A Motion Learning-based Framework for Enhancing Keyframe Character Animation

Chao Jin

A Thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fullfilment of the Requirements

for the Degree of Doctor of Philosophy at

Concordia University

Montreal, Quebec, Canada

July, 2012

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: Chao Jin

Entitled: A Motion Learning-based Framework for Enhancing Keyframe Character Animation

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. D. Dysart-Gale	Chair
Dr. W.S. Lee	ExternalExaminer
Dr. P. Bhattacharya	External to Program
Dr. A. Krzyzak	Examiner
Dr. J. Rilling	Examiner
Dr. S. Mudur	Thesis Co-Supervisor
Dr. T. Fevens	Thesis Co-Supervisor

Approved By Dr. V. Haarslev, Graduate Program Director

Dr. Robin Drew, Dean
Engineering and Computer Science

Date July 17th, 2012

ABSTRACT

A Motion Learning-based Framework for Enhancing Keyframe Character Animation

Chao Jin

In the field of computer animation, character animation using keyframes remains a popular technique. In this, the animation sequence is represented compactly by just the more important character poses, termed as keyframes and the rest of the frames, known as in-betweens are generated when needed, say during playing the animation, using the keyframes. The animator specifies (creates) the keyframes while the in-between frames are computed using a suitable interpolation scheme. Interpolation parameters are usually under the control of the animator. Most animation software today will include support for keyframe animation. However, specifying the parameters so that the generated animation sequence fulfils the animator expectations and other motion requirements, say like preserving area or volume, or satisfying a physical constraint, can be quite difficult. The number of degrees of freedom is very high for skeleton-based animation and much higher for mesh-based animation. Physically-based animation techniques have been proposed for character poses to satisfy physics constraints. But animators find it difficult and non-intuitive to specify physics parameters, like body mass, forces. etc. and seem to very much dislike the fact that they lose control over the final animation. Trial and error using the keyframe technique is presently the most popularly adopted solution by animators.

Our main thesis is that for a character action, given just the keyframe representation or the entire animation sequence, we can recover a characterizing motion representation in lower dimension space using manifold learning. This characterizing motion recovers distinguishing information hidden in the huge amount of correlated and coherent character animation data in high dimensional space. Then we can use it to enhance keyframe animation techniques, which can considerably reduce animator effort required for specifying the keyframes for desired quality of animation. Further, these new techniques are equally applicable to 3D skeleton and mesh animation.

In our first major contribution, we present a formulation to adopt the technique of locally linear embedding (LLE) to project the given character animation data into a much lower dimension embedding space. We show that animations depicting distinguishable activities, say walking, running, jumping, etc. take distinct characteristic shapes in lower dimensional embedding space. Based on this embedding, we present a new framework consisting of a reconstruction matrix combined with motion represented in the low dimension embedding space. This framework enables us to generate complete animation sequences in the original high dimensional space while maintaining desirable physical properties in a deformable shape. The latter is done by introducing the concept of a property map in the embedding space of values for the different physical properties of the mesh, for example area, volume, etc. A probability distribution function in embedding space then helps us rapidly choose the required number of in-between poses with desired physical properties. The reconstruction of the animation sequence is achieved by using the whole set of keyframes during interpolation for generating each in-between. This framework has many other applications and we demonstrate this by introducing two other new techniques which further enhance key frame animation.

In another contribution of this research, we present a non-physically based method for incorporating perceivable variations in repetitive motion of an autonomous virtual character while retaining its principal characteristics. Usually, this is rather difficult to achieve using standard keyframe animation techniques, since even small changes in keyframes can result in less predictable changes in the final interpolated animation. The basis for our method is provided by asymmetric bilinear factorization of a given animation derived using the above framework. Keeping the action unchanged (that is the characterizing motion extracted as the embedding in the low dimensional space), we define a method to incorporate controlled perturbations into the reconstruction matrix so as to yield variations of the same motion. Further, to join the varied motion segments into a longer animation sequence, we present an embedding space method, which again makes use of the property map to maintain the desired physical properties.

We also present an effective method for optimized keyframe selection from complete animations or motion capture sequences. Given the fact that most animators are very comfortable with the keyframe animation technique, this will enable animators to work easily with previously created animations or motion capture data. Our solution uses animation saliency and combines it with the embedding. For this, we use the representation of the animation in the form of matrix multiplication of reconstruction matrix with combination weights and then cast the keyframe selection problem into a constrained matrix factorization problem. The error metric that is minimized however uses animation saliency computed in the original high dimensional space. This way, the time consuming optimal search required by the matrix factorization problem in high dimensional space is simplified to a much more efficient method in low dimensional space.

These enhancements to the character keyframe animation technique are possible because of capability of the manifold learning technique like LLE to effectively capture in low dimensional space the characterizing motion information that is present in a given character animation. Together these form the most significant contributions of this thesis.

ACKNOWLEDGEMENTS

I am sincerely thankful to my supervisors, Dr. Thomas Fevens and Dr. Sudhir Mudur, for their guidance, helpfulness, patience, and kindness throughout my doctoral research. Without their help, this thesis would not be possible. I feel very fortunate to work with two excellent professors. I learned a lot from them. It has been a very wonderful academic experience. I am also very thankful to all my thesis committee members Dr. W.S. Lee, Dr. P. Bhattacharya, Dr. A. Krzyzak and Dr. J. Rilling for their valuable suggestions to improve my research work. I would like to express my gratitude to my friends Yingying She, Yue Wang, Xin Tong, Xi Deng and Min Ning for their emotional support. Above all, I am very grateful to my husband Hui Wang and my parents for their priceless love, support and encouragements during the whole memorable journey.

To the memorable journey

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Computer Animation

In this research, we address several issues in creating computer based character animations using the keyframe animation technique. To give a clear idea of what is character animation, we first introduce the subject of animation. Animation has a very long history, which began in the early 1900's, when the basic foundations in current animations were laid. Johnston and Thomas [6], and Greenberg [7], explain that animation is a deliberately interpreted illusion of life. Originally, animation involved the creation of a series of individual drawings replayed as successive frames. The illusion of movement is actually generated by the human eye/brain which interpolates these successive drawings. Gleicher emphasized that animation is a uniquely expressive art form in [8] as " it provides the creator with control over both the appearance and the movement of characters and objects. This gives artists tremendous freedom, which when used well, can create works with tremendous impact". Animation is a very broad concept, and any object can be animated, such as a bumping ball, animal, human or imaginary creature.

1.1.1 Character Animation

Character animation specifically deals with articulated (limbed) character motion. Usually, but not always, it refers to a human-like character performing a recognizable daily life motion. Moreover, computer based character animation mostly focuses on the use of computers to generate realistic motions for virtual characters. 3D Character animations usually use two types of representations as below:

Skeleton based animations Skeleton based animations use a simplified bone structure to represent the character. A frame contains the pre-defined bone structure and a group of joint angles. The pre-defined bone structure remains the same during the whole animation. The character movement is recorded as a sequence of joint angle changes which occur with the progress of time.

Mesh based animations Mesh based animations basically use a group of polygons or triangles to represent the surface (skin) of the character. A frame contains the position of the vertices and the connectivity among vertices (face index). The face index remain the same during animation. The character movement is recorded by the movement of vertex positions which occur with the progress of time. Mesh based animations may contain additional information such as texture coordinates, normal, and material properties, required for rendering and possibly other operations.

Generating realistic motions for virtual characters has become a significant part of creating computer animations. The invention of the motion capture system, which captures 3D movement data from a live performing actor, has been blooming in many real world applications and is also part of extensive research by various academic researchers around the world. Character animation itself in general has

gathered a lot of interest in research, especially in the last 20 years. The increasing interest in this topic is reflected in the increasing number of accepted papers at the conferences in Siggraph[1], Eurogaphics[2], and other annual conferences and journals in the world. As shown in Figure (1.1), realistic motion generation and computer animation-related papers were submitted in great numbers at the Siggraph and Eurographics conferences. Since the year 2005, Siggraph has established a festival named as *"the Computer Animation Festival"*, which selects and shows the best computer animations worldwide. These animations represent the best computer animation not only for the computational techniques used, but also for the artistic effects included. The festival builds a bridge between the cutting edge techniques and the daily needs of the animation industry.

These researches for generating realistic motion for virtual characters have led to many significant achievements. These vast research achievements have also enabled computer generated realistic character motions to proliferate the field of visual effects in media other than films, such as interactive video games and robotics. Along with films, more and more games now rely on animation of game characters to add realism, perform stunts, and behave believably within their respective contexts. For example, at the end of 2009, the movie "*Avatar*"[3] [9] (see Figure (1.2)) brought us an extraordinary visual and audio experience. In *Avatar*, half of the movie was generated using computer animations. The stunning $3D$ virtual planet, named Pandora, is full of photo-realistic computer generated characters with beautifully defined character animations. In the market, the movie also had a huge success. The film earned USD 749 million in America, and USD 2.7 billion worldwide, which

[1]Siggraph, ACM's premier international conference and exhibition on computer graphics and interactive techniques

[2]Eurographics, from the European association for computer graphics

[3]by director James Cameron

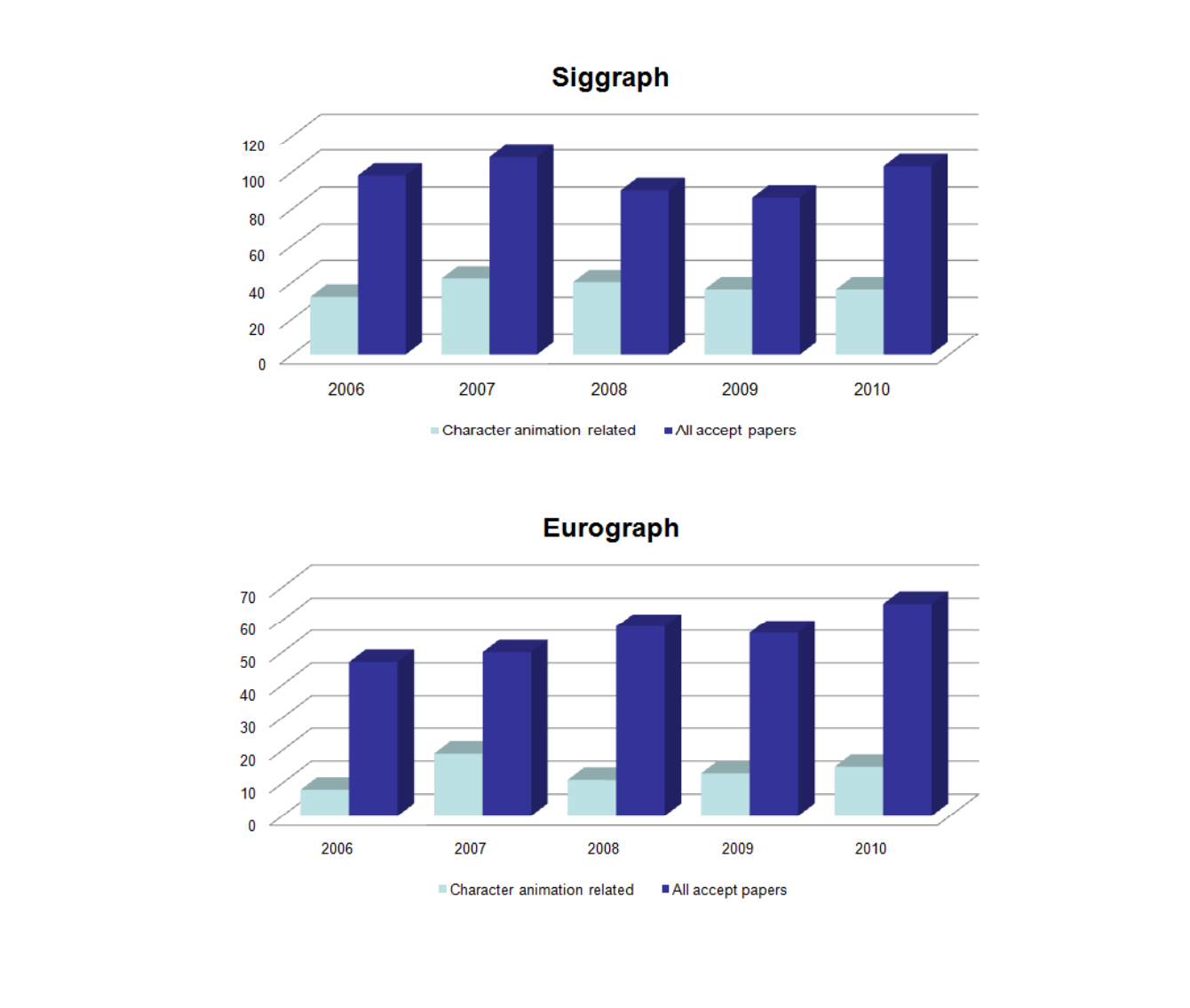


Figure 1.1: Number of accepted papers and the number of papers that have been accepted related to character animation, for Siggraph and Eurographics conferences from 2006 to 2010.

Figure 1.2: A poster of the film *Avatar*, 2009 (images from *Fox* [1])

made it reach the first place at the box office in all-time sales [10].

1.1.2 Character Animation Tools and Techniques

In this subsection, we give a brief overview of the different techniques used for creating character animations. In the next chapter a detailed review of these techniques will be provided.

Since the $1990's$, most animators create their character animations using computers along with some kind of animation software such as Maya, 3DS MAX, and Motion Builder. All these software systems include tools for creating animations using a well established technique known as keyframe animation. In this technique, to create an animation, such as a walking character, animators first create the character with a skeleton and skin, texture and material properties. After designing and loading the character into the computer, animators need to create a group of keyframes that are thought of as key poses to describe the walking motion for the

character and they need to choose the interpolation methods for the in-betweens[4].

It is here that the very important concept, that of keyframes, appears. Keyframe techniques are very important not only for animation creation but also when we display the animations. Usually, complete animation sequences are huge in size and it is a big burden to render, transfer or store them. Therefore the common method is to use only keyframes to represent the whole animation, and generate in-betweens when necessary. With keyframes and interpolation methods, the animation software can then automatically create the whole sequence just from the keyframes and display the complete resulting animation in real-time on the computer screen in front of the animators. The animator can check the visual result and adjust the keyframes (add, remove or change the key poses) to obtain the desired visual result for the full animation sequence. Clearly, a lot of animator effort is required to create the right set of keyframes for a desired motion effect.

The other way to obtain character animation sequences is to directly acquire them from human actors using a motion capture (mocap) system. Mocap systems commonly use optical or magnetic sensors which are strategically placed markers, as shown in Figure (1.3). Magnetic mocap systems calculate the position and orientation by the relative magnetic flux of three orthogonal coils on both the transmitter and every receiver [11]. The relative intensity of the voltage or the current of the three coils allows these systems to calculate both the range and orientation by meticulously mapping the tracking volume. An optical mocap system utilizes triangulation from multiple cameras to estimate the 3D positions of the markers. Typically, it requires a minimal set of $40-50$ markers to capture a full skeleton of a human in motion. The newly introduced Kinect(IR) sensor from Microsoft uses 3D

[4]in-betweens are those animation frames, which are not specified by the animator but are usually interpolated from keyframes

scans and video images to track and capture motion information, without the need for any markers. However, if one wishes to capture the subtler human movements, the marker based equipment will still be required and it may be necessary to have as many as over 300 markers. The $3D$ trajectories of the set of markers constitute a motion sequence for each frame, i.e., the pose is represented by a vector of the marker positions. As the number of markers and/or the number of frames increases, the mocap data easily grows to very large sizes. A typical animation mesh has at least thousands of vertices with a fixed connectivity among the vertices. A mocap system can record complex movements and realistic physical interactions such as secondary animations, and the exchange of forces more easily, and these movements can be recreated in a physically accurate manner. However, it also has its share of weaknesses:

1. It is very noisy due to the deficiency of the sensors.

2. It is still expensive. Nowadays, the price of a full body magnetic mocap system is around USD ten thousand; and a good optical system is around USD a quarter of million. Moreover, they often require a large dedicated space for capturing the motions.

3. The motion capture process is labour intensive and time-consuming for actors and technicians [7, 12].

4. The recording data needs to be suitably transformed for transfer from the performer to the rendered (virtual) character [13–15].

In order to make the mocap technique more widely applicable, the acquired data needs to be made reusable [8, 16–18]. This would enable us to create needed

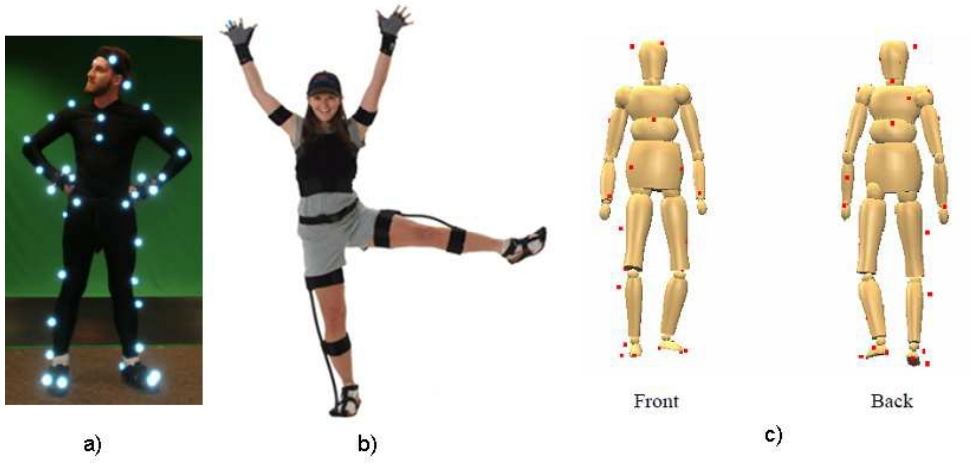


Figure 1.3: Motion capture systems. a) is an example of optical mocap system. The pictures come from forums.worldofwarcraft.com; b) is an example of magnetic system from Measurand Inc.; c) A human pose constructed from the mocap marker placements from [2]. The red dots indicate the marker positions.

motions by reusing pre-recorded mocap data. Furthermore, with the increased availability of mocap data and motion editing techniques, a current trend is to create new high quality animations by joining multiple samples of motions from a mocap database [19–21]. This alternative approach potentially provides animators with a relatively cheap and time-saving approach to quickly obtain high quality motion data for animating their creatures/characters. It does not require the animator to create, edit or adjust keyframes, but does require appropriate setting of parameters to control the joining of motion clips for obtaining the desired animation. Without proper values for these parameters, the resulting animation may not be realistic.

In spite of the advent of mocap systems and techniques, even to this day, keyframe animation continues to be the most popular animation technique used by animators [22]. This is because of the fact that it gives complete interactive control over the animation to the animator. This interactive procedure, however, can be very time consuming, and the animation result is highly dependent on the animators'

abilities. No two animators will choose/set exactly the same set of keyframes even for the same desired animations. Also, the quality of the result is very hard to measure/compare with objective metrics.

1.1.3 Character Animation Representation

Character animation data can be interpreted as a recorded sequence of poses (frames) of a character. Every frame records a particular pose of the character at a certain time. The recorded pose representation could be meshes, skeletons or frame-based videos. A sequence of character animation data has the following properties:

- The data contains a set of N frames, $\mathbf{f}_i, 1 \leq i \leq N$.
- Each frame has the same number and definition of free variables, known as the Degrees of Freedom (DOFs). Different types of recordings have different definition of DOFs. A mesh uses the vertex positions as its DOFs. Depending on the complexity of the virtual character's appearance, the number could reach tens of thousands of DOFs. A skeleton is a simplified representation of a character using joint angles between bones, so it has relatively less number of DOFs, typically, one hundred or so for a virtual human. Videos use the colour value of pixels as the DOFs. This again, depending on the resolution and color range, may number in hundreds of thousands.
- The data can be represented by a matrix M. M is an $N \times n$ matrix, where N is the number of frames and n is the number of DOFs for every frame $\mathbf{f}_i$. The columns represent different frames. Each row represents a DOF value along the different times. By this definition, video clips, mesh animation and skeleton animation can all be represented as M. If we put the row as the axis in the R^n space, where n is the number of rows in M, then every column is a vertex

in the R^n space. Because frame $\mathbf{f}_i$ is a function of time t, the whole matrix is a function of time t in the R^n space. In other words, the matrix defines a curve C in the R^n space. With the continuous variable t, the time-space data becomes a trajectory curve in the high dimensional space R^n.

1.1.4 Major Issues in Character Animation

Most research concerned with generation, analysis and manipulation of character animation data such as video, mesh and skeleton sequences can be categorized into the four following topics:

Motion structure extraction and simplification: Motion structure extraction and simplification techniques are used to find the simplified representation for the original motion sequence with a much smaller number of frames. Video summarization techniques, as in [23–25], can be applied to create a subset of keyframes which contains as much information as possible from the original video. Summaries are important because they can rapidly provide users with some information about the content of a large video or create a short version of the original one.

Similar to video summarization, mesh-based keyframe extraction is the process of selecting meaningful frames, reducing and omitting the redundant, and perceptually meaningless frames [26–28]. One purpose of extracting keyframes is to greatly reduce the animation data sizes. The extraction or simplification is necessary for mocap data because data acquisition technologies such as motion capture can produce a very large volume of animation data. If this data is to be used in a computer game or virtual world, researchers would like to pack as much of it as possible into a limited amount of memory. As in [20, 29],

the extraction or simplification may also be important in a film production environment for easy access to animation databases.

Motion blending: Motion blending is a group of algorithms which can produce new motions (blends) by combining multiple animation clips according to a set of time-varying weights. Motion blending has several applications. For example, blending can be used to create seamless transitions between motions, by allowing one to build lengthy, complicated motions out of simpler actions. Another application is interpolation, or creating motions "in-between" for the initial set to produce a parameterized space of motions. The great advantage of motion blending is to avoid the expensive and time-consuming effort of generating original animations from scratch [30–34]

Motion manipulation and synthesis: Motion manipulation and synthesis techniques focus on modifying an existing motion by adding new "elements", such as new personal styles or a new actor, to create new motion sequences. Synthesis of realistic character animation is an active area of research and has many applications in the entertainment and bio-mechanical industry. 2D based motion manipulation starts from image warping and deformation. Image warping and deformation techniques have had a long history [35]. Recent efforts of image warping and deformation have focused on deformation controlled by a user who pulls on various handles [36, 37] while minimizing the distortion of local shapes, as measured by the local deviation from conformal or rigid transformations. These methods, which build on earlier work in as-rigid-as-possible shape interpolation [38], are able to minimize perceivable distortion much more effectively than traditional space-warp methods [39] or standard scattered data

interpolation [40]. For 3D animation, in 1998, Gleicher [41] presented a technique for adapting an animated motion from one articulated figure to another figure with an identical structure but with different segment lengths. After that, many researchers have worked on motion manipulation areas for skeleton models that have much less DOFs than meshes [42–44]. More recently, direct manipulation has proven to be an invaluable tool for mesh editing since it provides an intuitive way for the user to interact with a mesh during the modeling process. Sophisticated deformation algorithms propagate the users' changes throughout the mesh so that the features are deformed in a natural way [36, 45–51].

Motion recognition and tracking: Three-dimensional, human shape estimation and motion tracking are important and challenging research problems. They also have numerous applications such as: 1) posture and gait analysis used for training athletes and physically-challenged persons, 2) human body, hands, and face animation, and 3) automatic annotation of human activities in video databases. Motion information can include the position and the velocity, which are incorporated with intensity values. This motion information is employed to establish matching between consecutive frames. After feature correspondence between successive frames is constructed, the next step is to understand the behaviour of these features. To recognize human activities from an image sequence, researchers typically use one of two types of approaches: approaches based on a state space model or ones which use a template matching technique. In the first case, the features used for recognition include points, lines, and $2D$ blobs. Methods used for template matching usually are applied to meshes of a subject on the image to identify a particular movement.

Gavrila [52] formulated pose-recovery as a search problem. Moreover, he developed a hierarchical decomposition approach to overcome the difficulties brought by the high dimensionality of the search space. Bregler [53] developed a region-based estimation framework (using twists and exponential maps), with the cost function based on the optic flow. In Yamamoto's work [54], tracking was performed by estimating the pose increment of the body parts from multiple images and by assuming small increments in pose changes. In [55], Yamamoto extended his previous work with Yagishita, and employed scene constraints to reduce the model's DOFs. Cham [56] developed a probabilistic multiple-hypothesis framework for tracking articulated objects, where the key insight was in the representation and tracking of the modes in the posterior state density function. Delamarre and Faugeras [57] presented a force-based method for tracking humans, which was based on an elaboration of the ideas of force-based tracking presented in [58]. Drummond and Cipolla [59] proposed a real-time visual tracking system, based on an internal Computer-aided design (CAD) model of the object to be tracked. The object is rendered using a binary space partition tree to perform the hidden line removal. Billon *et al.* [60] provided a real-time recognition method for motion capture data with the idea of reducing the mocap system to a single artificial signature. Bulbul *et al.* [61] provided a color based face tracking with mobile devices.

All of these problems are complex and hard to handle due to the huge dimensionality of the data. For example, it is very difficult to generate high quality in-betweens by interpolation for mesh data with thousands of DOFs. It would be difficult to constrain the interpolation to preserve desired object properties, say volume preservation. The flaws are spread over thousands of DOFs, and are difficult

to locate and correct. Motion editing techniques, such as constrained optimization methods [62], also face the difficulty caused by high dimensionality. Optimization has proven difficult for complex articulated character animations and those that require longer animations. It has been shown that we have more difficulties in setting the physical constraints in the torque-based optimization functions used for skeleton-based models. The data from mocap systems, which is represented as skeletons, always have many redundant frames that need to be simplified and compressed for reducing the size of the data. Only compressed mocap data can be accepted by a large motion database for further transmission and reuse.

1.2 Problem Statement and Research Methodology

We can now state the problem addressed in this thesis as follows:

Given that the high dimensionality of character animation in skeleton or mesh format leads to a number of difficulties in many of the existing techniques for manipulating and creating a desired character animation, there is a need for the development of new computer-based techniques which can enhance currently used character animation techniques and reduce animator effort required.

A very interesting observation is that people are capable of easily recognizing and distinguishing among different motions [63, 64]. This recognition ability is very powerful. No matter whether the information comes from, the real world or the virtual world, our brain can tell us what the motion is, easily and instantly. No matter how different the performer's personal style is, our brain can still distil the motion. This observation reveals a very fundamental and important fact, namely that, every motion has its distinguishable characteristics. These characteristics allow

us to tell them apart from other motions [65,66], so that no one will confuse a running motion with a walking motion or a jumping motion. Our research methodology is primarily based on this observation.

If we analyze the motion data, we can find that the data has two other important properties:

1. **DOFs are correlated with each other:** This is because the motion is coordinated; for example, the legs and arms work together to generate the required velocity in real world. So controlling the feet landing on the ground can therefore be represented as simple functions of just a few driving signals. Pullen and Bregler [67] used this observation for motion synthesis and texturing. Jenkins and Mataric [68] also used this observation for identifying behaviour primitives.

2. **DOFs have temporal coherence:** Motion with large intentional movements of physical contacts exhibits the greatest degree of feature coordination. However, the passive resting motion such as ambient change in a static stance has less-prevalent features. This property makes motion synthesis an interesting research problem, especially because there are physical limits to exploring the degree of differences between two subsequent frames in an animation sequence [19, 69]. Chai and Hodgins [70] utilized temporal coherence of the control signals to accelerate the nearest neighbor search for similar poses and dynamically constructed a local linear model for the poses to be estimated.

Based on the above discussion, we can conclude that most action animation sequences implicitly include the characteristic information of a particular action which is easily recognized by humans viewing the animation. In spite of this, the characterizing motion information is embedded and mixed with much redundant

information in such a way that makes it not so straightforward to extract this. The methodology we pursued is to use machine learning techniques to extract and represent this characterizing motion in such a manner that it can be further used in the development of new enhanced techniques.

1.3 Major Contributions

Our first major contribution is the formulation to extract characterizing motion information from a given animation character sequence motion involving 3D models. We use a manifold learning technique, local linear embedding (LLE) for this and obtain a representation of this characterizing motion in the form of a curve in low dimension embedding space. Next we have formulated a framework which includes this embedding curve and a reconstruction matrix that maps any point in the embedding space to a pose in the original high dimension space of the character. The mapped pose is computed using the entire set of frames used for computing the LLE embedding, thus eliminating any bias that may be caused by local information. This opens up the opportunity to create new techniques which operate in the low and/or high dimension space as appropriate. Moreover, unlike most previous work in the field of character animation, this framework is equally applicable to mesh or skeleton-based character animations.

Our other major contributions are based on the above framework and consist of new techniques which enhance character animation using the keyframe techniques. These are the following:

In-between generation with physical properties preservation: With just the keyframes, we use the framework to generate in-betweens which can satisfy physical properties. It uses a physical property map in the low dimension

embedding space for this. This technique releases animators from the difficult task of creating the key poses so that the physical properties are not violated by simple geometric interpolation techniques. Our framework has the ability to collect information from the entire animation represented in the form of a set of keyframes.

Variation generation in repetitive motions: With a given character animation, either the keyframes or the sequence, we use the framework of the characterizing motion in the LLE embedding space along with the reconstruction matrix to obtain a bilinear factor model for the character animation. Then we permit controlled perturbations in the reconstruction matrix to generate motion variations while preserving its principal characteristics. These variations in the motions of animated digital characters will make the different instances of the performed action appear slightly different for increased realism in the movements of autonomous characters like the non-playing characters (NPCs) in games.

Content based key information extraction: Given a complete character animation sequence, we extract an optimal set of keyframes using the framework for reconstructing the given animation through the keyframe animation technique. This set of keyframes forms a compact representation for the given animation and also permits manipulation of the animation using other techniques. For extracting keyframes, we use an animation saliency map computed in the high dimension space of the character's geometric model, and the low dimension embedding of the animation sequence obtained using LLE. While the animation reconstruction error is minimized in the original high dimension space, the search for the optimal keyframe set is made more efficient by

carrying it out in the low dimension embedding space.

1.4 Organization of this thesis

The rest of this thesis is organized as follows. In Chapter 2 we provide a comprehensive review of character animation techniques and the required background on manifold learning and its application in character animation. Chapter 3 presents the motion learning scheme using locally linear embedding and shows how it decomposes the animation into a core motion component and a variable motion component. The core motion component is in the form of a curve in low dimension embedding space and is referred to in the rest of this thesis as characterizing motion information. In the rest of this chapter, we describe the development of the framework which maps a point in the low dimension embedding space into a character pose in the original high dimension space of the character. In Chapter 4, we present our first new technique based on this framework to enhance keyframe animation. This chapter addresses the problem of generating in-betweens which satisfy physical properties such as volume, area etc. The concept of property maps in embedding space is introduced for this purpose. The next chapter addresses the problem that in the real world, two instances of the same action in different shots or scenes performed by the same actor will not be exactly identical. We present a technique to introduce noticeable variations in the motion of a character while preserving its principle characteristics. The bilinear factorization of the animation afforded by the decomposition discussed earlier is used for this. Chapter 6 presents a technique for optimized extraction of a set of keyframes, given an entire animation sequence previously generated or captured using mocap systems. In chapter 7, we give the concluding remarks on our research so far, and we list some possible directions for future work.

Chapter 2

A Review of Related Work

In this chapter, we give a detailed review of related work in character animation and manifold learning techniques.

In Section (2.1), we classify the literature on creation of character animation into three groups:

1. traditional character animations which require heavy involvement from animators, such as key framing, inverse kinematics and morphs;
2. character animations from motion capture systems;
3. character animations created by motion editing techniques applied to existing animation segments, such as motion synthesis, motion manipulation and motion blending.

In Section (2.3), we discuss relevant techniques for manifold learning, which we use in our work to cast the higher dimensional problem into a lower dimensional space. This discussion begins with traditional linear manifold learning techniques such as Principal Component Analysis (PCA), Factor Analysis and Multidimensional Scaling, and moves on to introduce the non-linear manifold

learning technique, which is relatively recent in the field of machine learning. For non-linear manifold learning, we give some details and comparisons about the three most popular techniques: locally linear embeddings, Laplacian Eigenmaps, and ISOMAP.

2.1 Creation of Character Animation

An ongoing goal in computer animation is to create realistic character animations. There are many ways to create character animations, but we can categorize them into two groups: the virtual world versus the real world. The former group is mainly based on animators' works. Usually animators in this group create $3D$ models and then have two choices: 1) to define the privileged information needed for animating the $3D$ model of a geometric and/or kinematic nature; 2) to provide physical data, such as forces, direction, mass and so on, to describe the motion and use a physics engine to do the simulation for determine the motion. In the second group, animation depends mainly on geometry capture systems, e.g., a motion capture system. Data is captured from a live performer carrying out the required motions. Both approaches are very costly in term of time and effort. A comparatively cheaper way is to create new animations from editing and/or manipulating existing ones. These methods are also referred as the data driven approach.

By categorizing character animation as coming from the virtual world, we mean that the animations are generated by animators using their imagination, visualization and creative abilities. The characters and their poses in the animation frames are created as per the animator's requirements. The motion control method could provide either local or global control. For locally controlled animated characters, the controlling methods are normally driven by geometric data. The motion

may be defined in terms of coordinates, angles and other shape characteristics or it may be specified using velocities and accelerations; but no forces are involved. Among the locally controlled animation group, key framing, morphing, kinematics (forward or inverse) are three of the more widely used techniques. Among these three methods, key framing is the basic one and morphing and kinematics methods are usually applied combined with the keyframe technique. For globally controlled animated characters, the motion of the character is determined by solving a group of dynamic equations, which are defined by physical parameters such as character mass, forces and constraints. These usually go by the name, physically based animation techniques.

Next, we give an overview of previous work on keyframe, morphing, kinematics and physically based animation techniques.

Key framing

Computer based key framing is an old technique consisting of the automatic generation of intermediate frames, called in-betweens, which are based on a set of keyframes (also called key poses for characters) supplied by the animator. These in-betweens are obtained by interpolating the keyframes. Linear interpolation is the most basic and common keyframe interpolation method used in the industry due to its most significant advantage, which is speed. Given two parameters p_0 and p_1, at time t_0 and t_1, an intermediate value is given by:

$$p(t) = \frac{t_1 - t}{t_1 - t_0} p_0 + \frac{t - t_0}{t_1 - t_0} p_1 \tag{2.1}$$

It can also be written as:

$$p(t) = p_0 \times (1 - t) + p_1 \times t \qquad t \in [0, 1] \tag{2.2}$$

The advantage of this technique, besides speed, is that the rate of change within a segment is constant. Therefore, it can be easily controlled. But, it also has disadvantages. The linear interpolation method produces undesirable effects, such as a lack of smoothness in motion, discontinuities in the speed of motion, distortions in rotations, non-preservation of physical properties like length, area, volume etc.

Spline interpolation methods can be used to reduce some of these drawbacks. Splines can be described mathematically as piecewise approximations of cubic polynomial functions. Two kinds of spline interpolations are very popular: interpolating splines with C^1 continuity at knots, and approximating splines with C^2 continuity at knots. For animation, the most interesting splines are the interpolating splines: cardinal splines, Catmull-Rom splines, and Kochanek-Bartels [71] splines.

Another way of producing a better result is to interpolate parameters of a model representing the object. This technique is called parametric keyframe animation and it is commonly used in most commercial animation systems. In a parametric model, the animator creates keyframes by specifying an appropriate set of parameter values. Parameters are then interpolated and poses for each frame of the animation are finally individually constructed from the interpolated parameters. Spline interpolation is generally used for the interpolation.

A new trend in the keyframe technique is to develop a nice user interaction scheme to make it easier for the animator to specify requirements and generate character animations. Terra [72] presented a novel method for capturing the user's desired timing, correlating it to the character's motion path, and adjusting the timing of the animation to reflect the user's desires. Igarashi *et al.* [36] also proposed to let novice users create animations for arbitrary 3D characters quickly and easily, with a standard input device such as a mouse.

Morphing

Morphing is a technique which has attracted much attention in the past two decades because of its astonishing effects [35, 73–78]. It is derived from shape transformations and deals with the metamorphosis of an object into another object over time. Researchers first applied the morphing technique on the image space. Image morphing manipulates two-dimensional images instead of three-dimensional objects and generates a sequence of in-between images from two images. These techniques have been widely used for creating special effects in television commercials, music videos, and movies. In generating an in-between image, the most difficult part is to compute warps for distorting the given images.

The problem of image morphing lies basically in knowing how an in-between image can be effectively generated from two given images. A simple way for deriving an in-between image is to interpolate the colours of each pixel between two images. However, this method tends to wash away the features on the images and it does not give a realistic metamorphosis. Hence, any successful image morphing technique must interpolate the features between two images to obtain a natural in-between image. Feature interpolation is performed by combining warps with the colour interpolation. A warp is a two-dimensional geometric transformation and the warping generates a distorted image when it is applied to an image. For two given images, the features on the images and their pixel based correspondences are specified by an animator with a set of points or line segments. Then, warps are computed to distort the images so that the features have intermediate positions and shapes. The colour interpolation between the distorted images finally gives an in-between image. More detailed processes for obtaining an in-between image are described by Wolberg [79].

A number of researchers studied morphing directly in a three-dimensional

space. The techniques fall into two major categories: the volume-based approach and the surface-based approach. The volume-based approach uses a 3D representation of the object, such as a set of voxels. Lerios *et al.* [80] proposed a 3D morphing method using fields of influence of 3D primitives such as points and lines to warp volumes. Cohen-Or *et al.* [74] proposed a distance field metamorphosis scheme with a guided interpolation by using a warping function. The surface-based morphing approach usually contains two main steps. The first one is to find the one-to-one correspondence between two polygon meshes; the second step involves defining the interpolation paths for each pair of corresponding vertices on the meshes. These paths are used to calculate the in-between shapes [73, 75]. Lee and Huang [81] presented an interactive system, which provides animators with easy morph control and fast morph creation.

Kinematic animation

Kinematic animation is a well established technique in the field of robotics and is used for controlling robot manipulators. Researchers in computer graphics have been using this technique to generate animations of multi-segment characters, such as animals and human figures. Its applications include: editing keyframe postures, generating interactive animations, and generating animations for actions such as reaching, walking etc.

Kinematic animation is comprised of two classes of techniques: forward kinematics and inverse kinematics. Forward kinematics consists of specifying the state vector of an articulated figure over time. This state vector is usually specified for a small set of keyframes, while interpolation techniques are used to generate the in-between frames. The main problems lie in the design of convenient keyframes,

and in the choice of adequate interpolation techniques. Designing keyframe skeleton poses usually lies in the animator's hands, and the quality of resulting motions greatly depends on the animator's skills, and, in many cases, the available physical and biomechanical knowledge of the motions. The exclusive use of forward kinematics makes it difficult to add constraints to the motion, such as those specifying that the feet should not penetrate into the ground during the support phases. These constraints may be solved by using inverse kinematic technique, discussed next.

Inverse Kinematics (IK) permits direct specification of the end point position. It has been widely used for synthesizing motions of linked bodies in robotics and in character animation [82]. The basic IK problem is to find the character pose that satisfies the specified constraints. Among the number of IK methods, the method based on Lagrange multipliers is known to be the best, as its computational cost increases only linearly with the number of degree of freedoms (DOFs), and the motions generated are natural. However, the IK technique has a big drawback, as it is often an under-constrained problem. Many action poses may satisfy the specified constraints. This leaves an animator with the task of specifying significantly more constraints than necessary. Inequality constraints are necessary when solving problems for multi-body structures that have joints with limitations in the ranges of motion, or collisions between different segments. The other problem is that computation time for IK calculation grows in the order of time cubed proportional to the number of constraints. Hence, when controlling multiple characters under multiple constraints, the performance of IK drops significantly. Grochow *et al.* [83] proposed an IK system that utilizes captured motion data to address this problem.

This forms part of the more recent direction in IK, combine IK with motion capture to develop IK solvers, and to let animators or users edit or control the human

characters interactively. The interaction is based on captured motion data [84]. More recently, researchers have applied IK technique directly on mesh data, which has a much higher number of DOFs [47, 49]. But these methods always need to pre-group the vertices into bigger patches to reduce the complexity of the problem.

Physically-based Animation

Unlike the local motion control methods, physically based character animations handle the motion globally. In methods such as described in [43, 85, 86], animators provide physical data for the character and the environment, and the motion is obtained by solving the dynamic equations. We may distinguish between methods based on parameter adjustment versus constraint based methods. For the latter, the animator states, in terms of constraints, the properties that the character model and the environment is supposed to have, without needing to adjust the parameters. For example, space-time constraints may be used to create character animation, by solving constrained optimization. For realistic simulation of deformations, an elastic model based on the Lagrange equation may be used. A finite-element can model the deformations of human flesh due to the flexion of body parts and the contact among body parts. In physically-based animation, collision detection and response are obviously important.

Physically-based character animation is based on the computation of a character's motion within a physically simulated environment, and the joint torques [1] are generated to cause a character movement. Animating certain aspects of real movement, such as how quickly a character tumbles in the air or how a character reacts as it falls on the ground, are directly governed by the physics of the situation and can be well captured. However, the robust controls for character animation,

[1] joint torques is the torque produced by the muscle force about the joint center [87]

such as human or animal actions, are very hard to define. The controls for character animation can even be undefinable. It is because of the fact that the interface between our intentions and muscle actions is unobservable and complex [86].

In the methods mentioned above, the goal is to help animators to convert their observations of real world motion into virtual world animations. Next, we will discuss motion capture systems which can acquire character animation data by sensing and recording movements in the real world.

2.1.1 Motion Capture

As discussed briefly in the previous chapter, motion capture (mocap) is a technique for sensing and digitally recording the movements of live performers directly. It started as an analysis tool in biomechanical research. Since it's invention, mocap has grown to be increasingly important as a source of motion data for character animation with applications in medicine, sports, education, training and recently for both cinema and video games. It employs special sensors, called trackers, to record the motion of a human performer (See Fig. (2.1)) [2]. By using magnetic or optical technologies, it is possible to store the positions and orientations of points located on the human body. The typical procedure for a mocap system is listed as follows:

1. Studio set-up
2. Calibration of capture area
3. Capture of movement
4. Clean-up of data

[2]The image is from Ascension Technology Corporation.

5. Post-processing of data

A further computation provides the link between the synthetic skeleton and the real skeleton, in order to adapt data to new character morphologies. Several techniques [18,88] have been introduced to adapt captured trajectories to a different synthetic skeleton. The method consists in recovering angular trajectories, which are applied to a synthetic articulated body. Given sensor positions and orientations, a modified inverse kinematic optimization algorithm is used to produce the desired joint trajectories. The synthetic skeleton thus exhibits exactly the same motion as the real actor. For most applications however, the captured motion needs to be modified in order to create a variety of specific animations, that take the synthetic environment into account. For instance, when an interaction between two synthetic actors is required, their movements have to be modified to model this interaction (e.g., by dealing with problems of contacts, trajectory tracking, etc.). Overall, since motion data is obtained from real world performers, the resulting animations are very realistic.

However, the mocap technique also has its weaknesses. The acquisition of data for mocap systems is still a challenging problem - setup time is large, performing naturally with markers in a synthesized environment needs practice and training and the data obtained is very noisy. Secondly, motion capture data contains large quantities of unstructured data. This huge amount of unstructured data is cumbersome to manipulate. Both of the above weaknesses make the mocap data difficult to alter, especially since the key "essence" of the performer's actions are not easily distinguished from the large amount of potentially irrelevant details.

As is clear from the above discussion, the creation of new animations either by using keyframe animation or by using a mocap system is very costly in efforts

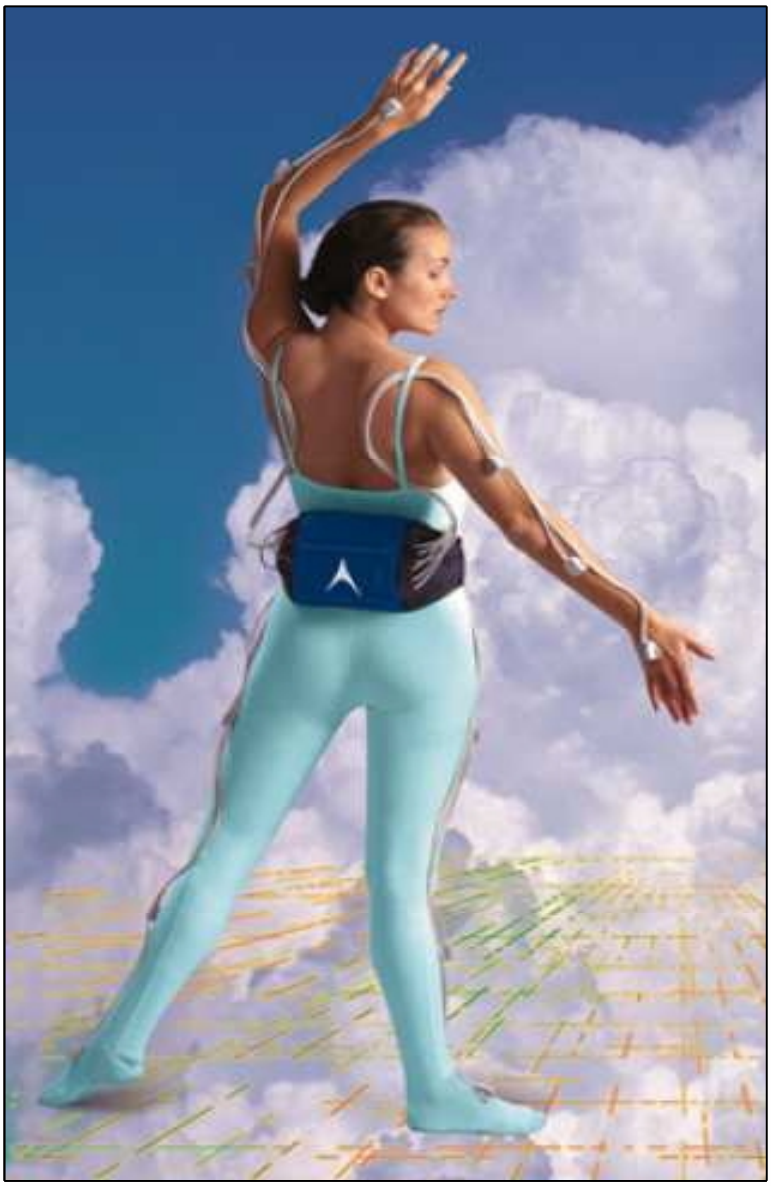


Figure 2.1: Motion Capture: a performer wearing a motion capture apparatus. The device shown is a full body magnetic tracking system with a wireless interface. The Image was take from the Ascension Technology Corporation [3].

and time. A cheaper way as mentioned earlier is to create new animations from editing and/or manipulating existing ones. These methods are also referred as the data driven approach. We will discuss this group of methods next.

2.1.2 Data-driven motion generation

These techniques are based on reuse of existing character animation segments from the motion database. Such techniques are called as data driven approaches. One popular data-driven approach to obtain animations are motion graphs, which represent the allowable transitions between poses in two actions [19, 21, 89–91]. Motion graphs are used to create an animation by taking pieces from a motion database and then reassembling them to form a new motion. However, graph-based representations cannot be used to generate motion data for new styles and/or actors because they are not used to modify the captured motion data. One major stumbling block is the lack of understanding of the effects of the motion graph's data structure. For example, it is not known with any confidence about what a character can or cannot do when animated by a particular motion graph in a particular environment. Due to this lack of knowledge of a motion graph's capabilities, these graphs are not reliable enough for many applications, especially interactive applications where the character must always be in a flexible and controllable state, regardless of the control decisions of the user. Even if a particular motion graph does happen to fulfill all of the requirements, the reliability will be unknown. For a risk-averse application, the result is un-useable without a method to evaluate and certify the capability of the motion graph. Reitsma and Pollard [92] proposed the definition of a motion graph's capability, described a method to quantitatively evaluate that capability, and presented an analysis of some representative motion graphs for this problem with some limitations.

Another way to create character animation are weighted interpolations of registered motion examples [31, 93–96] using the idea that human motion itself can be represented as a weighted interpolation. The new character animation can be generated by interpolating motion examples with different styles but it fails to model motion variations. In computer vision area, Troje [97] explored a similar representation for walking motions, in which the temporal variations in poses were expressed as a linear combination of sinusoidal basis functions. The temporal variations in poses were used to recognize genders from optical mocap data.

In the past decade, statistical motion models have been used for: interpolation of keyframes [98]; interactive posing of 3D human characters [83]; performance animation from low-dimensional control signals [70]; generalization of motion to match various forms of spatial-temporal constraints specified by the user [99]; and interactive motion synthesis with direct manipulation interfaces and sketching interfaces [100]. However, none of these approaches interprets motion variations using style or identity factors.

2.2 Character Animation Editing

In this section we shall discuss various techniques devised for editing of character animations.

2.2.1 Keypose Editing

Because of the increasing requirements for realism in computer graphics and the wide availability of 3D scanners, animations of character meshes with high geometric complexity are becoming commonplace. Keyframe techniques, the fundamental category of techniques employed in practice, use shape interpolation among local neighbors [71, 101]. Most often, the high complexity is reduced by interpolating the

skeletons (bone-structure). This is then followed by a "skinning" operation that deforms in-between mesh vertices according to a weighted average of bone transformations. However, using a simple skinning model makes it very difficult to capture articulated shape deformation without significant visual artifacts [94].

Some researchers have extended the skeleton animation ideas to mesh based animation. They create realistic animations from examples. One approach is to warp an existing animation [13, 102] or to interpolate between sequences [93]. Mesh surface deformation has been studied as a geometric problem [103, 104], or using inverse-kinematics [105], or in the form of a physics-based problem [106], where the emphasis is on computing realistic shape changes based on physical properties and applicable forces. Usually, the deformations are small and while such techniques find application in facial animation, they cannot be easily extended to limbed character animation [47].

Over the last two decades, a number of techniques for easing the problem of keypose specification have evolved, mainly through interactive keypose editing. Authors have described systems for producing new keyposes from examples, either by direct copying and blending of poses [89] [69] [19], by learning a likelihood function over sequences [98], or through inverse kinematics on reduced models [49]. The importance of preservation of shape properties such as volume are addressed in [107]. Although the keyframe interpolation based paradigm is presently most popular in animation practice, there still exist the inherent problems due to the use of merely local neighborhood information to generate the new poses. Hence considerable human intervention and care are needed for obtaining the desired motion. In applications like cinema and gaming, where most motions are predefined, the quality of the final result is often used to justify the cost of manual labor [108, 109].

2.2.2 Creating Motion Variations

The problem of creating varying motion sequences in a controlled fashion has received considerable attention in past research. There are two main categories – those which work on modifying the frames in a given single motion and others that work on generating varying motion sequences by composing motion fragments.

Variations in a given motion

These are techniques analogous to adding texture to images/surfaces. Variations are often generated through the addition of noise functions, such as Perlin-noise [110] or hand crafted noise functions based on biomechanical considerations [111]. Frequency analysis of motion and subsequent addition of noise (texture) has also been another approach. Unuma *et al.* [112] use Fourier analysis to manipulate motion data by performing interpolation, extrapolation, and transitional tasks, as well as to alter the style. Bruderlin and Williams [113] apply a number of different signal processing techniques to motion data to allow editing. Pullen and Bregler [114] create cyclic motions by sampling motion signals in a 'signal pyramid'. Lee and Shin [115] develop a multi-resolution analysis method that guarantees coordinate invariance for use in motion editing operations such as smoothing, blending, and stitching. Similarly, statistical analysis based techniques which are usually based on principal component analysis have also been proposed [116–118].

Yet other research in creating motion variations is in the area of editing a given motion to adapt to different constraints while preserving the style of the original motion. Witkin and Popović [102] warped motion data between keyframe-like constraints set by the animator. Motion clips are combined by the overlapping and blending of the parameter curves. They showed that whole families of realistic motions can be derived from a single captured motion sequence using only a few

keyframes to specify the motion warp. The physically based space-time constraints method of Witkin and Kass [62] was applied to adapt a set of motion data to characters of different size. Popović and Witkin [119] describe a physics based method in which editing is performed in a reduced dimensionality space. In [120], Sun and Metaxas provide different solutions for automatic gait generation based on the use of sagittal elevation angles, uneven terrain handling and high level control over path specification. Their work is targeted towards easy-to-use, real-time, and fully automated animation system, specifically for walking motion.

Variations through Motion Fragment Composition

Analogous to the video texture concept [121], Sattler *et al.* [122] propose an algorithm to create new user controlled animation sequences based only on a few keyframes by the analysis of velocity and position coherence. The simplicity of the method is achieved by carrying out the calculations on the main principal components of the reference animation, thus reducing the dimensionality of the input data. Brand and Hertzmann [123] have used hidden Markov models along with an entropy minimization procedure to learn and synthesize motions with particular styles. In [98], motion data is divided into motion *textons*. A statistical model is learned from the captured data which enables the realistic synthesis of new movements by sampling the original captured sequences. Motions are synthesized by considering the likelihood of switching from one *texton* to the next.

Another approach is to search an existing database of motion fragments to produce new motions driven by parameters such as speed or style of motion [124]. In the work of Pullen and Bregler [67], the animator sets high level constraints and a random search algorithm is used to find appropriate pieces of motion data for the "joins"; the frames in the pieces that blend one motion fragment to another.

Similarly, missing degrees of freedom in a motion are fetched from a motion capture database. In the work of Lee *et al.* [90], animations are created by searching through a motion data base using a clustering algorithm. Kovar *et al.* [19] introduced the concept of a motion graph which contains original motion and automatically generated translations. Hus *et al.* [125] present an example based human motion generator by interpreting input control specification to create a designed target motion. More recently, Shin and Oh [126] have presented the idea of "fat graphs" for user controlled character motions.

2.2.3 Keyframe extraction

Due to the increasing popularity of motion capture technology, various methods have been proposed for keyframe extraction. The focus is on extracting keyframes from skeletal animation sequences. In [127–129], keyframe extraction is addressed as the high dimensional motion curve simplification problem. The extracted keyframes are the junctions between curve segments. Lim and Thalmann [127] divided the motion curve into segments recursively until the maximum distance of any point on the segment piece is less than a certain prescribed value. Li *et al.* [128] provided another algorithm to simplify the motion curve by decimating the less important frames iteratively. Halit and Capin [129] also select high saliency frames as keyframes on the motion curve but in a lower dimensional embedding space computed using the PCA technique. Curve simplification methods usually are very efficient. However, considering that the above approaches focus on local motion curve segment, the results are not optimized globally.

Other keyframe extraction works convert the keyframe extraction to a clustering problem. This group of methods, first applied on videos [130, 131], cluster frames with defined distance measures. A representative frame will be selected as

the keyframe from each individual group. Liu *et al.* [132] use a weighted function of distance between joints as the distance measure to cluster frames. After the clusters are formed, the first frame in each cluster is selected as the keyframe. Park and Shin [133] used quaternions as their representation for motion data. Then they utilized PCA and k-means clustering to linearize the quaternions and cluster them. The scattered data interpolation was used to extract keyframes from the clustered motion data. How to incorporate the frame order (time-wise) to generate less reconstruction error is still a challenging problem for this type of approach.

Yet another group of keyframe extraction methods represent the animations as matrices and converts the keyframe extraction problem to a matrix factorization problem. Huang *et al.* [26] convert a sequence of frames to a matrix formed by placing all of the vertices of a frame in a row. This matrix is then approximately factorized into a weight matrix and a keyframe matrix. The optimal solution to the matrix factorization equation generates the keyframes and the combination weight for in-betweens. Similar to [26], Lee *et al.* [28] introduced the deformation-driven genetic algorithm to simplify the optimization of good representative animation keyframes. The disadvantages of this type of method is that they are very time consuming due to the high dimensionality of DOFs.

From the above detailed review of character animation techniques, it is clear that the subject has been researched extensively. A lot of effort is required for creating animations as per animator expectations. In spite of the numerous techniques which have evolved, there is still the need for new techniques which can reduce the amount of animator effort involved and address many of the difficulties arising due to the high dimensionality of character animation data. The idea which we have pursued in this research is to use manifold learning techniques to automatically

identify the distinguishing characteristics of any given character motion and then to use this for developing new techniques. We shall therefore give next an overview of applicable manifold learning techniques.

2.3 Manifold Learning

In this section, we give an overview of relevant techniques in manifold learning. The components of the data points tend to be correlated with each other in many real-world applications with high-dimensional observation data, such as images, videos and animations [67]. Also in many cases, the data points lie close to a low-dimensional nonlinear manifold. Manifold learning addresses the problem of finding a low-dimensional structure within collections of high-dimensional data to overcome the difficulties caused by high-dimensionality for human understandings.

In the last several decades, manifold learning has been studied and used in many practical applications, such as data classification and data mining. These studies have lead to many impressive results about how to discover the intrinsical features of a manifold. Manifold learning is based on an assumption that data sources with a large amount of data, such as images, animations, speech and characters with varying intrinsic principal features, can be thought of as constituting highly nonlinear manifolds in the high-dimensional observation space.

Traditional dimension reduction techniques such as Principal Component Analysis (PCA), Factor Analysis (FA) and Multidimensional Scaling (MDS), usually work well when the underlying manifold is a linear (affine) subspace in the input space [134]. Hence, we cannot, in general, discover nonlinear structures embedded in the set of data points with PCA, FA and MDS.

Recently, several entirely new approaches have been devised to address this

Table 2.1: Comparison among two groups of recently successful manifold learning methods, global vs. local.

Manifold Learning Approach	Advantages	Disadvantages
Local approach (LLE and Laplacian Eigenmaps)	1) Computational efficiency due to sparse eigenvalue problem 2) Representational capacity	Results depend on neighbourhood parameter
Global approach (ISOMAP)	1) Gives more faithful representation of global structure 2) Its metric-preserving properties are better understood theoretically	Not stable for topological structure

nonlinear mapping problem. These methods combine the advantages of PCA and MDS, such as computational efficiency, few free parameters, and non-iterative global optimization of a natural cost function, with the ability to recover the intrinsic geometric structure of a broad class of nonlinear data manifolds.

We can classify these algorithms into two groups: local approaches and global approaches. Locally Linear Embedding (LLE) [135] and Laplacian Eigenmaps [136] are local approaches which attempt to preserve the local geometry of the data; essentially, they seek to preserve the neighborhood on the embedding space. On the other hand, the global approach, as in ISOMAP [137], attempts to preserve the geometry at all scales, by mapping nearby points on the manifold to nearby points, and far away points to far away points in the embedding space.

Table (2.1) lists the advantages and disadvantages of local and global approaches used in manifold learning. Following that we discuss these approaches in some detail to understand their relative advantages and disadvantages specifically for our research goals.

To give a mathematical description, manifold learning can be defined as follows: Given a set of n observation data points $\mathbf{x} :< x_i, x_2, ..., x_n >$ in space R^l, we can hypothesize that there is a set of points $\mathbf{y} :< y_1, y_2, ..., y_n >$ in space R^m, where

$m << l$, which is a mapping representation of $\mathbf{x}$. In other words, we assume that there is a smooth mapping of $\rho : R^l \rightarrow R^m$. Dimension reduction techniques are ways to find the smooth mapping ρ. In the next subsections, we give some details about three nonlinear dimension reduction techniques: LLE, Laplacian Eigenmaps, and ISOMAP.

2.3.1 Local Approach: LLE

Locally Linear Embedding (LLE) [135] is one of the many promising frameworks for nonlinear dimensionality reduction. LLE frameworks were shown to be able to embed nonlinear manifolds from high dimensional data into low-dimensional Euclidean spaces. Such approaches are able to embed mesh ensembles nonlinearly into low dimensional spaces, where various orthogonal perceptual aspects can be shown to correspond to certain directions or clusters in the embedding spaces. In this sense, such nonlinear dimensionality reduction frameworks present an alternative solution to analyze the object's motion.

For a given data set $\mathbf{x}$, to compute the embedding, the LLE method uses three steps (See Fig (2.2)):

1. For each $x_i \in \mathbf{x}$, find the k nearest neighbors x_j of x_i by defining the distance function (such as Euclidean distance), k is a parameter.

2. Compute weights w_{ij} such that the reconstruction cost $\left\| x_i - \sum_j w_{ij} x_j \right\|^2$ is minimized; subject to the constraint $\sum_j w_{ij} = 1$.

3. Compute the low dimensional vector Y (the embeddings) reconstructed with w_{ij} to the minimize the cost function: $\left\| y_i - \sum_j w_{ij} y_j \right\|^2$, subject to the constraint $\left\langle y_i y_i^T \right\rangle = 1$.

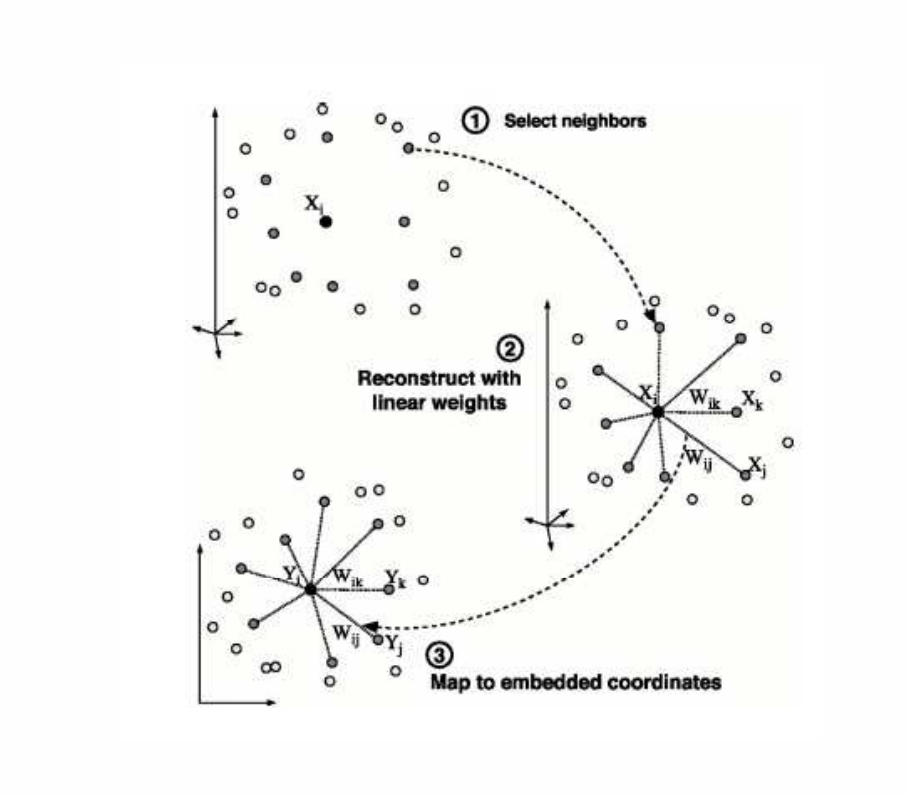


Figure 2.2: Summary of the LLE Algorithm, which performs the nonlinear dimension reduction via local linear reconstruction of weights. The image was taken from Saul and Roweis' work [4]

2.3.2 Local Approach: Laplacian Eigenmaps

In 2003, Belkin and Niyogi [136] explored an approach that builds a graph which incorporates the neighborhood information of the data set. Using the notion of the Laplacian of the graph, we then compute a low-dimensional representation of the data set. This representation optimally preserves local neighborhood information in a certain sense. The representation map generated by the algorithm may be viewed as a discrete approximation to a continuous map. The map naturally arises from the geometry of the manifold.

For a given data set $\mathbf{x}$, there are three steps of the Laplacian Eigenmaps computation listed as follows:

1. For each x_i in $\mathbf{x}$, find $\Gamma(i)$, the n nearest neighbors x_j in $\mathbf{x}$ of x_i by the defined distance function, such as Euclidean distance, subject to it being symmetric ($i \in \Gamma(j)$ if and only if $j \in \Gamma(i)$).

2. Construct weighted adjacency matrix W in equation (2.3) below, which is symmetric, where $W_{ij} \neq 0$ if and only if $i \in \Gamma(j)$.

$$W_{ij} = exp\left\{-\frac{1}{2\delta^2}\left\|x_i - x_j\right\|^2\right\} \tag{2.3}$$

   where, δ sets the scale of the isotropic Gaussian kernel.

3. Compute embedding from normalized Laplacian with embedding Y subject to minimization of the following function:

$$\Phi_{lap}(Y) = \sum_{i=1}^{n}\sum_{j\neq i}\tilde{W}_{ij}\left\|y_i - y_j\right\|^2 = \frac{1}{2}trace(Y^T(I-\tilde{W})Y) \tag{2.4}$$

2.3.3 Global Approach: ISOMAP

ISOMAP [137] generalizes low dimensional embedding based on replacing the Euclidean distance by an approximation of the geodesic distance on the manifold. The calculation of embedding for ISOMAP also contains three stages for given dataset $\mathbf{x}$:

1. For each x_i in $\mathbf{x}$, determine a neighborhood graph G of the observed data $\mathbf{x}$ in a way, such that G contain edge $x_i x_j$ iff $\|x_i - x_j\| < \epsilon$.

2. Compute the shortest paths in the graph for all pairs of data points. Each edge $x_i x_j$ in the graph is weighted by its length, such as $\|x_i - x_j\|$.

3. Apply multi-dimensional scaling method [138] on the resulting shortest path to distance matrix D, to determine the new embedding in low dimensional space.

Fig. (2.3) illustrates the non-linear dimension reduction result from 600 dimensions into 2 dimensions for the benchmark data set S-curve. All three methods can successfully achieve the reduction goal.

In our research on learning the characterizing motion information in a given character animation, we have chosen to adopt LLE. In the next chapter (3), we discuss in detail why we choose LLE over the other methods and how we use it in characterizing the motion in a set of keyframes. We also demonstrate its applicability with the help of a number of experimental results from our implementation.

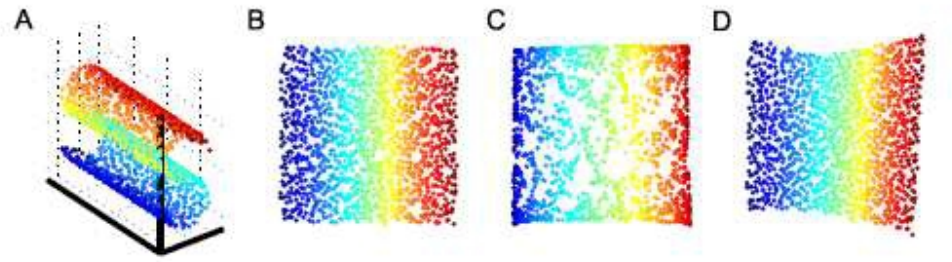


Figure 2.3: Embedding Result: A) Shows 600 points sampled from the S-curve. B) to D) demonstrate: Two-dimensional embeddings result, from B) ISOMAP , C) Laplacian Eigenmap , and D) LLE. The figure was taken from the work of Ham *et al.* [5]

Chapter 3

Motion Learning Scheme with Locally Linear Embedding

In this chapter, we present our main formulation which uses the *locally linear embedding* method to project the high dimensional motion data into a lower dimensional space. This method will allow us to extract the characterizing motion information for further usage, such as:

- Generating in-between frames satisfying physical properties. By using the given keyframes of a motion, we find the motion curve embedded in low dimensional space. Then we build up a physical property map in the embedding space, which can help us generate the in-between frames with desired physical property values. See details in Chapter 4.

- Manipulating motions. Given a complete animation sequence, we use LLE to find the motion curve in the embedding space for this animation. Then, we analyze the mapping matrix between the embedding space and the original space, with a Generalized Radial Basis Function. We apply singular value decomposition to the mapping matrix and use the decomposition result to manipulate the postures to create variations in this animation. See details in Chapter 5.

- Extracting a keyframe representation. Again, given a complete animation sequence, we use the LLE embedding in low dimensional space to define a subset as keyframes, which forms a compressed representation of the input motion. See details in Chapter 6.

The organization of this chapter is as follows: After a brief introduction about how we can apply motion learning to analyse character animation in Section (3.1), we give the detailed description about our motion learning scheme in the embedding space with LLE method in Section (3.2). In Section (3.2), we focus on the following: 1) How do we cast the LLE in our motion learning scheme; 2) How do we select the required user input parameters to obtain reliable results; and 3) We present a number of positive experimental results. In Section (3.3), we introduce the reverse mapping (from the embedding space to the original animation space) using the Generalized Radial Basis Function (GRBF).

3.1 The Motion Curve

As mentioned in Chapter 1, an animation sequence contains the characterizing motion information which distinguishes the motion present in this animation from other motions independent of the data type used for recording the animation. Our primary goal is to extract this characterizing motion information and to interpret it to enhance keyframe techniques for character animation. We consider all the given input poses as providing the global motion information that should contribute to every other pose's configuration. For this, we would first like to learn/discover the motion recorded in a character animation as a characterizing motion curve, by using all the significant information present in the given set of poses. This characterizing motion curve would be identical and/or contain identical information for different

animations containing the same type of action. If we consider that the character animation records a character performing a certain activity or gesture, then the shape of character's body changes over time. The changes can be observed by the changing of the DOFs. These deformations are constrained in two ways: by the physical body constraints and by the temporal constraints, posed by the action being performed. Due to the size of the DOFs, the motion curve is in a high dimensional space and the curve is beyond the perception of users.

Finding an accurate definition of the characterizing motion curve in a high dimensional space from the discrete sample poses is not only very hard, but also unnecessary. In many real-world applications with high-dimensional data, the components of the data points tend to be correlated, and in many cases the data points lie close to a low-dimensional nonlinear manifold. Earlier in Section (2.3), we have reviewed three nonlinear dimension reduction techniques, LLE, Laplacian Eigenmaps, and ISOMAP. In this chapter, we present how we incorporate LLE in our research to provide a motion learning scheme, in which we analyze the given animation data for recovering the characterizing motion information in low dimension embedding space. We provide detailed information about using LLE in our research in the following sections.

3.2 Motion Learning in Embedding Space

Given the assumption that each data point and its neighbors lie on a locally linear patch of the manifold, each point can be reconstructed as a weighted combination of its neighbors. The objective is to find the reconstruction of weights that minimizes the global reconstruction error. An optimal solution for such an optimization problem can be found by solving a least squares problem. Since the recovered weights

reflect the intrinsic geometric structure of the manifold, an embedded manifold in a low dimensional space can be constructed by using the same weights. The embedding is determined by solving the optimization problem: for a given set of points, we need to minimize the reconstruction error with fixed weights.

Compared to other methods, LLE has the following advantages:

i) Since the weights w_{ij} are symmetrical, for any particular data point, they are invariant to rotation, re-scaling and translation of the data points and their neighbors. By enforcing $\sum_j w_{ij} = 1$ (See Equation (3.2)), the solution also achieves translation invariance.

ii) More importantly, by assuming the local linear transformation, LLE discovers the intrinsic characteristic present in high dimensional data.

iii) LLE leverages overlapping local information to uncover the global structure. This advantage is achieved by successively computing different dimensions in the embedding space and by computing the bottom eigenvectors from Equation (3.3), one at a time.

Therefore, we have adopted this LLE framework to embed the deformation of mesh data nonlinearly into a lower dimensional space. We consider that our input dataset consists of N frames $F = \{\mathbf{f}_0, \mathbf{f}_1, ...\mathbf{f}_N\}$ in the desired animation.

1. For each $\mathbf{f}_i \in F$, we form the subset $F_i' = \{\mathbf{f}_j \in F | \mathbf{f}_j \neq \mathbf{f}_i$, and $\mathbf{f}_j$ is one of the k nearest neighbors of $\mathbf{f}_i\}$, where k is a parameter specified by user.

2. For each $\mathbf{f}_i \in F$, we build the reconstruction weights, w_{ij}, with its k neighbors, forming the $n \times k$ matrix W, with the minimal reconstruction error,

$$\epsilon_i = \Delta(\mathbf{f}_i, \tilde{\mathbf{f}}_i) \tag{3.1}$$

where n is the degree of freedom, and $\Delta(\cdot)$ is a function to measure the difference between two meshes. Then $\tilde{\mathbf{f}}_i$ is a reconstruction of $\mathbf{f}_i$ using its k nearest neighbors:

$$\tilde{\mathbf{f}}_i = \sum_{j=1, \mathbf{f}_j \in F_i'}^{k} w_{ij}\mathbf{f}_j, \qquad \sum_{j=1}^{k} w_{ij} = 1. \tag{3.2}$$

3. Compute the embedding based on the reconstruction weights w_{ij}. LLE converts the minimization problem to an eigenvalue problem. The optimal embedding is the eigenvectors of the symmetric, sparse matrix M:

$$M = (I - W)^T(I - W) \tag{3.3}$$

Where I is the identity matrix.

Figure (3.1) shows the results of applying the LLE method to the data of a keyframe motion sequence that depicts a horse galloping. The horse galloping data has 12 key poses, each defined as a mesh with 8431 vertices, i.e., a total of 25293 DOFs. The mapping in two-dimensional space clearly shows the inherent relationship amongst the meshes in the galloping motion.

In the LLE embedding calculation, there is a user-controlled parameter k. In our experiments, we found that the value of k affects the embedding result. In the next subsection, we provide a discussion about the selection of parameter k.

3.2.1 LLE Parameter Selection

When we used LLE to obtain the motion content, we noticed that the LLE method results are highly dependent on the parameter k, which is the neighbor set size used to approximate every individual frame when we calculate the embedding. A poor choice of k could result in singularities of the matrix M in Equation (3.3). Very small or very large values of k are both bad choices, and both are contrary to the

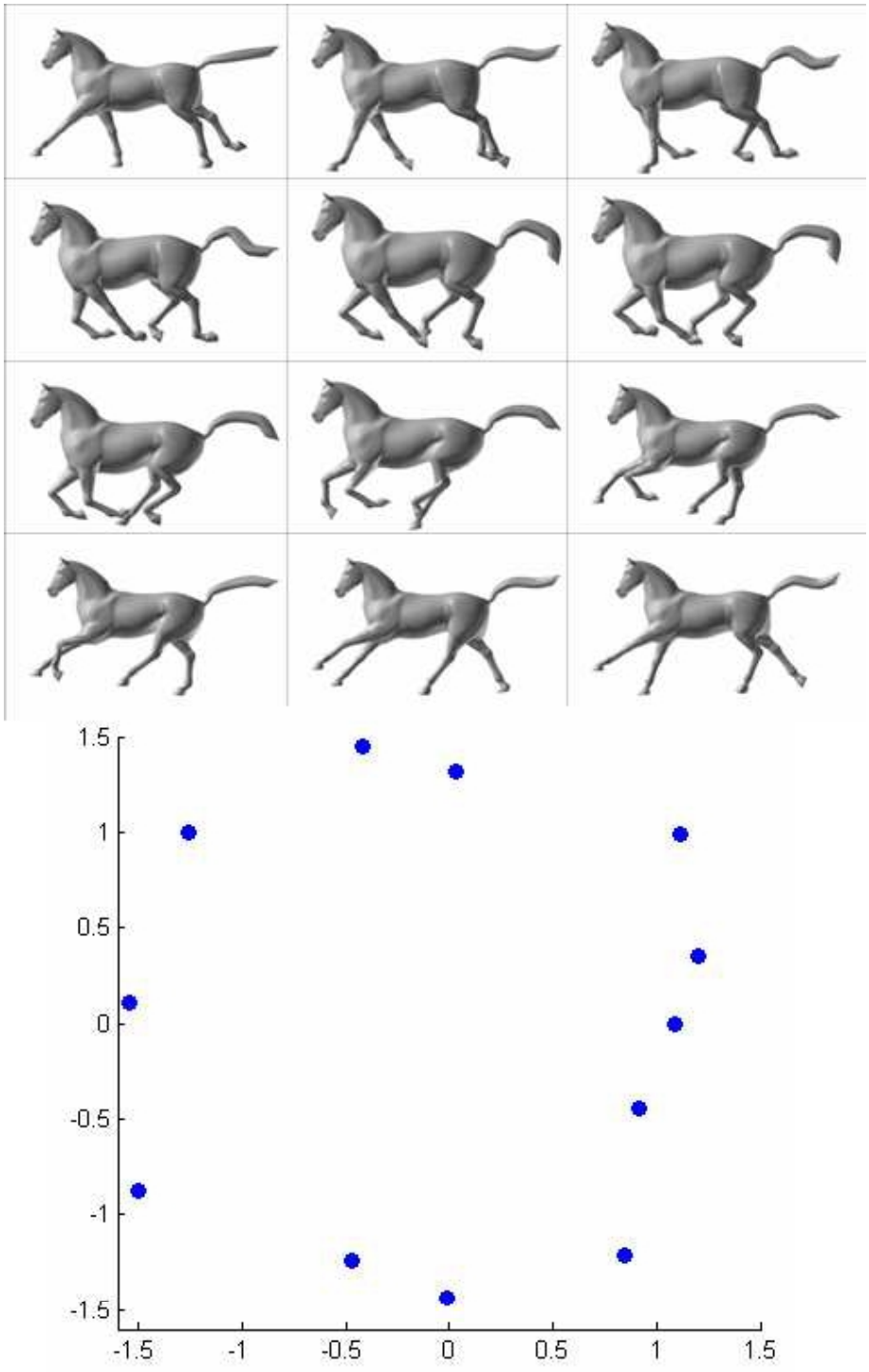


Figure 3.1: Illustration of unsupervised learning: A small set of 12 key poses (motion of a horse galloping) is mapped into 2D embedding space. The embedding is cyclic as can be expected for any periodic motion.

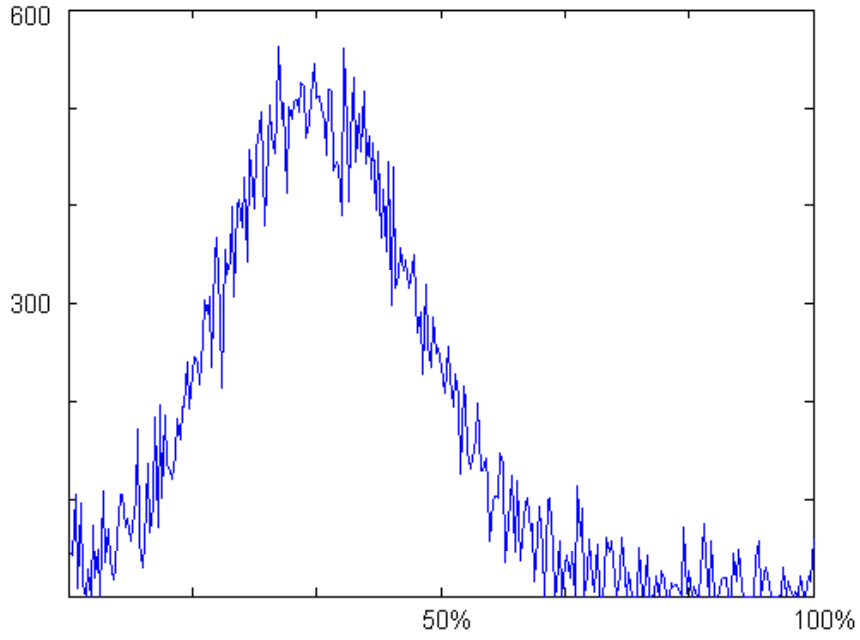


Figure 3.2: The value of k affect the result of embedding. The x-axis reflects the value of k (represented in percentage of the number of frames); The y-axis reflects the number of times we obtain reliable embeddings.

local neighbourhood idea of LLE. A very small value of k, such as less than 5% of the input size, gives the LLE a calculation bias by having less correlation information. Secondly a very large value of k, such as over 50% of the input size, is also contrary to the assumption of LLE, such that when when we look at every sample close enough, then we can notice the linear structure. With our experiments, we conclude that $[15\%, 35\%]$ is an effective range of k. As shown in Figure (3.2), we perform a group test to see the safe range of parameter k. First of all, we selected 584 motion sequences in total. Then, we performed LLE with the $k \in [1\%, 100\%]$, shown in the x-axis. We put the number of successful embeddings on the y-axis. The figure shows the range $[15\%, 50\%]$ could give us a reliable embedding result; however, based on the assumption the number of neighbors should be small to maintain a locally linear embedding, we set $[15\%, 35\%]$ as an effective range instead.

3.2.2 Experimental results

In this section, we describe experimental results to show the performance of our motion learning scheme with LLE. We carried out a number of experiments on different types of motions with multiple data types (see Table (3.1)). The skeleton animation data was obtained from Graphic Lab, Carnegie Mellon University [1]. and the mesh animation data from the work of Dr. Robert W. Sumner and Dr. Jovan Popovic [2]. Our implementation works equally well on both skeleton and mesh animations. It supports both traditional animation format as *.obj* file and new popular MD5 format [3]. For demonstration purposes, the dimensionality of embedding space was set to three for skeleton models and two for mesh models. In the real application, the dimensionality of embedding space can be increased. With the experiments, we demonstrate that our motion learning scheme with LLE has the following abilities:

1) Our motion learning scheme with LLE extracts distinct shapes of embeddings for specific actions.

2) Our motion learning scheme with LLE can handle motion analysis for a given character animation sequence even with a shuffled frame order.

3) Our motion learning scheme with LLE can extract the characterizing motion information even when a small number of frames are missing.

4) Our motion learning scheme with LLE can be applied when multiple character animation sequences are put together.

---

[1] We gratefully acknowledge the Computer Graphics Group at CMU, for building the mocap database, which is available on their website: http://mocap.cs.cmu.edu/

[2] We gratefully acknowledge Robert Sumner and Jovan Popović from the Computer Graphics Group at MIT, for the animated mesh data of the galloping horse and collapsing camel available on their website: http://people.csail.mit.edu/sumner/research/deftransfer/data.html

[3] MD5 is a open format specific to Doom 3. MD5 allows models to be animated by the use of an internal skeleton.

Table 3.1: Test data sets.

Name	Type of data	# of frames	# of DOF
character 1 running	skeleton	170	62
character 1 walking	skeleton	160	62
character 1 jumping	skeleton	150	62
character 2 walking	skeleton	155	62
character 2 running	skeleton	165	62
character 3 walking	meshes	31	16637
horse-galloping	meshes	12	25293
character 4 limp walking	MD5, meshes	45	1368

We will discuss each of these in more detail below.

Extraction of distinct characteristic motions

The first thing to verify with our motion learning scheme by using LLE is that different animation sequences containing different characteristic motions are extracted as distinct shapes of embeddings. As example, Figure (3.3) shows the embedding results on two motions, walking and running. As in Table (3.1), we used the running character 1, running character 2, walking character 1 and walking character 2, as the test data. As can be seen from the results, the embeddings for walking and running by the same person are distinctly different. On the other hand, the embedding results are very similar for two different persons with the same number of DOFs but with different number of frames (see Table (3.1)) in their respective sequence , when performing the same action individually.

Embedding extraction with a shuffled frame order

Next we designed a test to reflect the ability of our motion learning scheme with LLE to handle the data with shuffled frame orders. In Figure 3.4, the embedding result of the shuffled frame order has exactly the same shape and space coordinates as the original and correct frame order sequence. This property is of particular use

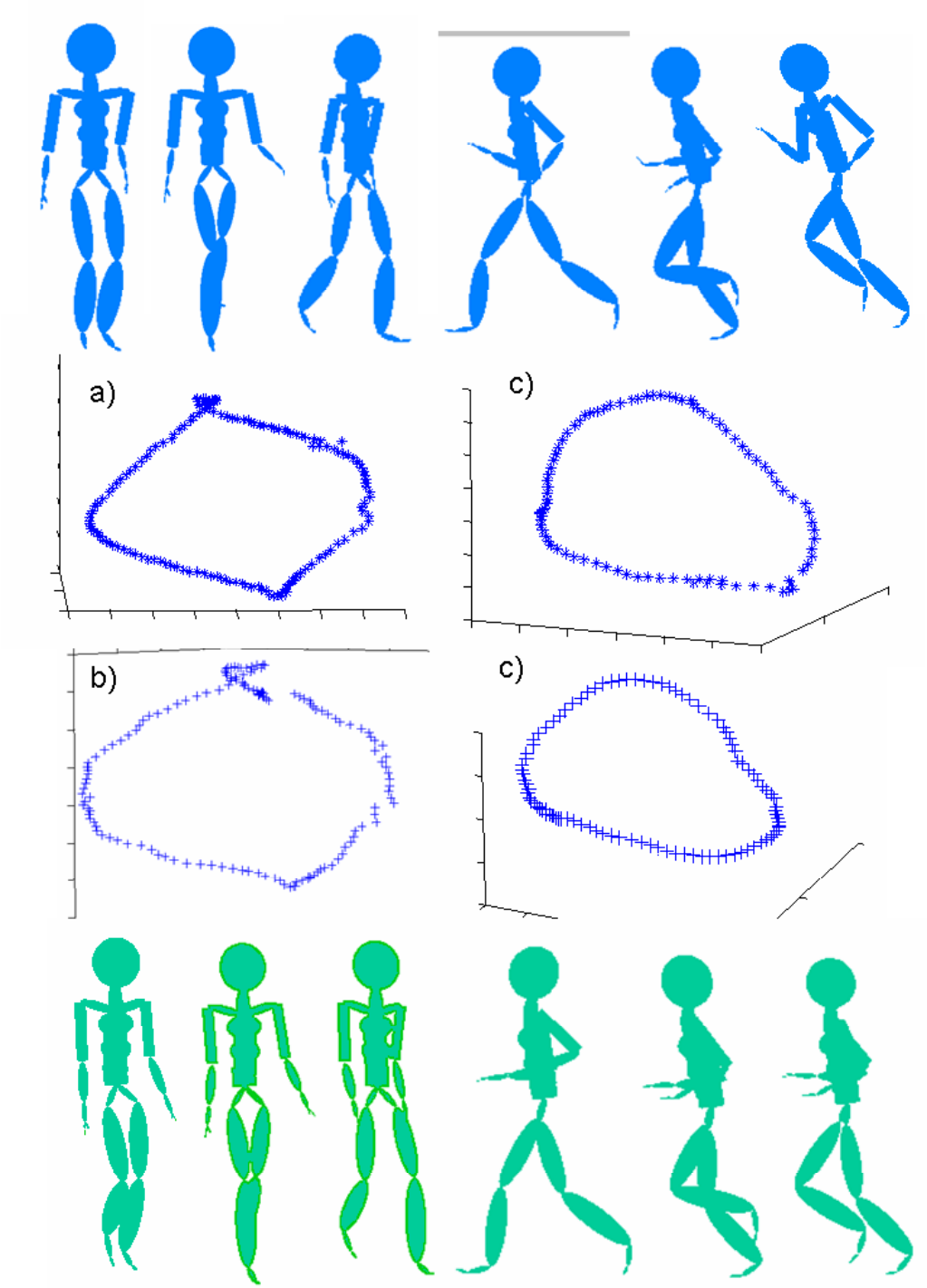


Figure 3.3: $3D$ LLE embeddings of two skeletons (character 1 shown at the top and character 2 shown at the bottom) performing walking and running actions. a) character 1 walking; b) character 2 walking; c) character 1 running; and d) character 2 running.

when dealing with animations transmitted over the Internet. Usually, animation sequences are very huge and it is very time consuming to transfer them over the Internet. But if we can cut the animation into smaller sequences of frames, then we can use a distributed method to transfer all the animation pieces in parallel, which saves transmission time. After transferring all the pieces, there remains the problem of all the pieces in the correct order. Due to the ability of LLE to treat each frame individually, and the order has no contribution to the embedding calculation, we can rebuild the correct frame order by comparison with the embedding.

Embedding extraction with a small number of frames

LLE embedding can also reflect the motion information even with some amount of incompleteness in the input set of frames for an animation sequence. In Figure 3.5, we compare embedding results of the original sequence with results obtained by reducing the input information, removing several frames from the original sequence. We can see that in the case where the number of frames has been reduced by one frame, the embedding has exactly the same shape as the full data embedding. Loss of 1 frame in 12 amounts to around 8% reduction in information. The shapes are very similar because the remaining 11 frames contribute enough to the characterizing motion information. We also tested the embedding calculation result after we truncated the sequence by two and three frames, i.e., up to 25% reduced information. The results show that if large amount of motion information is missing then it does affect the reliability of the embedding result.

Embedding extraction with multiple animation sequences

In the last test for our motion learning scheme with LLE, multiple animation sequences were put together for one embedding calculation. We put a walking motion,

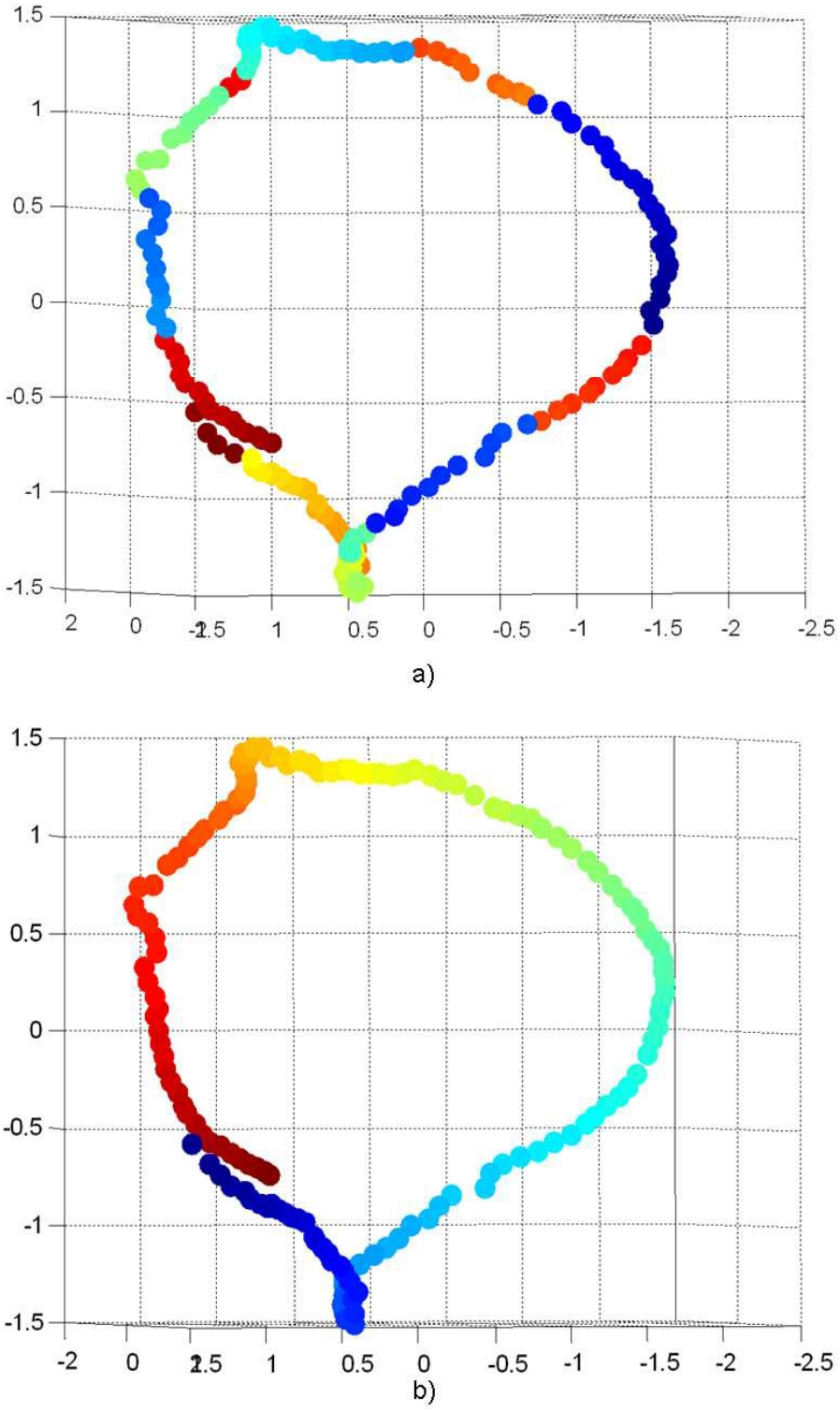


Figure 3.4: $3D$ embedding comparison between shuffled order and correct order for character 1 walking data. a) the embedding with shuffled frame order; b) the embedding with correct frame order.

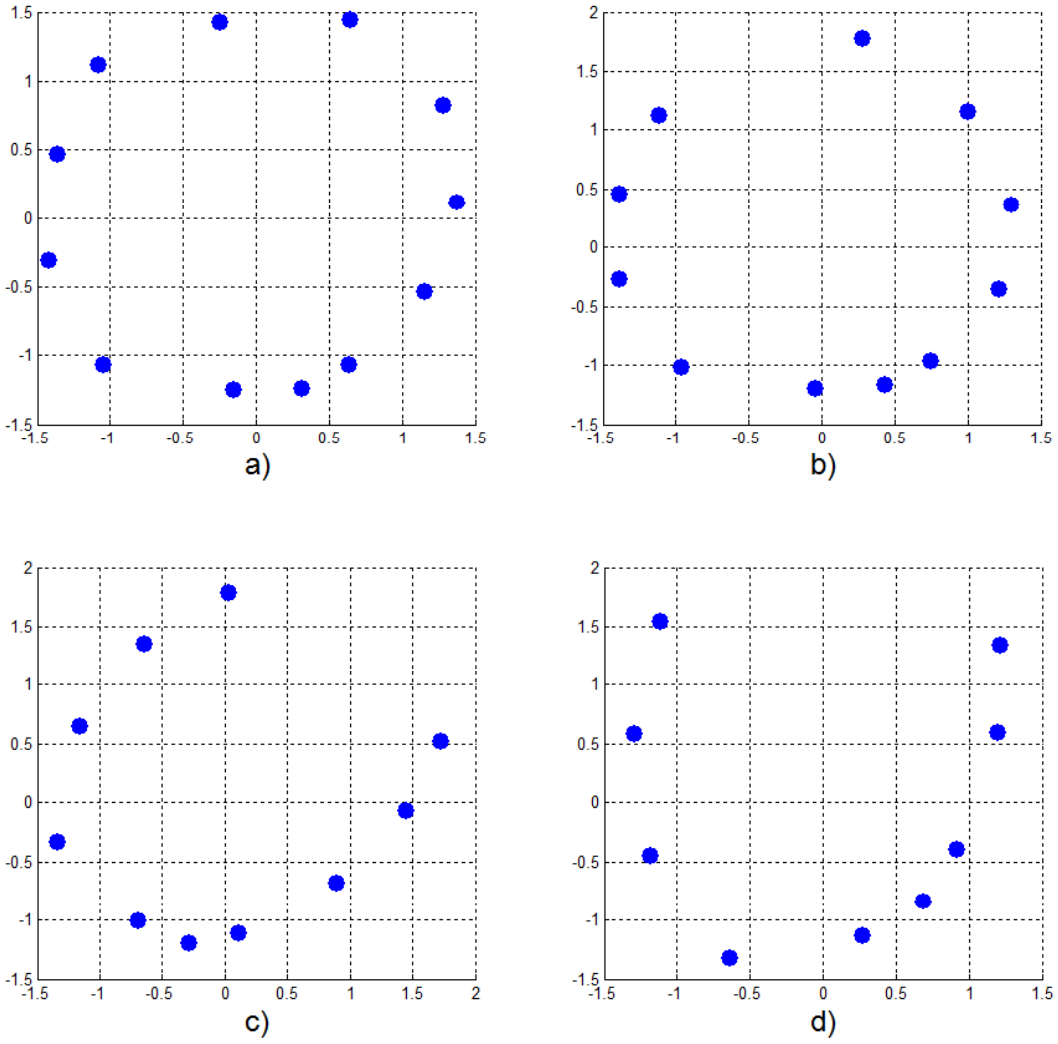


Figure 3.5: $2D$ LLE embedding comparison. a) with original 12 mesh frames from horse galloping; b) the embedding after one frame (frame number 12) removed; c) the embedding after two frames (frame numbers 11, 12) removed; d) the embedding after three frames (frame numbers 10, 11, 12) removed.

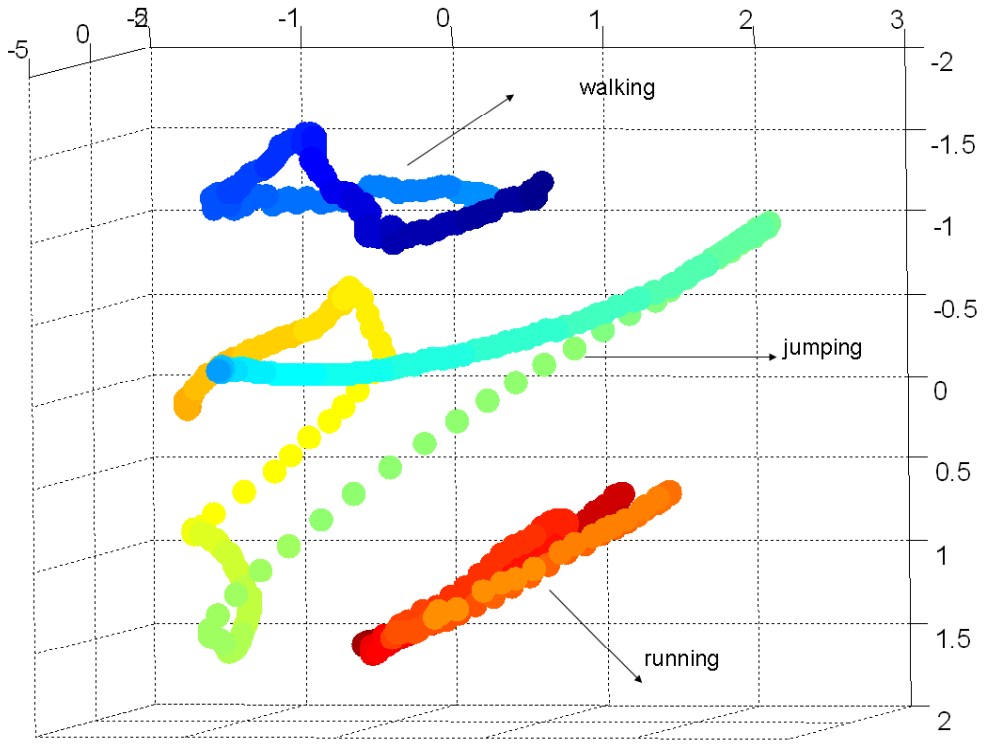


Figure 3.6: $3D$ LLE embedding comparison with multiple sequences for walking, jumping and running for the same character 1.

a running motion and a jumping motion of a skeleton model together to create a new large mixed data sequence. Then we calculated the embedding with the result shown in Figure 3.6. This figure shows that in the embedding space, all the motion curves are clearly separated from each other.

With these supporting experimental results, we concluded that our motion learning scheme with LLE is very useful for analyzing the motion information from character animations. It works equally well with mesh-based and skeleton-based animations. It also works with keyframe representations alone or with full animation sequences, like those obtained through mocap systems.

3.3 Reconstruction Matrix Formulation

Next we provide a reverse mapping method which lets us recover the animation frames back from the embedding space. In fields such as computer vision and pattern recognition, it usually suffices to do the analysis. But in computer animation, reconstruction and synthesis of animation sequences is the main goal. Hence such a reverse mapping method is essential.

We use Generalized Radial Basis Functions (GRBF) [139] to formulate the variable part of the motion. This method has been widely adopted by many researchers for height interpolation or for deformable models and could be executed in real time for the sizes of models used in our experiments [140, 141]. Of particular interest are functions of the form:

$$\mathbf{f} = p(e) + \sum_{j=1}^{N} \alpha_j \phi(|e - e_j|) \tag{3.4}$$

where e is any point in the embedding space ($e \in R^d$, d is the dimensionality of the embedding space); e_i are the reference points; α_i are the real coefficients; $p(e)$ is a linear polynomial function with real coefficients $\mathbf{c}$ of e; $|\cdot|$ is the Euclidean distance function and $\phi(\cdot)$ is a real-valued basis function. Typical choices of $\phi(\cdot)$ are biharmonic ($\phi(u) = u$), triharmonic ($\phi(u) = u^3$), thin-plate spline ($\phi(u) = u^2 \log(u)$), multiquadric ($\phi(u) = \sqrt{u^2 + a^2}$), Gaussian, *etc.* To ensure orthogonality and to keep the result well posed, we impose the following constraint:

$$\sum_{i=1}^{N} \alpha_i p(e_i) = 0, \tag{3.5}$$

For a given motion segment depicting a repetitive action, we aim to learn a decomposable generative model that explicitly consists of the following two factors:

Motion characteristic A representation of the intrinsic feature configuration, distinguishing characteristic in that motion. This factor is very similar in different

instances of that motion.

Variable part in the motion Parameters which can vary with each instance, but do not significantly affect perception of the characteristic in that motion.

The idea of decomposing motions has been explored by a number of other researchers. Most often, these decompositions are in the format of content + style [43,44,83,142] or in the format of signature + action [143]. In all the above cases, the aim is to learn similarity/difference in classes of motions. In our case decomposition is carried out for a given motion segment. We assume that frame $\mathbf{f}$ is a function of $\mathbf{b}$ (variable part of the motion), and the ψ (motion characteristic). The dimensionality of $\mathbf{b}$ and ψ are n and N, respectively. We learn a frame-based generative mapping model in the form:

$$\gamma : (\mathbf{b}, \psi) \rightarrow \mathbf{f} \tag{3.6}$$

We assume that $\gamma(\cdot)$ is a bilinear function given in its most general form by:

$$\mathbf{f} = \sum_{ij} \omega_{i,j} b_i \psi_j \tag{3.7}$$

where each ω_{ij} is a n-dimensional vector of parameters used to transform the motion characteristic component and variable component into skeletons. Following the development in [144], we combine the interaction terms ω_{ij} with $\mathbf{b}$, and get the Equation (3.7) into an asymmetric two factor model form:

$$\mathbf{f} = B \cdot \psi \tag{3.8}$$

where B denotes the matrix of the variable parts of the motion, and ψ is a vector of coefficients specific to the motion characteristic.

Applying LLE yields us the nonlinearly embedded representation of the motion manifold in a low dimensional Euclidean embedding space, R^d. Let us recall

that these embeddings describe the distinguishing information of the motion, such as walking, running and dancing. Then with generalized radial basis functions (GRBF), we map from embedding space to original space: $R^d \rightarrow R^n$ using a nonlinear mapping function: $R^d \rightarrow R^N$ and a linear mapping: $R^N \rightarrow R^n$. By setting the non-linear mapping as our ψ, and linear mapping as our B, we obtain our bilinear model.

Using the asymmetric bilinear model form of Equation (3.8), with extended dimensionality so that the frames are mapped from embedding points, the whole reverse mapping can be written in a matrix form as:

$$\mathbf{f} = B' \cdot \psi'(e) \tag{3.9}$$

where B' is an extended $n \times (N + d + 1)$ matrix composed of two submatrices: an $n \times N$ matrix B with jth row $[\alpha_1, ..., \alpha_N]$ and an $n \times (d+1)$ matrix C with jth row $[c_j^T]$, where $\psi'(e)$ is an extended matrix composed of $\psi(e) = [\phi(|e-e_1|), ..., \phi(|e-e_N|)]$ and the vector $[1, e^T]^T$.

For the N frames case, the B and C can be calculated directly by solving the linear system:

$$\begin{pmatrix} \mathbf{f} \\ \\ 0_{d+1} \end{pmatrix} = \begin{pmatrix} A & P \\ \\ P^T & 0_{(d+1)\times(d+1)} \end{pmatrix} \times \begin{pmatrix} B^T \\ \\ C^T \end{pmatrix} \tag{3.10}$$

where $A = \phi(|e - e_i|)$ is the kernel matrix. If we use the polynomial part of the GRBF in Equation (3.9), which has the form $p(e) = [1, e^T] \cdot \mathbf{c}$, then P is the matrix with the ith row $(1, e_i^T)$.

To verify the reconstruction result, we calculated the embedding of reconstructed character animation, and compared the embedding result with the one from the original character animation. We compared the embedding results between original character 1 working skeleton animations with 160 frames and 62 DOFs and the

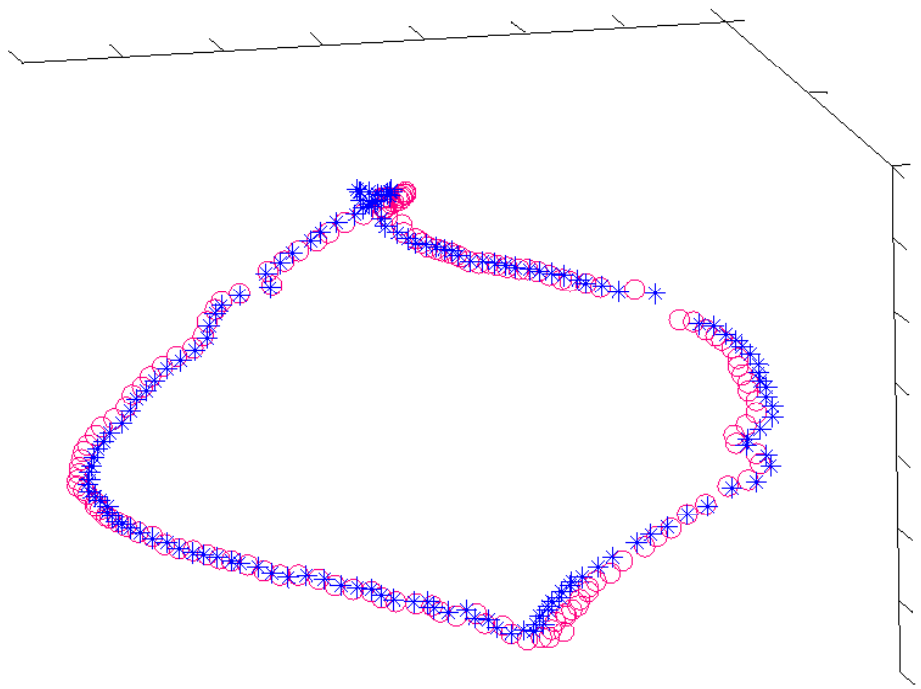


Figure 3.7: Comparison between an input animation and a reconstruction of the animation from the embedding space. '$*$' is the original one and '$\circ$' is the reconstructed one.

GRBF reconstructed animations. Figure (3.7) shows the comparison result and it clearly shows that the two embeddings match each other nearly perfectly.

In this chapter, we have provided a detailed discussion about our motion learning scheme with LLE. It projects the high dimensional motion data into a lower dimensional space. We describe the results of many experiments to show that our motion learning scheme with LLE is a very powerful tool to analyze character animations. This scheme will allow us to extract the characterizing motion information for many other uses. From next chapter onwards, we will provide three new applications in enhancing keyframe character animation using the results of our motion

learning scheme.

Chapter 4

Generation of In-betweens Using Characteristic Motion Representation

In this chapter, we introduce our first enhancement of the keyframe character animation technique. We use the LLE embedding in low dimensional space computed for a given keyframe representation of a character animation to generate in-betweens in a way to make it easier to meet animator requirements.

This chapter is organized as follows:

1) In Section (4.1), we discuss in-between generation for character animation with preservation of physical properties for mesh-based animation and present our framework.

2) We explain how we apply this framework in Subsection (4.2). We first provide our definition of the feature space in Subsection 4.2.1. Then we introduce the property maps and its usage in Subsection (4.2.2). In Subsection (4.2.3), we discuss the algorithm for in-between mesh reconstruction.

3) In Section (4.3), we provide experimental results for mesh-based character animation, showing the effectiveness of our approach.

4) We introduce details about how we apply our framework to character animation in skeleton form with experimental results in Section (4.4), demonstrating that our methods work equally well for mesh and skeleton animations.

4.1 Introduction to the Motion Learning-based Framework

Skeleton-based character animation data can be obtained by using sensing hardware equipment such as a mocap system. Even though the technology of surface sensing/scanning has made tremendous advances, mesh-based character animation data mainly depends on the animator's "hand-made" works. As we described in Section (2.1), specifying the key poses of the desired motion is the most popular technique in practice today for animators to generate character animation sequences. Given the key poses in the form of unarticulated 3D meshes, the dominant method of producing the deformation for the in-between meshes is to use a suitable interpolant, such as a linear or a spline interpolation. Given the high sensitivity of these interpolants to the supplied sample data, this imposes a significant burden on the animator to ensure that her/his requirements are satisfied by the automatically generated in-betweens. In addition to the creative aspects of defining the animation, the animator must carefully provide the number and spacing of the key poses to suit the interpolant so that the generated in-between meshes satisfy the required physical properties of the desired motion, such as mesh surface area or volume, or any other such computable mesh metric.

The interpolants in use today are generally driven by issues such as computational efficiency and geometric continuity of individual vertices of the mesh. They

do not take into consideration the high correlation amongst mesh vertices or the desirable constraints on physical properties of the meshes. Related to such constraints, in [47,49] the conventional inverse kinematics method for limbed structures has been extended to meshes. Similarly, physics-based methods such as forward and inverse dynamics methods have been proposed, which are also used in bio-mechanics and rely on physical properties like joint torques for limbed structures [106].

The goal of our proposed framework is to let computers generate in-between frames while preserving physical properties in the generated poses. Moreover, our framework aims to release animators from the heavy burden of having to create the best set of key poses. We provide a framework, which has the ability to collect information from the entire motion, and to synthesize the in-between frames with desirable physical properties. In order to provide the right perspective for our framework, let us look at the following example.

Consider the goal of producing an animation sequence that shows a galloping horse represented with meshes. Let us assume that the horse is defined by using a mesh with n vertices. Suppose we are given the positions of the vertices of the mesh at certain discrete key poses, denoted by $\mathbf{f}_i$. Since the galloping motion is defined as a continuous deformation of the mesh, $\mathbf{f}_i$ can be considered as a function of parameter t with the animation sequence defined between the start of the gallop t_0 to the end of the motion t_1. Thus, at any intermediate pose at sequence position i corresponding to time $t_i \in [t_0, t_1]$, we can view the corresponding deformed mesh $\mathbf{f}_i = f(t_i)$ as a point in the high dimensional space $R^D = R^{n\times 3}$. In the same sense, the entire galloping motion can be treated as a motion curve in the R^D space, with t as the curve parameter. Methods for automatically deriving this motion curve are extremely difficult because of the fact that the dimensionality of space R^D is very

high; usually, the character meshes can have tens of thousands of vertices. And compared to this high dimensionality, we can only expect to have data for a very limited number of key poses for the animation.

Due to the difficulty involved in constraining the motion to yield a desirable curve, most traditional keyframe-based techniques adopt the simple method of interpolation among local neighbors. While this considerably simplifies the problem of defining a curve path in the high dimensional space of meshes, it means that an automatically generated in-between pose is only related to a small subset of input key-poses; in other words, the influence is only from a highly restricted part of the whole motion. Hence, the resulting in-betweens may not be ones with a high probability of being on the curve that defines the desired motion. For example, when we do simple linear interpolation among two key poses of the horse, there is no guarantee that we will obtain a valid mesh for the new in-between pose. Ideally, we would like to obtain in-between meshes that are not only valid, but also satisfy the desired physical properties that are consistent with the motion.

In the rest of this chapter, we present our motion learning-based framework to produce more natural animation sequences from keyframes. This framework consists of the following three processes:

1) **Motion structure learning:** We use motion learning scheme with LLE , introduced in Chapter 2, to cast the motion, specified by key poses in a high dimensional space, into a lower dimensional space, such as two or three dimensions. Every point in the embedding space corresponds to a mesh configuration, which is reconstructed using Generalized Radial Basis Functions.

2)**Property maps initialization:** We compute maps of property values in the LLE space. For illustration, we choose the mesh volume and surface area as

the fundamental two properties to build the property maps. These two are chosen because they are stable during the motion performed in the real world. Some other properties could also be used to build the property maps, such as the vertex normal.

3)**Frame generation:** We derive a probability distribution for the desired property in the neighborhood of each pose, and choose points in the embedding space with the highest probability to synthesize the in-between meshes for the desired motion.

4.2 Motion Learning-based Framework Components

In this section, we will give a more detailed description of the components in our motion learning-based framework. We will discuss it first for mesh-based animations and then apply the same to skeleton-based animations.

4.2.1 Feature Space

Clearly, the motion learning result found in Section (3.2) is highly dependent on the difference measurement, which is the $\Delta(\cdot)$ in Equation (3.1). We must provide meaningful features for our framework, which can then be used to define the $\Delta(\cdot)$. Therefore each input pose $\mathbf{f}_i$ has a corresponding feature vector $\mathbf{s}_i$, where i is an index over the input poses. A fundamental feature, which can be used to represent the difference between two mesh configurations, is the difference in data values. That means we can view each pose as a $3n$-tuple in the space $R^S = R^{3\times n}$, where S is the dimension of the feature space, and n is the number of vertices in the character mesh. Sumner *et al.* found deformation gradients to be another very important

feature to capture the differences between poses [45, 47, 49]. Although only mesh vertices positions are used as features in our experiments, our formulation can easily accommodate deformation gradients and other such features.

Let the distance between two frames $\Delta(\cdot)$ be the following:

$$\Delta(\mathbf{f}_i, \tilde{\mathbf{f}}_j) = |\mathbf{s}_i - \tilde{\mathbf{s}}_j|^2 \tag{4.1}$$

Combining this definition with that in Equation (3.1), we obtain:

$$\begin{aligned} \epsilon_i &= |\mathbf{s}_i - \sum_j \omega_{ij}\mathbf{s}_j|^2 \\ &= |\sum_j \omega_{ij}(\mathbf{s}_i - \mathbf{s}_j)|^2 \\ &= \sum_{jk} \omega_{ij}\omega_{ik}G_{jk} \end{aligned} \tag{4.2}$$

where $G_{jk} = (\mathbf{s}_i - \mathbf{s}_j) \cdot (\mathbf{s}_i - \mathbf{s}_k)$, and $\sum_j \omega_{ij} = 1$.

The reconstruction error can be minimized analytically using the Lagrange multiplier. In the R^3 space, a vertex position is a 1×3 vector. However, the above method separates each primitive into 3 individual components, which cannot represent the vertex-based information of the 3D mesh properly. To overcome this issue, we group the feature values in $\mathbf{s}_i$ into their corresponding vertex representations to create a $3 \times n$ matrix $\mathbf{S}_i$ of feature vectors. To be able to compute the corresponding dot product in Equation (4.2) with our new formulation, we first compute the L_1 norm magnitude of the feature vectors by creating a $1 \times n$ matrix of scalar values, which is denoted as $|\mathbf{S}_i|_1$. Therefore, modeling the corresponding ϵ_i in the same way as in Equation (4.2), we approximate the matrix G_{jk} as:

$$G_{jk} = |\mathbf{S}_i - \mathbf{S}_j|_1 \cdot |\mathbf{S}_i - \mathbf{S}_k|_1 \tag{4.3}$$

The results that were shown in Figure (3.1) use this definition to perform the LLE.

4.2.2 Property Maps

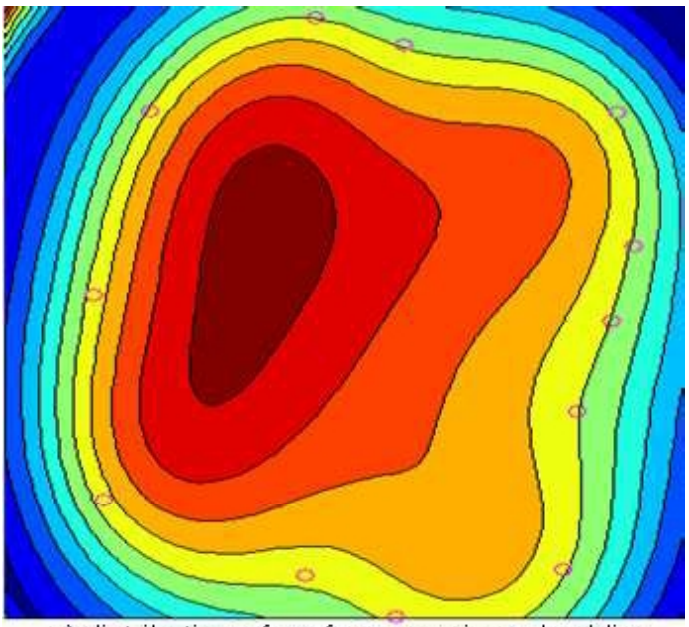
The ability to preserve the physical properties during the in-betweens generation is a big advantage of our framework. This is achieved by using property maps. In this section, we present the creation and usage of property maps. Every point in the embedding space corresponds to one pose in the original high dimensional space. However, only a few of points in the embedding space satisfy the constraints that:

1) The points in the embedding space are part of the desired motion sequence;

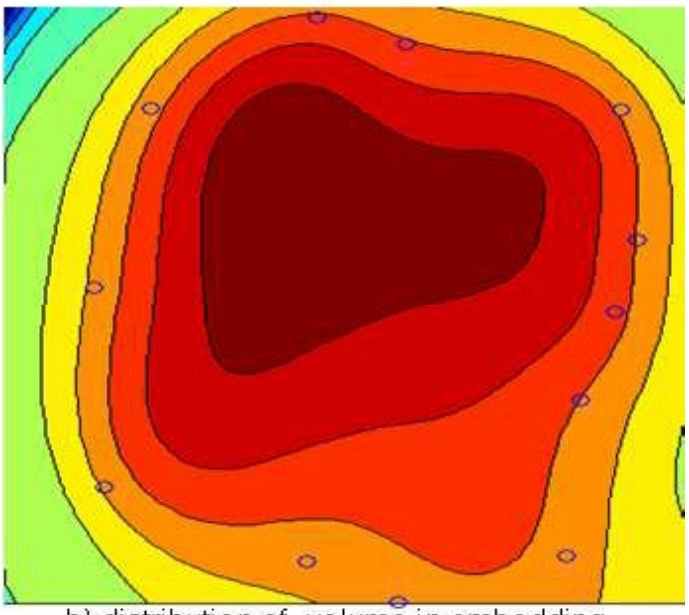
2) They maintain desired physical property constraints, for example, a stable surface area.

To be able to identify such desirable intermediate poses, in this step, we determine a path for this motion in the low dimensional space. Assume that the motion sequence is parameterized from parametric values u_0 to u_1. Then, by using the embedding obtained from LLE, our goal is to determine the points on the motion curve $f_d(u) : u \rightarrow R^d$ where $u \in [u_0, u_1]$ and R^d is the embedding space.

To build up the property maps in the embedding space, we create a regular grid in the embedding space with step size δ in each dimensional direction of the embedding space. At every vertex on the grid, we generate the corresponding pose with the inverse mapping of $M_D : R^d \rightarrow R^D$. The inverse mapping is calculated by using GRBF (Equation 3.9). More details are given in Section (4.2.3) that gives a mesh from which the property value for the grid point is calculated. We can also calculate the required physical properties for each mesh configuration. In this way, we are able to bind embedding space points with physical properties. The physical



a) distribution of surface area in embedding space.



b) distribution of volume in embedding space

Figure 4.1: Distribution of physical properties of shapes in embedding space.

properties that we have experimented with, like surface area, body volume, etc., seem to have an orderly distribution in the embedding space, as shown in Figure (4.1). In the figure, the calculation of physical properties, by using interpolation between grid points, over the entire domain in the LLE space is for demonstration purposes only. In actual practice, we need to compute the physical properties only in the vicinity of the mapped motion curve.

Observing that the physical properties are also functions of the parameter u during the motion, we build a probability distribution function P, which indicates the possibility that the correlated point is located on the motion curve in the embedding space. Without loss of generality, we assume P is a normal distribution, $P \sim N(\mu, \Sigma)$, with mean vector μ and covariance matrix Σ. With the maximum likelihood parameter estimation (ML) [145], we have:

$$\hat{\mu} = \frac{1}{N}\sum_{k=1}^{N}\mathbf{p}_k \tag{4.4}$$

$$\hat{\Sigma} = \frac{1}{N}\sum_{k=1}^{N}(\mathbf{p}_k - \hat{\mu})(\mathbf{p}_k - \hat{\mu})^T \tag{4.5}$$

where N is the given number of key poses. With the estimated probability distribution function, we calculate the probability of grid vertices in the embedding space, as shown in Figure (4.2). By adding the most probable mesh configurations, we are able to rapidly determine points on the desired motion curve in the embedding space. The motion curve generation result is shown in Figure (4.3).

4.2.3 In-between Mesh Reconstruction

After we have the motion curve in the embedding space, we will transform the vertices on the motion curve in the embedding space into an animation frame represented by meshes. Given any value of parameter $u_i \in [u_0, u_1]$, an embedding point

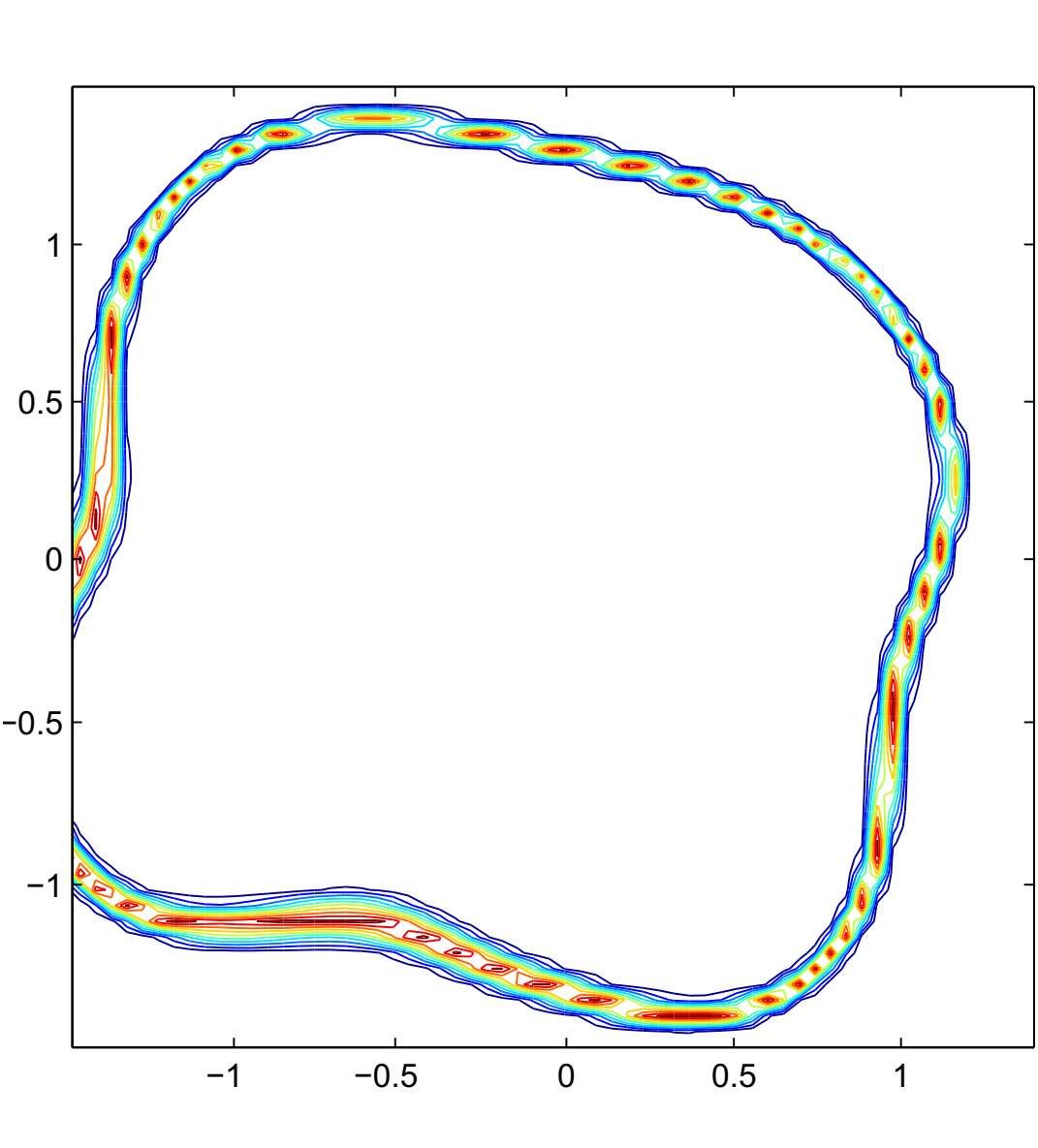


Figure 4.2: Probability Contour.

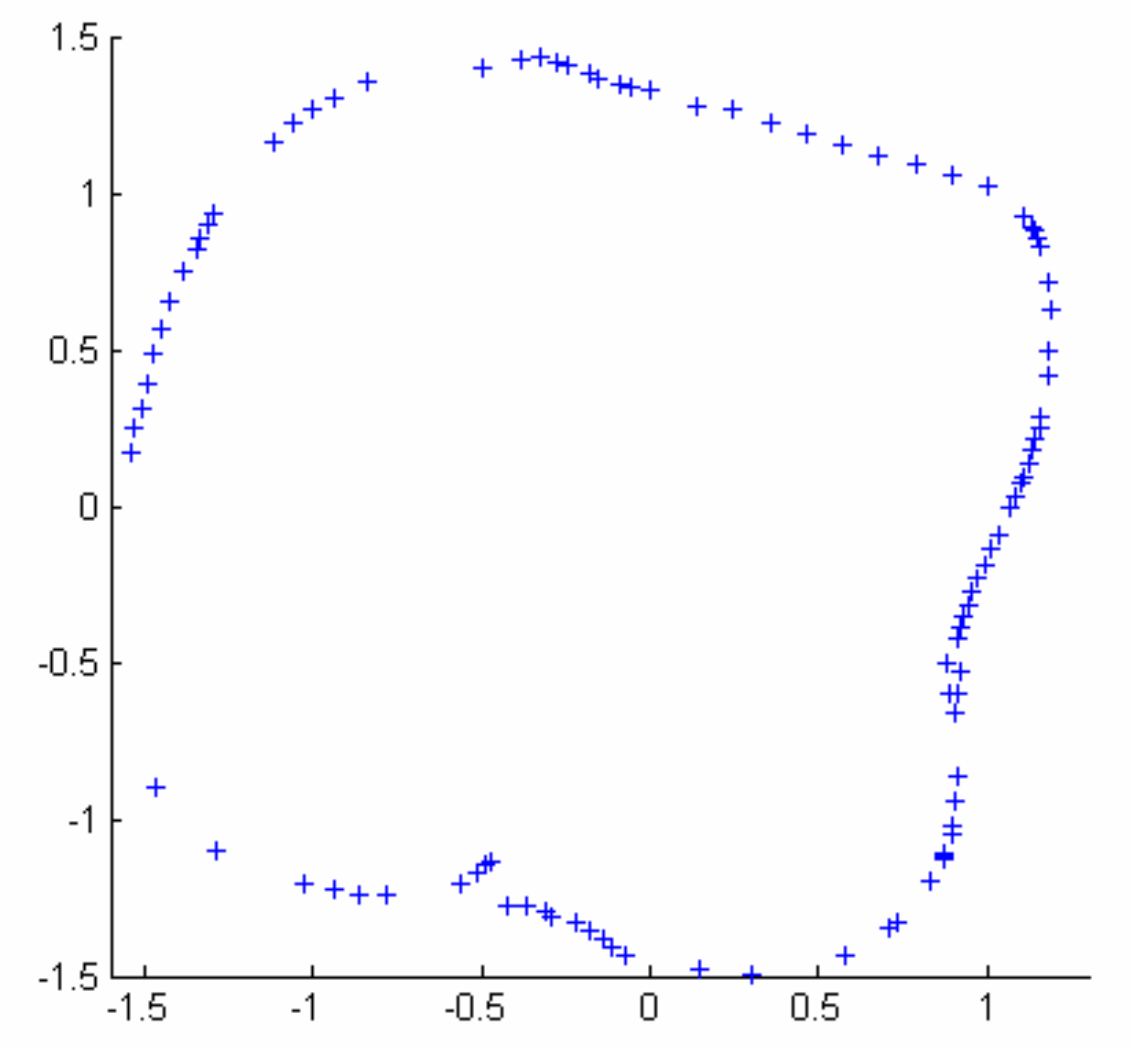


Figure 4.3: Motion curve (the physical property used is the body volume). The points have high probability to lie on the motion curve based the body volume property.

can be obtained from the motion curve function. Then, we must reconstruct the corresponding mesh configuration $\mathbf{f}_i$ with the nonlinear mapping from the embedding space to the mesh space, as $M_D : R^d \rightarrow R^D$. As already discussed in Chapter 3, we use the Generalized Radial Basis Function (GRBF) interpolation framework represented by Equation (3.9) for this purpose [139]. Unlike the reconstruction method used by traditional interpolation methods, we use the entire set of keyframes and not just the nearby keyframes to contribute to in-between frame generation.

4.3 Experimental Results

To demonstrate the capabilities of our proposed framework, we carried out a number of experiments including the motions of a galloping horse and a collapsing camel, shown below.

1. The all-but-one experiment: In the first experiment, we randomly removed one of the given 12 key poses, denoted as target pose ($\mathbf{f}_{tar}$), and then used the data in the remaining 11 poses to learn the motion and synthesize the target pose. By visually comparing the synthesized result with the target mesh, as well as by comparing geometric differences, we found that the newly synthesized mesh was very much like the target mesh. Figure (4.4) shows the synthesized results for poses 2, 7, and 12. The physical properties that we maintain are both body volume and surface area, with equal weights. This clearly demonstrates that our method can produce a desired in-between pose by considering that the entire set of key poses provide reconstruction information in a global fashion. And the desired in-between pose is much less critically dependant on specific neighbouring poses.

2. Robustness experiment: We also did an experiment to demonstrate that our framework is not very highly sensitive to key poses. Our framework can

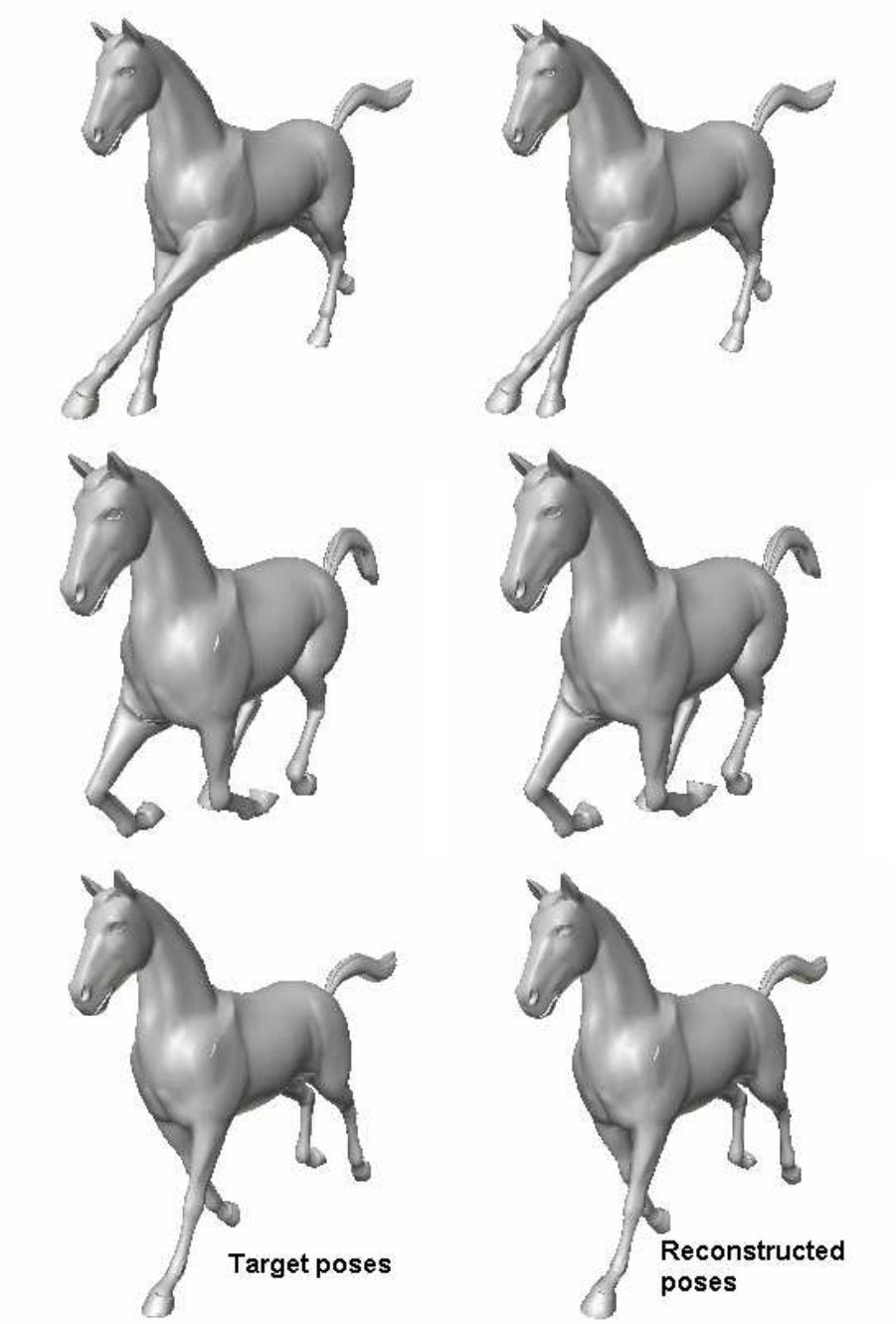


Figure 4.4: In-between pose generation. Out of 12 key poses, we randomly remove one (top: 2, middle: 7, bottom: 12), learn the motion from the remaining 11, and reconstruct the removed one. The removed pose is shown in the left column, and the reconstructed one is shown in the right column. The likeness between the two can be easily seen.

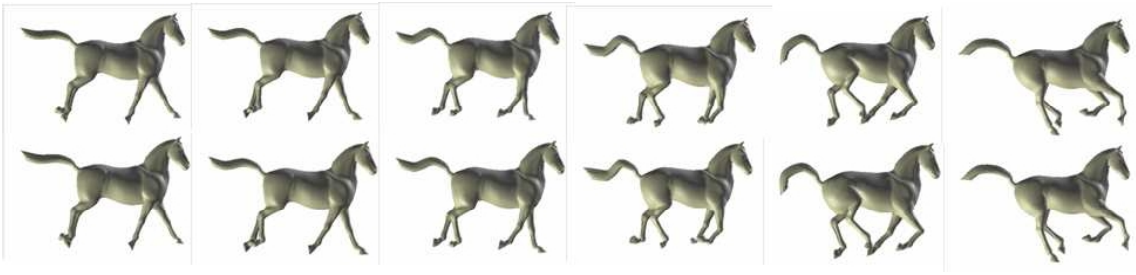


Figure 4.5: Visual comparison of poses in original animation (top row) and in motion reconstructed from small set of key poses selected by stratified sampling (bottom row).

handle motion well, as long as the specified key poses adequately incorporate the inherently desired motion behavior. We have 56 poses that make up the galloping motion sequence. We divided these 56 poses into 12 subsequences, and used the stratified sampling method to randomly pick one pose from each subsequence, which is a stratum in our experiment. Stratified sampling is used to test whether the significant characteristics of the motion are retained in our framework. We use these 12 poses from the 12 subsequences as our input to reconstruct the desired motion. We compare the reconstructed full sequence of 56 frames, the horse-galloping animation with the original sequence. The result is shown in Figure (4.5).

For comparison purposes, we focus on the motion of a characteristic part(s) of the horse, such as the motion of the horse tail, head, front legs, etc. We follow the motion of a randomly picked vertex from the part under focus, and record its movement for both the original and the reconstructed sequence. Figure (4.6) shows this for a vertex on the horse's tail; from Figure (4.6c), one can see that our reconstructed sequence is much better than the linear reconstruction. Since the two sequences are a collection of discrete samples from a continuous motion, we do not restrict the reconstructed in-between to be occurring at exactly the same time as in the original sequence. The two records of movement are not perfectly identical, but

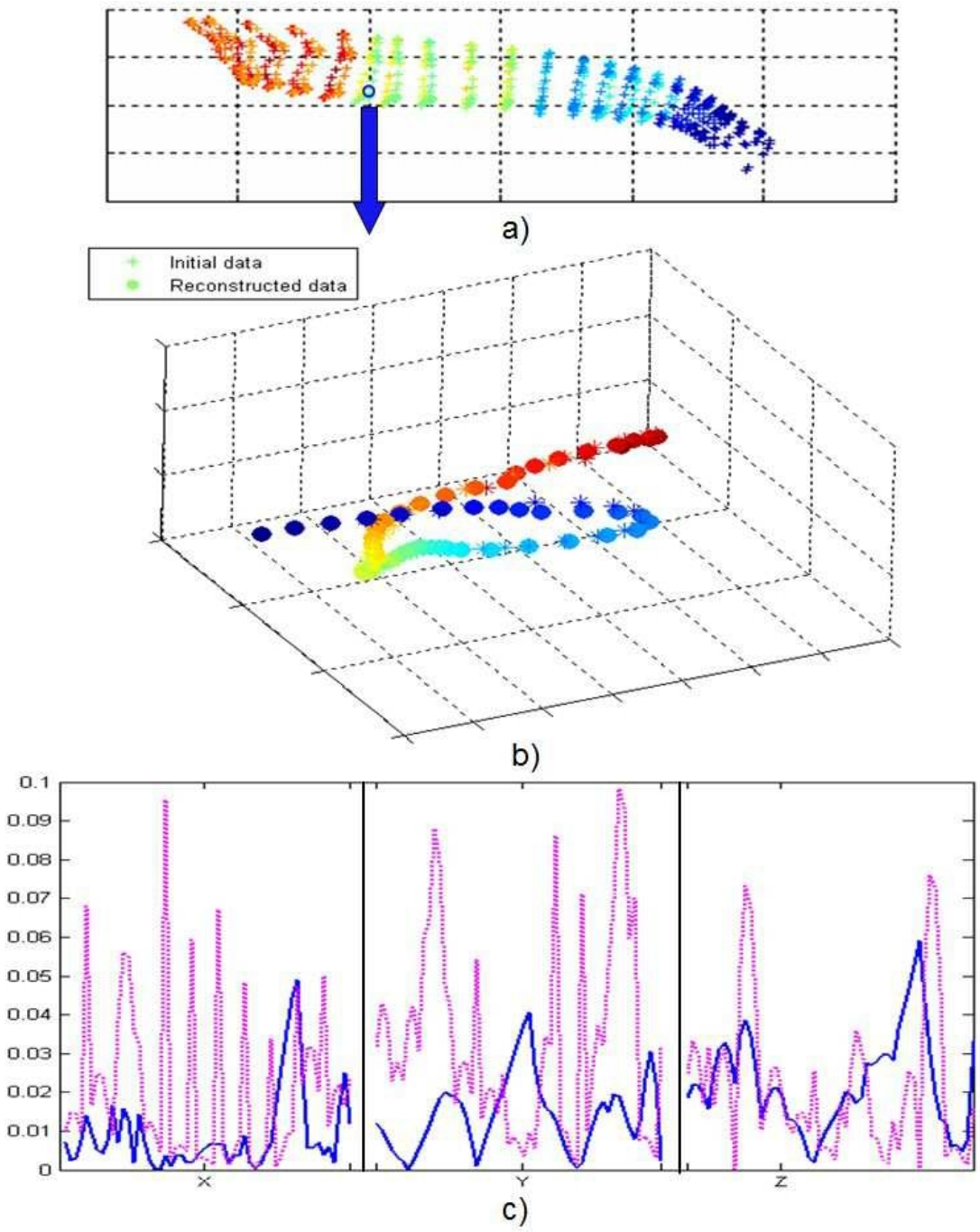


Figure 4.6: a) Comparing motion paths of select parts. We select one vertex (marked by the circle) from the tail part, and record its movement during the entire motion. b) We illustrate the selected tail vertex movement for the original 56 poses and for the synthesized sequence. c) Plot chart showing percentage difference, between the original and reconstructed sequence of the vertex location in X, Y, and Z direction individually. The dashed line shows the percentage difference of linear interpolation results; and the solid line shows our method results.

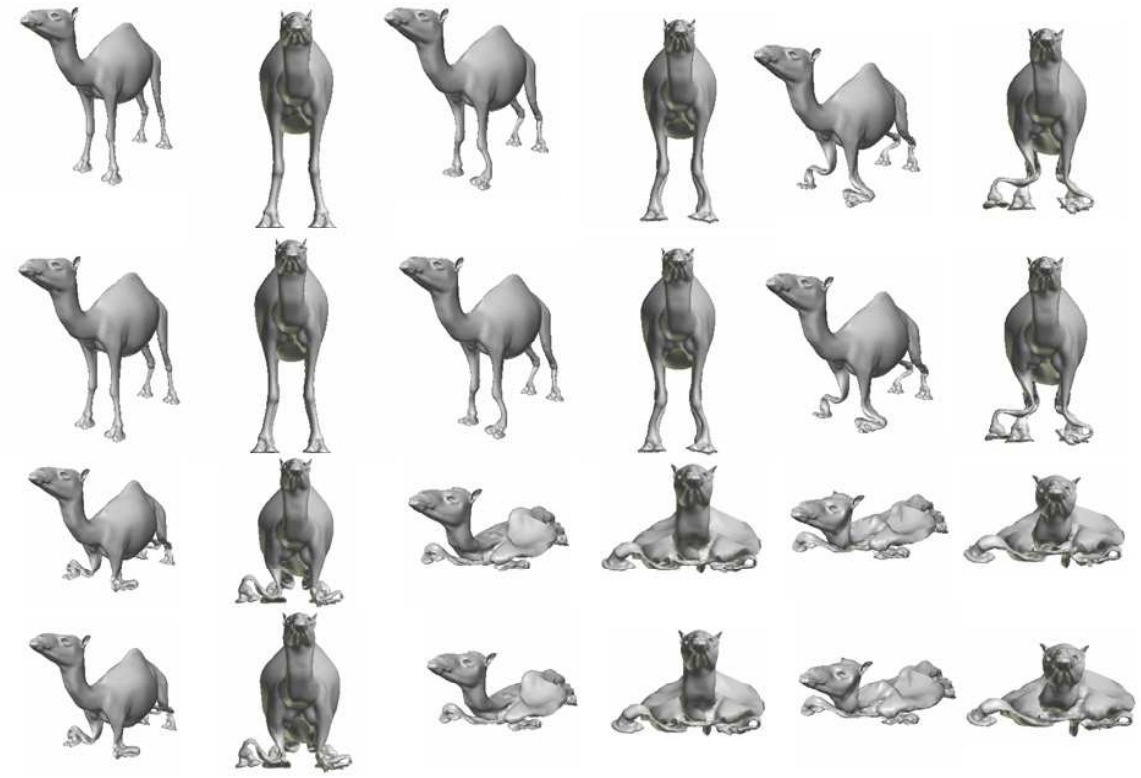


Figure 4.7: The first and third rows are meshes from original animation sequence and the second and fourth rows are reconstructed meshes using just 10 key poses as input.

are very close to each other, as can be expected for similar motions.

Figure (4.7) shows another similar experiment, with a different desirable physical property. The motion is that of a collapsing camel, given as a sequence with 50 poses. Each pose is a mesh with 21877 vertices. Fixing the start and end poses, we choose 8 internal poses by stratified sampling. Then, by using these 10 poses (start and end poses plus eight chosen poses) as the input, the motion is reconstructed as 50 frames with the desirable physical property, defined as a stable area and a monotonically decreasing volume.

4.4 Applying the Framework on Skeleton data

Compared to a mesh, the skeleton is simpler/smaller representation of character for keyframe animation purposes. Skeleton models use a hierarchical set of interconnected bones instead of geometric skin surface to represent the character. The character motions are recorded by changing the joint angles. So far, we have discussed the use of our framework for mesh-based character animations. However, our framework can also be applied to skeleton data with a proper definition of the property maps. In this section, we provide the performance of our framework on skeleton data. Unlike mesh data which uses position information as the DOFs, for a character, skeleton data has a fixed body model with a set of joint angles, which we can use as the DOFs. Matching with change in the type of DOFs, we define a new feature space for skeleton data.

Here, we demonstrate some experimental results. We have a walking motion with 160 frames. From the 160 frames, we select 10 frames as our keyframes (see Figure (4.8)). Without doing any time consuming selection operation for the keyframes, we simply selected the keyframes through stratified sampling because our framework has the ability to match the requirements of the desired properties with a global reconstruction approach.

For the skeleton character animations, balance check is a very important step. For example, in walking motion, at least one of the feet of the skeleton has to be in contact with the ground at any time. We can use this one to define our property maps. In Figure (4.9), we show the physical property function used to construct this property map. We chose the foot pattern of the walking motion, which was simulated conveniently from a walking video.

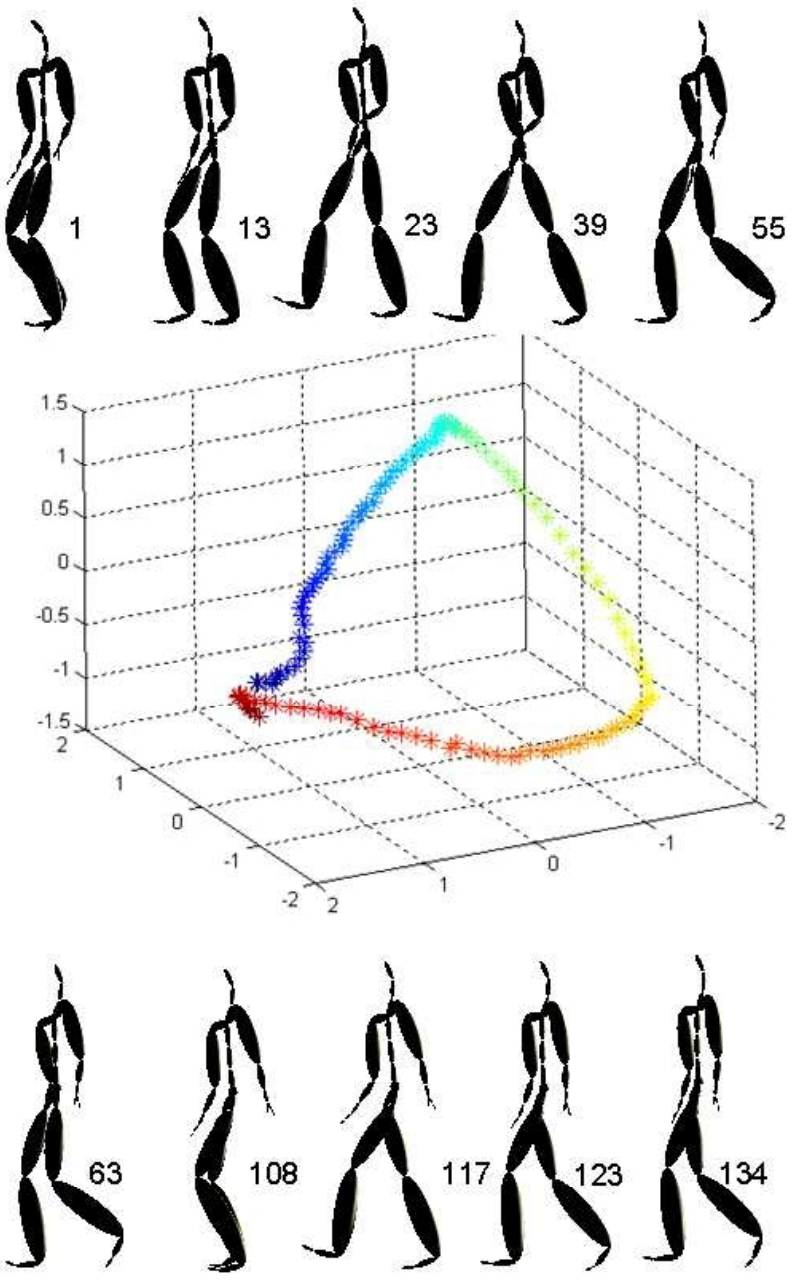


Figure 4.8: We show the embedding space for the original 160 walking frames, and the selected keyframes we used to synthesis the reconstruction motion.

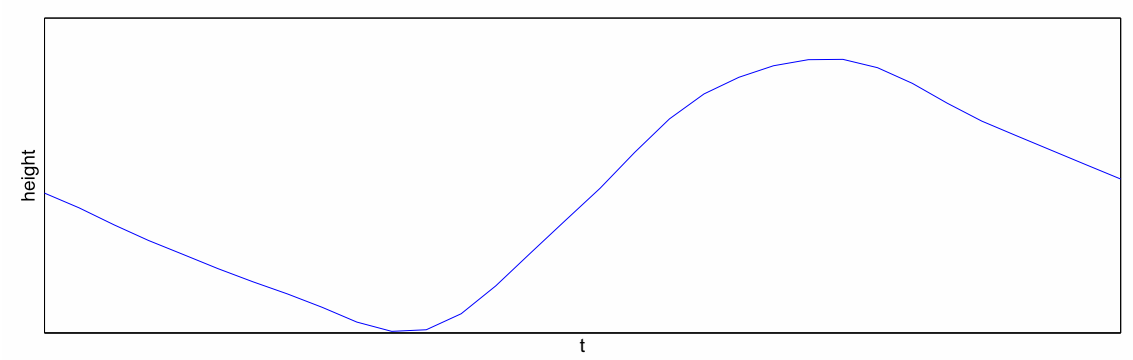


Figure 4.9: The two feet show a simple sine-like curve in the walking motion. We obtained the feet placement curve from a video clip of a walking motion.

4.5 Conclusions

We have introduced a framework for shape animation, which works equally well for mesh-based and skeleton-based keyframe character animation. Firstly, for any coordinated motion of a character, the framework uses a small set of key poses to learn the motion in a global fashion. This motion is learned as a path in a low dimensional space, using the nonlinear technique of locally linear embedding. Second, the framework incorporates a new idea of physical property maps of the deformable shape and enables the synthesis of the complete animation sequence based on desirable physical properties. We are not aware of any other earlier work that uses the LLE framework for motion learning and also uses property maps in low dimensional space in this manner. Such a framework will considerably ease the task of animators as they can now work with a small set of key poses and specify the synthesized motion in the form of desirable physical properties of the deformable shape during the course of the motion. With the help of a number of experiments, we have demonstrated that this framework works with a small set of sample key poses, produces in-betweens with the desirable properties, and is also not overly sensitive to the number and spacing of key poses. In-between reconstruction is

linear in computational complexity. However the property map computations take longer and the complexity depends on the grid size chosen for its representation. This can be done as a preprocessing step, for real-time animation.

Chapter 5

Generation of Variations in Repetitive Motion by Using Bilinear Factorization

In the last chapter, the successful creation of in-betweens confirms our idea that the characterizing motion content embedded inside the character animation sequences can be extracted by LLE and then used to guide the generation of new frames. This is an important enhancement to the keyframe animation technique, as it requires the animator to provide fewer keyframes and specify physical properties to be satisfied by the in-betweens.

In this chapter, we describe the second enhancement to keyframe animation for the generation of variations in repetitive motion by using bilinear factorization based on the LLE embedding of the characterizing motion in low dimensional space. The organization of this chapter is as follows:

1) In Section (5.1), we explain why creating variations in repetitive motion is very important for realistic motion generations and provide an overview of our approach.

2) In Section (5.2) we first provide a detailed review of motion variation methods

and then present our proposed method.

3) In Section (5.3) we describe results of experiments from an implementation of our method.

4) In Section (5.4), we conclude with a brief analysis of our method and its potential for further extension.

5.1 Introduction to Motion Variation

Two instances of the same action in different shots or scenes performed by the same actor will not be exactly identical. In spite of this, most 3D games and other applications with animated 3D characters rely on building up and using a repertoire of basic motions that are called upon repeatedly and executed identically each time. Often only short basic motions exist. This is because creating animation sequences manually is a difficult and time consuming process. Realistic motion behavior would require that repeated actions carried out by these animated digital characters incorporate variations that make different instances of the performed action appear slightly different. However, it is certainly non-trivial to be able to introduce such variations in a given motion while ensuring that the principal characteristics of the given motion are not altered.

As discussed earlier, the three primary approaches to creating basic motions are keyframe animation, physically based animation and motion capture driven animation [146]. Incorporating variations through keyframe animation would require the specification of how the keyframes change for different instances of the motion. It is well known that the specification of keyframes for a single motion is itself a highly skilled and labor intensive task. Specifying variations in motion curves of keyframe parameters in a controlled manner such that the principal characteristics

of the motion are unaltered is tremendously difficult. The physically based methods provide much higher level control to the animator by requiring specification of the physical parameters of the character/environment and constraints, and then solving for the entire motion as a constrained optimization problem. As well as being far too computationally intensive and frequently unstable, these methods are not very intuitive to be used interactively by the animator. Animators, in general, are not comfortable specifying values for mass, force, friction, etc. It is also unclear, short of simple trial and error, as to what guidance can be provided to the animator to manage changes in the physically based parameters for incorporating variations. The third approach, motion capture, depends entirely on data captured via a live performer from one or more instances of that action. Given that considerable effort is involved in getting usable animation data for one instance of the motion, it is again clearly far too much effort to directly capture, store and retrieve different motion variations, in addition to the number of captured variations being bound to be limited.

A simple motion variation method is to introduce suitable noise in the motion. However, during the process, physical properties are very hard to maintain especially for mesh based models. Another approach is to string and blend together motion fragments derived either from a longer motion sequence or searched from a database of pre-created motion fragments. This method will also meet problems for the mesh models. In most cases, the assumption is that reuse of motion fragments will retain the principal characteristic in the motion. A more detailed review of motion variation methods was given in the Section (2.2.2).

The method presented in this chapter is based on the well-accepted hypothesis in human perception that any repeated motion can be assumed to contain an

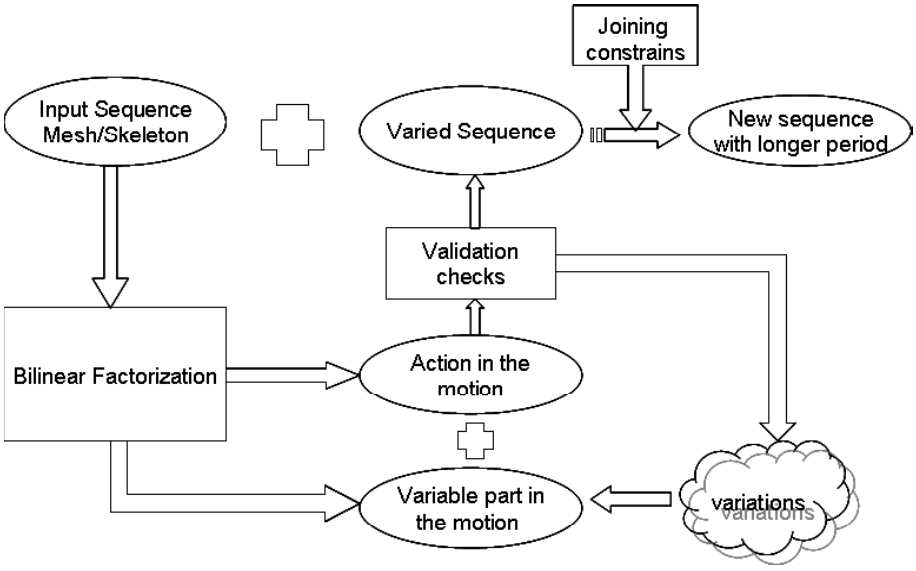


Figure 5.1: Workflow for creating motion variations

invariant component, which we shall call as the characterizing motion information, and which distinctly characterizes all instances of the same activity. We factor out this invariant from a given motion sequence and the introduce variations in the remainder. Our approach differs from previous work in three distinct ways:

1) We present a powerful technique for computing a bilinear model for the entire motion, which supports generation of motion variations while preserving its principal characteristics. For this, we use the asymmetric bilinear model [144, 147] applied to a motion. Given one instance of the animation sequence, either the complete animation sequence or just the keyframe representation, we first use LLE to obtain the characterizing motion information as a curve in low dimensional space and the reconstruction matrix discussed in the earlier chapters. It is this reconstruction matrix that encapsulates the variation in motion. We then apply singular value decomposition (SVD) to the reconstruction matrix to result in a set of scalar variables which can be adjusted, say, with controlled randomness, to provide variations in the given motion. These steps are described in detail in Section (5.2).

2) We have devised an effective method of joining together varying instances of a motion segment for generating a longer repetitive motion sequence. Again, the segments are composed together in LLE space while at the same time preserving desired physical properties for the "join" using property maps in LLE space [147] as described in Chapter 4.

3) An important and significant by-product of the above two processes is that our method works equally well for skeleton and mesh-based models.

5.2 Proposed method

The complete workflow for our method is shown in Figure (5.1). The initial input consists of the geometric data in the frames (or keyframes) for a given motion. The animator may choose to select all of the geometric elements in each frame (for skeleton animation it is joint angle data and for mesh animation it is vertex coordinate data), or may choose a subset of the geometric data, to avoid any parts that cannot be varied. The first step is bilinear decomposition and the singular value decomposition of the reconstruction matrix. Next, targeted variations are created as described later in subsection (5.2.3). Validation checks are carried out for each frame in the varied motion segment. This consists of different types of checks – say, for skeleton model, the balance of every frame or foot-ground position, or for a mesh model, constancy of volume or area of the mesh model. Here inverse kinematics is also used to make small value adjustments that are required to satisfy any hard constraints such as maintaining contact with the floor. Once a validated variation of the given motion segment is generated, the next step is to join it with the preceding motion segment; this is described in subsection (5.2.4).

5.2.1 Asymmetric Bilinear Model Decomposition

As described in Section (3.3), we can decompose the character animations into two parts: characterizing motion information and variable motion. Applying LLE yields us the nonlinearly embedded representation of the characterizing motion information. Then by setting the GRBF mapping from embedding space to the original high dimensional space, we obtain the variable part in the motion.

5.2.2 Computation of Characterizing Motion Information

Our first experiment was to verify that adding variations in the variable part do not alter the characterizing motion. For this we carried out a number of experiments. We randomly perturbed the values (small perturbations) in the reconstruction matrix, and created a new animation sequence. Then we obtained the LLE embedding for this new animation sequence. The results for one such typical variation for the walking motion of Figure (3.3-a)) above are shown in Figure (3.7). We can observe that the embedding of the varied animation sequence is almost identical to that of the original motion. The same results were achieved for other motions as well. Another option would have been to keep the reconstruction matrix unchanged, but to vary the characteristic curve in LLE space. However, the result of even small variations in the embedding curve is far more unpredictable, somewhat similar to changing keyframes or physical parameter values. Hence, creation of controlled or targetted variations while maintaining the distinguishing characteristic of a given motion would be very difficult, and certainly not something that an animator would find easy to specify.

5.2.3 Control Factors for Targeted Variation

While we have seen that perturbations of the motion reconstruction matrix indeed yield motion variations that leave the distinguishing action in the motion unaltered, the large matrix form makes it rather unwieldy for use by an animator, except for simple matrix transformations, or un-targeted random perturbations. Ideally, we would like to provide the animator with just a few handles to specify motion variation and further we would like these controls to be related to features of the skeleton. For this, let us again consider our example of the walking motion (Figure 3.3). It has a deformable skeleton with 62 DOFs, and consists of 150 frames. This results in a reconstruction matrix with 62 rows and 150 columns. Singular value decomposition (SVD) is a common technique for the analysis of multivariate data [148]. Let B denote an $n \times N$ matrix of real-valued data with rank r. The equation for SVD of B is as follows:

$$B = USV^T \tag{5.1}$$

where U is an $n \times n$ matrix, S is an $n \times N$ diagonal sparse matrix, and V^T is an $N \times N$ matrix. Both U and V are orthogonal. And the elements of S are only nonzero on the diagonal, and are called as singular values. By convention, the ordering of the vectors is determined by a high-to-low sorting of singular values. Another important property is that the squares of singular values λ_i are equivalent to the eigenvalues of the covariance matrix of B.

Singular values λ_i of the matrix B can be used to control motion variation, and we denote them as variation control factors (VCFs). They provide simple to use control handles, as change in a single VCF by multiplying it by a scaling variable s in a variation over the entire motion. Moreover, through various experiments we observed that these scalar variables have some interesting properties which makes

their use simple and intuitive. Variations in animations can be procedurally encoded as suitable changes in VCFs for each instance of the animation. In order to assist the animator to associate scaling variables in VCFs with the different features of the character, we can easily pre-generate a table showing the DOFs significantly affected by each scaling variable, for any given motion. Significant affect is assumed when the change in the motion curve of any DOF exceeds a given threshold. Table (5.1) shows an example of one such table for the walking motion (Figure 3.3). A DOF can be associated with multiple VCFs; and a VCF can affect multiple DOFs. This table can be used by animators to decide on the VCFs that need to be varied according to which features are to be changed (see experimental results in Section(5.3)). An interactive tool is easily developed with intuitive control for specifying skeleton feature-based variations.

The following are our observations about VCFs:

Local effect Only a small subset of DOFs is affected by a change in any one of the λ_i variables. Further, increase/decrease in the value of λ_i causes the affected DOFs also to increase/decrease in a corresponding fashion. A map of λ_i variables and affected DOFs can be used by the animator to make adjustments according to whether certain DOFs are to be changed or not. This is illustrated in Figure (5.2). We scaled λ_1 by a scaling variable of 1.1, and generated the variation in motion. By comparing the average change in every DOF, shown in Figure (5.2) a) and the motion curve for every DOF shown in Figure (5.2) b), we observe that λ_1 affects DOF No. 3 significantly, and DOF No. 54 and DOF No. 30 to a lesser extent. Other DOFs are affected only very slightly.

Additive ability Changes in individual values are additive, as can be expected for

features	i	features	i
root position	1,5,48 49,50	root orientation	22,34,36 37,39,41
lower back	18,47	upper back	42,43,45
thorax	40, 52	head	44, 46 53, 54
lower neck	18,47	upper neck	30,31
right humerus	32,35	left humerus	20,21,29
right radius	7,28	left radius	23
right wrist	25	left wrist	6,27
right hand	12,19	left hand	2,51
right thumb	55	left thumb	9
right femur	4	left femur	8,16
right tibia	33	left tibia	39
right foot	10,13,15	left foot	11,14,17
right toes	3	left toes	3

Table 5.1: Association between VCFs and joint associated with a DOF. i is the index of VCFs.

SVD. We can either change λ_i and λ_j separately, or change the two together. The final effects of the two operations are the same. Figure (5.3) illustrates this with an example. We scaled λ_1 by a factor of 1.1 to create variation A of the animation sequence, and scaled λ_2 by a scaling variable of 1.1 separately to create variation B. The modified motion curves for affected DOFs for A and B are shown in Figure (5.3) a) and (5.3) b); then we scale both together by the same factor, and obtain the variation AB; modified motion curves for the same DOFs are shown in Figure (5.3) c). Figure (5.3) d) shows the difference between the variation of AB alone and the sum of the variation A and the variation B. For an animator this gives the flexibility of carrying out changes individually.

Changes in DOFs, though small, do not guarantee satisfaction of hard constraints, such as, feet having to be always in contact with the floor, etc., for a walking

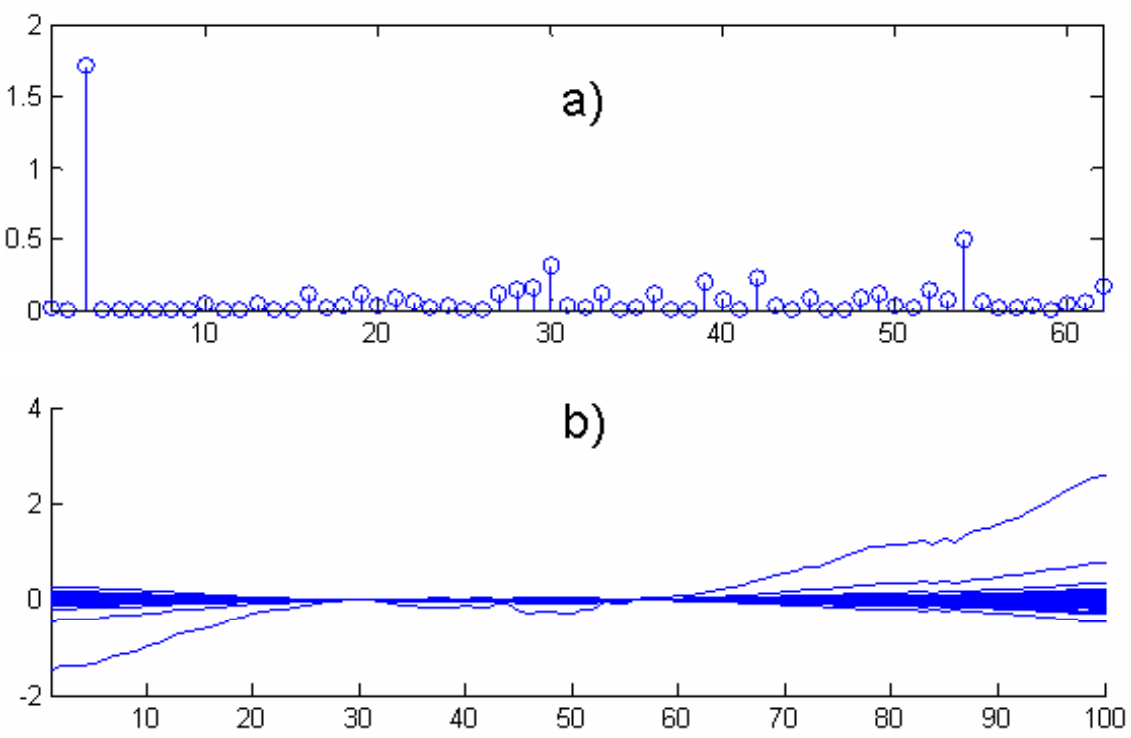


Figure 5.2: Locality of VCFs value. The horizontal axis of a) represents the DOFs; vertical axis of a) represents the average changes in every DOF percentage-wise. The horizontal axis of b) represents the frame indices; and vertical axis of b) represents the difference value of DOFs.

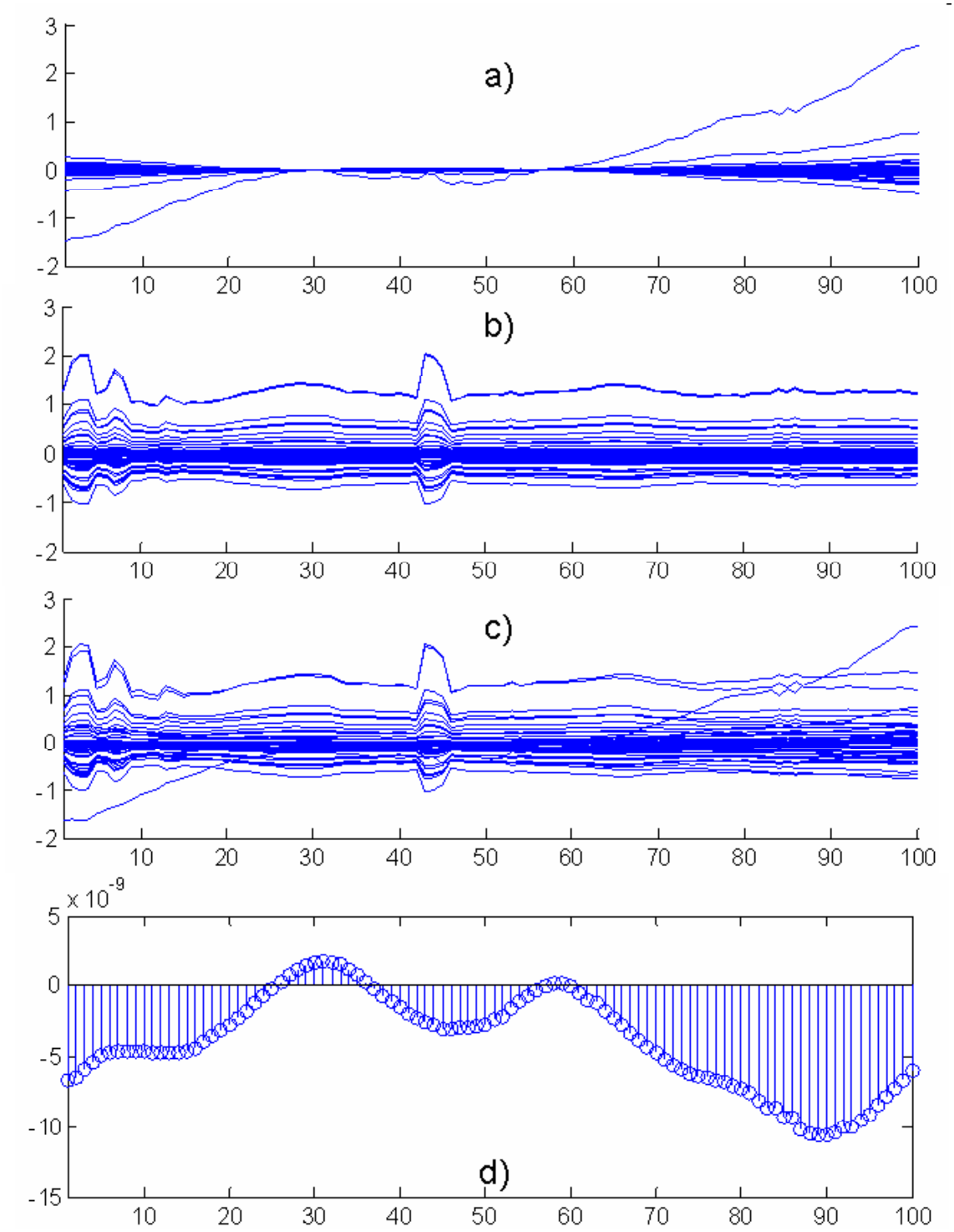


Figure 5.3: Illustration for showing additive property of VCFs. The horizontal axes of a), b), c) and d) are frame indices; and the vertical axes are difference values.

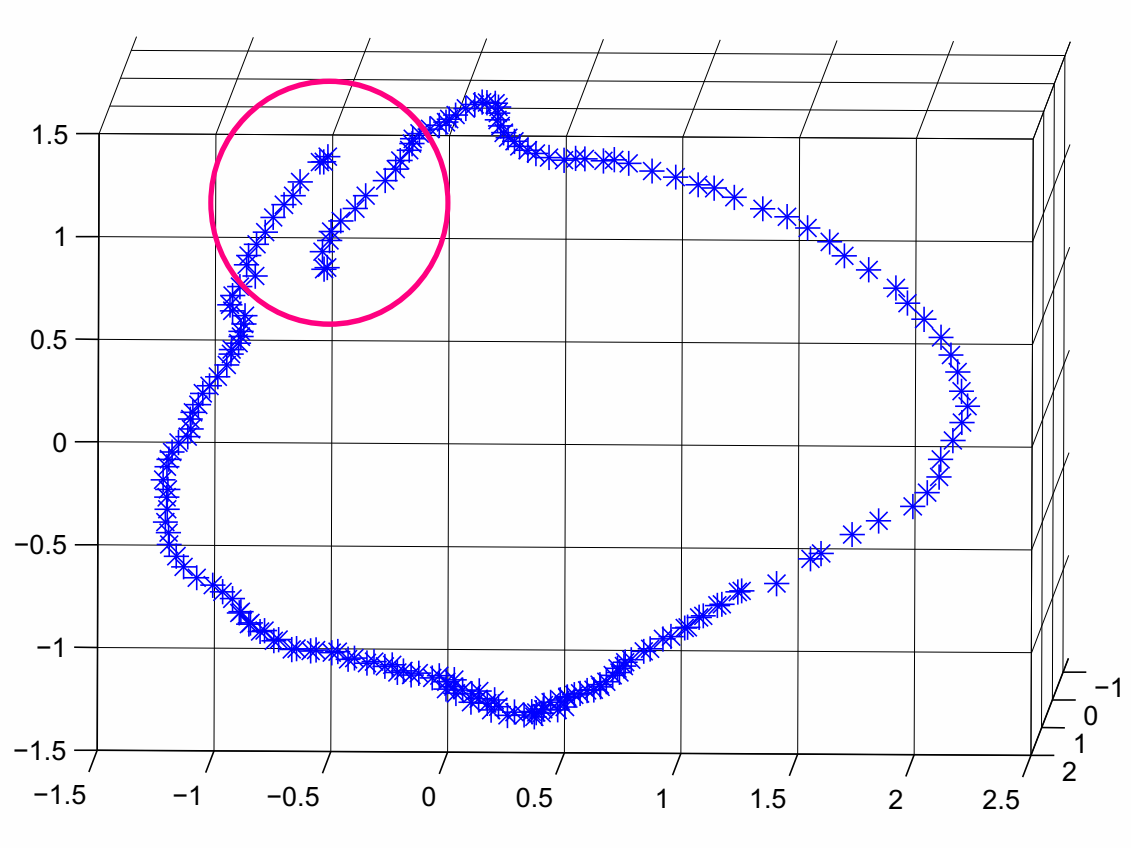


Figure 5.4: Overlapping motion fragment in $3-D$ LLE space (parallel curve segments inside the circle)

motion. To avoid violation generation, we only change the generating variations a small amount at each step. If a violation occurs at a certain step, the change in variations will be discarded and the variation generation go back to its values in the previous step.

5.2.4 Joining Two Motion Segments

To join two motion segments smoothly is non-trivial. The traditional approach is to carry out this operation in the frame space. For skeleton models this usually amounts to ensuring smooth changes in individual DOFs through the use of suitable

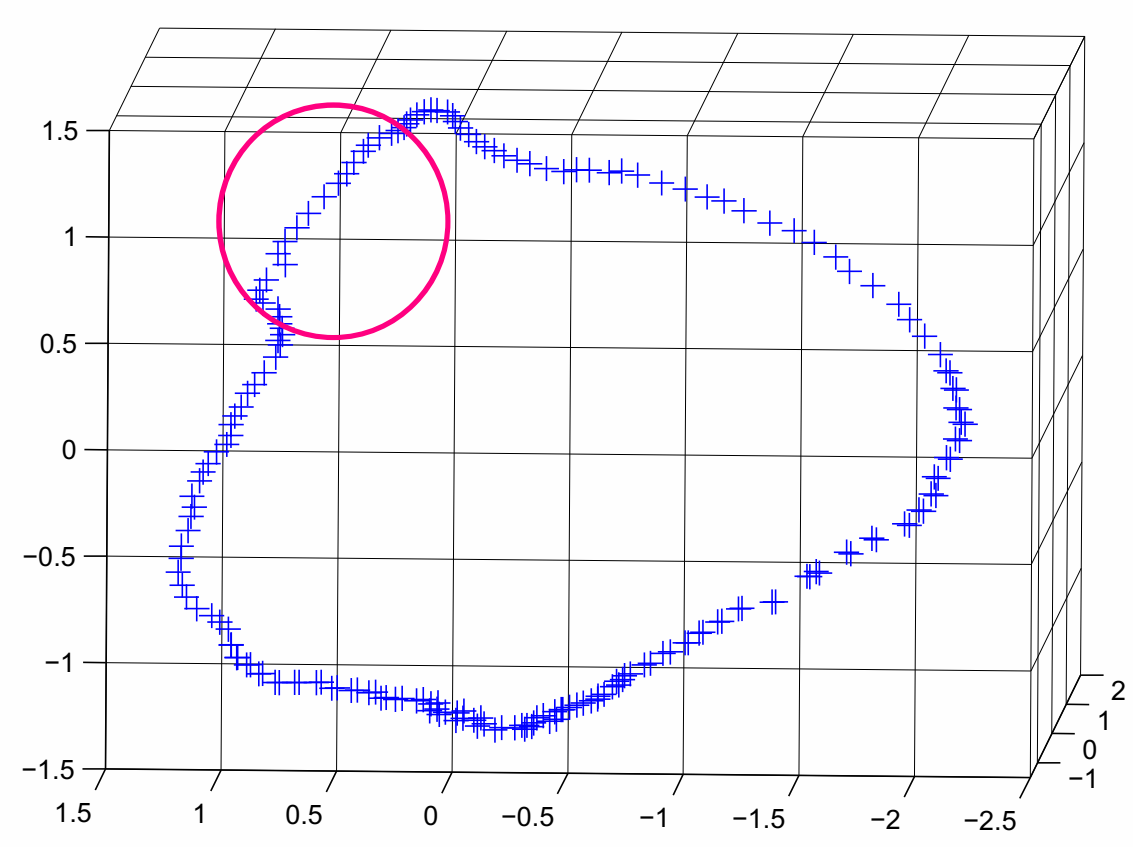


Figure 5.5: Smooth join in LLE space

interpolants. For mesh models, it is obviously much harder. And simple interpolation may result in unnatural solutions. What would be needed is more similar to morphing with appropriate constraints, a computationally expensive operation.

Here, we propose a method that blends two segments directly in the embedding space. Because the variation in segments is created from the input sample, we can ensure that the sampling rates of the varied segment is the same as the original. The first task in joining the segments together is to determine the temporal relation of the two segments. There are two cases: 1) the two segments have a overlapping period where the two motions exist at the same time; or 2) there exists a gap in the time sequence where neither segment exists.

To determine which case the given input belongs to, we put the two segments together as a set of data and apply LLE to them. As Figure (5.4) shows, if in embedding space, the motion curves of the two segments have parallel or overlapping portions, then it belongs to the first case; else if the motion curves of the two segments have a gap between them, then it follows that this is the second case.

In the first case, we blend the two motion curves in embedding space, then convert the new blended sequence back to frame space. This approach is based on the observation that parallel parts actually represent the same motion with slight perceivable differences. Therefore, we can search the area in between these parallel segments in the embedding space for similar motion. The solution should satisfy two constraints: 1) the blending result should be smooth; 2) the blended frame should satisfy the desired physical properties, such as, balance for skeleton models and volume for mesh models.

For the second case, we predict the motion curve in the gap using the fixed physical properties constraints with the method described in Chapter 4, which works

well both for skeleton and meshes and uses pre-computed property maps in LLE space.

5.3 Experimental Results

To demonstrate the capabilities of our proposed method, we carried out a number of experiments on both skeleton and mesh based animations.

Skeleton animation: As can be seen in Figure (5.7), the targeted features for this variation are the left and right tibia. When changes are made to VCF with indices 33 and 39 these features are affected. At the same time, for other DOFs, the variations are too small to notice. Given the input walking animation with 150 frames, we modulate the VCF with indices 33 and 39, which are associated with left and right tibia, with scalar factor s. The first row shows the frame No. 19 and the second row shows the frame No. 80. When $s = 1$, the middle column, it represents the original frames. The first, second, fourth and fifth columns show the variations created with $s = 2, 1.5, 0.75, 0.5$. From this result we can observe that when we increase the value, the left and right thighs move toward each other; and when we decrease the value, the two move away from each other. This example demonstrates quite clearly that the locality and additive properties of VCFs make this method simple to use for targeted variations.

Mesh based animation: In Figure (5.6), we show results of our method on mesh based horse galloping. The original motion sequence has 12 keyframes, and each mesh has 8431 vertices. The 12 frames form a cycle. After bilinear factorization and SVD decomposition, we have in total 13 variables to control the variation added to the sequence with the 13th frame representing the first frame of next cycle and placed at the end to close the sequence. Correspondingly we set 13 modulation

factors s_i as $\{1.3, 1.4, 1.3, 1.3, 1.2, 1, 1, 1, 1, 1, 1, 1, 1\}$, for the corresponding VCFs. We demonstrate the original and the new animations in separate rows. Each column shows corresponding frames for original and the newly created animations.

We also show the difference between our generated sequence and original input visually in detail. For example, in column 3, we show the same frame from the fixed visual angle. As displayed in the first two rows, although the two meshes both represent the No. 10 keyframe, the angle between the two back legs are different, as shown in row 3. In the figure, the first row shows the No. 2 and 10 keyframes, and the second row shows the variation results we have generated. In third row, we compare and display the difference of front hooves, tail, back legs, and back hooves. For example, in third row, we observe that compared to the input keyframe 10, the height of the lifting left front leg is lower; the tail is more straight, and the angle between two back legs are smaller. All these variations are achieved under the constraints that the mesh volume is invariant during the motion. This is done using pre-computed LLE property maps as shown in Chapter 4.

5.4 Conclusion

In this chapter, we have presented a new method to create perceivable variations in a given animation while ensuring that the principal characteristics of the given motion are kept invariant. The input animation may be given as a set of keyframes or as a complete animation created using any of the available animation techniques. The method works for skeleton models as well as mesh models. We believe that the idea of factorizing a given single motion into a distinguishing characteristic part and a set of variation control factors is both interesting and powerful, and needs to be explored further. While computing the lower dimensional LLE space embedding for

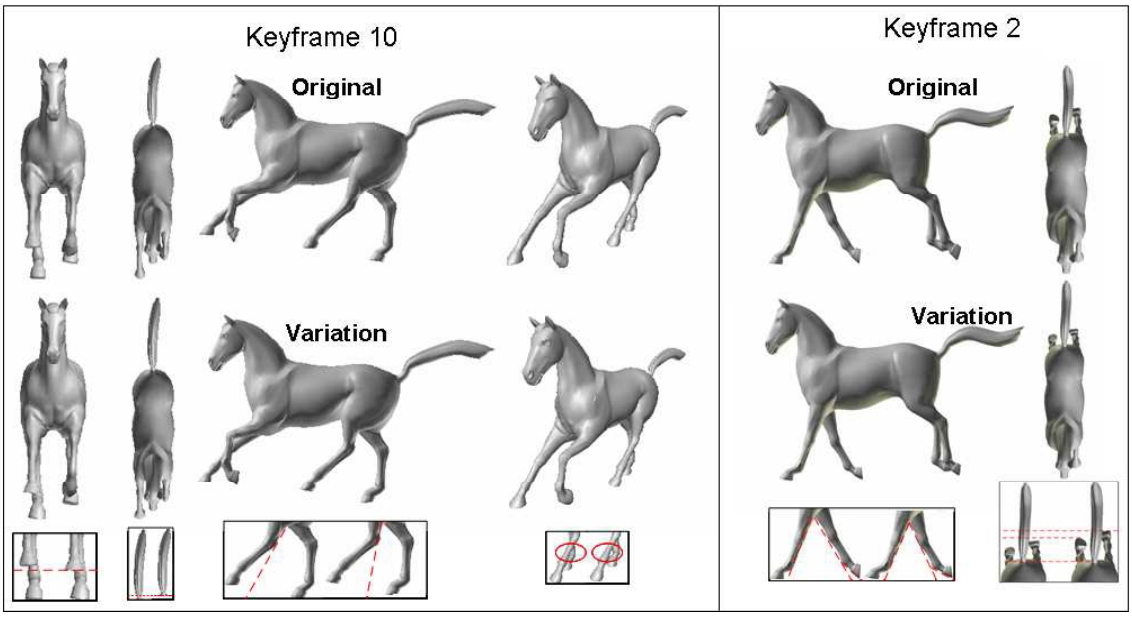


Figure 5.6: Mesh-based Variation

large models can be time consuming, reconstruction of frames from the embedding space is not, and can be done in real time.

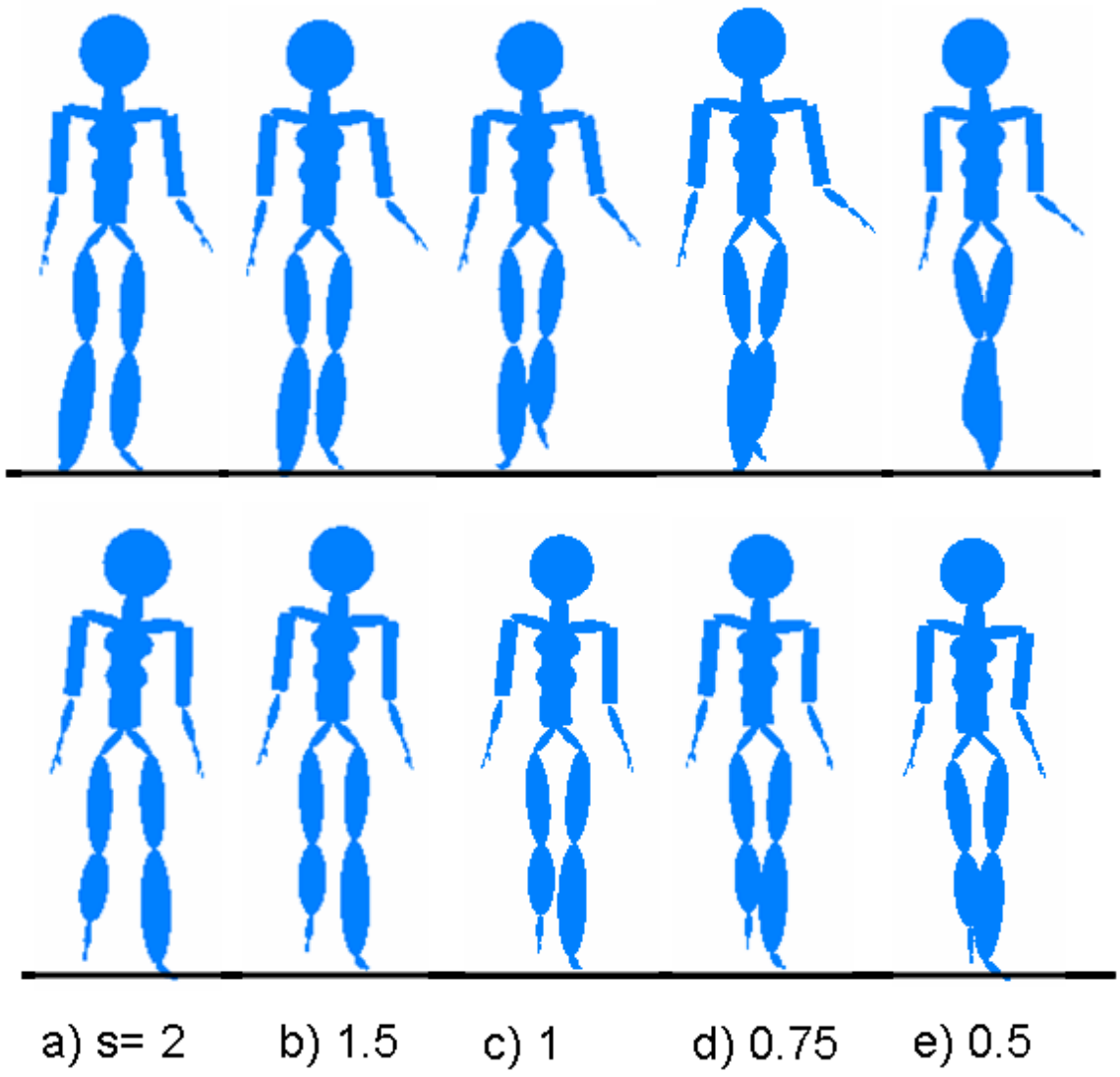


Figure 5.7: Skeleton-based Variation. s is the VCF factor. $s = 1$ represent the original frames, shows in column c); and variations are shown in columns a), b), d) and e).

Chapter 6

Content Based Key Information Extraction

In Chapter 4, we presented our method for extraction of the characterizing motion information in the form of a curve embedded in low dimension space, given just the keyframes. This embedding has subsequently been used to enhance keyframe animation for generating the complete animation sequence and for generating animations with variations satisfying desired physical properties. A very related question is regarding the inverse problem. That is, how do we obtain the keyframes given a complete character animation sequence. If we can use the characterizing motion content from LLE to extract the keyframes, then we can edit/correct/create new animation sequences for the character using previously developed techniques for this purpose. In this chapter, we discuss this third enhancement to the keyframe animation technique, namely keyframe extraction.

This chapter is organized as follows:

1) In Section (6.1), we introduce the importance of keyframe extraction and discuss major problems in selecting the optimal keyframe set for any given previously created or captured animation sequence.

2) Then in Section (6.2) we present our methodology which uses animation saliency

maps in low dimension embedding space, but with saliency values computed in the original space of the animation. We also present an iterative keyframe selection algorithm that minimizes reconstruction error.

3) In Section (6.3), we demonstrate the experimental results from an implementation of our method for both skeletal and mesh animations and compare it with other popular methods for keyframe extraction.

4) Concluding remarks are presented in Section (6.4).

6.1 Introduction to Keyframe Extraction

Keyframe extraction is a widely used technique in computer animation, computer games and in video processing. It uses a sequence of frames as the input, and outputs a simplified keyframe representation which is popularly adopted in animation blending, motion synthesis, compression and retrieval. Many animation techniques rely on a good keyframe representation. For example, mesh deformation is almost always performed only on keyframes of a sequence [149,150]. Motion editing is much easier to apply to keyframes than a whole sequence [47]. To compress animations, finding keyframes is usually the first step performed [20].

There are two typical questions when we have the input data for a complete animation sequence, and we wish to locate the keyframes to best represent it: 1) How many keyframes do we need? 2) Where are these keyframes located? To the first question, there is no absolute answer. It is always a tradeoff between the efficiency and quality. The more keyframes, the less lossy the reconstructed result will be with respect to the original sequence. It is also a question which depends on user requirements. The minimal reconstruction error is a global optimization question, which makes the question even harder to answer. The time complexity to

do an exhaustive search and select the set of keyframes with minimum reconstruction error is exponential, making this impractical, except for animation sequences of very small lengths.

The motion present in a sequence can be considered as a trajectory curve in high dimensional space. Each frame is a vertex on the curve. In the keyframe representation of any input animation, this trajectory curve has to be preserved. The keyframes selection problem is thus converted to a curve approximation problem, where the objective is to define a subset of vertices on a curve which can be used to reconstruct the curve with minimal error. As in lower dimensional space, such as $2D$ or $3D$, to best approximate the original curve, we usually need to locate the points where the absolute curvature, κ, of the curve is a local maximum [127].

To define κ we note that for a curve $y = f(x)$ that the absolute curvature at a point is :

$$\kappa = |\frac{y''}{(1 + (y')^2)^{\frac{3}{2}}}|$$

where $y' = \frac{dy}{dx}$.

For our purposes, we will assume that the points of interest will have $y' \ll 1$ (e.g. for maxima) such that:

$$\kappa = |y''| \tag{6.1}$$

For example, we have a simple 2D curve $y = \sin(x)$. In Figure 6.1, in addition to the start and end points, the point at $x = \frac{\pi}{2}$ and $x = \frac{3\pi}{2}$ should be included in key point sets rather than the point at $x = \pi$. In other words, when the point $< x_i, y_i >$ is a maximum value in Equation (6.1), we need to consider it as a key point to approximate the curve. On the other hand, when the point is an intermediate value in Equation (6.1), it means the vertex can be replaced by its neighbors .

However, the high dimensionality of the input animation makes the process

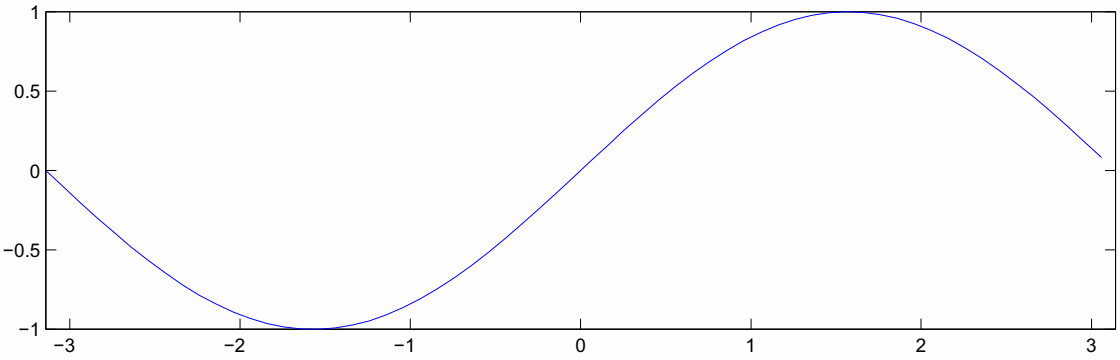


Figure 6.1: Example of $2D$ curve approximation.

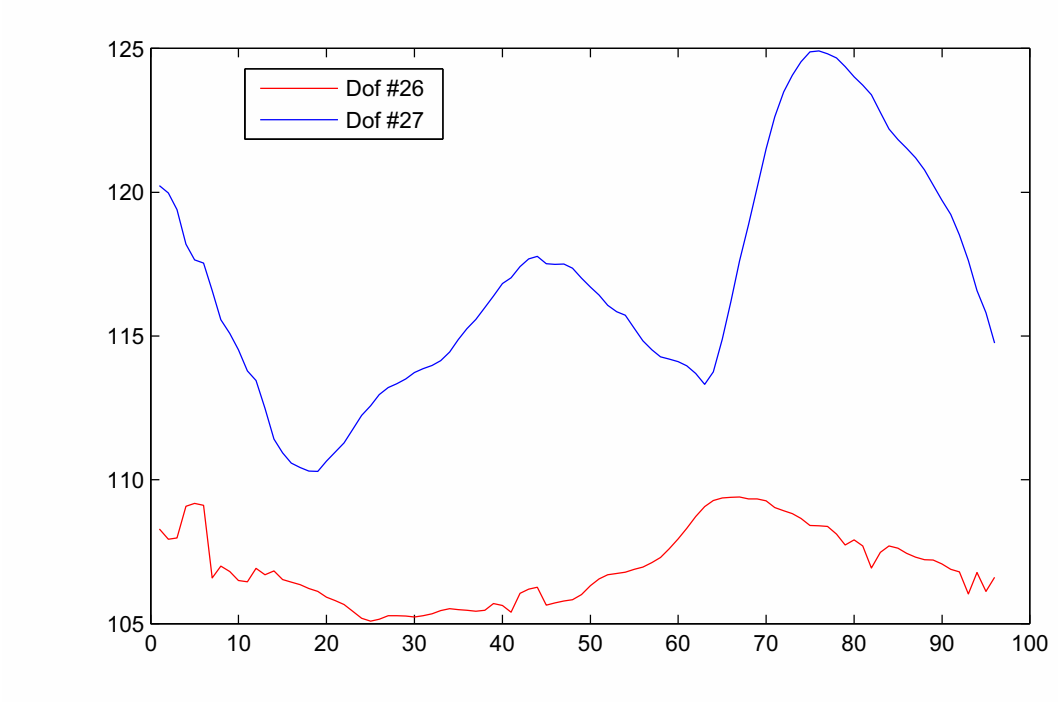


Figure 6.2: The motion curve of DOFs varied very much even for closed joint in skeleton model.

more complex. The $3D$ data has a large number of degree of freedoms (DOFs), and every DOF is a time variable function. Rarely is there a uniform changing pattern of the whole collection of DOFs. Retaining any one DOF's changing pattern may cause us to lose the pattern of other DOFs (see Figure (6.2)). Therefore, we need to take all DOFs into consideration globally. This problem becomes even more acute when applied on meshes, since the dimensionality increases many-fold. Meshes use geometry (i.e., vertices) and connectivity (e.g., triangles) information to represent a shape in $3D$. The interpolation among mesh data is more difficult than for skeletal data, because the latter has only to deal with angle interpolation which has a large tolerance for error. For mesh data, on the other hand, inaccurate interpolation may result in in-between frames having invalid data, such as self-intersections, a decrease in body volume, etc.

It is important to reiterate here that within the huge amount of DOFs, some DOFs contribute more than others to the motion performance and its perception. To isolate those DOFs we use animation saliency, a perception-inspired metric to help in identifying the important frames. Saliency, which characterizes the level of significance of the subject being observed, has been a focus of cognitive sciences for more than 20 years. It is closely related to many disciplines, including artificial intelligence, neuroscience, psychology, and recently, computer graphics [151]. Researchers first used saliency in 2D images to distinguish areas with higher visual attentions. Lee *et al.* [152] extended this idea to 3D meshes, and calculated the mesh saliency based on geometry features at multiple scales. Lee *et al.* use mesh saliency to perform the mesh simplification while preserving more visual features.

Let us recall that character animation data is of very high dimensionality and

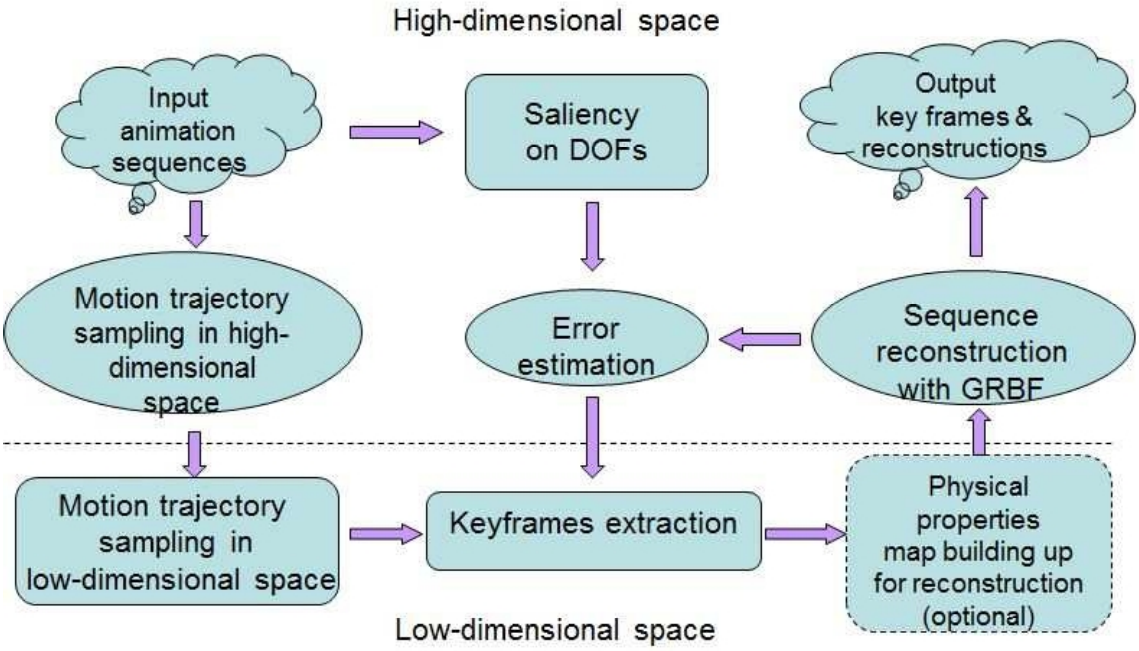


Figure 6.3: Flowchart for our keyframe extraction method.

has huge amount of information which contains important coherence and correlations [67]. To avoid the difficulties caused by the high dimensionality, our method for keyframe selection first projects the animation sequence to lower-dimensional embedding space. Now, every frame corresponds to a point in the embedding space. Next, we sample the curve in the embedding space and select the keyframe candidates with the largest values in Equation (6.1). Lastly, we use an iterative keyframe refinement scheme to minimize an error function which incorporates saliency of all DOFs in the input animation. Figure (6.3) shows the flowchart of our method.

The advantages of our method are following:

1. The time complexity is highly reduced because we cast the keyframe extraction in lower dimensional space.
2. With the saliency maps of DOFs in high dimensional space, we can find a set of keyframes which can construct the whole animation sequence with less loss

from the perception angle.

3. The reconstruction is done by having all contributions from the whole keyframe set. Plus, we use physical property maps to guide the reconstruction. Together, these make the reconstruction more reliable.
4. Our method can be applied to both skeletal animations, including motion capture, and to mesh animations.

6.2 Methodology

The vast information in character animation sequences has two layers: the static information and the dynamic information. The static information contains the appearance model definition, such as the geometric structure of the model, the texture of the surface of the model, etc. The dynamic information concerns model deformation and transformation, i.e., information directly related to the motion performed. In a few cases, we can isolate the dynamic information away from the static information, and represent the dynamic information as an equation of time such as the fast Fourier transform for water or cloth movement [153]. However, for 3D character animation, which usually has more complicated movement, such as in walking, running, dancing etc., it is too hard to isolate the movement with a space time equation. Hence, a sequence of animation frames is the chosen format to represent a character animation. And every frame contains both static information and dynamic information. As a consequence, all frames of a character animation sequence have coherence and correlations. Based on the representation types and the defined model, the size of every frame can be varied. The degrees of freedom (DOFs) will be larger when the character model becomes more finely detailed. Among the huge amount of DOFs, some of the DOFs are more meaningful for a viewer's perception

of the animation. As mentioned earlier, to locate these DOFs, we calculate the saliency of the animation that is due to motion.

6.2.1 Saliency Maps on DOFs

During the animation, from a perception viewpoint, DOFs contribute to motion unevenly. For example, for a walking character, those DOFs representing the mid-section contribute much less to the walking motion than those DOFs representing the knee or foot. Therefore, in our work, we apply DOFs based saliency on the animation sequence to capture what would be considered as the most visually interesting regions in an animation sequence. The human-perception inspired importance measure computed by our saliency operator results in more visually pleasing results for the animation created from the generated keyframes. Also we have noticed that some DOFs have large ranges of variation but change smoothly along frames. From an interpolation angle, although those DOFs can be easily interpolated by the same position of DOFs from neighbor frames, they may bring larger numerical reconstruction error than some other DOFs which have smaller variation range but sharp and irregular changes. An advantage of using saliency in our work is that it eliminates this bias caused by smoothly changing DOFs with large scalar value ranges.

For skeletal animation, we consider the ith DOF, dof_i, as defining a motion curve in 2D space. We use the standard curvature of motion curve as the saliency feature s. Then, similar to [151], we calculate the Gaussian-weighted center-surround differences for several center surround scales. Next, we normalize all the saliency maps at different scales, and sum them altogether to get the saliency value of dof_i. We will represent a DOF of a frame $\mathbf{f}_j$ ($j \in [1, n]$) in an animation sequence of n frames as dof_i^j ($i \in [1, N]$). For every dof_i^j, we build a neighbor set $N(dof_i^j, d)$ of all dof_i^k within distance d along the DOF curve dof_i. Here, we use the Euclidean

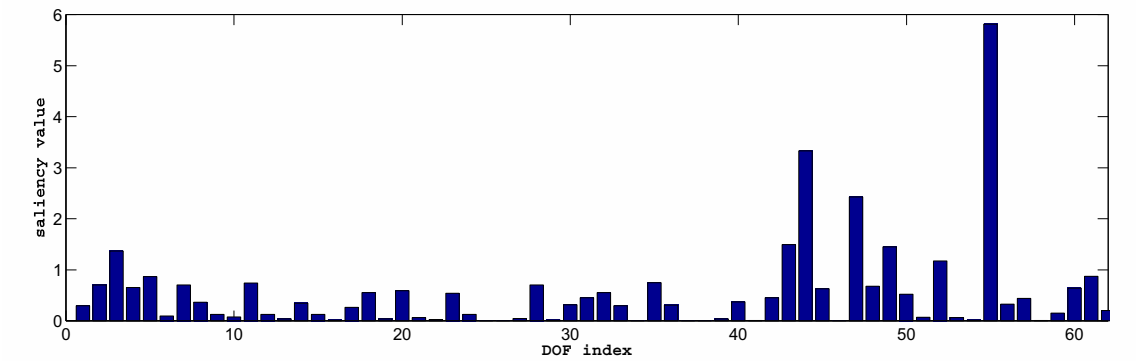


Figure 6.4: Saliency value for a running skeletal animation with 62 DOFs and 161 frames.

distance metric so we calculate the Gaussian-weighted average of saliency feature s for the vertices that are in $N(dof_i^j, 2d)$:

$$G(dof_i^j, s, d) = \frac{\sum_{k \in N(dof_i^j, 2d)} s(dof_i^k) \exp(-\frac{\|dof_i^k - dof_i^j\|^2}{2d^2})}{\sum_{k \in N(dof_i^j, 2d)} \exp(-\frac{\|dof_i^k - dof_i^j\|^2}{2d^2})} \tag{6.2}$$

Then, we build the saliency map M for a dof_i for the whole frame as the absolute difference between the Gaussian weighted averages computed at fine and coarse scales $d = 1, 2, 3$.

$$M(dof_i^j, s) = \|G(dof_i^j, s, d) - G(dof_i^j, s, 2d)\| \tag{6.3}$$

For mesh animations, other than the features we used for skeletal animation, we also use a saliency calculation similar to Lee *et al.* [152]. We use the DOFs of a frame $\mathbf{f}_i$ to build a graph. Every DOF is a vertex in the graph, and it is connected by edges. Since the frame $\mathbf{f}_j$ is a graph, for every dof_i^j, we build a neighbor set $N(dof_i^j, d)$ that is the set of all dof_k^j at most d path length from dof_i^j. For the saliency of a dof_i of the mesh we use a weighted combination of the average value over all $N(dof_i^j, d)$ and the motion curve saliency feature defined for skeletal

Figure 6.5: Saliency map for rabbit walking animation of 85 frames. Light color represent large saliency value.

animations. Figure (6.4) shows the saliency calculation result for a running skeletal animation consisting of 62 DOFs and 161 frames. Figure (6.5) shows the saliency map calculated for a mesh animation, a walking rabbit sequence consisting of 85 frames.

6.2.2 Motion Learning in Embedding Space

Using our LLE based technique described in Chapter 3, we cast the given animation sequence trajectory in the R^N space into a δ-dimensional projection of the N-dimensional frames ($\delta \ll N$), so that the point $< e_1, ..., e_\delta >$ in the embedding space generates a vector of $3D$ frames in the form of $\left\langle S(dof_i^1), S(dof_i^2), ..., S(dof_i^N) \right\rangle$. Recall that under the assumption that each data point and its neighbors lie on a locally linear patch of the manifold, each of the N-dimensional frames can be reconstructed as a weighted combination of its neighbors, such that the global reconstruction error is minimized.

6.2.3 Iterative Keyframe Selection with Saliency Map

An in-between frame $\mathbf{f}_i$ is defined as a parameterized interpolation of keyframes as in Equation (3.9). We re-write Eq. (3.9) into matrix form and get the following:

$$F_i = W(Key); \tag{6.4}$$

As a consequence, given an animation sequence it can be represented as a matrix A as follows:

$$A = W \times \Psi(Key) \tag{6.5}$$

This way we convert the keyframe selection problem into a constrained matrix factorization problem.

We extended the update rules in [26] for extracting keyframes in each iteration as given by the following equations:

$$\begin{aligned} \varepsilon &= argminE(A, A^*) \\ Key^{k+1} &\leftarrow Key^i \oplus Key^\dagger \\ \Psi^{k+1} &\leftarrow \Psi(Key^{k+1}) \\ W^{k+1} &\leftarrow A \times \Psi^{k+1^{-1}} \end{aligned} \tag{6.6}$$

where E is the error function and k is the number of iterations. As compared to existing methods, E, defined in Equation (6.7) below, incorporates the DOFs' saliency through suitable weights. In addition, unlike other existing error functions, E eliminates the bias caused by DOFs having large scalar value ranges but smoothly changing values.

$$E(A, A^*) = \sum_{j=0}^{n} \sum_{i=0}^{N} \omega_i \| dof_i^j - dof_i^{j^*} \| \tag{6.7}$$

where N is the number of DOFs; n is the number of frames; and ω_i is the weight based on the saliency.

The lower-dimensional embedding of an animation sequence is an important property; it is an invisible feature of the animation in high-dimensional space. A good assumption for keyframe selection is to choose the critical vertices e_i in the embedding space so that those e_i can best represent the motion trajectory. The assumption is that the corresponding frames $< dof_i >$ in the R^N space also best represent the input motion trajectory in R^N space. To select the keyframe candidates, we apply Gaussian filters and locate the e_i which have large values in Equation (6.1) in the embedding space.

Table 6.1: Test for number of keyframes removed versus reconstruction error.

removed #		5	15	26	37	45
error in total	7293	9408	12485	10396	9286	9671
removed #	56	67	76	88	97	106
error in total	10156	9909	9217	11313	11841	10580

Table 6.2: Test for number of keyframes added versus reconstruction error.

added #		11	21	31	41	51
error in total	7293	7323	6783	6301	7140	7059
added #	61	71	81	83	91	101
error in total	7168	7089	6594	6591	7290	6854

6.2.4 Number of keyframes

For any keyframe selection methods, there is always a tradeoff between quality and quantity. A larger number of keyframes definitely leads to less reconstruction error locally or/and globally. The maximal keyframe set consisting of all the frames has zero reconstruction error. However, as we increase the number of keyframes, reconstruction error decreases slower and slower. Here we give Tables (6.1) to illustrate the idea of the tradeoff between number of keyframes and the reconstruction error. For this illustration, we use a skeleton model animation sequence with 111 frames, depicting a human character running. Tables (6.2) illustration showing that adding another frame as keyframe does not cause huge reduction in reconstruction error. For example, adding frame #11 into keyframes set, causes the reconstruction error to decrease from 7485 to 7410 (about 1% reduction in error).

6.3 Experimental Results

In this section, we demonstrate the experimental results from an implementation of our method and compare it with other popular methods for keyframe extraction as follows:

Table 6.3: Skeletal animation sequences.

data	run 1	walk 1	run 2	walk 2
# of frames	161	188	321	341
data	jump	dance 1	salsa	dance 2
# of frames	550	851	901	941

Uniformly distributed method This method selects the keyframes at even intervals throughout the whole animation sequence.

PCA-based method This method maps the whole animation sequence into a linear embedding space. Then, the method calculates the absolute curvature value for every embedding point and sorts the corresponding frames according to this. The top k frames will be chosen as keyframes. If a group of keyframes are located within a very small range, the method chooses the center one as the keyframe and removes the others from the keyframe set.

Halit and Captin method [129] This method first maps the animation into k-dimensional embedding space with PCA ($k \in [7, 10]$). Then, the method analyzes the 2D curve saliency for each dimension individually and locates the points with high curve saliency. After summarizing all dimension results, the method chose the saliency value points larger than average curve saliency value as the keyframes candidates. Keyframes are finally selected via a clustering step.

These tests can be separated into two sets: 1) tests on skeletal models; 2) tests on mesh models. All errors discussed are average DOF reconstruction errors.

6.3.1 Experiments on Skeletal Models

We applied our method to a group of skeletal animation sequences (table 6.3) with varying number of frames. The skeleton in all the input animations contains 62

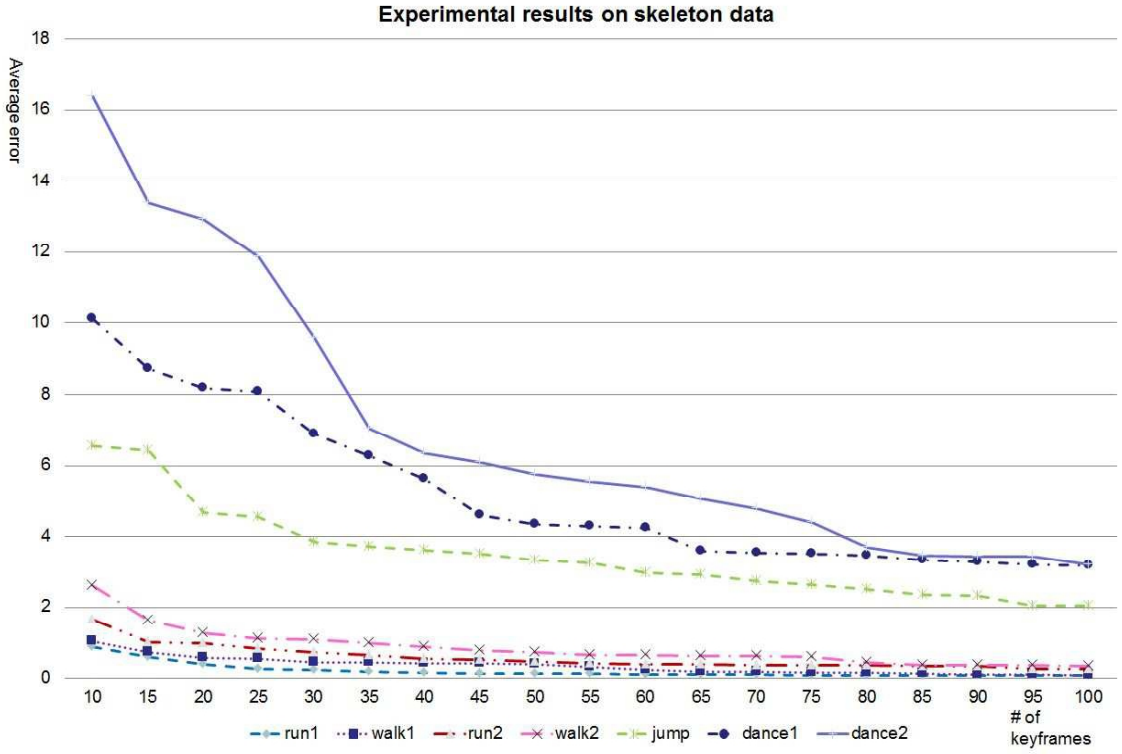


Figure 6.6: Experimental results on skeletal animations

DOFs. The reconstruction error is shown in Figure (6.6).

Compared to methods which use PCA, we see that LLE is a much better way to generate reliable embedding when the input data represents complex motion with many correlations and coherence. For example, Figure (6.7) shows a comparison between our method and the PCA-based method on a skeletal dancing sequence with 902 frames. The upper two plots *a*) and *b*) show the embedding result for LLE and PCA. PCA results in a very noisy graph curve for the segment (inside the circle) near the end of the input sequence. Details can be seen in the lower two plots *c*) and *d*). We believe this is due to the fact that PCA is a linear projection and LLE is a nonlinear projection. For complex motions like dance, it is quite obvious that an assumption of non-linearity is closer to the content structure. Based on the

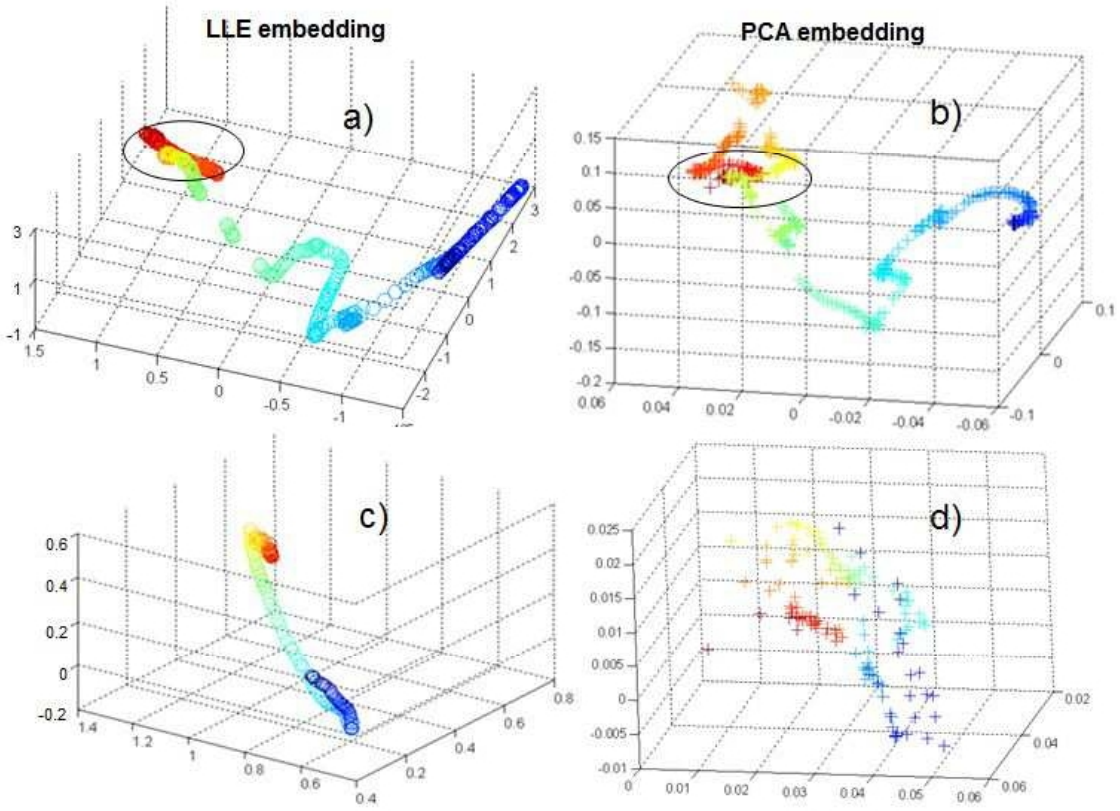


Figure 6.7: 3-dimensional embedding results shows using LLE is a good quality embedding whereas PCA yields a less coherent, noisy one.

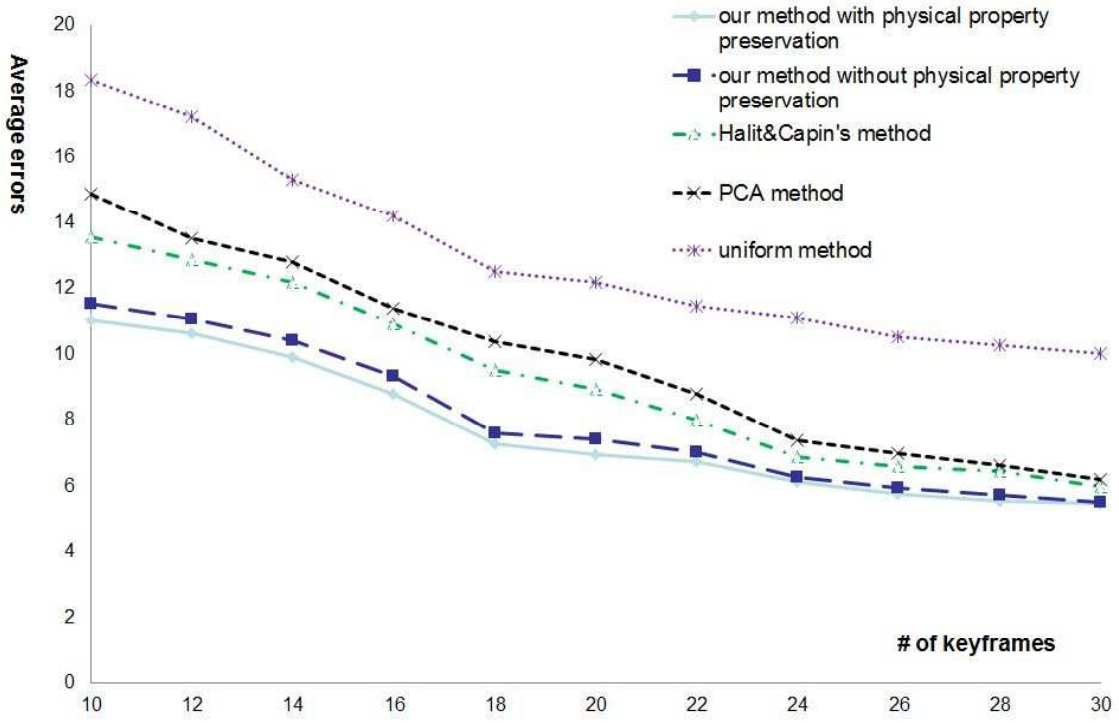


Figure 6.8: Comparison between our method with a uniformly distributed method, Halit & Capin's method, and PCA-based method on Salsa Data.

embedding result of the PCA-based method, the selected keyframes result in larger reconstruction error.

Another example we used is a salsa dancing sequence with 901 frames. In this comparison experiment, we chose the following methods for keyframe selection - uniformly distributed, PCA, the method of Halit & Capin [129], and our method with and without using property maps during reconstruction. We show the comparison result in Figure (6.8). From this figure, we can see that as the number of keyframes increases, the reconstruction error decreases. When it has same number of keyframes, LLE with property map reconstruction has the minimum reconstruction error, and LLE with simple linear interpolation reconstruction has a slightly worse performance. As we had expected, PCA-based methods have far higher error.

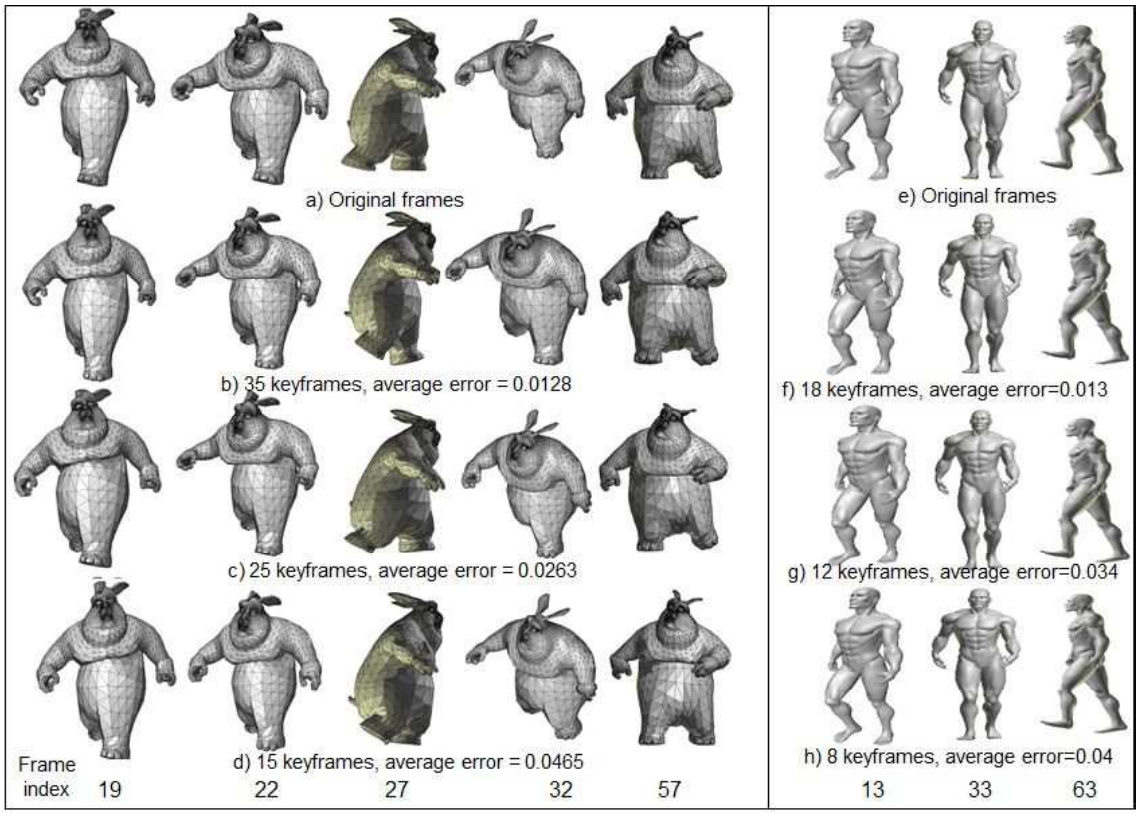


Figure 6.9: Experimental results for rabbit and walking man animations. For rabbit animation, the frames in the first row a) come from the original animation sequence. The frames in the second, third, and four rows come from reconstruction sequences with 35, 25, and 15 keyframes, respectively. For walking man animations, the frames in the first row e) come from the original animation sequence. The frames in the second, third, and four rows come from reconstruction sequences with 18, 12, and 8 keyframes, respectively.

From a computational time perspective, uniformly distributed method is very fast, but the reconstruction results it provides are clearly poor.

6.3.2 Experiments on Mesh Models

Compared to skeletal data, mesh data usually cause more difficulties for keyframe extraction. As we have mentioned one great advantage of our method is that it works equally well for skeletal and mesh animation data. We tested our method

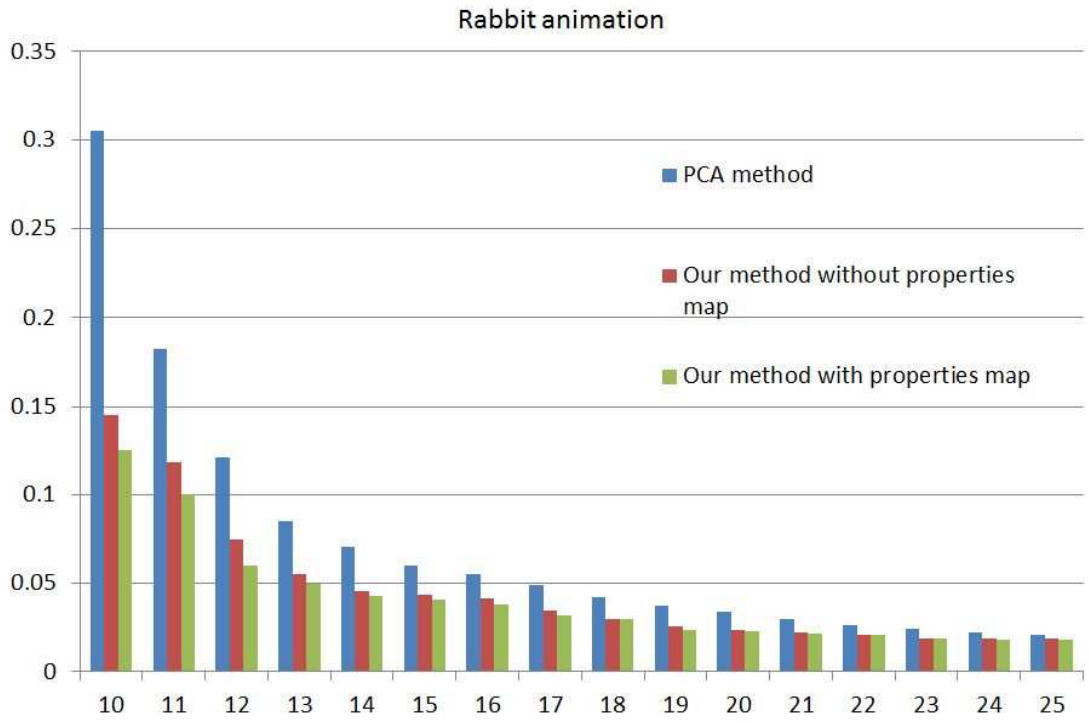


Figure 6.10: Comparison between our method and the PCA method on the rabbit animation sequence. The $x-$axis are the number of keyframes we used to reconstruct the whole animation sequences; and the $y-$axis shows the average reconstruction error.

on a mesh animation sequence depicting the motion of a rabbit, created by the Big Buck Bunny animation group [154]. The original sequence contains 85 frames. Every frame contains 4138 vertices, and in total 12414 DOFs. Figure (6.9) shows the reconstruction result with different numbers of selected keyframes in total. Row a) shows selected frames from the original sequence. Row b) shows the frames constructed with 35 keyframes and the average error for construction is 0.0128. Row c) shows the frames constructed with 25 keyframes and the average error for reconstruction is 0.0263. Row d) shows the frames constructed with 15 keyframes and the average error for reconstruction is 0.0465.

The comparison result between our method with PCA-based method is shown

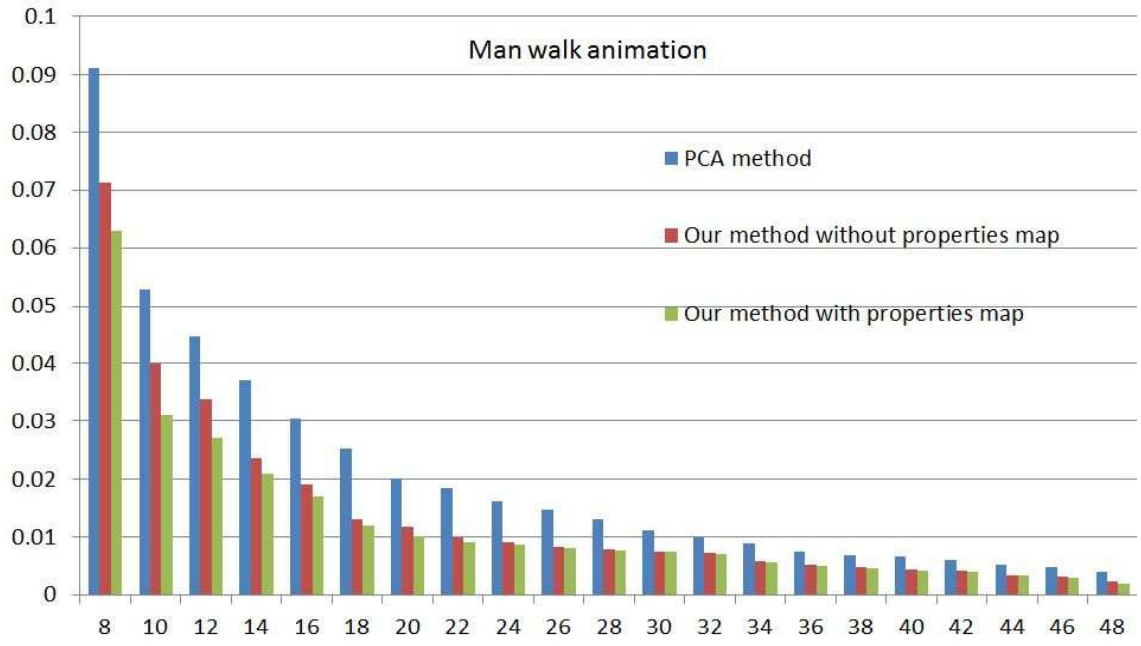


Figure 6.11: Comparison between our method and the PCA method on the man walk animation sequence. The $x-$axis are the number of keyframes we used to reconstruct the whole animation sequences; and the $y-$axis shows the average reconstruction error.

in Figure (6.10). We can see that with the same number of keyframes, our method's selection of keyframes generates the original sequence with much less error. Uniform method causes many topology errors; and Halit and Capin's method works only on skeleton animations. Therefore there is no comparison shown for these two methods here. We can also use the physical properties map preservation method, as presented in Section (4.2) to help us generate the mesh frames. Figure (6.12) shows the volume properties map for the rabbit animation. We also tested our method on a man walking animation (mesh data) with 49911 DOFs and 94 frames. The comparison between our method with PCA-based method is shown in Figure (6.11). Here again, we can see that with the same number of keyframes, our method selects keyframes so as to reconstruct the original sequence with much less error.

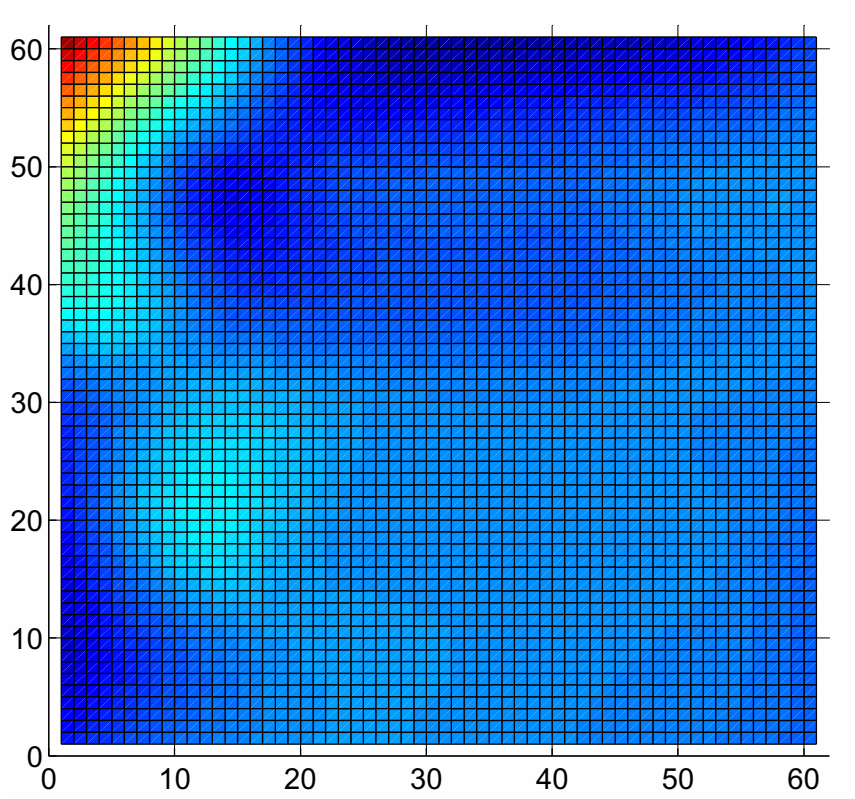


Figure 6.12: The volume properties map we used to reconstruct the sequence. We take the embedding space and divided it into a 61×61 grid, shows as the x-axis and y-axis. The scalar values shows the mesh volumes for corresponding meshes.

6.4 Concluding Remarks

In this chapter, we presented an effective method for optimized keyframe selection from complete animation or motion capture sequences. Our method works for skeletal as well as mesh data. Our solution uses animation saliency combined with dimensionality reduction using the locally linear embedding method. This way it transforms the problem from high dimensional space to a lower dimensional space to avoid the difficulties caused by high data dimensionality. In addition, using the mapping function from low dimensional space to high dimensional space, we represent the animation as a matrix multiplication form by keyframe matrix and combination

weights. Therefore, the time consuming optimal search for the matrix factorization problem in high dimensional space is simplified to a much more efficient way in low dimensional space. Yet, the error metric that is minimized uses animation saliency computed in the original high dimensional space itself, which helps to increase the fidelity of our keyframe representation. The reconstruction of the original sequence is achieved by interpolating every in-between using the whole set of keyframes. The experimental results show that our method produces better results for both skeletal and mesh animation sequences.

Chapter 7

Conclusions

In this chapter, we summarize the significant contributions in this thesis and also discuss future work. The organization of this chapter is as follows.

- First, in section (7.1), we summarize our methodology and contributions consisting of three new techniques for enhancing keyframe character animation technique using motion learning with locally linear embedding (3) to 6.
- Then, in section (7.2), we present the future works and some more concluding remarks about our research work.

7.1 Summary of Contributions

We have introduced a framework for character animation, which has three distinguishing components. Firstly, for any coordinated motion of a character, the framework uses a small set of key poses to learn the motion in a global fashion. This motion is learned as a path in a low dimensional space, using the nonlinear technique of locally linear embedding. Second, a reconstruction matrix is formulated which enables us to map any point in the low dimension embedding space to a pose in the original high dimension space of the character. Thirdly, the framework incorporates a new idea of physical property maps of the deformable shape in embedding space and enables the synthesis of the complete animation sequence based on desirable physical properties. We are not aware of any other earlier work that uses property maps in this manner. Such a framework will considerably ease the task of animators as they can now work with a small set of key poses and specify the synthesized motion in the form of desirable physical properties of the deformable shape during the course of the motion. We have demonstrated that the framework works well even with a small set of sample key poses, produces in-betweens with desirable properties, and is also not overly sensitive to the number and spacing of key poses. Mesh reconstruction is linear in computational complexity. However the property map computations take longer, and could be applied as a preprocessing step, for real-time animation.

After introducing our framework to create in-betweens, we have presented a new method to create perceivable variations in a given motion while ensuring that the principal characteristics of the given motion are kept invariant. As in the case of in-between generation, our method works for skeleton as well as mesh models. We believe that the idea of factorizing a given single motion into a distinguishing

characteristic part and a set of variation control factors is both interesting and powerful, and needs to be explored further. While computing LLE space embedding for large models can be time consuming, reconstruction of frames from the embedding space is not, and can be done in real time. From an implementation perspective we plan to investigate the possibility of computing the LLE embedding in a background thread, while the animation variations are being displayed.

Finally, we present an effective method for keyframe selection from complete animation or motion capture sequences. Again, our method works for skeleton as well as mesh data. Our solution uses animation saliency and combines it with the other components in the framework. This way it transforms the problem from high dimensional space to a lower dimensional space to avoid the difficulties caused by high data dimensionality. In addition, using the mapping function from low dimensional space to high dimensional space, we represent the animation as a matrix multiplication form by keyframe matrix and combination weights. Therefore, the time consuming optimal search for the matrix factorization problem in high dimensional space is simplified to a much more efficient way in low dimensional space. However, the error metric that is minimized uses animation saliency computed in the original high dimensional space. Again, the reconstruction of the original sequence is achieved by interpolating every in-between using the whole set of keyframes.

7.2 Future Work

Beyond the methods we addressed in this thesis, there are many possible research directions for further exploration. First of all, there are other shape properties we can take into account that may improve our estimates in future, including changes in topology for different types of motions. The more physical properties we use, the

better frame estimation results we can generate. Also, we notice that while computing LLE space embedding for large models can be time consuming, reconstruction of frames from the embedding space is not, and can be done in real time. From an implementation perspective we could investigate the possibility of computing the LLE embedding in a background thread, while the animation variations are being displayed. To achieve this, we should explore the possibility to build up a comparison system in embeddings space among different character animation data for one detailed motion. To build up the comparison system, we need to put different character animation sequences into one single embedding space. However, different types of data have very significant DOFs' representation, and a mapping relation between different data representation is necessary. The mapping relations could be built manually, but it would be very time consuming.

With a numerical representation of the animations in the form of embeddings, we could apply many motion editing and synthesis operations in the low dimension embedding space. For example, we can look into mapping of the motions between different character animation data types, such as from video clip to skeleton or mesh animations. Also, we can investigate techniques for adding some extra style to create new animations and to perform motion transfer to new models. Or we can join two different animations together with a step of blending in the motion embedding space.

Bibliography

[1] Twentieth century fox film corporation.

[2] Guodong Liu and Leonard McMillan. Segment-based human motion compression. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 127–135, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[3] Ascension Technology Corporation. http://www.ascension-tech.com.

[4] L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

[5] J. H., D. D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 47, New York, NY, USA, 2004.

[6] Ollie Johnston and Frank Thomas. *The Illusion of Life: Disney Animation*. Thomas, New York, hyperion edition, 1981.

[7] Gordon Cameron, Andre Bustanoby, Ken Cope, Steph Greenberg, Craig Hayes, and Olivier Ozoux. Motion capture and cg character animation (panel). In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 442–445, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[8] M. Gleicher. Animation from observation: Motion capture and motion editing. *SIGGRAPH Compututer Graphics*, 33(4):51–54, 2000.

[9] James Cameron. http://www.avatarmovie.com/index.html.

[10] http://www.boxoffice.com/.

[11] Steven R. Lantz. Magnetoconvection dynamics in a stratified layer. ii. a low-order model of the tilting instability (revised 03/94). Technical report, Ithaca, NY, USA, 1993.

[12] Antonio Carlos Sementille, Luís Escaramuzi Lourenço, José Remo Ferreira Brega, and Ildeberto Rodello. A motion capture system using passive markers. In *VRCAI '04: Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, pages 440–447, New York, NY, USA, 2004. ACM.

[13] A. Bruderlin and L. Williams. Motion signal processing. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 97–104, New York, NY, USA, 1995.

[14] Bobby Bodenheimer, Seth Rosenthal, John Pellainteractive, and Media Production. The process of motion capture: Dealing with the data, 1997.

[15] Victor Brian Zordan and Nicholas C. Van Der Horst. Mapping optical motion capture data to skeletal motion using a physical model. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 245–250, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[16] Kwang-Jin Choi, Sang-Hyun Park, and Hyeong-Seok Ko. Processing motion capture data to achieve positional accuracy. *Graphical Models and Image Processing*, 61(5):260–273, 1999.

[17] Tom Molet, Ronan Boulic, and Daniel Thalmann. A real time anatomical converter for human motion capture. In *Proceedings of the Eurographics workshop on Computer animation and simulation*, pages 79–94, New York, NY, USA, 1996. Springer-Verlag New York, Inc.

[18] V. B. Zordan and N. C. Van D. Horst. Mapping optical motion capture data to skeletal motion using a physical model. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 245–250, Aire-la-Ville, Switzerland, Switzerland, 2003.

[19] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 473–482, New York, NY, USA, 2002.

[20] Okan Arikan. Compression of motion capture databases. *ACM Transaction on Graphics*, 25(3):890–897, 2006. SIGGRAPH 2006.

[21] Alla Safonova and Jessica K. Hodgins. Construction and optimal search of interpolated motion graphs. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 106, New York, NY, USA, 2007. ACM.

[22] R. Williams. *The Animator's Survival Kit–Revised Edition: A Manual of Methods, Principles and Formulas for Classical, Computer, Games, Stop Motion and Internet Animators.* Faber & Faber, 2009.

[23] L.F. Cheong, Y. Wang, and H.L. Wang. Establishment shot detection using qualitative motion. In *In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages II: 85–90, 2003.

[24] A. Aner Wolf. Extracting semantic information through illumination classification. In *In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages I: 269–274, 2004.

[25] H.L. Wang and L.F. Cheong. Film shot classification using directing semantics. In *International Conference on Pattern Recognition*, pages 1–4, 2008.

[26] KeSen Huang, ChunFa Chang, Yu Yao Hsu, and Shi Nine Yang. Key probe: a technique for animation keyframe extraction. *The Visual Computer*, 21(8):532 – 541, 09 2005.

[27] O. Onder, U. Gudukbay, B. Ozguc, T. Erdem, C. Erdem, and M. Ozkan. Keyframe reduction techniques for motion capture data. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2008*, pages 293–296, May 2008.

[28] Tong-Yee Lee, Chao-Hung Lin, Yu-Shuen Wang, and Tai-Guang Chen. Animation key-frame extraction and simplification using deformation analysis. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(4):478–486, April 2008.

[29] V. Brian Z., A. Majkowska, B. Chiu, and M. Fast. Dynamic response for motion capture animation. *ACM Transactions on Graphics*, 24(3):697–701, 2005. Proceedings of SIGGRAPH 2005.

[30] Alla Safonova and Jessica K. Hodgins. Analyzing the physical correctness of interpolated human motion. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 171–180, New York, NY, USA, 2005. ACM.

[31] T. Mukai and S. Kuriyama. Geostatistical motion interpolation. *ACM Transactions on Graphics*, 24(3):1062–1070, 2005. Proceedings of SIGGRAPH 2005.

[32] Z. Deng, P. Chiang, P. Fox, and U. Neumann. Animating blendshape faces by cross-mapping motion capture data. In *SI3D'06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 43–48, New York, NY, USA, 2006.

[33] Hai-Yin Xu, Dan Li, and Jian Wang. Implicit curve oriented in-betweening for motion animation. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 87–91, New York, NY, USA, 2006. ACM.

[34] L. Reveret, L. Favreau, C. Depraz, and M. Cani. Morphable model of quadrupeds skeletons for animating 3D animals. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 135–142, New York, NY, USA, 2005.

[35] Jonas Gomes, Lucia Darsa, Bruno Costa, and Luiz Velho. *Warping and morphing of graphical objects*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

[36] T. Igarashi, T. Moscovich, and J. F. Hughes. Spatial keyframing for performance-driven animation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 107–115, New York, NY, USA, 2005.

[37] Scott Schaefer, Travis McPhail, and Joe Warren. Image deformation using moving least squares. *ACM Transaction on Graphics*, 25(3):533–540, 2006. Proceedings of SIGGRAPH 2006.

[38] Marc Alexa, Daniel Cohen-Or, and David Levin. As-rigid-as-possible shape interpolation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 157–164, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[39] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 35–42, New York, NY, USA, 1992. ACM Press.

[40] F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(6):567–585, 1989.

[41] Michael Gleicher. Retargeting motion to new characters. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 33–42, 1998.

[42] Hyun Joon Shin, Jehee Lee, Sung Yong Shin, and Michael Gleicher. Computer puppetry: An importance-based approach. *ACM Trans. Graph.*, 20(2):67–94, 2001.

[43] C. K. Liu, A. H., and Z. Popovìc. Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics*, 24(3):1071–1081, 2005. Proceedings of SIGGRAPH 2005.

[44] Eugene Hsu, Kari Pulli, and Jovan Popović. Style translation for human motion. *ACM Transactions on Graphics*, 24(3):1082–1089, 2005. Proceedings of SIGGRAPH 2005.

[45] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Transaction on Graphics*, 23(3):399–405, 2004. Proceedings of SIGGRAPH 2004.

[46] C. Lin and T. Lee. Metamorphosis of $3D$ polyhedral models using progressive connectivity transformations. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 11(1):2–12, 2005.

[47] R. W. Sumner, M. Zwicker, C. Gotsman, and J. Popović. Mesh-based inverse kinematics. *ACM Transactions on Graphics*, 24(3):488–495, 2005. SIGGRAPH 2005.

[48] Y. Cao, W. C. Tien, P. Faloutsos, and Frédéric Pighin. Expressive speech-driven facial animation. *ACM Transactions on Graphics*, 24(4):1283–1302, 2005.

[49] K. G. Der, R. W. Sumner, and J. Popović. Inverse kinematics for reduced deformable models. *ACM Transactions on Graphics*, 25(3):1174–1179, 2006. Proceedings of SIGGRAPH 2006.

[50] Robert W. Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 80, New York, NY, USA, 2007. ACM.

[51] Ilya Baran, Daniel Vlasic, Eitan Grinspun, and Jovan Popović. Semantic deformation transfer. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pages 1–6, New York, NY, USA, 2009. ACM.

[52] D. M. Gavrila and L. S. Davis. 3-d model-based tracking of humans in action: a multi-view approach. In *CVPR '96: Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, page 73, Washington, DC, USA, 1996. IEEE Computer Society.

[53] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *CVPR '98: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, page 8, Washington, DC, USA, 1998. IEEE Computer Society.

[54] M. Yamamoto, A. Sato, S. Kawada, T. Kondo, and Y. Osaki. Incremental tracking of human actions from multiple views. In *CVPR '98: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, page 2, Washington, DC, USA, 1998. IEEE Computer Society.

[55] Masanobu Yamamoto and Katsutoshi Yagishita. Scene constraints-aided tracking of human body. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:1151, 2000.

[56] Tat-Jen Cham and J.M. Rehg. A multiple hypothesis approach to figure tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, page 244, 1999.

[57] Quentin Delamarre and Olivier Faugeras. 3d articulated models and multi-view tracking with silhouettes. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 716, Washington, DC, USA, 1999. IEEE Computer Society.

[58] I. A. Kakadiaris and D. Metaxas. Model-based estimation of 3d human motion with occlusion based on active multi-viewpoint selection. In *CVPR '96: Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, page 81, Washington, DC, USA, 1996. IEEE Computer Society.

[59] Tom Drummond and Roberto Cipolla. Real-time visual tracking of complex structures. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):932–946, 2002.

[60] Ronan Billon, Alexis Nédélec, and Jacques Tisseau. Gesture recognition in flow based on pca analysis using multiagent system. In *ACE '08: Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, pages 139–146, New York, NY, USA, 2008. ACM.

[61] Abdullah Bulbul, Cetin Koca, Tolga Capin, and Uğur Güdükbay. Saliency for animated meshes with material properties. In *Proceedings of the 7th Symposium on Applied Perception in Graphics and Visualization*, APGV '10, pages 81–88, New York, NY, USA, 2010. ACM.

[62] Andrew Witkin and Michael Kass. Spacetime constraints. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 159–168, New York, NY, USA, 1988. ACM.

[63] James C. Thompson, Michele Clarke, Tennille Stewart, and Aina Puce. Configural Processing of Biological Motion in Human Superior Temporal Sulcus. *J. Neurosci.*, 25(39):9059–9066, 2005.

[64] Pedro Sáchez Orellana, Claudio Castellanos Sáchez, Edgar del Angel-Guerrero, and Tomá Mártiez-Arenas. Bio-inspired architecture for visual recognition

of humans walking. In Wen Yu and Edgar Sanchez, editors, *Advances in Computational Intelligence*, volume 116 of *Advances in Soft Computing*, pages 443–452. Springer Berlin/Heidelberg, 2009. 10.1007/978-3-642-03156-445.

[65] S.-J. Blakemore, P. Boyer, M. Pachot-Clouard, A. Meltzoff, C. Segebarth, and J. Decety. The Detection of Contingency and Animacy from Simple Animations in the Human Brain. *Cerebral Cortex*, 13(8):837–844, 2003.

[66] J. K. Aggarwal and Sangho Park. Human motion: Modeling and recognition of actions and interactions. In *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium*, pages 640–647, Washington, DC, USA, 2004. IEEE Computer Society.

[67] Katherine Pullen and Christoph Bregler. Motion capture assisted animation: texturing and synthesis. *ACM Transaction on Graphics*, 21(3):501–508, 2002.

[68] Odest Chadwicke Jenkins and Maja J. Mataric. Automated derivation of behavior vocabularies for autonomous humanoid motion. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 225–232, New York, NY, USA, 2003. ACM.

[69] O. Arikan, D. A. Forsyth, and J. F. O'Brien. Motion synthesis from annotations. *ACM Transaction on Graphics*, 22(3):402–408, 2003.

[70] J. Chai and J. K. Hodgins. Performance animation from low-dimensional control signals. *ACM Transactions on Graphics*, 24(3):686–696, 2005. Proceedings of SIGGRAPH 2005.

[71] D. H. U. Kochanek and R. H. Bartels. Interpolating splines with local tension, continuity, and bias control. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 33–41, New York, NY, USA, 1984.

[72] S. C. L. Terra and R. A. Metoyer. Performance timing for keyframe animation. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 253–258, New York, NY, USA, 2004.

[73] James R. Kent, Wayne E. Carlson, and Richard E. Parent. Shape transformation for polyhedral objects. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 47–54, New York, NY, USA, 1992. ACM Press.

[74] Brian Cabral, Nancy Cam, and Jim Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *VVS '94: Proceedings of the 1994 symposium on Volume visualization*, pages 91–98, New York, NY, USA, 1994. ACM Press.

[75] Aaron W. F. Lee, David Dobkin, Wim Sweldens, and Peter Schröder. Multiresolution mesh morphing. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 343–350, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[76] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[77] Martin Kraus and Thomas Ertl. Adaptive texture maps. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, HWWS '02, pages 7–15, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[78] Alfred R. Fuller, Hari Krishnan, Karim Mahrous, Bernd Hamann, and Kenneth I. Joy. Real-time procedural volumetric fire. In *Proceedings of the 2007*

symposium on Interactive 3D graphics and games, I3D '07, pages 175–180, New York, NY, USA, 2007. ACM.

[79] George Wolberg. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1994.

[80] Apostolos Lerios, Chase D. Garfinkle, and Marc Levoy. Feature-based volume metamorphosis. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 449–456, New York, NY, USA, 1995. ACM Press.

[81] T. Y. Lee and P. H. Huang. Fast and intuitive metamorphosis of 3D polyhedral models using smcc mesh merging scheme. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 9(1):85–98, 2003.

[82] M. Girard and A. A. Maciejewski. Computational modeling for the computer animation of legged figures. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 263–270, New York, NY, USA, 1985.

[83] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović. Style-based inverse kinematics. *ACM Transactions on Graphics*, 23(3):522–531, 2004. Proceedings of SIGGRAPH 2004.

[84] E. S. L. Ho, T. Komura, and R. W. H. Lau. Computing inverse kinematics with linear programming. In *VRST '05: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 163–166, New York, NY, USA, 2005.

[85] M. Neff and E. Fiume. Modeling tension and relaxation for computer animation. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 81–88, New York, NY, USA, 2002.

[86] Joseph Laszlo, Michiel van de Panne, and Eugene Fiume. Interactive control for physically-based animation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 201–208, New York, NY, USA, 2000.

[87] Vladimir Zatsiorsky. *Kinetics of Human Motion*. Human Kinetics, 1 edition, April 2002.

[88] L. Kovar, J. Schreiner, and M. Gleicher. Footskate cleanup for motion capture editing. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 97–104, New York, NY, USA, 2002.

[89] O. Arikan and D. A. Forsyth. Interactive motion generation from examples. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 483–490, New York, NY, USA, 2002.

[90] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 21(3):491–500, 2002.

[91] Kang Hoon Lee, Myung Geol Choi, and Jehee Lee. Motion patches: building blocks for virtual environments annotated with motion data. *ACM Transaction on Graphics*, 25(3):898–906, 2006. SIGGRAPH 2006.

[92] Paul S. A. Reitsma and Nancy S. Pollard. Evaluating motion graphs for character animation. *ACM Transaction on Graphics*, 26(4):18, 2007.

[93] C. Rose, M. F. Cohen, and B. Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics Application*, 18(5):32–40, 1998.

[94] L. Kovar and M. Gleicher. Flexible automatic motion blending with registration curves. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 214–224, Aire-la-Ville, Switzerland, 2003.

[95] Taesoo Kwon and Sung Yong Shin. Motion modeling for on-line locomotion synthesis. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 29–38, New York, NY, USA, 2005. ACM.

[96] Rachel Heck and Michael Gleicher. Parametric motion graphs. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 129–136, New York, NY, USA, 2007. ACM.

[97] Nikolaus F. Troje. Decomposing biological motion: A framework for analysis and synthesis of human gait patterns. *Journal of Vision*, 2, 2002.

[98] Y. Li, T. Wang, and H. Y. Shum. Motion texture: a two-level statistical model for character motion synthesis. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 465–472, New York, NY, USA, 2002.

[99] Jinxiang Chai and Jessica K. Hodgins. Constraint-based motion optimization using a statistical dynamic model. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 8, New York, NY, USA, 2007. ACM.

[100] Jianyuan Min, Yen-Lin Chen, and Jinxiang Chai. Interactive generation of human animation with deformable motion models. *ACM Transaction on Graphics*, 29(1):1–12, 2009.

[101] N. M. Thalmann. *Computer animation: theory and practice*. Springer-Verlag New York, Inc., New York, NY, USA, 1985.

[102] A. Witkin and Z. J. Popović. Motion warping. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 105–108, New York, NY, USA, 1995.

[103] A. Pentland and J. Williams. Perception of non-rigid motion: Inference of shape, material and force. In *Proceedings of the 11th International Joint Conferences on Artificial Intelligence*, pages 1565–1570, Detroit, MI, 1989.

[104] D. L. James and D. K. Pai. DyRT: dynamic response textures for real time deformation simulation with graphics hardware. *ACM Transaction on Graphics*, 21(3):582–585, 2002.

[105] L. Zhang, N. Snavely, B. Curless, and S. M. Seitz. Spacetime faces: high resolution capture for modeling and animation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 548–558, New York, NY, USA, 2004.

[106] A. Witkin and D. Baraff. Physically-based modelling. *ACM Siggraph 2001, Course Notes*, 2001.

[107] Y. Lipman, D. Cohen-Or, G. Ran, and D. Levin. Volume and shape preservation via moving frame manipulation. *ACM Transaction on Graphics*, 26(1):5, 2007.

[108] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 165–172, New York, NY, USA, 2000.

[109] P. J. Sloan, C. F. Rose III, and M. F. Cohen. Shape by example. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 135–143, New York, NY, USA, 2001.

[110] Ken Perlin and Athomas Goldberg. Improv: A system for scripting interactive actors in virtual worlds. In *SIGGRAPH '96: Proceedings of the 23th annual*

conference on Computer graphics and interactive techniques, pages 205–216, 1996.

[111] Bobby Bodenheimer, Anna V. Shleyfman, and Jessica K. Hodgins. The effects of noise on the perception of animated human running. In *Computer Animation and Simulation '99*, pages 53–63, 1999.

[112] Munetoshi Unuma, Ken ichi Anjyo, and Ryozo Takeuchi. Fourier principles for emotion-based human figure animation. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 91–96, 1995.

[113] Armin Bruderlin and Lance Williams. Motion signal processing. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 97–104, 1995.

[114] Katherine Pullen and Christoph Bregler. Animating by multi-level sampling. In *Computer Animation*, pages 36–42, 2000.

[115] Jehee Lee and Sung Yong Shin. A coordinate-invariant approach to multiresolution motion analysis. *Graphical Models*, 63(2):87–105, 2001.

[116] Radek Grzeszczuk and Demetri Terzopoulos. Automated learning of muscle-actuated locomotion through control abstraction. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 63–70, 1995.

[117] Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey E. Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 9–20, 1998.

[118] Maja J. Mataric. Getting humanoids to move and imitate. *IEEE Intelligent Systems*, 15(4):18–24, 2000.

[119] Zoran Popović and Andrew P. Witkin. Physically based motion transformation. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 11–20, 1999.

[120] Harold C. Sun and Dimitris N. Metaxas. Automating gait generation. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 261–270, New York, NY, USA, 2001. ACM Press.

[121] Arno Schödl, Richard Szeliski, David Salesin, and Irfan A. Essa. Video textures. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 489–498, 2000.

[122] Mirko Sattler, Ralf Sarlette, and Reinhard Klein. Probabilistic motion sequence generation. In *CGI '04: Proceedings of the Computer Graphics International (CGI'04)*, pages 514–517, Washington, DC, USA, 2004. IEEE Computer Society.

[123] Matthew Brand and Aaron Hertzmann. Style machines. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 183–192, 2000.

[124] Pascal Glardon, Ronan Boulic, and Daniel Thalmann. PCA-based walking engine using motion capture data. In *CGI '04: Proceedings of the Computer Graphics International (CGI'04)*, pages 292–298, Washington, DC, USA, 2004. IEEE Computer Society.

[125] Eugene Hsu, Sommer Gentry, and Jovan Popović. Example-based control of human motion. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 69–77, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[126] Hyun Joon Shin and Hyun Seok Oh. Fat graphs: constructing an interactive character with continuous controls. In *SCA '06: Proceedings of the 2006 ACM*

SIGGRAPH/Eurographics symposium on Computer animation, pages 291–298, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[127] Ik Soo Lim and D. Thalmann. Construction of animation models out of captured data. In *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*, volume 1, pages 829 – 832 vol.1, 2002.

[128] S. Li, M. Okuda, and S. Takahashi. Improved kinematics based motion compression for human figure animation. In *Communications, Circuits and Systems, 2005. Proceedings. 2005 International Conference on*, volume 2, pages 2 vol. (xviii+1411), may 2005.

[129] Cihan Halit and Tolga Capin. Multiscale motion saliency for keyframe extraction from motion capture sequences. *Computer Animation and Virtual Worlds*, 22(1):3–14, 2011.

[130] Shingo Uchihashi, Jonathan Foote, Andreas Girgensohn, and John S. Boreczky. Video manga: generating semantically meaningful video summaries. In *7th ACM International Conference on Multimedia*, pages 383–392, 1999.

[131] Andreas Girgensohn and John Boreczky. Time-constrained keyframe selection technique. *Multimedia Tools Appl.*, 11:347–358, August 2000.

[132] Feng Liu, Yueting Zhuang, Fei Wu, and Yunhe Pan. 3d motion retrieval with motion index tree. *Comput. Vis. Image Underst.*, 92:265–284, November 2003.

[133] Min Je Park and Sung Yong Shin. Example-based motion cloning: Research articles. *Comput. Animat. Virtual Worlds*, 15:245–257, July 2004.

[134] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, second edition, 2009.

[135] L. K. Saul and S. T. Roweis. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323– 2326, 2000.

[136] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[137] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[138] S. S. Schiffman, M. L. Reynolds, and F. W. Young. *Introduction to Multidimensional Scaling*. Academic Press, New York.

[139] T. Poggio and F. Girosi. Network for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, September 1990.

[140] Martin D. Buhmann and M. D. Buhmann. *Radial Basis Functions*. Cambridge University Press, New York, NY, USA, 2003.

[141] M. Samozino, M. Alexa, P. Alliez, and M. Yvinec. Reconstruction with voronoi centered radial basis functions. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 51–60, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[142] Daniel Vlasic, Matthew Brand, Hanspeter Pfister, and Jovan Popović. Face transfer with multilinear models. *ACM Transactions on Graphics*, 24(3):426–433, 2005. Proceedings of SIGGRAPH 2005.

[143] M. Alex and O. Vasilescu. Human motion signatures: Analysis, synthesis, recognition. In *ICPR '02: Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 3*, page 30456, Washington, DC, USA, 2002. IEEE Computer Society.

[144] Joshua B. Tenenbaum and William T. Freeman. Separating style and content. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 662. The MIT Press, 1997.

[145] In Jae Myung. Tutorial on maximum likelihood estimation. *J. Math. Psychol.*, 47(1):90–100, February 2003.

[146] Rick Parent. *Computer Animation: Algorithms and Techniques*. Morgan Kaufmann, second edition, 2007.

[147] Chao Jin, Thomas Fevens, Shuo Li, and Sudhir Mudur. Motion learning-Based framework for unarticulated shape animation. *Visual Computer*, 23:735–761, 2007.

[148] Marc Alexa and Wolfgang Müller. Representing animations by principal components. *Computer Graphics Forum*, 19(3):411–418, August 2000. ISSN 1067-7055.

[149] Weiwei Xu, Kun Zhou, Yizhou Yu, Qifeng Tan, Qunsheng Peng, and Baining Guo. Gradient domain editing of deforming mesh sequences. *ACM Transaction of Graphics*, 26(3):84, 2007. ACM SIGGRAPH 2007.

[150] Scott Kircher and Michael Garland. Free-form motion processing. *ACM Transsaction on Graphics*, 27(2):1–13, 2008.

[151] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(11):1254 –1259, nov 1998.

[152] Chang Ha Lee, Amitabh Varshney, and David W. Jacobs. Mesh saliency. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 659–666, New York, NY, USA, 2005. ACM.

[153] Damien Hinsinger, Fabrice Neyret, and Marie-Paule Cani. Interactive animation of ocean waves. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '02, pages 161–166, New York, NY, USA, 2002. ACM.

[154] Big Buck Bunny Animation group. Blender foundation, http://www.blender.org, 2011.

Appendix A

List of publications

- Chao Jin, Thomas Fevens, and Sudhir Mudur, "Optimized Keyframe Extraction for $3D$ Character Animations", accepted by *Journal of Computer Animation and Virtual Worlds*, published by Wiley, 2012.

- Chao Jin, Thomas Fevens, and Sudhir Mudur,"Generation of Variations in Repetitive Motion using Bilinear Factorization", in Proceeding of 4^{th} *International North American Conference on Intelligent Games and Simulation*, page 91-99, McGill University, Montreal, Canada, August 13-15, 2008.

- Chao Jin, Thomas Fevens, Shuo Li and Sudhir Mudur, "Motion Learning-Based Framework for Unarticulated Shape Animation", in the journal of *The Visual Computer* 23(9-11): 753-761, published by Springer, 2007.

- Chao Jin, Thomas Fevens, Shuo Li and Sudhir Mudur, "Feature Preserving Volumetric Data Simplification for Application in Medical Imaging", in Proceeding of *the 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2005*, page 235-242, Plzen-Bory, Czech Republic, January, 2005.

Appendix B

Supporting Videos

Generation of In-betweens Using Characteristic Motion Representation

http://dl.dropbox.com/u/8057957/InbetweenGeneration.wmv

Generation of Variations in Repetitive Motion by Using Bilinear Factorization

http://dl.dropbox.com/u/8057957/variations.wmv