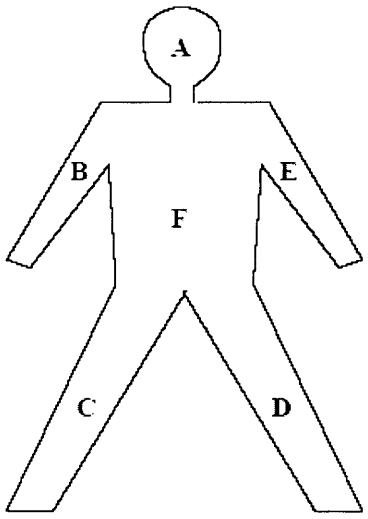


Figure 2.2. A 2D contour human body model [1].

ing the position and motion of a human arm in 3D in a monocular video sequence by modelling the upper and lower arm as truncated right-circular cones, and the shoulder and elbow joints are modelled as spherical joints. The hand tip is considered to be an extension of

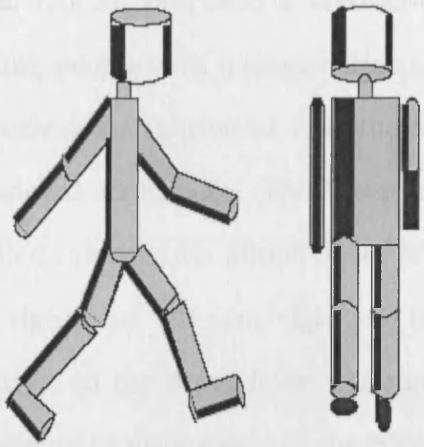


Figure 2.3. A volumetric (cylinder) human body model (viewed from two directions) [2].

the forearm joints. Recent work by Park *et al.* [29] proposed a method to model human body parts by combining an ellipse representation and a convex hull-based polygonal representation for the recognition of two-person interactions using a hierarchical Bayesian network. The model can represent the human body part accuracy. However, when the behaviours is complex, the model cannot recognise the two-people interactions accurately.

Once the geometry model of the human body is built, a motion model of the human body is also needed. The purpose is to make the geometrical model change its poses, to mimic real human poses in the video sequences. In recent years, many researchers have worked on building a statistical model for tracking human motion [20, 24, 30–32] in video sequences.

Bowden [31] used PCA to simplify the motion, K-means clustering was used to collect similar motions, and modelled the dynamics of human motion using an HMM. The HMM was used to reconstruct 3D postures from monocular image sequences.

Karaulova *et al.* [20, 25] proposed a hierarchical model of human dynamics for tracking people with a single video camera. The top level of the hierarchy models the motion of the whole body as a HMM in the reduced dimensional eigenspace. The lower levels of the hierarchy contain more detailed information about poses of some subpart of the body, for example, right arm, left arm, right leg, left leg and torso with the head. The motion on the lower lever was modelled as a GMM in the reduced dimensionality eigenspace of the corresponding body part poses. The results showed that the lower level models are more accurate in representing the motion of different body parts than a single model representing the motion of the whole body.

Lawrence [33, 34] introduced a new underlying probabilistic model for PCA, that is a Gaussian process latent variable model (GPLVM). They described probabilistic principal component analysis (PPCA) which is formulated as a latent variable model, and showed how PCA can be interpreted as a Gaussian process mapping from a latent space to a data space. The algorithm for GPLVM is a non-linear process, which has three main components, sparsification, latent variable optimisation and Kernel optimisation. The model has an advantage that due to the various spectral clustering algorithms used, it is a generative process with an underlying probabilistic interpretation.

Scaled Gaussian process latent variable models (SGPLVMs) were used for learning the model parameters from training data in low-dimensional space. The model is based on the Gaussian Process (GP) [35] model. The SGPLVM optimises the low-dimensional latent space embedding human pose space. Grochow *et al.* [6] introduced the use of the SGPLVM of human poses for interactive computer animation.

Given the training poses, a SGPLVM is used to represent the probability distribution function over poses. Urtasun *et al.* [32] proposed the use of the SGPLVM to learn prior models of 3D human pose for 3D people tracking. The SGPLVM simultaneously optimises a low-dimensional embedding of the high-dimensional pose data and a density function. The optimisation of these two features yields higher probability to points close to training data and provides a nonlinear probabilistic mapping from the low-dimensional latent space to the full-dimensional pose space.

Caillette *et al.* [30,36] learned behaviours with variable length Markov model (VLMM) for tracking 3D human body in real-time. VLMM deals with a class of random processes in which the memory length varies. The advantage of VLMM is the ability to locally optimise the length of memory required for prediction compared with a fixed memory Markov model. This results in a more flexible and efficient representation which is particularly attractive in cases where the higher-order temporal dependencies in some parts of the behaviour and lower-order dependencies elsewhere needed to capture.

2.2 Reducing the Dimensionality of the State Space

Modelling the motion of the human body is a very difficult problem in computer vision and computer graphics areas, because human motion is high-dimensional. Generally, high quality human motion needs to be represented by fifty to sixty dimensions [37]. However, the movements of the joints are highly correlated for many behaviours. For example, during the run/walk motion, the arms, legs and torso tend to move in a similar oscillatory pattern. Therefore, the dimensionality of motions

can be reduced by applying a simple dimensionality reduction technique to poses taken from human motion sequences. For example, six to eight dimensions are enough to represent a human jump that looks similar to the original high-dimensional version [7]. In recent years, a simple dimensionality reduction technique, such as PCA [38], has been used to reduce the dimensionality of the data set in learning the statistical models of human motion [20, 31, 39].

Troje [40] presented an approach to linearize human walking data. Each walking sequence (3D MoCap data) is decomposed into a PCA space. A reduced dimension space is produced by this, and the discriminant functions are determined to compute corresponding coefficients for a given parameter (male/female, happy/sad, relaxed/nervous). However, a disadvantage resides in that changing a stylistic parameter can modify locomotion speed. Moreover, as the data are computed in global 3D space, no retargeting on humans of different size is possible.

Cosker *et al.* [41] described a hierarchical image-based facial animation system capable of producing coarticulted mouth animation given audio input alone. The method is model-based, and it applies PCA on the appearance and speech training set (obtained from the captured video and audio sequences) to reduce dimensionality of the training data.

Glardon *et al.* [42] proposed an approach to generate new human walking patterns using MoCap data. The method applies PCA on motion data to yield a reduced dimension space, leading to a real time engine intended for virtual human animation. This representation allows for style-based interpolation and classification, but the motions used must be segmented first. These spaces are most useful when deal-

ing with cyclic motions such as walking.

Safonova *et al.* [7] proposed a motion synthesis framework able to synthesise human motions by optimising a set of constraints within a low-dimensional space constructed with PCA. Grochow *et al.* [6] solved the low-dimensional human motion synthesis problem by applying a non-linear PCA to the data set. Carvalho *et al.* [43] presented an approach by combining motion models and prioritized inverse kinematics for interactive low-dimensional human motion synthesis.

2.3 2D vs. 3D Tracking

Object tracking in 3D space has various applications, such as human-computer interface, behaviours analysis and so on. In recent years, many researchers have been working on 2D and 3D motion tracking, such as tracking the pose between 2D image and 2D model state (2D-2D model-based tracking approach) [44, 45], estimating the pose between 2D image and 3D model state (2D-3D model-based tracking approach) [46, 47] and tracking motion between 3D image and 3D model state (3D-3D model-based tracking approach).

2D-2D model-based tracking approach is an independent processing (2D tracking can be processed independently) [48]. This method can easily extract the information from the single image. The downside of it is the restriction to a single view suitable for tracking as the one built in the model. 3D-3D model-based tracking approach can provide more information for the object from both 3D image and 3D model. Thus, it can produce more accuracy tracking compared with 2D-2D model-based tracking approach. However, this method is more complex to process. In 2D-3D model-based tracking approach, 3D human

motion are projected onto 2D images and by evaluating the consistency between 2D estimation and image features. This method had tight interactions between 3D and 2D object positions, Thus, the tracking is more robust than the 2D tracking. The view of the tracked object also is independent.

Hicks [46] presented an approach for tracking 3D articulated motion from monocular 2D video sequences. Her method is capable of recovering 3D information from 2D video with good accuracy, it is capable of dealing with self occlusions and partial occlusions by other object. The method is not restricted to any particular view.

Marchand *et al.* [47] proposed an method for tracking complex objects in a 2D image sequence which can be approximately modelled by a polyhedral shape. The approach relies on the estimation of the 2D object image motion along with the computation of the 3D object pose. This method fulfills real-time constraints along with reliability and robustness requirements.

Howe *et al.* [49] presented a system that reconstructs the 3D motion of human subjects from single-camera video. The system tracks joints and body parts as they move in the 2D video, then combines the tracking information with the prior model of human motion to form a best estimate of the body's motion in 3D.

2.4 Visual Tracking Methods in Real Video

After learning the geometry model and motion model of a human body, tracking the motion of a 3D person in a real video became another issue in this research. Tracking human motion in video sequences constitutes the most basic block of image processing to understand its dynamic be-

havior. The main aim is to track human motion in a sequence of video frames. Then, the results of tracking are analysed mathematically to translate the motion behaviours of a human [50]. There are many areas of applications in visual tracking. An important application area in computer vision is surveillance. The most common application is to track one or more people's motion [51, 52] or vehicle's position [53]. The surveillance setting can involve access control, parking lots, supermarkets and traffic.

Another application area is virtual reality, which includes interactive virtual worlds [54], games and character animation [6, 7, 55, 56].

Another important area is user interface, which involves sign-language translation [57], gesture driven control and gait analysis [50]. These applications deal with human-computer interaction, which attempt to interact with users in a natural way [58–60].

Tracking people can be divided into several groups according to the applications. Some of them need to detect particular parts of the body, for example, hands [61, 62], leg [63, 64] or face/head [65], whilst others need to track the motion of full human body [11, 12, 59]. In this research, we work on tracking motion of full human body in real video. Conventionally, two approaches are used for tracking moving people, motion-based and model-based.

- Motion-based approaches depend on a robust method for grouping visual motions consistently over time [66]. They tend to be fast, but do not guarantee that the tracked regions have any semantic meaning [67].

- Model-based approaches can impose high-level semantic knowledge but suffer from being computationally expensive due to the

need to cope with scaling, translation, rotation and deformation.

Recent reviews of techniques for human motion tracking can be found in the survey papers by Cedras *et al.* [68], Gavrila [45], and Wang *et al.* [50]. In this research, the focus is on estimating 3D poses of a person from the 2D real video sequence using a model-based approach. The earlier work on body tracking was done by Hogg [24]. He developed a model-based walking vision system, which is illustrated the machine-generated picture over the original recorded images. Hogg's WALKER model uses a set of cylinders to represent rigid body parts, with posture represented by parameterised joint angles.

In recent years, a variety of methods for tracking a person in video have been produced. The Kalman filter (KF) [69–71] has been used successfully in the vision tracking and estimation due to its simplicity and robustness. It is simply an optimal recursive data processing algorithm, and provides a recursive solution to the linear optimal filtering problem in linear dynamical system. However, the application of the KF to non-linear systems can be difficult. The most common approach is to use the Extended Kalman Filter (EKF) for modelling complex movement of objects. The EKF approximates the models used for the dynamics and measurement process in order to approximate the probability density of a Gaussian random variable. But if the true density is non-Gaussian, then a Gaussian model is not adequate. In such cases, particle filtering may be used because it approximates the density directly using a finite number of samples.

Maskell *et al.* [10] gives a tutorial on particle filtering (PF) for nonlinear/non-Gaussian Bayesian tracking. They described the nonlinear/non-Gaussian tracking problem and its optimal Bayesian solution. Particle

filtering methods and their extensions have become popular due to their robustness to noise, clutter and occlusions in video. A number of different types of particle filters exist and some have been shown to outperform others when used for particular applications.

The Sequential Importance Sampling (SIS) particle filter [72] has a common problem that is known as the degeneracy phenomenon (the particles have negligible weight) after a few iterations. For the Resampling particle filter [10], the resampling step reduces the effects of the degeneracy problem, but it introduces other practical problems, such as, it limits the opportunity to parallelise the process since all the particles must be combined.

The Sampling Importance Resampling (SIR) particle filter [73] has the advantage that the importance weights are easily evaluated and the importance density can be easily sampled. But it can be inefficient and is sensitive to outliers.

The Regularised Particle Filter (RPF) [74] is identical to the SIR filter except for the resampling stage. It is simpler than the SIR, but has the theoretic disadvantage that the samples are no longer guaranteed to asymptotically approximate those from the posterior distribution. To design a specific type of particle filter for a particular application, it is critical to select the correct density function.

Isard and Blake [9] developed the CONDENSATION (conditional density propagation) algorithm. The CONDENSATION algorithm is a class of particle filtering algorithm [10]. It uses "factored sampling", in which the probability distribution of possible interpretations is represented by a randomly generated set. The algorithm samples stochastically from a probability density function (PDF) of a set of N possible

particles (which are feature vectors parameterising the target object), applies predictive dynamics to each particle, and evaluates each particle to create a new PDF for the next time step. This technique allows fast tracking of an object in cluttered scenes. The result of the tracking is more effective in clutter compared with Kalman filter [75]. However, a number of problems in its application exist:

1. The large number of particles required to track the motion, which grows exponentially with the dimensionality of the search space [9].

2. The CONDENSATION algorithm needs a large amount of samples to achieve better tracking result, so large amount of computation is needed.

3. The generated motion is not necessarily smooth, because all samples are generated randomly from GMM.

Deutscher *et al.* [12] described the development of a modified particle filter for general tracking without restrictive assumptions. This new algorithm is called *annealed particle filter* (APF). The standard particle filter is not suitable for full body human motion capture, because of the difficulties encountered when constructing a valid observation model as a normalised probability density distribution. For the APF algorithm, there are several important tracking parameters.

1. The weighting function. It must be general and simple, and edges and foreground silhouette, foreground-background segmentation are useful features.

2. The number of particles and the number of annealing layers. Doubling the number of annealing layers reduces the number of particles needed for successful tracking by more than half.

3. The diffusion variance vectors. Each element in the variance vector is allocated a value equal to half the maximum expected movement over one time step.

Compared with standard CONDENSATION, the APF can improve tracking performance when given equivalent computational resources [11].

Khan *et al.* [76] proposed a particle filter that effectively deals with interacting targets. For traditional particle filters such as the Bayes filter and SIR particle filter, there are problems on tracking multiple targets when the targets interact. Hence, they have introduced several filters to track multiple targets:

- The Markov random field (MRF) motion model can reduce the number of tracker failures by explicitly modelling interactions.

- The Markov chain Monte Carlo (MCMC) based particle filtering can track targets when they are not interacting, but also deals with efficiently complicated interactions when targets approach each other.

- The Reversible-jump MCMC (RJMCMC) particle filter can be extended to deal with varying number of targets, and it is prevalent in practice.

2.5 Interactive Behaviour between Two People

In 1994, Baumberg *et al.* [77] described a method for generating a similar flexible shape model automatically from real image data. The system takes live video images from a static camera, and segments moving objects from the background image via a thresholding procedure. Then, an efficient method for extracting a shape vector based on a cubic B-spline is utilised. The system can process large amounts of data in near real time to generate a compact data set. Statistical component analysis of the spline data gives a simple but effective model. The model is "data-centred" in the sense that it is constructed from real image data. An advantage of this approach is that it is easy to fit the model to new inputs, but it is still not a high level description of a human. The spline model is then be used for fast segmentation (and real time tracking) and gives a global estimate of object pose in the high-level model space.

Johnson *et al.* [5, 59] developed a system capable of producing a 2D silhouette of a virtual human interacting with a real person in video and they demonstrated it working with a handshake behaviour. They acquired training data by automatically locating and tracking individuals within a video corpus of typical interactions. Tracking is accomplished using an extension of a 2D silhouette extraction method to collect training data. Then a probabilistic model is learnt from the training data, and used as the basis for a higher level model for the behaviour model. Interaction with a virtual human is achieved using the model in parallel with a tracking algorithm.

Jebara *et al.* [78] proposed a dynamic human face, which mimics human speech in response to events. However, as Jebara states him-

self, the system exhibited only limited intelligent behaviour. Both of the above systems automatically learnt the intelligent behaviours from observed video data and represented them using HMMs [13, 79], which are commonly used for representing temporal dynamics of the data.

Hogg *et al.* [80] described the way in which interactive behaviours are learnt. This model is then used to generate an interactive virtual person. In order to obtain a behaviour model, image profiles are modelled by a B-spline contour (represented by control points). Subsequently, the mean profile is obtained by using PCA. This analysis can provide a model for the target shape with fewer parameters. Two applications of this work are:

- Interactions between a pair of individuals with application to human-computer interaction. This is addressed in [59].
- Dealing with interactions between people and motor vehicles which can find applications in surveillance.

Oliver *et al.* [81] described a real-time computer vision and machine learning system for modelling and recognising human behaviours in a visual surveillance task. The system combines top-down and bottom-up information in a feedback loop. They proposed and compared two different state-based learning architectures, namely, HMMs and Coupled Hidden Markov Models (CHMMs) for modelling behaviors and interactions. In one model, they compared CHMM and HMM architectures with data from synthetic agents; the other one analysed real pedestrian data using both synthetic and site-specific models. According to their findings on synthetic data and real pedestrian data, it was deduced that the CHMMs outperformed HMM architectures in terms

of both training efficiency and classification accuracy.

In [29, 82] Park and Aggarwal suggested a method for the recognition of two-person interactions using hierarchical Bayesian Network (BN). The recognition algorithm is preceded by a feature extraction algorithm that extracts body-pose features from the segmented and tracked (manually) body-part region in a video frame. The interacting human body parts were modelled by combining an ellipse representation and a convex hull-based polygonal. The poses of tracked body parts (respectively head pose, arm pose, leg pose) are estimated at the low level of the BN and the overall body pose is reconstructed at the high level of the BN. The evolution of the poses of the multiple body parts during the two-person interactions is achieved by incorporating the whole-body motion and spatial/temporal constraints on relative positions and causal relations between the two persons.

Later, Park and Aggarwal [83] presented a new framework for describing human actions and interactions at a semantic level with a natural language description. The representation of human interaction is based on a hierarchy; a two-person interaction is a combination of single-person actions, and the single-person action is composed of multiple body part gestures such as torso motion and arm/leg motion. The human body is represented as both the subject and object in the two-person interaction, and each person is both subject and object. Human action is represented in terms of "*subject* + *verb* + *object*", semantics and human interaction is represented via "*cause* + *effect*" semantics between human action.

2.6 Generating A Computer Graphics Character

Creating realistic motion for computer graphics characters is an important problem with applications ranging from the special effects industry to interactive games and simulations. The use of motion capture data for animating virtual characters has become a popular technique in recent years [84].

An inverse kinematics (IK) system based on a learned model of human pose is described by Grochow *et al.* [6]. Given a set of constraints, the system can produce the most likely pose satisfying those constraints in real time. The main idea of the approach is to learn a PDF over character poses from motion data, and then use this to select new pose during IK. The model is represented as an objective function over poses as a PDF, which describes the likelihood function over poses. This means that the IK system can generate any pose, but favours poses that are most similar to the space of poses in the training poses. The limitation of the style-based IK system is that if the training data does not match the desired poses well, then more constraints will be needed. However, with a generic training data set, the style-based IK produces much more natural poses than existing approaches.

Zordan *et al.* [17] presented an approach for dealing with optical motion data through the use of a dynamics model to simulate joint trajectories. The system is used to control the motion of the animated articulation, which attempts to match the positional marker data points. This approach is less sensitive to error within the resulting animated articulation, because of its dynamics based knowledge, which aids the construction process.

In [85, 86] Cosker *et al.* described a non-linear hierachical speech-

appearance model of the face capable of producing high-quality video-realistic animation given a speech input. A non-linear speech-appearance model is trained on the vectors formed by the low-dimensional appearance parameters (that include both shape and texture parameters) and low-dimensional speech parameters. They then used the non-linear speech-appearance model to calculate the associated appearance parameter of an input speech parameter for every video frame, in order to obtain the synthesized facial information.

More recently, Cosker *et al.* [41] developed a hierarchical image based facial model which is driven from speech. He also demonstrated how animation of the entire face can be created from animations of the mouth and how the colour may be incorporated and reproduced compactly without being modelled explicitly. A dual HMM is trained with two sets of states. The first HMM is built using the appearance parameter training set, and the second HMM is built on the basis of the speech training parameter set. However, the transition probability is the same as the first one. This dual HMM framework can estimate a hidden state sequence given any speech observation with the Viterbi algorithm.

Safonova *et al.* [7] proposed a motion synthesis framework able to synthesise a physically realistic motion. They utilised the technique of PCA to process the motion capture database because of a high degree of coordination between movements of human joints and constructed a low-dimensional motion space. Then they used IK on the characters limbs, as a second step to clean undesirable artifacts.

2.7 Summary

In recent years, many researchers showed interest in generating interactive behaviours for virtual characters in 2D and 3D. The applications for such technology includes the areas of computer games, film production and virtual environments.

In this thesis, the aim is to create a 3D virtual character capable of responding to actions obtained from observing a real person in video in a realistic and sensible manner. Human motion is very complex and unpredictable, so it is difficult to track human motion from real video and generate the interactive behaviours for a virtual character. The challenging problem has been dealt with, through the introduction a priori models of geometry and motion. Such models have been automatically learnt from real human motion data, obtained using commercial motion capture equipment, and contained the information in the original motion data.

Tracking people is also a challenging task because of the high dimensionality of full body kinematics, the ambiguity caused by body articulation and the fast movement. Moreover, loose clothing, mutual occlusion between body part or shadows may complicate the inference problem. In recent years, many approaches have been used for tracking the motion of a person in real video, as was discussed in this chapter.

In this chapter, the relevant researches have been reviewed. These include modelling the motion of the human body, visual tracking methods in video sequence, and methods for generating human behaviour.

Chapter 3

VIRTUAL CHARACTER GENERATING SYSTEM OVERVIEW

In recent years, many researchers have shown an interest in producing virtual worlds and populating them with virtual characters [15–19, 56, 58]. There has also been a limited amount of research into enabling virtual characters with the ability to produce intelligent behaviour on the basis of visual analysis of the scene. These were mainly conducted in the computer vision area.

A virtual character is the creation or re-creation of a human being in an image using computer-generated imagery. It must act and react in the environment, drawing on the disciplines of automated reasoning and planning [87]. The behaviours for the virtual character are learnt from real motion data and generated in response to video automatically. Virtual characters are becoming more and more popular, and used in many applications such as character animation, computer games and virtual environments (for the visual creation of 3D characters to populate virtual environments and to be used as virtual actors for film and television).

The aim of this research is to create a virtual character capable of responding to actions obtained from observing a real person in video in a realistic and sensible manner. To this end, a "virtual character generating system" (Figure 3.1) is developed. The system is used to generate interactive behaviours for a virtual character responding to a real person in video.

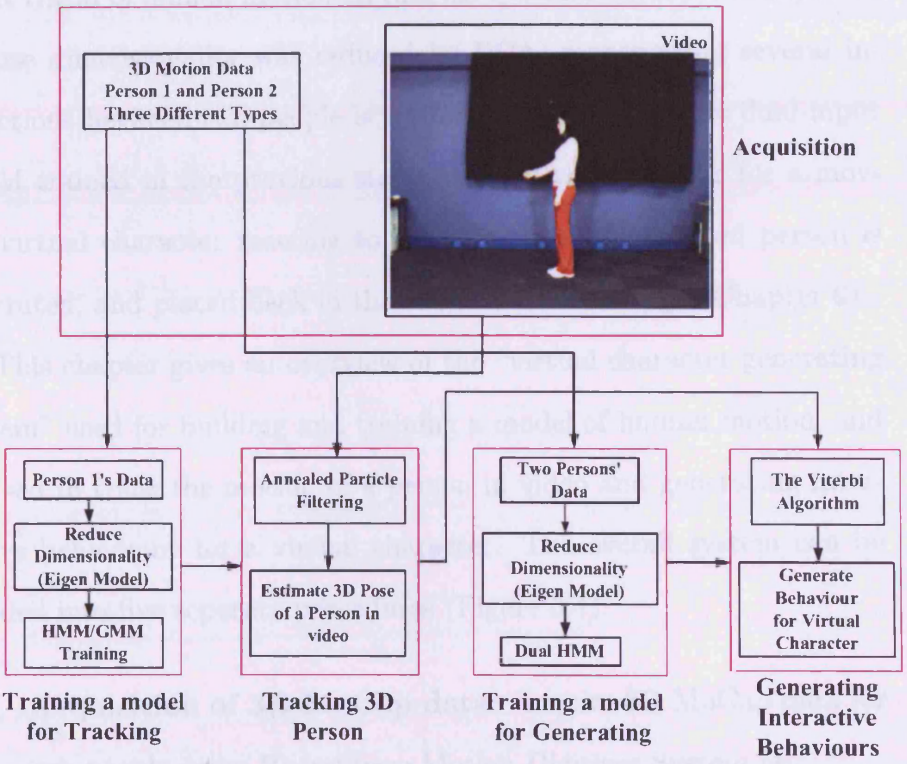


Figure 3.1. Overview of the virtual character generating system.

In the above figure, 3D MoCap data representing a number of interactions between two people is first captured using the PhaseSpace Motion Digitizer System (Chapter 4.1 and Appendix A), such as two people walking and shaking hands, one person pulling another person, and one person pushing another person. Several videos of a person's motion with a single camera (corresponding to the MoCap data) were

also recorded, as these video sequences were used to analyse human behaviour from which, 3D poses of a person were estimated. A collection of HMMs on 3D low-dimensional MoCap data (whose dimensionality was reduced by PCA) representing several motion vector for one person (Chapter 4) is then trained, after which 3D poses of a single person in 2D real video is tracked using this model (Chapter 5). Next, a dual-input HMM of human motion on that 3D low-dimensional MoCap data (whose dimensionality was reduced by PCA) representing several interactions between two people is trained. Finally, using the dual-input HMM trained in the previous stage, interactive behaviour for a moving virtual character reacting to the motion of the tracked person is generated, and placed back in the original video footage (Chapter 6).

This chapter gives an overview of the "virtual character generating system" used for building and training a model of human motion, and utilised to track the motion of a person in video and generating interactive behaviours for a virtual character. The overall system can be divided into five separate procedures (Figure 3.1):

1. **Acquisition of 3D MoCap data:** Acquire 3D MoCap data for two people using PhaseSpace Motion Digitizer System [4].

2. **Learning a model of human motion for tracking motion of a real person in video:** Train a Hidden Markov Model (HMM) of human motion (used for tracking a 3D person in a video) on 3D MoCap data of one person.

3. **Tracking the motion of a real person in video:** Track the motion of a real person in the video sequence using the HMM built in **Procedure** 2 working with the Annealed Particle Filtering

(APF).

4. **Learning a model of human motion for generating interactive behaviours for a virtual character:** Train a dual-input HMM [41] of human motion (used for generating interactive behaviours for a virtual character responding to the tracked person in the video) on 3D MoCap data set depicting interactive behaviour of two people.

5. **Generating interactive behaviours for a virtual character:** Generate interactive behaviours for a virtual character responding to the tracked person in video using the dual-input HMM built in **Procedure** 4 in conjunction with the tracked result in **Procedure** 3.

A brief description of each of these procedures in the virtual character generating system is presented in the following subsections.

3.1 Acquisition of 3D MoCap Data

The first step for training a model of human motion begins with capturing two persons' 3D MoCap data using the PhaseSpace Motion Digitizer System (Chapter 4.1). This system can capture complex motion data in real-time using advanced hardware and software technology [4].

Three different types of human motion between two people are captured. They are two individuals shaking hands, one person pulling another person, and one person pushing another person. 30 markers are placed on a person around the different joints, for example, elbow, hip and knee. Figure 3.2 shows the placement of all markers, but two

markers on the back are not visible in the figure. Each marker is represented by X, Y and Z positions, therefore a pose in each frame is represented by a 90-dimensional vector. Each MoCap data sequence is sampled at 30 frames per second (fps), and has around 150 pose vectors. In total, 14 sequences of MoCap data for shaking hands, 9 sequences of MoCap data for pushing and 7 sequences of MoCap data for pulling have been obtained. All the original MoCap data sequences can be viewed in the CD at back of the thesis in folder **Original-shaking hands**, **Original-pushing** and **Original-pulling** respectively. The acquisition of 3D MoCap Data is described in details in Chapter 4.

Several videos of a person moving with a single camera at 30 fps corresponding to the 3D MoCap data is also recorded. The recorded video data is then exported into a set of RGB images in order to analyse human behaviour and estimate 3D poses of a person in each image.

3.2 Learning a Model of Human Motion for Tracking Motion of a real person in video

The learning step involves training a model of human motion using 3D MoCap data of one person. The model is used for tracking a 3D articulated person in real video. In order to train the model of dynamics, one person's MoCap data is used. In the experiments, several sets of motion data in 3D space with 30 markers are captured, therefore a pose in each frame is represented by a 90-dimensional vector. Such data is always constrained by physical and dynamic factors, thus the dimensionality of the data set needs to be reduced using PCA, before proceeding. The seven largest eigenvectors are kept in the model, which accounts for approximately 90% of the total eigenenergy, and then train the HMM

Figure 3.2. Human model with 30 markers (Two markers on the back are not visible).

on a number of such vectors.

3.3 Tracking a 3D Person in Real Video

To track the fully articulated 3D motion of a person in video sequence, the motion of a real person in video need to be analysed. In recent years, a variety of methods to extract 3D articulated motion of a person moving in video have been developed [45]. The majority of these methods rely on some kind of a human body model and/or motion. For tracking, many approaches have used the CONDENSATION algorithm [9] or similar methods based on particle filters [10]. The problem with these approaches is the large number of particles required to track the motion, which grows exponentially with the dimensionality of the search space. For a fully articulated motion of human body this pro-

duces unacceptably long processing times. In this research, a method based on particle filtering is modified to avoid the high dimensionality problem, namely, the annealed particle filtering (APF) [12].

When tracking 3D pose of a person in 2D video, the video sequence needs to be preprocessed by cancelling the background [88], and thus a sequence of binary images are obtained. Figure 3.3 (a) shows the original image of a person from a video sequence, and the person in the scene without the background in Figure 3.3 (b). An APF is used together with an HMM trained in the previous stage (Section 3.2) to estimate 3D poses of the tracked person in the video. The result of the tracking process is a sequence of 90 dimensional vectors, each estimating a 3D pose of the tracked person in the video. Chapter 5 gives a more detailed account of tracking a person in real video.

(a) (b)

Figure 3.3. (a) Original image of a person, (b) the person in the scene without background.

3.4 Learning a Model of Human Motion for Generating Interactive Behaviours

This step involves training a dual-input HMM [41] (using 3D MoCap data of two individuals) for generating interactive behaviours responding to the tracked person in the video. Similar to the way for representing the interactive motion of one person, in the dual-input HMM, there are two sets of states, but this time only one transition matrix. The first set of states models the poses for the first person **A**, the second set of states models the poses for the second person **B**. Each state in the model is modelled with a Gaussian distribution. The details of training a model will be explained in Chapter 6.

3.5 Generating Interactive Behaviours for a Virtual Character Responding to the Tracked Person

The final process in the system generates interactive behaviours for a virtual character responding to the tracked person in the video. The model is a dual-input HMM [41, 89] which is trained in the previous stage (Section 3.4). It is capable of representing a variety of interactive behaviours, for instance, shaking hands.

Given a sequence of 3D poses of the first person **A** as input (the tracking result in Section 3.3), it is possible to project it into the dual-input HMM, and generate a corresponding sequence of poses for a virtual character using the standard Viterbi algorithm [13] and the windowed Viterbi algorithm [14]. Consequently, the performances of the standard Viterbi algorithm and the windowed Viterbi algorithm within the virtual character generating system are compared. The standard

Viterbi algorithm requires the full observation sequence before the processing starts, thus making real-time processing impossible. When the windowed Viterbi method is used instead, it does not require the full observation sequence before the processing starts, thus it can be used in a real-time system. Moreover, realistic generated interactions behaviours can still be obtained. These details will be described in Chapter 6.

3.6 Summary

This chapter has given a brief description of each step procedure pertaining to the virtual character generating system. The system is used for generating interactive behaviours for a virtual character responding to the tracked person in the video. The structure of the next four chapters, in relation to these processes, is as follows. Acquisition of 3D MoCap data and training a model of human motion for a single person will be explained in Chapter 4. The process of tracking a 3D person in real video sequence is described in Chapter 5. The process of learning a model of human motion for two people and generate interactive behaviours responding to the tracked person in the video is detailed in Chapter 6. Additionally, Chapter 7 proposes a novel technique which efficiently splits the complex motion data for better tracking ability, in order to improve the model of human motion.

Chapter 4

A MODEL OF HUMAN MOTION FOR A SINGLE PERSON

In this chapter, a model of human motion based on HMMs is presented. Recent developments in computer graphics have led to the incorporation of motion capture technology into everyday usage by artists. Motion capture is a technique of digitally recording the actions of human actors or an object, this information can be used to animate digital character models. The latest developments have included the use of statistical models trained on MoCap data to represent particular types of motion [6]. This allows an artist to generate new motions given only a few constraints, and to interpolate between different motions in the low-dimensional space of the model.

The approach adopted in this thesis is to model statistically the motion of a person. The data set for training the model are obtained using the PhaseSpace Motion Digitizer System. The dimensionality of the training data is reduced using PCA, since such data is always constrained by physical and dynamical factors. Finally, a model of human motion is built using a HMM.

In this chapter, a human motion capture system is first described, which is set up to obtain the MoCap data for building models of human motion. The details of the 3D MoCap data is then given. Finally, a model of human motion learnt from the collected MoCap data is demonstrated. The MoCap system is described in Section 4.1, and the obtained data is described in Section 4.2. The statistical modelling techniques used in this thesis are described in Section 4.3, and the chapter is summarised in Section 4.4.

4.1 Motion Capture Systems

Motion capture is a technique for recording movements of subjects in 3D space. It is also a simple way to track a subject's movement as the subject changes position relative to a fixed point in space. Motion capture data can be used for the mapping of motion onto a computer model (such as actor, computer character) with extra software, for example, MotionBuilder [90] and Maya [91]. In recent years, motion capture has been used for a wide variety of applications, including virtual reality, entertainment (games, movies, and television), medicine and robotics.

As motion capture technology developed, several uniquely different types of motion capture systems evolved. The types of motion capture input systems are: magnetic, mechanical, and optical [92–95]. Magnetic motion capture systems [96] use sensors placed on the body to measure the low-frequency magnetic field generated by a transmitter source. It is ideally suited for situations in which the motion range is limited. Another advantage is that all the data is in relation to a single object without occlusion. One of the biggest disadvantages is their sensitivity to metal, and it is also not suitable for fast movements, since the data

sampling rate is too slow.

Mechanical motion capture systems [92] directly track body joint angles and are often referred to as skeleton motion capture systems, due to the way the sensors are attached to the body. It allows real-time processing, and is not range limited. However, while the device can provide continuous data of the position of the object, it cannot capture continuous motion.

The basic idea of optical motion capture [93, 95] is the tracking of markers on a subject in real-time. In a typical optical motion capture system, cameras are placed on the circle of a capture area to track passive or active markers [97]. In passive optical system, the markers used for reflecting light back. So only the reflective markers can be sampled by the cameras. A large number of markers at high frame rate can be captured using this system, and the frame rate for this system is traded off between resolution and speed. Active marker systems can either strobing one marker on at a time or tracking multiple markers over time. This system can capture more subtle movements by having both higher spatial and temporal resolution. More processing is required in active marker systems compared with passive optical system.

Optical motion capture system offers several advantages: high resolution cameras, real-time data process, capture of high speed motion and tracking multiple subjects. However, optical motion capture systems have the following disadvantages: occlusion (markers cannot be seen by enough cameras, or markers may be blocked by limbs, bodies or other markers), challenging to calibrate and operate, and sensitivity to light and reflections. While each system has its strengths, there is not a single motion capture system that is perfect for every possible

use.

In the experiments an optical motion capture system named PhaseSpace Motion Digitizer System [4] has been used to collect MoCap data. A detailed description of PhaseSpace Motion Digitizer System used in the Communication Research Centre (CRC) lab at Cardiff University can be found in Appendix A.

The PhaseSpace Motion Digitizer System consists of twelve specialised CCD (Charge Couple Device) cameras and several infra-red Light Emitting Diode (LED) markers. The system is based on tracking LED markers attached to the areas of interest on a person. All "cameras" have a resolution of $3,600 \times 3,600$ (12 Megapixel) and streaming frequency of 120fps. In the experiments, the full body motion needs to be captured, thus the best place to locate cameras is in a circular configuration with the field of view being in the centre of the circle. The greater the field of view desired, the larger the circle should be. Figure 4.1 shows the position of the cameras for full body motion capture.

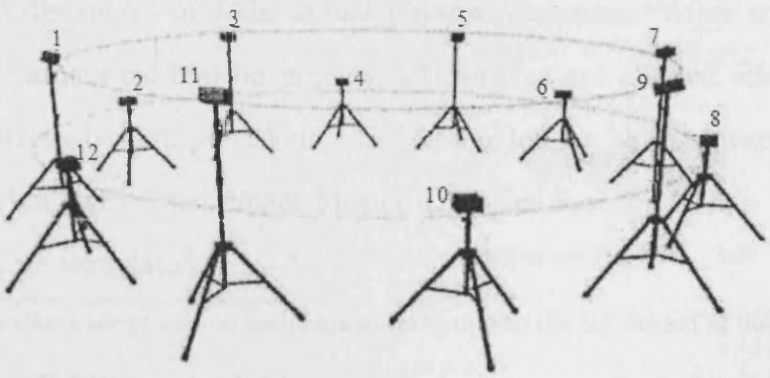


Figure 4.1. Example of 12 cameras positioned for full body motion capture.

The LEDs are placed on the body of two subjects on the places of different joints. For each person, the total number of markers is 30[1]. Figure 3.2 on Page 34 shows the placement of all markers, except two markers on the back are not visible in the figure.

Before capturing motion data, camera calibration is an important step. The PhaseSpace camera calibration is performed using the proprietary calibration software which can be run remotely on a client machine, which is very accurate and requires only several seconds to perform. This program uses a calibration object and calibration wand as shown in Figure 4.2. The current version of calibration software involves two steps. The first step is a coarse calibration that uses the calibration object. During this step the system uses an inverse transform to determine the relative position of the cameras. As for the second step defined as incremental calibration, a calibration wand with LEDs at a known distance apart is moved throughout the desired capture area. Since the system knows the physical distance between these LEDs, minor adjustments are made to the camera positions as well as to the optical maps of the cameras so as to get a best fit between the "virtual distances" and the actual physical distances. After completing the camera calibration process, all cameras are aligned effectively with extremely high precision. The MotionBuilder [90] software package working with PhaseSpace Motion Digitizer System is then used to capture motion data[2].

[1] 30 markers are placed on each human body due to the limitation of our MoCap system.

[2] MotionBuilder is a real-time animation system specifically designed to create realistic character animation. It can work with a variety of motion capture devices, such as PhaseSpace and CyberGlove [98].

a b

Figure 4.2. Calibration object (a) and calibration wand (b) used during the calibration process. The red lights are LEDs.

4.2 3D MoCap Data

The data used in this research is 3D MoCap data captured by the motion capture system described in the previous section. Three different types of motion (walking and shaking hands, one person pulling another person, and one person pushing another person) are filmed at the frame rate of 30 fps. There are five reasons for choosing these interactive behaviours:

1. These three interactive behaviours are simple, and easy to model and analyse. Yet they cover a wide variation of motion.

2. In Gavrila's [45] survey paper, he only adopted four generic interactions with people are shaking hands, embracing, pushing and hitting. For our purposes, hitting is hard to capture, and so we evaluate in a similar range of interaction to Gavrila.

3. In CMU's Graphics Lab Motion Capture Database [99], they provided some human interaction with two subjects, such as shaking hands, pulling, quarrelling and dancing. For our purposes, quar-

relling is hard to capture, and dancing is not so interactive and occlusion between the human bodies.

4. In Hogg's [59] work, he developed a system capable of producing a 2D silhouette of a virtual human interacting with a real person in video and they demonstrated it working with a handshake behaviour.
5. For our purpose, we argue that three interactive behaviours are sufficient.

30 markers are placed on the front of the person's body at the different joints, as in Figure 3.2 on Page 34. The position of each marker is represented using 3D Cartesian Coordinates. Figure 4.3 shows the original captured MoCap data of three different types.

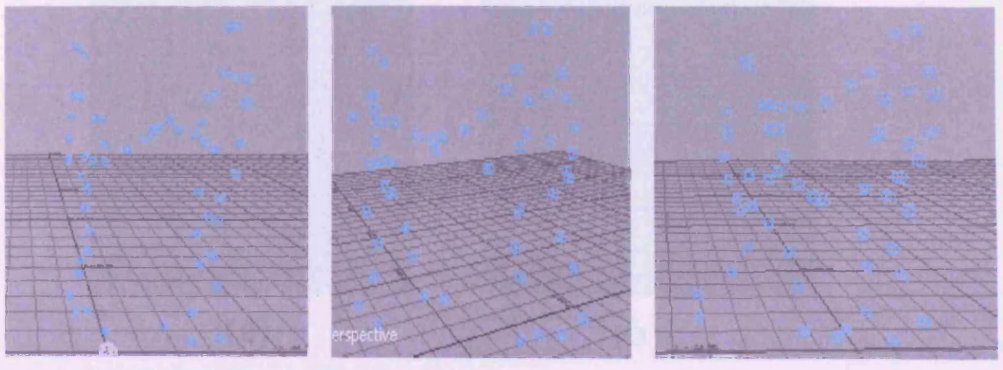


Figure 4.3. Original captured 3D MoCap data. (a) Shaking Hands Behaviour, (b) Pulling Behaviour, (c) Pushing Behaviour.

However, sometimes markers cannot be seen by enough cameras, or markers may be concealed by limbs, bodies or other markers (as shown in Figure 4.4) during the capturing process. In order to obtain high quality MoCap data, this missing data need to be interpolated manually

(the reinterpolate filter in MotionBuilder is employed to estimate the position of missing markers) after the data were captured. The final output constitutes a file containing a list of 3D Cartesian coordinates of the markers in each frame, and therefore a pose in each frame is represented by a 90-dimensional vector. Each MoCap data sequence has around 150 pose vectors. In total, 14 sequences of MoCap data for shaking hands, 9 sequences of MoCap data for pushing and 7 sequences of MoCap data for pulling have been obtained. All the original MoCap data sequences can be viewed in the CD at back of the thesis in folder **Original-shaking hands**, **Original-pushing** and **Original-pulling** respectively.

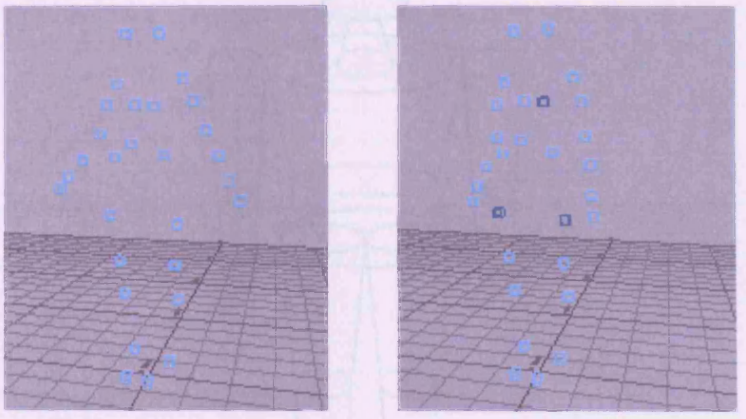


Figure 4.4. Motion data. (a) No markers are occluded. (b) The markers with dark blue are occluded during the capturing process, so they need to found manually with interpolation.

4.3 A Model of Human Body Motion

The model of geometry of a human body is deliberately kept simple, but our approach adequately models the full body as required. It consists

of a number of vertices representing significant points on the body of a person, for example, elbows, knees and hips. Some of the vertices are connected with segments representing the corresponding body parts, such as lower arms, upper arms, lower legs and upper legs. 20 vertices are used on the human body and 16 segments connect the points. These segments are dressed in truncated cones. The silhouette of the produced model of geometry roughly resembles a human figure. Figure 4.5 shows the geometry model of the human body. The vertices are represented by red crosses and the segments of human body are represented by truncated cones.

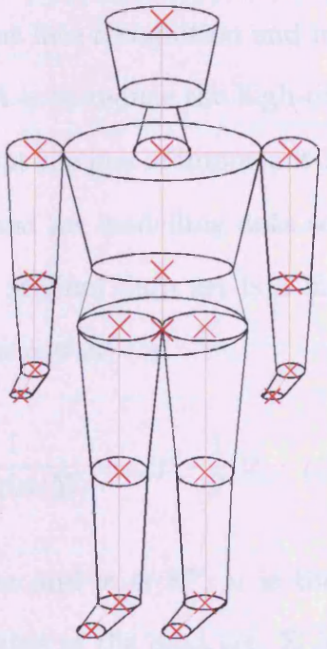


Figure 4.5. A model of geometry of a human body. It consists of 16 segments (represented as truncated cones in the figure) connecting 20 vertices on the body (represented as red crosses in the figure).

The goal is to obtain a model of human motion which is used for tracking the motion of a real person in video. As mentioned in the previous section, several sets of MoCap data in 3D space with 30 markers

are captured, therefore a pose in each frame is represented by a 90-dimensional vector. Such data is always constrained by physical and dynamical factors, thus we would like to reduce the dimensionality of this data distribution using Principal Component Analysis (PCA).

4.3.1 Principal Component Analysis: MoCap Data Dimension Reduction

PCA [38, 100, 101] is a useful statistical technique that can be used to simplify a data set of high dimension. More formally, it is a transform that chooses a new coordinate system for the data set. It has found application in fields such as face recognition and image compression. The main advantage of PCA is to reduce the high-dimensional data to low-dimensional data without the loss of important information. Therefore, PCA is a popular method for modelling data sets.

Assuming that the original data set is a multivariate Gaussian, a single multivariate Gaussian is:

$$g(\boldsymbol{x}, \mu, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2} det(\boldsymbol{\Sigma})^{1/2}} exp[-\frac{1}{2}(\boldsymbol{x} - \mu)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \mu)] \quad (4.3.1)$$

where $\boldsymbol{x}$ is the data set and $\boldsymbol{x} \in \Re^n$, n is the dimensionality of the space, μ is the mean value of the data set, $\boldsymbol{\Sigma}$ is the covariance matrix of size $n \times n$, $(\cdot)^T$ is matrix transpose, $(\cdot)^{-1}$ is the matrix inverse and $det(\cdot)$ is the matrix determinant.

PCA can be used for reducing dimensionality in a data set which retains the main characteristics, but eliminates the less significant principal components. A multidimensional Gaussian distribution is also

referred to as an *eigenmodel*. An eigenmodel Θ can be denoted as

$$\Theta = (\mu_e, \mathbf{U}_e, \mathbf{\Lambda}_e) \tag{4.3.2}$$

where μ_e is an origin in the original n-dimensional space, $\mathbf{U}_e$ is an $n \times n$ orthogonal matrix, with columns called normalized *eigenvectors*, and $\mathbf{\Lambda}_e$ is an diagonal matrix with diagonal values called *eigenvalues*.

For PCA, it is necessary to find eigenvectors and corresponding eigenvalues through eigenvalue decomposition (EVD) of the covariance matrix of the original data set. Singular value decomposition (SVD) also can be used for an arbitrary shaped matrix. EVD represents the covariance matrix as below:

$$\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \tag{4.3.3}$$

A standard way to calculate PCA is to use a covariance matrix formed from the original data [100]. Suppose we have an $m \times n$ data matrix $\mathbf{A}$ (m data vectors of n dimensions), and we want to rearrange our data into a k dimensional representation ($k < n$):

1. Form some $m \times n$ data matrix $\mathbf{A}$. For example, two types of MoCap data are chosen, walking and shaking hands, as the training data matrix $\mathbf{A}$. A pose in each frame is represented by 90-dimensional vector and each MoCap data sequence has 194 pose vectors. Therefore, the training data matrix $\mathbf{A}$ is 194×90. The original training data are shown in Figure 4.6.

2. Find the mean value in each column dimension, such that the mean matrix μ is 90×1.

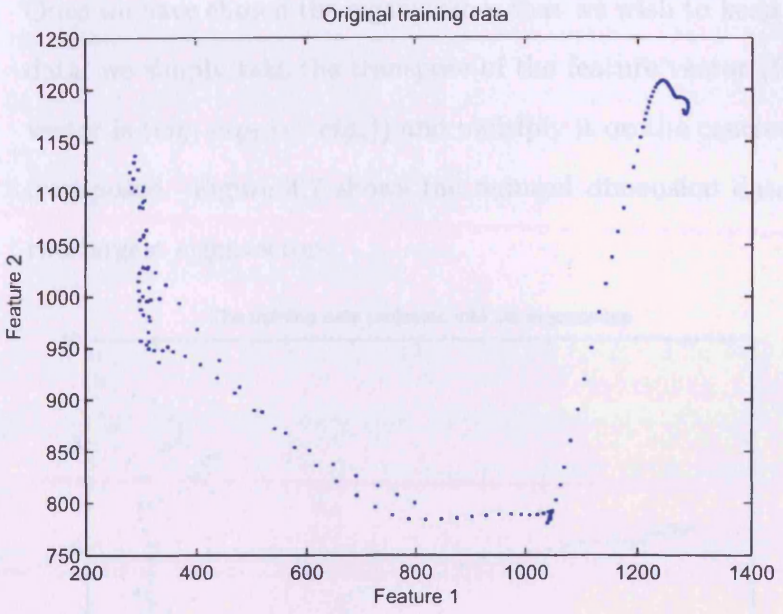


Figure 4.6. The original training data of the motion for a person walking and shaking hand distribution visualised in 2D. Blue dots represent the original motion data.

3. Subtract the mean value from each data dimension. This produces a data set whose mean is zero. Then store the centred data matrix in $\mathbf{S}$.

4. Calculate the covariance matrix $\mathbf{\Sigma}$ of $\mathbf{S}$: $\mathbf{\Sigma} = \mathbf{S}^T \cdot \mathbf{S}$. Since the original data has 90 dimensions, the covariance matrix will be 90×90.

5. Calculate the eigenvectors $\mathbf{U}$ and eigenvalues $\mathbf{\Lambda}$ of the covariance matrix. Since the covariance matrix is square, the eigenvectors and eigenvalues can be calculated for this matrix. These are important, as they provide useful information about our data.

6. Choose the first k components ($1 \leq k \leq n$) of the eigenvectors and derive the new data set (New data has only k dimensions).

Once we have chosen the eigenvectors that we wish to keep in the data, we simply take the transpose of the feature vector (feature vector is $(eig_1\ eig_2 \cdots eig_n)$) and multiply it on the centred data transposed. Figure 4.7 shows the reduced dimension data with two largest eigenvectors.

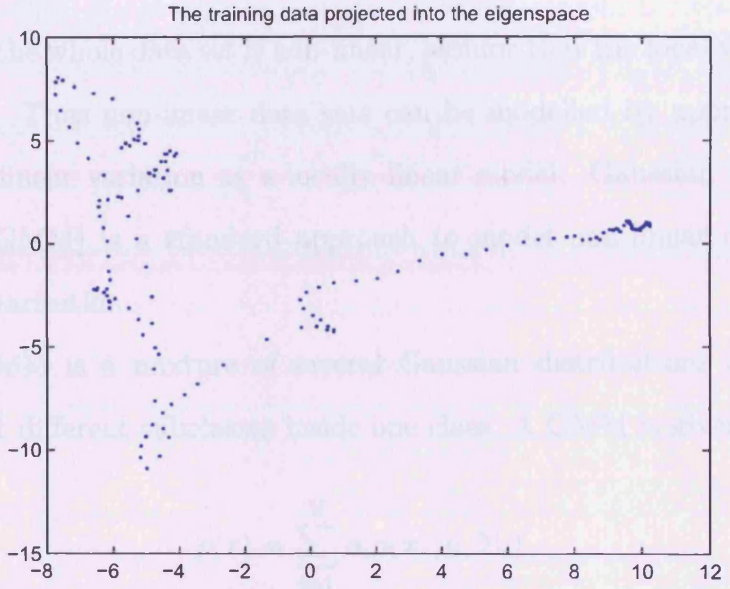


Figure 4.7. Reduced dimension data with two biggest eigenvectors modelling the walking and shaking hand motion of the whole human body. Blue dots represent the motion data.

PCA also provides a transformation of the vectors from the original $\Re^n$ space to the eigenspace Θ. Any vector $\mathbf{x}$ from the original space $\Re^n$ can be transformed to a corresponding vector $\mathbf{y}$ in eigenspace Θ using the following:

$$\mathbf{y} = \mathbf{U}^T\mathbf{x} \tag{4.3.4}$$

The vector $\mathbf{y}$ can be transformed back to the original space $\Re^n$ by multiplying it by U.

$$\mathbf{x} = \mathbf{U}\mathbf{y} \tag{4.3.5}$$

PCA can model efficiently a single Gaussian distribution, since it is a linear basis transformation. However, if the data set is non-linear (such as human motion data), then it is better to model non-linear data as a GMM.

4.3.2 Gaussian Mixture Models

Though the whole data set is non-linear, assume that the local variation is linear. Thus non-linear data sets can be modelled by approximating non-linear variation as a locally linear model. Gaussian Mixture Model (GMM) is a standard approach to model non-linear data via local linearisation.

A GMM is a mixture of several Gaussian distributions and can represent different subclasses inside one class. A GMM is given by:

$$p(x) = \sum_{i=1}^{M} \alpha_i g(\boldsymbol{x}, \mu_i, \boldsymbol{\Sigma}_i) \tag{4.3.6}$$

where α_i denotes the prior probability of each Gaussian, μ_i are the centres of the Gaussians, $\boldsymbol{\Sigma}_i$ are the covariance matrices and M denotes the number of Gaussians [102].

Expectation-Maximization (EM) [102] is a widely used method for estimating the parameter set of the GMM. It is an iterative procedure. EM proceeds iteratively in two steps, the Expectation Step and the Maximization Step.

- The expectation step: Calculate the expectation of the log-likelihood over all possible assignments of data points to sources.
- The maximization step: Maximize the expectation by differentiating written current parameters.

In the previous section, the dimensionality of the data set needs to be reduced using PCA. Here we propose to model the reduced dimensionality data set as a GMM in order to reduce the complexity of the programme. Figure 4.8 shows a GMM fitted to the low dimensional data, the red ellipses represent Gaussians and the blue dots denote the motion data. We choose the number of Gaussians to be 12 experimentally, to adequately represent the data distribution. Figure 4.9 shows the same data distribution visualised in 2D and fitted with 4, 8, 16 and 30 Gaussians respectively. It is clear that when the number of Gaussians are 4 and 8, the data distribution from the model does not match well. When the number of Gaussians are 16 and 30, some clusters are overlapped. The model cannot represent the data very well.

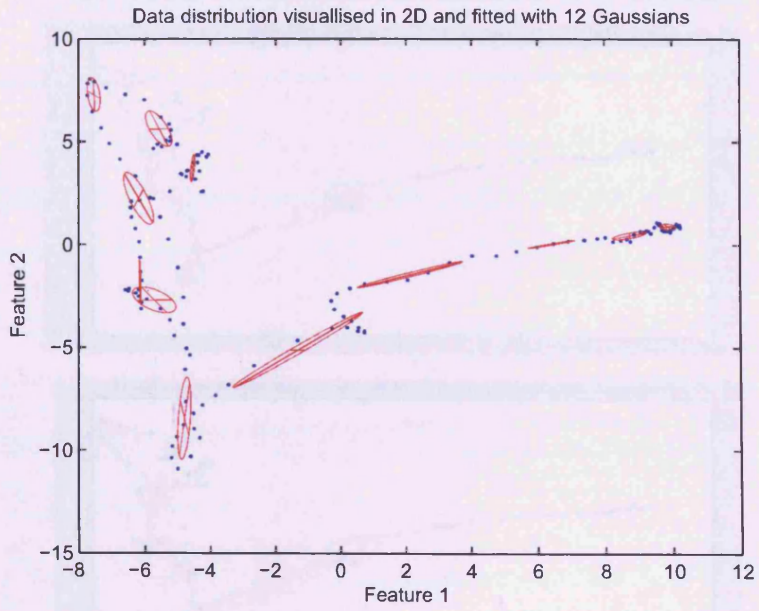


Figure 4.8. Data distribution visualised in 2D and fitted with 12 Gaussians. The red ellipses represent Gaussian and the blue dots denote the motion data.

When the reduced dimensionality data set were modelled as a GMM,

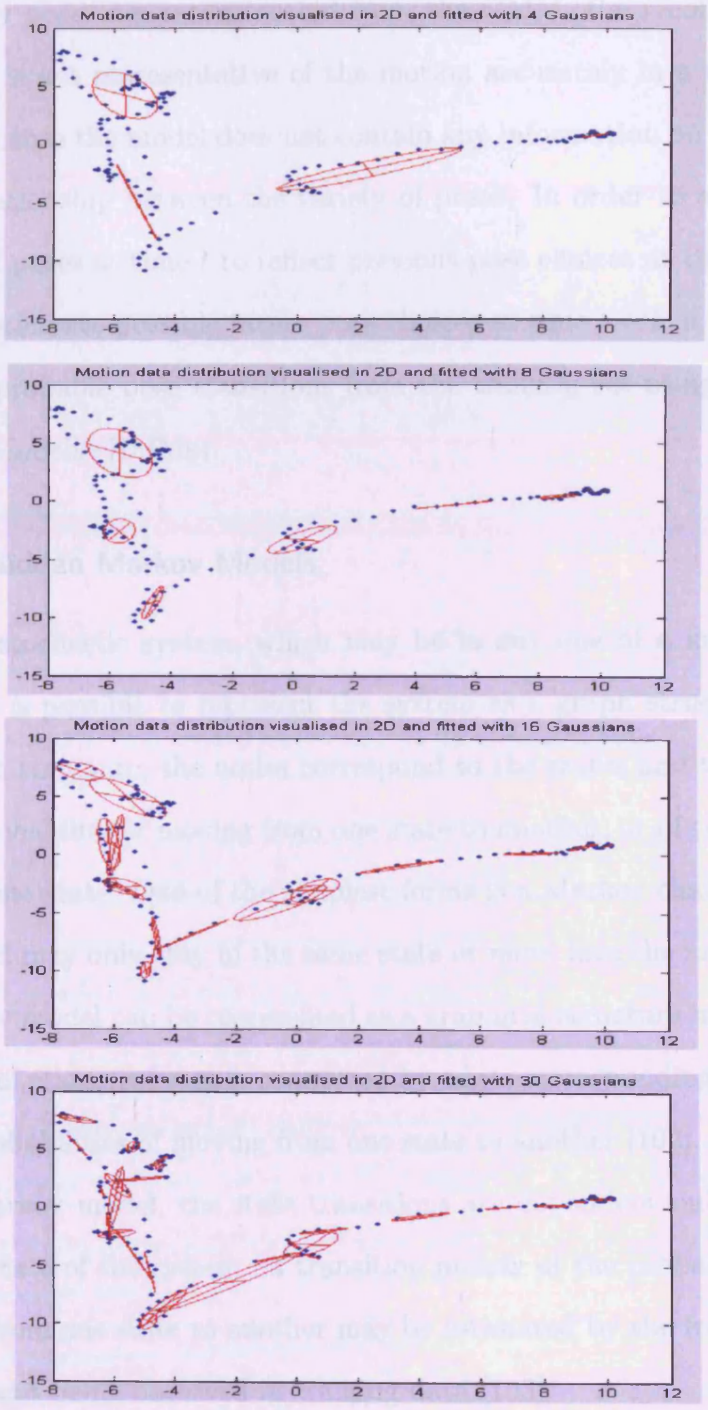


Figure 4.9. Data distribution visualised in 2D and fitted with 4,8,16 and 30 Gaussians. The red ellipses represent Gaussian and the blue dots denote the motion data.

a number of Gaussian centres are chosen in the GMM. When the human body poses are reconstructed from the GMM, the reconstructed sequence is not representative of the motion accurately in a temporal sense, because the model does not contain any information on the temporal relationship between the variety of poses. In order to select the variety of poses at time t to reflect previous pose choices at time $t-1$, and to anticipate possible future pose choices at time $t+1$, it is better to learn probable pose transitions from the training set using Hidden Markov Models (HMMs).

4.3.3 Hidden Markov Models

Given a stochastic system, which may be in any one of a number of states, it is possible to represent the system as a graph structure. In the graph structure, the nodes correspond to the states and the edges to the probability of moving from one state to another, or of remaining in the same state. One of the simplest forms is a Markov chain, where the model may only stay in the same state or move into the next state. A Markov model can be represented as a graphical structure in which a network of states is formed, connected by edges corresponding to transition probabilities of moving from one state to another [102]. In a first order Markov model, the state transitions are dependent only on the current state of the system. A transition matrix of the probabilities of moving from one state to another may be estimated by the frequencies of the event being observed in training data [103].

Hidden Markov models (HMMs) are first order Markov models which have been extended by introducing a set of hidden states (which are probabilistically linked to the observable states). The basic struc-

ture of an HMM is shown in Figure 4.10. A continuous HMM contains two sets of states, two sets of probabilities and one initial state [13, 104, 105].

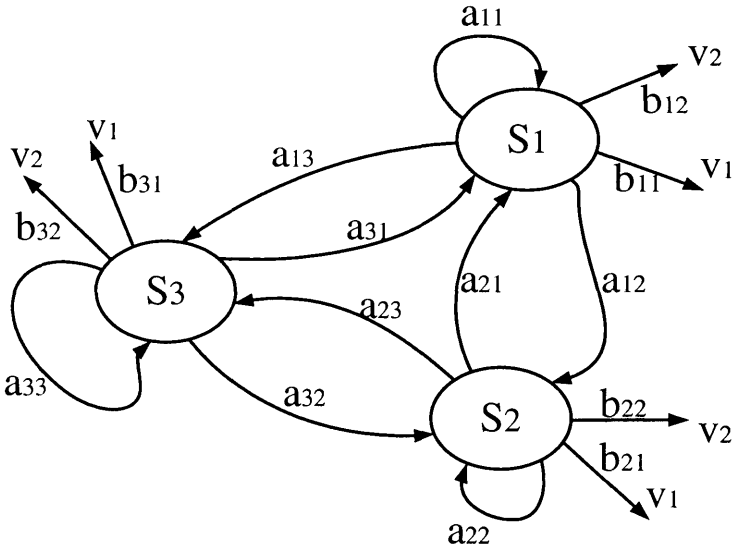


Figure 4.10. The Basic Structure of an Hidden Markov Model

In the HMM description that follows it is assumed that the person **A**'s MoCap data is used for construction.

- The number of states n in the model. The state are hidden, and may be represented by Gaussian mixtures for the purposes of a continuous HMM. The states are denoted $\mathbf{S} = \{S_1, S_2, ..., S_n\}$ and the state at time t as q_t. In this research, each hidden state is represented one of Gaussians with a GMM modelling the person **A**'s MoCap data.

- Visible states $\mathbf{v}_t = \{v(t_1), v(t_2), ..., v(t_m)\}$: the states of the process that are 'visible'. In our case, the visible state is a sequence of the person **A**'s MoCap data.

- The state transition probabilities between hidden states $\{a_{ij}\}$:

hold the probability of being in a hidden state S_j at time $t+1$ given the hidden state S_i at time t.

$$a_{ij} = P(S_j(t+1)|S_i(t)), \quad 1 \leq i, j \leq n \tag{4.3.7}$$

The state transition coefficients have the properties

$$a_{ij} \geq 0 \tag{4.3.8}$$

$$\sum_{j=1}^{n} a_{ij} = 1 \tag{4.3.9}$$

- The observation probabilities of a visible state $\{b_j(v_t)\}$: contain the probability of observing a particular visible state given that the hidden model is in a particular hidden state.

$$b_j(v_t) = P(v_t|S_j(t)) \quad 1 \leq j \leq n \tag{4.3.10}$$

- The initial state probability distribution $\pi = \pi_i$ where

$$\pi = P(q_1 = S_i) \quad 1 \leq i \leq n. \tag{4.3.11}$$

Thus an HMM is a standard Markov process augmented by a set of visible states, and some probabilistic relations between them and the hidden states.

For convenience, we use the compact notation:

$$\lambda = (\mathbf{A}, \mathbf{B}, \pi) \tag{4.3.12}$$

where $\mathbf{A} = \{a_{ij}\}$, $\mathbf{B} = \{b_j(v_t)\}$ and $\pi = \pi_i$, to indicate the complete parameter set of the model.

Defining an observation sequence $\mathbf{O} = \{o_1, o_2, ..., o_T\}$, where o_t is an observation at time t, there are three basic problems associated with a HMM. The first two are pattern recognition problems: finding the probability of a sequence of visible states given an HMM (evaluation); and finding the sequence of hidden states that most probably generated an observed sequence (decoding). The third problem is generating an HMM given a sequence of observations (learning) [13, 79, 102].

- **Evaluation** problem: Given the HMM $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ and the observation sequence $\mathbf{O} = \{o_1, o_2, ..., o_T\}$, calculate the probability that model λ has generated sequence $\mathbf{O}$. The **Forward-Backward algorithm** can be used to estimate these probabilities.

- **Decoding** problem: Given the HMM $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ and the observation sequence $\mathbf{O} = \{o_1, o_2, ..., o_T\}$, calculate the most likely sequence of hidden states S_i that produced this observation sequence. The **Viterbi algorithm** is used to find the most likely path of hidden states.

- **Learning** problem: Given some training observation sequences $\mathbf{O} = \{o_1, o_2, ..., o_T\}$ and general structure of an HMM (numbers of hidden and visible states), determine the HMM parameters $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ that best fit the training data, that is, maximizes $P(\mathbf{O}|\lambda)$. The **Baum-Welch algorithm** is used to find the unknown parameters of the HMM.

In this research, the two problems of concern are:

1. Given a number of training data, to train an HMM which has high likelihood of producing the training data. To do this, the **Baum-Welch algorithm** is used to determine the best HMM parameters. The transition probabilities and the observation probabilities are initialised using random numbers at the beginning, and then iteratively improve the estimates. This HMM is used to track a 3D person in real video.

2. Given an HMM λ and the observation sequence $\mathbf{O}$, the **Viterbi algorithm** needs to be used to find the best state sequence that produced the input observation sequence during the generating process. The result of the generating process is a sequence of 90-dimensional vectors, each estimating a 3D pose of the virtual character responding to the tracked person in the video.

4.3.4 Training a Model of Dynamics

In the experiments, the HMM is trained on one person's MoCap data. We obtained the transition probabilities in the HMM after 3 iterations. Figure 4.11 shows a GMM with 12 Gaussians, fitted to a person's MoCap data. 12 Gaussians have been used to cover the data distribution well. In the figure, the red ellipses represent Gaussian and the blue dots denote the motion data.

Table 4.1 shows the transition probability matrix of the HMM used here. It is clear that the highest probability values are on the diagonal, which corresponds to no transition. The off diagonal, there is only one more non-zero value, which means there is one transition of the cluster from the current state to another state. In the transition probability matrix, a probability equal to unity, means that the transition ends up

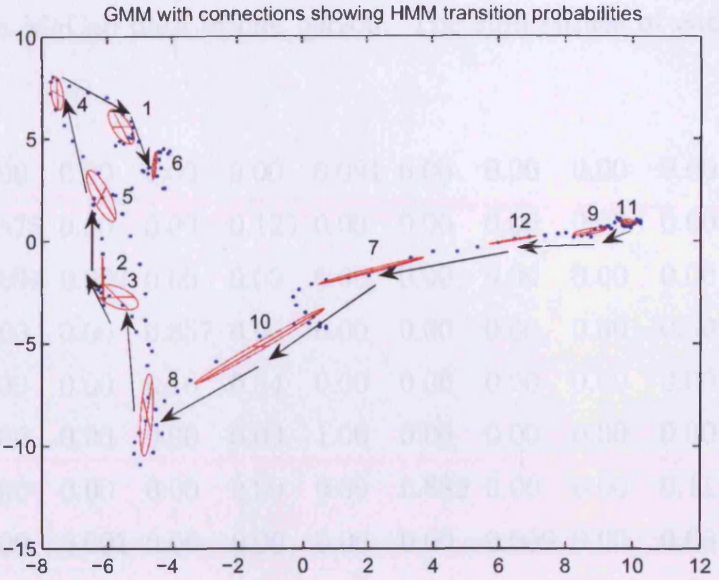


Figure 4.11. GMM with connections showing HMM transition probabilities with values greater than 0.01.

in this state.

Now we have an accurate model of human motion represented by an HMM, where each state in the HMM is represented by a single Gaussian, this model can be used to determine the poses in the video frame.

4.4 Summary

In this chapter, the motion capture system is introduced, which is used to capture the real motion data. The PhaseSpace Motion Digitizer System uses PhaseSpace cameras to capture the motion of subjects who have LEDs attached to their bodies. MotionBuilder software is then used to capture the motion data. These data will be used for modelling human motion, tracking motion of a 3D person in real video, and generating human interactive behaviours for a virtual character

Table 4.1. Probability transition matrix trained on walking and shaking hands MoCap data of one person. The sum values of each row is one.

0.909	0.00	0.00	0.00	0.00	0.091	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.875	0.00	0.00	0.125	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.091	0.909	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.143	0.00	0.00	0.857	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.16	0.84	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.889	0.00	0.00	0.111	0.00	0.00
0.00	0.00	0.091	0.00	0.00	0.00	0.00	0.909	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.917	0.00	0.00	0.083
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.059	0.00	0.941	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.032	0.00	0.968	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.20	0.00	0.00	0.00	0.00	0.80

responding to the tracked person.

This chapter also outlines the basic theory which is used for modelling human interactive behaviours. PCA is a well known method for reducing the dimensionality of the data sets. It allows the computation of a transformation that maps our MoCap data from a high dimensional space to a low dimensional space. The goal of applying PCA to our data is to reduce the dimensionality of the training data, and to retain as much as possible of the variation present in the original data set at the same time. This is because it is difficult to analyse data in the high dimensional space. The HMMs are then trained on the reduced dimensional data to represent interactive behaviours for tracking the motion of a real person in a real video sequence.

In the next chapter we are going to explain in detail the process of tracking the motion of a 3D person in real video using the model of human motion.

Chapter 5

TRACKING A 3D PERSON IN A 2D REAL VIDEO

5.1 Introduction

In the previous chapter, we described a model of human motion. In this chapter, we apply this model to tracking motion of a 3D person in real video sequence.

Analysing human behaviour and tracking the full human body in video have been an active area of research for over twenty years. The recent appearance of cheap digital video equipment has made this area even more appealing to researchers, with applications ranging from biomedical human motion analysis and video surveillance to computer games and film production. Tracking people is a challenging task, because of the high dimensionality of a full body kinematics, the ambiguity caused by body articulation, and the fast movement of the human body. Moreover, loose clothing, mutual occlusion between body parts or shadows may complicate the inference problem. These ambiguities make it hard to track moving body parts.

In this research, the purpose is to create a 3D virtual character capable of responding to actions obtained from observing a real person

in video. To achieve this goal, the motion of a real person in video needs to be tracked. In this chapter, the annealed particle filtering (APF) [12] is applied to track the fully articulated 3D motion of a person in video in conjunction with a model of human geometry and a model of dynamics of human motion built using the techniques described in Chapter 4. The tracked motion of a person is then used for generating the interactive behaviours for a virtual character.

The model of geometry of a human body is deliberately kept simple, which is sufficient for this purpose. It consists of 16 segments connecting 20 vertices positioned on the body and representing places like elbows, knees, etc. These segments are described in truncated cones as shown in Figure 4.5 on Page 46. The silhouette of the produced model of geometry roughly resembles a human figure, which is needed during the tracking process.

The remainder of the chapter is as follows: In Section 5.2 the tracking algorithm APF is discussed. It is used for tracking an articulated 3D motion of a person in real video. The results of tracking in real video is then presented in Section 5.3. Section 5.4 concludes this chapter.

5.2 Tracking Method – Annealed Particle Filtering (APF)

The aim is to track the motion of a 3D person from 2D real video sequence with good accuracy, from frame to frame. There are two problems when tracking an articulated 3D person from 2D video: the first is to locate the person in a video frame, the second is to estimate the articulated motion from 2D data. Since video sequences of a single moving person are captured in a controlled environment, the person in the video sequence can be located easily using thresholding techniques

[88].

In order to track a fully articulated motion of human body efficiently and to obtain the desired information, the annealed particle filtering [12] was adopted. This is based on particle filtering, but modified to avoid the high dimensionality problem of standard PFs. This approach is similar to that of simulated annealing (SA) [106].

SA was developed by Kirkpatrick [106] as a way of handling multiple modes in an optimisation context. It employs a series of distributions, with probability densities given by $p_0(x)$ to $p_M(x)$, in which each $p_m(x)$ differs only from $p_{m+1}(x)$. Samples actually need to be drawn from the distribution $p_0(x)$. The distribution p_M is designed so that the Markov chain used to sample from it allows movement between all regions of the state/search space.

An annealing run is started in some initial state, from which a Markov chain designed to converge to p_M is first simulated. Some number of iterations of a Markov chain designed to converge to p_{M-1} are simulated next, starting from the final state of the previous simulation. The process is continued in this fashion, using the final state of the simulation for p_m as the initial state for the simulation for p_{m-1}, until the chain designed to converge to p_0 is finally simulated.

The idea of annealing for optimisation is now adapted to perform a particle based stochastic search within the framework of an annealed particle filter. The detailed of the APF approach is described in [12].

Assume that a series of weight functions $\omega_0(\mathbf{Z}, \mathbf{X})$ to $\omega_m(\mathbf{Z}, \mathbf{X})$[1] are employed, where m is the number of annealing layers. An un-weighted

[1]The weighting function ω are constructed by two image features: edges and foreground silhouette. The detailed of the weight function can be found in [12].

set of particles are denoted as:

$$\mathbf{S}_{k,m} = \{(\mathbf{s}_{k,m}^{(0)}) \dots (\mathbf{s}_{k,m}^{(N-1)})\} \tag{5.2.1}$$

where N is the number of particles on each layer. A set of weighted particles represents the state of the tracker after an annealing run in each layer m,

$$\mathbf{S}_{k,m}^{\pi} = \{(\mathbf{s}_{k,m}^{(0)}, \pi_{k,m}^{(0)}) \dots (\mathbf{s}_{k,m}^{(N-1)}, \pi_{k,m}^{(N-1)})\} \tag{5.2.2}$$

where $\pi_{k,m}^{(i)}$ is the corresponding particle weighting.

Each annealing run can be broken down as follows [12]:

1. Select the number of layers m and the number of particles N in each layers. For each time t_k an annealing run is started at layer m.

2. Initialise a set of un-weighted particles $\mathbf{S}_{k,m}$ on each layer of an annealing run.

3. Assign a weight on each of the particles,

$$\pi_{k,m}^{(i)} \propto \omega_m(\mathbf{Z}_k, \mathbf{s}_{k,m}^{(i)}) \tag{5.2.3}$$

   which are normalised so that $\sum_N \pi_{k,m}^{(i)} = 1$. The set of weighted particles $\mathbf{S}_{k,m}$ is now formed.

4. N particles are drawn randomly from $\mathbf{S}_{k,m}^{\pi}$ with replacement and with a probability equal to their weighting $\pi_{k,m}^{(i)}$. As the n^{th} parti-

cle $\mathbf{s}_{k,m}^{(n)}$ is chosen, it is used to produce the particle $\mathbf{s}_{k,m-1}^{(n)}$ using:

$$\mathbf{s}_{k,m-1}^{(n)} = \mathbf{s}_{k,m}^{(n)} + \mathbf{B}_m \tag{5.2.4}$$

where $\mathbf{B}_m$ is a multi-variate Gaussian random variable with variance $\mathbf{P}_m$ and mean 0.

5. The set $\mathbf{S}_{k,m-1}$ has now been produced, and can be used to initialise layer $m-1$. The process is repeated from Step 3 to 4 until we arrive at the set $\mathbf{S}_{k,0}$ (layer 0).

6. $\mathbf{S}_{k,0}^{\pi}$ is used to estimate the optimal model configuration χ_k using:

$$\chi_k = \sum_{i=1}^{N} \mathbf{s}_{k,0}^{(i)} \pi_{k,0}^{(i)} \tag{5.2.5}$$

7. The set $\mathbf{S}_{k+1,M}$ is then produced from $\mathbf{S}_{k,0}^{\pi}$ using:

$$\mathbf{s}_{k+1,M}^{(n)} = \mathbf{s}_{k,0}^{(n)} + \mathbf{B}_0 \tag{5.2.6}$$

The set is then used to initialise layer M of the next annealing run at t_{k+1}.

5.3 Tracking the Motion of a Person in Real Video

In the previous section, the APF used to extract 3D articulated motion of a person moving in video from frame to frame is presented. It can avoid the high dimensionality problem of the data-set, and recovers 3D poses of person from the original video sequences. In this section, the model (built using techniques described in Chapter 4) is applied to

recover 3D poses of human figures from the video sequences. To do so, the volumetric model is represented by cylinders (Figure 4.5).

Firstly, statistical background subtraction [88] is used to detect the silhouette of a person in each frame. Assuming that the person's height and other parameters, such as radius of torso, the length of legs and arms are known. Next APF [12] is used to estimate the whole body poses.

5.3.1 Tracking Process

In the tracking process, the 3D pose and position of a person in a real video sequence need to be found. To do this, the following steps are performed:

1. A single video camera is calibrated using Zhang's method [107] and record several video sequences (simple background with no person present and a person moving in the scene).

2. A moving person is identified by subtracting the background, resulting in a binary image (black and white) with the persons silhouette in white, as shown in Figure 5.4.

3. A dynamic model is built on a single person's MoCap data.

4. The whole body pose and 3D information is estimated using the APF.

5. Repeat step 2 to 4 for all frames of the image sequence.

The above steps will be described in detail in the following sections.

5.3.2 Calibrating the Camera

Camera calibration is the process of transformation the 3D position and orientation of the camera frame in world space into 2D image co-ordinates [108–110]. Camera calibration is an important step in many computer vision applications. The intrinsic properties of the camera are obtained through this process, such as focal length, image center and image distortion coefficients. The extrinsic camera parameters, such as translation components and rotation angles for the transformation between the world space and camera co-ordinates, are then obtained by using Zhang's method [107] (Zhang's method, using a simple planar pattern has provided the research community with both an easy-to-use and accurate algorithm for obtaining both intrinsic and extrinsic camera parameters). This algorithm was implemented in the Matlab Camera Calibration Toolbox [111] by Jean-Yves Bouguet and C++ in Intel OpenCV library [112]. These libraries are currently two of the most widely used tools for camera calibration. Both intrinsic and extrinsic camera calibration parameters need to be saved, such as focal length, image distortion coefficients, translation components and rotation angles. Those parameters are used for locating the person in the video frame and merge the virtual character back into original the video sequence.

Video sequences are obtained by recording a person's movement using a single camera. First, simple background frames without a person present are recorded. Then several videos are filmed for one person's action (walking and shaking hands, pulling and pushing) corresponding to the MoCap data which are used in this research to analyse human behaviour and estimate 3D poses of a person in video sequence. All

videos are recorded at 30 fps. Some of them are background videos without a person present (Figure 5.1), some videos show one person walking and shaking hands (Figure 5.2), some videos showed pulling behaviours and the remaining show pushing behaviours. Finally, the recorded video data is exported into a sequence of RGB images. Selected frames from the original video sequence for walking and shaking hands behaviours are shown in Figure 5.3.

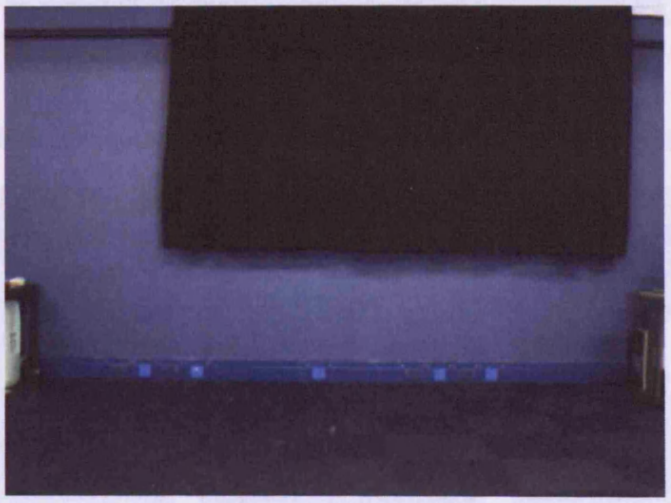


Figure 5.1. Original background image without a person in the scene.

5.3.3 Training Data

The 3D MoCap data are obtained by capturing two people's motion using Phasespace system (Chapter 3). 30 markers are placed on each person at the joints shown in Figure 3.2 on Page 34. The 3D position of markers were recorded at each pose, therefore a pose of a person in each frame is represented by a 90-dimensional vector. The data collected consists of the markers' coordinates in each frame through

Figure 5.2. Original video image with a person present in the scene.

Figure 5.3. Selected video images of one person walking and shaking hands at time 2.2s, 3s and 4.83s respectively.

a number of sequences describing three different types of motion. In total, 14 sequences of MoCap data for shaking hands, 9 sequences of MoCap data for pushing and 7 sequences of MoCap data for pulling have been obtained. All the original MoCap data sequences can be viewed in the CD at back of the thesis in folder **Original-shaking hands**, **Original-pushing** and **Original-pulling** respectively.

5.3.4 Subtracting Background

To locate the 3D position of a person in an image, the person's contour need to be first determined by subtracting the background image from all other video frames. For this purpose, the statistical subtraction method described by Horprasert *et al.* [88] is used to detect a moving foreground object (the shape of the person) from a background scene using color images.

The basic idea of background subtraction is to subtract the image B from a reference image A that models the background scene. The basic steps of the algorithm are as follows [88].

- Background modelling constructs a reference image representing the background. In the background training process, the reference background image and parameters are computed over a number of background images.

- Threshold selection determines appropriate threshold values by a statistical learning procedure used in the subtraction operation to obtain a desired detection rate.

- Pixel classification classifies the type of a given pixel, that is, the pixel is the part of background, or it is a moving foreground object. In this step, the difference between the background image A and the current image B is evaluated. When the difference between each pixel is greater than the threshold value, then this pixel will change to white in image B. Otherwise, when the difference between each pixel is less than the threshold value, this pixel will change to black in image B. As a result, the black is the background and the white is the silhouette of the human body.

Figure 5.4 shows the result of the image after subtracting the background.

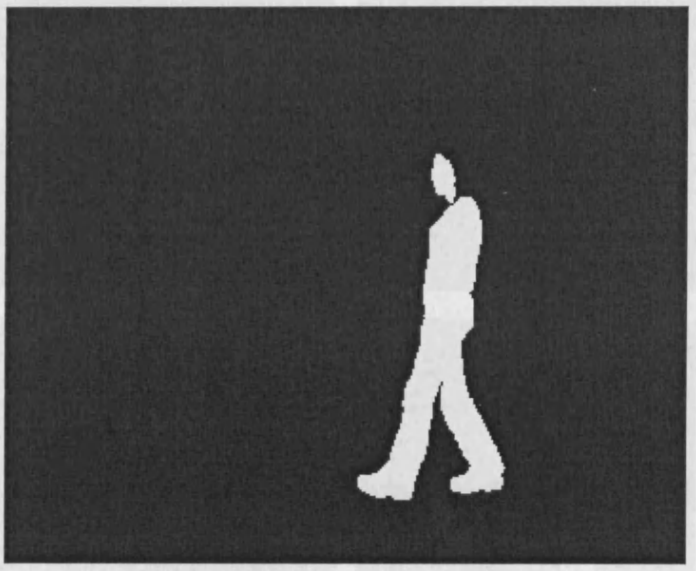


Figure 5.4. Binary image after subtracting background.

Selection frames from the result of the image after subtracting the background are shown in Figure 5.5. These result images show the robustness and reliability of the statistical subtraction algorithm.

5.3.5 Model of Human Motion

In the following experiments, the model of dynamics of the motion of a single person is represented using an HMM as described in Chapter 4. The model of geometry of the human body used in the following experiments is described in Chapter 4, Figure 4.5. An HMM is defined as

$$\lambda = (\mathbf{A}, \mathbf{B}, \pi) \tag{5.3.1}$$

where $\mathbf{A} = \{a_{ij}\}$ is the state transition probability matrix, $\mathbf{B} = \{b_j(v_t)\}$ where $b_j(v_t)$ is the observation density distribution at state j and π is

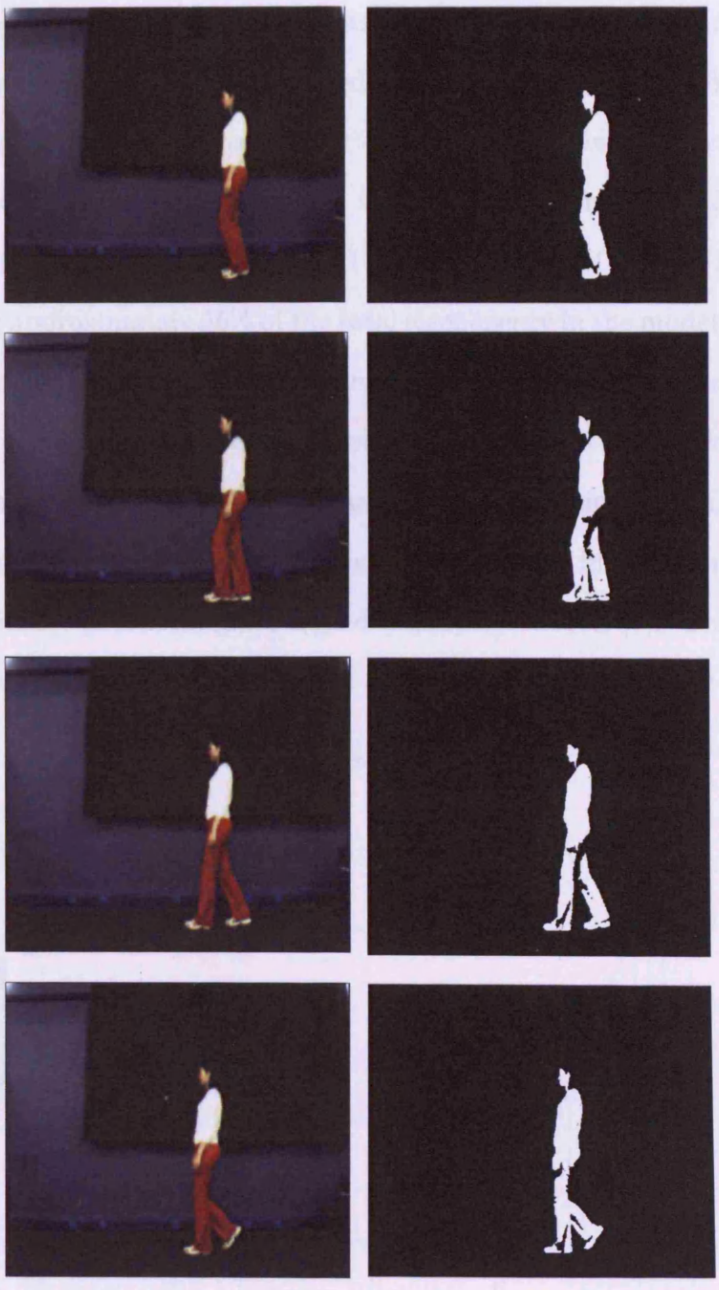


Figure 5.5. Selection frames from Original video sequence and binary images after subtracting background.

the initial state probability distribution.

To train the model of dynamics, the MoCap data for one person need to be used. In the experiments several sets of motion data in 3D space with 30 markers are captured, therefore a pose in each frame is represented by a 90-dimensional vector. Such data is always constrained by physical and dynamical factors, thus the dimensionality of the data set need to be reduced using PCA. The model is trained on 1600 frames, keeping approximately 90% of the total eigenenergy in the model, which accounts for seven largest eigenvectors, and approximately 60 states are used in the model. Figure 5.6 shows the relations between percentage of eigenenergy and number of dimensions. As mentioned in Chapter 2, six to eight dimensions are enough to represent a human jump that looks similar to the original high-dimensional version [7]. From the figure, we decided to choose 90% of the total eigenenergy in our model.

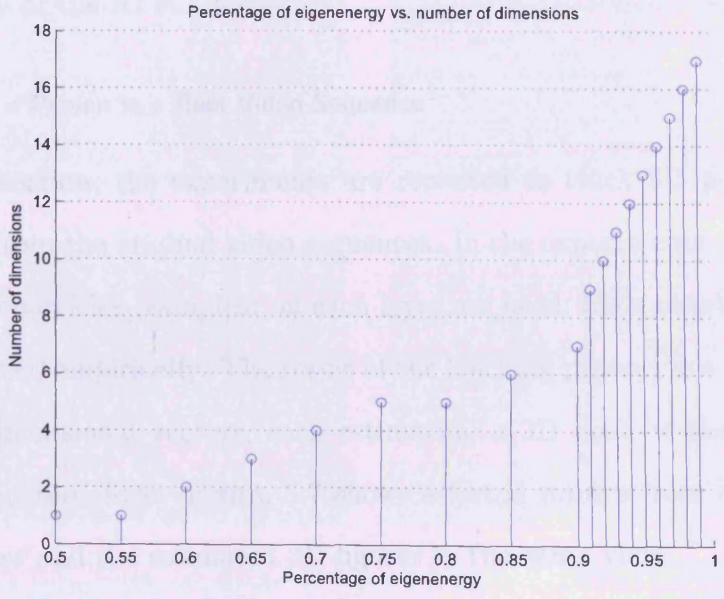


Figure 5.6. Percentage of eigenenergy vs. number of dimensions

Then the HMM on a number of such vectors is trained using the Baum-Welch algorithm [13]. Finally, the APF algorithm as described in Section 5.2 is used to track the motion of a person in real video.

5.3.6 Tracking Results

The 3D articulated motion of a person in a 2D real video sequence needs to be tracked. The video sequence is preprocessed by subtracting the background [88] and thus a sequence of binary images (Figure 5.5) is obtained. Next, the APF described in Section 5.2 is used together with the HMM trained in the previous stage to estimate the 3D poses of the tracked person in the video.

In the following experiments, we use the same tracking algorithm to track a person in a real video sequence and track a person using the synthesised data. The purpose of these experiments is to test the accuracy of the APF.

Tracking a Person in a Real Video Sequence

In this section, the experiments are repeated to track 3D poses of a person from the original video sequences. In the experiments, 10 layers and 256 particles (samples) on each layer are used; both numbers were determined empirically. The result of the tracking process is a sequence of 90 dimensional vectors, each estimating a 3D pose of the tracked person in the video. Figure 5.7 shows selected frames from the video sequences and the estimated 3D figures in the same view.

Since there are no markers placed on the human body in the real video sequence, the only way to assess the accuracy of tracking in video sequence is visual. In the next section, the synthesised data are used

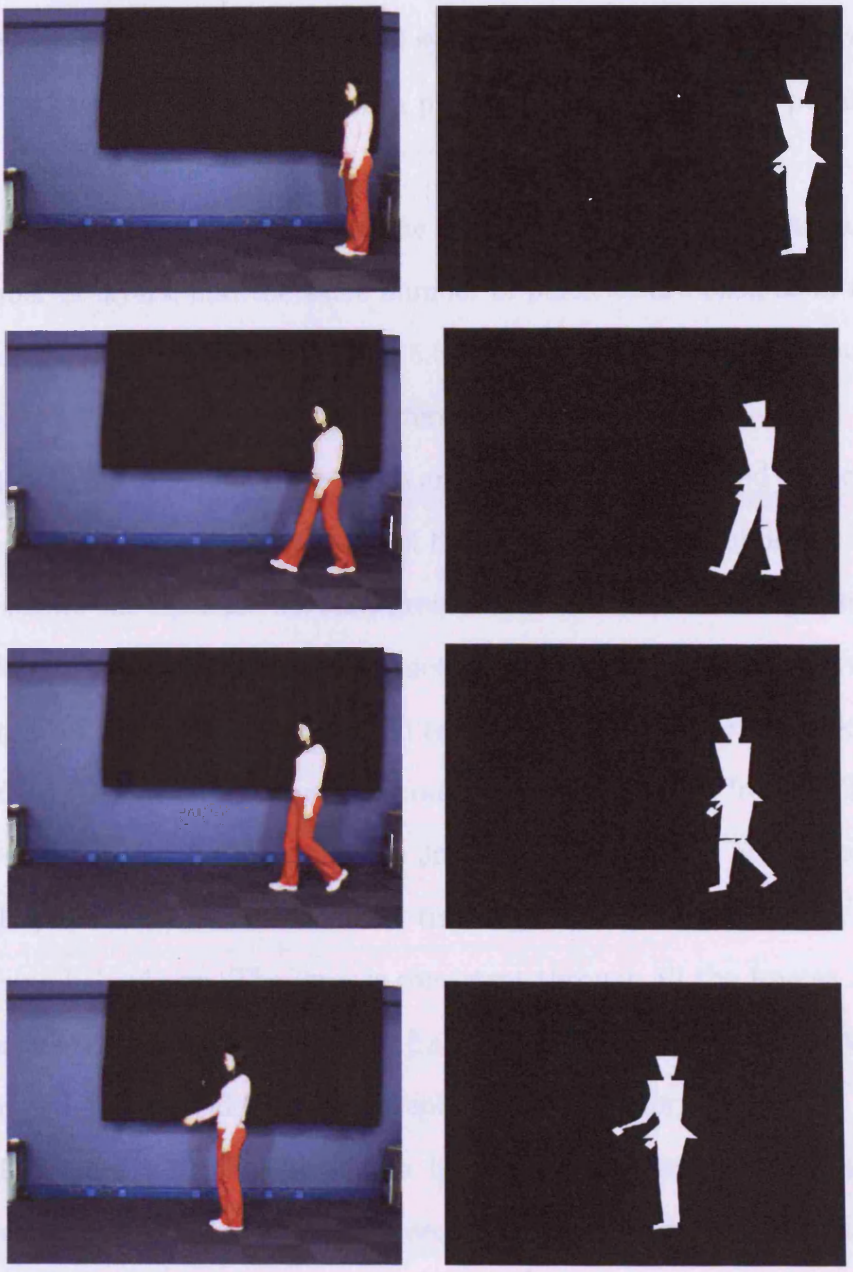


Figure 5.7. Original images (first column) and the estimated 3D figures in the same view (second column).

to assess the accuracy of tracking.

Tracking a Person using the Synthesised Frames

In this section, several sequences of synthetic frames are generated from the 3D MoCap data of one person performing shaking hands, pushing and pulling motion.

In the following experiments, the same tracking algorithm, the same number of layers, and the same number of particles are used as in the above real video sequence. Figure 5.8 - 5.10 shows the tracking results for the synthetic data on three different types of motion.

For Figure 5.8, blue trajectories are ground truth while red trajectories are tracking result in the top of figure. In the bottom of figure, the line shows the distance between ground truth and the tracking result. It is clear to see that the tracked motion is similar to the ground truth in most of frames. In the frame 30 to 40, the average error of tracking is 7mm, which is worse than the tracking precision in all frames. The overall average error of tracking is 3mm, so we feel is still acceptable.

Figure 5.9 shows the error of tracking on a synthesised data for pushing behaviours. The error is consistent through all the frames and its average value through all the frames is around 2mm. It is a very good estimate considering the absence of any 3D information.

In Figure 5.10, The error is a little worse in the last 30 frames, which stabilises in the rest of the sequence. Overall, the average error is a little worse than in the pushing motion for the same person, which can be attributed to a different type of motion.

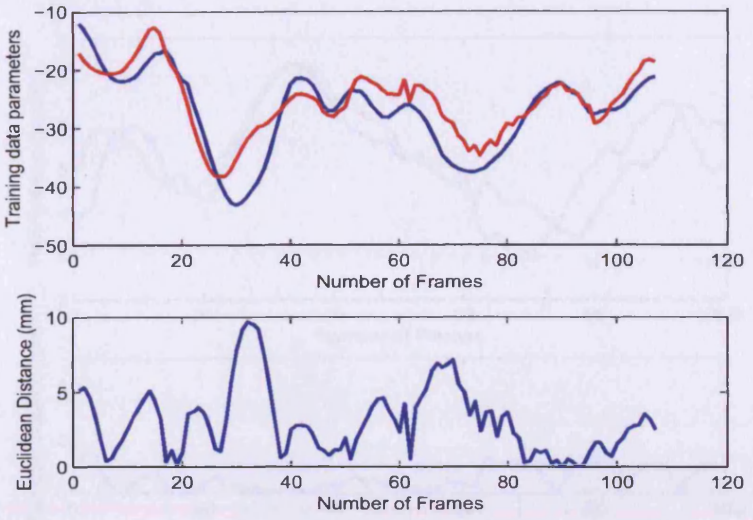


Figure 5.8. Tracking result for the synthesised data (shaking hands behaviours). In the top of figure, blue trajectories are ground truth while red trajectories are tracking result. In the bottom of figure, the line shows the distance between ground truth and the tracking result.

5.4 Summary

In this chapter, we have described the details of APF for tracking 3D human motion in real video. It can search high dimensional configuration spaces and is capable of recovering full articulated body motion efficiently. The model of human motion and the APF are applied to recover 3D poses of human figures from the video sequences. The good tracking results are obtained in the experiments, however, the results are dependent on how well the built models represent the tracked person. In the next chapter, the tracked motion of a person is used for generating the interactive behaviours for a virtual character.

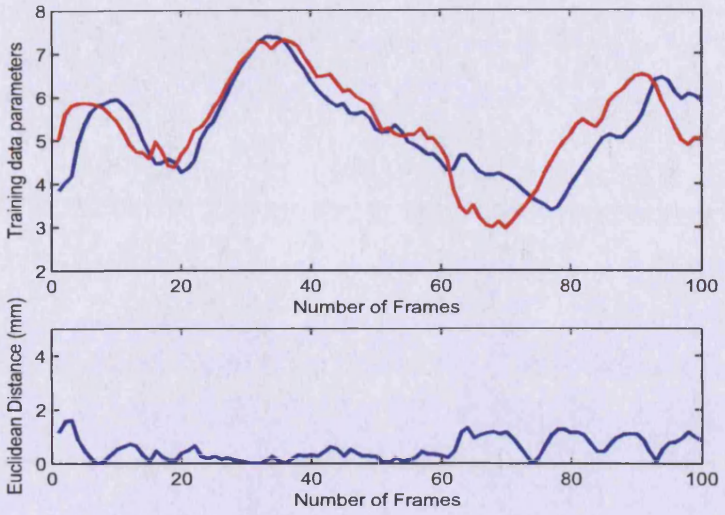


Figure 5.9. Tracking result for the synthesised data (pushing behaviours). In the top of figure, blue trajectories are ground truth while red trajectories are tracking result. In the bottom of figure, the line shows the distance between ground truth and the tracking result.

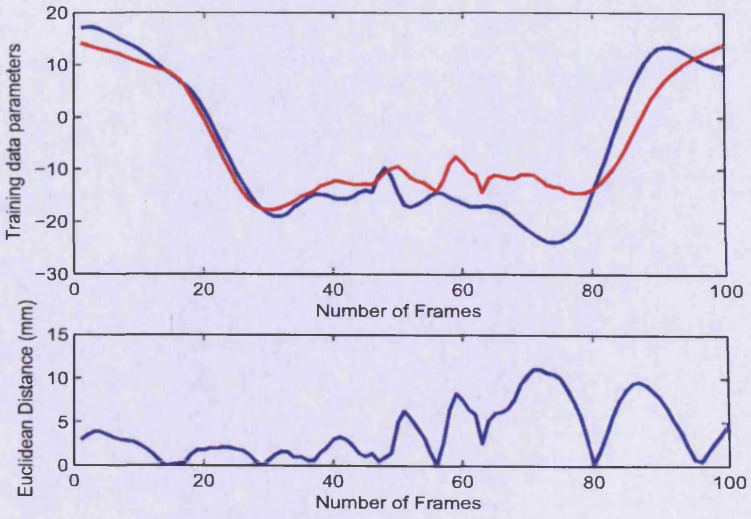


Figure 5.10. Tracking result for the synthesised data (pulling behaviours). In the top of figure, blue trajectories are ground truth while red trajectories are tracking result. In the bottom of figure, the line shows the distance between ground truth and the tracking result.

Chapter 6

GENERATING BEHAVIORS FOR A VIRTUAL CHARACTER

In the previous two chapters, we described how the model of human motion was built and how to track motion of a real person in the video. In this chapter, the process of generating interactive behaviours for a virtual character responding to the tracked person is presented.

6.1 Introduction

In recent years, many researchers have become interested in producing virtual worlds and populating them with virtual characters [15–19]. There has been also a limited amount of research into enabling virtual characters with the ability to produce intelligent behaviour on the basis of visual analysis of the scene, which mainly was conducted in the computer vision area. The applications for this technology include the areas of films, computer games and virtual environments (for the visual creation of 3D characters to populate virtual environments and used as virtual actors for film and television).

The main contributions presented in this chapter are:

- A novel approach for generating intelligent behaviours for fully articulated 3D virtual characters on the basis of visual analysis of the motion of a real person in ordinary 2D video using the dual-input HMM and the standard Viterbi algorithm.

- A new approach for generating interactive behaviours for virtual characters using the windowed Viterbi algorithm, with real-time capability.

To do this, a dual-input HMM is learnt on 3D MoCap data of two individuals motion. The dual-input HMM has two sets of states. The first set of states models the poses for person **A**, and the second set of states models the poses for person **B**. Each state in the model is modelled with a single Gaussian. The standard Viterbi algorithm and the windowed Viterbi algorithm are then used to generate the responsive behaviours for a virtual character, given a sequence of 3D poses of the tracked person (for person **A**) in video. Finally, the generated motion is mapped onto a virtual character and the virtual character is placed back into real video.

The organisation of the Chapter is as follows: in Section 6.2, the dual-input HMM trained on 3D MoCap data of two persons' interactive behaviours is described. The behaviour generating part is explained in Section 6.3 and the methodology to place the virtual character into real video is outlined in Section 6.4. The experimental results and the assessment of the accuracy of the generated behaviours in Sections 6.5 and 6.6 respectively, followed by the visual inspection of the generated motion by ten independent observers are presented in Section 6.7.

6.2 Model of Interactive Behaviour

The model is trained on the 3D MoCap data of two real persons. It can represent a variety of interactive behaviours. The model uses a dual-input HMM. In particular, the model is trained on the following behaviours: a handshake between two people; one person pulling another person,the and one person pushing another person. However, the model can be trained to represent other types of motion given the appropriate training data. In the experiments, the PhaseSpace Motion Digitizer System (Chapter 4.1) is used to capture several sets of motion data in 3D space with 30 markers. Therefore a pose in each frame is represented by a 90-dimensional vector. Such data is always constrained by physical and dynamic factors, thus the dimensionality of the data set needs to be reduced using PCA before proceeding with anything else. Approximately 90% of the total eigenenergy are kept in the model, which accounts for seven largest eigenvectors, and then train the HMM on a number of such vectors.

6.2.1 Training Data

The data used in this research is 3D MoCap data captured using the motion capture system described in Chapter 4.1. As mentioned earlier, this system consists of 12 specialist cameras. 30 markers are placed on the person's body, as shown in Figure 3.2 on Page 34. The data collected consists of the markers' coordinates in each frame. Three different types of interactive behaviours (walking and shaking hand, one person pulling another person and one person pushing another person) are captured for person **A** and person **B**.

6.2.2 Dual-Input HMM Construction

Dual-input HMMs were described by Brand in [113]. In his work, he mapped audio parameters with visual shape and velocity parameters through an entropic HMM, hence estimating of the hidden visual state sequence from a new speech observation. In this thesis, the method employs the standard HMM to model the human interactive behaviours for two persons. After the HMM stage, motion parameters are calculated for a virtual character from a state sequence.

The model is trained on the 3D MoCap data of two real people. The model uses a dual-input HMM (each HMM describes the motion of a whole human body), with two sets of states. One set of states models the poses for person **A**. Likewise, the other set of states models the poses for person **B**. Each state is modelled with a single Gaussian variable. The model is trained on 1600 frames, keeping approximately 90% of the total eigenenergy in the model, which accounts for seven largest eigenvectors (See Figure 5.6 on Page 74).

The construction process is as follows. First an n state HMM is defined as:

$$\lambda_{\mathbf{B}} = (\mathbf{A_B}, \mathbf{B_B}, \pi_{\mathbf{B}}) \tag{6.2.1}$$

where $\mathbf{A_B}$ is the state transition probability matrix, $\mathbf{B_B}$ is the observation probability distribution, and $\pi_{\mathbf{B}}$ is the initial state probability distribution. It is constructed using the training data representing the motion of person **B**. The mean and covariance of each state in $\lambda_{\mathbf{B}}$ are defined as $\mu^i_{\mathbf{B}}$ and $\Sigma^i_{\mathbf{B}}$, where $i = 1, \ldots, n$.

After training this HMM, the matrix $\gamma_t(i)$ is also automatically obtained, which defines the probability of being in state i at time t.

Since the length of the observation sequence is equal to the number of training vectors, $T = N$ during training. Using the matrix $\gamma_t(i)$, we build a second HMM with the new n states which is defined as

$$\lambda_{\mathbf{A}} = (\mathbf{A_B}, \mathbf{B_B}, \pi_{\mathbf{B}}) \tag{6.2.2}$$

It has the same transition, observation and prior probabilities as $\lambda_{\mathbf{B}}$. The means $\mu^i_{\mathbf{A}}$ and covariances $\Sigma^i_{\mathbf{A}}$ are calculated from the training data representing the motion of person **A**.

$$\mu^i_{\mathbf{A}} = \frac{\Sigma_{t=1}^{T} \gamma_t(i) \cdot \mathbf{b}^t_{\mathbf{A}}}{\Sigma_{t=1}^{T} \gamma_t(i)} \quad 1 \leq i \leq n, \quad T = N \tag{6.2.3}$$

and

$$\Sigma^i_{\mathbf{A}} = \frac{\Sigma_{t=1}^{T} \gamma_t(i) \cdot (\mathbf{b}^t_{\mathbf{A}} - \mu^i_{\mathbf{A}})(\mathbf{b}^t_{\mathbf{A}} - \mu^i_{\mathbf{A}})^T}{\Sigma_{t=1}^{T} \gamma_t(i)} \quad 1 \leq i \leq n, \quad T = N \tag{6.2.4}$$

where $\mathbf{b}$ is the observation sequence of person **A**'s motion data.

Given $\lambda_{\mathbf{A}}$ and a sequence of 3D poses of the tracked person in video, we can estimate the best state sequence Q using the standard Viterbi algorithm. Given the shared transition, observation and initial probabilities between $\lambda_{\mathbf{B}}$ and $\lambda_{\mathbf{A}}$, the means and covariance of $\lambda_{\mathbf{A}}$ are constructed using the state observation matrix $\gamma_t(i)$ and the training data representing the motion of person **A**, the state sequence Q also corresponds to a state sequence through $\lambda_{\mathbf{B}}$. The state sequence Q defining a set of hidden states through both λ_B and $\lambda_{\mathbf{A}}$, also relates to a unique set of mean and covariance matrices of the training data representing the motion of person **B** for each time step t, and a unique set of mean

and covariance matrices of the training data representing the motion of person **A** for each time step t. This way, the shared properties between $\lambda_{\mathbf{B}}$ and $\lambda_{\mathbf{A}}$ associate Q_t with $\mu^t_{\mathbf{B}}$ and $\Sigma^t_{\mathbf{B}}$, as well as $\mu^t_{\mathbf{A}}$ and $\Sigma^t_{\mathbf{A}}$.

The HMM $\lambda_{\mathbf{A}}$ is defined as a dual-input HMM since it has two sets of means and covariance's, constructed using person **A** and person **B**'s motion data respectively. This is different from coupled HMM in which two HMM are coupled together.

A summary of the dual-input HMM is as follows:

1. Given the training data representing the motion of person **B**, build the HMM $\lambda_{\mathbf{B}}$ and keep the matrix $\gamma_t(i)$.

2. Using $\gamma_t(i)$, the training data for person **A** corresponding to the training data representing the motion of person **B** used in $\lambda_{\mathbf{B}}$ construction, and calculate the new means $\mu^i_{\mathbf{A}}$ and covariances $\Sigma^i_{\mathbf{A}}$ using Equation (6.2.3) and (6.2.4).

3. Define the dual-input HMM $\lambda_{\mathbf{A}}$ using the new means and covariances. It has the same transition, observation and prior probabilities as $\lambda_{\mathbf{B}}$.

4. Using $\lambda_{\mathbf{B}}$, classify the motion data for person **B** for each observation to a state, by minimising the Mahalanobis distance .

5. Using the one-to-one first-and-second correspondences from the training set, form clusters for each HMM state.

Using this dual-input HMM, a sequence of 3D poses for a virtual character can be estimated, given a sequence of 3D poses of the tracked person in the video. This is described in the next section.

6.3 Generating Interactive Behaviours

In the virtual character generating system, the 3D articulated motion of a real person in a video is tracked using the APF described in the previous chapter, after which the obtained 3D data for the tracked person are used in combination with the dual-input HMM and the standard Viterbi algorithm or the windowed Viterbi algorithm to generate the responsive behaviour for a virtual character. Finally, the virtual character performing the generated motion can be placed back into the original video sequence, as shown in Figures 6.8, 6.9 and 6.10.

In the following subsections the theory of both the standard Viterbi algorithm and the windowed Viterbi algorithm are described briefly. Then the process of generating interactive behaviours is described given a sequence of poses (tracking result) for person **A**.

6.3.1 The Standard Viterbi Algorithm

The standard Viterbi algorithm is able to find the single best state sequence $\mathbf{Q} = \{q_1, q_2, \ldots, q_T\}$ for given an observation sequence $\mathbf{O} = \{o_1, o_2, \ldots, o_T\}$. For example, this will be a good "animation" synthesis of a given sequence. The best state sequence is the most likely state sequence that could have generated the observation sequence. A terms need to be defined:

$$\delta_t(i) = \max_i P(\mathbf{Q}, \mathbf{O} \mid \lambda) \tag{6.3.1}$$

where $\delta_t(i)$ is the highest probability along a single path at time t. Then,

$$\delta_{t+1}(j) = \max_i [\delta_t(i) a_{ij}] b_j(o_{t+1}) \tag{6.3.2}$$

In the experiments, the observation sequence is the original MoCap data of person **A**. The standard Viterbi algorithm can be described as follows [13]:

1. Intialise the probability calculated by the initial hidden state probabilities with the associated observation probabilities.

$$\delta_1(i) = \pi_i b_i(o_1) \tag{6.3.3}$$

$$\psi_1(i) = 0 \tag{6.3.4}$$

where $\psi_t(i)$ is used to keep track of the argument which maximises Equation (6.3.1) for each t and j.

2. Consider all products of transition probabilities a_{ij} with the maximal probabilities derived for the preceding step to determine the most probable route to the next state. Keep the highest probability and its corresponding path at iteration.

$$\delta_t(j) = \max_i [\delta_{t-1}(i) a_{ij}] b_j(o_t) \qquad 2 \leq t \leq T \tag{6.3.5}$$

$$\psi_t(j) = \arg\max_i [\delta_{t-1}(i) a_{ij}] \qquad 2 \leq t \leq T \tag{6.3.6}$$

3. Determine the path (state sequence) at final iteration.

$$q_T = \arg\max_i [\delta_T(i)] \tag{6.3.7}$$

4. Backtrack through the path, following the most probable route. The sequence obtained from this process will hold the most probable sequence of hidden states for a given observation sequence.

$$q_t = \psi_{t+1}(q_{t+1}) \qquad t = T-1, T-2, \ldots 1 \tag{6.3.8}$$

The process of the standard Viterbi algorithm is described in Figure 6.1. S_i is the sequence of hidden states, $a_{max}(t)$ is the maximum probability at time t and T is the number of frames.

The standard Viterbi algorithm requires the full observation sequence $O = \{o_1, o_2, \ldots, o_T\}$ before the processing starts, which makes real-time processing impossible. In order to exploit the benefits of the Viterbi without having to wait for the entire full state sequence, we employ the windowed Viterbi algorithm [14, 114] that can obtain the best state sequence. This is achieved by processing the full state sequence in a block wise manner (instead of the whole data), thus it can be used in a real-time system.

6.3.2 The Windowed Viterbi Algorithm

In a real-time system, the windowed Viterbi algorithm will cause a small delay in the state estimate, but as long as the delay is not too long, the estimate may still be useful. For example, if the training data are captured at 30 frames per second, and the window size is chosen by 10 frames. Thus the delay time only is 0.033 second. It is

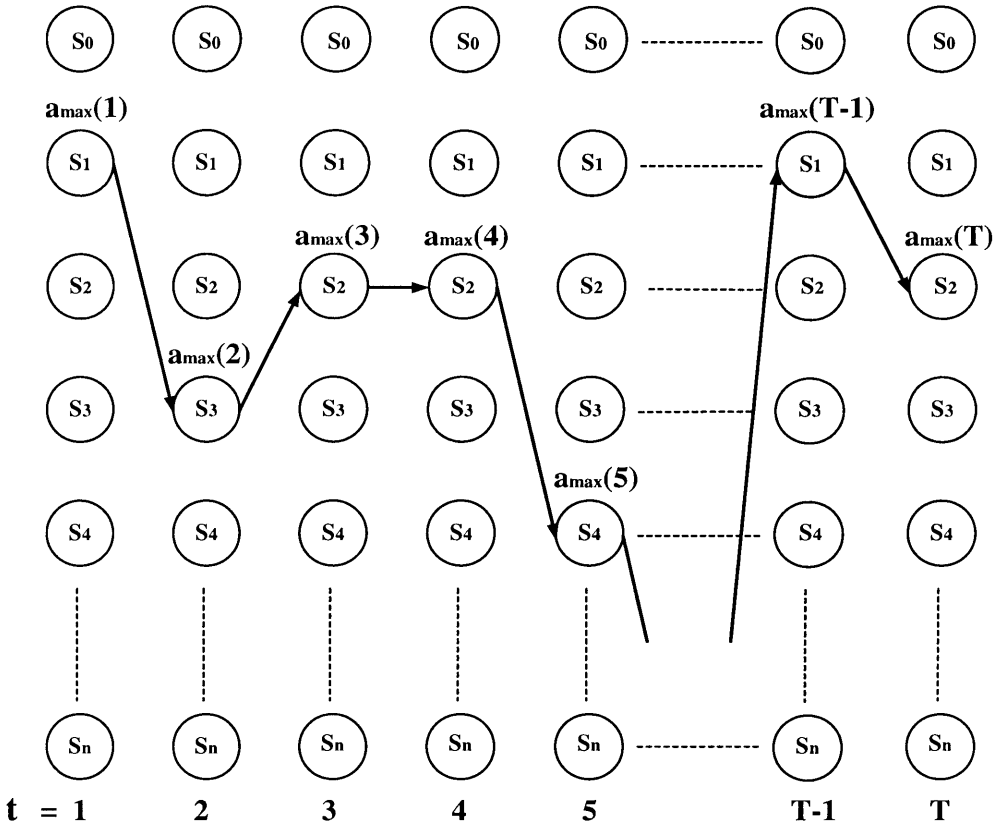


Figure 6.1. Illustration of the Viterbi Trellis. S_i is the sequence of hidden states and $a_{max}(t)$ is the maximum probability at the time t. T is the number of frames.

not usually noticeable in most of the generated motions. However, it is possible to avoid this delay by adjusting the time correspondences between interacting people in the training data.

The standard Viterbi algorithm extract the best path with the assumption that the whole motion sequence is available, that is a non-causal filter of length and delay equal to the sequence length. This might be impractical in some applications. Moreover memory and efficiency problem will be caused for long sequences the length of the trellis. In order to avoid that, a method is devised that uses a windowed trellis that probes the best future path and uses only the first portion of it to make up the overall path.

At any given time t, a time slice of T samples is taken and used as the input to the Viterbi algorithm. We retain only the second state q_2 of the sequence of states $\mathbf{Q} = \{q_1, q_2, \ldots, q_T\}$ corresponding to the best path for the actual time slice.

In summary, the windowed Viterbi algorithm can be described in the following way [14].

1. Select the length of the window T and select the initial state probability π for the n states using the following way as in [14].

$$\pi_{\frac{n+1}{2}} = 1, \pi_{i \neq \frac{n+1}{2}} = 0 \tag{6.3.9}$$

2. The best path $\{q_1, q_2, \ldots, q_T\}$ for the window T can be obtained using the standard Viterbi processing.

3. Retain the second state q_2 as the output at time t, and let

$$\pi_{q_2} = 1, \pi_{i \neq q_2} = 0 \tag{6.3.10}$$

   for processing of the next window.

4. Repeat steps 2-3 (slice trellis forward) until the end state is reached.

Assume a window $T = 5$ samples of the input sequence is chosen, and the windowed Viterbi algorithm is applied to it. Figure 6.2 illustrates the general process of the windowed Viterbi algorithm. The dashed blue box represents the length of the window, the pink node represents the state at time $t = 2$ which is used to build up the output sequence at time $t = 1$ (see black node in the Figure 6.2).

In the next section, the obtained motion data for the tracked person in the video is fed to the dual-input HMM with the standard Viterbi al-

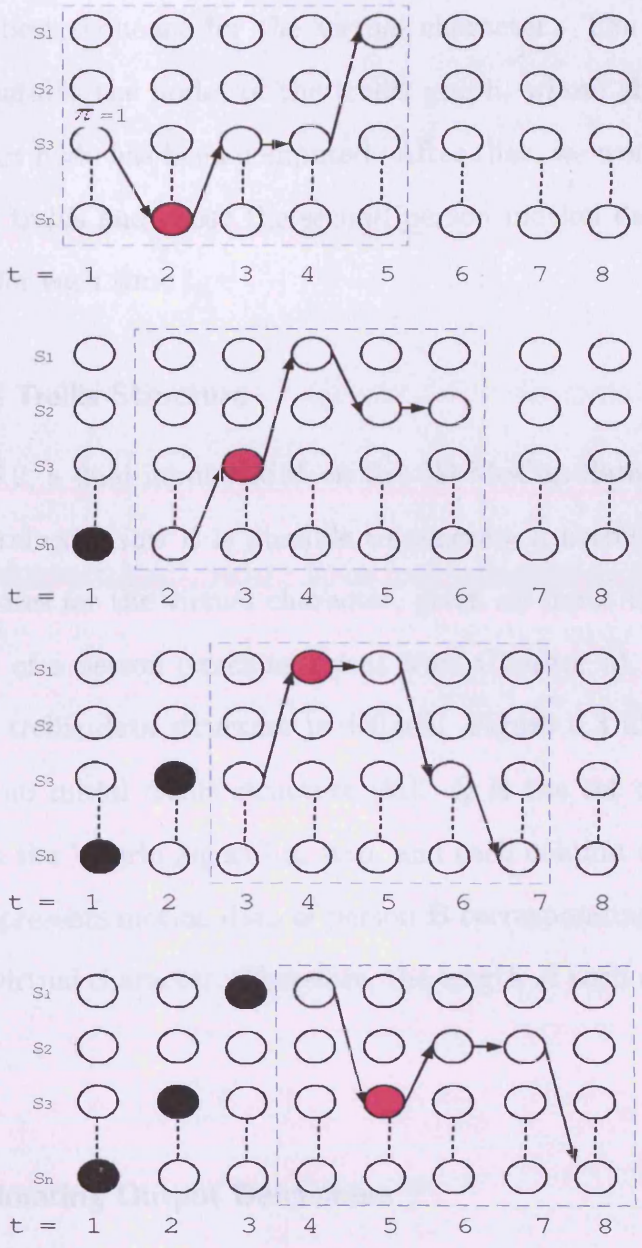


Figure 6.2. Illustration of the windowed Viterbi algorithm. S_i is the sequence of hidden states and $\pi_{\frac{n+1}{2}}$ is the maximum probability at the time $t = 1$. The dashed blue box represents the length of the window T. The pink node represents the second state of the best path for the window T. The black node represents the output at the time t.

gorithm, or the windowed Viterbi algorithm (to generate the responsive behaviour for a virtual character). A trellis data structure is defined to find the best sequence for the virtual character. The trellis data structure contains the nodes of the trellis graph, whose shortest path from the start node has been computed. After that, we work backward through the trellis and chose the second person motion data with the lowest cost for each time t.

6.3.3 The Trellis Structure

In Section 6.2, a dual-input HMM on the 3D MoCap data of two real persons is trained. Now it is possible to generate a corresponding sequence of poses for the virtual character, given an input the sequence of 3D poses of a person (tracking result from Chapter 5). To achieve this goal, a trellis data structure is defined. Figure 6.3 illustrates an example of an initial trellis structure [41]. Q is the set of states resulting from the Viterbi algorithm step, and each column of the trellis structure represents motion data of person $\mathbf{B}$ corresponding to the motion of the virtual character. Therefore, the length of each column may vary.

6.3.4 Estimating Output Behaviours

When the trellis is built, the best path needs to be found through this trellis according to the minimum distance between each data vector and a state at time t. Thus, an error value (the Mahalanobis distance [115]) to each element in each column is assigned, and then we work backward through the trellis to choose the motion data for the virtual character

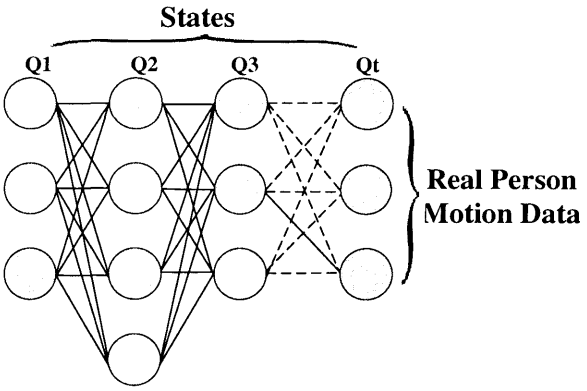


Figure 6.3. Initial Trellis data structure for generating motion data. Q is a set of states.

with the lowest cost for each time t.

Figure 6.4 shows how errors are calculated according to the trellis. $\mathbf{C}_t^j$ is the data vector for person **B** for cell j at time t, $\mathbf{b}_{input}$ is the new input signal (the tracking result from previous chapter) and $\mathbf{b}_a$ is the data vector for person **A**, which has the same location as the data vector for person **B**.

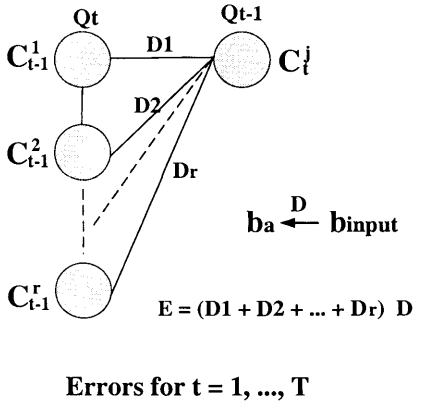


Figure 6.4. A representation of error Calculation for the generating motion data. $\mathbf{C}_t^j$ is the data vector for person **B** for cell j at time t, $\mathbf{b}_{input}$ is the new input signal, $\mathbf{b}_a$ is the data vector for person **A**, E represents the error.

For state 1 ($t = 1$) of the trellis structure, we only have the distance between the new input signal $\mathbf{b}_{input}$ and the data vector $\mathbf{b}_a$ for person

A (denoted as D in Figure 6.4). Therefore, we calculate the error using:

$$E(\mathbf{C}_i, Q_1) = (\mathbf{b}_a^i - \mathbf{b}_{input}^1)^T \boldsymbol{\Sigma}_{Q_1}^{\mathbf{A}} (\mathbf{b}_a^i - \mathbf{b}_{input}^1)$$

$$i = 1, \ldots, p \tag{6.3.11}$$

where $E(\mathbf{C}_i, Q_1)$ is the error between the new input signal and the data for person **A** in state Q_1, p is the number of parameters $\mathbf{C}_i$ in state Q_1, $\mathbf{b}_{input}^1$ is the input signal vector observed at time $t = 1$, and $\boldsymbol{\Sigma}_{Q_1}^{\mathbf{A}}$ is the covariances matrix of $\lambda_{\mathbf{A}}$ for state Q_1.

For the remaining states $t = 2, ..., T$ (T is the number of frames of the new input signal), the distances between $\mathbf{C}_t^j$ in a column at time t and $\mathbf{C}_{t-1}^j$ in a column at time $t-1$ (denoted as D_1, D_2 and D_r) are calculated. Thus, the error can be obtained as $E = (D_1 + D_2 + \ldots + D_r) \times D$ for other state as follows.

$$E(\mathbf{C}_j, Q_t) = [\sum_{i=1}^{r} (\mathbf{C}_{t-1}^i - \mathbf{C}_t^j)^T \boldsymbol{\Sigma}_{Q_t}^{\mathbf{B}} (\mathbf{C}_{t-1}^i - \mathbf{C}_t^j)] \times (\mathbf{b}_a^j - \mathbf{b}_{input}^t)^T \boldsymbol{\Sigma}_{Q_t}^{\mathbf{A}} (\mathbf{b}_a^j - \mathbf{b}_{input}^t)$$

$$j = 1, \ldots, p \tag{6.3.12}$$

where $E(\mathbf{C}_j, Q_t)$ is the error between $\mathbf{C}_t^j$ in a column at time t and $\mathbf{C}_{t-1}^j$ in a column at time $t-1$, p is the number of parameters $\mathbf{C}_j$ in state Q_t, r is the number of parameters $\mathbf{C}_i$ in state Q_{t-1} and $\boldsymbol{\Sigma}_{Q_t}^{\mathbf{B}}$ is the covariances matrix of $\lambda_{\mathbf{B}}$ for state Q_t.

When the errors are calculated in the trellis, the best path is calculated by working backwards through the trellis (from $T \geq t \geq 1$), and choosing the motion data $\mathbf{C}_j^{out}$ for the virtual character in each column

with the lowest error value at time t.

$$\mathbf{C}_j^{out} = \min E(\mathbf{C}_j, Q_t); \qquad j = 1, \ldots, n; \qquad t = T, \ldots, 1. \quad (6.3.13)$$

Through this process, the generated interactive behaviour for the virtual character can thus be obtained.

6.3.5 Trajectory Post-processing

Through the above process, the generated interactive behaviour for a virtual character can be achieved. Then trajectory post-processing is employed to smooth the estimated sequence of poses. A good approach is to use a weighted averaging filter on the estimated sequence of poses, thus removing spikes in the trajectory caused by the incorrect motion choice. This is because the spike causes large changes in the estimated motion over short time periods. Finally, the motion data is transformed back to original dimensional space.

6.3.6 Motion Resynthesis Summary

A summary of the motion resynthesis is as follows.

1. Calculate the best state sequence Q through the dual-input HMM using the input sequence of 3D poses of a person $\mathbf{b}_{input}^t$ (the tracked result), where $t = 1, \ldots, T$.

2. Using the state sequence Q, construct a trellis of the parameter $\mathbf{C}$.

3. Assign errors to each trellis mode at $t = 1$, using Equation

(6.3.11).

4. Assign errors to each trellis mode at $t = 2, \ldots, T$, using Equation (6.3.12).

5. Choose the best state $\mathbf{C}_j^{out}$ from T to 1, using Equation (6.3.13).

6. Apply post-processing (Section 6.3.5) to the motion data for the virtual character.

Using the approach described in this section, the response behaviour was generated for the virtual character from the dual-input HMM, given three different types of motion data for the tracked person in the video as input. In all three cases an appropriate behaviour was generated, and the generated behaviour appeared naturalistic. All the generated videos can be viewed in the CD at back of the thesis. The videos in folder **Generated results - standard** are generated using the standard Viterbi algorithm, and the videos in folder **Generated results - windowed** are generated using the windowed Viterbi algorithm.

6.4 Placing Virtual Character Back into the Real Video

In this section, a virtual character is placed back into the real video. Two main commercial software packages, MotionBuilder [8] and Shake [116], are used for producing the animation sequences. A summary of the production process is now provided.

1. Mapping the generated motion onto a 3D actor with MotionBuilder. The process of mapping to a 3D actor can be found in Appendix B. Figure 6.5 shows the generated motion is matched with a 3D actor in MotionBuilder.

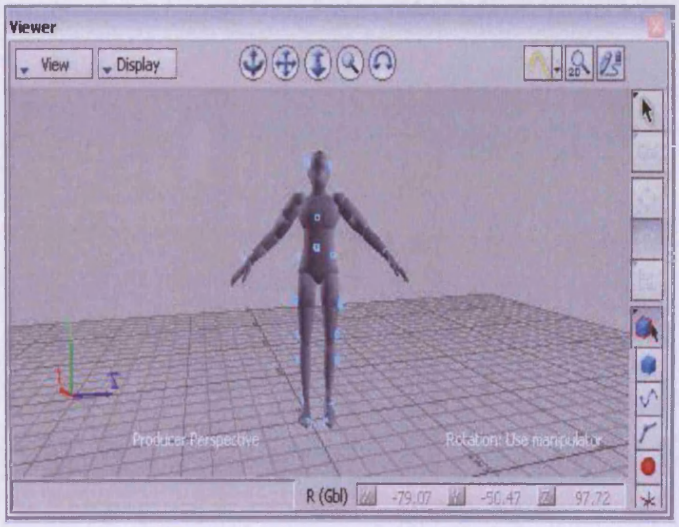


Figure 6.5. Match the generated motion which is represented by certain points (denote as blue points) on the body onto a 3D actor.

2. Rendering the animation sequence as *.avi* file using **Render** option under **File** in MotionBuilder.

3. Importing the animation sequence, the real video sequence and the background sequence into Shake using the **FileIn** node from the **Image Tool tab** (Figure 6.6). Further details can be found in [117].

4. Using the relevant tools button, such as **Resize**, **Over**, **Subtract** and **ColorMatch** to create the animation video (Figure 6.7). Further details can be found in [117].

5. Saving the animation video to the disk using the **FileOut** node from the **Image Tool tab**.

Figure 6.6. The animation sequence, the real video sequence and the background sequence are imported to Shake.

During this process, a lighting system[1] must be created to simulate the light in the original real video and use alpha channels in a digital composite program to generate occlusions, when the 3D character is occluded by the real character in the video.

6.5 Experiments with 3D MoCap Data

In this section, three experiments are conducted, each with a different type of motion: two people shaking hands, one person pulling another person, and one person pushing another person.

[1]The process of creating a lighting system is complex, more details about that can be found in [117]

Figure 6.7. The process of producing the animation video.

6.5.1 Generating Algorithm

In each experiment, the training data consisted of seven MoCap data sequences with two individual, with a total of around 1600 pose vectors. Each experiment consisted of the following steps.

1. PCA is performed on the data set representing the motion of two people, keeping 90% of the eigenenergy (seven dimensions remain) and project the data set in the reduced dimensionality eigenspace.

2. A dual-input HMM is then trained on the above data set depicting interactive behaviour of two persons.

3. A trellis data structure is built for generating motion data.

4. A response behaviour for a virtual character is generated using the dual-input HMM and working with the standard Viterbi algo-

rithm or the windowed Viterbi algorithm as described in Section 6.3.

5. The virtual character is inserted into real video sequence.

6.5.2 Animation Video Sequences

Selected frames from the generated video sequences are shown in Figures 6.8, 6.9 and 6.10. A selection of generated video sequences are available in the attached CD as well. The videos in folder **Generated results - standard** are generated using the standard Viterbi algorithm, and the videos in folder **Generated results - windowed** are generated using the windowed Viterbi algorithm.

6.6 Assessing the Accuracy of the Generated Behaviour

In the following experiments the performances of the standard Viterbi algorithm and the windowed Viterbi algorithm are compared when generating interactive behaviours for a virtual character. To do so, the Euclidean distance [118] are calculated between original motion and the generated motion generated by both approaches to assess the accuracy of the generated behaviour.

A dual-input HMM was trained on 3D MoCap data representing three different types of motion of two persons: handshaking, one person pulling another person, and one person pushing another person. Given a sequence of MoCap data extracted from video sequence for person **A**, we generated interactive motion for person **B** using the standard Viterbi algorithm and the windowed Viterbi algorithm (we choose the length the window size to be 10, which was determined empirically).

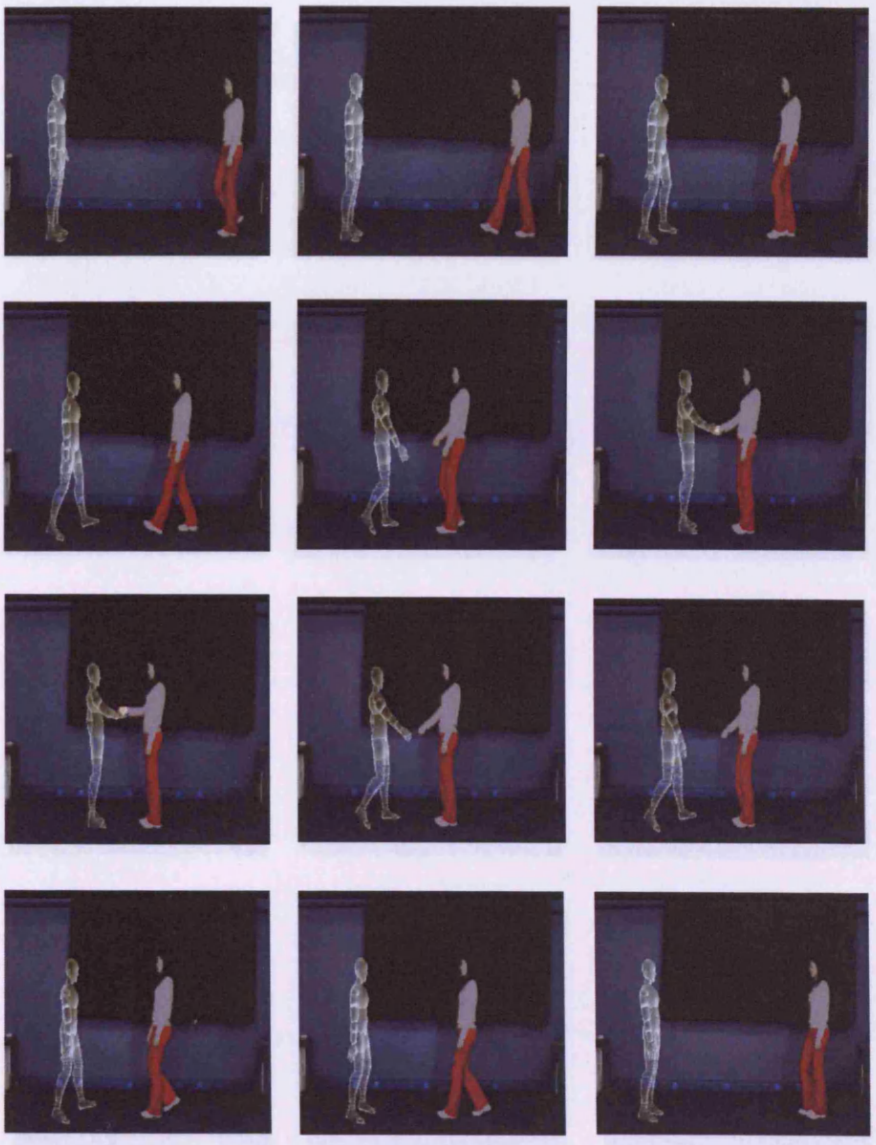


Figure 6.8. Interactions with virtual character (Handshake behaviour).

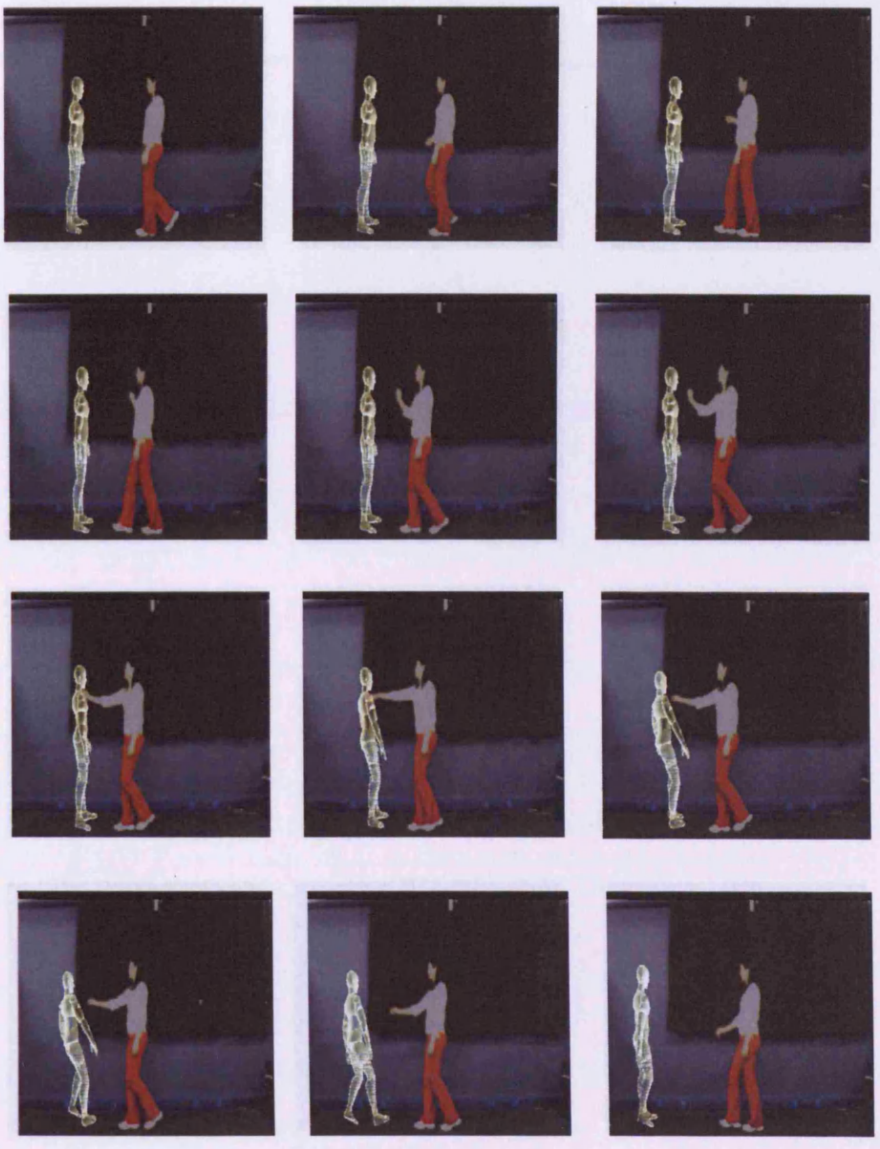


Figure 6.9. Interactions with virtual character (Pushing behaviour).

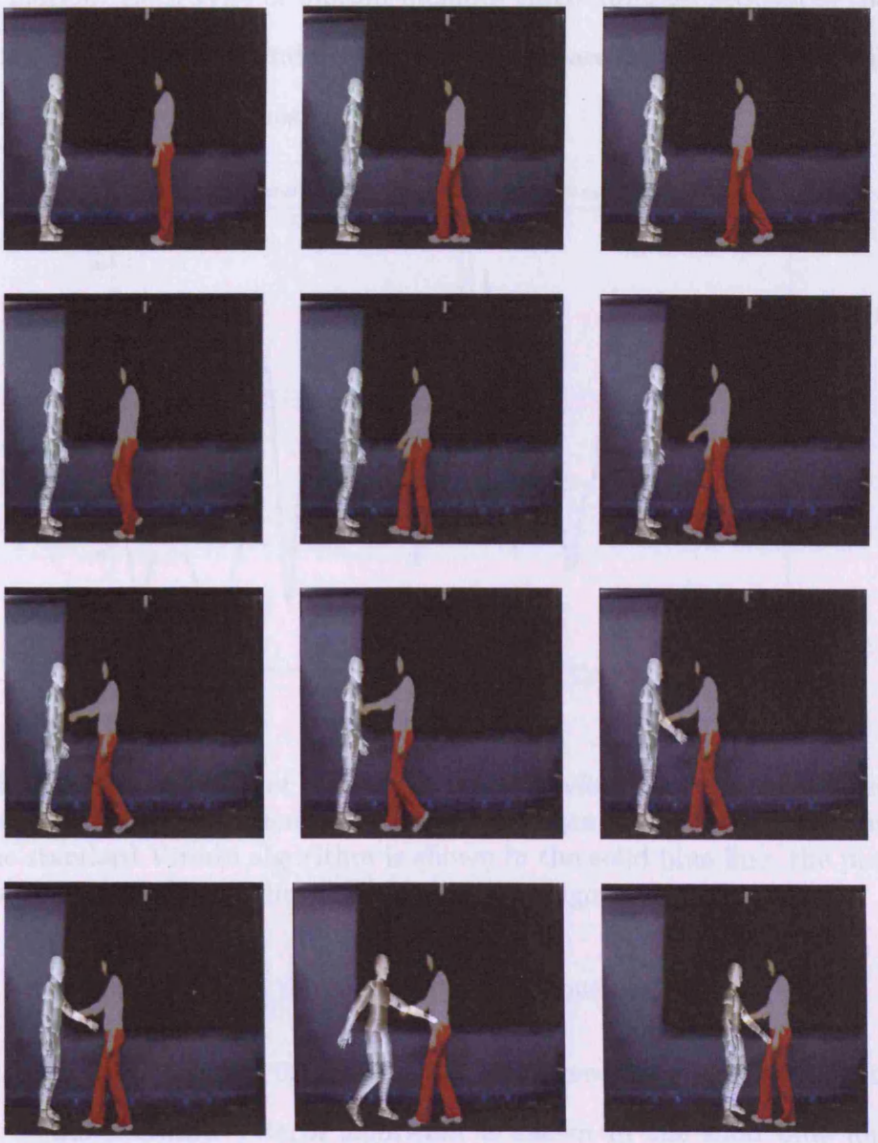


Figure 6.10. Interactions with virtual character (Pulling behaviour).

Figures 6.11 - 6.19 show the Euclidean distance between the original motion and the generated motion when using the standard Viterbi algorithm and the windowed Viterbi algorithm on three different types of motion. Each type of motion includes three different sequences (Sequence 1, Sequence 2 and Sequence 3), which are captured by the same person on different times.

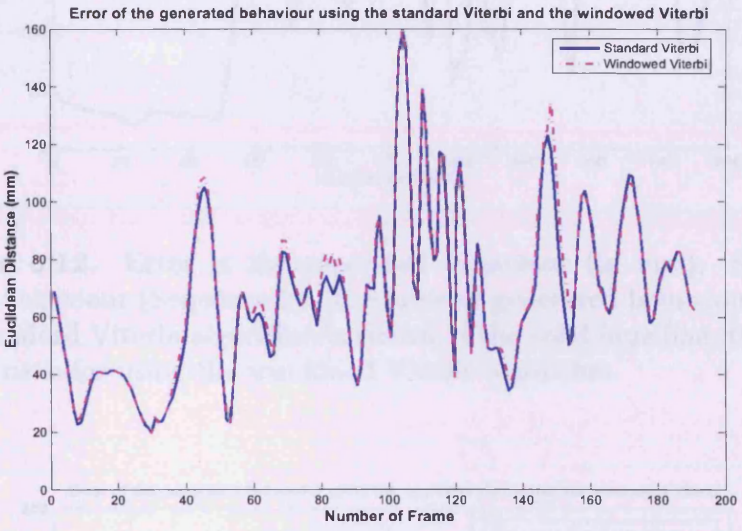


Figure 6.11. Error of the generated behaviour (in mm). Shaking hands behaviour (Sequence 1). The error of generated behaviour using the standard Viterbi algorithm is shown in the solid blue line, the pink dash line is for using the windowed Viterbi algorithm.

From these figures, the following conclusions are made:

1. In Figures 6.11 - 6.19, the error of the generated behaviour using the standard Viterbi algorithm is shown in the solid blue line. The pink dash line shows the error of generating behaviour using the windowed Viterbi algorithm. It is easy to see the errors of generating behaviours using both algorithms are similar. It means the generated motion using the windowed Viterbi algorithm is as

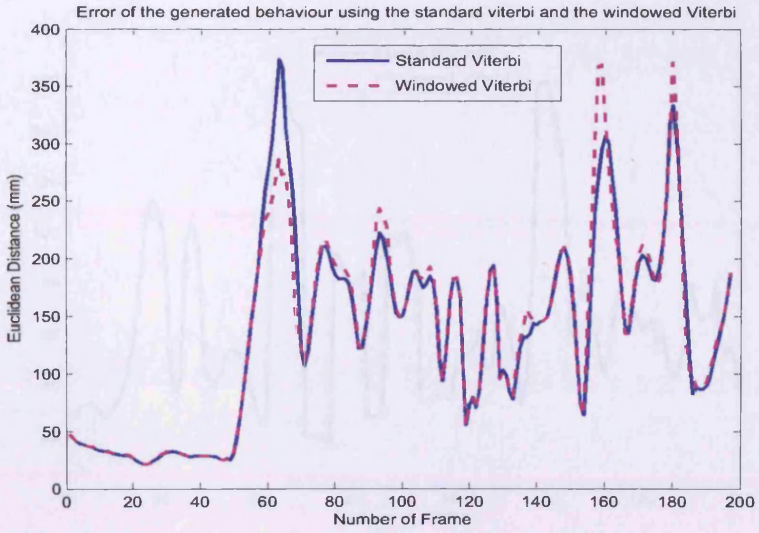


Figure 6.12. Error of the generated behaviour (in mm). Shaking hands behaviour (Sequence 2). The error of generated behaviour using the standard Viterbi algorithm is shown in the solid blue line, the pink dash line is for using the windowed Viterbi algorithm.

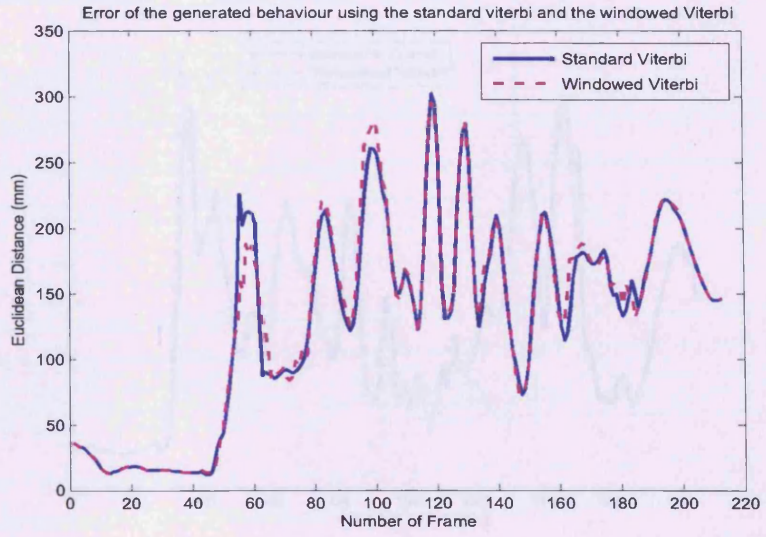


Figure 6.13. Error of the generated behaviour (in mm). Shaking hands behaviour (Sequence 3). The error of generated behaviour using the standard Viterbi algorithm is shown in the solid blue line, the pink dash line is for using the windowed Viterbi algorithm.

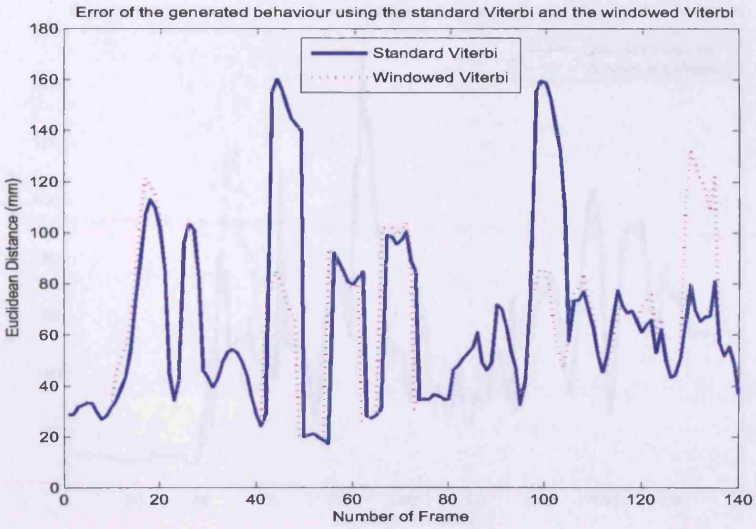


Figure 6.14. Error of the generated behaviour (in mm). Pulling behaviour (Sequence 1). The error of generated behaviour using the standard Viterbi algorithm is shown solid in blue line, the pink dash line is for using the windowed Viterbi algorithm.

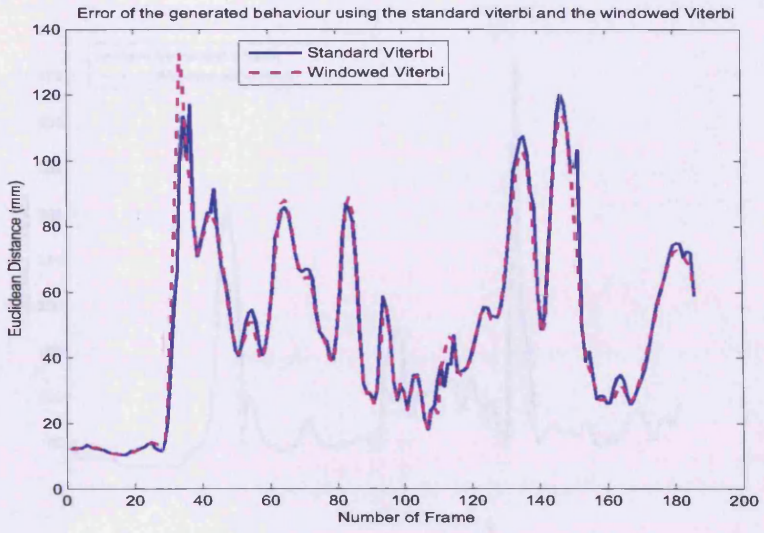


Figure 6.15. Error of the generated behaviour (in mm). Pulling behaviour (Sequence 2). The error of generated behaviour using the standard Viterbi algorithm is shown solid in blue line, the pink dash line is for using the windowed Viterbi algorithm.

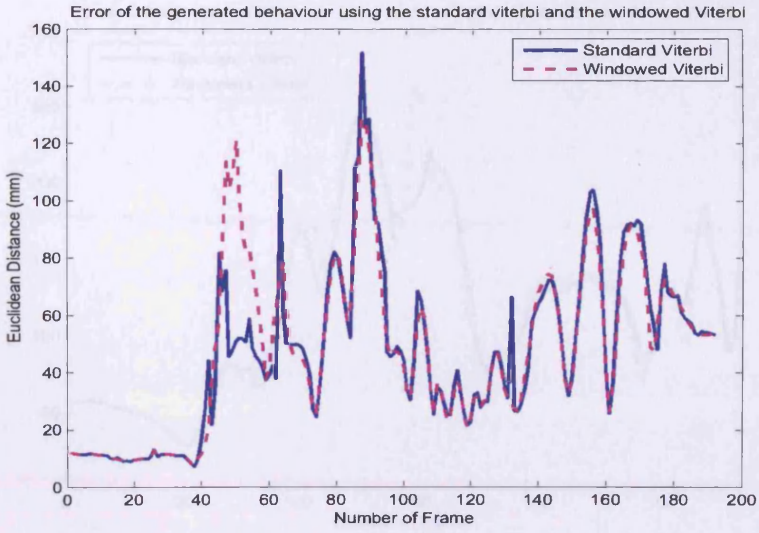


Figure 6.16. Error of the generated behaviour (in mm). Pulling behaviour (Sequence 3). The error of generated behaviour using the standard Viterbi algorithm is shown solid in blue line, the pink dash line is for using the windowed Viterbi algorithm.

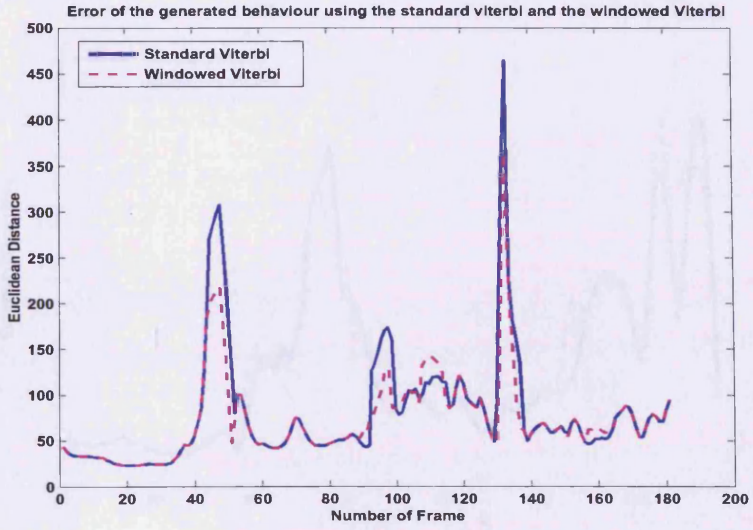


Figure 6.17. Error of the generated behaviour (in mm). Pushing behaviour (Sequence 1). The error of generated behaviour using the standard Viterbi algorithm is shown in solid blue line, the pink dash line is for using the windowed Viterbi algorithm.

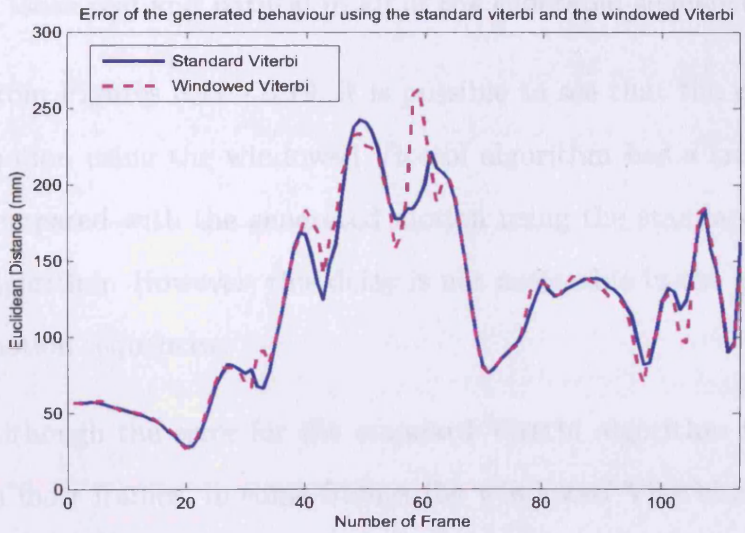


Figure 6.18. Error of the generated behaviour (in mm). Pushing behaviour (Sequence 2). The error of generated behaviour using the standard Viterbi algorithm is shown in solid blue line, the pink dash line is for using the windowed Viterbi algorithm.

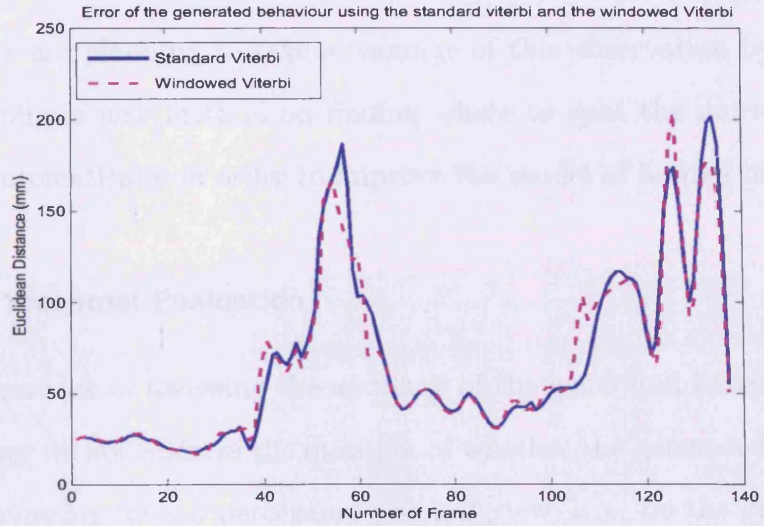


Figure 6.19. Error of the generated behaviour (in mm). Pushing behaviour (Sequence 3). The error of generated behaviour using the standard Viterbi algorithm is shown in solid blue line, the pink dash line is for using the windowed Viterbi algorithm.

good as the motion generated using the standard Viterbi. Hence, it looks real and natural in all of the generated sequences.

2. From Figures 6.11 - 6.19, it is possible to see that the generated motion using the windowed Viterbi algorithm has a small delay compared with the generated motion using the standard Viterbi algorithm. However, this delay is not noticeable in the generated motion sequences.

3. Although the error for the standard Viterbi algorithm is similar in most frames, in some frames the windowed Viterbi algorithm produces smaller error. After inspecting the generated motion sequences visually, we found that such frames corresponded to sudden changes in the motion. We conclude that the windowed Viterbi algorithm can cope with sudden changes in motion better as it uses less history of motion. In the following chapter, we are planning to take advantage of this observation by developing a new method on finding where to split the motion data automatically in order to improve the model of human motion.

6.7 Perceptual Evaluation

The drawback of assessing the accuracy of the generated behaviours is that they do not address the question of whether the generated motion are convincing form a perceptual point of view, i. e. do the generated motion actually look convincing and realistic? In this section, the goal was to evaluate how convincing the generated motion was and evaluate the perception quality of the generated motions, from the point of view of a panel of independent observers. For these purposes, nine test video

sequences were generated.

- Three test video sequences showed original MoCap data collected from two persons performing handshake, pushing, and pulling actions.
- Three test video sequences showed the generated motion using the standard Viterbi algorithm (Chapter 6.3.1) interacts with the original MoCap data of person **A**.
- In the remaining three sequences, the original MoCap data of person **A** was substituted with motion data generated using the windowed Viterbi algorithm (Chapter 6.3.2).

All videos showed only the motion of certain points on the body, not the whole body. Figure 6.20 shows selected frames from the test video sequences. According to Johansson's [119] experiments in psychology (Johansson has shown that an animation sequence consisting of a few points placed on the joints of the articulated human body is enough to create a perception evaluation.), the video sequences are acceptable for visual evaluation.

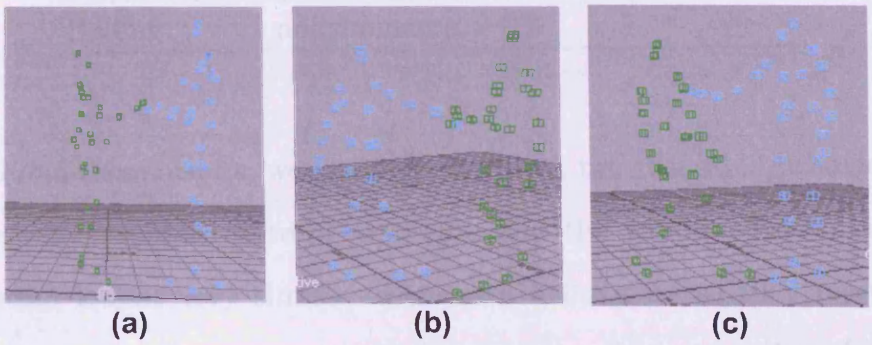


Figure 6.20. Test video sequences showed only the motion of certain points on the body.

Each of the above video sequences was shown to ten independent observers[2]. The observers were told that the videos showed the motion data of two people performing some action, and they were asked to answer two questions:

1. To identify the actions.

2. To comment if they noticed anything strange or unusual about the motion of the people.

All ten subjects were able to identify the actions in all nine video sequences correctly. Table 6.1 and Table 6.2 show the evaluation results whose motion were generated using the standard Viterbi algorithm and the windowed Viterbi algorithm respectively.

Table 6.1. Evaluation Results. The resulting motions are generated using the standard Viterbi algorithm.

Motion	*Comments*
Shaking hands	5/10 - generated motion is floaty.
Pushing	3/10 - original motion is wobbly. 1/10 - generated person did not touch the real person.
Pulling	no comments.

From these results, we conclude that both the generated behaviours using the standard Viterbi algorithm and the windowed Viterbi algorithm looked very similar to the real behaviours as they received approximately the same amount of comments. With exception of, perhaps, generated "shaking hands" motion, the observers did not notice

[2]Using ten independent observers to evaluate the perception quality of the generated motions was accepted in all my conference publications.

Table 6.2. Evaluation Results. The resulting motions are generated using the windowed Viterbi algorithm.

Motion	*Comments*
Shaking hands	2/10 - generated person did not touch the real person at the beginning of shaking hands. 2/10 - generated motion is wobbly.
Pushing	2/10 - generated motion is floaty.
Pulling	no comments.

anything unusual about the generated motion compared against the real motion.

6.8 Summary

In this chapter, two novel approaches were described to generate a variety of complex behaviour responses for a virtual character responding to the tracked person in real video in 3D. The model is trained on motion capture data depicting three different interactive behaviours by two subjects. Such data is always constrained by physical and dynamical factors, thus the dimensionality of the data set needs to be reduced using PCA before proceeding with anything else. The model is a dual-input HMM, with two sets of states, but only one transition matrix. The first set of states represents the poses for person **A**. The second set of states represent the poses for person **B** using the same HMM transition matrix.

Interactive behaviours are then generated for a virtual character on the basis of motion of the tracked person in video. A corresponding sequence of states is estimated in the HMM using the standard Viterbi

algorithm and the windowed Viterbi algorithm. As a result, a sequence of poses for the virtual character is obtained. In all experiments, the generated behaviour appeared naturalistic.

In the next chapter, the further researches are described on finding where to split the motion data automatically, in order to improve the model of human motion for obtaining better tracking results.

Chapter 7

IMPROVING THE MODEL OF HUMAN MOTION

7.1 Introduction

In the previous chapters, the virtual character generating system was developed. Using this system, a 3D moving virtual character reacting to the motion of the person in video was generated and placed back into the original video footage. Consequently, the performances of the standard Viterbi algorithm and the windowed Viterbi algorithm within the virtual character generating system were compared.

The standard Viterbi algorithm requires the full observation sequence before the processing starts, thus making real-time processing impossible. When the windowed Viterbi method is used instead, it does not require the full observation sequence before the processing starts, thus it can be used in a real-time system. Moreover, realistic generated interactive behaviours can still be obtained.

By analysing the generated behaviours in Figures 6.11, - 6.19, we found that the error for the standard Viterbi algorithm is similar in most frames. In some frames the windowed Viterbi algorithm produces smaller error. After inspecting the generated motion sequences visually,

we found that such frames corresponded to sudden changes in the motion. We concluded that the windowed Viterbi algorithm can cope with sudden changes in motion better as it uses less history of the motion.

Blake *et al.* [120] also introduced a new segmentation method based on HMM to deal with problems of the shadows of vehicles. The method performed accurate segmentation of the foreground objects from background objects and shadows. Then the HMMs are employed to deal with three different categories including background, foreground and the shadows of vehicles. However, they approximate the distribution of the background and shadow by Gaussian densities and the distribution of the foreground as a uniform probability density. Thus, the model parameters of each HMM are estimated from a particular learning sequence in the learning process. While in the segmentation process, one series of optimal states is found for each HMM over time.

In this chapter, we take advantage of our observation in the previous research and the idea of Blake's work by developing a *new* method on finding the place to split the motion data automatically in order to improve the model of human motion for obtaining better tracking results. Figure 7.1 presents the general idea of the work contained in this chapter.

To do so, the combined model trained on the particular motion data is introduced. The combined model employed is based on the following premise. When a complex sequence of MoCap data representing motion of two persons contains several different behaviours together (normally it is a high dimensional data set), such as walking, pushing, running and jumping. We would like to split the complex motion sequence to several subsequences. Figure 7.2 shows an example of splitting a com-

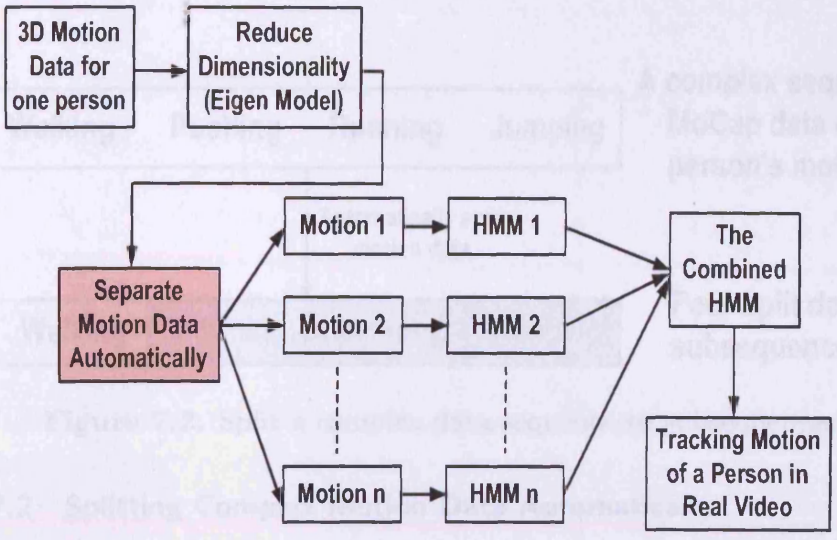


Figure 7.1. The general idea of the work contained in this chapter.

plex sequence data to subsequences. The statistical models have been trained on different subsequences of motion data and then fused models together, we wish the combined model can represent the data distributions more accurately. Thus, we may obtain better tracking results when tracking the motion of a person in real video using this combined model. We can improve on the above assumption by comparing the performance of the tracking results when using the combined model and the normal model (a model trained on 3D MoCap data representing the complex motion at once).

In this chapter, a novel method is presented for finding where to split the motion data automatically in Section 7.2. The combined HMM trained on the split data sequence is then introduced in Section 7.3. The detail of the combined HMM and the whole process of applying on the MoCap data are explained in Section 7.4. In Section 7.5, the tracking results and the performance analysis are presented, followed by the summary in Section 7.6.

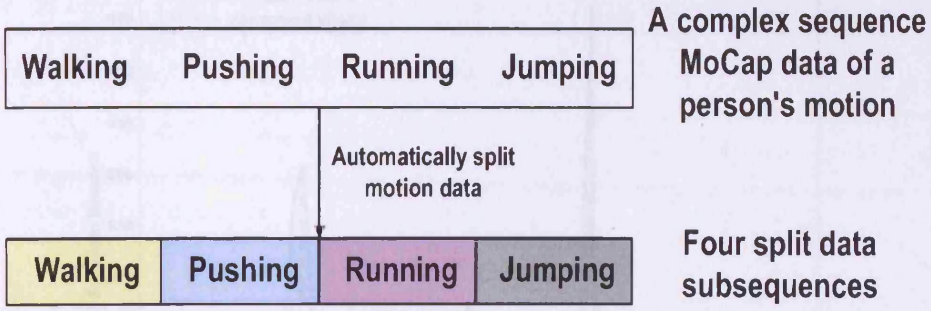


Figure 7.2. Split a complex data sequence to subsequences.

7.2 Splitting Complex Motion Data Automatically

Previously in Chapter 6, human interactive behaviours for a virtual character were generated using the standard Viterbi algorithm and the windowed Viterbi algorithm. The performances of the standard Viterbi algorithm and the windowed Viterbi algorithm were compared. We do so by calculating the Euclidean distance [118] between original motion and the motion generated by both approaches to assess the accuracy of the generated behaviour. Figure 7.3 shows the Euclidean distance between the original motion and the generated motion when using the standard Viterbi algorithm and the windowed Viterbi algorithm on pushing behaviour.

Although the error of the standard Viterbi algorithm is similar in most of the frames, in some frames the windowed Viterbi algorithm produces smaller error, such as at frames 46, 97 and 133 respectively (as black dash lines shown in Figure 7.3). These frames corresponded to sudden changes in the motion. Therefore, the windowed Viterbi algorithm can be used to find sudden changes in the complex motion sequence. In this section, this observation is exploited by developing

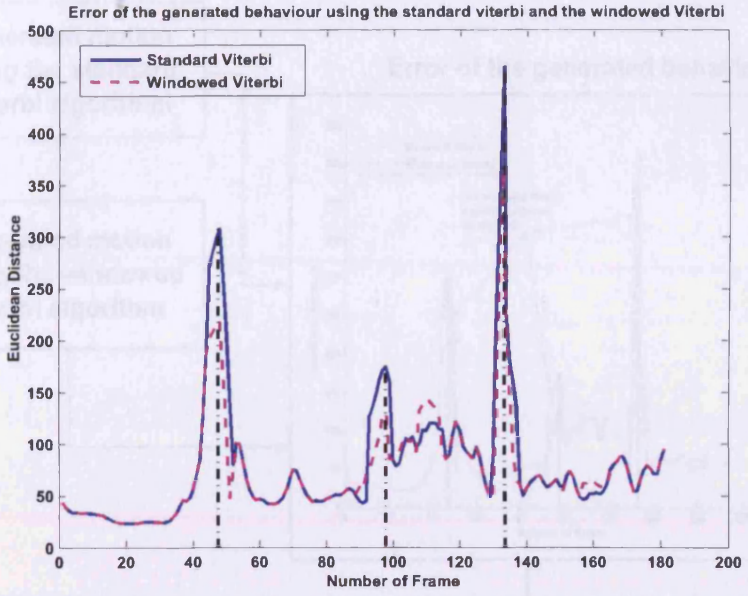


Figure 7.3. Error of the generated behaviour (in mm). Pushing behaviour. The error of generated behaviour using the standard Viterbi algorithm is shown in blue solid line, the pink dash line is for using the windowed Viterbi algorithm. The black dash lines show the place to split the motion.

a new approach for splitting the complex motion data automatically. Figure 7.4 shows an illustration of the splitting process. We split the motion automatically into several motion parts using morphology operators by detecting the peaks in the error curve.

The mathematical morphology methods are introduced to detect the objects as local maxima. They can be used to obtain the essential conformation of an object through the operation of objects and structuring elements. The primary morphological operations are dilation and erosion, and from these two, more complex morphological operations such as opening and closing [118, 121]. In this Chapter, the purpose is to find the peaks in a curve, therefore the Top-Hat operator is used to do that. The Top-Hat transformation [3] is a grey scale mor-

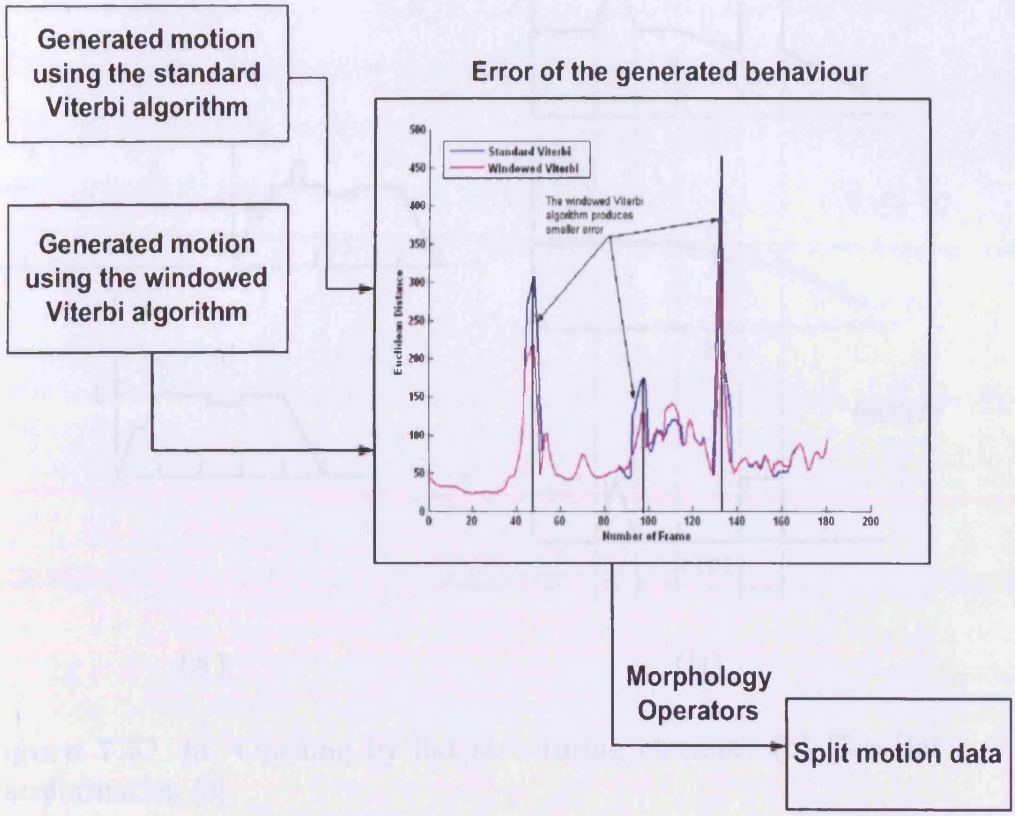


Figure 7.4. An illustration of splitting process.

phologic algorithm. It is used for finding pixel clusters that are light on a surrounding relatively dark background. The operation is illustrated in Figure 7.5. The original signal f is processed with opening by flat structuring element g, and then the peaks are detected as a Top-Hat by subtracting an opened image form the original signal.

Through this process, the position of the peaks together with description of their height are obtained. The motion sequence is then separated at those points. After inspecting the split motion data sequences visually, we find that each of them represents a particular motion. In the next section, a combined HMM is introduced. The idea here is to train an HMM on each part of the split motion data and combine two

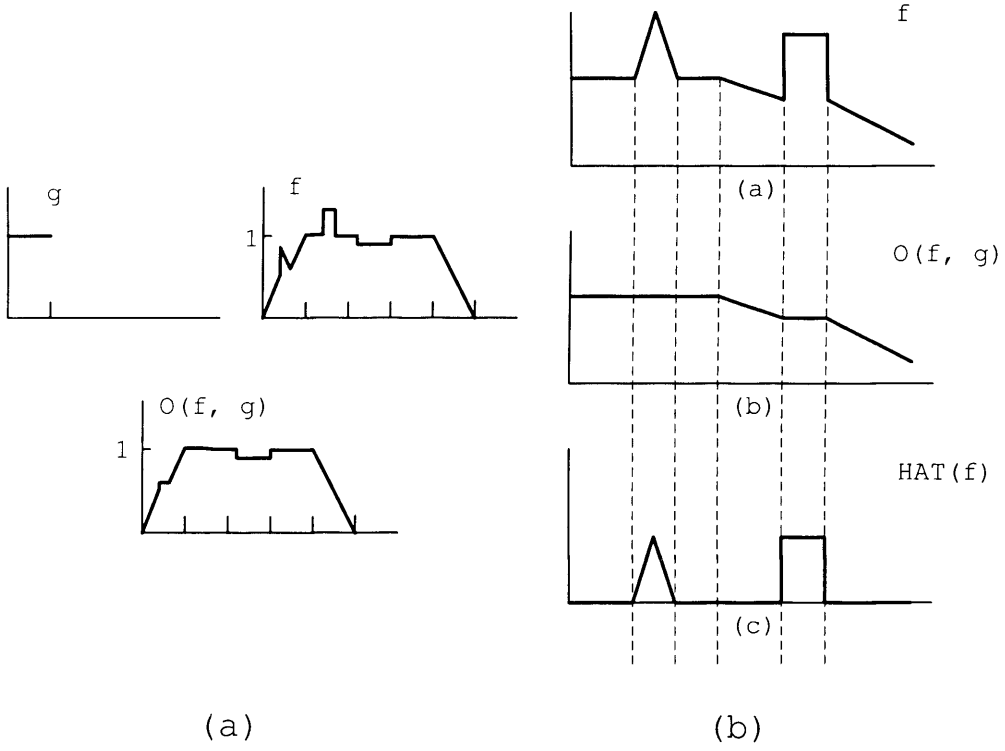


Figure 7.5. (a) Opening by flat structuring element; (b) Top-Hat transformation [3].

or more separate HMMs into one. Then the combined HMM can be used to track the motion of a person in real video.

7.3 The Combined Hidden Markov Model

In the previous section, the complex motion data sequence was split into several subsequences automatically. Here, a combined HMM is introduced in order to obtain better tracking results in real video. The idea of the combined HMM is to combine two or more separate HMMs into one [122]. Each separate HMM is trained on the data of a particular motion, such as walking, shaking hands or pushing. Next the parameters of each HMM are combined together. Finally, the transition matrix of the combined HMM is updated using the Baum-Welch

algorithm [13]. The example structure diagram for the combined HMM is shown in Figure 7.6. Sub-model *1* and sub-model *2* are learnt on different parts of the split data with different numbers of Gaussians. The combined HMM consists of all information from both sub-models, and connected with a transition probability a from sub-model *1* to sub-model *2*.

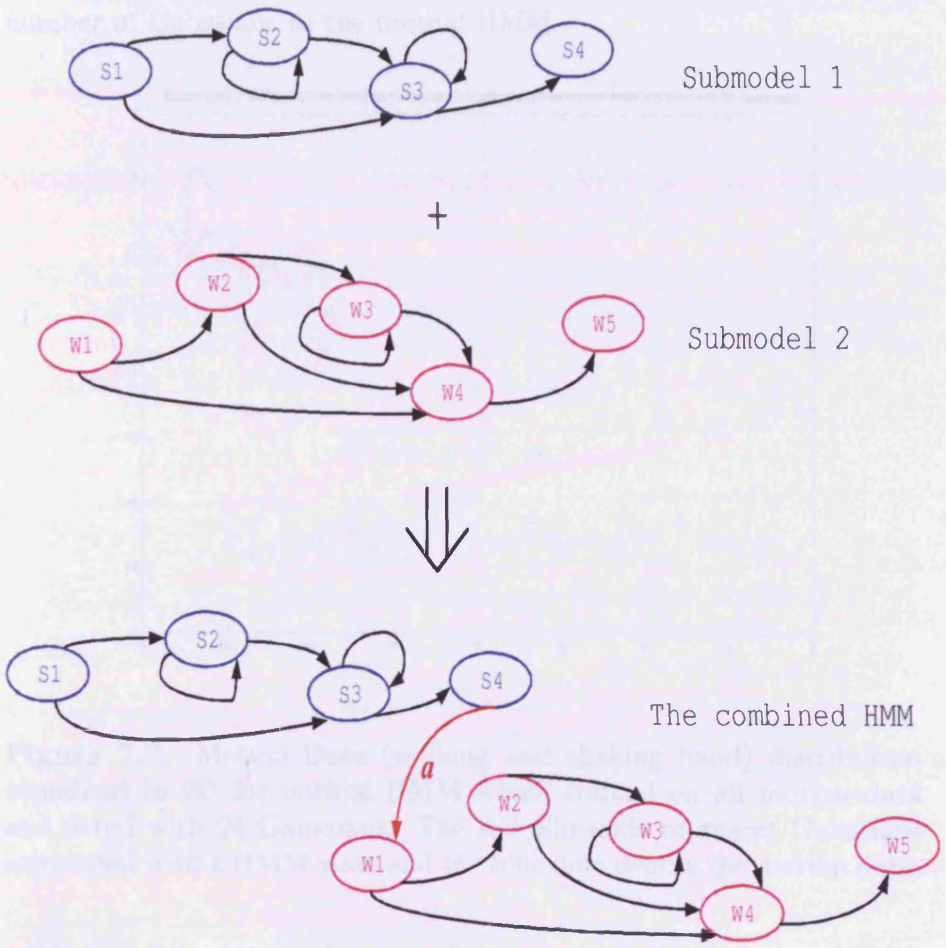


Figure 7.6. The example structure diagram of two simple HMMs for the combined HMM are shown. a is the transition probability from sub-model 1 to sub-model 2. It can be obtained using the Baum-Welch algorithm.

Figure 7.7 shows the motion data distribution visualised in 2D for the normal HMM which was trained on two sequences complex motion data at once and fitted with 24 Gaussians, for example, one person walking and shaking hands. Figure 7.8 shows the motion data distribution visualised in 2D for the combined HMM on the same data set with the same number of Gaussians. It is clear to see the combined HMM can cover the data distribution very well when choosing the same number of Gaussians as the normal HMM.

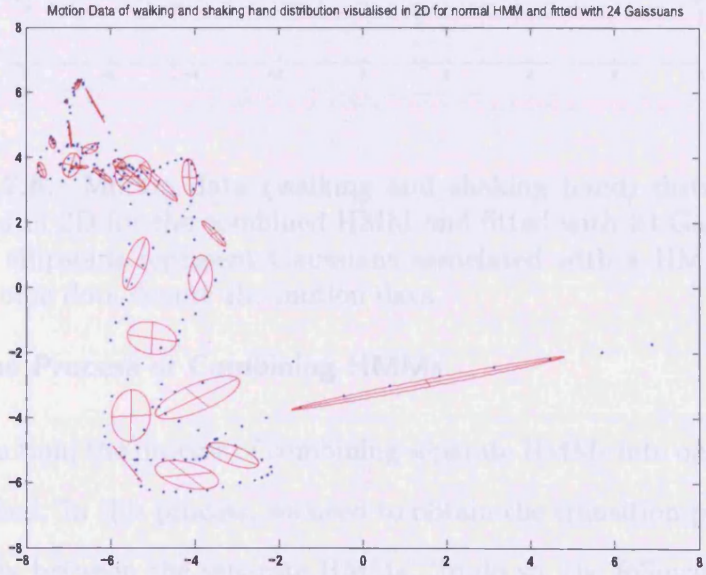


Figure 7.7. Motion Data (walking and shaking hand) distribution visualised in 2D for normal HMM which trained on all motion data and fitted with 24 Gaussians. The red ellipsoids represent Gaussians associated with a HMM state and the blue dots denote the motion data.

In the following section, the detail of combining separate HMMs into one model is presented.

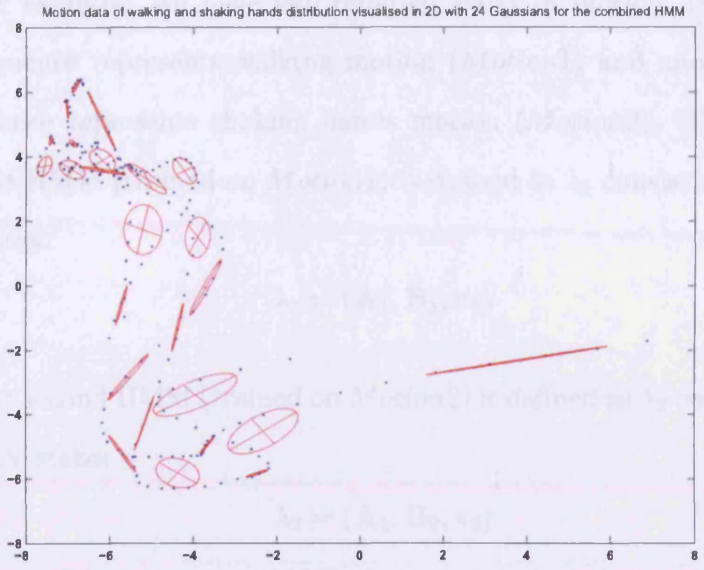


Figure 7.8. Motion data (walking and shaking hand) distribution visualised in 2D for the combined HMM and fitted with 24 Gaussians. The red ellipsoids represent Gaussians associated with a HMM state and the blue dots denote the motion data.

7.4 The Process of Combining HMMs

In this section, the process of combining separate HMMs into one model is described. In this process, we need to obtain the transition probability matrix between the separate HMMs. To do so, the following steps are performed:

1. Perform PCA on the 3D MoCap data representing the motion of one person.

2. Automatically split the complex motion sequence to several different motion parts using morphology operators by detecting the peaks in the error curve (such as one part is walking, another part is shaking hands). A detailed is described in Section 7.2.

3. Train an HMM on each particular motion data (Chapter 4).

For example, we have two different parts of motion data, one sequence represents walking motion ($Motion1$) and another sequence represents shaking hands motion ($Motion2$). Then the first HMM (trained on $Motion1$) is defined as λ_1 consisting of M states

$$\lambda_1 = (\mathbf{A}_1, \mathbf{B}_1, \pi_1) \tag{7.4.1}$$

The second HMM (trained on $Motion2$) is defined as λ_2 consisting of N states

$$\lambda_2 = (\mathbf{A}_2, \mathbf{B}_2, \pi_2) \tag{7.4.2}$$

Figure 7.9 shows low-dimensional walking data ($Motion1$) distribution visualised in 2D and fitted with 10 Gaussians.

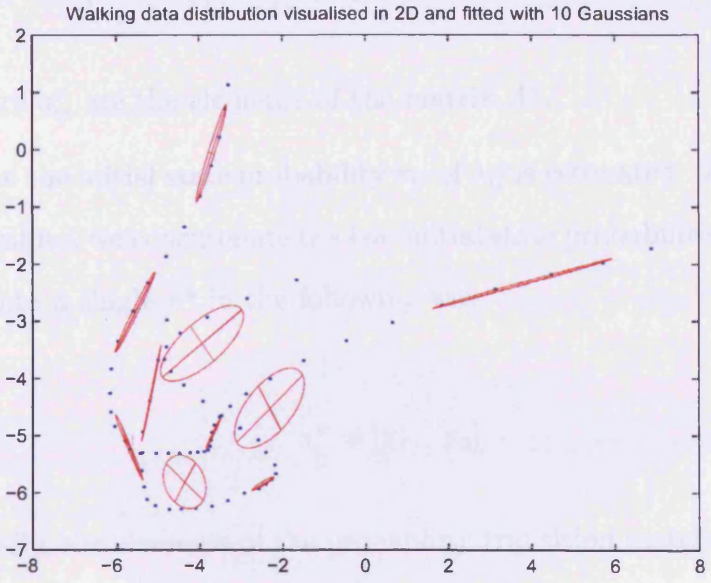


Figure 7.9. Motion data (walking) distribution visualised in 2D and fitted with 10 Gaussians. The red ellipses represent Gaussians and the blue dots denote the motion data.

4. Combine the parameters of each HMM which are obtained in

Step 3 and update the transition matrix using the Baum-Welch algorithm [13]. In this step, we want to combine the parameters of each HMM together to update the transition matrix of the combined HMM $\lambda_C = (\mathbf{A}_C, \mathbf{B}_C, \pi_C)$.

To update λ_C we firstly need to estimate the new transition matrix. The first stage of estimating the probability transition matrix $\mathbf{A}_C$ of the combined HMM λ_C is to combine the two original transition matrices $\mathbf{A}_1$ of size $M \times M$ and $\mathbf{A}_2$ of size $N \times N$ into a single matrix $\mathbf{A}^u$ of size $n \times n$, where $n = M + N$.

$$A^u = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \qquad \sum_{j=1}^{n} a_{ij}^u = 1 \qquad 1 \leq i \leq n \qquad (7.4.3)$$

where a_{ij}^u are the elements of the matrix A^u.

Then the initial state probability π_C of λ_C is estimated. To obtain its values, we concatenate the two initial state probabilities π_1 and π_2 into a single π^u in the following way:

$$\pi^u = [\pi_1 \quad \pi_2] \qquad (7.4.4)$$

Finally, the elements of the probability transition matrix $\mathbf{A}_C$ and the initial state probability π_C of λ_C are updated using the Baum-Welch algorithm [13].

Figure 7.10 shows the transition matrix of the normal HMM trained on all walking and shaking hand motion at once. Figure 7.11 shows the transition matrix of the combined HMM trained

on the split data representing the same behaviours.

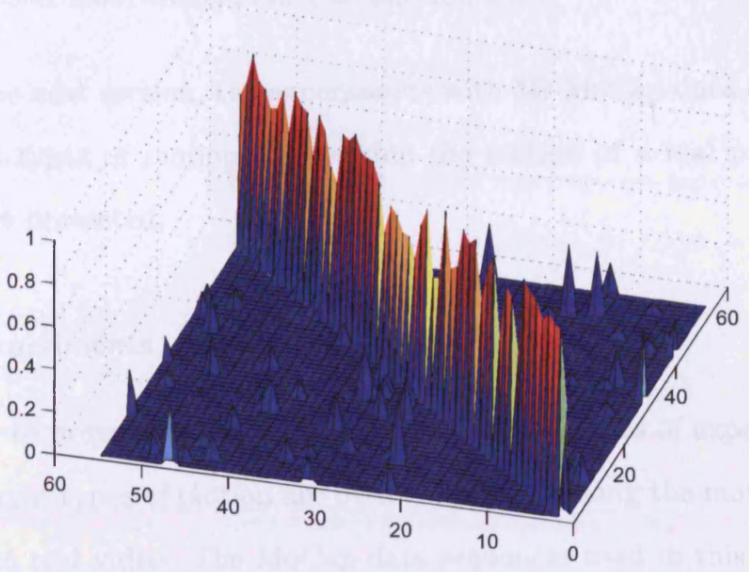


Figure 7.10. Transition matrix for the normal HMM (55 states).

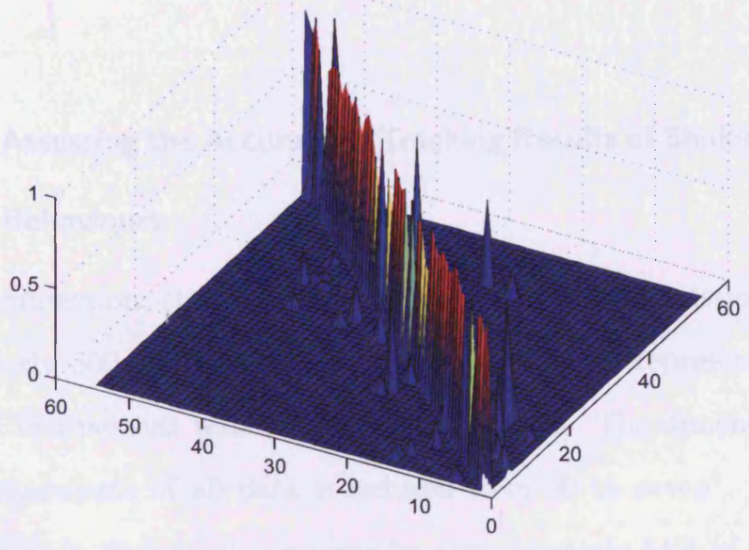


Figure 7.11. Transition matrix for the combined HMM (55 states).

It can be observed that the transition matrix in Figure 7.11 is

strongly diagonal, as there is only few more non-zero values on the off-diagonal. It means there is few closest transition of the cluster from current state to another state.

In the next section, the experiments with 3D MoCap data on three different types of motion for tracking the motion of a real person in video are presented.

7.5 Experiments with 3D MoCap data

In order to prove the hypothesis in Section 7.1, a series of experiments for different types of motion are performed for tracking the motion of a person in real video. The MoCap data sequences used in this chapter are different from the MoCap data used in the previous experiments. The new MoCap data are captured by the different person, but they are performed the same types of motion (shaking hands, pushing and puling).

7.5.1 Assessing the Accuracy of Tracking Results of Shaking Hands Behaviours

In this subsection, the model is trained on the data consisting of approximately 800 frames and 90-dimensional vectors representing the poses of two persons walking and shaking hands. The dimensionality of the eigenspace of all data is reduced from 90 to seven[1]. The remaining seven dimensions accounts for approximately 84% of the total eigenenergy.

[1]Since in the previous tracking experiments the dimensionality of the data set is reduced to seven. In order to compare the tracking results, here the dimensionality of the data set is reduced to seven again.

Five separate HMMs are trained in the reduced dimensionality eigenspace, each on the vector sequences representing the different types of motion (for example, walking, then shaking hands, then walking back to the original place). To speed up the experiments, we choose the number of clusters representing each kind of motion in the different five HMMs to be equal to 15, 11, 19, 5 and 5 respectively, rather than estimate them automatically. Then five separate HMMs are combined together into a combined HMM. Finally, a 3D person in real video is tracked using this combined HMM and the APF as described in Chapter 5.

Figures 7.12 (a) - (e) show the error of the tracked behaviours for shaking hands motion when using each separate HMM respectively. Figures 7.13 and 7.15 show the Euclidean distance between the original video motion and the tracked behaviours using the normal HMM (a model trained on 3D MoCap data representing the complex motion) and the combined HMM on different video sequences. It is easy to see that the error of the motion tracking using the combined HMM is smaller than the error using the normal HMM in all frames. This means the tracked motion using the combined HMM is better than the motion tracked using the normal HMM.

7.5.2 Assessing the Accuracy of Tracking Results of Pushing Behaviours

In this subsection, the model is trained on the data consisting of approximately 550 frames and 90-dimensional vectors representing the poses of a person pushing another person. The dimensionality of the eigenspace of all data is reduced from 90 to seven[2]. The remaining seven

[2]Since in the previous tracking experiments the dimensionality of the data set is reduced to seven. In order to compare the tracking results, here the dimensionality

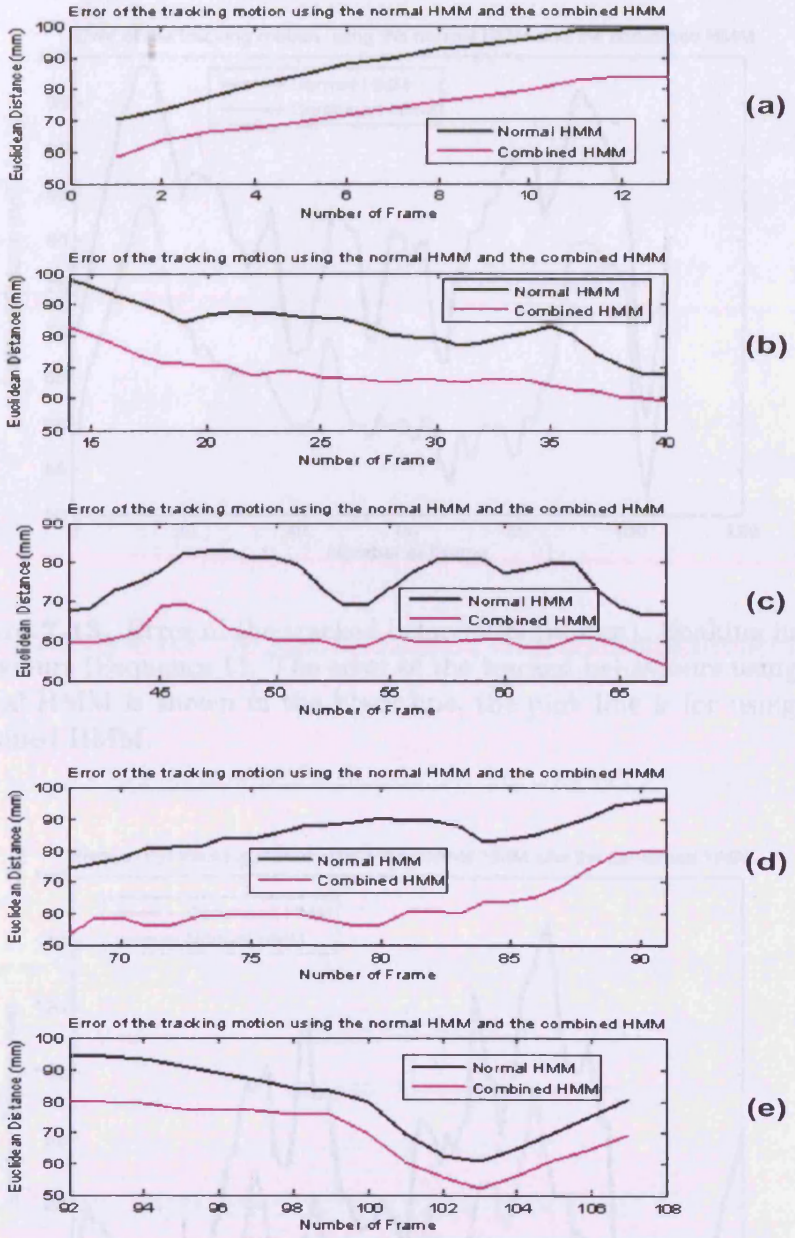


Figure 7.12. Error of the tracked behaviours (in mm). Shaking hands behaviours (Sequence 1). (a) - (e) show the error of the tracked behaviours when using each separate HMM respectively.

dimensions accounts for approximately 83% of the total eigenenergy.

Four separate HMMs are trained in the reduced dimensionality

of the data set is reduced to seven again.

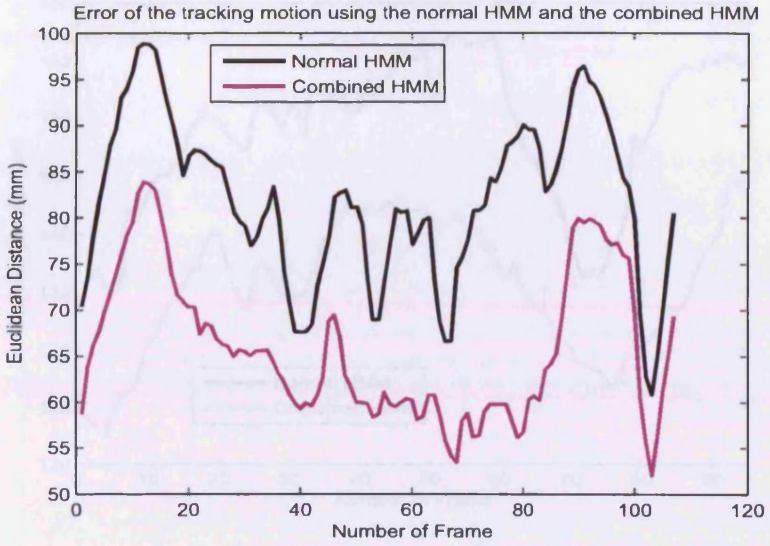


Figure 7.13. Error of the tracked behaviours (in mm). Shaking hands behaviours (Sequence 1). The error of the tracked behaviours using the normal HMM is shown in the black line, the pink line is for using the combined HMM.

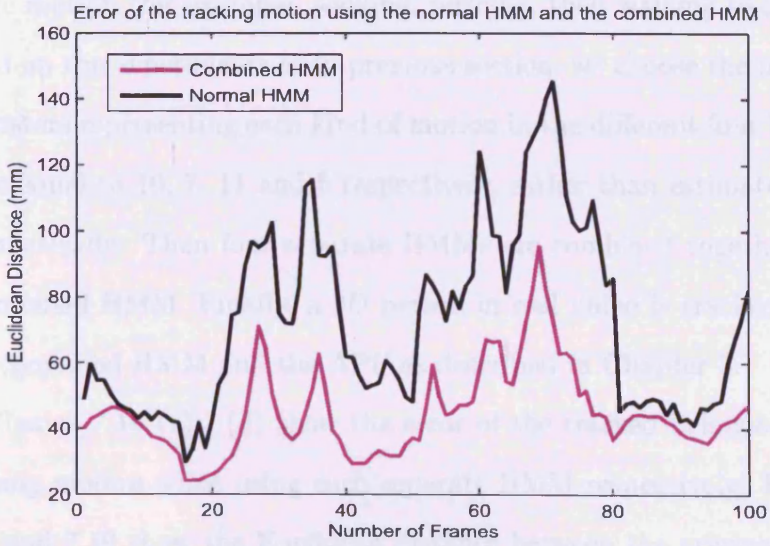


Figure 7.14. Error of the tracked behaviours (in mm). Shaking hands behaviours (Sequence 2). The error of the tracked behaviours using the normal HMM is shown in the black line, the pink line is for using the combined HMM.

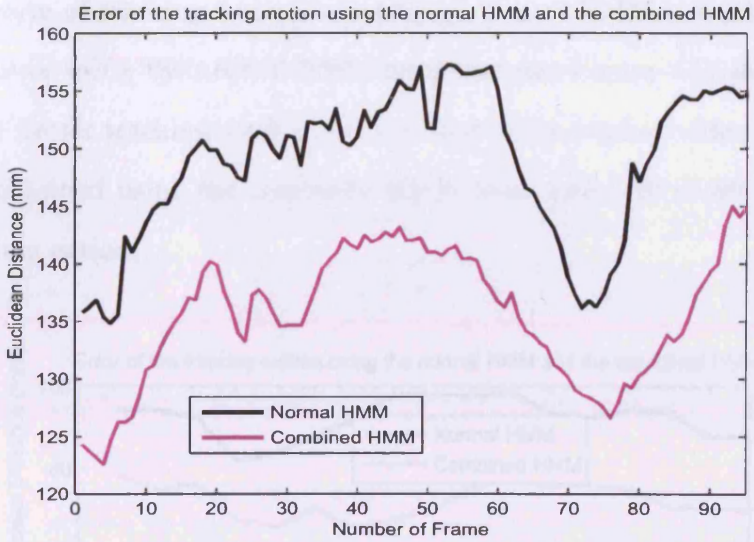


Figure 7.15. Error of the tracked behaviours (in mm). Shaking hands behaviours (Sequence 3). The error of the tracked behaviours using the normal HMM is shown in the black line, the pink line is for using the combined HMM.

eigenspace, each on the vector sequences representing the different type of the motion (for example, walking, pushing, then walking back). To speed up the experiments as in previous section, we choose the number of clusters representing each kind of motion in the different four HMMs to be equal to 10, 7, 11 and 5 respectively, rather than estimate them automatically. Then four separate HMMs are combined together into a combined HMM. Finally, a 3D person in real video is tracked using this combined HMM and the APF as described in Chapter 5.

Figures 7.16 (a) - (d) show the error of the tracked behaviours for pushing motion when using each separate HMM respectively. Figures 7.17 and 7.19 show the Euclidean distance between the original video motion and the tracked behaviours using the normal HMM (a model trained on 3D MoCap data representing the complex motion) and the combined HMM on different video sequences. It is easy to see that

the error of tracking motion using the combined HMM is smaller than the error using the normal HMM in all frames. Figure 7.17 shows the much better tracking results (more similar to the original video motion) are obtained using the combined HMM from frame 60 to 80 (part of pushing action).

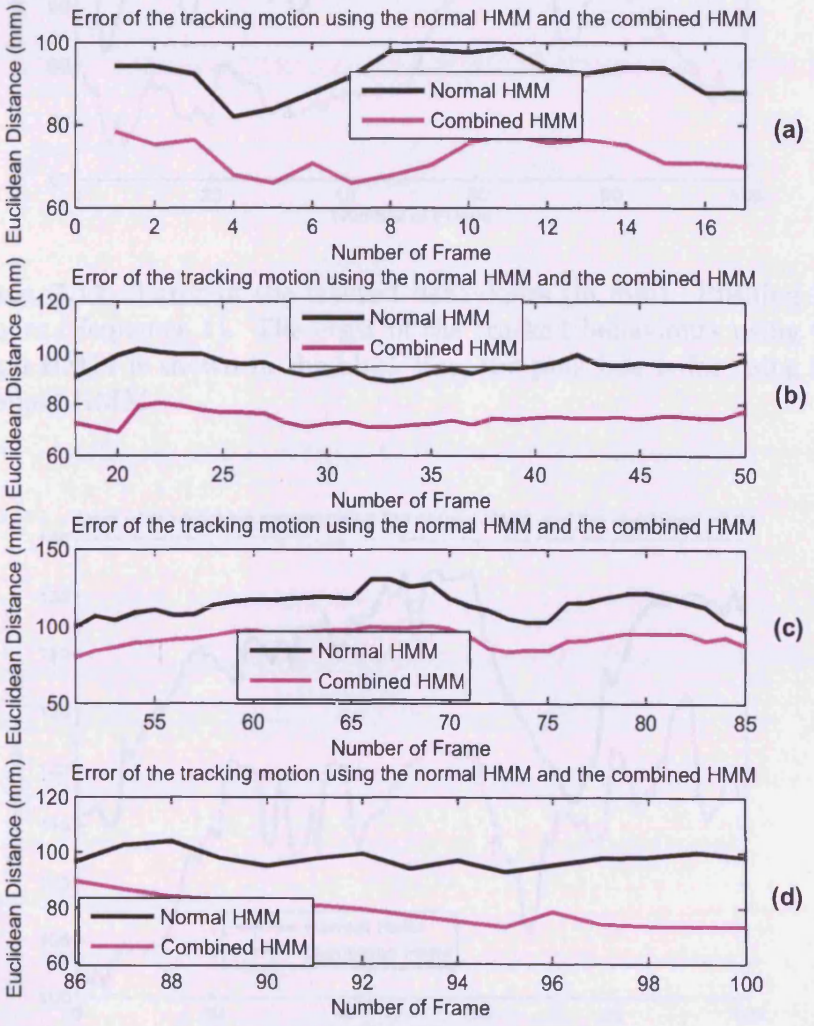


Figure 7.16. Error of the tracked behaviours (in mm). Pushing behaviours (Sequence 1). (a) - (d) show the error when using each separate HMM respectively.

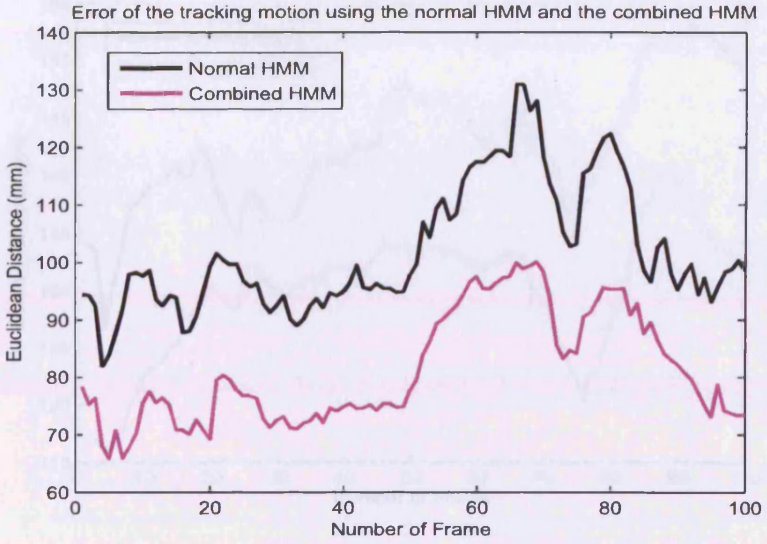


Figure 7.17. Error of the tracked behaviours (in mm). Pushing behaviours (Sequence 1). The error of the tracked behaviours using the normal HMM is shown in the black line, the pink line is for using the combined HMM.

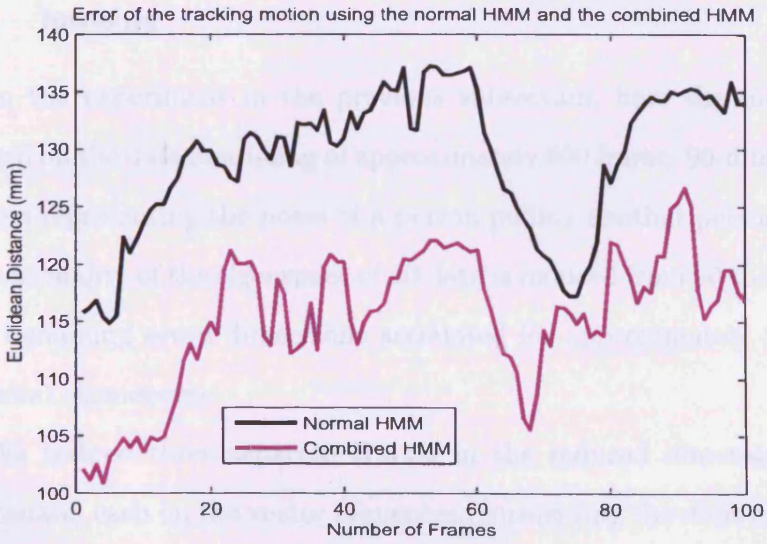


Figure 7.18. Error of the tracked behaviours (in mm). Pushing behaviours (Sequence 2). The error of the tracked behaviours using the normal HMM is shown in the black line, the pink line is for using the combined HMM.

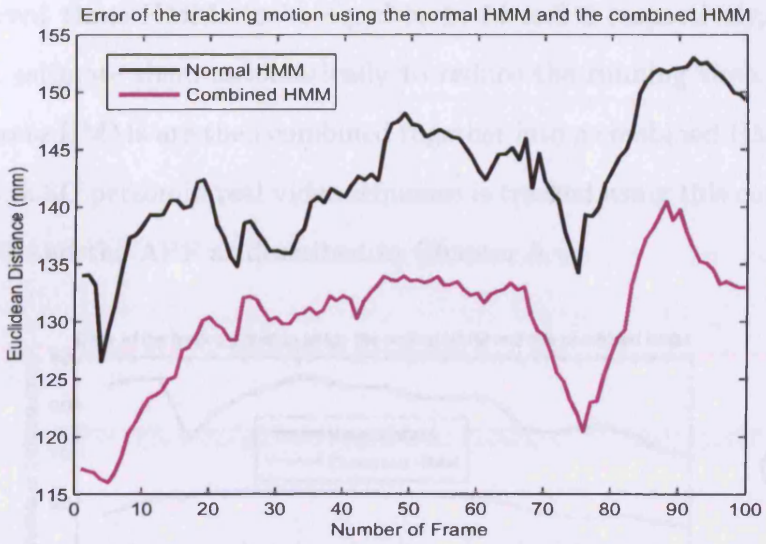


Figure 7.19. Error of the tracked behaviours (in mm). Pushing behaviours (Sequence 3). The error of the tracked behaviours using the normal HMM is shown in the black line, the pink line is for using the combined HMM.

7.5.3 Assessing the Accuracy of Tracking Results of Pulling Behaviours

As in the experiment in the previous subsection, here the model is trained on the data consisting of approximately 500 frame, 90-dimensional vectors representing the poses of a person pulling another person. The dimensionality of the eigenspace of all data is reduced from 90 to seven[3]. The remaining seven dimensions accounted for approximately 81% of the total eigenenergy.

We trained three separate HMMs in the reduced dimensionality eigenspace, each on the vector sequences representing the different type of the motion (for example, walking, pulling, then walking back). We choose the number of clusters representing each kind of motion in the

[3]Since in the previous tracking experiments the dimensionality of the data set is reduced to seven. In order to compare the tracking results, here the dimensionality of the data set is reduced to seven again.

different three HMMs to be equal to 8, 14 and 6 respectively, rather than estimate them automatically to reduce the running time. Three separate HMMs are then combined together into a combined HMM. Finally, a 3D person in real video sequence is tracked using this combined HMM and the APF as described in Chapter 5.

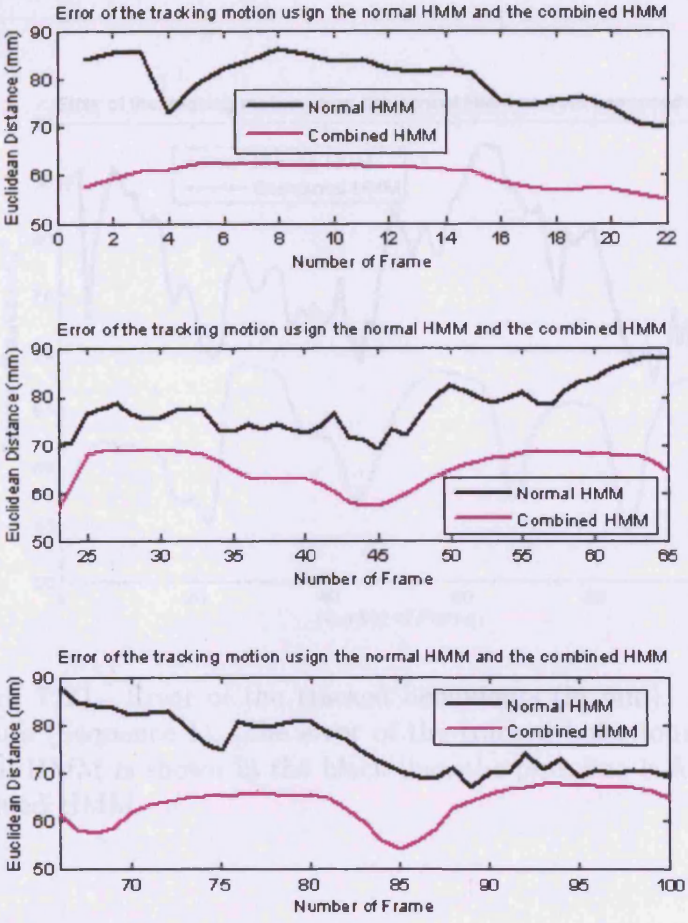


Figure 7.20. Error of the tracked behaviours (in mm). Pulling behaviours (Sequence 1). (a) - (c) show the error when using each separate HMM respectively.

Figures 7.20 (a) - (c) show the error of the tracked behaviours for pulling motion when using each separate HMM respectively. Fig-

ures 7.21 and 7.23 show the Euclidean distance between original video motion and the tracked behaviours using the normal HMM (a model trained on 3D MoCap data representing the complex motion) and the combined HMM on different video sequences. As in the previous section, it can be observed that the tracked motion using the combined HMM is more accurate than the motion tracked using the normal HMM.

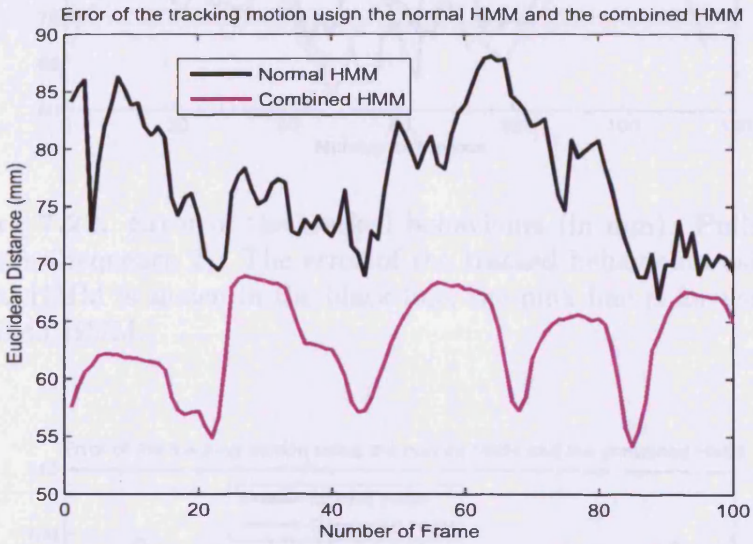


Figure 7.21. Error of the tracked behaviours (in mm). Pulling behaviours (Sequence 1). The error of the tracked behaviours using the normal HMM is shown in the black line, the pink line is for using the combined HMM.

7.6 Summary

In this chapter, a *new* approach is developed for finding where to split the complex motion data automatically in order to improve the model of human motion for obtaining better tracking results. To do so, the complex motion sequence is first divided into several subsequences. The combined HMM is then introduced, which learns a model of human

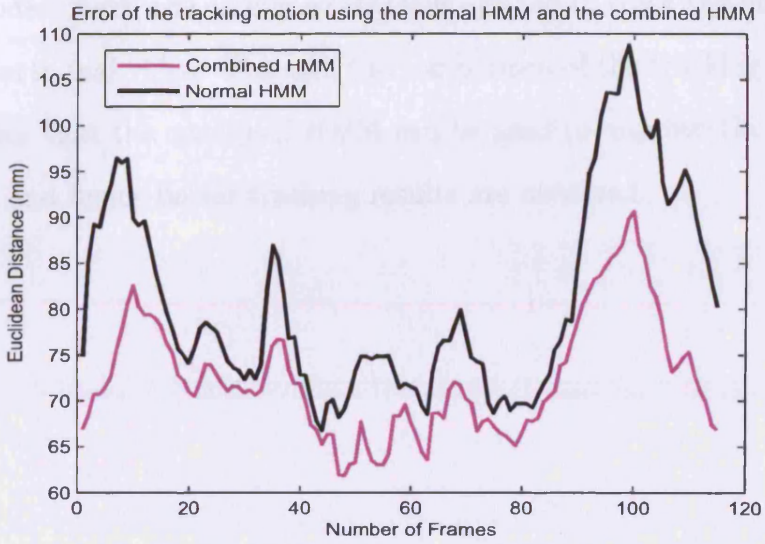


Figure 7.22. Error of the tracked behaviours (in mm). Pulling behaviours (Sequence 2). The error of the tracked behaviours using the normal HMM is shown in the black line, the pink line is for using the combined HMM.

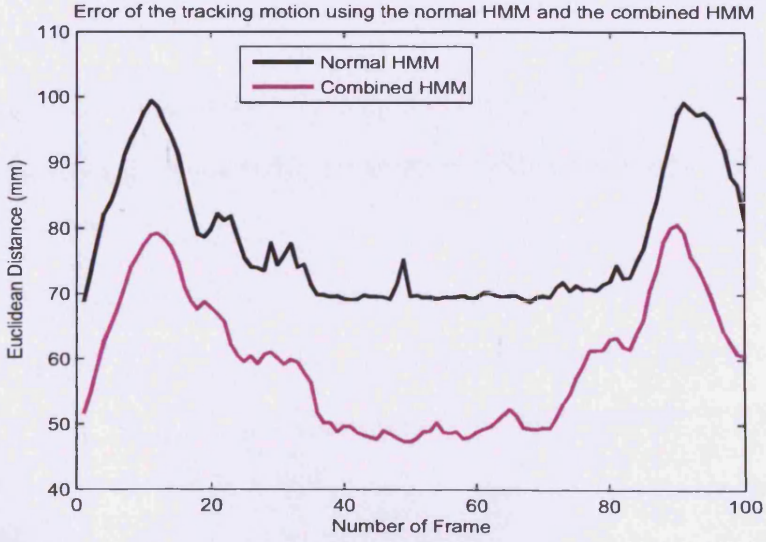


Figure 7.23. Error of the tracked behaviours (in mm). Pulling behaviours (Sequence 3). The error of the tracked behaviours using the normal HMM is shown in the black line, the pink line is for using the combined HMM.

motion on the different parts of the split data and combines them into one model. Next, the combined HMM is applied to track the motion of a person in real video. Through the comparison of the tracking results, it is clear that the combined HMM can be used to improve the normal HMM, and hence better tracking results are obtained.

Chapter 8

CONCLUSION AND FUTURE RESEARCH

The key points of the work presented in this thesis are reviewed, followed by a summary of the main contributions and the future research.

8.1 Conclusion

The work in this thesis was motivated by the desire to develop methods for creating a 3D virtual character capable of responding to actions obtained from observing a real person in video in a realistic and sensible manner. Virtual characters are becoming more and more popular and used in many applications such as character animation, computer games and virtual environments.

In this research, a "virtual character generating system" was developed, and used for generating interactive behaviours for a virtual character responding to a real person in real video. The system builds an accurate model of human motion, tracks and analyses the behaviour of a real person in a video input, and produces a fully articulated 3D character interacting with the real person in the video input. Next, a motion capture system was described, the PhaseSpace Motion Digitizer

System which was set up and used for capturing 3D MoCap data. Afterwards, an accurate model of human motion represented by an HMM was learnt, where each state in the HMM is represented by a single Gaussian. This framework can model the data distribution well and can be used for predicting the poses in the video frame.

A method for tracking a 3D articulated human person in real video sequences was then presented. The tracking method is based on particle filtering, namely the annealing particle filtering (APF). The APF is capable of recovering full articulated body motion efficiently and leads to very robust tracking results.

Next, two approaches for generating interactive behaviours for a virtual character were considered: the standard Viterbi algorithm and the windowed Viterbi algorithm. Both methods were presented based on the dual-input HMM. The dual-input HMM was learnt on two persons' MoCap data, with two sets of states. It can be used for synthesising and estimating the motion data for person **B** from the motion data for person **A** by statistically encoding relations between them to produce animation for a virtual character. Both the standard Viterbi algorithm and the windowed Viterbi algorithm can be used for generating very similar behaviours to the real behaviours, and it was shown that the windowed Viterbi algorithm can be used for generating behaviours in real-time. The generated motion was also mapped to a virtual character and then a number of animations of a virtual character interacting with a real person in an original video sequence were produced. These animations can be found on the attached CD. The produced animations have smooth, natural behaviours.

Finally, a novel approach was developed for finding the places to

split the complex motion data automatically in order to improve the model of human motion for obtaining better tracking results . The combined HMM was then introduced. The idea of this model is to train the separate models on the split motion data and then combine them together. Experimental results showed that the combined HMM trained on the split data can represent the data distribution more accurately than the normal model trained on 3D MoCap data representing the complex motion at once. Next this model was applied to track 3D articulated human motion in the video sequences (the same video sequences as Chapter 5).

In summary, the main contributions in this thesis are:

- A novel approach for generating intelligent behaviours for fully articulated 3D virtual characters on the basis of visual analysis of the motion of a real person in ordinary 2D video using the dual-input HMM and the standard Viterbi algorithm.

- A new approach for generating interactive behaviours for virtual characters using the windowed Viterbi algorithm, capable of doing so in real-time.

- A novel method for improving the model of the human motion due to the ability to split the complex human motion automatically.

8.2 Future Research

Many opportunities for future research await to be explored. In Chapter 4, an HMM was introduced to learn the motion of real people. Because the MoCap data used in this work are high-dimensional, it is necessary to reduce the dimensionality using PCA before the processing

starts. When modelling human motion in the reduced dimensionality eigenspace, there is a balance between keeping detail in the model and the growing dimensionality the of the eigenspace. The more detail we keep, the higher is the dimensionality of the eigenspace representing the data distribution. It is possible to extend this model to the hierarchical model [46, 123] in order to keep as much of detail in the model as possible. The idea of the hierarchical model of human motion comes from analysing the motion of different body parts as the whole body performs some action. Through the use of the hierarchy, we expect to improve the accuracy of the model of human motion.

The windowed Viterbi algorithm in Chapter 6 has been utilised for generating interaction behaviours for a virtual character responding to the tracked person in real video. The window size was fixed in this algorithm. This can possibly be extended to incremental Viterbi algorithm [124]. That means the length of window size can be increased. The incremental Viterbi algorithm is shown to reduce memory usage in long state sequence problems compared with the standard Viterbi algorithm.

As we described before, virtual characters are becoming more and more popular and used in many applications such as character animation, computer games, films and virtual environments. It can also be extended to the application in television programme and education area. Those virtual humans would interact with people through speech and gestures. In the television programme, virtual presenters can be created and used for presenting programmes in different languages. In the education area, virtual humans can be used for providing tutor support for students in e-learning courses by answering questions and

offering help with problems. This would make the learning process with computers more enjoyable for students [125]. In a word, realistic interactive virtual characters will almost certainly populate our near future, guiding us toward opportunities to learn, enjoy, and consume [87].

Appendix A

PHASESPACE MOTION DIGITIZER SYSTEM

PhaseSpace is a high resolution, real time optical motion capture system. A PhaseSpace motion digitizer captures complex motion data in real-time using advanced hardware and software technology [4]. Motion capture is accomplished by placing the PhaseSpace cameras around an area, and moving subjects with the LED markers attached to them. The cameras detect the positions of the LED markers and transmit information to a central computer, that processes the data and calculates and stores the actual X, Y and Z positions for the markers. Then, the marker positional data can be used to do motion analysis and applications.

The basic PhaseSpace system consists of:

- CCD (Charge Couple Device) video cameras
- LED drivers
- Infra-red LEDs
- A HUB into which the cameras and LED drivers connect
- A server computer which runs Linux and communicates with the

HUB

- Calibration objects
- Server software
- Dynamic link libraries that enable a user to construct client programs

The PhaseSpace motion digitizer system consists of a number of specialised CCD (Charge Couple Device) cameras. Each "camera" is high-speed at up to 480 fps and high-resolution with $3,600 \times 3,600$ (12 Megapixel), and used to measure the position of infra-red LED markers in real time. Each LED marker placed on the human body should be visible from at least two cameras at any time, or preferably from three cameras. All LEDs are attached to LED strings which are connected to an LED drive unit. Figure A.1 shows the layout of the whole PhaseSpace product. For the full body motion capture, the best place to locate cameras is in a circular configuration with the field of view being in the centre of the circle. The greater the field of view desired, the larger the circle should be.

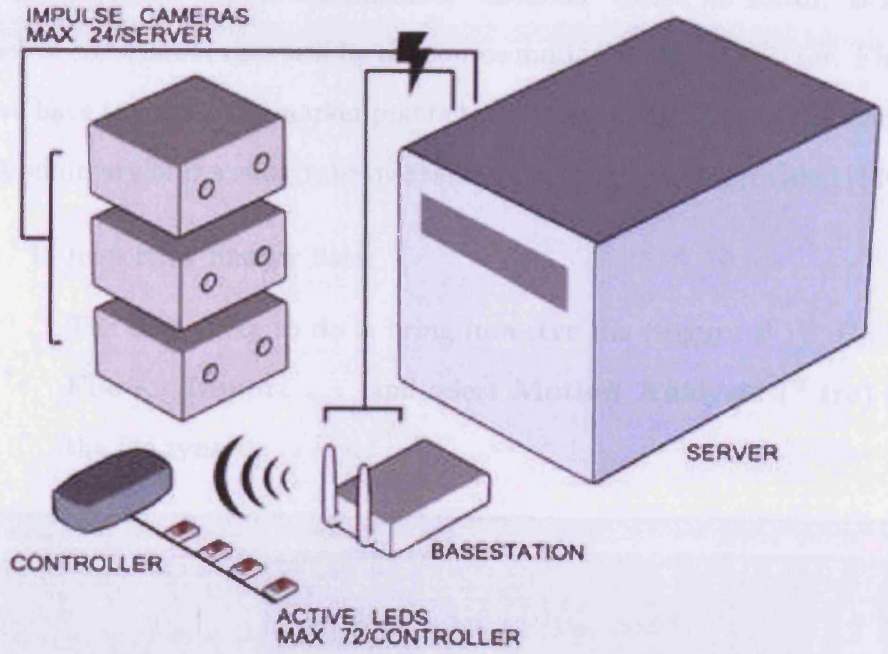


Figure A.1. PhaseSpace products layout [4].

Appendix B

MOTIONBUILDER: ACTOR

MotionBuilder has an intermediate "skeleton" called an Actor. It's a set of constraints that will be the source motion for the character. First we have to match the marker points to corresponding areas of the actor. A summary of the match the markers to the actor is now provided [126].

1. Importing marker data.

   The first thing to do is bring in a **.trc** file (Figure B.1). Go to **File -> Import ...** and select **Motion Analysis (*.trc)** as the file type.

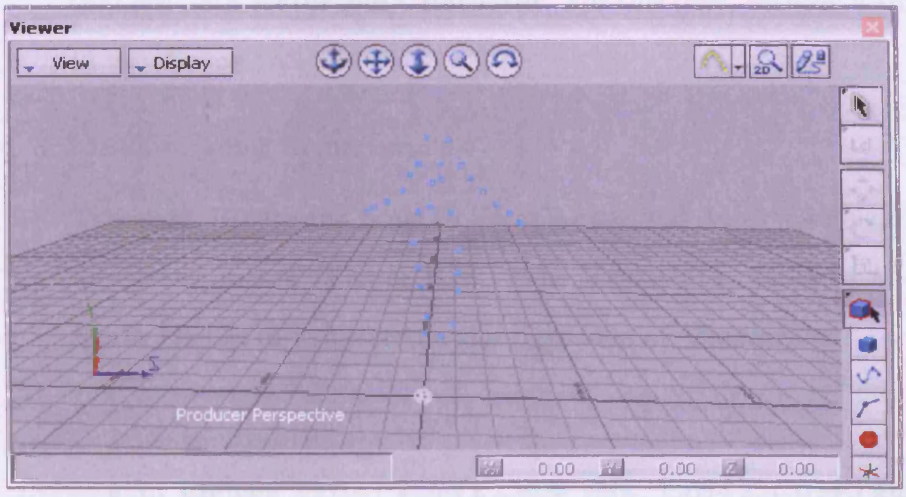


Figure B.1. 3D motion data

2. Bringing in Actor.

Next, we need to drag an actor into the space. The **Actor** icon is located in the **Templates** section of the **Asset Browser** window (Figure B.2).

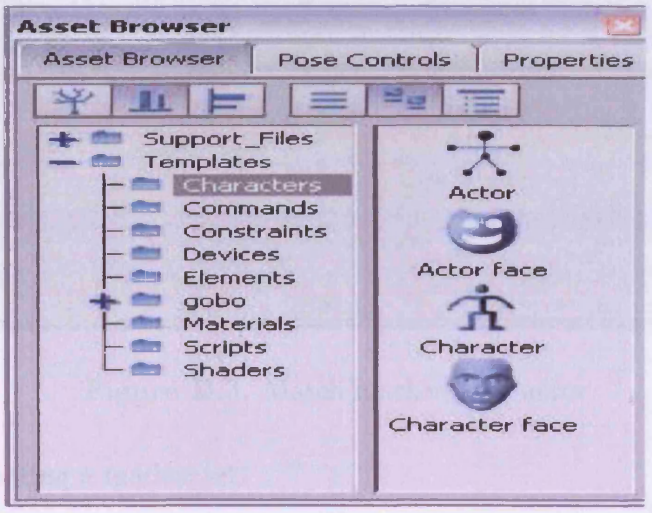


Figure B.2. Asset Browser window

Within **Templates** directory is the **Characters** section. Within that are both **Actor** and **Character**. For this part, we only want to use **Actor**.

3. Matching actor to markers.

   Translate, Rotate and Scale Uniform **Actor** and Actor's segments until they match (Figure B.3). The **Translate**, **Rotate** and **Scale Uniform** icons are on the right side of **Viewer**.

   Note:

   - In **Viewer**, Ctrl+1 gives a single view. Ctrl+2, Ctrl+3 and Ctrl+4 give two, three and four views respectively.
   - Ctrl+Z can used to undo the last operation.

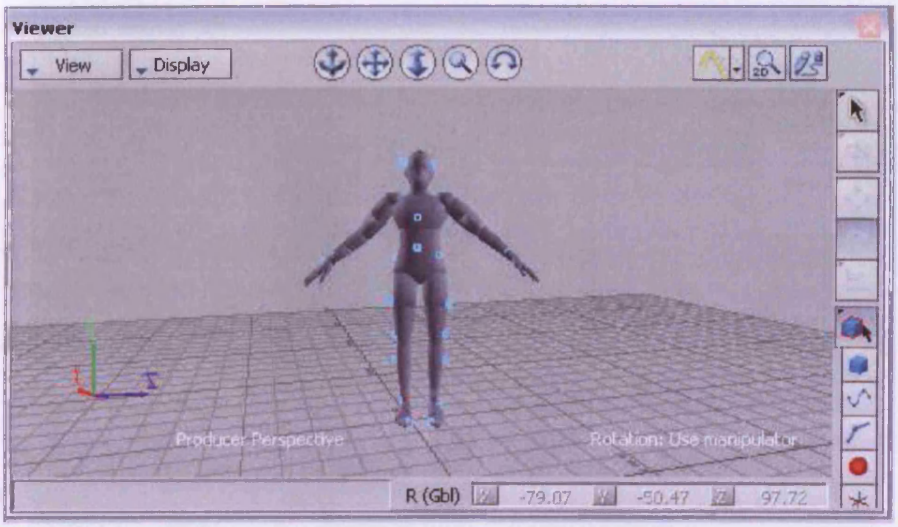


Figure B.3. Match markers and actor

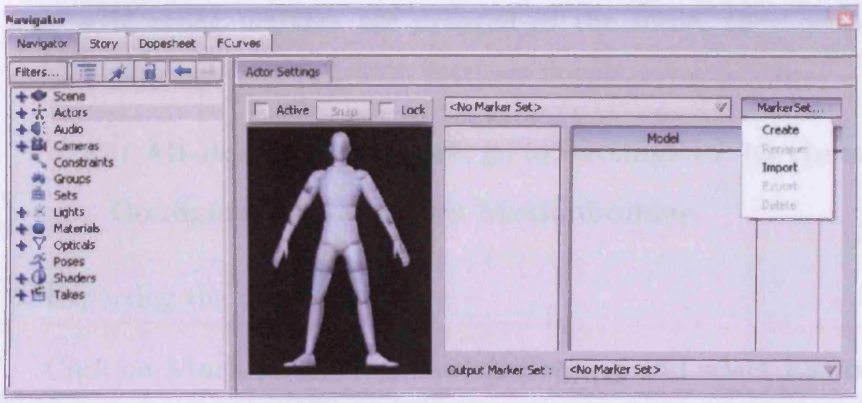
4. Creating a marker set.

   Once the actor is fitting within the markers, we need to decide which markers go with which part of **Actor**.

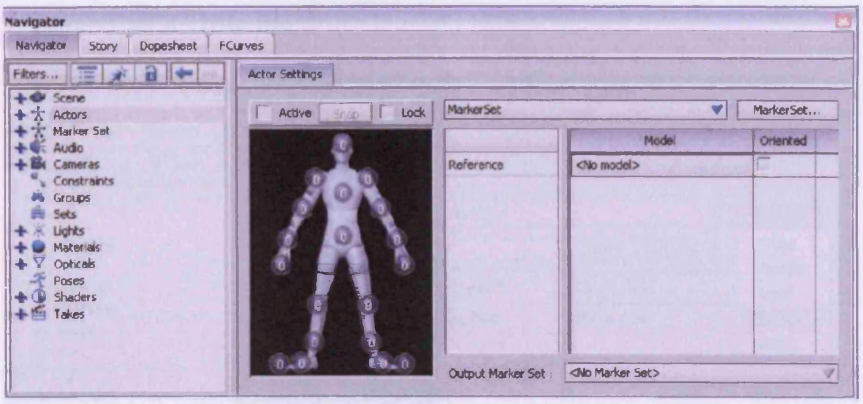
   In the **Navigator** window, click on **Actor** under **Actors**. **Actor Setting** should appear. Click on **MarkerSet...** and select **create** from the drop down menu (Figure B.4 (a)). Now we can see different areas representative of the head, arm, etc:, with the number 0 (Figure B.4 (b)). We need to select markers in the **Viewer** and drop them into the correct locations (Figure B.4 (c)).

   Notes:

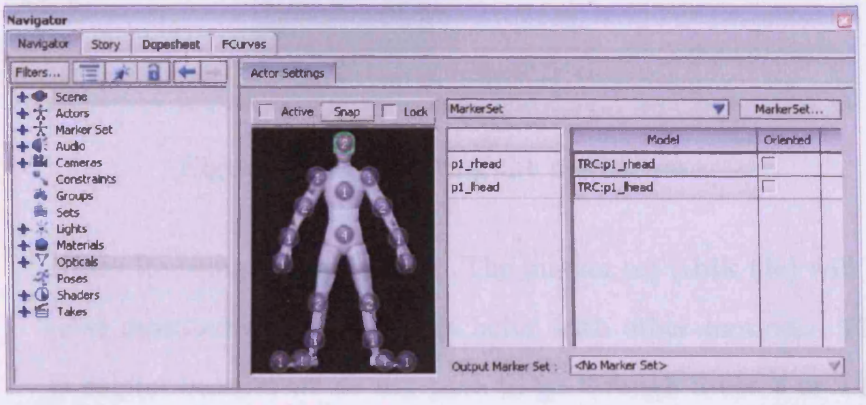
   - **Alt-drag** is used for dropping the markers to the correct location.
   - The number of the markers in the hip of the actor must be



(a)



(b)



(c)

Figure B.4. Navigator window for create actor

at least 3.

- If wrong markers are dragged in the scene, they can be deleted.
- If **Alt-drag** does not work, go to **Settings -> Keyboard Configurations** and select **Motionbuilder**.

5. Exporting the marker set.

   Click on **MarkerSet...** in **Actor Setting** and select **Export** from the drop down menu (Figure B.5). The marker set will be saved as a **.hik** file.

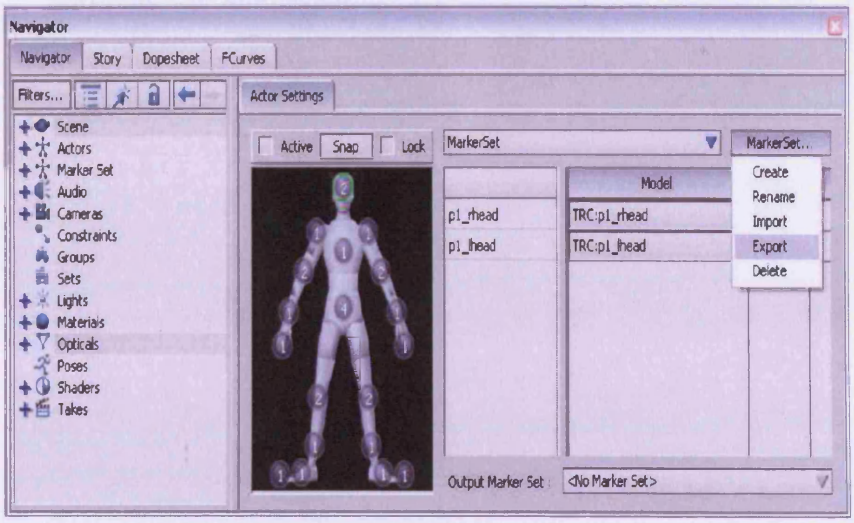


Figure B.5. Exporting the marker set

Note: Re-using the marker set. The marker set (**.hik** file) will be used repeatedly to apply to an actor with other motions. This is helpful because we do not have to go through Steps 3 or 4 for every motion.

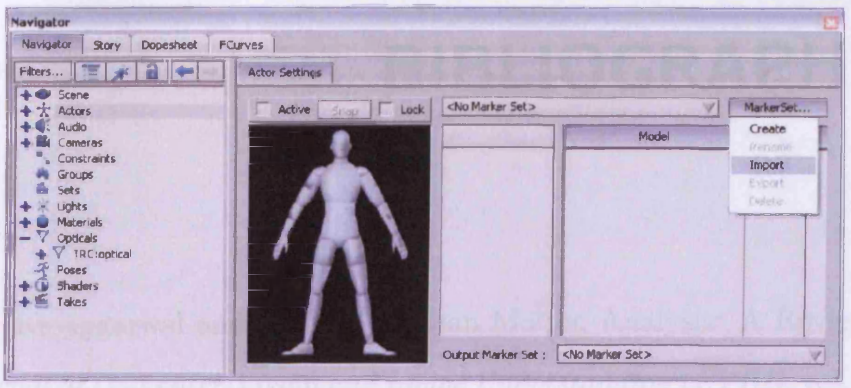
(a) Import a new **.trc** file and a new **Actor**.

(b) In the **Navigator** window, click on **Actor** under **Actors**. Click on **MarkerSet...** in the **Actor Settings** and select **import** from the drop down menu (Figure B.6 (a)).

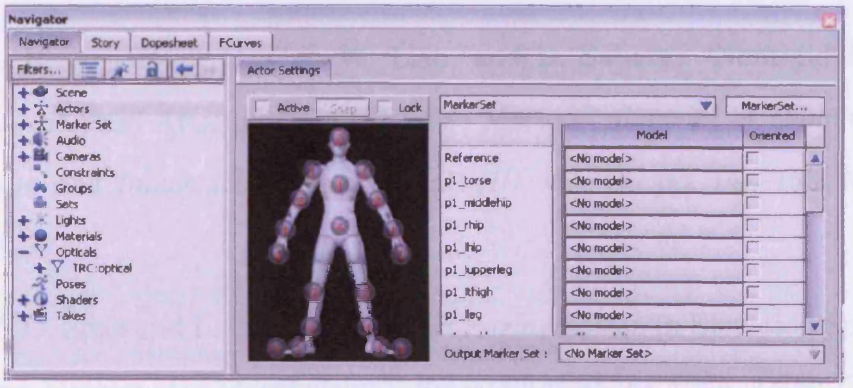
(c) Select all of the optical data points, alt-drag, and drop them into **Model** part of the **Reference** section of the actor (Figure B.6 (b)). Then **MotionBuilder** drops all the markers into the correct place (Figure B.6 (c)).

6. Activate the character.

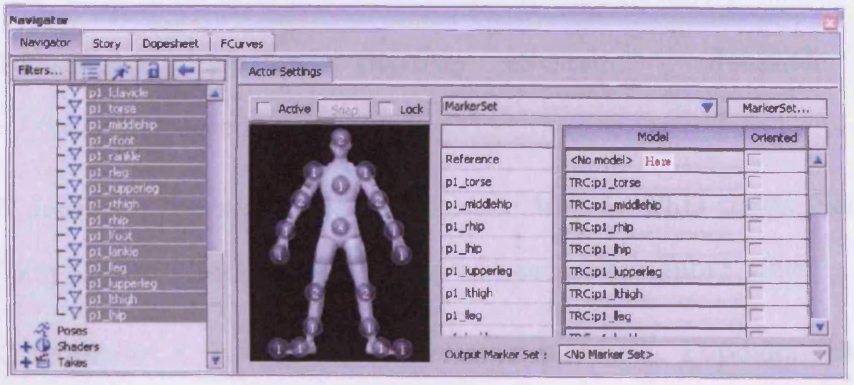
   Activate the character by clicking on **Actor** in **Actor Setting** and play the animation.



(a)



(b)



(c)

Figure B.6. Re-use the market set.

BIBLIOGRAPHY

[1] J. K. Aggarwal and Q. Cai, "Human Motion Analysis: A Review," *Journal of Computer Vision and Image Understanding (CVIU)*, vol. 73, no. 3, pp. 428–440, 1999.

[2] J. K. Aggarwal, Q. Cai, W. Liao, and B. Sabata, "Nonrigid Motion Analysis: Articulated and Elastic Motion," *Journal of Computer Vision and Image Understanding (CVIU)*, vol. 70, pp. 142–156, May 1998.

[3] Y. C. Shao and L. C. Chen, "Object Segmentation in Elevation Space Using Mathematic Morphology," *Proceedings of the 22nd Asian Conference on Remote Sensing*, pp. 227–232, 2001.

[4] "PhaseSpace Motion Digitizer System," *Available at http://www.phasespace.com/*, 2007.

[5] N. Johnson, *Learning Object Behaviour Models.* PhD thesis, School of Computer Studies, The University of Leeds, September 1998.

[6] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popovic, "Style-based Inverse Kinematics," *ACM Transactions on Graphics (TOG)*, vol. 32, pp. 522–531, August 2004.

[7] A. Safonova, J. K. Hodgins, and N. S. Pollard, "Synthesizing Physically Realistic Human Motion in Low-dimensional, Behavior-specific

Spaces," *ACM Transactions on Graphics Journal*, vol. 23, pp. 514–521, August 2004.

[8] "Alias MotionBuilder 6 User's Guide," November 2004.

[9] M. Isard and A. Blake, "CONDENSATION - Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[10] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A Tutorial on Particle Filters for On-line Nonlinear/Non-Gaussian Bayesian Tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, February 2002.

[11] A. O. Balan, L. Sigal, and M. J. Black, "A Quantitative Evaluation of Video-based 3D Person Tracking," *Proceedings of the 14th International Conference on Computer Communications and Networks*, pp. 349–356, October 2005.

[12] J. Deutscher, A. Blake, and I. Reid, "Articulated Body Motion Capture by Annealed Particle Filtering," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 126–133, 2000.

[13] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, February 1989.

[14] M. Pilu, "Video Stabilization as A Variational Problem and Numerical Solution with the Viterbi Method," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 625–630, 2004.

[15] N. I. Badler, "Real-Time Virtual Humans," *Proceedings of 5th Pacific Conference on Computer Graphics and Applications*, pp. 4–13, October 1997.

[16] T. K. Capin, H. Noser, D. Thalmann, I. S. Pandzic, and N. M. Thalmann, "Virtual Human Representation and Communication in the VLNet Networked Virtual Environments," *IEEE Computer Graphics and Applications*, vol. 17, pp. 42–53, March 1997.

[17] V. B. Zordan and N. C. V. D. Horst, "Mapping Optical Motion Capture Data to Skeletal Motion using a Physical Model," *Proceedings of ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 245–250, 2003.

[18] M. Meredith and S. Maddock, "Adapting Motion Capture Data using Weighted Real-time Inverse Kinematics," *Comput. Entertain.*, vol. 3, pp. 5–5, January/March 2005.

[19] G. J. Wen, Z. Q. Wang, S. H. Xia, and D. M. Zhu, "From Motion Capture Data to Character Animation," *Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 165–168, 2006.

[20] I. A. Karaulova, P. M. Hall, and A. D. Marshall, "A Hierarchical Model of Dynamics for Tracking People with a Single Video Camera," *British Machine Vision Conference (BMVC)*, pp. 262–352, Septemeber 2002.

[21] E. Hsu, K. Pulli, and J. Popovic, "Style Translation for Human Motion," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 1082–1089, 2005.

[22] E. Hsu, S. Gentry, and J. Popovic, "Example-based control of human motion," *Proceedings of ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 69–77, 2004.

[23] M. K. Leung and Y. H. Yang, "First Sight: A Human Body Outline Labeling System," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 4, pp. 359–377, 1995.

[24] D. C. Hogg, "Model-Based Vision: a Program to See a Walking Person," *Image and Vision Computing*, vol. 1, pp. 5–20, February 1983.

[25] I. A. Karaulova, P. M. Hall, and A. D. Marshall, "Tracking People in Three Dimensions using a Hierarchical Model of Dynamics," *Image and Vision Computing*, vol. 20, pp. 691–700, August 2002.

[26] D. Thalmann, "Human Modelling and Animation," *Eurographics '93 State-of-the-Art Reports, Chapter 7*, 1993.

[27] J. M. Rehg and T. Kanade, "Model-based Tracking of Self-occluding Articulated Objects," *Proceedings of the 5th International Conference on Computer Vision*, pp. 612–617, June 1995.

[28] L. Goncalves, E. D. Bernardo, E. Ursella, and P. Perona, "Monocular Tracking of the Human Arm in 3D," *Proceedings of the 5th International Conference on Computer Vision*, pp. 764–770, 1995.

[29] S. Park and J. K. Aggarwal, "Recognition of Two-person Interactions using a Hierarchical Bayesian Network," *International Multimedia Conference, First ACM SIGMM International Workshop on Video Surveillance*, pp. 65–76, 2003.

[30] F. Caillette, A. Galata, and T. Howard, "Real-Time 3-D Human Body Tracking using Variable Length Markov Models," *British Machine Vision Conference (BMVC)*, vol. 1, pp. 469–478, September 2005.

[31] R. Bowden, "Learning Statistical Models of Human Motion," *IEEE Workshop on Human Modelling, Analysis and Synthesis*, pp. 10–17, July 2000.

[32] R. Urtasun, D. J. Fleet, and A. H. P. Fua, "Priors for People Tracking from Small Tracking Sets," *Proceedings of the 10th IEEE International Conference on Computer Vision*, pp. 403–410, October 2005.

[33] N. D. Lawrence, "Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data," *Advances in Neural Information Processing Systems*, pp. 329–336, 2003.

[34] N. D. Lawrence, "The Gaussian Process Latent Variable Model," tech. rep., The University of Sheffield, Department of Computer Science., 2006.

[35] D. J. C. Mackay, "Introduction to Gaussian Processes," *In: C. M. Bishop (Editor), Neural Networks and Machine Learning, NATO ASI Series, Kluwer Academic Press*, pp. 133–166, 1998.

[36] F. Caillette, *Real-Time Markerless 3-D Human Body Tracking.* PhD thesis, School of Computer Science, University of Manchester, 2006.

[37] A. Safonova, *Reducing the Search Space for Physically Realistic Human Motion Synthesis.* PhD thesis, School of Computer Science, Carnegie Mellon University, September 2006.

[38] I. T. Jolliffe, *Principal Component Analysis.* Springer Series in Statistics. Springer-Verlag, 2nd ed., 2002.

[39] R. Bowden, *Learning Non-linear Models of Shape and Motion.* PhD thesis, Department of Systems Engineering, Brunel University, October 1999.

[40] N. F. Troje, "Decomposing biological motion: A framework for analysis and synthesis of human gait patterns," *Journal of Vision*, vol. 2, p. 371C387, 2002.

[41] D. Cosker, D. Marshall, P. Rosin, and Y. A. Hicks, "Speech Driven Facial Animation using a Hidden Markov Coarticulation Model," *17th International Conference on Pattern Recognition (ICPR)*, vol. 1, pp. 128–131, August 2004.

[42] P. Glardon, R. Boulic, and D. Thalmann, "PCA-based Walking Engine using Motion Capture Data," *Proceedings of Computer Graphics International (CGI)*, pp. 292–298, June 2004.

[43] S. R. Carvalho, R. Boulic, and D. Thalmann, "Interactive Low-dimensional Human Motion Synthesis by Combining Motion Models and PIK," *Computer Animation and Virtual Worlds*, vol. 18, pp. 493–503, September 2007.

[44] L. Wang, W. Hu, and T. Tan, "Recent Developments in Human Motion Analysis," *Pattern Recognition*, vol. 36, pp. 585–601, March 2003.

[45] D. M. Gavrila, "The Visual Analysis of Human Movement: A Survey," *Computer Vision and Image Understanding*, vol. 73, pp. 82–98, January 1999.

[46] Y. A. Hicks, *Modelling and Tracking of Articulated Human Motion.* PhD thesis, Cardiff University, School of Computer Science, September 2003.

[47] E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau, "Robust Real-time Visual Tracking using a 2D-3D Model-based Approach," *IEEE Transacations on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 932–946, July 2002.

[48] Y. Matsumoto, T. Kato, and T. Wada, "An Occlusion Robust Likelihood integration Method for Multi-Camera People Head Tracking," *Proceedings of 4th International Conference on Newworked Sensing Systems*, pp. 235–242, June 2007.

[49] N. Howe, M. Leventon, and W. Freeman., "Bayesian Reconstruction of 3D Human Motion from Single-Camera Video," *Advances in Neural Information Processing Systems*, 1999.

[50] J. J. Wang and S. Singh, "Video Analysis of Human Dynamics: A Survey," *Real-Time Imaging*, vol. 9, pp. 321–346, October 2003.

[51] N. T. Siebel and S. Maybank, "Real-Time Tracking of Pedestrians and Vehicles," *2nd IEEE Workshop on Performance Evaluation of Tracking and Surveillance*, December 2001.

[52] I. Haritaolu, D. Harwood, and L. S. Davis, "W4: Real-time Surveillance of People and Their Activities," *IEEE Transactions Pattern Analysis, and Machine Intelligence*, vol. 22, no. 8, pp. 809–830, 2000.

[53] J. M. Ferryman, S. J. Maybank, and A. D. Worrall, "Visual Surveillance for Moving Vehicles," *International Journal of Computer Vision*, vol. 37, pp. 187–197, June 2000.

[54] R. Torre, P. Fua, S. Balcisoy, M. Ponder, and D. Thalmann, "Interaction Between Real and Virtual Humans: Playing Checkers," *Proceedings of Eurographics Workshop On Virtual Environments*, 2000.

[55] M. Meredith and S. Maddock, "Inverse Skinning," *3rd European Conference on Visual Media Production*, pp. 163–172, 2006.

[56] Y. Zheng, Y. Hicks, D. Cosker, and D. Marshall, "Generating human interactive behaviours using the windowed Viterbi algorithm," *3rd International Conference on Computer Graphics Theory and Applications*, January 2008.

[57] T. Starner and A. Pentland, "Real-Time American Sign Language Recognition from Video Using Hidden Markov Models," tech. rep., Perceptual Computing Section Technical Report No. 375, MIT Media Lab, Cambridge, MA, 1995.

[58] Y. Zheng, Y. Hicks, D. Cosker, D. Marshall, J. C. Mostaza, and J. A. Chambers, "Virtual Friend: Tracking and Generating Natural Interactive Behaviours in Real Video," *Proceedings of the 8th Internation Conference on Signal Processing, China*, November 2006.

[59] N. Jonson, A. Galata, and D. Hogg, "The Acquisition and Use of Interaction Behaviour Model," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 866–871, June 1998.

[60] Y. Zheng, Y. Hicks, D. Cosker, D. Marshall, and J. A. Chambers, "Generating 3D Interactive Behaviours," *Proceedings of the 3rd European Conference on Visual Media Production (CVMP 2006), London*, November 2006.

[61] B. Stenger, P. R. S. Mendonça, and R. Cipolla, "Model-Based 3D Tracking of an Articulated Hand," *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 310–315, December 2001.

[62] K. Nirei, H. Saito, M. Mochimaru, and S. Ozawa, "Human Hand Tracking from Binocular Image Sequences," *Proceedings of 22th International Conference on Industrial Electronics, Control, and Instrumentation*, pp. 297–302, August 1996.

[63] S. X. Ju, M. J. Black, and Y. Yacoob, "Cardboard People: a Parameterized Model of Articulated Image Motion," *Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition*, pp. 38–44, October 1996.

[64] F. Lerasle, G. Rives, M. Dhome, and A. Yassine, "Human Body Tracking by Monocular Vision," *Proceedings of the 4th European Conference on Computer Vision*, pp. 518–527, 1996.

[65] M. H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting Faces in Images: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 34–58, January 2002.

[66] S. Gong and H. Buxton, "Bayesian Nets for Mapping Contextual Knowledge to Computational Constraints in Motion Segmentation and Tracking," *British Machine Vision Conference (BMVC)*, pp. 229–239, September 1993.

[67] S. Gil, R. Milanese, and T. Pun, "Combining Multiple Motion Estimates for Vehicle Tracking," *European Conference on Computer Vision (ECCV)*, vol. 2, pp. 307–320, 1996.

[68] C. Cedras and M. Shah, "Motion-based Recognition: A Survey," *Image and Vision Computing*, vol. 13, pp. 129–155, March 1995.

[69] R. E. Kalman, "A New approach to Linear Filtering and Prediction Problems," *Transactions of the ASME, Ser. D., Journal of Basic Engineering*, vol. 82, pp. 34–45, March 1960.

[70] E. Cuevas, D. Zaldivar, and R. Rojas, "Kalman Filter for Vision Tracking," tech. rep., Freie Universität Berlin, Institue für Informatik, Takuster, Germany, 2005.

[71] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," *SIGGRAPH*, 2001.

[72] A. Doucet, J. F. G. de Freitas, and N. J. Gordon, "An Introduction to Sequential Monte Carlo Methods," *In: A. Doucet, J. F. G. de Freitas and N. J. Gordon (Editors), Sequential Monte Carlo Methods in Practice, Springer-Verlag, New York*, 2001.

[73] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel Approach to Nonlinear and Non-gaussian Bayesian State Estimation," *IEE Proceedings-F, Radar and Signal Processing*, vol. 140, pp. 107–113, April 1993.

[74] C. Musso, N. Oudjane, and F. LeGland, "Improving Regularised Particle Filter," *In: A. Doucet, J. F. G. de Freitas and N. J. Gordon (Editors), Sequential Monte Carlo Methods in Practice, Springer-Verlag, New York*, pp. 247–271, 2001.

[75] A. H. Jazwinski, *Stochastic Processes and Filtering Theory.* Academic Press: New York, 1970.

[76] Z. Khan, T. Balch, and F. Dellaert, "MCMC-Based Particle Filtering for Tracking a Variable Number of Interacting Targets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1805–1918, November 2005.

[77] A. M. Baumberg and D. C. Hogg, "Learning Flexible Models from Image Sequence," *Proceedings of the 3rd European Conference on Computer Vision*, vol. 1, pp. 299–308, 1994.

[78] T. Jebara and A. Pentland, "Statistical Imitative Learning from Perceptual Data," *Proceedings of the 2nd International Conference on Development and Learning*, pp. 191–196, June 2002.

[79] L. R. Rabiner and B. H. Juang, "An Introduction to Hidden Markov Models," *IEEE ASSP Journal*, pp. 4–16, June 1986.

[80] D. Hogg, N. Johnson, R. Morris, and D. Buesching, "Visual Models of Interaction," *Proceedings of 2nd International Workshop on Cooperative Distributed Vision, Kyoto, Japan*, pp. 5/1–5/2, July 1998.

[81] N. M. Oliver, B. Rosario, and A. P. Pentland, "A Bayesian Computer Vision System for Modelling Human Interactions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 831–843, August 2000.

[82] S. Park and J. K. Aggarwal, "A Hierarchical Bayesian Network for Event Recognition of Human Actions and Interactions," *ACM Journal of Multimedia Systems*, pp. 164–179, October 2004.

[83] S. Park and J. K. Aggarwal, "Event Semantics on Two-person Interactions," *International Conference on Pattern Recognition*, vol. 4, pp. 227–230, August 2004.

[84] M. Lau and J. J. Kuffner, "Behavior Planning for Character Animation," *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, August 2005.

[85] D. Cosker, D. Marshall, P. Rosin, and Y. A. Hicks, "Video Realistic Talking Heads using Hierarchical Non-linear Speech-appearance Models," *Proceedings of Mirage*, pp. 20–27, Match 2003.

[86] D. Cosker, D. Marshall, P. Rosin, and Y. A. Hicks, "Speaker-independent Speech-driven facial Animation using a Hierarchical Facial Model," *IEE Proceedings of Visual Information Engineering (VIE)*, pp. 169–172, July 2003.

[87] J. Gratch, J. Rickel, E. Andre, J. Cassell, E. Petajan, and N. Badler, "Creating Interactive Virtual Humans: Some Assembly Required," *IEEE Intelligent Systems*, vol. 17, pp. 54–63, July/August 2002.

[88] T. Horprasert, D. Harwood, and L. S. Davis, "A Statistical Approach for Real-time Robust Background Subtraction and Shadow Detection," *Proceedings of IEEE International Conference on Computer Vision (ICCV) FRAME-RATE Workshop, Kerkyra, Greece*, pp. 1–19, September 1999.

[89] M. Brand, "Voice puppetry," *Proceedings of ACM SIGGRAPH*, pp. 21–28, 1999.

[90] "Autodesk MotionBuilder," *Available at http://www.autodesk.com/*, November 2004.

[91] "Autodesk Maya," *Available at http://www.autodesk.com/*, 2008.

[92] C. Grow, I. Gordon, R. D. Stuart, and A. Adalja, "Motion Capture as a Means for Data Aquisition," *Available at http://vizproto.prism.asu.edu/datacapture/motioncapture1/*, 1998.

[93] "Motion Capture," *Available at http://www.metamotion.com/motion-capture/motion-capture.htm.*

[94] B. Delaney, "On the Trail of the Shadow Woman: the Mystery of Motion Capture," *IEEE Computer Graphics and Applications*, vol. 18, no. 5, pp. 14–19, 1998.

[95] "Motion Capture (Mocap) Studios," *Available at http://www.motioncapturestudios.com/article/types_of_mocap_system.htm.*

[96] J. F. O'Brien, R. E. Bodenheimer, G. J. Brostow, and J. K. Hodgins, "Automatic Joint Parameter Estimation from Magnetic Motion Capture Data," *Proceedings of Graphics Interface*, pp. 53–60, 2000.

[97] "Motion capture," *Available at http://en.wikipedia.org/wiki/Motion_capture.*

[98] "Wireless Data Glove: The CyberGlove II System," *Available at http://www.immersion.com/3d/products/cyber_glove.php.*

[99] "CMU Graphics Lab Motion Capture Database," *Available at http://mocap.cs.cmu.edu/.*

[100] L. I. Smith, "A Tutorial on Principal Component Analysis," *Available at http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf*, February 2002.

[101] J. E. Jackson, *A User's Guide to Principal Components.* Wiley-IEEE, 1st ed., 1991.

[102] P. E. Hart, R. O. Duda, and D. G. Stork, *Pattern Classification.* A Wiley-interscience Publication, 2nd ed., 2001.

[103] C. J. Needham, *Tracking and Modelling of Team Game Interactions.* PhD thesis, The University of Leeds, School of Computing, October 2003.

[104] J. R. Movellan, "Tutorial on Hidden Markov Models," tech. rep., Machine Perception Laboratory, 1995.

[105] J. R. Deller, J. H. L. Hansen, and J. G. Proakis, *Discrete-Time Processing of Speech Signals.* Wiley-IEEE Press, 1999.

[106] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, pp. 671–680, May 1983.

[107] Z. Zhang, "A Flexible New Technique for Camera Calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[108] P. Çivicioglu and E. Besdok, "Implicit Camera Calibration by Using Resilient Neural Networks," *The 13th International Conference on Neural Information Processing (ICONIP)*, pp. 632–640, 2006.

[109] P. Parent, *Computer Animation: Algorithms and Techniques.* Morgan Kaufmann, 2nd ed., 2007.

[110] M. X. Li and J. Lavest, "Some Aspects of Zoom Lens Camera Calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 1105–1110, Novermber 1996.

[111] J. Y. Bouguet, "Camera Calibration Toolbox for Matlab," *Available at http://www.vision.caltech.edu/bouguetj/calib_doc/.*

[112] "Open Source Computer Vision Library," *Available from http://www.intel.com/technology/computing/opencv/index.htm.*

[113] M. Brand, "An Entropic Estimator for Structure Discovery," *Proceedings of Neural Information Processing Systems*, pp. 723–729, 1998.

[114] P. E. Rybski and M. M. Veloso, "Robust Real-Time Human Activity Recognition from Tracked Face Displacements," *Proceedings of the 12th Portuguese Conference on Artificial Intelligence*, vol. 1, pp. 87–98, December 2005.

[115] B. D. Ripley, *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.

[116] "Apple - Shake," *Available at http://www.apple.com/shake/.*

[117] "Shake 4 Tutorials," *Available at http://www.apple.com/*, 2005.

[118] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision.* International Thomson Publishing, 2nd ed., 1998.

[119] G. Johansson, "Visual Perception of Biological Motion and a Model for its Analysis," *Perception and Psychophysics*, vol. 14, no. 2, pp. 201–211, 1973.

[120] J. Rittscher, T. Watanabe, J. Kato, S. Joga, and A. Blake, "An HMM-based Segmentation Method for Traffic Monitoring," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, pp. 1291–1296, September 2002.

[121] K. R. Castleman, *Digital Image Processing.* Prentice Hall International, 1996.

[122] E. Birney, *Sequence Alignment in Bioinformatics.* PhD thesis, The Sanger Centre, Cambridge, U.K, March 2000.

[123] D. Cosker, *Animation of a Hierarchical Appearance Based Facial Model and Perceptual Analysis of Visual Speech.* PhD thesis, Cardiff University, School of Computer Science, July 2006.

[124] T. P. Minka, D. S. Bloomberg, and K. Popat, "Document Image Decoding using Iterated Complete Path Search," *International Symposium on Electronic Imaging: Science and Technology, Document Recognition and Retrieval VIII*, pp. 344–349, January 2001.

[125] "Virtual Humans Part of the Future?," *Available at http://www.axistive.com/virtual-humans-part-of-the-future.html*, June 2007.

[126] "MotionBuilder: Actor," *Available at http://atec.utdallas.edu/midori/Handouts/motionBuilder_actor.htm*, September 2006.