# Realistic Motion Estimation using Accelerometers

Liguang Xie

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Masters of Science
in
Computer Science and Applications

Yong Cao, Chair
Roger W. Ehrich
Francis Quek

June 18, 2009
Blacksburg, Virginia

Keywords: Performance animation, accelerometer, interpolation, optimization, motion synthesis

# Realistic Motion Estimation using Accelerometers

Liguang Xie

## Abstract

A challenging goal for both the game industry and the research community of computer graphics is the generation of 3D virtual avatars that automatically perform realistic human motions with high speed at low monetary cost. So far, full body motion estimation of human complexity remains an important open problem. We propose a realistic motion estimation framework to control the animation of 3D avatars. Instead of relying on a motion capture device as the control signal, we use low-cost and ubiquitously available 3D accelerometer sensors. The framework is developed in a data-driven fashion, which includes two phases: model learning from an existing high quality motion database, and motion synthesis from the control signal. In the phase of model learning, we built a high quality motion model of less complexity that learned from a large motion capture database. Then, by taking the 3D accelerometer sensor signal as input, we were able to synthesize high-quality motion from the motion model we learned.

In this thesis, we present two different techniques for model learning and motion synthesis, respectively. Linear and nonlinear reduction techniques for data dimensionality are applied to search for the proper low dimensional representation of motion data. Two motion synthesis methods, interpolation and optimization, are compared using the 3D acceleration signals with high noise. We evaluate the result visually compared to the real video and quantitatively compared to the ground truth motion. The system performs well, which makes it available to a wide range of interactive applications, such as character control in 3D virtual environments and occupational training.

# Dedication

*To my loving Mom and Dad for the life and wonderful education they have given me*

*To my wife Ye for the dedicated love and support in my whole life*

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Interactive Control of Full Body Animation

Computer animation has been very popular in both academia and industry for a long time since Walt Disney introduced cartoon animation worldwide. So far, automatically generating realistic character motion remains one of the main challenges in computer animation. A lot of effort has been devoted to speed up motion synthesis, reduce monetary expense and increase the motion variety. However, the most important goal is to increase the realism of synthesized motion. Computer animation can be categorized into several research areas, including full body animation, facial animation, crowd simulation, hand animation and hair animation etc. Our work focus on developing techniques for realistic human-like full body animation.

During the past 30 years of fast development, the computer animation research community has provided various techniques for the creation of character motion. In the 1980s, Walt Disney presented a traditional technique called *key-framing* which later became very popular for generation of character motion. Animators would create a series of key frames by hand; then all of the frames

in between can be generated by interpolation. *Physically based* techniques reply on the laws of physics, such as *dynamics* and *kinematics*, to realistically simulate character motion interacting with the environment. Another important technique is *motion capture*. Motion capture technique is commonly used in commercial systems because the captured data naturally keeps every detail in human motion.

Much research effort has been devoted to develop motion capture data-driven approaches to take advantage of retained attributes of the original motion and to get rid of the original limitation of motion capture systems. Traditional motion capture systems are very expensive and restricted by the capture environment. Even worse, motions that can be used are limited to the existing motion data which is difficult to edit. Some early research focused on techniques for modifying existing motion, such as blending, retargeting, path editing and physically based approaches. Gleicher presented a nice overview of early work on motion editing methods [14]. Most approaches involved the reuse of motion capture data or the generation of new motion sequences. One popular motion generation technique, called the *motion graph*, was proposed in 2002, in which a graph representation is constructed for motion capture data. Motion clips are considered as graph components, such as vertexes or edges. New animation sequences can be synthesized by searching for a path that satisfies certain user's constraints, such as sketched paths [22, 24], motion duration and specific pose at given keyframes [3] or behavior selection [24]. In the past decade, motion capture data-driven approaches remain the most popular in research of full-body character animation.

In the research of full body character animation, providing the user with an intuitive interface to control motion of characters is difficult. The difficulty arises because human motion includes a wealth of details. The motion data is usually high dimensional while most of available input devices are not. Input devices include mice, keyboards, joysticks and other devices such as vision based tracking systems (e.g., Sony© EyeToy) and inertial/accelerometer sensors (e.g., Wii™ Controller). Such user interfaces provide simple and direct control signals with limited numbers of

degrees of freedom, such as target location, velocity of movement and behavior type. The input information is not enough and has to be supplemented using a prerecorded database that provides various behaviors and transitions to generate the complete motion of characters. Using most user interfaces, therefore, it is difficult to provide performance-driven control for complex human motions. Among the available input devices, accelerometers can provide users with intuitive control. Additionally accelerometers have the potential to be ubiquitous because of their low cost and the ”real” feeling of control.

## 1.2 Problem Statement

Automatic generation of realistic character motion remains one of the main challenges in computer graphics. Traditional motion capture techniques are very expensive and restricted by the capture environment. Recently, data-driven motion capture approaches for character animation can preserve the nature of the original motion, but synthesized motion is limited to existing motion data that is difficult to edit. Thus a large motion database is required to extend scalability. In addition, accelerometer-based animation systems have become very popular for the research of character animation [39, 40] because accelerometers are low-cost and can be found everywhere. However, the inaccuracy of the acceleration signal makes it very difficult to create realistic animation directly controlled by accelerometers.

Our goal in this thesis is to provide the research community with a motion estimation framework to control the animation of 3D avatars interactively using a set of low-cost 3D accelerometers. Our work involves developing techniques for model learning and motion synthesis using a specific type of control signal – the 3D acceleration signal received from accelerometers.

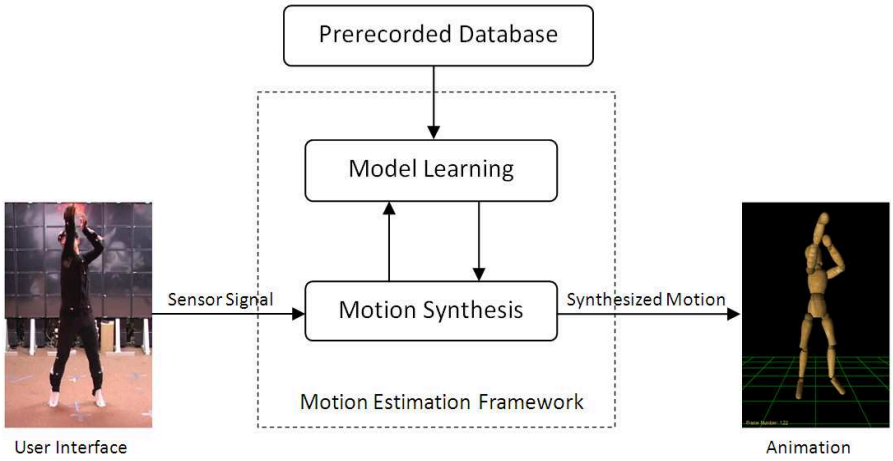



Figure 1.1: System architecture.

## 1.3 Methodology

In this thesis, we propose a realistic motion synthesis framework to control the animation of 3D avatars interactively using a set of low-cost 3D accelerometers. A prototype system is built on a small number of Nintendo© Wii™controllers which are easy to attach to the human body. Using Wii controllers as input devices, we are able to generate high quality motion data using our motion estimation framework. The system we developed is easy to set up and imposes little or no restriction on the data acquisition environment. The system takes advantage of realism and accuracy of the data-driven approach, as well as the intuitive control of 3D accelerometers.

We estimate full body motion in two phases. Figure 1.1 shows a schematic overview of our system. During the first phase, the data collection phase, we collect motion data from a professional performer using a commercially available professional motion capture system. At the same time we also capture 3D acceleration data from eight sensors (four Wii controllers) attached to the performer's body. This one-to-one mapped, time synchronized data is used to create a large, high quality motion capture database. In the second phase, the model learning phase, the data dimen-

sionality is reduced, and a set of local linear models are built from the motion capture database. In the third phase, the motion synthesis phase, we capture motion from a 'regular user' using only the attached accelerometer sensors. We then estimate the corresponding motion using a local linear model created from the motion capture database. The proposed local linear model can estimate high quality motion from low-dimensional noisy accelerometer data. Our local modeling approach also enables us to scale the database to incorporate large amounts of motion without performance degradation.

A technical contribution of our motion estimation framework is a nonlinear manifold learning model from a prerecorded human motion database which is critical for the quality of synthesized motion. The challenge of this research is to generate detailed and high dimensional motion data (more than 80 dimensions) from noisy and lower dimensional sensor signals (fewer than 25 dimensions). We adopt a data-driven approach that builds a statistical mapping model between the low dimensional input signal and high dimensional motion data by learning from a pre-captured sensor and motion database. In our approach, the database consists of motion from various sport actions, such as tennis, basketball, golf, etc. Global linear models [8, 36, 46] have difficulties in handling this type of heterogeneous motion database as the high-dimensional database contains a large number of complex and nonlinear structures (see Chapter 5 for more detail). To solve the limitation of linear models, we use *Locally Linear Embedding (LLE)* to find a nonlinear manifold in high dimensional motion data and to reduce data dimensionality. Our result suggests that the nonlinear manifold learning model has better performance than traditional linear models.

Another technical contribution of our work is a proposed optimization method for motion synthesis using an intuitive control signal. Compared to our previous work using interpolation, our optimization method has several advantages. Optimization provides an optimal and "reasonable" solution that satisfies the objective function and constraints set by users. Our results show that, using the acceleration received from accelerometers as control signals, our framework is capable

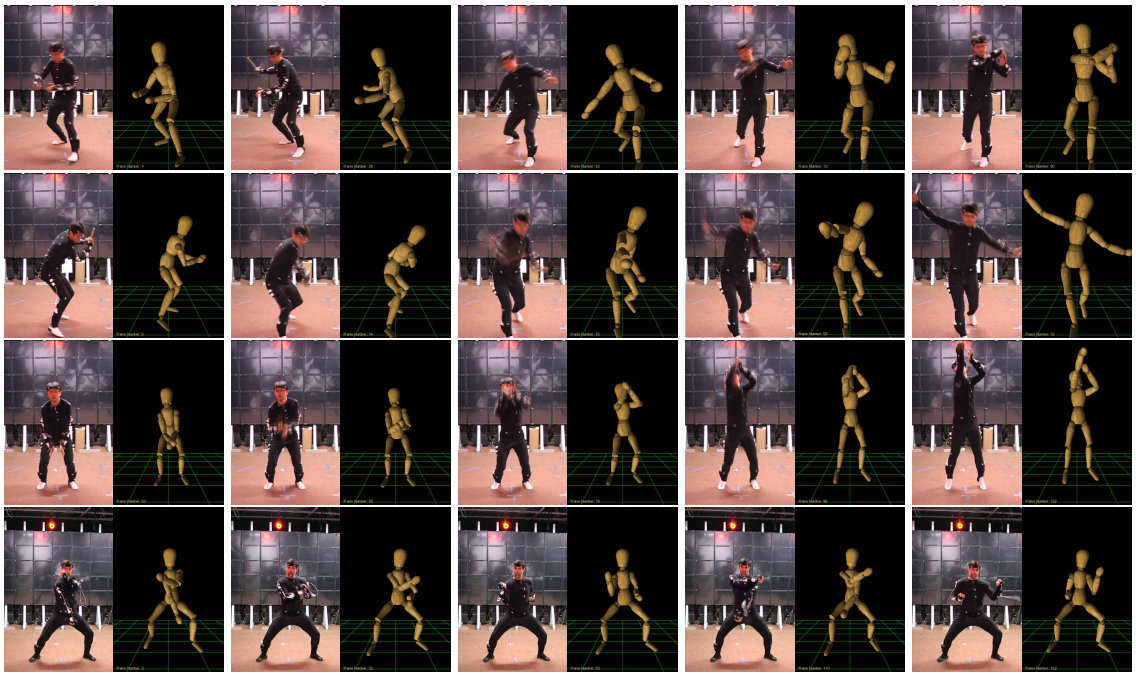


Figure 1.2: Result of motion synthesis using interpolation and PCA. Four different actions (one in each row) synthesized by our system. Each frame shows on the left side the actual pose and on the right side the synthesized pose.

of generating more natural and more smooth motion than motion interpolation methods.

We aim to make our motion synthesis framework as convenient as video capture systems and make it applicable to a wide range of applications, from video game interfaces and occupational training to interactive character control in virtual environments (VEs).

## 1.4 Summary of Results

Relying on the database built during the data collection phase, we test the performance of our system with two subjects performing various sport actions. The sensor signals are used as input to our system to produce our human motion. This subsection summarizes our results from three different techniques in our framework (Please refer Chapter 7 for more detail). Figure 1.2 shows

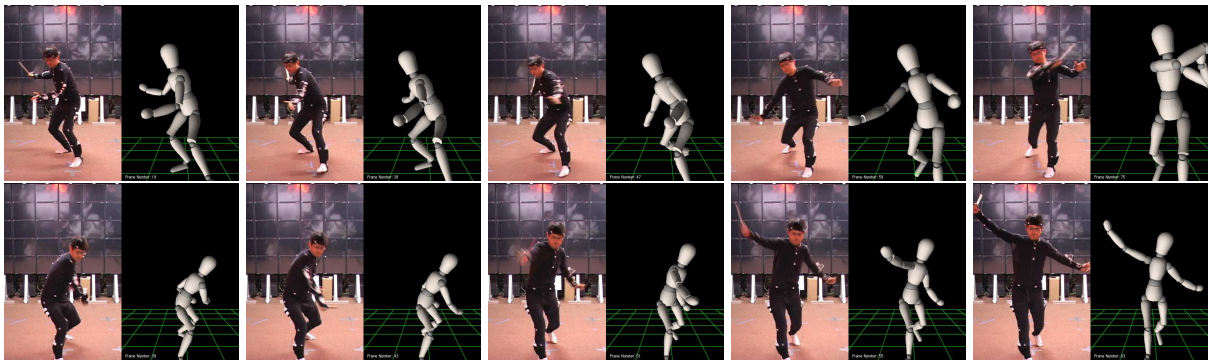


Figure 1.3: Result of motion synthesis using interpolation and LLE. Two different actions (one in each row) synthesized by our system. Each frame shows on the left side the actual pose and on the right side the synthesized pose.

four different sports actions (one in each row) synthesized by our system using interpolation and a linear learning model. Rows from top to bottom show a series of poses of a tennis forehand, a tennis backhand, a basketball shot and a karate middle block respectively. Each frame shows the actual poses on the left side and the synthesized poses on the right side.

Figure 1.3 shows two different sport actions (one in each row) synthesized by our system using interpolation and a nonlinear learning model. The first row is a tennis forehand shot, the second row is a tennis backhand shot. Each frame shows the actual poses on the left side and the synthesized poses on the right side. Figure 1.4 shows two different actions (one in each row) synthesized by our system using optimization.

## 1.5 Contribution

In this work we demonstrate that a performance animation framework can be built by using low-cost accelerometers to control the animation of 3D avatar. Various techniques are used for model learning and motion synthesis. The performance of these techniques is also analyzed.

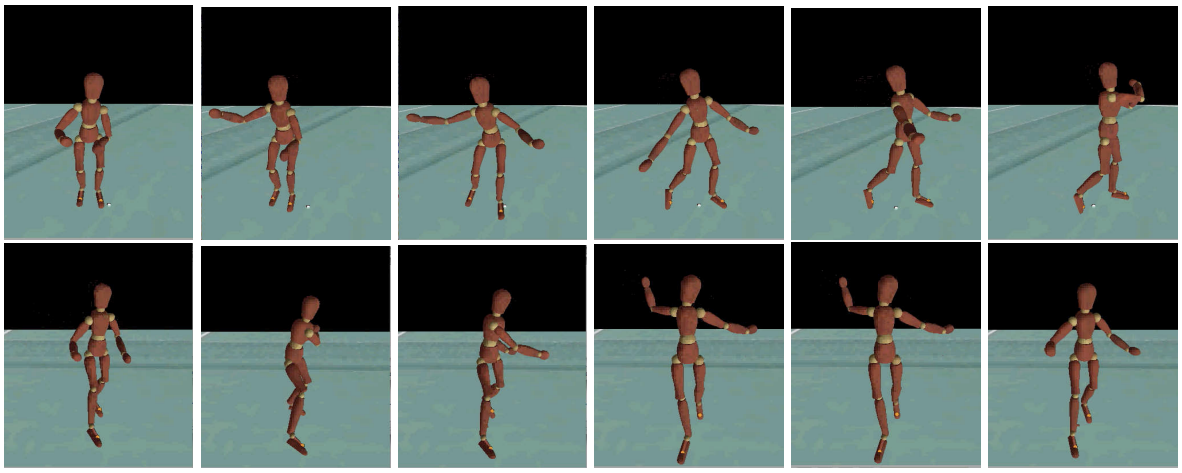


Figure 1.4: Result of motion synthesis using optimization. Two different actions (one in each row) synthesized by our system. The first row is tennis forehand, the second row is tennis backhand.

This thesis is an integration of my previous papers [45, 46]. First, in model learning phase, we investigate two different dimensionality reduction techniques, Principal Component Analysis (PCA) and Locally Linear Embedding. The accuracy of the two techniques are compared to the ground truth motion which is simultaneously captured by an optical motion capture system. We demonstrate that LLE is a more appropriate technique for full body motion data. Second, in the motion synthesis phase, we investigate two different techniques for motion synthesis: interpolation and optimization. The accuracy of both methods is equally good and thus have different performance advantages. Interpolation is very fast, especially using a local linear model. However, the synthesized motion is not smooth because motion is synthesized frame by frame. Optimization helps synthesize smoother motion by using temporal information. But the computational cost of optimization is high which makes it infeasible for interactive applications.

The main contributions of the thesis can be summarized as follows:

- We present a nonlinear manifold learning model for exacting the key features of high dimensional motion data and for speeding up the motion synthesis.
- We implement an optimization method for motion synthesis using an intuitive control signal

and demonstrate the advantage over interpolation in term of smoothness.

- We demonstrate the accuracy of our framework by applying it to several complex motion sequences for tennis and basketball.

## 1.6 Thesis outline

The remainder of the thesis is organized as follows. Chapter 2 provides the background and describes the related work in this area. Chapter 3 explains the system architecture while Chapters 4, 5 and 6 provide the detailed descriptions of three phases in our framework. Chapter 7 shows the results to demonstrate the accuracy of our approach. Results from different methods, such as PCA and LLE, or interpolation and optimization, are compared. Chapter 8 summarizes the paper and discusses the contribution and limitations.

# Chapter 2

# Related Work

Our proposed work can be categorized into the research of performance capture for human motion, where human performance can be recorded by sensors and re-generated in the form of 3D animated avatars. In this section, we will describe the existing work in this area, and then explain why we are motivated to use accelerometer-based, data-driven approach for this research. We will focus on reviewing interactive control for performance capture and data-driven approaches, respectively. Related work involving nonlinear dimensionality reduction techniques for motion synthesis is briefly discussed in the last part of this chapter.

## 2.1 Interactive Control of Performance Capture for Human Motion

There exist a variety of performance capture systems for animation of human motion, which are widely used for producing animated films for education, training, sports, and video games. Depending upon the technique used, current systems can be classified into two groups: optical sys-

tems and non-optical systems. This section gives a review of both systems because our approach involves both of them.

**Optical systems**

Optical systems utilize image data captured from several cameras that either track special markers attached to a subject, or recognize surface features identified dynamically for each particular subject. Marker-based optical systems can generate high-fidelity animation data with all the subtle motion detail. These systems perform best in the applications that mostly play back the original data, e.g., animated movies. However, most popular optical motion capture systems, such as VICON or Motion Analysis, are too costly to use for interactive motion capture and motion control.

In order to lower the monetary cost of performance capture, researchers explored the possibility of using standard video cameras, compared with expensive professional cameras. Chai etc. [9] proposed a vision-based system that synthesized human motion online. His system required only two inexpensive video cameras and a few markers attached to a body. Using the captured markers' location as a low-dimensional control signal, the system can synthesize a wide variety of human motion by querying a high quality motion capture database. Liu et al. [26] developed a data-driven model to predict all motion capture markers using only a subset of the markers. However, they both suffer from the occlusion problems characteristic of vision based tracking systems.

Recently, Aguiar et al. [1] presented a markerless approach for video-based performance capture. This approach requires 8 high resolution cameras and produces feature-rich human motion, comprised of high-quality geometry, life-like motion data and surface texture of recorded subjects. This multi-view stereo approach supports people wearing a wide variety of everyday apparel who are performing rapid motion. Due to the feature-rich output format, this method supplements and exceeds the capabilities of marker-based optical capturing systems. However, it is still a high cost

approach and is limited by the motion capture environment.

**Non-optical systems**

Non-optical systems use acoustic, inertial or magnetic sensors or combinations of these sensors, which are usually low-cost. The sensors signals provide a digital representation of motion that provides control signals for motion synthesis. Oore et al. [29] presented a novel interface with six degrees-of-freedom to interactively drive the character locomotion. Recently, Vlasic et al. [43] developed a hybrid motion capture system by combining accelerometer, inertial and acoustic sensors. All the sensors are sewn into a wearable suit which removes the traditional restriction of motion capture environments. However, the monetary cost is still high, and it is not an interactive system because it requires post-processing time.

More recently, Shiratori et al. [39] designed and tested three user interfaces to physically control a biped character using 3D accelerometers in Nintendo Wiimote. They showed that accelerometers provide a better and easier user control compared with joysticks. Slyper et al. [40] presented a performance animation system using five accelerometers sewn into a shirt. The sensor signals are received by a computer through wires and continuously matched against a prerecorded motion capture database. The scheme used in the system is simple because recorded motions simply playback by searching for the closest match; however, their system is not capable of generating new motion sequences, which restricts its application.

## 2.2 Data-Driven Full Body Animation

Although marker-based motion capture systems can generate high-fidelity animation data with subtle human motion detail, these systems reach the best performance only in the applications

that mostly play back the original data. Editing the original motion capture data usually results in non-realistic animations.

Many researchers have developed techniques for motion capture data reuse by editing and synthesizing new motion sequences from motion capture data. Three different approaches have been used: reordering motion clips using a motion graph [3, 4, 22, 49], interpolating motion to generate a new motion sequence [15, 21, 31, 44], and by optimizing a function to synthesize a new motion sequence [8, 9, 36].

In this section, we first introduce the basic idea of a motion graph because a number of recent articles have proposed methods based on the motion graph. In the following two subsections, previous work using interpolation and optimization for motion synthesis is discussed.

### 2.2.1 Basic Motion Graph Approach

Motion graph methods were proposed in 2002 [3, 22]. Since then, motion graphs have became one of the most popular methods for full body animation [4, 21] and facial animation [49]. The basic idea of a motion graph approach is to organize a motion database as a graph and to reassemble the graph components to form a new graph path which corresponds to a new motion. We review three papers that capture the fundamental work on motion graphs.

Kovar etc. [22]were among the first to propose the motion graph to control and generate motion. With the graph representation of motion capture data, the approach can be used to synthesize different types of human motion with smooth transitions. The generated motion should satisfy a set of user's constraints, such as an arbitrary path specified by a user. Based on the prerecorded database, Kovar's approach allows automatically constructing a graph that captures connections among different pieces of motion in the database. A branch and bound algorithm is then applied in this graph to search for a motion path that satisfies user constraints.

Arikan [3] proposed a similar approach at almost the same time. The basic differences are a different representation of motion graph and a different graph search algorithm. In Arikan's approach, nodes of the graph are individual motion sequences and edges are the transitions from frame to frame, while in Kovar's approach, all edges correspond to clips of motion and nodes are choice points connecting these clips. Because of the different graph representation, branch and bound algorithms are not suitable. Instead, a randomized search algorithm is used.

Arikan et al. [4] proposed a motion synthesis framework that allow users to control the motion by painting a timeline. The user would specify what types of behavior should occur during the motion. The synthesized motion should satisfy the user constraints, including annotation constraints, frame constraints and position constraints. The motion also should be continuous with smooth and natural transitions between different behaviors. As a data-driven approach, however, the approach suffers from the following limitations. First, the approach is not able to synthesize single behaviors and composite behaviors if they are not in the prerecorded database. The approach assembles motion clips from a database. The boundaries of motion blocks are not naturally connected, thus the search algorithm can put any motion clip after another. To solve this problem, the author proposed a post-processing phase to make the final motion smooth. The scheme also reduce the interactivity of the system.

Motion graph approaches can preserve all the subtle details in the original motion data; however, the synthesized motion is restricted to existing motion database because the approaches basically select, not modify existing motion data.

### 2.2.2 Motion Synthesis using Interpolation

Using interpolation, the synthesized motion may have spatial/temporal variations that are not present in the motion capture database. Interpolation allows novel and natural motion synthesis

to satisfy the constraints specified by the users. Guo and Roberge [15] and Wiley and Hahn [44] presented linear interpolation techniques for motion synthesis, while Rose et al. [31] use radial basis functions (RBFs) to interpolate motions located in an irregular parametric space.

Kovar et al. [21] presented a novel data structure called a *registration curve* to automatically blend input motions into a new realistic motion. By constructing registration curves for input motions, the algorithm can determine the relationships involving the timing, local coordinate frame and constraints of the input motions. Compared with linear interpolation, registration curves can provide more realistic interpolation via schemes, such as coordinate frame alignment, timing and constraints matches. The data structure also provides a shared interface for common blending operations, such as transmission, interpolation and continuous control. However, the approach suffers from a problem when the input motions have dissimilar poses. Moreover, since the approach doesn't employ physical constraints, the blending may suffer from a foot sliding problem where the supporting foot of character skeleton may slide when the motion is simply blended.

Safonova and Hodgins [35] provided an improved method based upon motion graphs. By applying a linear interpolation of two time-scaled paths in a motion graph, novel and natural motions are generated to satisfy the constraints specified by the users. The interpolated motion graph takes advantage of the natural transitions provided by the motion graph and physically realistic motion provided by the interpolation. However, searching in an interpolated motion graph is time-intensive, which makes it unsuitable for interactive applications.

### 2.2.3 Motion Synthesis using Optimization

Chai etc. [9] proposed a vision-based system that synthesize human motion online. The system requires only two inexpensive video cameras and a few of markers attached to a body. Using the captured markers location as low-dimensional control signal, the system can synthesize a wide va-

riety of human motion by querying a high quality motion capture database. The main contribution is that only using low-cost motion capture equipment and a small set of markers, the system could synthesize high quality human motion in real time. The statistical model presented in the paper could fast handle a large motion database because of low-dimensional representation of data and the fast search algorithm. However, as a data-driven approach, the approach has the same problem that other data-driven approaches have. Without similar motion data in the database, the approach is impossible to synthesize realistic human motion.

Safonova et al. [36] utilize an existing motion capture database and proposed a optimization algorithm in low-dimensional spaces to synthesize human motion. PCA is used on motion with similar behavior to reduce data dimensionality thus the subject behavior has to be specific. The limitation results from dimensionality reduction technique used because for a global linear method, such as PCA, it is hard to model a heterogeneous database which is possibly nonlinear. Global linear models might be appropriate for Safonova's application which is synthesis of motion without a continuous control signals, however, it is not a best choice for us. Similarly, Carvalho et al. [8] presented a low-dimensional optimization framework that used a Prioritized Inverse Kinematics (PIK) strategy. Two local motion models, i.e. PCA and Probabilistic PCA, are used to reduce dimensionality and their performances are compared for solving the optimization problem. However, the approach is only limited to specific behavior, i.e., golf swing, suffering from the model problem.

## 2.3 Nonlinear Dimensionality Reduction for Motion Synthesis

In this section, we will explain why we use nonlinear manifold learning methods for our approach. PCA or other linear reduction techniques (see Section 5.1 for more detail) are commonly used in the industry and research community of computer animation. However, it is difficult for linear di-

mensionality reduction techniques to model a heterogeneous database which is possibly nonlinear.

Since late 1990s, a number of work has been devoted to develop more complex models to reduce data dimensionality in a nonlinear way. The nonlinear methods are designed to explore the nonlinear structure for high dimensional data, considering the geometric factors instead of simply projecting data by linear transformation. Isomap [41] is a nonlinear generalization of MDS [10]. Isomap is designed to preserve the geodesic distance between pairs of multivariate data point, instead of simply taking Euclidean distance. The geodesic distance can present the distances along the manifold. Roweis et al. [32] and Saul et al. [37] proposed LLE, an unsupervised learning algorithm that calculates low dimensional embedding with topology preserving of neighborhood relationship in high dimensional data. LLE is capable of discovering nonlinear structure in high dimensional data by optimally preserving the local configurations of nearest neighbors. The advantage of LLE over linear dimensionality reduction technique (e.g. PCA and MDS) is that LLE can correctly detect the nonlinear structure and project the multivariate data into a single global coordinate system of low dimension.

Yeasin et al. [47] discussed the performances of several linear and nonlinear dimensionality reduction techniques in classifying universal facial expressions, i.e., PCA, Non-negative Matrix Factorization (NMF) and LLE. Their results show LLE has highest recognition accuracy.

Elgammal et al. [12] employed LLE to find the low dimensional embeddings of silhouette manifold. Given sequences of silhouette from monocular uncalibrated camera, a sequence of human 3D poses are produced by RBF interpolations from the silhouette manifold to 3D pose in body joint space. Likewise, Jaeggli et al. [17] proposed a body pose estimation system using video sequence as input. The pose is synthesized from a statical model of 3D pose, dynamics, activity transition and silhouette using sparse kernel regressors. Both of the approaches are offline. Our approach is partly similar to Elgammal's, however, our goal is different. We focus on real-time animation driven by accelerometer sensors.

# Chapter 3

# Motion Estimation Framework

We propose a data-driven approach that uses a small set of low-cost 3D accelerometer sensors for full-body motion synthesis. The challenge of our approach is that the sensor signal provides a limited number of degree of freedom which do not adequately constrain all joint angles of a full-body human model. The challenge of this thesis is how to use the low-dimensional sensor signal to control the high-dimensional human motion.

To fill up the gap between the low-dimensional control signal and the high-dimensional output, We utilize a data-driven approach with a prerecorded database of human motion to supplement insufficient information provided by the sensor data. Given a new control signal, a high-quality motion is synthesized using a piecewise linear model learned from the prerecorded database. In our approach, there are three major parts: data collection, off-line model learning and on-line motion synthesis. The following three chapters explain each part individually. Figure 3.1 shows the overview of our system.

**Data collection (Chapter 4):** We first perform a series of off-line motion capture sessions simultaneously using an optical motion capture system (VICON) and accelerometer sensors (Wii

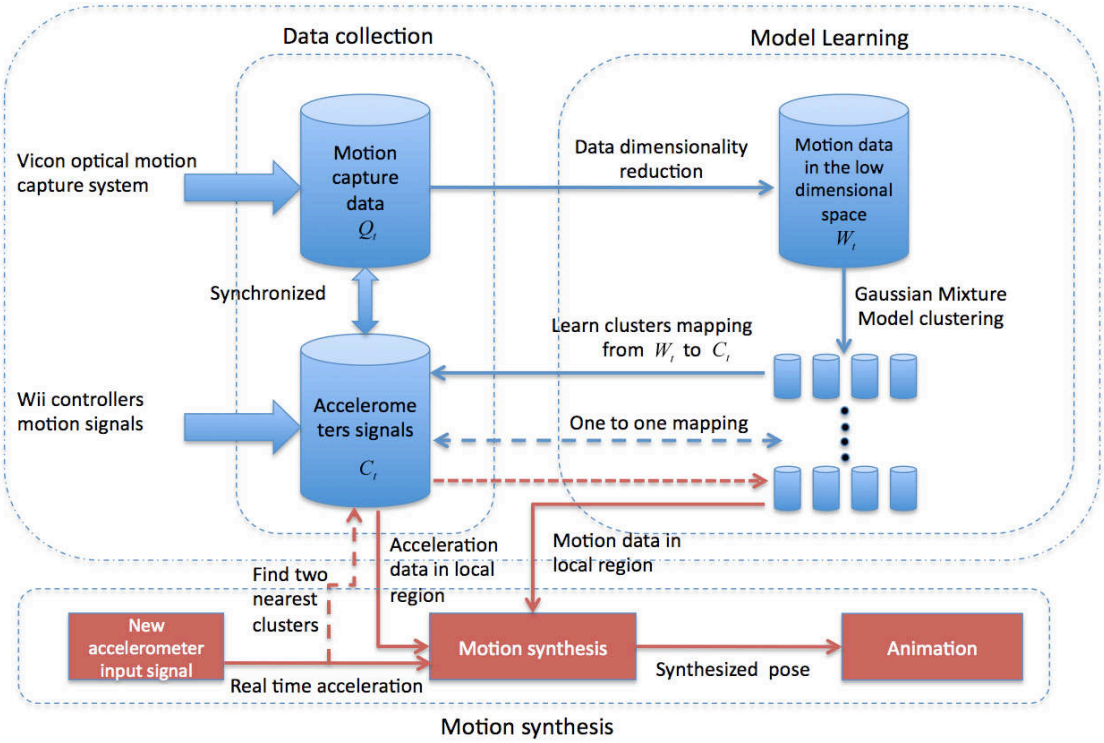



Figure 3.1: System overview.

controllers). Both motion capture data $Q$ and sensor data $C$ are pre-processed to reduce noise. We then synchronize the motion data with the sensor data in order to get a precise frame-to-frame mapping, i.e. $Q_t <=> C_t$ for every $t = 1...N$, where $t$ is the time frame and $N$ is the total number of frames in $Q$ and $C$. All data is then stored in a database for motion synthesis.

**Off-line Model learning (Chapter 5):** We propose to learn a piecewise linear model from the existing high quality motion database. Our model learning consists of three steps: data dimensionality reduction, data clustering and mapping, and piecewise local model building. First, data dimensionality reduction techniques are used to represented human motion data and sensor data in the low-dimensional space. The low-dimensional representation reduces the computational time required for the following phase of motion synthesis. More importantly, the low-dimensional representation discloses the underlying structure in high-dimensional data, which is helpful for data

clustering algorithm obtaining a suitable partition. Second, a data clustering technique is to partition the human motion data into several clusters in low-dimensional space. The motion data within one cluster shares high similarity, which makes local linear model building possible. The final step of model learning is to pre-compute a series of local models in the low-dimensional space. The quality of the local models are important to realism and smoothness of synthesized motion.

In Chapter 5, two types of piecewise models, i.e., *Radial Basis Function (RBF) interpolation model* and *multivariate Gaussian model*, are presented respectively. They are followed by two different motion synthesis methods described in Chapter 6.

**Online Motion Synthesis (Chapter 6):** During the motion synthesis phase, the user performs actions using only 3D acceleration sensors attached to the body. Using this sensor data as in input, we synthesize high quality motion data frame by frame. Here we use two different strategies, i.e., interpolation and optimization for motion synthesis. Both strategies has their own advantages as well as limitations, which are compared and analyzed in Chapters 6 and 7.

Interpolation: Motion estimation with interpolation relys on the RBFs interpolation model built in the previous phase. Given new value of control signal $\tilde{c}_t$ at time frame $t$, the motion sequences in the RBFs model can be interpolated to compute a new motion frame $\tilde{q}_t$ that matches the specified control signal. Technically, for each frame of the input sensor data $\tilde{c}_t$, we apply the RBF interpolation function of the cluster associated with the input data. Interpolation is fast and easy to implement, however, the problem is the result is not that smooth because of lack of temporal knowledge [46].

Optimization: Motion estimation with optimization depends on the multivariate Gaussian model. For each frame of the input sensor data $\tilde{c}_t$, we optimize an objective function to get an optimal motion frame $\tilde{q}_t$ that satisfies a variety of constraints. Our objective function uses three terms: model, control and smoothness.

# Chapter 4

# Data Collection and Representation

In this chapter we describe data collection in our system to capture a high quality database, which is important to the quality of the synthesized motion. Our data collection process includes data capture, synchronization and pre-processing, each of which is described in this chapter respectively. Explanation of the symbols and terminology used throughout this thesis are also presented.

## 4.1 Data Capture

We perform an off-line motion capture session to create a motion database which consists of two types of synchronized data. One is high quality motion capture data, acquired using a Vicon optical motion capture system. The other type is accelerometer sensor data received from Wii controllers.

To capture human motion, we use a system with 8 Vicon MX series cameras at a frame rate of 60 frames per second. Simultaneously, accelerometer sensor data are captured using eight low cost 3D accelerometers embedded in Wii controllers (e.g. four Wii Remote and four Wii Nunchuks) with a range of $\pm 3g$. The sensor data is transmitted through a bluetooth interface at a frame rate of

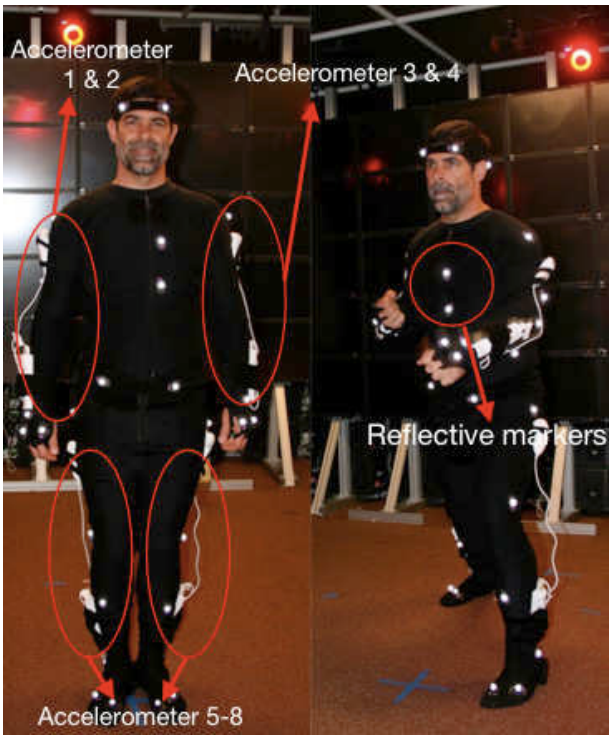



Figure 4.1: Data collection: an optical motion capture system and a 3D acceleration sensor based data acquisition system are used in parallel. There are 45 retro-reflective markers and eight sensors (four Wii$^{TM}$Nintendo controllers) attached to the performer.

up to 100 frames per second. Compared with professional motion capture systems, accelerometer sensors are cheap and easy to set up. Moreover, they doesn't suffer from occlusion problem like a vision based system,

Figure 4.1 shows the studio for motion capture, where 45 retro-reflective markers and eight accelerometers are attached to the performer's body. The sensors are attached to the arms and legs since they provide most of the movements for a majority of human actions. The signals of all the 3D acceleration sensors are transmitted to a computer, then converted into sensor frames.

After captured, sensors frames are synchronized with motion data frames because two types of data have inconsistent frame-rates (see Section 4.2). Pre-processing is performed to reduce noise that are resulted from the wireless environment (see Section 4.3).

## 4.2 Data Synchronization

The motion data and the sensor data are acquired simultaneously; however, they have inconsistent frame-rates, which makes them not suitable for direct use. In this section, both types of data are synchronized by building an one-to-one correspondence between the motion data and the sensor data from the optical motion capture system and all the accelerometers, respectively. When data is being captured, the sensors transmit signal independently of each other in the wireless medium. Moreover, data received from different sensors is at variable frame-rates owing to packet loss in the wireless environment, in contrast to the motion data at a consistent frame-rate. A simple idea of data synchronization is that the sensors should be synchronized at a constant frame-rate of 60Hz to match the frame rate of the motion capture system. Detailed solution is given in our previous paper [46].

## 4.3 Data Pre-Processing

Before storing the captured data into the database, we remove noise in both the the optical motion capture data (i.e. motion data) and sensor data. Noise reduction is crucial for the quality of the output data and this pre-processed data is used later in model learning (see Chapter 5) and motion synthesis(see Chapter 6). For the motion data, we use quaternions to represent joint rotation so that congruent angles (e.g. $0^\circ$ and $360^\circ$) are represented using the same numerical value. Noise in the motion data due to marker occlusion (e.g. missing markers, jittery animation) is removed in the post-processing step of the data capture stage.

Noise in the sensor data mainly results from the wireless environment. To attain high resolution accelerator data, we use high-bandwidth Bluetooth receivers which are very sensitive but usually receive some noise from the wireless environment. Not surprisingly, there exist a few arbitrary

Table 4.1: Motion types, frame numbers and duration in the database.

| **Actions** | **Frame number** | **Duration** |
|---|---|---|
| tennis forehand | 1121 | 18.7s |
| tennis backhand | 1004 | 16.7s |
| basketball shot | 1300 | 21.7s |
| golf swing | 1201 | 20.0s |
| karate middle block | 456 | 7.6s |
| Total | 5082 | 84.7s |

frames the values of which are beyond the value range we expect from the sensors. By using a low-pass filter, these values are automatically detected and replaced by quantities that are estimated from the neighboring data using least square function.

## 4.4 Data Representation

In the data collection process, we capture a high quality database which contains five different types of full-body actions: tennis forehand, tennis backhand, basketball shot, golf swing and karate middle block. The subjects perform each action several times ( from four to eight depending on the action). To collect a dense database, the subjects perform one action with slightly different styles. For example, tennis forehand action has strokes at a high point, strokes at a middle point and strokes at a low point. Table 4.1 shows the frame number and duration of each action.

There are $N$ frames of data in the database, which can be represented as

$$\{(\mathbf{c}_t, \mathbf{q}_t)|t = 1, \ldots, N\},$$

where $\mathbf{c}_t$ is a 24 dimensional frame of sensor data representing the acceleration signal on the body

at the time frame $t$. $\mathbf{c}_t$ is received by a total of eight sensors, each of which provides 3 dimensional acceleration signal. $\mathbf{q}_t$ is a 88 dimensional frame of optical motion capture data and it represents a pose at the time frame $t$. $\mathbf{q}_t$ contains the necessary information depicting a pose, including a global root location, a global orientation and local joint rotations. Local joint rotations are represented in the form of quaternion format. In addition, $N = 5082$ is the total number of frames in our database. Note that there is one-to-one correspondence between $\mathbf{c}_t$ and $\mathbf{q}_t$.

# Chapter 5

# Model Learning

Acquired in the phase of data collection, our large and heterogeneous database includes a variety of motion data. However, there exist quite a few difficulties in directly using such a database for motion synthesis. First, the database contains a variety of nonlinear complex structure. Moreover, the acceleration signal from a small set of accelerometers sensors attached on a user is low-dimensional while the motion we want to synthesize is high-dimensional. Thus the acceleration signal does not provide sufficient controlling information to adequately constrain the joint angles space of a full-body human model. Furthermore, the acceleration signal provided by accelerometers is usually noisy with many outliers because of the accuracy limitation of electronic devices. Using a noise control signal can easily cause a jitter problem in the generated animation.

Several data-driven approaches are under consideration for our problem. A motion playback approach may be the first that comes to mind. Not surprisingly, some research efforts have been made. Slyper et al. [40] use the accelerometer sensor signal to query a structured database of human motion, and select the closest matching sequences of motion from the motion database. Similarity, Kumar [23] utilizes a motion graph approach for motion synthesis. A common weakness of motion playback approaches is the incapability of synthesizing completely new motion which is

not included in the existing database. A model-based approach is a popular alternative for motion synthesis. Essentially a mathematical model is learned from the existing data and developed to recognize the fundamental features. However, not every model is suitable for our problem, e.g. a global model. The heterogeneous and highly non-linear property of our database is very difficult to represent using a global model. In addition, a global model is computationally expensive.

To address the issues mentioned above, we propose to learn a piecewise linear model from the existing high quality motion database. A piecewise linear model can exploit effective techniques (i.e. dimensionality reduction techniques) to uncover the non-linear structure of motion database. With regard to computational cost, a piecewise model can reduce the cost by constraining the solution space. While used in the following motion synthesis phase, the local models we present are capable of constraining the solution space and also restricting the jitter problem. In this chapter, two types of piecewise models, i.e., radial basis function interpolation model and multivariate Gaussian model, are presented that are utilized in two different motion synthesis methods described in Chapter 6.

Our problem is stated as follows: Given the human motion data $\{(\mathbf{q}_t)|t = 1,\ldots,N\}$ in the data set of motion capture $Q$ and the sensor data $\{(\mathbf{c}_t)|t = 1,\ldots,N\}$ in the data set of sensor signal $C$, our piecewise local model builds a mapping function from $C$ space to $Q$ space (see Section 5.3.1 ) or measures a-priori probability of a new motion frame $\tilde{\mathbf{q}}_t$ in its local region (see Section 5.3.2).

Our solution for model learning consists of three steps: data dimensionality reduction, data clustering and mapping, and piecewise local model building. First, data dimensionality reduction techniques are used to represented human motion data and sensor data in the low-dimensional space. The low-dimensional representation reduces the computational time required for motion synthesis described in Chapter 6. More importantly, the low-dimensional representation discloses the underlying structure in high-dimensional data, which is helpful to data clustering algorithm

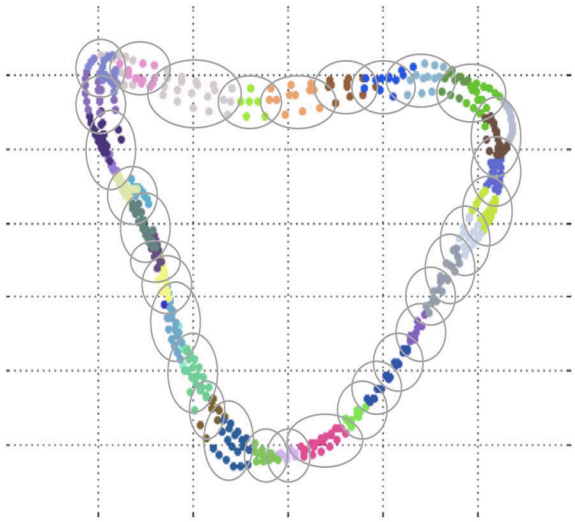


Figure 5.1: An illustrative diagram for sample motion data. Each color denotes one cluster.

for obtaining a suitable partition (see Fig. 5.1). Second, a data clustering technique partitions the human motion data into several clusters in low-dimensional space. The motion data within one cluster shares high similarity, which makes local linear model building possible. The final step of model learning is to pre-compute a series of local models in the low-dimensional space. The quality of the local models is important to realism and smoothness of synthesized motion.

Figure 5.1 is an illustrative diagram for sample motion data. The motion data forms a cycle in a 2D space. The 2D representation of motion data, which retains a majority of the original information, is difficult to analyze by traditional statistical methods. Not surprisingly, there exist a variety of nonlinear manifolds in a higher dimensional space in the original motion data. The key idea of our approach is to pre-compute piecewise low-dimensional linear models which are used during the online motion synthesis phase.

In this chapter, we describe model learning process, including dimensionality reduction, database clustering and mapping, and piecewise local model building, respectively.

## 5.1 Data Dimensionality Reduction

### 5.1.1 Introduction to Dimensionality Reduction Techniques

In past decades, advances in motion capture data collection have led to an information overload in computer animation. There are many free motion capture databases available on the Internet. For instance, the CMU graphics lab motion capture database is one popular database in the computer animation research community. The dramatic increase in motion data brings researchers a challenge. The increasing number of captured frames and variables associated with each frames are difficult to handle. In particular, the dimension of human motion is usually large (more than 60) because many subtle details have be included in human motion.

One of the problems associated with high-dimensional data is that many traditional statistical methods, such as statistical clustering methods, fail to analyze the data. Additionally, the high-dimensional data is very hard to visualize and understand. It would be interesting to reduce the the dimensionality of the original data while retaining its original attributes. What makes dimensionality reduction possible is that not all the dimensions in the high-dimensional data are that ”important” for understanding the principal phenomena. For instance, a number of degrees of freedom are highly correlated in human motion, and only a few degrees of freedom are really ”important” for capturing the body movement.

In the mathematical terms, data dimensionality reduction can be stated as follows [13]: given a $p$-dimensional random variable $x = (x_1, ..., x_p)^T$, find a lower dimensional representation $y$, where $y = (y_1, ....y_k)^T$ with $k < p$. The new representation with $k$ components can preserve the majority of the information in the original data while reducing data dimension according to some different criteria.

Based upon different criteria, dimensionality reduction techniques are classified into two major

types: linear and non-linear. Popular examples of linear dimensionality reduction techniques are principle component analysis (PCA), independent component analysis (ICA), multidimensional scaling (MDS) and factor analysis (FA). Popular examples of non-linear dimensionality reduction techniques are Isomap, Isotop and locally linear embedding (LLE). Fodor [13] give a nice review of dimensionality reduction techniques while Lee and Verleysen [25] discuss existing and advanced methods for reducing data dimensionality nonlinearly.

### 5.1.2 Linear Dimensionality Reduction

In essence, linear dimensionality reduction techniques attempt to find $k < p$ of the new variables to represent the original data. Each of the new variable $y_i$, where $i = 1, ..., k$, is the linear combination of the original variables.

$$y_i = w_{i,1}x_1 + w_{i,2}x_2 + ... + w_{i,p}x_p \ . \tag{5.1}$$

For the original data set $X$ of dimension $p \times n$, the new representation $Y = WX$, where Y is $k \times n$ and W is a linear transformation matrix with dimension $k \times p$. Linear dimensionality reduction techniques are easy to understand and implement. The differences among the various linear techniques are the means of finding the transformation matrix $W$.

Since introduced in 1986, Principle Component Analysis has remained the best linear dimensionality reduction technique in the mean-square error sense [16, 20]. PCA is commonly used in computer animation research because of its simplicity and efficiency. The basic idea of PCA is to approximate the data and find a set of new bases by minimizing the mean square error. Given a data set, PCA first identifies a set of the most meaningful bases to represent the original data set; PCA then projects the original data set onto the new bases for the low-dimensional representation. Here we refer to a new meaningful basis as a principle component. All of the new bases are orthogonal. The value of an associated variance indicates the importance of a principal component. The top

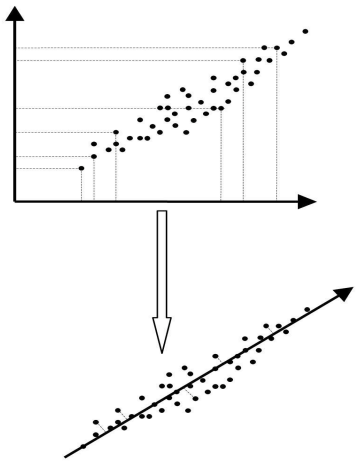


Figure 5.2: Applying PCA to 2D sample data.

principal component associated with large variances represents important hidden structure while those with lower variances represents noise in the original data set. Figure 5.2 shows an example of applying PCA to 2D sample data.

There are several algebraic solutions to PCA, including eigenvector decomposition and singular value decomposition (SVD). Here we describe the eigenvector decomposition which consists of the following steps:

1. Subtract the mean $\bar{X}$ from original data set $X$.

2. Compute the covariance matrix $\sum_{p\times p} = \frac{1}{n}(X-\bar{X})(X-\bar{X})^T$. The larger values in $\sum$ reflect larger correlations between two variables. In the diagonal terms, large values mean more importance of the corresponding values. In the off-diagonal terms, large values present large correlations between two variables which means high redundancy of the data.

3. Compute the eigenvectors of $\sum_{p\times p}$. Based upon decomposition theorems, $\sum = U\Lambda U^T$ where $\Lambda = diag(\lambda_1,...,\lambda_p)$ is the diagonal matrix of the ordered eigenvalues $\lambda_1 > \lambda_2 > .... > \lambda_p$

and $U$ is a $p \times p$ orthogonal matrix containing the eigenvectors.

4. Compute the new representation (principal components) $Y$ by $Y = U^T X$. We usually use parts of $U$ including biggest $k$ eigenvectors corresponding to largest $k$ eigenvalues.

**Applying PCA to the Existing Database**

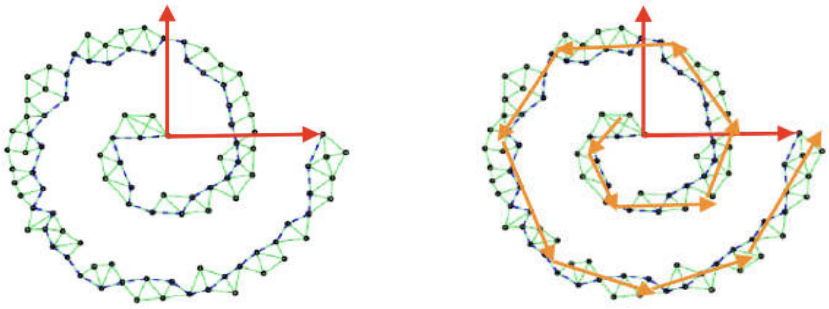
For our motion capture data set $Q$, where $Q = \{\mathbf{q}_i | i = 1, 2, .., N\}$ ($N$ is the total number of frames), we first apply PCA to reduce the dimensionality of motion capture data $Q$. In our case we reduce the dimensionality from 72 to 7 (Before applying PCA, character orientation and empty degree of freedoms are removed so that the dimension drops from 88 to 72). PCA users have to specify the lower dimension. Here we select 7 as the lower dimension by considering the computational cost and accuracy. The top seven principal components can capture as high as 99% of the motion variance while dramatically reducing the computational cost of model learning and motion synthesis. The reduced dimension data $\mathbf{r}_i$ is produced by PCA using Equation 5.2:

$$\mathbf{r}_i = (\mathbf{q}_i - \bar{\mathbf{q}})A^{-1}, \tag{5.2}$$

where $\bar{\mathbf{q}}$ is the mean value, and $A$ is the transformation matrix built from the eigenvectors corresponding to the largest seven eigenvalues of the covariance matrix.

**Limitation of Linear Dimensionality Reduction Techniques**

When the data dimension is high, linear dimensionality reduction techniques, such as PCA, are very useful for removing a large number of useless dimensions. However, PCA and other linear techniques perform poorly when applied to the data with complex nonlinear structure. Figure 5.3(a) shows a set of 2D points with nonlinear structure. Traditional linear techniques poorly



(a) (b)

Figure 5.3: An example of 2D Swiss roll with non-linear structure. (a) Applying PCA to 2D swiss roll. A black point is connected to its neighboring points with green lines. The red arrow lines show two bases detected by PCA. (b) Applying piecewise PCA to 2D swiss roll. Orange arrow lines show a set of bases in each cluster detected by piecewise PCA.

support such a nonlinear underlying structure. No matter which direction the original data is projected to, much data variance would be lost, as shown in Fig. 5.3(a).

Piecewise linear techniques were proposed to solve the limitation of traditional linear techniques. Typically, piecewise linear techniques segment the data space into disjoint regions using a clustering algorithm. Traditional PCA is then applied to every cluster. Figure 5.3(b) shows piecewise techniques can partially disclose the nonlinear structure; however, the original data is not projected onto a single coordinate. Furthermore, the performance of piecewise techniques depends highly upon the number and the quality of clusters.

### 5.1.3 Nonlinear Dimensionality Reduction

To fully disclose a nonlinear structure, much work has been devoted to develop a more complex model for reducing data dimensionality in a nonlinear way since the late 1990s. These nonlinear methods take geometric factors into consideration, instead of simply projecting data by a linear transformation. Lee and Verleysen [25] classified nonlinear methods into two types: distance

preservation methods and topology preservation methods. Distance preservation methods attempt to preserve pairwise distances measured in the data set while topology preservation methods attempt to reproduce the data topology in the low-dimensional space. Distance preservation methods, such as Isomap and Kernel PCA, are easy to understand and compute. However, they are constrained by the distance function. In human motion data, using only a distance does not work well because the motion of distinguished behaviors may look close in term of distance. Here the topology captures the similarity of human motion more clearly than distance. Therefore, we employ *Locally Linear Embedding*, a nonlinear dimensionality reduction technique with topology preservation as described in the following subsection.

**Applying LLE to the Existing Database**

In our approach, the database consists of motions from various sports, such as tennis, basketball and golf, etc. Global linear models have difficulties handling these types of heterogeneous motion databases [8, 36, 46]. Nonlinear dimensionality reduction techniques with distance preservation, such as Isomap, have difficulties in interpreting the similarity of human motion data [38]. To circumvent such limitations, we use a nonlinear model in a reduced-dimension space with topology preservation. We use LLE to learn the nonlinear manifold in the high-dimensional motion data. Our result shows that the nonlinear manifold learning model has better performance than linear models (see Section 7.2).

LLE is initially introduced for the problem of nonlinear dimensionality reduction by Roweis et al. [32] and Saul et al. [37]. ”It is an unsupervised learning algorithm that computes low-dimensional embedding with neighborhood relationship preserving of high-dimensional data.” [32]. A key assumption is that for some number $K$, the $K$-neighbors of points are locally linear. By this assumption, LLE can be used to discover nonlinear structure in high-dimensional data by optimally preserving the local configurations of nearest neighbors. The advantage of LLE over linear

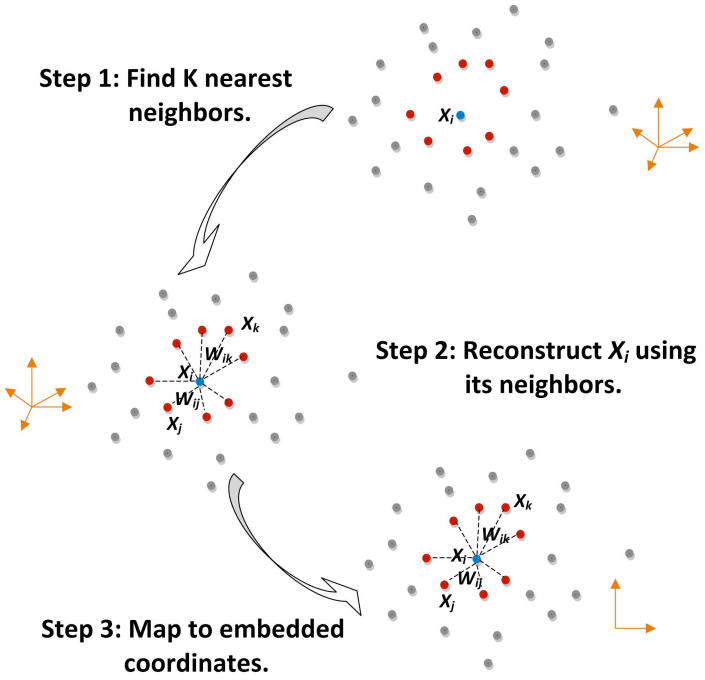



Figure 5.4: Steps of locally linear embedding [32].

dimensionality reduction techniques, such as PCA and MDS, is that LLE can correctly detect the nonlinear structure and project the multivariate data into a single global coordinate system of low dimension.

The basic idea of LLE is: given $N$ input vectors $\{x_1, x_2, \ldots, x_N\}, x_i \in R^d$, LLE calculates new vectors $\{y_1, y_2, \ldots, y_N\}, y_i \in R^m$, where $m \leq d$, such that an objective function $\phi(y_i)$ is minimized. Normally, LLE consists three steps illustrated in Fig. 5.4 :

- Select neighbours: find the $k$ nearest neighbors for each $x_i$.
- Reconstruct with linear weights: measure reconstruction error resulting from the approximation of each point by its neighbors and calculate the reconstruction weights $w_{ij}$ which minimize the error.
- Map to embedded coordinates: compute the low-embedding vectors $y_i$ by minimizing the

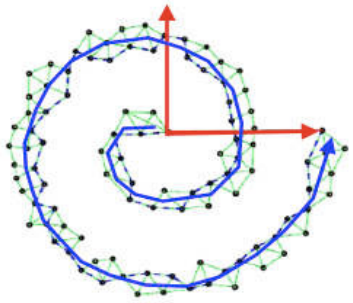


Figure 5.5: Applying LLE to 2D Swiss roll. Blue arrow lines show a basis detected by LLE.

embedding function with the reconstruction weights $w_{ij}$.

In the first stage, we find $k$ nearest neighbors for all $x_i \in R^d$ in the d-dimensional space. The Euclidean distance is used to measure the geodesic distance between two vectors.

In the second stage, we calculate reconstruction error by:

$$\varepsilon(W) = s\sum_{i=1}^{N} |x_i - \sum_{j=1}^{k} w_{ij}x_{ij}|^2, \tag{5.3}$$

where $x_{ij}|j = 1,\ldots,k$ denote the $k$ nearest neighbors of $x_i$, and $w_{ij}$ is the weight of the neighbor. Here, we should notice that $w_{ij} = 0$ when $x_j$ is not counted as the neighbor of $x_i$, and for all the neighbors of $x_i$, $\sum_{j=1}^{k} w_{ij} = 1$. As the design of LLE, $w_{ij}$ reflects the intrinsic geometric properties of the original data, and we can find a linear mapping to be a approximate representation of the data.

In the final stage, we are going to compute the embedding vectors $y_i$ of the original vector data $x_i$ in the low-dimentional embedding space. To preserve the local geometric properties of the original space, we minimize the following embedding cost function:

$$\phi(Y) = \sum_{i=1}^{N} |y_i - \sum_{j=1}^{k} w_{ij}y_{ij}|^2. \tag{5.4}$$

Table 5.1: Comparison of PCA and LLE in term of model assumption, complexity, advantage and limitation.

| **Comparison** | **PCA** | **LLE** |
|---|---|---|
| Assumption | global linearity | local linearity of neighbor points |
| Model Complexity | $O(D^2N)$ | $O(N^2P)$ per interation |
| Advantage | simple and robust for large data sets with linear structure | topology preservation and robust for data sets with nonlinear structure |
| Limitation | not good for nonlinear data | parameters setting affects the accuracy |

This embedding cost function is calculated based upon the previous locally linear reconstruction errors, and the weights $w_{ij}$ are fixed when optimizing $y_i$. In this procedure, the high-dimentional vector data $x_i$ is mapped one-to-one to the low-dimentional vector $y_i$. Thus we can apply a clustering algorithm more simply on this low-dimentional space than on the originally high-dimensional space.

**Applying LLE to 2D Sample Data**

Figure 5.5 shows that LLE can fully disclose the nonlinear structure in the data of the 2D Swiss roll. LLE is capable of unrolling the roll and projecting the original data onto a flat plain. The topology in the original data is preserved after applying LLE.

### 5.1.4 Discussion of PCA and LLE

Based upon the discussion above, Table 5.1 gives a comparison of two techniques in terms of model assumption, complexity, advantage and limitation. In model complexity [25], $N$ means the number of observations, $D$ is the value of the high dimension while $P$ is the value of the low dimension.

There are several limitations of LLE, and the biggest one is the parameter setting. It was reported

by Saul and Roweis [37] that finding good parameters is not so easy as claimed in [32]. In practice, we have to set two two parameters for LLE; one is the number of neighbor points $K$, and the other is the regularization factor $\Delta$. We repeatedly adjust two parameters until we find an optimum that has the least reconstruction error.

## 5.2 Database Clustering and Clusters Mapping

After a dimensionality reduction technique is used, the data dimensions are reduced and the underlying structure in the data has been disclosed. A clustering algorithm is employed to partition the database into small groups of data, in which a series of low-dimensional local models are computed (see Section 5.3). The quality and rationality of the clustering results are very important to model learning. This section is the bridge between the dimensionality reduction techniques described in Section 5.1 and the piecewise linear model building described in Section 5.3.

In this section, we start with a general introduction to data clustering methods in Section 5.2.1. In Section 5.2.2 we describe the *Gaussian Mixture Model* (GMM) for data clustering, and in Section 5.2.3 we will show how to apply GMM to our data. Finally we discuss clustering mapping from the human motion data to the sensor data.

### 5.2.1 Introduction to Data Clustering Methods

Data clustering is developed to classify data items into groups in term of similarity. Clustering problems are ubiquitous; thus clustering methods play an important role in many disciplines, such as statistics, computer science and economics. There are various types of clustering methods. Typically they are categorized into three types [19]: hierarchical clustering, partitional clustering and fuzzy clustering.

- **Hierarchical clustering** algorithms are popular in applications involving small data sets [19]. The most popular hierarchical clustering methods use an agglomerative approach, which starts with a initial number of clusters. Similar clusters are merged to form larger clusters until the desired number of clusters is obtained. There are different ways of measuring the similarity between a pair of clusters. In single link clustering, the distance between two clusters is characterized as the minimum distance between all pairs of members from two clusters. In complete link clustering, the distance between two clusters is characterized as the maximum distance between all pairs of members from two clusters.

- **Partitional clustering** algorithms directly partition the data by optimizing a criterion function instead of using a hierarchical clustering structure. The criterion function is defined either in a local region (i.e., in a subset of data) or at a global scale (i.e., in the entire data set). Compared to hierarchical clustering algorithms, partitional clustering algorithms are less expensive in applications having a large data set [19]. However, a common problem of partitional clustering algorithms is that the algorithms are very sensitive to the selection of the desired number of clusters, which has to be defined manually by the users. Popular partitional algorithms are *k*-means clustering [27], graph-theoretic clustering [48] and model-based clustering [18].

- **Fuzzy clustering** algorithms [5, 34] assign degree of membership in several clusters to every input vector. In fuzzy clustering, an input vector can be assigned to several clusters by means of proportions summed to 1. The clustering is not an explicit partition, which allowing some overlap between clusters; while in partitional clustering, an input vector belongs to one and only one cluster. This is one key advantage of fuzzy clustering. However, it is not easy to obtain the proportion values. Additionally, a clear presentation of clusters is required [19], which indicates the clusters have been labeled for certain meaning.

As discussed above, all three types of methods have their intrinsic advantages and disadvantages. A hierarchical algorithm is simple to implement but it is not suitable for large data set. Fuzzy clustering needs a clear description of clusters which is not known in the human motion data. Thus neither is a good option for our problem.

Our approach uses a mixture model for data clustering. Our human motion data is high dimensional and heterogeneous. It usually contains an unknown number of clusters, which meets the assumption of mixture model-based clustering methods. We use a mixture model to capture such ambiguities in the data and we assume that the densities meet a multivariate Gaussian distribution.

### 5.2.2 Model-based Clustering with Gaussian Mixture Model

#### Model-based Clustering (Single *vs* Mixture Model)

In addition to human motion data, there exists much data which contains an unknown number of densities and unobserved variables. A single model cannot adequately represent such data. For example, given the scores received by students in a midterm exam of a computer animation class, we may want to classify all the students in this class into different groups according to student type (e.g. engineer, artist), and then predict final exam scores for them. The score information is a vector specifying the scores a student received for every question in the midterm exam. We assume that different types of students may achieve success in different types of questions. Apparently this ”student type” is unknown but has an influence on the clustering and prediction results. The unknown information includes the number of student types and the number of students in each type. The number of types can be determined from the data [6]. However, a single model is incapable of modeling different types and is not able to partition the data in a reasonable way. By comparison, a mixture model can capture such ambiguous and complex data more precisely.

**Gaussian Mixture Model**

There are a number of mixture models. In our approach, we assume the data densities are under the Gaussian distribution the same as most of the preceding work [19], and we employ a simple model, i.e., GMM, to segment the motion capture data sets into a number of clusters.

Suppose we have a observation vector $x$, where $x$ can be one of $L$ possible types. A general mixture model is defined as follows [42]:

$$p(x|\theta) = \sum_{j=1}^{L} \pi_j P(x|j) \ , \tag{5.5}$$

where $L$ is the number of types and $P(x|j)$ is a conditional distribution. $\pi_j$ is a mixing weight satisfying $0 \leq \pi_j \leq 1$ and $\Sigma_{j=1}^{L} \pi_j = 1$, for $j = 1 \ldots L$.

GMM considers $P(X|j)$ as a multivariate Gaussian distribution, defined follows:

$$p(x|\theta) = \sum_{j=1}^{L} \pi_j \eta(x|\mu_j, \Sigma_j), \tag{5.6}$$

where $L$ is the number of clusters and $\theta = \{\pi_j, \mu_j, \Sigma_j\}$ are the model parameters. $\pi_j$ is a mixing weight satisfying $0 \leq \pi_j \leq 1$ and $\Sigma_{j=1}^{L} \pi_j = 1$. $\mu_j$ and $\Sigma_j$ denote the mean and covariance of the mixture cluster $j$. Every cluster has various model parameters $\theta_j$, which measure the data distribution in the specific cluster.

**Estimating Mixtures Using Expectation Maximization Algorithm**

The model-based clustering problem is to identify the number of mixture components (i.e. clusters) and to estimate the parameters of each components. In most cases, the expectation maximization (EM) algorithm is utilized to estimate the parameters in the data set. The basic EM procedure is

interpreted as follows [11]: the algorithm begins with an initial estimation of the parameters of Gaussian mixtures, i.e. $\theta^{(0)}$; then iteratively updates the clustering likelihood and the parameters $\theta$ until the achieved likelihood is below a user-defined threshold.

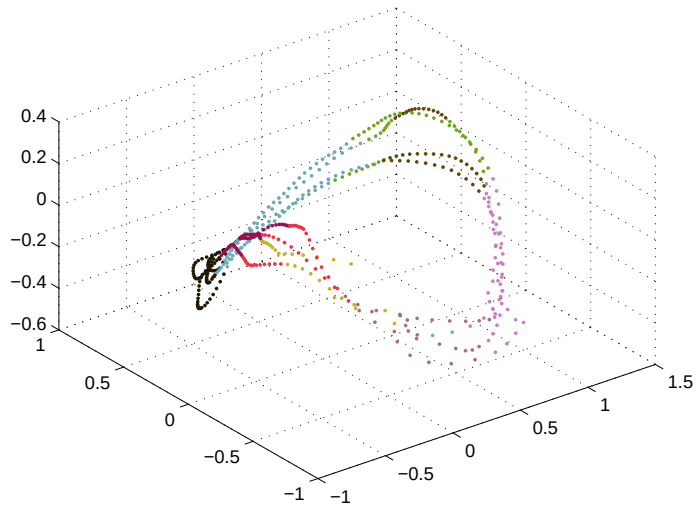
### 5.2.3 Applying Clustering to the Existing Data

As described above, we need to identify the number of clusters and the parameters for every cluster. We use the Cluster software package [6] to estimate the number of clusters $L$ which best fit the motion capture dataset $Q$. Cluster uses Bouman's unsupervised algorithm [6] to automatically estimate parameters of a GMM from sample data. By applying the EM algorithm together with an agglomerative clustering method, the estimation is based on the minimum description length (MDL) criteria. Once we know the value of $L$, we perform unsupervised classification on $Q$ and get the model parameter set $\theta$.

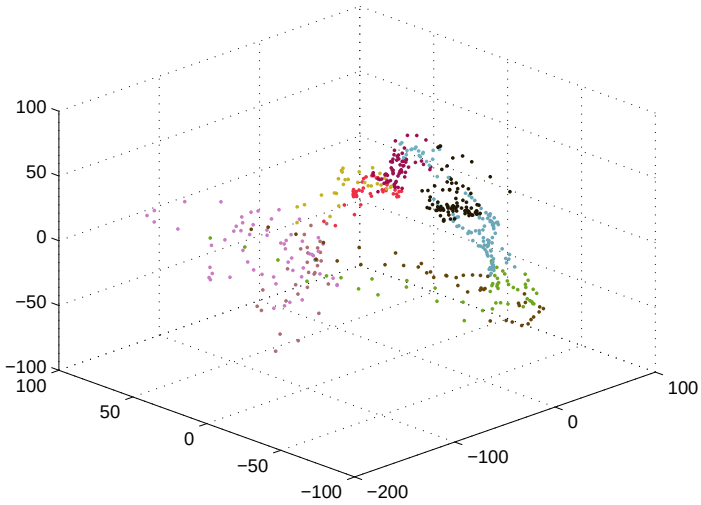
As in $K$-means clustering, GMM based clustering is unlikely to generates the same clustering results when the procedure is run several times because the EM algorithm usually converges to a locally optimal solution. It does not guarantee an optimal solution for every run. In practice, we run the procedure multiple times in order to provide a reasonable result.

### 5.2.4 Data Mapping

After clustering data set $Q$, we classify the sensor dataset $C = \{\mathbf{c}_t | t = 1, \ldots, N\}$ into $L$ clusters by a one-to-one mapping from $\mathbf{q}_t$ to $\mathbf{c}_t$ during data synchronization. In our database, all 5,082 frames are classified into 81 clusters. Figure 5.6 is an illustration of motion capture dataset clustering and sensor dataset clustering for four consecutive tennis forehand actions. Ideally, a data cluster should come from several short sequences of motion frames with high similarity. Such clusters




(a)




(b)

Figure 5.6: An example of clustering of motion and sensor data sets from four consecutive tennis forehands. A set of data points with the same color represent one cluster. The dataset are plotted in the top three components of PCA space of motion capture data (a) and sensor data (b), respectively.

have distinct features that are easily expressed by local models.

## 5.3 Piecewise Linear Model Building

The clustering algorithm using GMM (Gaussian mixture model) provides a series of clusters with distinct features which are easily modeled. In the next step, we compute a piecewise linear model in the reduced-dimensional space for each cluster, which is used for motion synthesis. The local models are capable of reducing computational expense by constraining the solution space and also restricting the jitter problem in the motion synthesis phase.

In this section, two types of local linear models, i.e., radial basis function interpolation model and optimization model are introduced. In Chapter 6, two different techniques relying upon the interpolation model and the optimization model will be presented.

### 5.3.1 Radial Basis Function Interpolation Model

Given a number of clusters, each of which consists of motion data and sensor data with a one-to-one mapping, we build a piecewise local model using Radial Basis Functions (RBFs) [7] for each cluster. In each local linear model, a set of mapping functions, i.e., RBFs, are built from sensor data to motion data in the same cluster.

Our problem is stated as follows: for the $j^{\text{th}}$ cluster, we can build a local linear model using RBFs to learn the mapping function $F_j : \mathbf{R}^a \rightarrow \mathbf{R}^b$ such that

$$\mathbf{q}_{ij} = F_j(\mathbf{c}_{ij}) \ , \tag{5.7}$$

where $\mathbf{c}_{ij} \in C_j = \{\mathbf{c}_{ij} | i = 1, 2, .., p_j\}$, $\mathbf{q}_{ij} \in Q_j = \{\mathbf{q}_{ij} | i = 1, 2, .., p_j\}$ and $p_j$ is the number of frames

in cluster $j$. $a$ and $b$ are the number of reduced dimensions of sensor data and motion data, respectively. $F_j$ is a mapping function from sensor data space to motion data space. In this model, $F_j$ consists of a set of radial basis functions, the values of which depend upon the distance between the input data and the sample data. The radial basis itself has a maximum when the input data is exactly matched to the sample data, i.e. the distance is zero.

Our radial basis function interpolation model is described as follows: given a new input sensor data point $\tilde{\mathbf{c}}_t$ at the time frame $t$, if this data is classified as the $j^{\text{th}}$ cluster, the corresponding motion data $\tilde{\mathbf{q}}_t$ is generated by interpolating the example motion data. The interpolation is expressed in Equation 5.8 as follows,

$$\tilde{\mathbf{q}}_t = F_j(\tilde{\mathbf{c}}_t) = \bar{\mathbf{q}_j} + A_j \sum_{i=1}^{p_j} \mathbf{w}_{ij} \phi(||\tilde{\mathbf{c}}_t - \mathbf{c}_{ij}||) \,, \tag{5.8}$$

where $\tilde{\mathbf{q}}_t$ is the high quality pose we want to synthesize, $\mathbf{w}_{ij}$ and $\phi()$ are the radial basis function wegiths and radial basis functions themselves, respectively. $||\cdot||$ denotes a metric – in our case Euclidian distance. Recall that $\bar{q_j}$ and $j^{\text{th}}$ are both parameters in the PCA model. $\bar{q_j}$ is the mean of motion data in the $j^{\text{th}}$ cluster, and $A_j$ is the transformation matrix built from the eigenvectors corresponding to the largest seven eigenvalues of the covariance matrix.

There are several choices for $\phi()$, including Gaussians, multi-quadratics, or thin plate splines. We chose the Gaussian function,

$$\phi(r) = e^{-r^2/\sigma^2},$$

because it is non-linear and provides good results when applied locally. The width $\sigma$, determines the area covered by the Gaussian function on the data. Since data points are not uniformly distributed in data space, in order to improve output quality we implement a dynamic $\sigma$ [7] dependent upon the density of local data.

Thus our RBF interpolation model is

$$\tilde{\mathbf{q}}_t = F_j(\tilde{\mathbf{c}}_t) = \bar{\mathbf{q}}_j + A_j \sum_{i=1}^{p_j} \mathbf{w}_{ij} e^{-(||\tilde{\mathbf{c}}_t - \mathbf{c}_{ij}||)^2/\sigma^2} \ . \tag{5.9}$$

In Equation 5.9, all the parameters are known except $\mathbf{w}_{ij}$. $\mathbf{c}_{ij}$ and $\mathbf{q}_{ij}$ are the sample sensor data and motion data from the clustering of our captured database. $\tilde{\mathbf{c}}_t$ is the new sensor data input. $\bar{\mathbf{q}}_j$ and $A_j$ are learned from the PCA model. $\sigma$ is also learned from the existing data.

**Model Learning from the Existing Database**

The radial basis function weight $\mathbf{w}_{ij}$, for $i = 1, ..., p_j$ in a specific cluster number $j$, is an unknown vector in the interpolation model. We derive the value of $\mathbf{w}_{ij}$ by taking advantage of the properties of radial basis functions. As mentioned, a radial basis function $\phi$ reaches it maximum at $\mathbf{c}_{ij}$ (i.e. when the distance is zero). By using the local cluster data $\{\mathbf{c}_i^j, \mathbf{q}_i^j\}$, we can solve the linear system in Equation 5.9 for the unknown weights $\mathbf{w}_{ij}$.

Please refer to the seminal work of Micchelli [28] and the survey by Powell [30] for details of this interpolation model. Interpolation using radial basis functions has been used for a variety of applications, such as image warping [2, 33] and motion generation [31].

### 5.3.2 Multivariate Gaussian Model

A multivariate Gaussian model, also called a multivariate normal model, is a statistical model satisfying a multivariate Gaussian distribution, which is a generalization of the one-dimensional Gaussian distribution. As with the RBF interpolation model, we build a Multivariate Gaussian Model for each cluster. The difference is that the new model is learned from the motion data, aiming at measuring a-priori likelihood of some pose using the existing motion data in a selected

cluster.

A probability density function (pdf) for a multivariate Gaussian distribution, where $x \in R^d$ and $d$ is the dimension of $x$, is defined as

$$N(x;\mu,\Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))} , \tag{5.10}$$

where $\mu$ denotes the mean of the distribution and $\Sigma$ is a $d \times d$ matrix specifying the covariance of the distribution.

Based upon Equation 5.10, the multivariate Gaussian distribution in our motion data can be formulated as

$$N(\tilde{\mathbf{q}}_t|Q_j) = \frac{1}{(2\pi)^{D/2}|\Lambda|^{1/2}} e^{(-\frac{1}{2}((\tilde{\mathbf{q}}_t-\bar{q_j})^T\Lambda^{-1}((\tilde{\mathbf{q}}_t-\bar{q_j}))} , \tag{5.11}$$

where $\tilde{\mathbf{q}}_t$ is a new pose and $Q_j$ is the motion data in the $j^{\text{th}}$ cluster. $D$ is the dimension of $\tilde{\mathbf{q}}_t$ and $Q_j$. $\bar{q_j}$ denotes the mean vector of the motion data in the specific cluster $Q_j$ and $\Lambda$ is a $D \times D$ matrix specifying the data covariance. $|\Lambda|$ is the determinant of $\Lambda$. As in Chai et al. [9], we use the negative log of $N(\tilde{\mathbf{q}}_t|Q_j)$ to derive a multivariate Gaussian model defined as

$$P(\tilde{\mathbf{q}}_t|Q_j) = ((\tilde{\mathbf{q}}_t - \bar{q_j})^T\Lambda^{-1}((\tilde{\mathbf{q}}_t - \bar{q_j})) , \tag{5.12}$$

which is used as a model term in the optimization method for motion synthesis. The output of this model shows the probabilistic distribution of the specific pose in the local region. The smaller the output value is, the more likely the pose is in the cluster.

# Chapter 6

# Motion Synthesis

When our system is online, the user attaches eight accelerometers which are embedded in Wii controllers positioned at appropriate body locations. Our system starts receiving a series of acceleration signals $\tilde{\mathbf{c}}_t, t = 1, \ldots, M$, where $M$ is the length of motion, from the accelerometers. Our system uses $\tilde{\mathbf{c}}_t, t = 1, \ldots, M$ as input to synthesize high quality motion data $\tilde{\mathbf{q}}_t, t = 1, \ldots, M$, which is the animation of a 3D avatar. Compared with $\tilde{\mathbf{q}}_t$, our control signal $\tilde{\mathbf{c}}_t$ is relatively low-dimensional, which means that $\tilde{\mathbf{c}}_t$ lacks of much necessary information for controlling animation. In addition, signals provided by the accelerometers are limited by the accuracy and sensitivity of the equipment itself. Now the challenge is to determine how to use such a low dimensional and less accurate signal to control a high-dimensional motion data with high fidelity.

In this chapter, we explain our approach to motion synthesis using two techniques, interpolation and optimization, based upon the piecewise linear models defined in Section 5. Our interpolation method depends upon the radial basis function interpolation model described in Section 5.3.1. Similarly, our optimization method depends upon the multivariate Gaussian model described in Section 5.3.2.

## 6.1 Motion Estimation with Interpolation

Given the new input sensor data $\tilde{\mathbf{c}}_t$, where $t$ is an integer frame number, we apply the piecewise RBF interpolation models learned from the previous step to synthesize the new high quality motion $\tilde{\mathbf{q}}_t$. For the input sensor data $\tilde{\mathbf{c}}_t$ at frame $t$, we identify its closest cluster by calculating the shortest distance between $\tilde{\mathbf{c}}_t$ and the mean vectors of all the clusters of sensor data, characterized as $\bar{\mathbf{c}}_j, 1 \leq j \leq K$. If it is classified as cluster $j$, we use RBF mapping function $F_j()$ defined in Equation 5.9 to synthesize a new motion data frame $\tilde{\mathbf{q}}_t$.

### 6.1.1 Jitter and Motion Post-processing

Motion synthesis using interpolation is likely to generate jitter in the animation. The main reason is that the motion which is generated frame by frame using interpolation depends upon an unstable control signal. In the process of motion estimation, the animation highly depends upon the acceleration signal $\tilde{\mathbf{c}}_t$, which is the signal at one frame. Without considering the temporal relationships between consecutive frames, only spatial information is under consideration. The animation will be mixed up if $\tilde{\mathbf{c}}_t$ is unstable. Unfortunately, accelerometers usually provide noisy signals with many outliers. In our experiments, motion synthesis using interpolation is incapable of guaranteeing smoothness in the synthesized motion.

To alleviate the jitter problem, we perform a motion post-processing step. We simply smooth the synthesized motion data $\tilde{\mathbf{Q}}$, using a smoothing window size of 8. For each $\tilde{\mathbf{q}}_t$, we select a smoothing window starting from $\tilde{\mathbf{q}_{t-4}}$ to $\tilde{\mathbf{q}_{t+3}}$. The value of a new vector, averaged out over the consecutive frames, replaces the original $\tilde{\mathbf{q}}_t$. The smoothing window starts at the beginning of motion sequence and moves frame by frame until the end of the animation.

## 6.2 Motion Estimation with Optimization

As discussed above, an inherent problem of motion interpolation is the jitter problem, i.e., the generated animation is not smooth. Motion smoothness is one critical issue in our previous approach. We solve the jitter problem by using an optimization scheme, in which there exists a smoothness term in an objective function. Our goal is to generate smooth animation without post-processing.

Optimization is commonly used in computer animation research, specifically for motion synthesis. Many constraints and parameters should be taken into account for motion synthesis, e.g. smoothness constraint, control constraint, foot constraint etc. Optimization is a means that can assemble a variety of constraints and parameters into one objective function. However, the dimension of human motion is so high that it is very difficult for a motion optimization method to achieve optimal results in such an unconstrained high-dimensional solution space. Setting reasonable constraints with relatively low dimension definitely will help solve the optimization problems. In this thesis, the piecewise local multivariate Gaussian model presented in Section 5.3.2 serves this purpose.

During the motion synthesis phase, the user performs actions with only 3D acceleration sensors (Wii controllers) attached to the body. Using the sensor signals $\tilde{\mathbf{c}}_t, t = 1, \ldots, M$ as input, we attempt to synthesize high-dimensional motion data $\tilde{\mathbf{q}}_t, t = 1, \ldots, M$, where $M$ is the length of motion. For each frame of the input sensor data $\tilde{\mathbf{c}}_t$, we search in the previously built clustering results of the acceleration signals database $C$, find the the nearest cluster for the input $\tilde{\mathbf{c}}_t$ by calculating the distance between $\tilde{\mathbf{c}}_t$ and the mean vectors of all the clusters of sensor data. Then we use the local linear model associated with the selected cluster as a hard constraint to delimit the solution space. Finally we solve an optimization problem to get an optimal pose $\tilde{\mathbf{q}}_t$ at current frame $t$.

In this optimization problem, there exist several crucial factors, such as the selected multivariate Gaussian models, smoothness and the control signal, that determine the fidelity of the generated animation. We have three terms in the objective function to deal with. These terms are the model

term, the smoothness term and the control term, which are designed under the following considerations:

- **The model term** measures the distance between the synthesized motion $\tilde{\mathbf{q}}_t$ and the "feature" pose in the selected cluster by calculating the distance between $\tilde{\mathbf{c}}_t$ and the mean of all the clusters of sensor data. The model term measures the a-priori likelihood of the synthesized pose using the information from the nearest clusters and shows the probabilistic distribution of the synthesized pose in the local region. The smaller the value is, the more likely the synthesized pose is in the cluster. This term provides a "fundamental" pose which is regulated by the other terms.

- **The smoothness term** measures the distance between the current pose $\tilde{\mathbf{q}}_t$ and the previous two poses $\tilde{\mathbf{q}_{t-1}}$ and $\tilde{\mathbf{q}_{t-2}}$. This term improves the smoothness of the synthesis motion by limiting the distance between the new $\tilde{\mathbf{q}}_t$ and the previous two frame $\tilde{\mathbf{q}_{t-1}}$, $\tilde{\mathbf{q}_{t-2}}$.

- **The control term** measures the distance between the control signal $\tilde{\mathbf{c}}_t$ and the estimated signal from the synthesized pose $\tilde{\mathbf{q}}_t$. This term guides the optimization toward the input sensor signal.

Our problem would be formulated as an optimization problem

$$argmin_{\tilde{\mathbf{r}}_t} E(\tilde{\mathbf{c}}_t, \tilde{\mathbf{q}}_t), \tag{6.1}$$

where the objective function $E(\tilde{\mathbf{c}}_t, \tilde{\mathbf{q}}_t)$ can be written as

$$E(\tilde{\mathbf{c}}_t, \tilde{\mathbf{q}}_t) = (\tilde{\mathbf{q}}_t - \bar{q_t})^T \Lambda^{-1} (\tilde{\mathbf{q}}_t - \bar{q_t}) \tag{6.2}$$

$$+\omega_s ||\tilde{\mathbf{q}}_t - 2\tilde{\mathbf{q}_{t-1}} + \tilde{\mathbf{q}_{t-2}}||^2 \tag{6.3}$$

$$+\omega_c ||G_j(\tilde{\mathbf{q}}_t) - \tilde{\mathbf{c}}_t||^2 . \tag{6.4}$$

Notice that $\tilde{\mathbf{q}}_t$ is represented by $\tilde{\mathbf{r}}_t$ in the low-dimensional space, i.e. $\tilde{\mathbf{q}}_t = \bar{q_t} + A_j \tilde{\mathbf{r}}_t$. Equation 6.2 denotes the model term, where $Q_t$ is a set of sample motion data in the neighboring clusters for current signal $\tilde{\mathbf{c}}_t$. $\Lambda$ is the covariance matrix for $Q_t$ and $\bar{q_t}$ is the mean vector. $A$ is the eigenvalue of $\Lambda$. Equation 6.3 denotes the smoothness term, where $\tilde{\mathbf{q}_{t-1}}$ and $\tilde{\mathbf{q}_{t-2}}$ are the synthesized poses in the previous two frames. Equation 6.4 denotes the control term, where $G_j(\tilde{\mathbf{q}}_t)$ is a reverse mapping function from $Q$ to $C$, for estimating the control signal given a pose $\tilde{\mathbf{q}}_t$. This mapping function is implemented with a RBF interpolation similar to the one used for model learning, but the reverse in mapping direction. In addition, $\omega_s$ and $\omega_c$ are weights for the smoothness and control terms, respectively, satisfying $\omega_s + \omega_c = 1$. In our experiment, $\omega_s = 0.2$ and $\omega_c = 0.8$. Our problem is nonlinear, and a Levenberg-Marquardt algorithm is employed to solve this problem.

Each of three terms plays a role in the final synthesized motion. The model term provides hard constraints in the final solution space, while the control term can make the synthesized motion close to real motion by minimizing distance between them. The smoothness term is the key driving force that improves the jitter problem. Moreover, these terms are independent. For example, if there is noise in the received signals, the control term would become unexpectedly large, but the model and smoothness terms can help drag the synthesized motion back to the ”right” position.

### 6.2.1 Advantage and Disadvantage of the Optimization Method

The proposed the optimization method aims at removing the jitter in the generated animation. The utilization of the smoothness term does alleviate the jitter problem, if not totally remove it, because the term makes it possible to take temporal information into account. Although choosing the frame-by-frame fashion for motion synthesis should produce some jitter in the animation, in practice the generated motion sequence without any post-processing looks smooth and acceptable (see Section 7.3 for more detail).

As mentioned, we still use the optimization method in a frame-by-frame fashion similar to the RBF interpolation method. The computational cost is a significant factor, since the optimization method is usually computational-intensive. Traditional optimization of spacial and temporal factors over a relatively long period of time (i.e. space-time optimization) needs a large amount of computational time to converge, which is suitable for off-line application, but not our approach. Although we select a frame-by-frame method and carefully tune the parameters to make sure the solution is flexible, this optimization method is slower than the interpolation method (see Section 7.3 for more detail). High computational cost is a weakness of our method.

Although optimization is restricted by the high cost in terms of computational time, there exist various impressive research efforts on this to extend the optimization method to real-time applications. For example, Chai's system[9] builds a local statistical model online, which can accelerate the motion optimization in real-time. The goal of our system is similar to theirs, where realistic human motion is reconstructed in a performance capture system exploiting control signals from lost cost devices. Both systems differ in the input devices and the quality of input signal. They employ two video cameras and a few reflective markers. Using the the markers' location as input signals in an energy function, they can achieve real-time performance. Compared to their input signals, our accelerometer sensor signals are much noisier which takes our optimization function more time to converge. Choosing more efficient solvers and tuning the parameters more carefully may help.

# Chapter 7

# Experimental Results and Numerical Comparison

In this chapter, we give our results relying upon the approach described previously. We evaluate the results visually compared with the real video and quantitatively compared with the ground truth motion. We also show a comparison of results using two dimensionality reduction techniques (i.e., PCA and LLE) and two techniques for motion synthesis (i.e., interpolation and optimization).

## 7.1 Results and Evaluation

Relying upon the database built in the data collection, we test the performance of our system with two subjects performing various actions. The new sensor signals are used as an input of our system to produce our on-line animation.

We perform an end-to-end evaluation to measure the accuracy of our system. During capture sessions we also recorded the high quality motions using an optical motion capture system. The

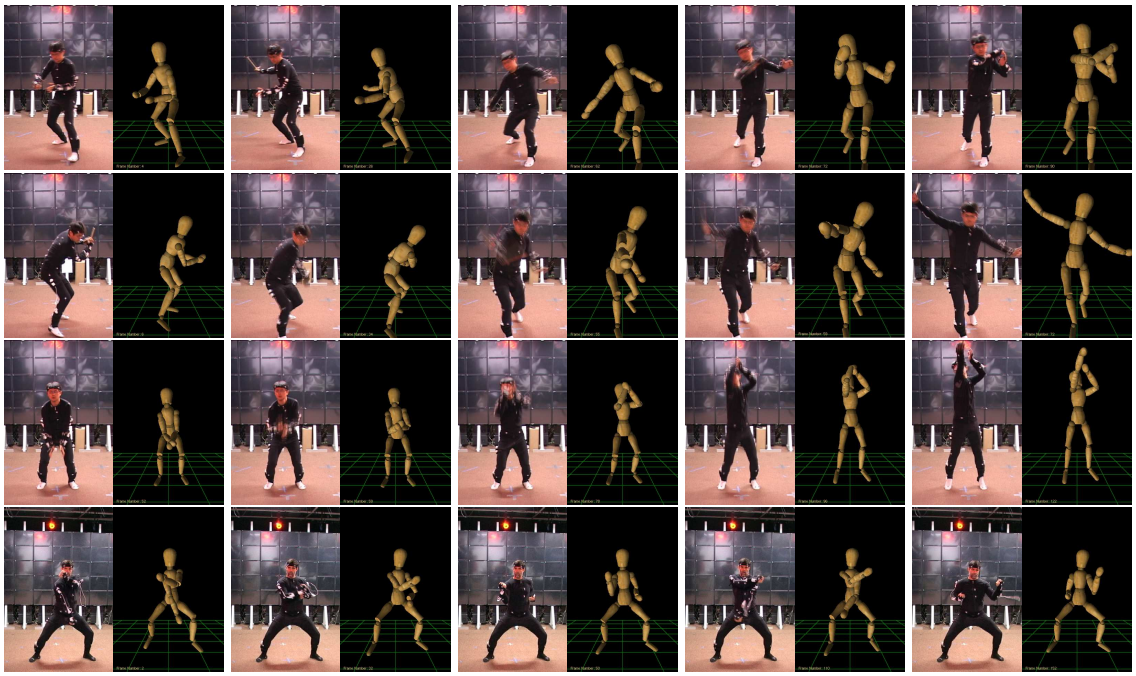


Figure 7.1: Result of motion synthesis using interpolation and PCA. Four different actions (one in each row) synthesized by our system. Each frame shows on the left side the actual pose and on the right side the synthesized pose.

recorded high quality motions are used as ground truth that can be compared against the synthesized motion frame by frame. The recorded motions and the synthesized motions are both converted from quaternion data to joint angle data for error calculation. We then use the normalized Root Mean Square (RMS) distance $e$ to measure the distance quantitatively. The unit of $e$ is degrees of freedom per angle. $e$ is defined as below:

$$e = RMS(\tilde{\mathbf{q}}_\mathbf{k}, \mathbf{q}_\mathbf{k}) = \sqrt{\frac{\sum_{i=1}^{n}(\tilde{\mathbf{q}}_{\mathbf{k},\mathbf{i}} - \mathbf{q}_{\mathbf{k},\mathbf{i}})^2}{n}}, \tag{7.1}$$

where $k$ is the frame index, $\tilde{\mathbf{q}}_\mathbf{k}$ is the synthesize motion, $\mathbf{q}_\mathbf{k}$ is the ground truth motion and $\mathbf{q}_{\mathbf{k},\mathbf{i}}$ is the $i^{\text{th}}$ dimension of $\mathbf{q}_\mathbf{k}$. Similarly, $\tilde{\mathbf{q}}_{\mathbf{k},\mathbf{i}}$ is the $i^{\text{th}}$ dimension of $\tilde{\mathbf{q}}_\mathbf{k}$.

Table 7.1: Normalized RMS distance is used to compare, for each action, the synthesized motion with the ground truth motion captured directly by the optical motion capture system.

| Actions | Frame Number | Average RMS | Processing Time |
|---|---|---|---|
| Basketball shot | 302 | 0.41 | 5.78 sec. |
| Tennis Forehand | 256 | 0.21 | 4.90 sec. |
| Tennis Backhand | 206 | 0.40 | 3.94 sec. |
| Middle Block | 160 | 0.54 | 3.07 sec. |

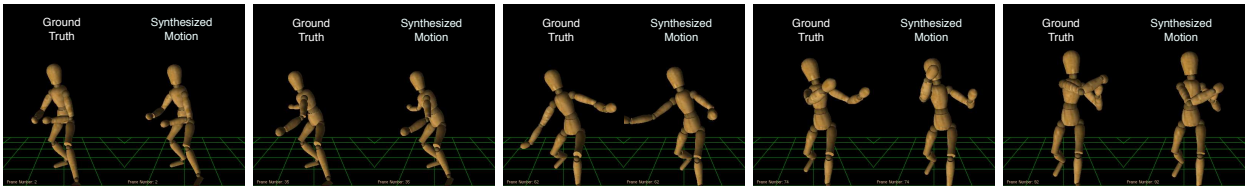



Figure 7.2: Synthesized motion compared to the ground truth motion. Each frame shows on the left side the ground truth motion and on the right side the synthesized motion.

### 7.1.1 Results using PCA and Interpolation

Figure 7.1 shows the results using PCA for dimensionality reduction along with interpolation for motion synthesis, consisting of four synthesized actions: tennis forehand, tennis backhand, basketball shot and karate middle block. The results clearly show that the synthesized motion precisely captured the poses of the subjects.

Table 7.1 shows the RMS distances for four synthesized motions. Figure 7.2 shows a comparison of one of the synthesized motions with the corresponding ground-truth motion. The results of visual and quantitative comparisons show that our low cost system generates motions with the quality equivalent to that of an expensive optical motion capture systems. In terms of computational expense, our system is efficient. The motion synthesis is implemented in *Matlab*® at a rate of about 0.019 seconds/frame, compared with real time at a rate of 0.016 seconds/frame (for 60

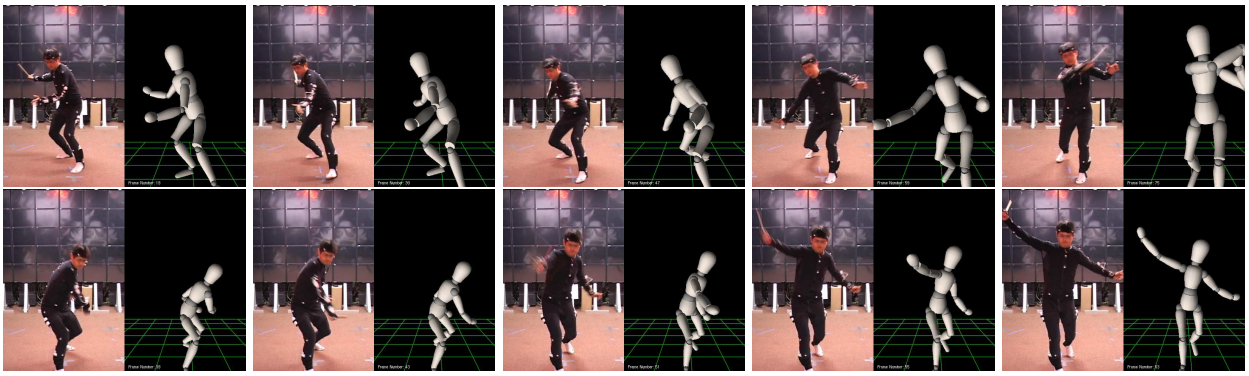


Figure 7.3: Result of motion synthesis using interpolation and LLE. Two different actions (one in each row) synthesized by our system. Each frame shows on the left side the actual pose and on the right side the synthesized pose.

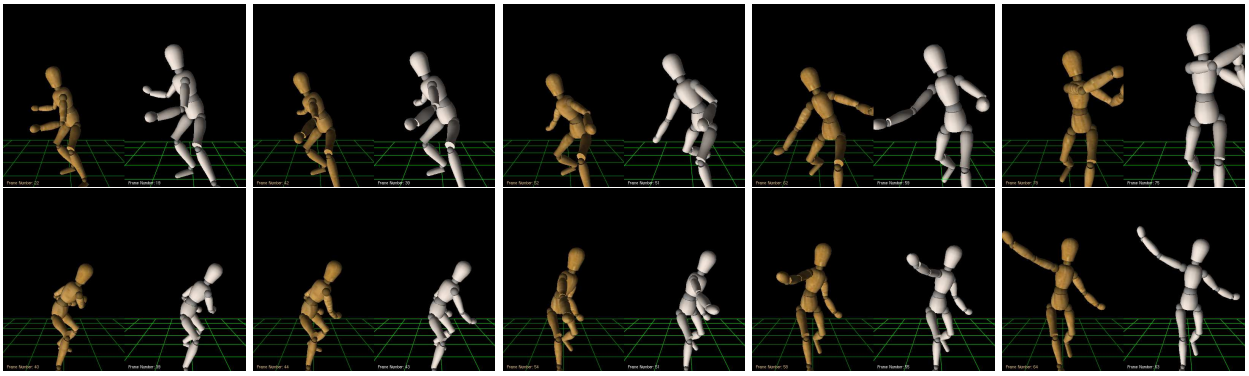


Figure 7.4: Synthesized motion compared to the ground truth motion. Each frame shows on the left side the ground truth motion and on the right side the synthesized motion.

frames per second animation).

### 7.1.2 Results using LLE and Interpolation

Figure 7.3 shows the results using LLE for dimensionality reduction along with interpolation for motion synthesis, consisting of two synthesized actions: tennis forehand and backhand. Table 7.2 shows the RMS distances for two synthesized motions. Figure 7.4 shows a numerical comparison

Table 7.2: Normalized RMS distance is used to compare, for each action, the synthesized motion with the ground truth motion captured directly by the optical motion capture system.

| Actions | Frame Number | Average RMS |
|---|---|---|
| Tennis Forehand | 256 | 0.062 |
| Tennis Backhand | 206 | 0.057 |

of the synthesized motions with the corresponding ground-truth motion.

## 7.2 Comparison between PCA and LLE

The performance of the system relies on the ability to represent human motion in a low-dimensional space. Without this low dimensional representation, the clustering algorithm has difficulty in clustering high dimensional data. We attempt to depend on PCA to reduce data dimensionality; however, as we discussed in Chapter 2, it is hard for a global linear method (such as PCA and MDS) to model a heterogeneous database which is possibly nonlinear. In comparison, LLE can preserve the topology in the high dimensional data and compute low dimensional embedding with neighborhood relationships preserving high dimensional data.

In this section, we compare the performance of our nonlinear manifold learning algorithm (i.e., LLE) with the linear learning algorithm (i.e. PCA) for dimensionality reduction. Figure 7.5 plots the RMS errors of synthesized motion for both methods. It shows that LLE creates more accurate results than PCA and demonstrates LLE is a more appropriate technique to uncover the underlying structure of our data.

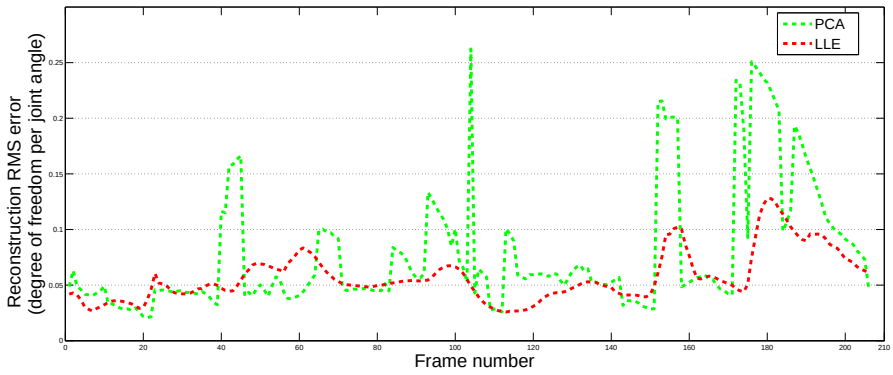



Figure 7.5: Comparison of construction RMS errors of synthesized tennis backhand motions from low-dimensional control signals. The average RMS error for PCA is 0.078 degree/joint per frame while average error of LLE is 0.057 degree/joint per frame. All the test motions are not in the database.

## 7.3 Comparison between Interpolation and Optimization

Without post-processing, the synthesized motion using optimization is visually compared with the synthesized motion using interpolation in Fig. 7.6. Both results are generated by using PCA as the dimensionality reduction technique and GMM for data clustering. The interpolation method depends upon the RBF interpolation model while the optimization method depends upon the multivariate Gaussian model.

In our experiments, we compare two synthesized motion both without post-processing. The motion generated by interpolation has a serious jitter problem while the motion generated by optimization maintains smoothness in most cases. Our experiments demonstrate optimization makes much progress with regard to animation smoothness. On the other hand, the interpolation method generates a motion sequence at an interactive frame-rate while the optimization method has a high computational cost. For example, generating the motion sequence of around 300 frames (i.e., 5 seconds) of a basketball shot takes the interpolation method only 5.87 seconds while the same motion takes the optimization method around 5 minutes.

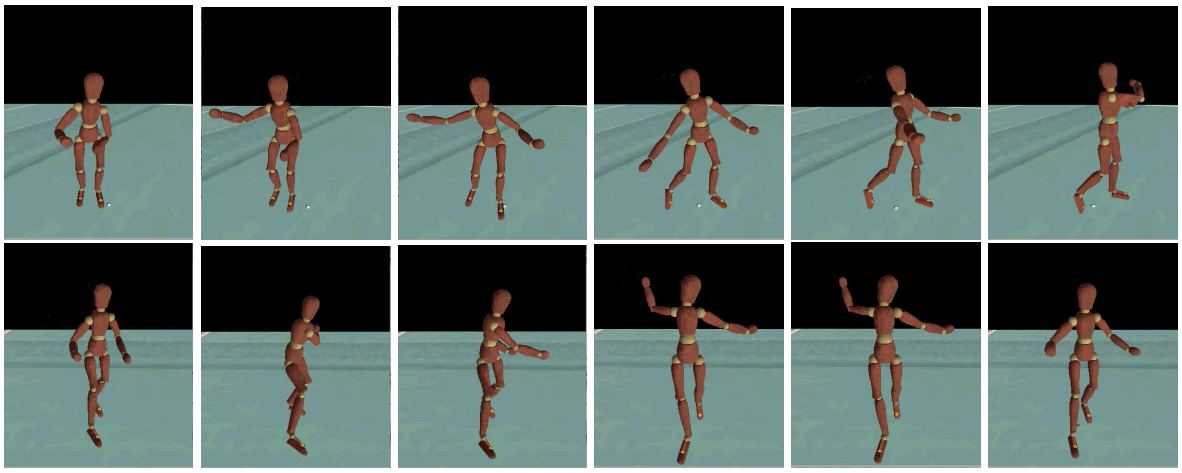


Figure 7.6: Result of motion synthesis using optimization. Two different actions (one in each row) synthesized by our system. The first row is tennis forehand, the second row is tennis backhand.

In short, the interpolation method is preferred in a number of interactive applications which require immediate response at the expense of reduced quality of synthesized motion; by comparison, the optimization method sacrifices the time so as to generate animation of a higher fidelity.

# Chapter 8

# Conclusion

In this thesis, we propose a realistic motion estimation framework to control the animation of 3D avatars. A prototype system is built upon a small number of Nintendo Wii controllers that are easy to attach to the human body. Using Wii controllers as input devices, we are able to generate high quality motion data using our motion estimation framework.

The framework is developed in a data-driven fashion, which includes two phases: model learning from an existing high quality motion database, and motion synthesis from the control signal. In the model learning phase, we build a high quality piecewise local model that learned from a large motion database. We present two different techniques for model learning. Principal component analysis and locally linear embedding are applied to reduce the dimensionality of human motion data. We demonstrate that the nonlinear manifold learning model using LLE can identify the nonlinear structure in motion data more precisely. In the motion synthesis phase, we take a 3D accelerometer sensor signal as input, and synthesize high-quality motion from the piecewise local model we learned. We implement an interpolation method and an optimization method for motion synthesis using the intuitive control signal. We compare the advantage and disadvantage of both methods with regard to smoothness and computational cost.

Our system takes advantage of high fidelity and accuracy of data-driven approach, as well as the intuitive control of 3D accelerometers. The system performs well, which makes it available to a wide range of interactive applications, such as character control in 3D virtual environments and occupational training.

# Bibliography


[1] AGUIAR, E., STOLL, C., THEOBALT, C., AHMED, N., SEIDEL, H., AND THRUN, S. Performance capture from sparse multi-view video. *Proceedings of ACM SIGGRAPH 2008* (2008), 1–10.

[2] ARAD, N., DYN, N., REISFELD, D., AND YESHURUN, Y. Image warping by radial basis functions: applications to facial expressions. *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing 56*, 2 (1994), 161–172.

[3] ARIKAN, O., AND FORSYTH, D. A. Synthesizing constrained motions from examples. *ACM Transactions on Graphics 21*, 3 (July 2002), 483–490.

[4] ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. Motion synthesis from annotations. *ACM Transactions on Graphics 22*, 3 (2003), 402–408.

[5] BEZDEK, J. C. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[6] BOUMAN, C. A. Cluster: An unsupervised algorithm for modeling gaussian mixtures. Available from http:// www.ece.purdue.edu/˜bouman, April 1997.

[7] BUHMANN, M. D. *Radial Basis Functions : Theory and Implementations*. Cambridge University Press, 2003.

[8] CARVALHO, S. R., BOULIC, R., AND THALMANN, D. Interactive low-dimensional human motion synthesis by combining motion models and pik. *Computer Animation and Virtual Worlds 18*, 4-5 (2007), 493–503.

[9] CHAI, J., AND HODGINS, J. K. Performance animation from low-dimensional control signals. *ACM Transactions on Graphics 24*, 3 (2005), 686–696.

[10] COX, T. F., AND COX, M. A. A. *Multidimensional Scaling*. Chapman & Hall, London, 1994.

[11] DEMPSTER, A. P., LAIRD, N. M., AND RUBIN, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological) 39*, 1 (1977), 1–38.

[12] ELGAMMAL, A., AND LEE, C. Inferring 3d body pose from silhouettes using activity manifold learning. *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on 2* (2004), II681–688 Vol.2.

[13] FODOR, I. A survey of dimension reduction techniques. Tech. rep., Lawrence Livermore National Laboratory, 2002.

[14] GLEICHER, M. Comparing constraint-based motion editing methods. *Graphical Models 63* (2001), 107–134.

[15] GUO, S., AND ROBERGÉ, J. A high-level control mechanism for human locomotion based on parametric frame space interpolation. *Proceedings of the Eurographics workshop on Computer animation and simulation '96* (1996), 95–107.

[16] JACKSON, J. A user's guide to principal components. *New York: John Wiley and Sons* (1991).

[17] JAEGGLI, T., KOLLER-MEIER, E., AND GOOL, L. V. Multi-activity tracking in LLE body pose space. *Human Motion - Understanding, Modeling, Capture and Animation, Proceedings 4814* (2007), 42–57.

[18] JAIN, A. K., AND DUBES, R. C. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[19] JAIN, A. K., MURTY, M. N., AND FLYNN, P. J. Data clustering: a review. *ACM Computing Surveys 31*, 3 (1999), 264–323.

[20] JOLLIFFE, I. Principal component analysis. *Springer-verlag* (1986).

[21] KOVAR, L., AND GLEICHER, M. Flexible automatic motion blending with registration curves. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), 214–224.

[22] KOVAR, L., GLEICHER, M., AND PIGHIN, F. Motion graphs. *ACM Transactions on Graphics 21*, 3 (July 2002), 473–482.

[23] KUMAR, M. A motion graph approach for interactive 3d animation using low-cost sensors. Master's thesis, Virginia Tech, 2008.

[24] LEE, J., CHAI, J., REITSMA, P. S. A., HODGINS, J. K., AND POLLARD, N. S. Interactive control of avatars animated with human motion data. *Proceedings of ACM Siggraph 2002* (2002), 491–500.

[25] LEE, J. A., AND VERLEYSEN, M. Nonlinear dimensionality reduction. *International Statistical Review 76*, 2 (Aug. 2008), 308–309.

[26] LIU, G., ZHANG, J., WANG, W., AND MCMILLAN, L. Human motion estimation from a reduced marker set. *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games* (2006), 35–42.

[27] MACQUEEN, J. Some methods for classification and analysis of multivariate observations. *Fifth Berkeley Symposium on Mathematical Statistics and Probability* (1967), 281–297.

[28] MICCHELLI, C. A. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation 2* (1986), 11–22.

[29] OORE, S., TERZOPOULOS, D., AND HINTON, G. A desktop input device and interface for interactive 3D character animation. *Proceedings of Graphics Interface 2002* (May 2002), 133–140.

[30] POWELL, M. J. D. Radial basis functions for multivariable interpolation: a review. *Algorithms for Approximation* (1987), 143–167.

[31] ROSE, C., COHEN, M. F., AND BODENHEIMER, B. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 5 (1998), 32–41.

[32] ROWEIS, S. T., AND SAUL, L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 5500 (2000), 2323–2326.

[33] RUPRECHT, D., AND MÜLLER, H. Image warping with scattered data interpolation. *IEEE Computer Graphics and Applications 15*, 2 (1995), 37–43.

[34] RUSPINI, E. H. A new approach to clustering. *Information and Control 15*, 1 (1969), 22–32.

[35] SAFONOVA, A., AND HODGINS, J. K. Construction and optimal search of interpolated motion graphs. *Proceedings of ACM Siggraph 2007* (2007), 106.

[36] SAFONOVA, A., HODGINS, J. K., AND POLLARD, N. S. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics 23*, 3 (2004), 514–521.

[37] SAUL, L. K., AND ROWEIS, S. T. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research 4*, 2 (2004), 119–155.

[38] SHIN, H. J., AND LEE, J. Motion synthesis and editing in low-dimensional spaces: Research articles. *Comput. Animat. Virtual Worlds 17*, 3-4 (2006), 219–227.

[39] SHIRATORI, T., AND HODGINS, J. K. Accelerometer-based user interfaces for the control of a physically simulated character. *Proceedings of ACM Siggraph Asia 2008* (2008), 1–9.

[40] SLYPER, R., AND HODGINS, J. K. Action capture with accelerometers. *2008 ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (July 2008), 193–200.

[41] TENENBAUM, J. B., DE SILVA, V., AND LANGFORD, J. C. A global geometric framework for nonlinear dimensionality reduction. *Science 290*, 5500 (2000), 2319–2323.

[42] TITTERINGTON, D. M., SMITH, A. F. M., AND MAKOV, U. E. *Statistical Analysis of Finite Mixture Distributions*. John Wiley, New York, 1985.

[43] VLASIC, D., ADELSBERGER, R., VANNUCCI, G., BARNWELL, J., GROSS, M., MATUSIK, W., AND POPOVIĆ, J. Practical motion capture in everyday surroundings. *ACM Transactions on Graphics 26*, 3 (2007), 35.

[44] WILEY, D. J., AND HAHN, J. K. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Applications 17*, 6 (1997), 39–45.

[45] XIE, L., FANG, B., CAO, Y., AND QUEK, F. A nonlinear manifold learning framework for real-time motion estimation using low-cost sensors. *The 37th IEEE Applied Imagery Pattern Recognition workshop* (2008), 1–8.

[46] XIE, L., KUMAR, M., CAO, Y., GRACANIN, D., AND QUEK, F. Data-driven motion estimation with low-cost sensors. In *The 5th IET Visual Information Engineering 2008 Conference* (2008), pp. 600–605.

[47] YEASIN, M., AND BULLOT, B. Comparison of linear and non-linear data projection techniques in recognizing universal facial expressions. *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on 5* (2005), 3087–3092.

[48] ZAHN, C. Graph-theoretical methods for detecting and describing gestalt clusters. *Computers, IEEE Transactions on C-20*, 1 (Jan. 1971), 68–86.

[49] ZHANG, L., SNAVELY, N., CURLESS, B., AND SEITZ, S. M. Spacetime faces: high resolution capture for modeling and animation. *Proceedings of ACM Siggraph 2004* (2004), 548–558.