A Generative Statistical Model for Human Motion Synthesis

Master Thesis

Submitted by: M:Sc Cand. Han Du

Program of Study: Computer Science

Supervisors:

Prof. Dr. Dietrich Klakow

Prof. Dr. -Ing. Philipp Slusallek

Advisor:

Dr. -Ing. Martin Manns

Saarland University, 31. May. 2014

Statement

I hereby confirm that this thesis is my own work and that I have documented all sources used.

Saarbrücken, 31.05.2014

Acknowledgements

There are many people who help me throughout my thesis. Here, I want to give them my sincerest gratitude.

First of all, I would like to thank Prof. Klakow for telling me the opportunity to do my thesis in Daimler and introducing me into the field of statistical learning from his lecture, which is very useful in my thesis. His lectures have always been delightful and encouraged me to continue my study along this path.

Another person who deserves my special acknowledge is my advisor, Dr. Martin Manns. He offered me the chance to work on this interesting project when my former advisor Inga left unexpectedly. His guidance and suggestions throughout my work were invaluable, and without his support, I cannot get much progress in this challenging topic. Again, I want to thank him and other colleges from the team Integrierte Produktionsabsicherung Produktionsplanung, Daimler AG. Their kindness and sincere help make me have a very nice time in Daimler, which will be a memorable experience for me.

I am also grateful to the support from DFKI. Prof. Slusallek is my second supervisor for my thesis, and Dr. Klaus Fischer offered me the opportunity to continue my work in DFKI. Thank you very much for their supports and valuable advices.

Then I want to thank two of my colleagues in this project, Erik Herrmann from DFKI and Markus Mauer from Daimler. The GUI framework designed by Erik is very useful for me to visualize and analyze my work. And Markus helped me verify my approaches and gave me some remarkable suggestions. It has been my pleasure to work with them.

Furthermore, I would like to thank Dr. Nils Hasler from MPI Informatik. He helped me set up motion capture system and process captured data. And Dr. Kosmas Alexopoulos and his group provides us lots of help in constrained motion synthesis. They deserve recognition for support us.

Finally, I owe deep gratitude to my parents and my friends. My studies would not have been possible without their incomparable love and exceptional support.

Abstract

Today's highly competitive global market requires automotive manufactories to provide high quality products efficiently. However, it contradicts the traditional physical prototype based production verification, because setting up such a physical prototype is costly and time-consuming. The current trend for solving this contradiction is to apply digital automation tools for the tasks of design, verification, validation and modification in automotive industry. An advanced automation tool should be able to integrate the skilled workers' knowledge on executing manual assembly tasks in it. For this purpose, a powerful human motion analysis and synthesis model is required, which is the motivation of our work..

It is a very challenging problem to construct a generative and scalable human motion analysis and synthesis model for assembly actions, because assembly actions in the workshop are usually quite complex and constrained. The goal of this work is to reengineer and evaluate a novel human motion synthesis method: Motion Graphs++, since the traditional human motion modeling and analysis methods, such as manual modeling and analytic motion synthesis cannot meet this ambition. In this work, a statistical model for human motion synthesis and analysis has been explored. Although it is not novel to apply statistical analysis for modeling human motion, there is no practical usage in industry to the best of our knowledge. The performance of this approach is evaluated on our motion capture data, and compared with high quality Vicon motion capture data. Our test scenario is walking in a free space. The experiment results demonstrate that motion graphs++ approach does not work well with noisy motion capture data. So some modifications have been made to adapt the original algorithm to our data, and experiment results show it improves synthesized results for our case.

In the future steps, users specified and environmental constraints will be imbedded in this work to synthesize realistic motions with certain targets. And more kinds of motion will be test by our statistical model.

Key words: *human motion synthesis, motion graphs++, statistical models*

Contents

List of Figures

List of Tables

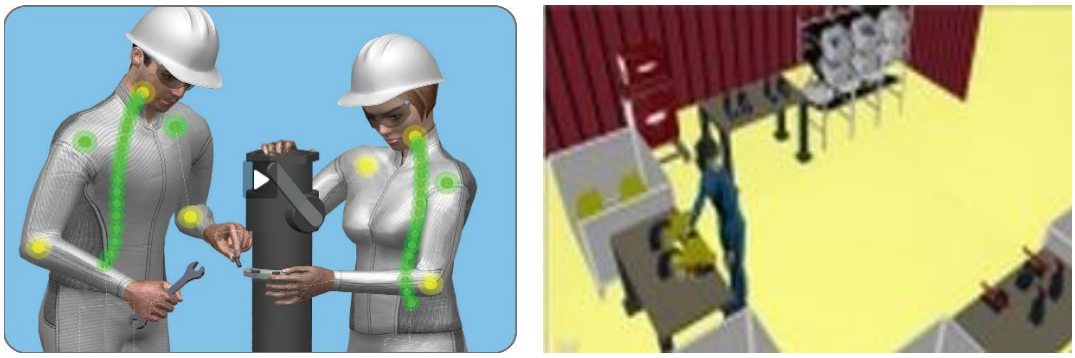
1 Introduction

1.1 Initial Situation

Today's fierce competition in global scale requires automotive manufacturers to enrich the variety of products and reduce the cost, meanwhile maintaining the high quality and efficiency [1]. However, this requirement contradicts the traditional way of production analysis and verification. In automotive industry, the most widely used method to design and verify a new product and its corresponding assembly system is based on a physical prototype currently [3]. A physical prototype is an early release of a product built to test concepts or processes of the product [4]. It defines the requirements for the whole assembly tasks, for instance, how to determine the workers' walking path, how to place workbench, which components will be used and the time analysis and ergonomic evaluation and so on. However, giving clear answers to these questions are not trivial tasks, due to it is very hard to consider all the possible cases on a very early stage of development. Therefore, a successful design of physical prototype usually undergoes several design cycles and variations until a final version is reached [5]. This procedure is very time-consuming and costly. And the cost for modification of the prototype increase when the product development cycle nears its end [6]. In addition, there is no guarantees for future extension of later products in all respects, because it is nearly impossible to simulate all possible situations by physical simulation at the very early stage [2]. Another problem is that each prototype is specific for one product, which is extremely unsuitable for today’s requirement of large variety of products. So in order to meet the requirements from customer driven market, novel ways to design and verify products and their corresponding assembly systems are highly demanded.

The most popular approach to replace physical prototype in automotive industry currently is to use virtual prototyping (VP) [5, 7, 8]. Virtual prototyping employs digital design and simulation tools for production analysis and verification. CAE/CAD tools, such as AutoCAD, CATIA, are the early virtual design prototyping. Their application reduces the design cycle time and improve quality standards [9]. However, in the early version virtual prototyping, the modeling of human activities is not included. Since today’s procedure of assembling automobiles still requires many manual works, a good digital simulation and verification tool should also be able to integrate the skilled workers' knowledge on executing manual assembly task [10]. In order to achieve this purpose, a powerful human motion synthesis model is needed to be imbedded into the early virtual prototyping. Several different approaches have already been proposed for this target.

One of the earliest methods is manual geometric modeling [11]. The user needs to explicitly tune the parameters of human skeleton within physical-valid range to generate the simulation. As human body enjoy a rich degree of freedom (DOF), the effort for modeling an accurate human motion is very high. For instance, one minute simulation could require several hours



(a) (b)

Figure 1. 1: Examples of digital manufacturing tools. (a) Delmia V5 (Dassault, http://www.3ds.com/products-services/delmia/products/ergonomics-specialists/ergonomics-analysis/), (b) EMA (IMK-automotive GmbH, http://www.imk-automotive.de/)

modeling [12]. And the quality of synthesized motion depends on the number of keyframes. The more keyframes drawn by designer, the better quality it has, and the higher modeling time it needs. A practical implementation of this approach is Delmia V5, a CATIA plugin tool from Daimler AG (cf. figure 1.1 (a)).

In order to reduce this high modeling effort, one possible solution is automatic motion synthesis, rather than manual modeling. It regards the motion parameters of each DOF as a single time varying signal [13]. An analytical function is defined for each joint, which will control the change of parameters of each joint. Motion captured data can also be used to train the parameters of analytical function in order to get natural looking result. One example of this approach is a simulation software from IMK-automotive GmbH called EMA (Editor Human Labor), see figure 1.1 (b). Basically, EMA can provide a virtual modeling and simulation environment for assembly task, and saving a lot of efforts and time compared with Delmia V5. However, the evaluation from designers from automotive industry shows that result quality is not sufficient for practical application. The actions from virtual workers are sometimes not collision awareness, for example, the hand of virtual worker could go through objects. In addition, the generated motion looks unnatural and even unrealistic sometimes.

1.2 Problem Statement

Because of the deficiencies of manual modeling and analytic motion synthesis, nowadays people are more interested in motion capture data driven motion synthesis, benefited from the fast development of motion capture techniques. Now, motion capture is a reliable way to get realistic human motion. However, the gap between human motion modeling in automotive industry and motion capture is directly using motion capture data to generate new motion is a nontrivial task. Because motion capture data is proven to be hard to modify [29]. The main problems of using motion capture data for animation are [28]:

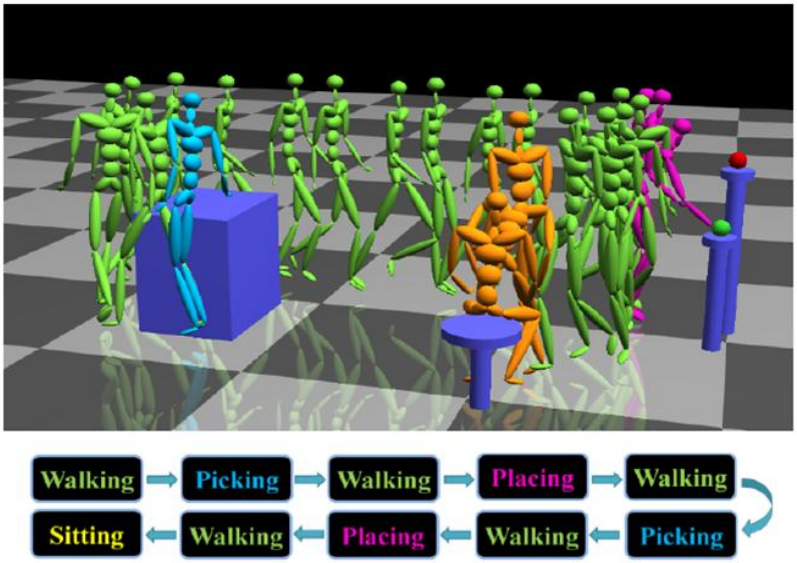


Figure 1. 2: Semantic motion analysis and synthesis. [15]

- Most motion capture systems are expensive to use, because for each specific task, we need to capture motions which satisfy the requirement. The processing time for motion data is usually high and motion data is not easy to be re-used.
- Some kinds of motion, for instance, special effects in movie and game industry, are very hard to perform by actors.

So algorithms, which can edit, re-use motion capture data to create continuous streams of motion are needed. Motion Graphs [14] takes the idea that decomposes the captured motion data into small parts, then creates a new motion by concatenating these small pieces with some proper methods. Motion graph transforms the motion synthesis problem into a selecting problem, search on the motion capture database. If the captured motion data is large enough, it theoretically can generate arbitrary motions. However, if the size of prerecorded data is limited, motion graphs will not have a rich variation of styles. In order to enjoy a rich repertoire of poses, which is more natural and realistic, we need a method which can generate infinite motion samples from finite prerecorded motion data. Motion Graphs++ [15] states that they can synthesize continuous style variant motions based on the assumption that the high dimensional human motion data has some intrinsic coordination which can be represented in a more compact way and follow some statistical distribution. motion capture data is firstly decomposed into motion segments and classified into pre-defined motion primitives. Then a statistical model is build to model the distribution of these recorded motion samples. So infinite motion samples can be generated just by sampling this statistical model. Figure 1.2 illustrates this idea. Each color is corresponding to one motion primitive, and two motion segments from the same primitive (green part) are different. They also assume that the transition between two motion primitives is followed some statistical distribution, so the whole sequence of synthesized motion looks natural.

1.3 Objective

The task of our work is to reengineer Motion Graph++ algorithm for free space walking scenario, and evaluate the performance of this algorithm on both noisy markerless motion capture data from our motion capture system and a high quality motion capture data from Vicon system. The comparison results could be used as a reference for deciding which quality of motion capture data is sufficient for this approach to build a generative, scalable, automatic and semantically interactive human motion synthesis model, which is capable of generating natural-looking motions. In addition, some modification of original motion graphs++ algorithm have been made to make it work for our motion capture data. Our motion capture data is stored as BVH file, which has a hierarchical structure in angular space. The experimental results will be evaluated by correctness and naturalness of synthesized motion from the human motion synthesis model.

This work is a preliminary step for the final goal, which is to construct an automated, scalable, semantically interactive and natural-looking motion synthesis model. The work does not include a complete motion graph for different kinds of motion. And embedding users' specified constraints and environmental constraints into the statistical model is also beyond the scope of this work, due to the limit of time.

1.4 Structure of Work

The remainder of this work is organized as follow:

- Chapter 2: introduces the background of our work. It firstly presents a overview of the state of research in human motion synthesis field. Then an introduction of motion capture system and the markerless motion capture system used in our work is followed. The last part introduces the data format used to store motion capture data in our work.
- Chapter 3: gives an introduction of the procedure of our approach for human motion synthesis, which is based on motion graphs++.
- Chapter 4: presents some experimental results and compares the performance of our approach on low quality motion capture data and high quality motion capture data.
- Chapter 5: summaries the thesis work and discusses the potential future work.

2 Background

Our work is based on the long history of human motion synthesis study and currently fast development of motion capture techniques. In this chapter, a brief overview of the state of research in human motion synthesis is presented in first part. Then, an introduction of different kinds of motion capture system is given in the following part. It also includes the markerless motion capture system in our work. In the last part, a description of BVH file is shown and some mathematic background of forward kinematic will be discussed.

2.1 Human Motion Synthesis

The study of human motion synthesis has a long history in movie and video game industry. This section will present an overview of most popular used motion synthesis methods in past two decades. In general, existing motion synthesis methods can be divided into three categories: manual synthesis, parametric motion synthesis and motion capture data driven motion synthesis.

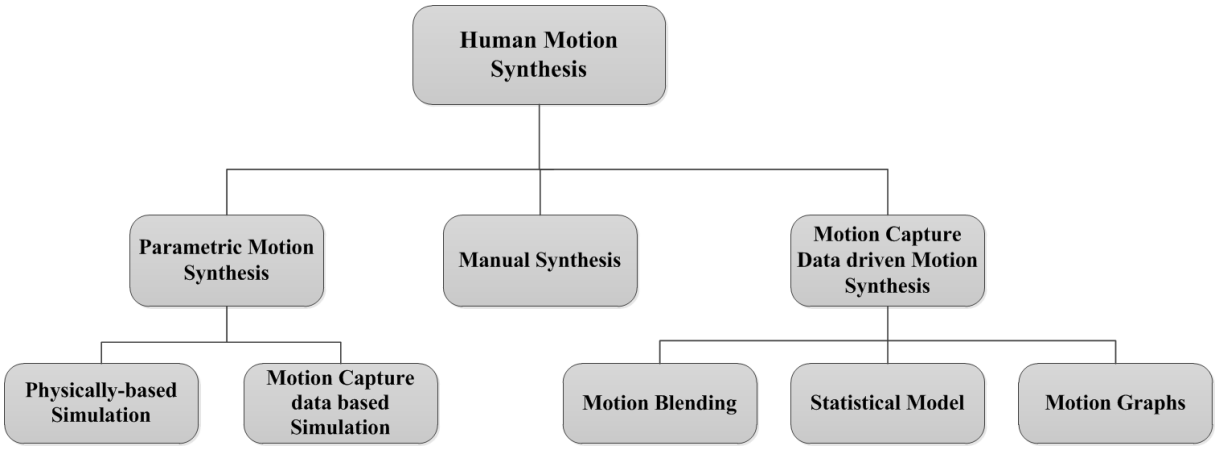


Figure 2. 1: Classification of human motion synthesis methods

2.1.1 Manual Synthesis

Manual synthesis is the oldest motion synthesis approach, which is widely used in early cartoon movie and game industries. It manually specifies the parameters of degrees of freedom (DOFs) for a pre-defined human skeleton, which are called as key frames. The frames between these manually drawn key frames are generated through some interpolation methods, for instance, linear interpolation. However, simple interpolation rarely gives realistic results unless the key frames are really dense. This work is tedious and time-consuming, not only because a lot of key frames must be created, but also because it is challenging to draw good key frames which are realistic and smooth when played in sequence. Sometimes it requires some expertise of art. A practical example of this approach is Delmia V5. As we discussed before, the result is not very natural and the simulation requires huge manually modeling work.

2.1.2 Parametric Motion Synthesis

Generating motion simply by interpolating usually give us unrealistic results, because the real and natural human movement is governed by many constraints, e.g. the laws of physics, custom of common behaviors and so on. Parametric motion synthesis embeds these constraints into controllers, which can control the whole skeleton or each joint. The control systems take input from high-level specifications, such as users' constraints or environmental constraints, then automatically generate the lower-level details of the motion. In manual synthesis, the animator must specify many of the lower-level details by hand. In order to achieve this, each controller uses one or several analytical functions to control the parameters of degrees of freedom (DOFs). The analytical functions can either set by physically valid constraints or learned from motion capture data.

Physically-based Simulation

The approach to use physical laws to control the movement of body is widely used in robots. Given the mass distribution for each body part and the torques generated by each joint, Newton's laws provide a system of ordinary differential equations that can be integrated to yield joint trajectories. However, while the mass distributions can be obtained from biomechanics literature [16], it is proven considerably hard to find proper joint torques that yield a particular motion. Hodgins et al. [17] proposed handcrafting joint controllers based on finite state machines and they demonstrated this approach on several motions, including running, bicycling and vaulting. Similar approaches [18, 19] were used to generate gymnastic motions like flipping or tumbling. Although this method has successfully created certain kinds of motion, a general way to design and select controller for different kinds of motion is still a difficult problem. Each joint in human skeleton is corresponding to one controller, and each controller could have different control strategies for different actions. How to automatically change control strategies when actions change is a nontrivial task. Faloutsos et al. [20] applied support vector machines to learn the conditions under which controllers for different actions can be composed, then switching control strategies to transition between actions. However, physically based simulation can automatically generate physically realistic motions, a motion that obeys physical laws does not mean it is natural.

Data-based Simulation

Rather than explicitly representing physical laws as equations to each controller, some other work employed motion capture data to train the parameters of each analytical function. The idea is that as the naturalness and physically valid are implicitly represented in real human motion, the analytical functions which can model the motion capture data well should able to generate realistic human motion while preserving naturalness meanwhile. A real example using this approach is the digital simulation and planning tool EMA. However, data-based simulation can only provide a relatively narrow range of realistic motion.

Comparing with manual synthesis, parametric motion synthesis can save a lot of manual modeling work. It also enables creation of autonomous actors that can react to user or the environment using higher-level control systems. However, despite the potential advantages of

parametric motion synthesis, robust control systems which can generate physically-valid and natural-looking motion are difficult to design by hand. And automatic methods are not yet sufficient enough to generate control systems for most human behaviors.

2.1.3 Motion Capture Data driven Motion Synthesis

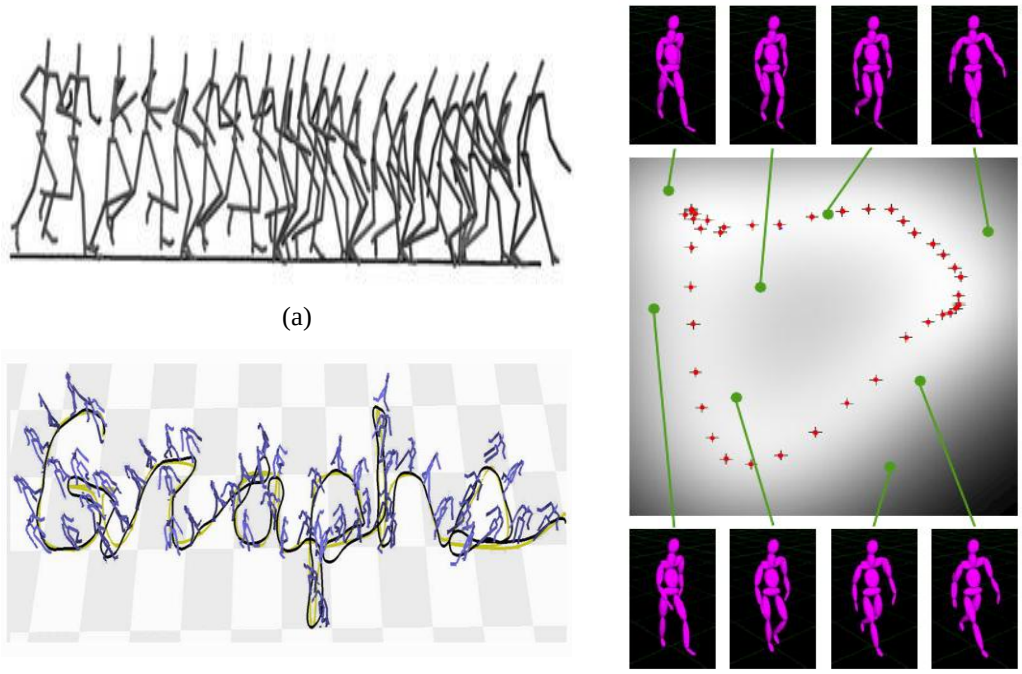
With the development of motion capture techniques, Motion capture data driven motion synthesis has caused more and more attention over the past two decades. Motion capture technology can provide a source of highly realistic example motions, which can serve as raw material for a variety of data driven synthesis methods. Compared with parametric motion synthesis, data driven approaches have the advantages of natural-looking, realistic and more generative. There is no need to build complex control systems for each joint. Many algorithms have been proposed to synthesize new motion based on motion capture data. In this section, we will go through the state of arts of data driven motion synthesis algorithms.

Motion Blending

The simplest way to create a new motion from pre-recorded motion is to just concatenate two motion and blend the transition part. Figure 2.2 (a) shows a combination of running and walking. However, simply putting one motion after another could cause two problems. One is the motion discontinuity at transition point, another is two motion could be in different speed scales, for instance, combining a slow running and a fast walking. For the first problem, it is usually addressed by motion blending [21], which can create seamless transitions between motions. The choice of interpolate function depends on the motion capture data. For instance, if the motion capture data is 3D coordinates of joints, then linear interpolation [22] can generate good transition. If the motion capture data is represented by relative orientation angles, like BVH file, radial basis functions [19] can be used to blend two motion examples. For the second problem, motions with different timing may be timewarped such that logically related events occur simultaneously. This mapping between corresponding frame index of motions indices as a timewarping curve. we can use dynamic time warping algorithm to pre-processing motion data, registering one motion to another [25]. However, for synthesizing some special motions, sometimes, it is not easy to find suitable motion capture. It is always necessary to capture new data for this purpose. So just simply blending and interpolation cannot make use of motion capture data efficiently.

Motion Graphs

Motion graphs (sometimes called "move trees") has been used in the video game industry for a long time for the purpose of character control [24]. Most motions more than a few seconds in duration are naturally thought of as sequences of atomic actions. For example, walking consists of an alternating sequence of left stance and right stance. Motion graphs will generate been constructed manually in the sense that users explicitly decided which motion clips would be connected [26, 27]. Given a motion capture data set, an animator identifies places where motions can connect together, then uses tools to adjust these motions so they will join seamlessly at these points, a tedious and time-consuming process.



(a) (b) (c)

Figure 2. 2: Examples of different human motion synthesis approaches. (a) motion blending between running and walking [25], (b) a walking motion generated to fit a specified path by motion graph [14], (c) random pose synthesis from statistical model [35].

Automated graph-based motion synthesis has been the subject of a considerable amount of previous work. The crucial point for automatically constructing motion graphs is to automatically identified transition locations. Kovar et al. [13], Lee et al. [29] and Arikan and Forsyth [28] all used a distance metric to find transition locations and then applied search algorithms on the resulting graph to extract motions that satisfied user-defined constraints. After selecting appropriate sequence of motions, to generate transitions is also an important part of building motion graphs. Perlin [30], Kovar et al [14] used a simple linear blending method to concatenate two motion clips. There are also many other transition methods, such as displacement maps [28, 29, 31], the torque-minimization algorithm of Rose et al. [32] or the blending method introduced in previous section. Figure 2.2 (b) shows an example of walking motion followed a user specified path generated by motion graphs.

Motion graphs can not only concatenate two motion samples as motion blending, but generate continuous streams of motion under user specified constrains as well. This approach is more flexible and can make use of motion capture data more efficient than simply blending. It is also possible to combine environment constrains or user specified constraints, for instance, sneak walking in this approach. As motion graphs transfers motion synthesis problem into a directed graph search problem, so all the requirement can be formulated as constraints in an optimization problem.

However, there are some limits of motion graphs. First of all, the synthesized motion is from database of motion capture. So it cannot have a rich variation of pose. Secondly, at the

transition point, the thresholds for similarity must be specified by hand, because different kinds of motion have different fidelity requirements. For the motions which are very familiar to people, such as walking, running and so on, people are very sensitive to discontinuity. So a high similarity threshold is required. But for some motions are less familiar for people, maybe a low similarity is enough.

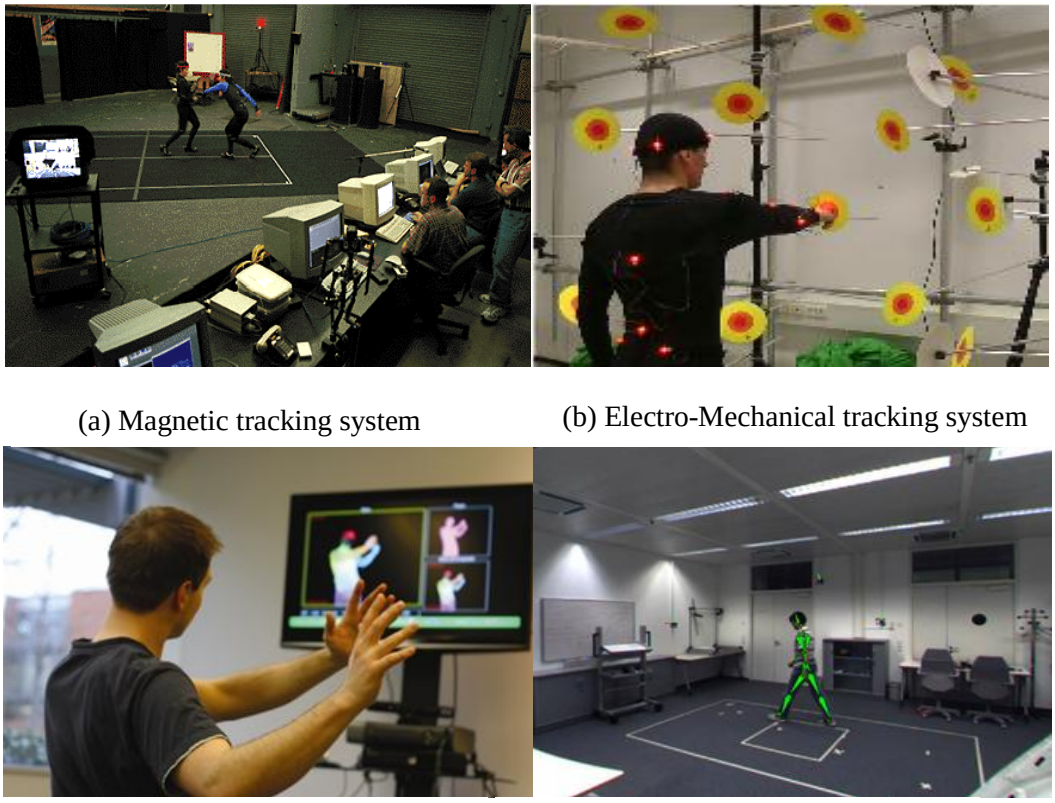
Statistical Model

The method is inspired by above mentioned problem, how to generate a variant of different motions from limited number of captured motions. The assumption is although human motion looks quite arbitrary and could have infinite styles, different poses of one motion should have some intrinsic relationship, which can be model by some statistical models. The models are usually represented as a set of mathematical equations or functions that describe human motion using a finite number of parameters and their associated probability distribution. So the task of motion synthesis is converted to construct a proper probability distribution model, which can well fit the finite motion capture data. A new variant motion can be generated by sampling the distribution. Theoretically, this approach can generate any possible poses of motion, but usually the motions which are most similar to the training data are taken. A lot of researches have been invested in this field. Grochow et al. [36] built an inverse kinematics system based on a learned model of different human poses. They stated that this system can generate any pose which satisfies the given constraints. Bowden [23], Brand and Hertzmann [33], Galata et al. [34] and Li et al. [35] combined motion graphs with statistical models. They have built graph-based statistical models. Each node in the graph is a simple generator of poses which is corresponding to a statistical model, and each edge represents possible transitions between different kinds of poses. Figure 2.2 (c) is an example of using statistical model to create different pose of walking. However, a fundamental limitation of statistical model approach is that it lacks the ability to control motion in a very precise way, and does not consider physical force that causes the motion.

2.2 Motion Capture

The main work of this thesis is to apply one of state of the art data driven motion synthesis algorithms [15] to our own motion capture system and the recorded motion data. In this section,. a brief overview of motion capture system will be given and a detail introduction for our motion capture system and data format for capture motion will be given.

Motion capture has been proven to be a successful technique for creating realistic animations. It is also the foundation for motion data driven motion synthesis approaches. A significant amount of research has been devoted to this area, and many practical motion capture systems have been used, especially in film and video game industry. A motion capture system records the pose of a human body at each time frame. Normally, the tracking system will not track the whole visual appearance of the actor, but the movements of some important points or joints in the human skeleton. During the tracking sessions, movements of each point will be sampled many times per second. These tracking data will be used for animation by mapping to a 3D human model so that the model could replay the same movement on the screen. Motion



(a) Magnetic tracking system (b) Electro-Mechanical tracking system

(c) Depth camera tracking system (d) Color camera tracking system

Figure 2. 3: Examples of different motion capture systems.

capture supports a wide range of applications, including animation for computer games, movie industry, virtual reality and human-computer interaction and so on.

There are several ways to classify motion capture systems, in terms of different algorithms or devices they use, for example, one camera system or multi-camera system. Here, we divide motion capture system based on the tracking sensors. Figure 2.3 shows some examples of different types of motion capture system.

2.2.1 Magnetic Sensor Based Systems

One of the earliest practical used motion capture system. They use electromagnetic sensors to capture the movement of human body. These sensors are connected to one or several computers which can process the data and produce 3D data in real time, so that the tracking trajectory will be display in the screen at the same time. The result is accurate and fast, without complex post-processing time. So this kind of system is widely used in game and movie industry. However, the major drawback of this system is restricted freedom of movement caused by the connected cables. There are also some other problems like the system is very expensive usually and it can only track the movement of sensors, which means the background environment of actors need to be constructed additionally.

2.2.2 Electro-Mechanical Sensor Based System

Due to the deficiencies of magnetic system, Electro-Mechanical systems have recently appeared in the market, which try to maintain the advantages of magnetic system, such as

high tracking quality and real time processing. Meanwhile, it reduces the cost and constrains of movement because of cabling. A very common example of this kind system is the electro-mechanical sensor suit. In the recording sessions, the actor has to wear this special suit with integrated electro-mechanical sensors that register the motion of the different articulations.

The drawbacks of this system are that it still cannot track the background of the actor, and the tracking skeleton is usually fixed.

2.2.3 Depth Camera System

Depth camera just appeared several years ago, and it becomes very popular in recent years, e.g. Kinect camera. The depth information is a good feature for motion tracking, and it can also track the background of actor, which means it has good interaction property. And depth cameras such as Kinect are cheaper than these special sensors. In addition, it also has the advantages for instance, full freedom of movement and the interaction between multi-actor. However, currently, the raw depth data from Kinect is quite noisy. So in order to get accurate motion tracking result, some good algorithms are needed.

2.2.4 Optical Color Camera System

Based on photogrammetric methods, optical systems have been applied to motion capture area for a long history. As depth camera system, traditional optical camera systems offer the advantage of complete freedom of movement and interaction with different actors. And this kind of system is usually very flexible. It can be extended by adding more cameras, which could provide larger tracking area and better tracking results. For optical camera tracking system, they can be divided into marker based and markerless. No matter using marker or not, the captured motion data is basically 3D position and relative orientation of each marker or joint in human skeleton. The major disadvantage of optical systems is that post-processing is required to extract captured motion data from record videos. So the tracking accuracy and processing time highly depends on motion capture algorithm which they used.

	Accuracy	Processing time	Cost	Tracking area	Interaction	Freedom of movement
The magnetic systems	+	++	-	- -	++	-
Electro-Mechanical senses	+	+	+	- -	++	-
Depth camera systems	-	-	++	+	+	+
Optical color camera system	-	-	+	++	+	+

+: good performance - : bad performance

Table 2. 1: Performance comparison of four motion capture system under different criterions.

Table 2.1 compares the performance of above mentioned motion capture systems. In this work, our motion capture data is from optical color camera system. Figure 2.3(d) shows the motion capture system installed in our laboratory. It is a markerless motion capture system named CapturyStudio from Max Planck Institute Informatik. It uses 8 color cameras to track

the joints in pre-defined skeleton, and the feature for tracking is color information of actor's visual appearance. We also have some very high quality motion capture data from Vicon system, which is marker-based optical motion capture system using 24 cameras. We will show the quality of Vicon motion capture data in chapter 4, and regard it as noise free data for our experiments.

2.2.5 Markerless Motion Capture System

The motion capture system used in our work is a markerless motion tracking system based on multiple optical color cameras [37]. The system uses the color of actor's surface as information to track instead of markers. Figure 2.4 illustrates how this markerless motion tracking system works. In our laboratory, eight fixed color cameras are installed. They are taking videos at the same time when the actor is acting. Then the user needs to extract actor's silhouette from background and match the pre-defined human skeleton model in the system shown in figure 2.5. By doing that, the system basically knows the color of each joint in human skeleton, and tracks that through all the frames in all videos.

The advantages of this system are: First of all, it is a low cost system, comparing with most afore mentioned systems, and it is easy to set up. Secondly, it is markerless, which means that the actors do not need to much preparation for tracking. They can immediately start to act when the cameras are running. However, for the marker based tracking system, it could take a long time to prepare before tracking, for instance, wearing some special suit, putting markers on the correct position and so on. Thirdly, the system can track a large area. And the more cameras installed, the larger area it can track. This is a good solution for tracking task in a workshop. Usually, we want to track the workers' action interacting with objects in the workshop, rather than the action itself.

However, there are some drawbacks of this system from our practical usage. Firstly, as it uses color information to track, so the capture result is highly influenced by light condition during tracking. If the light condition changes during tracking, or there are color distortion of some cameras, the capture result could probably be bad. Secondly, although this system is markerless, it still has a preference for the color of the suit. If the actor wears a colorful suit, which is easy to distinguish different part of body, then the tracking would be simple and captured results are accurate. However, if the actor wears a suit which has only single color, the system would suffer "cross legs" problem sometimes. Last but not least, although it doesn't require much preprocessing time to take motion videos, it needs considerable post-processing time to extract motion capture data from videos, especially when the quality of the videos are not good. Basically, the user needs to correct tracking errors in the frame stream, which is a tedious work.

2.3 Motion Data Representation

As mentioned above, the data from motion capture system is samples of the position of each joint or marker many times per second. Then the next problem is how to store these motion capture data. They can be stored by the absolute 3D geometric coordinates of joints or relative position and orientation of joints. The way of how to store motion capture will highly

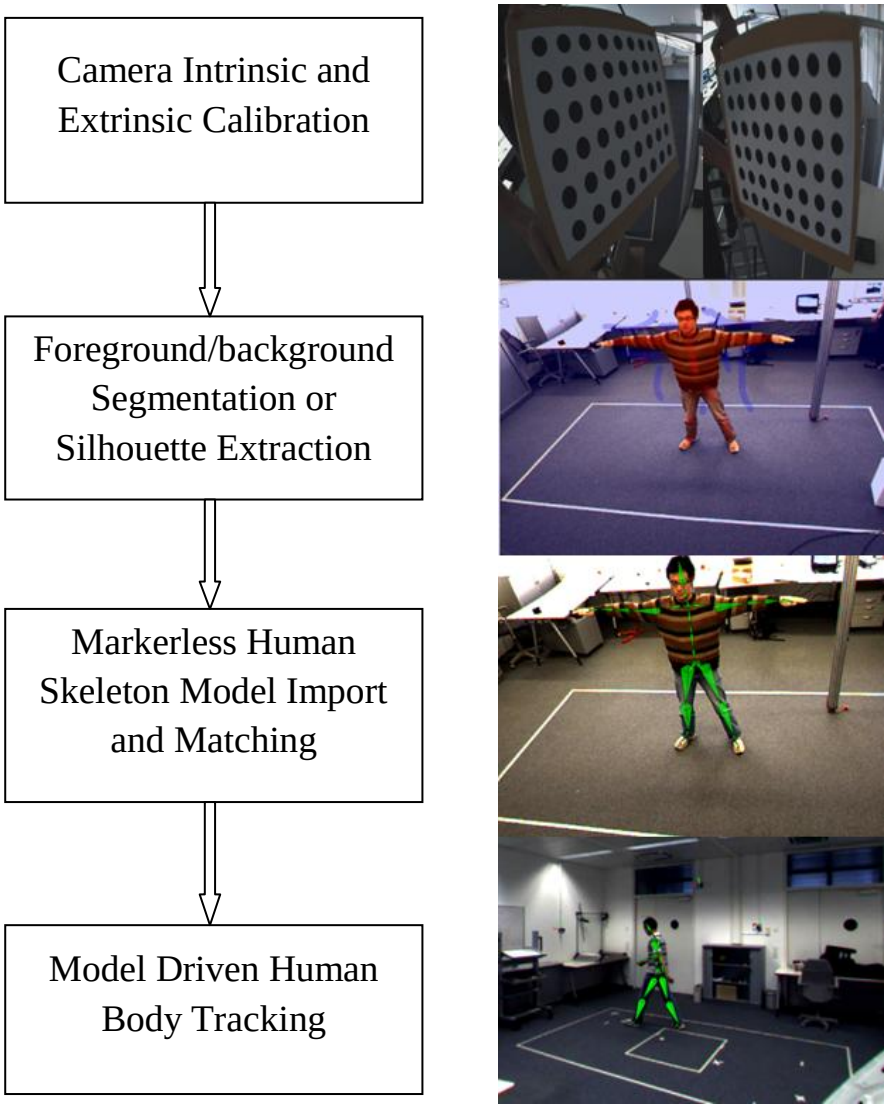


Figure 2. 4: The work-flow of the markerless model based tracking system.

influence the efficiency of our following processing algorithm. This section will firstly provide a short overview of most common used motion capture file format, then give a detail introduction of BioVision (BVH) file format, which we use in our work. Table 3.2 lists most common used file formats for motion capture data today, along with URLs for additional information.

2.3.1 Terminology

The following list outlines some of the more important keywords which could be found in BVH file format.

- Hierarchy: the first part of BVH file, which defines the actor's skeleton in a hierarchical structure. It is allowed to define more than one actors' skeleton in Hierarchy section, and the hierarchical skeleton usually starts from root joint.
- Offset: a 3D vector, which defines the bone length between two joints.

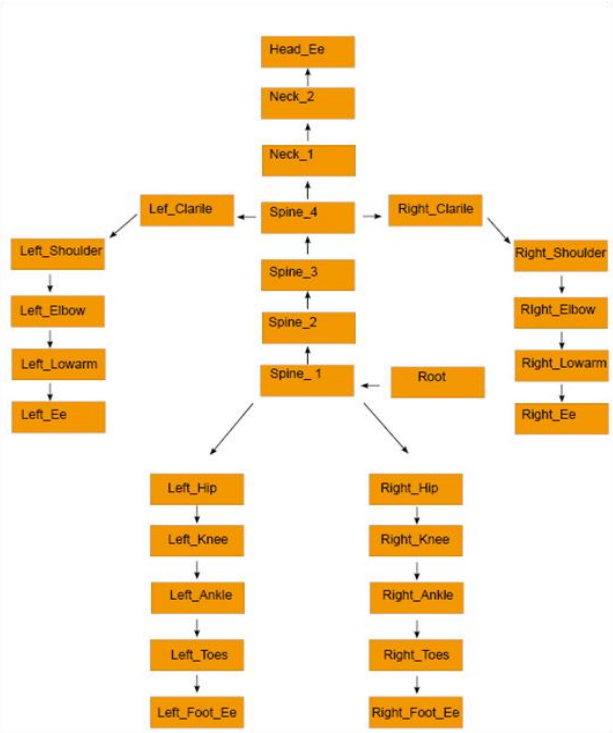


Figure 2. 5: Hierarchical structure of humanoid character of markerless mocap system.

- Channels: defines degree of freedom of a human skeleton. Each bone within a skeleton can be subject to position, orientation and scale change over the course of the animation, where each parameter is referred to as a channel.
- Motion: the second part of BVH file, which contains the motion capture data at each timestamp. Each line of motion represents one frame, and the data is ordered according to sequentially concatenating the channels in hierarchy part.

2.3.2 BVH File Format

The BVH format succeeded BioVision's BVA data format with the noticeable addition of a hierarchical data structure representing the bones of the skeleton. As mentioned above, the BVH file consists of two parts where the first part contains the hierarchy and initial pose of the skeleton and the second part is the motion data section, which describes channel data defined in first part for each frame.

In BVH file, the human skeleton is represented in a tree structure, usually starting from root node. Human skeleton can be reconstructed by deep first parsing this part. Figure 2.6 illustrates the mapping from hierarchical skeleton structure data to visualization of human skeleton. The definition of each bone is in the OFFSET line, which details the translation of the origin of the bone with respect to its parent joint's origin along the x, y and z-axis respectively. The movement of human skeleton is defined by rotation of each bone, which can be found in the line with keyword CHANNELS. The rotation order about x, y, z-axis is defined in HIERARCHY part, and the corresponding value can be found in MOTION part. It is important to note that the rotation is relative rotation about the parent joint's origin, not a

global rotation. By combining the hierarchical human skeleton description and motion data, we can get an animation from BVH file.

File Extension	Associated Company / Description	File Format Reference
ASC	Ascension	NO LINK
ASF & AMC	Acclaim	http://www.darwin3d.com/gamedev/acclaim.zip
ASK & SDL	BioVision/Alias	NO LINK
BVA & BVH	BioVision	http://www.biovision.com/bvh.html
BRD	LambSoft Magnetic Format	http://www.dcs.shef.ac.uk/ ~mikem/fileformats/brd.html
C3D	Biomechanics, Animation and Gait Analysis	http://www.c3d.org/c3d_format.htm
CSM	3D Studio Max, Character Studio	http://www.dcs.shef.ac.uk/ ~mikem/fileformats/csm.html
DAT	Polhemous	NO LINK
GTR, HTR & TRC	Motion Analysis	http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/ {HTR.html, TRC.html}
MOT & SKL	Acclaim-Motion Analysis	(Under Development - http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/SKL-MOT.html)

Table 2. 2: Motion capture file formats and references for additional format information [38].

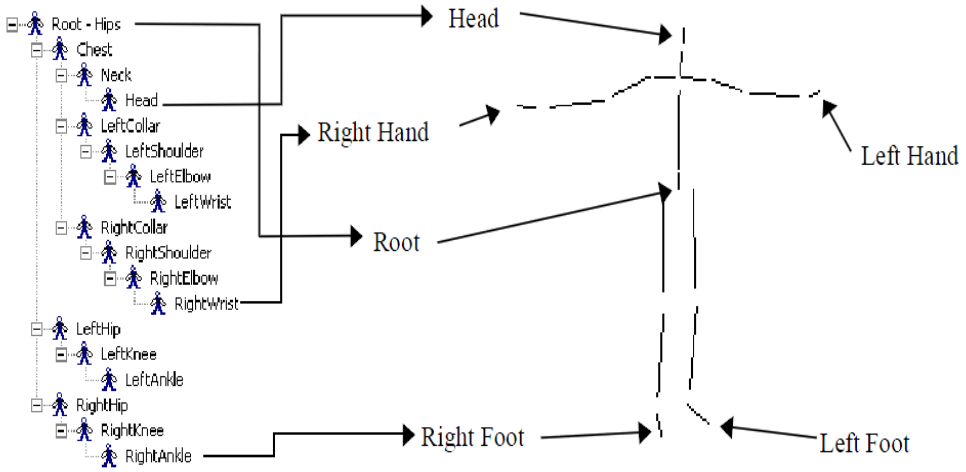


Figure 2. 6: Mapping from hierarchical structure data to a human figure [28].

One advantage of using BVH file for our work is that as mentioned above, each BVH file defines a human skeleton in it, so we don't need to pre-define a human skeleton for different motion capture system, which makes our statistical modeling approach more robust for different human skeleton model.

2.3.3 BVH Data Processing

In this section, some mathematical foundations will be introduced in order to calculate global position of each joint in human skeleton from raw BVH data. Global position of each joint is crucial for our motion synthesis work.

Assume the global position of a joint P_0 is already known, now we want to get the global position of its child joint P_1. The first step is to get local transform from raw BVH data, then apply local transform to global position P_0. Since BVH format does not contain any scaling information, only the rotation and translation need to be considered. In homogeneous coordinate system, rotation and translation can be generatively represented as rotation and translation matrix multiplication with point vector. An example of OFFSET and CHANNELS for one joint in BVH format:

```
OFFSET  0    112 -10
CHANNELS 3  Xrotation Yrotation Zrotation
```

Based on this, we can construct our rotation matrix *R* and translation matrix *T*. And the local transform *M* = *RT*. The order of channels gives the order of composing rotation matrix.

$$R = R_x R_y R_z \tag{2.1}$$

R_x, R_y and R_z are rotation matrix about x, y and z-axis respectively. The definitions are as below:

$$R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.2}$$

$$R_y = \begin{pmatrix} \cos\theta_y & 0 & -\sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.3}$$

$$R_z = \begin{pmatrix} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.4}$$

The translation matrix *T* is defined as:

$$T = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.5}$$

where the value of T_x, T_y and T_z are corresponding to the value in OFFSET respectively. So the global position of joint P_1 is:

$$P_1 = MP_0 = RTP_0 = R_x R_y R_z T P_0 \tag{2.6}$$

The global transform matrix of a give joint can be obtained by multiplying its parents' local transform iteratively. The global transform matrix M^n_{global} of joint n is:

$$M^n_{global} = \prod_{i=0}^{n} M^i_{local} \tag{2.7}$$

where *n* is the current joint, *n-1* is its parent joint. *n = 0* means the root joint of the hierarchy.

3 Methodology

Our work is based on the approach of Motion Graphs++ [15], which is a combination of Motion Graphs and statistical model. And some modifications are made to make this approach better fit our motion capture data. Similar to Motion Graphs [14], Motion Graphs++ represents motion synthesis problem as directed graph search problem. However, the construction of motion graph is different from the approach of Motion Graphs. In Motion Graphs, the edges of the directed graph are motion clips or single frames, and the nodes are actually transitions which are designed frames to seamlessly connect two edges in the graph. However, in Motion Graphs++, a node represents one kind of motion segment that are similar to each other in high-level structure. And edges are possible transition between each node.

As discussed before, one of the main difficulties for human motion synthesis is that human motion seems to have infinite degree of freedom (DoF). Motion Graphs++ makes the assumption that although natural human movement has a wide range of style variations within the same action, each action always consists of finite segments in terms of high-level structures. For instance, walking can be regards as a sequence of alternating double and single-stance. Figure 3.1 shows four different styles of walking. Basically, they can all be represented as a sequential combination of six kinds of segment in different order. From this observation, we can find a more efficient and compact way to represent complex human motion. Each class of motion segments is called motion primitive. Motion segments within one motion primitive should be very similar to each other. For example, for walking, motion segments must all start out on the same foot, having same number of steps and so on.

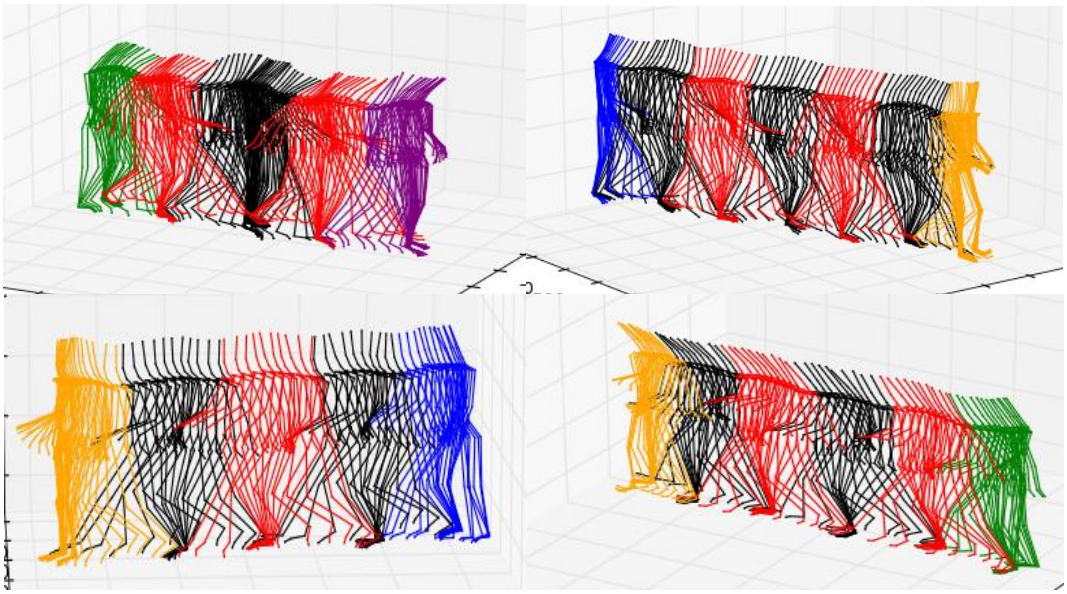


Figure 3. 1: Infinite walking styles can be decomposed as finite high-level structure. Each colored segment represents one motion primitive.

Therefore, based on the prior knowledge about different actions, we can define motion primitives for each action, then decompose it into distinctive motion primitives, and represent possible transition between motion primitives as a directed edge in the graph $G = (V, A)$. For example, figure 3.2 illustrates the graph representation of walking. Nodes are corresponding to motion primitives, including "right first starting" $M(\boldsymbol{s_0})$, "left first starting" $M(\boldsymbol{s_1})$, "right first walking" $M(\boldsymbol{s_2})$, "left first walking" $M(\boldsymbol{s_3})$, "left first ending" $M(\boldsymbol{s_4})$, and "right first ending" $M(\boldsymbol{s_5})$. In general, different styles of walking can be represented as a combination of six motion primitives. Edges represents how likely transition occurs from $M(\boldsymbol{s_i}) \rightarrow M(\boldsymbol{s_j})$.

In order to maintain rich variants of human actions, the model should able to generate infinite kinds of poses from finite samples. This is achieved by using a statistical model for each motion primitive. As mentioned before, statistical model based motion synthesis approaches have the advantage of generating infinite variant motion from finite training samples. Therefore, Motion Graphs++ has the ability to represent continuous style variants of actual human motion.

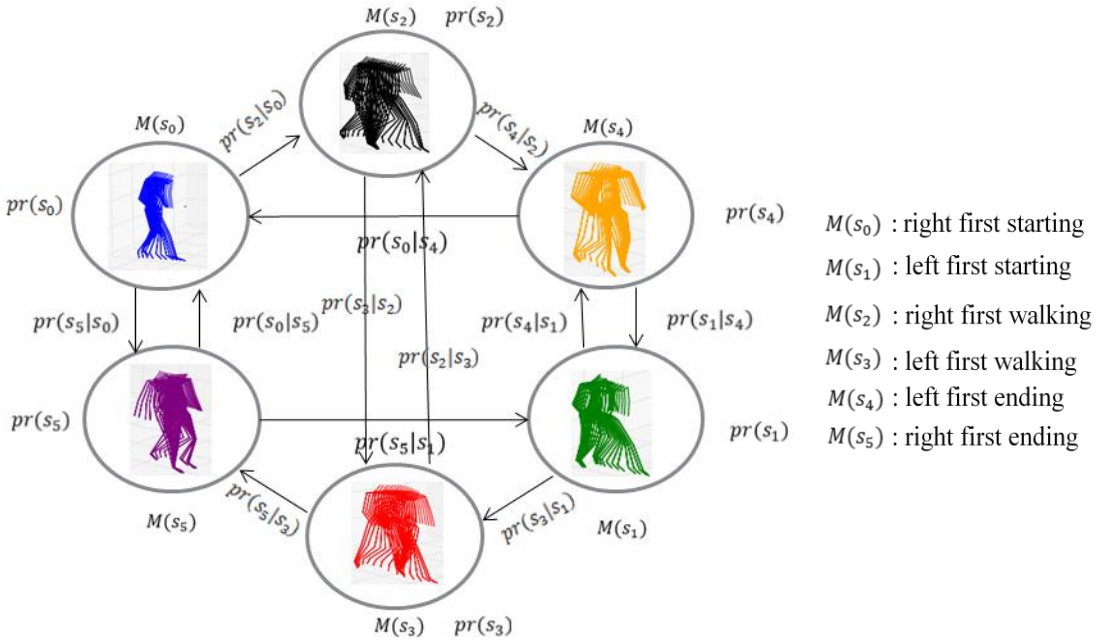


Figure 3. 2: A morphable graph model for walking. Walking motions are decomposed into six distinctive motion primitives, which are "*right first starting*", "*left first starting*", "*right first walking*", "*left first walking*", "*left first ending*", and "*right first ending*". Each graph node is a generative statistical model, which includes a morphable function $X_i = M(\boldsymbol{s_i})$, and a prior probability $pr(\boldsymbol{s_i})$. $\boldsymbol{s_i}$ is the low dimensional representation of each motion segment. The morphable function models continuous human motion style variations with a small number of parameters $\boldsymbol{s_i}$. Each edge represents a possible transition from motion primitive *i* to primitive *j*. This transition is also model by a statistical model with a probability distribution function $pr(\boldsymbol{s_j}|\boldsymbol{s_i})$.

Figure3.3 gives an overview of procedures of Motion Graphs++. The details of each step will be explained in the remaining part of this chapter. Section 3.1 gives an introduction of motion decomposition. Section 3.2 will introduce how to align decomposed motion segments to the reference motion segment. In section 3.3, we will discuss represent motion segment in a low dimensional, compact vector. Morphable model (nodes in motion graph) construction will be introduced in section 3.4 and transition distribution modeling (edges in motion graph) will be described in section 3.5. The remaining parts, motion synthesis with constraints are beyond the scope of this work.

In this chapter, a consistent mathematic notation is employed. Vectors are denoted by lower case bold Roman letters such as $\boldsymbol{x}$, and all vectors are assumed to be column vectors. A superscript T denotes the transpose of a matrix or vector, so that $\boldsymbol{x}^T$ will be a row vector. Uppercase bold roman letters, such as $\boldsymbol{M}$, denote matrices.

3.1 Motion Decomposition

The purpose of motion decomposition is to segment a long motion into small segments, group the structurally similar segments as same class (motion primitive). So the unstructured motion data becomes contact-consistent, structurally similar motion segments. This is the foundation of our approach, since our synthesized motion is based on motion primitives. A good decomposition should make the similarity very small for motion segments within the same motion primitive, and very large for motion segments in different motion primitives. Figure 3.4 illustrates the workflow of motion decomposition for walking in our work. Basically, it has three parts:

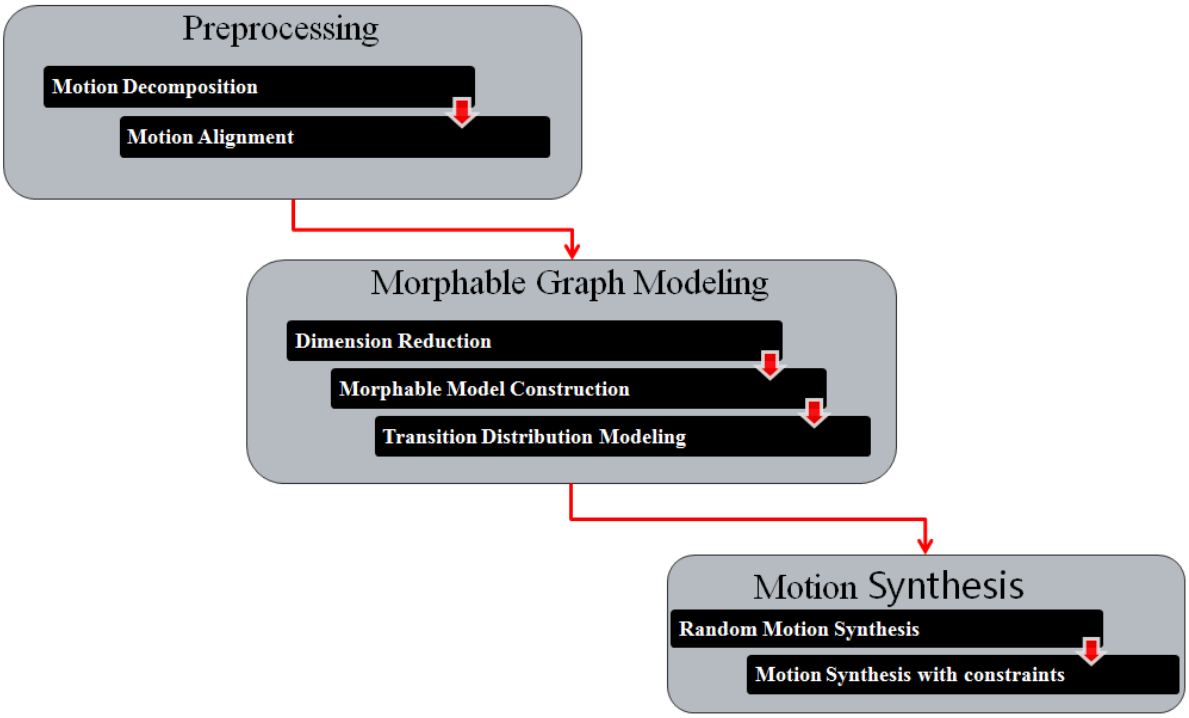


Figure 3. 3: The procedure of our approach

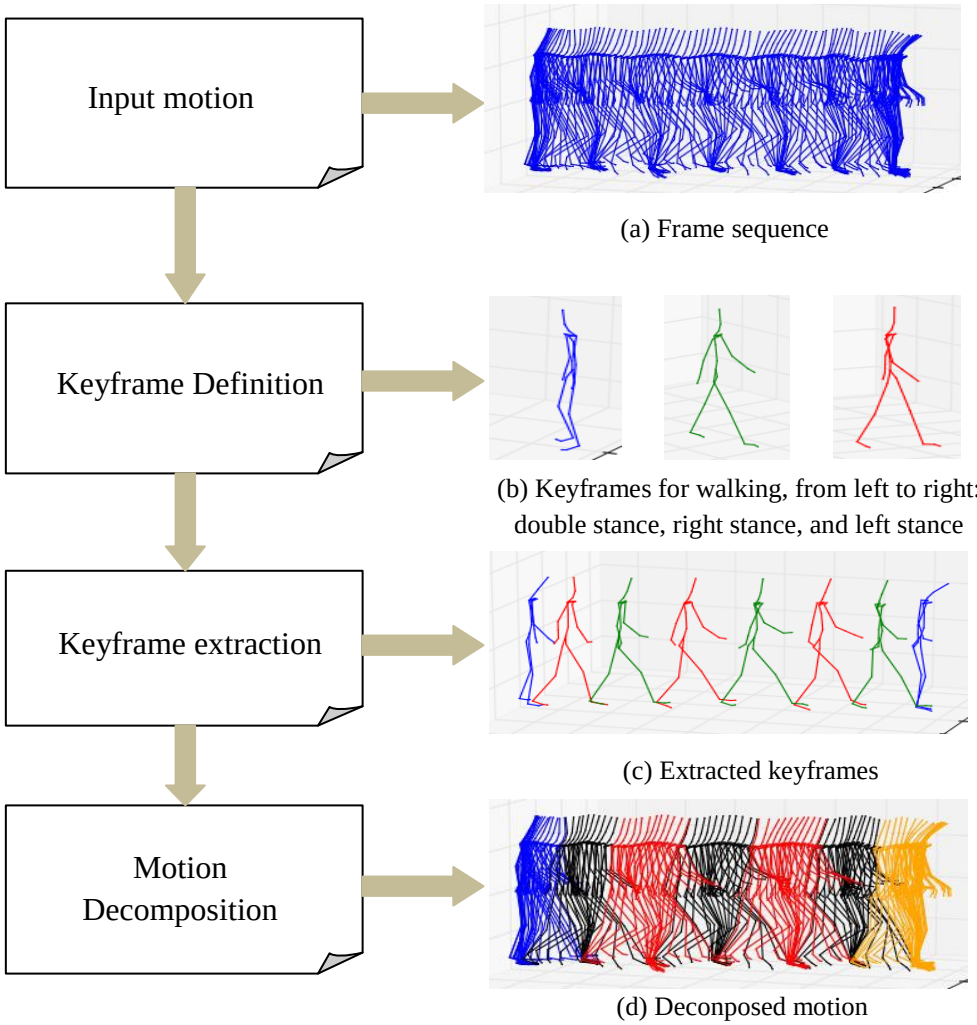


Figure 3. 4: Work-flow of motion decomposition

- Define keyframes from input motion
- Extract keyframes from input motion by manual labeling or automatic extraction.
- Decompose input motion into motion segments based on extracted key frames

3.1.1 KeyFrame Definition and Extraction

The first step of motion decomposition is to define a proper set of keyframes from input motion. Then the coming question is what is a proper keyframe for input motion ? Here, we use the same definition as motion graphs++ [15]:

1. Frame instance when contact state transition occurs. For example, during walking, foot contact with ground alternatively changes (e.g. left foot strike).
2. Frame instance when highest visual content changes. For example, for walking, from the pose left stance or right stance to double stance.

Based on these two criteria, we can manually define keyframes in the input motion. The number of the keyframes for a good decomposition depends on the complexity of the motion. If it is a simple motion, always repeating similar pattern, e.g. walking, running, then a small

number of keyframes is enough for a good segmentation. But if the motion is quite complex, e.g. working with a screwdriver on assembly line, a lot of keyframes are needed in order to make motion segments within the same primitive similar to each other. In figure 3.4, we define three keyframes for our walking scenario. Since walking is a quite straightforward motion, which is basically a sequence of alternating right and left-stance phases, so three keyframes: two frames of contact state changes: right stance and left stance and one frame with highest visual content changes: double stance(cf. figure 3.4 (b)) should be enough.

After we get clear definition of keyframes, the next question is how to find these keyframes from motions. The simplest solution is to select keyframes from input motion manually, which is easy to implement and the quality of result is good. However, this approach becomes very tedious and time-consuming when the motion data set is very large, especially for our statistical motion synthesis approach, a lot of training motions are required to get a realistic result. So that's why we want to explore some automatic keyframe extraction algorithms.

By using some extra sensors or some dominant features in visualization, it is not hard to tell whether contact points are reached or not, which makes detecting contact transition keyframes become a simple task. However, the detection of keyframes that highest visual content changes is not trivial, because highest visual content change a quite subjective measure. Different people could have different opinions about it. Therefore, we employ a supervised learning algorithm to automatically extract keyframes with highest visual content changes. Firstly, we select some motions as a training data, then manually annotate keyframes which are highest visual content changes in the training data. Next, we use a similarity measure between two frames [14] to find all frames which are similar to labeled frames in the rest motions. The implementation details of automatic keyframe extraction in our work will be presented in chapter 4.

3.1.2 KeyFrame based Motion Decomposition

The last step of motion decomposition is to use extracted keyframes to segment motions into small segments and group them into motion primitives. The number of motion primitives for each action is decided by the number of possible transitions from one key frame to another. Motion segments that share the same starting and ending keyframes belong to the same motion primitive. Figure 3.6 illustrates decomposition of walking. There are three keyframes for walking, and each of them can transfer to others. So there are six motion primitives in total.

3.2 Motion Alignment

After motion decomposition, the motion segments in each motion primitive are not ready for statistical modeling steps. Figure 3.6 (a) gives an example that two motion segments which are within the same motion primitive, could have different root position, orientation and number of frames. However, we want motion segments within same primitive should very similar to each other, and contain the same number of frames, which are suitable for statistical modeling, as shown in figure 3.6(b). Here, we use dynamic time warping techniques [39] to align all motion segments in one primitive to a reference motion segment. The reference segment can be manually selected or automatically found. In our work, we automatically pick

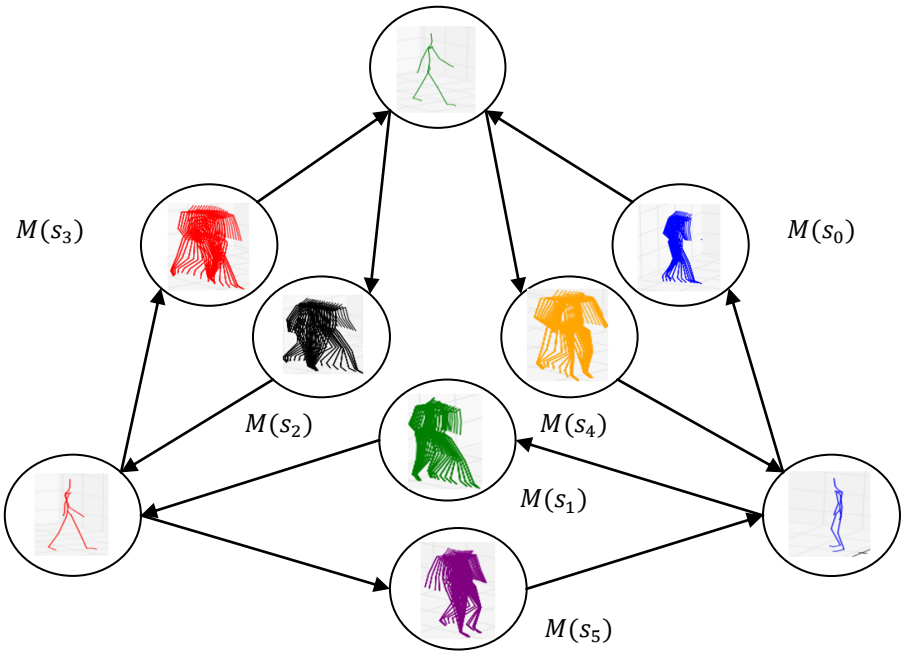


Figure 3. 5: Keyframe based motion decomposition for standard walking.

reference segments by taking the motion segment which has minimum average motion distance to others.

3.2.1 Coordinate Frame Distance

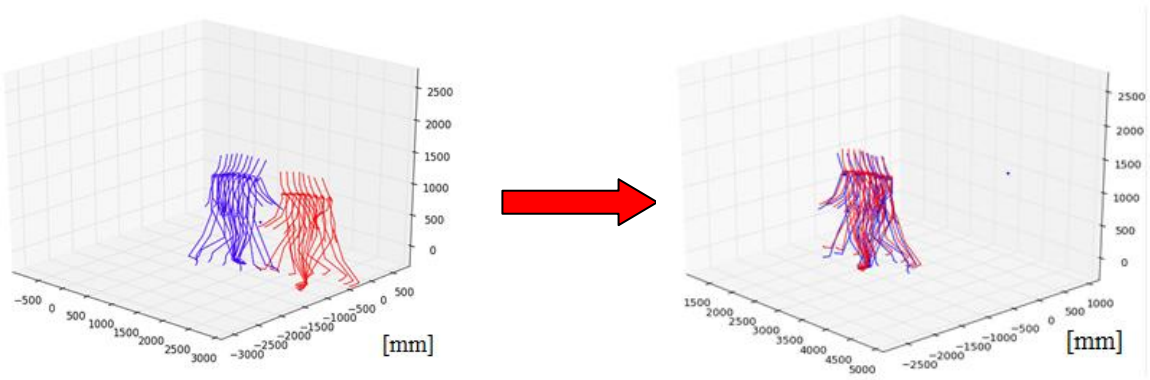
One of fundamental ingredients of dynamic time warping is similarity measure of two sequences. In our case, we want to align two sequences of frames. So a good frame distance measure to tell how similar each frame to frames in reference motion segment is needed. Eq. (3.1) defines the distance between two frames, which is a weighted sum of Euclidean distance of corresponding joints $\boldsymbol{p_i}$ and $\boldsymbol{p_i'}$ in two frames. As two structurally similar segments could have different position and orientation, we address this problem by aligning motion segments in a translation- and rotation- invariant way, which translates each pose in the floor plane and rotate it about the y (vertical) axis. The formula of weighted frame distance is

$$D(F_1, F_2) = \min_{\theta, x_0, z_0} \sum_{i=1}^{n} w_i \| \boldsymbol{p}_i - \boldsymbol{T}_{\theta, x_0, z_0} \boldsymbol{p}'_i \|^2 \tag{3.1}$$

where $\boldsymbol{p_i}$ and $\boldsymbol{p_i'}$ are global position of i^{th}joint respectively. $\boldsymbol{T}_{\theta, x_0, z_0}$ rotates point $\boldsymbol{p_i'}$ about y (vertical) axis by θ degrees and then translates it by $(x_0, 0, z_0)$, $w_i$is the weight for each joint.

$$\boldsymbol{T}_{\theta, x_0, z_0} = \begin{pmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_y & 0 & -\sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.2}$$

Plug (3.2) into weighted frame distance (3.1), we can have our objective function $f(\theta, x_0, z_0)$:



(a) (b)

Figure 3. 6: Comparison of two motions before alignment with after alignment. (a) Two motion segments belong to same motion primitive before alignment. (b) After motion alignment.

$$D(F_1, F_2) = \min_{\theta, x_0, z_0} f(\theta, x_0, z_0) \tag{3.3}$$

$$f(\theta, x_0, z_0) = \sum_{i=1}^{n} w_i \left\| \begin{pmatrix} p_{ix} \\ p_{iy} \\ p_{iz} \\ 1 \end{pmatrix} - \begin{pmatrix} p_{ix}' \cos\theta_y - p_{iz}' \sin\theta_y + x_0 \\ p_{iy}' \\ p_{ix}' \sin\theta_y + p_{iz}' \cos\theta_y + z_0 \\ 1 \end{pmatrix} \right\|^2 \text{..... ....} \tag{3.4}$$

The optimal solution for θ, x_0, z_0 can be computed by taking partial derivatives of objective function with respect to the variables θ, x_0, z_0.

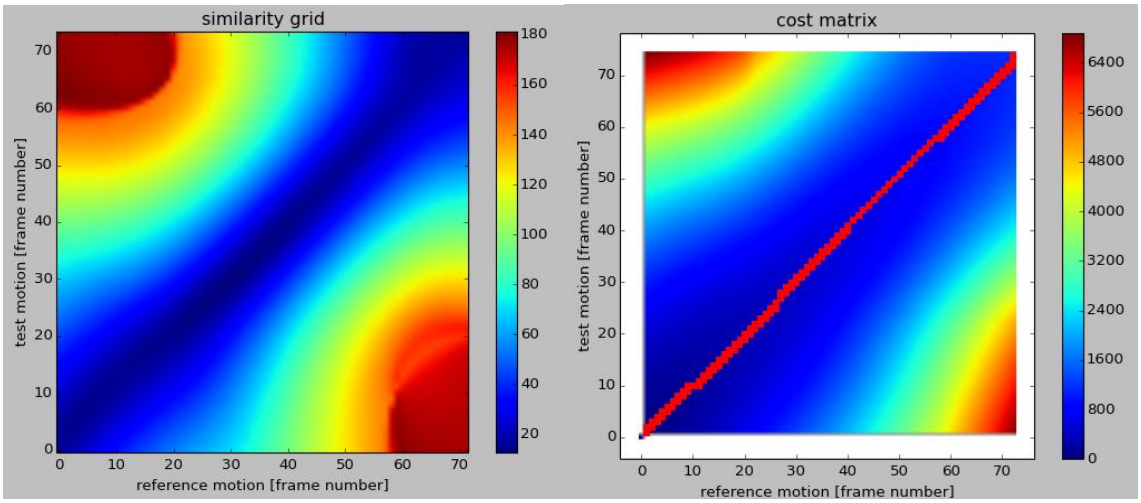
$$\theta = \arctan \frac{\sum_{i=1}^{n} w_i (x_i z_i' - x_i' z_i) - \sum_{i=1}^{n} w_i (\bar{x}\bar{z}' - \bar{x}'\bar{z})}{\sum_{i=1}^{n} w_i (x_i x_i' + z_i z_i') - \sum_{i=1}^{n} w_i (\bar{x}\bar{x}' + \bar{z}\bar{z}')} \tag{3.5}$$

$$x_0 = \bar{x} - \bar{x}' \cos\theta_y - \bar{z}' \sin\theta_y \tag{3.6}$$

$$z_0 = \bar{z} + \bar{x}' \sin\theta_y - \bar{z}' \cos\theta_y \tag{3.7}$$

where $\bar{x} = \sum_{i=1}^{n} w_i x_i / \sum_{i=1}^{n} w_i$ and the other barred terms are defined similarly.

Now, the remaining question is how to assign the weights w_i. In some previous works [15], the weights were assigned equally. However, we use a different approach in our work. Each joint is assigned a weight which is inverse linearly propositional to the depth of the joint in our hierarchical skeleton structure (cf. figure 3.3). For our motion capture data, we find out that to weight different joints preferentially could give a better result. This is because our markerless motion capture data is noisy, and the noise is heavier for some leaf joints, e.g. hand, foot toe and so on, than the joints which are close to root. If we do not differentiate the importance of joints, the distance function will translate and rotate the test motion in a way to



(a) (b)

Figure 3. 7: (a) Color coded distance matrix between reference motion and test motion. (b) Color coded cost matrix calculated from distance matrix, the red curve is time warping curve found by dynamic time warping.

minimize Euclidean distance of each joint equally. As there are quite a lot of joints in arms and legs in our skeleton model, so that the noise in some "unimportant" joints will heavily influence alignment results.

3.2.2 Dynamic Time Warping

Dynamic time warping (DTW) algorithm is a well-known technique to find the optimal alignment between two given sequences, which is widely applied to speech signal alignment [39]. In order to align two motions, the first step is to measure the similarity between two motions. Using the distance measure defined in equation 3.1, we can create a distance matrix which represents the similarity between reference motion and cost motion. Figure 3.7 (a) gives an example of distance matrix, where columns are corresponding to frames in test motion and rows are corresponding to frames in reference motion. For each frame in reference motion, we want to find an best matching frame in test motion. The final warping curve will be a path from left boundary to right boundary in figure 3.7 (a). Dynamic time warping algorithm is applied to find a minimal cost connecting path. Here, the cost is summed distance between corresponding frames. As shown in figure 3.7 (b), the red curve is the optimal path for the distance matrix. However, not arbitrary curves are valid for alignment, only the curves which satisfy four properties will be considered.

1. **Monotonicity.** The warping curve should be monotonic increase or decrease. It means that we go though reference frames either entirely forward or entirely backward in time.
2. **Continuity.** The warping curve should be a connected curve. Each cell on the curve must share a corner or edge with another cell on the curve.

Algorithm 1: Dynamic Time Warping Algorithm:

Given: Two motions $m_1 = (f_1^1, \dots, f_n^1), m_2 = (f_1^2, \dots, f_m^2)$ with n and m frames in each.

Step 1: Compute cost matrix

DTW(m_1, m_2) {
/* Let a two dimensional matrix C be the store of accumulated distance between two frame sequences, such that C[0,...,n; 0,...,m], and *i,j* are loop index. */
C[0,0] := 0
for i:=1 to n:
    $C[0,i] \coloneqq inf$
for j:=1 to m:
    $C[j,0] \coloneqq inf$
for i:=1 to n:
    for j:=1 to m:
        $C[i,j] \coloneqq Distance(f_i^1, f_j^2) + min\,(C[i-1,j], C[i,j-1], C[i-1,j-1])$
return $C[i,j]$

Step 2: Find optimal path in cost matrix

pathSearch(C[n,m]){
/* C[n,m] is the cost matrix. Let a two dimensional binary matrix P be the store of optimal path. 1 in position (i,j) means finding an optimal corresponding between frames f_i^1 and f_j^2.
P[0,...,n-1; 0,...,m-1], and i,j are loop index.*/
i := j := 1
while i != n and j != m:
    if reach slope limit in horizontal direction: // search in vertical direction
        $P[i',j'] \coloneqq 1$, // where $min_{[i',j']}\, C[i',j'], [i',j'] \in \{[i+1,j],[i+1,j+1]\}$
        $i \coloneqq i', j \coloneqq j'$, // update index
    else if reach slop limit in vertical direction: // search in horizontal direction
        $P[i',j'] \coloneqq 1$, // where $min_{[i',j']}\, C[i',j'], [i',j'] \in \{[i,j+1],[i+1,j+1]\}$
        $i \coloneqq i', j \coloneqq j'$, // update index
    else:
        $P[i',j'] \coloneqq 1$, // where $min_{[i',j']}\, C[i',j'], [i',j'] \in \{[i,j+1],[i+1,j+1],[i+1,j]\}$
        $i \coloneqq i', j \coloneqq j'$, // update index
if i == n: // go to upper-right corner vertically
    for j := j to m:
        $P[i,j] \coloneqq 1$,
if j == m: // go to upper-right corner horizontally
    for i := i to n:
        $P[i,j] \coloneqq 1$,
return $P[i,j]$
}

Output: path matrix P

3. **Boundary Point.** In our work, DTW is applied to motion segments within the same motion primitive. We have the prior knowledge that the first frame and last frame of each motion segment are the same key frame respectively. So the first frames and the last frames must correspond, which means that the path lower-left and upper-right corners of the grid in figure 3.7 (b).
4. **Slope Limit.** If there are many consecutive frames mapping from test motion to one frame in reference motion or vice versa, this means that these two motions are not similar to each other, which breaks our assumption that motion segments within one primitive should similar to each other. So we do not consider this case in our work. Based on previous study [25], we set a slope limit of 5 so that at most 5 consecutive horizontal steps or consecutive vertical steps may be taken. However, in order to satisfy boundary point condition as well, we allow exception for slope limit in the boundary.

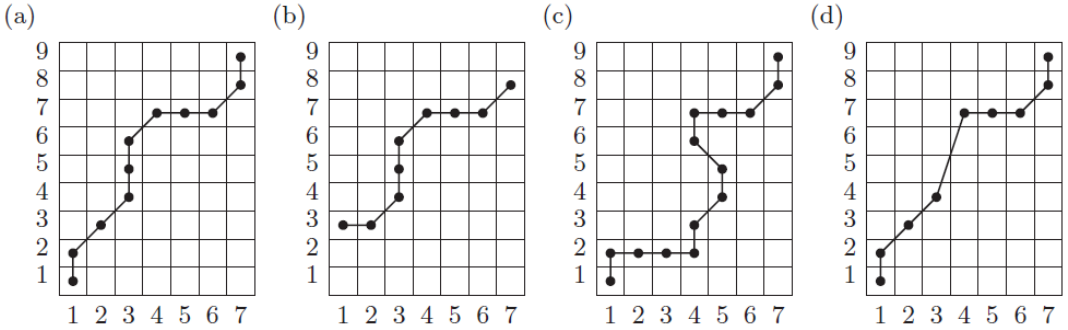


Figure 3. 8: Examples of legal and illegal paths. (a) Admissible warping path satisfying all constraints defined in 3.2.2. (b) Boundary point constraint is violated. (c) Monotonicity constraint is violated. (d) Continuity constraint is violated [41].

3.3 Dimension Reduction

After motion data preprocessing, the next step is to construct a statistical model to model the distribution of registered motion segments. Each motion segment is represented as a vector $\boldsymbol{s}_n$. which is a very long vector. $\boldsymbol{s}_n$ is constructed by sequentially concatenating all parameters in each frame of *n*-th motion segment. The dimensions of motion segment vector is:

$$d_{total} = n_f * d_f = n_f * \big((n_j - 1) * 3 + 6\big) \tag{3.8}$$

where n_f is the number of frames, d_f is the dimension of each frame and n_j is the number of joints in human skeleton model. Each joint has three degree of freedoms (DOF), which are rotations in x, y and z axis. And for the root joint, it also contains global position information.

In our case, the length of each motion segment vector is more than 4000. It is very hard to model the distribution on such a high dimensional space. The required number of training data increases exponentially with the increase of dimensions, which is known as curse of dimensionality. So before we throw motion data into statistical modeling, we need to transform the motion data from high dimensional space into low dimensional space. This is

possible because from observation, we know that most dynamic human behaviors are intrinsically low dimensional, for example, legs and arms operating in a coordinated way [42]. Based on this observation, one assumption can be drawn that most dynamic human motions can be represented by low dimensional degrees of freedom.

3.3.1 Principal Component Analysis

Principal component analysis (PCA) is one of the most simple and one of the most widely used methods for dimensionality reduction methods [43]. The idea of PCA is that given a set of data which is in the d dimensional space $\boldsymbol{S}$, we want to find an affine subspace $\boldsymbol{V}$ in $\mathbb{R}^k$, $k \leq d$, which can explain most of the variance of original data.

The task of finding affine subspace K can be formulized as minimizing least square error problem. Suppose we have N data points $\{\boldsymbol{x}_i^N\}$ in $\mathbb{R}^d$, and $\sum_{i=1}^N \boldsymbol{x}_i = \boldsymbol{0}$. What we want is to project the data on to a subspace $V = \{v_1, \dots, v_k\}$, which minimize the squared loss function:

$$err = \frac{1}{N}\sum_{i=1}^N ||\boldsymbol{x_i} - \boldsymbol{V}\boldsymbol{V}^{\boldsymbol{T}}\boldsymbol{x_i}||^2 \tag{3.9}$$

where $\boldsymbol{V} = \{\boldsymbol{v_1}, \dots, \boldsymbol{v_k}\}$ is the projection matrix, $\boldsymbol{x_i}, \boldsymbol{v_i}$ are both column vectors.

The close-form solution of equation (3.9) can be derived as follows:

$$\begin{aligned} err &= \underset{\boldsymbol{V}, \boldsymbol{v_i}^T\boldsymbol{v_j}=\delta(i-j), \boldsymbol{v_i}, \boldsymbol{v_j}\in\boldsymbol{V}}{\arg\min} \frac{1}{N}\sum_{i=1}^N \|\boldsymbol{x}_i - \boldsymbol{V}\boldsymbol{V}^T x_i\|^2 \\ &= \underset{\boldsymbol{V}, \boldsymbol{v_i}^T\boldsymbol{v_j}=\delta(i-j), \boldsymbol{v_i}, \boldsymbol{v_j}\in\boldsymbol{V}}{\arg\min} \frac{1}{N}\sum_{i=1}^N (\boldsymbol{x}_i^T\boldsymbol{x}_i - \boldsymbol{x}_i^T\boldsymbol{V}\boldsymbol{V}^T\boldsymbol{x}_i) \\ &= \underset{\boldsymbol{V}, \boldsymbol{v_i}^T\boldsymbol{v_j}=\delta(i-j), \boldsymbol{v_i}, \boldsymbol{v_j}\in\boldsymbol{V}}{\arg\max} \boldsymbol{V}\frac{1}{N}(\sum_{i=1}^N \boldsymbol{x}_i^T\boldsymbol{x}_i)\boldsymbol{V}^T \end{aligned} \tag{3.10}$$

where $\boldsymbol{C} = \frac{1}{N}\sum_{i=1}^N \boldsymbol{x_i}\boldsymbol{x_i^T}$ is the covariance matrix of the given data. From (3.10), we can see that minimizing projection residuals is equivalent to maximize projection variance.

Now, let us consider the projection variance in one dimension in subspace V:

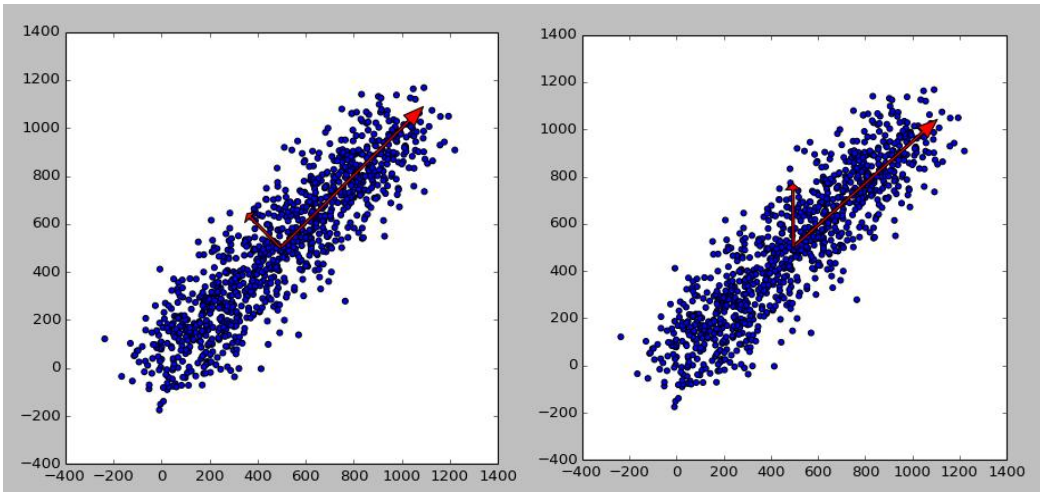
$$\sigma^2 = \underset{\boldsymbol{v}\in\Re^d, \|\boldsymbol{v}\|=1}{\arg\max} \boldsymbol{v^T C v} \tag{3.11}$$

Apply Lagrange multiplies, the objective function can be written as:

$$L(\boldsymbol{v}, \lambda) = \boldsymbol{v^T C v} - \lambda(\boldsymbol{v^T v} - 1) \tag{3.12}$$

The maxima can be found by taking partial derivative of $L(\boldsymbol{v}, \lambda)$ with respect to variables $\boldsymbol{v}, \lambda$.

$$\frac{\partial L}{\partial \boldsymbol{v}} = 2\boldsymbol{C}\boldsymbol{v} - 2\lambda\boldsymbol{v} = 0 \;\Rightarrow\; \boldsymbol{C}\boldsymbol{v} = \lambda\boldsymbol{v} \tag{3.13}$$



(a) (b)

Figure 3. 9: Illustration of PCA on 2D points. (a) Standard PCA: Points are projected to an orthogonal axes (red arrows). (b) Weighted PCA: Weights vector $\boldsymbol{w}$ = *(10, 1)*, the new projection axes are oblique rather than orthogonal.

$$\frac{\partial L}{\partial \lambda} = \boldsymbol{v}^T\boldsymbol{v} - 1 = 0 \;\Rightarrow\; \boldsymbol{v}^T\boldsymbol{v} = 1 \tag{3.14}$$

Plug (3.13) into (3.11):

$$\sigma^2 = \underset{\boldsymbol{v}\in\Re^d,\|\boldsymbol{v}\|=1}{\arg\max}\, \boldsymbol{v}^T\boldsymbol{C}\boldsymbol{v} = \underset{\boldsymbol{v}\in\Re^d,\|\boldsymbol{v}\|=1}{\arg\max}\, \boldsymbol{v}^T\lambda\boldsymbol{v} = \underset{\boldsymbol{v}\in\Re^d,\|\boldsymbol{v}\|=1}{\arg\max}\, \lambda\boldsymbol{v}^T\boldsymbol{v} = \lambda \tag{3.15}$$

So finding the projection direction with maximum variance is to find the maximum eigenvalue of covariance matrix, and the corresponding eigenvector is projection direction. So the affine subspace *k* consists of *k* eigenvectors of $\boldsymbol{C}$ corresponding to the *k* largest eigenvalues of $\boldsymbol{C}$. And these eigenvectors are called the principal components of the data. Basically, PCA can reduce the linear redundancy of original data. Figure 3.9 (a) shows an example of PCA in two dimensional space.

3.3.2 Weighted Principal Component Analysis

Conventional PCA treats the importance of each feature equally. However, this is not the case in our work. Each motion sample is represented as a vector of concatenating root position and joint orientation of each frame. As we mentioned before, BVH file uses a hierarchical skeleton structure (see figure 2.5), which means the change of rotation angles of each joint has different influence on the whole skeleton. For instance, a small change in root orientation is easy to be observed, because the change will not only influence the current joint, but influence all its child joints as well. So a small change in root joint will change the whole human pose. However, for the leaf joints, e.g. hand and toe, a small change of rotation angle is actually negligible. This observation is our motivation to use a feature weighted PCA rather than a standard PCA.

Now, our objective function is a weighted least squared loss:

$$err = \frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{W}(\boldsymbol{x_i} - \boldsymbol{V}\boldsymbol{V^T}\boldsymbol{x_i})||^2 \tag{3.16}$$

where $\boldsymbol{W} \in \Re^{d\times d}$ diagonal weight matrix, each element of $\boldsymbol{W}$ is the weight of corresponding feature in $\boldsymbol{x_i}$. Here, we choose the weights that are inversely proportional to the depth of each joint.

However, deriving close-form solution for weighted PCA is not as straightforward as standard PCA [53]. Let's consider a simplified case, finding a projection vector $\boldsymbol{v}$ to minimize the weighted least squared loss function:

$$\begin{aligned} err &= \arg\min_{v\in\Re^d, \|v\|=1} \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{W}(\boldsymbol{x_i} - \boldsymbol{v}\boldsymbol{v^T}\boldsymbol{x_i})\|^2 \\ &= \arg\min_{v\in\Re^d, \|v\|=1} \frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{x_i} - \boldsymbol{v}\boldsymbol{v^T}\boldsymbol{x_i})^T \boldsymbol{W}^2 (\boldsymbol{x_i} - \boldsymbol{v}\boldsymbol{v^T}\boldsymbol{x_i}) \\ &= \arg\min_{v\in\Re^d, \|v\|=1} \frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{x_i^T}\boldsymbol{W}^2\boldsymbol{x_i} - \boldsymbol{v^T}\boldsymbol{W}^2\boldsymbol{x_i}\boldsymbol{x_i^T}\boldsymbol{v} - \boldsymbol{v^T}\boldsymbol{x_i}\boldsymbol{x_i^T}\boldsymbol{W}^2\boldsymbol{v} + \boldsymbol{v^T}\boldsymbol{x_i}\boldsymbol{x_i^T}\boldsymbol{v}\boldsymbol{v^T}\boldsymbol{W}^2\boldsymbol{v}) \\ &= \arg\max_{v\in\Re^d, \|v\|=1} \boldsymbol{v^T}\boldsymbol{W}^2\boldsymbol{C}\boldsymbol{v} + \boldsymbol{v^T}\boldsymbol{C}\boldsymbol{W}^2 + \boldsymbol{v^T}\boldsymbol{C}\boldsymbol{v}\boldsymbol{v^T}\boldsymbol{W}^2\boldsymbol{v} \end{aligned} \tag{3.17}$$

Eq. (3.17) is more challenging to optimize than equation (3.10) because it is quartic in $\boldsymbol{v}$. Greenacre et al. addressed this problem by defining so called Generalized Singular Value Decomposition (GSVD) [54, 55]. The standard Singular Value Decomposition (SVD) provides a efficient and powerful tool to calculate eigenvalue and eigenvector. The SVD of a $(n \times m)$ matrix $\boldsymbol{X}$ is defined in equation (3.18):

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V^T} \tag{3.18}$$

where $\boldsymbol{U} \in \Re^{n\times r}, \boldsymbol{V} \in \Re^{m\times r}$ are orthonormal matrix, $\boldsymbol{D} \in \Re^{r\times r}$ is a diagonal matrix. And it is easy to prove that $\boldsymbol{V}$, $\boldsymbol{D}$ give eigenvectors and square roots of eigenvalues of matrix $\boldsymbol{X^T X}$ respectively.

Now suppose $\boldsymbol{\Omega}, \boldsymbol{\Phi}$ are positive definite symmetric matrices, GSVD is defined as:

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{M}\boldsymbol{B^T} \tag{3.19}$$

where $\boldsymbol{A} \in \Re^{n\times r}, \boldsymbol{B} \in \Re^{m\times r}$ satisfy $\boldsymbol{A^T\Omega A} = \boldsymbol{I_r}, \boldsymbol{B^T\phi B} = \boldsymbol{I_r}$ respectively, and $\boldsymbol{M} \in \Re^{r\times r}$ is a diagonal matrix.

From standard SVD, each data point $x_i \in X$ can be represented as:

$$\boldsymbol{x_i} = \sum_{k=1}^{r} u_{ik} d_k \boldsymbol{v_k} \tag{3.20}$$

where u_{ik} is the *(i,k)*-th element of $\boldsymbol{U}$, d_k is *k*-th diagonal element of $\boldsymbol{D}$, and $\boldsymbol{v_k}$ is k-th column of $\boldsymbol{V}$.

If we want to project data to a subspace *K*, the first *k* eigenvectors will be retained.

$$\tilde{\boldsymbol{x}}_i = \sum_{k=1}^{K} u_{ik} d_k \boldsymbol{v}_k \tag{3.21}$$

From previous derivation of PCA, we know that $\tilde{\boldsymbol{x}}_i$ gives best possible rank *k* approximation to $\boldsymbol{x}_i$ [56], in the sense of minimizing:

$$err = \frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i||^2 \tag{3.22}$$

In our case, we want to have best possible rank *k* approximation $\tilde{\tilde{\boldsymbol{x}}}_i$ to minimize weighted least squared loss:

$$err = \frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{W}(\boldsymbol{x_i} - \tilde{\boldsymbol{x}}_{\boldsymbol{i}})||^2 \tag{3.23}$$

It is can be shown (detailed derivation can be found in [55], p. 39) that if $\boldsymbol{X}$ is decomposed by GSVD, and $\boldsymbol{\Omega}, \boldsymbol{\Phi}$ are diagonal matrix, with elements $\omega_i, i = 1, \dots, n;\ \phi_i, i = 1, \dots, m$, then $\tilde{\tilde{\boldsymbol{x}}}_{\boldsymbol{i}}$ can minimize:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \omega_i \phi_j (x_{ij} - \tilde{\tilde{x}}_{ij})^2 \tag{3.24}$$

$$\tilde{\tilde{\boldsymbol{x}}}_i = \sum_{k=1}^{K} a_{ik} m_k \boldsymbol{b}_k \tag{3.25}$$

where x_{ij}, a_{ik} are the *(i,k)*-th element of $\boldsymbol{X}$ and $\boldsymbol{A}$, m_k is *k*-th diagonal element of $\boldsymbol{M}$, and b_k is *k*-th column of $\boldsymbol{B}$.

Since we only have weights for features, so $\boldsymbol{\Omega} = \boldsymbol{I_n}, \boldsymbol{\Phi} = \boldsymbol{W}$ in our case. Then the next question is how to calculate GSVD of $\boldsymbol{X}$. Define $\tilde{\boldsymbol{X}} = \boldsymbol{\Omega}^{1/2}\boldsymbol{X}\boldsymbol{\phi}^{1/2}$, the standard SVD of $\tilde{\boldsymbol{X}}$ can be written as:

$$\tilde{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\boldsymbol{T}} \tag{3.26}$$

$$\boldsymbol{X} = \boldsymbol{\Omega}^{-\frac{1}{2}}\tilde{\boldsymbol{X}}\boldsymbol{\phi}^{-\frac{1}{2}} = \boldsymbol{\Omega}^{-\frac{1}{2}}\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\boldsymbol{T}}\boldsymbol{\phi}^{-\frac{1}{2}} = \boldsymbol{A}\boldsymbol{M}\boldsymbol{B}^{\boldsymbol{T}} \tag{3.27}$$

If we plug $\boldsymbol{\Omega} = \boldsymbol{I_n}, \boldsymbol{\Phi} = \boldsymbol{W}$ into (3.27), then

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\boldsymbol{T}}\boldsymbol{W}^{-\frac{1}{2}} = \boldsymbol{A}\boldsymbol{M}\boldsymbol{B}^{\boldsymbol{T}} \ \Rightarrow \ \boldsymbol{A} = \boldsymbol{U}, \boldsymbol{M} = \boldsymbol{D}, \boldsymbol{B} = \boldsymbol{W}^{-\frac{1}{2}}\boldsymbol{V} \tag{3.28}$$

Figure 3.9 illustrates the effect of weights on PCA in two dimensional space. (a) is the result of standard PCA. The new projection axes are orthogonal to each other. (b) is the result after applying a weight matrix $\boldsymbol{W} = \begin{pmatrix} 10 & 0 \\ 0 & 1 \end{pmatrix}$. The optimal projection axes are not orthogonal any more but oblique [57]. This is because the optimal eigenvectors $\boldsymbol{B}$ for weighted PCA is from rescaling the eigenvectors $\boldsymbol{V}$ of $\tilde{\boldsymbol{X}}$, which could destroy the orthonormal constraint [58].

By applying weighted PCA, we can use a few linear combinations of the variables to represent most of the variation in the motion data. A motion sample $\boldsymbol{x_i} \in \Re^d$ in one motion primitive, which is a high dimensional vector, now can be represented as a low dimensional vector $\boldsymbol{s_i} \in \Re^m, m \ll d$**.** The mapping from $\boldsymbol{x_i}$ to $\boldsymbol{s_i}$ is called morphable function *M*.

$$\boldsymbol{x} = M(\boldsymbol{s}) = \boldsymbol{p_0} + \boldsymbol{s}^T[\boldsymbol{p_1}, \dots, \boldsymbol{p_m}] \tag{3.29}$$

where $\boldsymbol{p_0}$ is the mean motion of samples within this motion primitive,$\boldsymbol{p_1}, \dots, \boldsymbol{p_m}$ are first m eigenvectors, and $\boldsymbol{s}$ is projection in each eigenvector.

3.4 Morphable Model Construction

After projecting motion data into low dimensional space, we now are ready to construct motion graph. As we mentioned before, each node in the motion graph is a statistical model to represent a type of structurally similar motion examples, instead of the store of motion examples themselves. Basically, given the finite samples of motion primitive s_i, we want to construct a continuous morphable function $M(\boldsymbol{s}_i)$, which can generate infinite variant style of motions, and estimate the prior distribution $p(\boldsymbol{s}_i)$. The task is challenging because we need to find an appropriate statistical model to model the distribution of motion data.

3.4.1 Gaussian Mixture Model

As we know, normal distribution is very important statistical distribution pattern occurring in many natural phenomena, so normal distribution could also be a good statistical model for human motion data. However, is a uni-model enough to capture the distribution of motion data? This question is not easy to answer by directly observing the data. Because it is not a simple task to visualize the motion data, even in the low dimensional space. Figure 4.10 visualizes motion segments in the same motion primitive projected on the first two eigenvectors. Although this is not the real case for our low dimensional data, which still in a space far more than two dimensions, the projection on the first two most important dimensions still indicate that the motion data could have several clusters. So we model the probability distribution $p(\boldsymbol{s_i})$ with a Gaussian Mixture Model (GMM), and regard the uni-model as a special case of GMM.

A Gaussian Mixture Model is a parametric probability density function represented as a weighted sum of K Gaussians:

$$p(\boldsymbol{s}_i) = \sum_{k=1}^{K} w_k N(\boldsymbol{s}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{3.29}$$

where the parameters w_k are the mixture weights of each Gaussian. They should be always positive and sum up to one. Each mixture weight w_k represents the likelihood that one observation $\boldsymbol{s}_i$ belongs to the k-th Gaussian. $N(\boldsymbol{s}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is a Gaussian distribution with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$. Figure 3.11 gives an example that a set of 2D points can be represented by the mixture of 2 Gaussians.

3.4.2 Expectation Maximization Algorithm

If the number of Gaussians K is fixed, the parameters of the GMM are automatically estimated using Expectation-maximization algorithm.

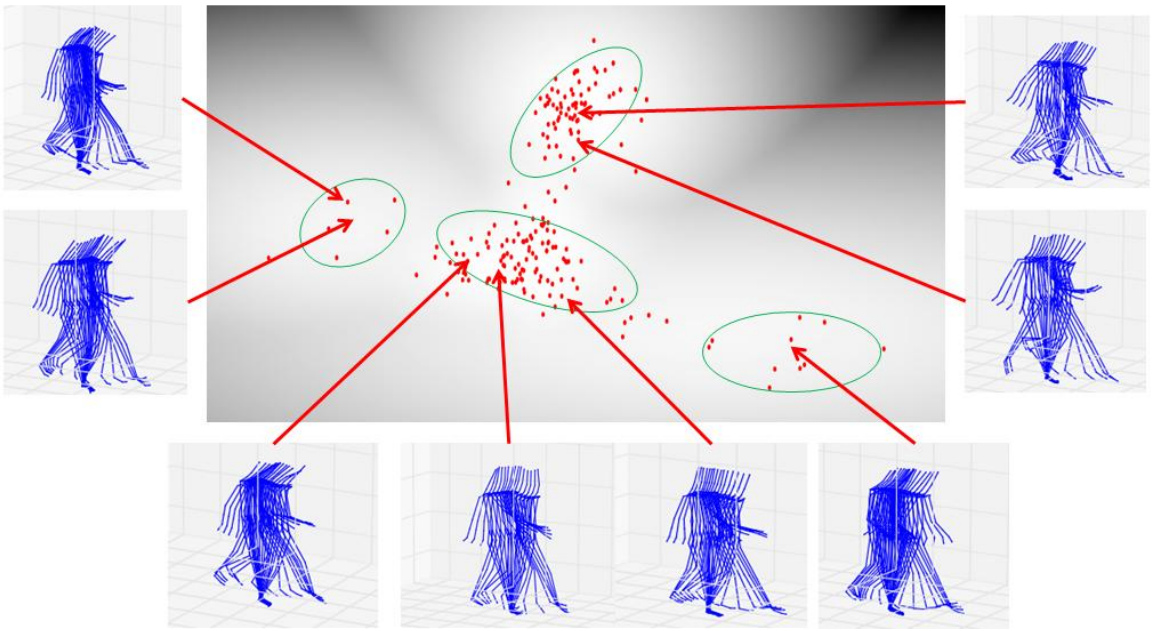


Figure 3. 10: Plot of motion segments in one motion primitive projected on first two eigenvectors. Each red dot represents a pre-record motion example. The background grayscale image is the distribution modeled by Gaussian mixture model.

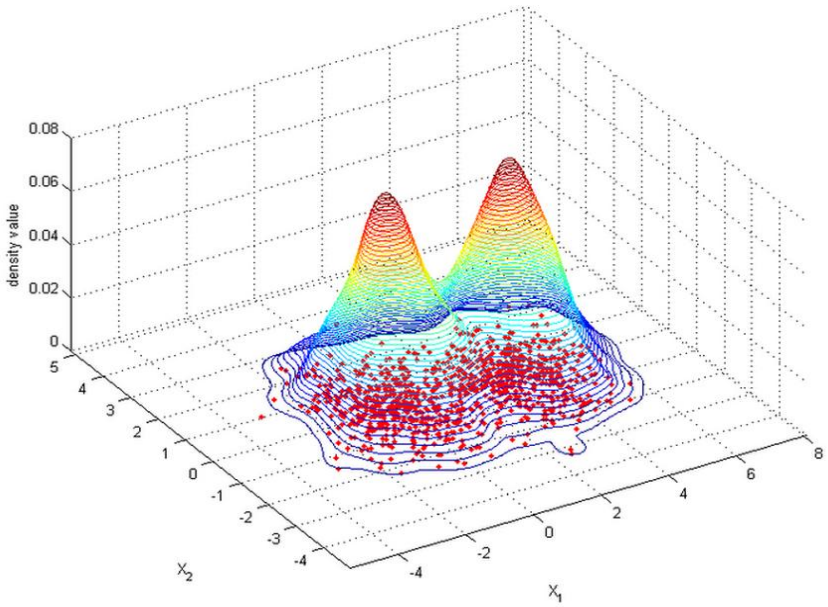


Figure 3. 11: Example of a set of 2D points drawn from the mixture of 2 Gaussians.

Algorithm 2: Expectation Maximization Algorithm

1. Initialization Step

Initialize the mean μ_k, covariance Σ_k and mixture weights w_k, and evaluate the initial value of log likelihood $ln\ p(\boldsymbol{s}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, w)$. s_i is a vector for data i.

2. Expectation Step

$w_k = p(C_k)$,C_k represents k-th Gaussian

Evaluate the responsibilities using the current parameter values:

$$p(C_i \mid \boldsymbol{s}_j) = \frac{p(C_i)N(\boldsymbol{s}_j \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{k=1}^{K} p(C_k)N(\boldsymbol{s}_j \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \tag{4.30}$$

3. Maximization Step

Re-estimate the parameters for each Gaussian model using the current responsibilities.

$$\boldsymbol{\mu}_k^{new} = \frac{\sum_{i=1}^{N} p(C_k \mid \boldsymbol{s}_i)\boldsymbol{s}_i}{\sum_{i=1}^{N} p(C_k \mid \boldsymbol{s}_i)} \tag{4.31}$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{\sum_{i=1}^{N} p_k(C_k \mid \boldsymbol{s}_i)(\boldsymbol{s}_i - \boldsymbol{\mu}_k^{new})(\boldsymbol{s}_i - \boldsymbol{\mu}_k^{new})^T}{\sum_{i=1}^{N} p(C_k \mid \boldsymbol{s}_i)} \tag{4.32}$$

$$w_k^{new} = p(C_k) = \frac{\sum_{i=1}^{N} p(k \mid \boldsymbol{s}_i)}{N} \tag{4.33}$$

4. Evaluation the log likelihood

$$\ln p(\boldsymbol{s} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, w) = \sum_{i=1}^{N} \ln\left(\sum_{k=1}^{K} w_k N(\boldsymbol{s}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \tag{4.34}$$

then check for convergence condition of either the parameters or the log likelihood. If the convergence condition is not satisfied, go to step 2.

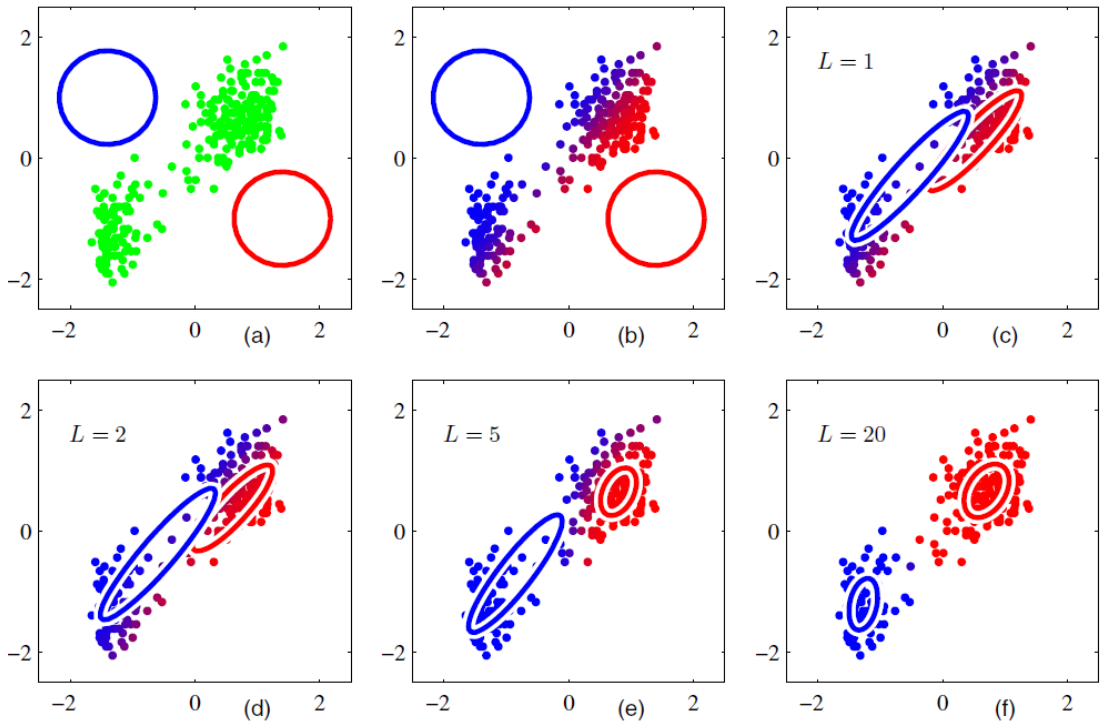


Figure 3. 12: Illustration of EM algorithm for a mixture of two Gaussians [44]. (a) shows data points in green with the initial guess of parameters of two Gaussians. (b) shows the initial E step, the color of each data point is proposional to the posterior probability of having been generated by each Gaussian. (c) shows the results after first M step. The parameters of two Gaussian change based on maximum likelihood estimators. (d), (e) and (f) show EM results at different iteration, and it finally converges when convergence criterion is reached.

EM algorithm is a popular method for finding maximum likelihood solutions for models with latent variables [44]. It finds the maximum a posteriori (MAP) estimates of parameters in statistical models iteratively. Figure 3.12 illustrates how EM estimates the parameters of a mixture of two Gaussian models.

3.4.3 Model Selection

Now, the remaining question is how to select the number of Gaussian K for GMM. This is a nontrival problem, because we do not have any prior knowledge about our motion data. Here, we use Bayesian information criterion (BIC) as our model seletion technique to automatically select the bset number of K for our model. BIC is a widely used model selection criterion if the true model is among the finite set of candidates. The reason to use BIC is because BIC has better performance in penalizing complexity of model than other model selection methods, such as AIC and cross validation [45]. This is very important for our work because the dimensions of our motion data is far more larger than the number of training data, which means the statistical model has a high risk to overfit the data.

$$BIC = \frac{N}{\sigma_\varepsilon^2}\left[\overline{err} + (\log N)\frac{d}{N}\sigma_\varepsilon^2\right] \tag{3.35}$$

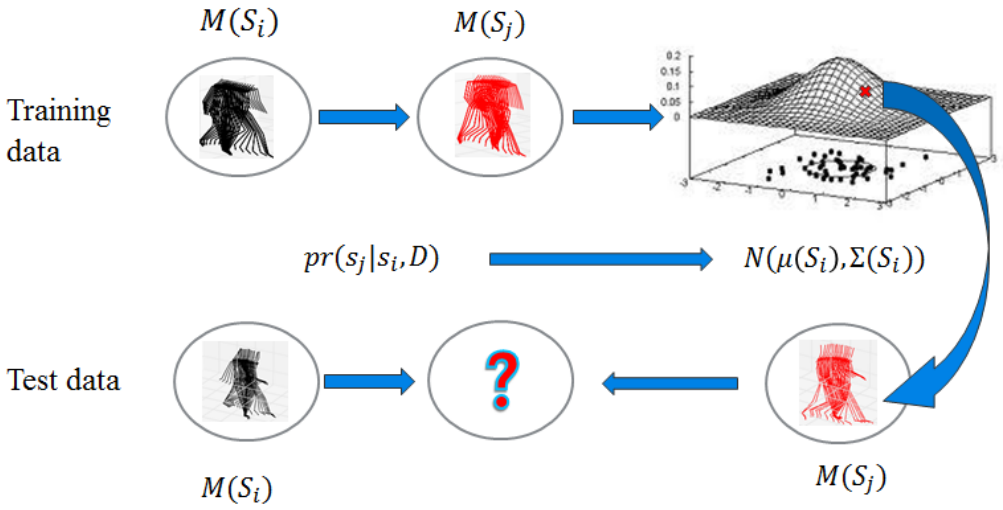


Figure 3. 13: Illustration of transition model based on Gaussian processes. $pr(s_j|s_i, D)$ models the distribution of transition from motion primitive $M(s_i)$ to $M(s_j)$ based on training data set *D*.

$$\overline{err} = -\frac{2}{N}\sum_{i=1}^{N}\ln p(\boldsymbol{s}_i \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, w) = -\frac{2}{N}\sum_{i=1}^{N}\ln(\sum_{k=1}^{K} w_k N(\boldsymbol{s}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)) \tag{3.36}$$

where *N* is the number of samples, *d* is the number of parameters needed to be estimated for the model, σ_ε^2 is variance of noise, which usually estimated by the mean squared error of a low-bias model.

3.5 Transition Model Construction

Each edge *(i,j)* in motion graph G represents a possible transition from motion primitive s_i to s_j. However, directly concatenating two motion segments from motion primitive s_i and s_j could suffer discontinuities at transition points. So how to seamlessly concatenate two motion segments is an important problem for data driven motion synthesis. As we introduced in chapter 2, the widely used solutions for this problem are motion interpolation and blending [21, 22, 46]. Kovar et al. [14] selected a good transition point by taking the points with minimum frame distance between last frame from previous motion and first frame from next motion. However, their approach is not practical in our case because they used finite training samples to create new motions, but our morphable models can theoretically generate infinite variant motions, which makes this approach intracable.

Min et al. [15] modeled the possible transition as a regression problem, and used Gaussian processes to learn the probality distribution function $pr(s_j|s_i)$ for predicting the morphable parameter at one node s_j from another node s_i. This approach is applied in our work. Figure 3.13 illustrates the idea of transition model, given the training data D, which is a set of pairs of motion segments from motion primitive $M(s_i)$ to $M(s_j)$, GP learns a prior probability $pr(s_j|s_i, D)$, which is a multidimensional Gaussian distribution $N(\mu(s_i), \Sigma(s_i))$. Its mean and

variance are funcitons of input motion segment s_i. For a new morphable parameter s_i at node i, the model will generate a new sample from $N(\mu(s_i), \Sigma(s_i))$, and a high value for $pr(s_j|s_i, D)$ means a good transition.

3.5.1 Gaussian Processes

Gaussian processes (GP) are a powerful tool for regression in high-dimensional space [48]. GP and its invariant GPLVM [47] have recetly been widely applied to many problems in computer animation, for instance, nonlinear dimensionality reduction for human pose [36], motion interpolation [46], motion editing [49], and motion synthesis [50, 51]. Gaussian process demonstrates its ability to model nonlinear properties of transition functions efficiently [15] and it requires only very few parameters to be tune manually [52].

A GP is defined as a probability distribution over functions $y(x)$ such that the set of values of $y(\boldsymbol{x})$ evaluated at an arbitrary set of points $(\boldsymbol{x}_1, \dots, \boldsymbol{x}_N)$ jointly have a Gaussian distribution. A standard linear regression with basis functions can be written as:

$$\boldsymbol{y} = \boldsymbol{w}^T \Phi(\boldsymbol{x}) \tag{3.37}$$

where $\boldsymbol{x}$ is the input vector, $\Phi(\boldsymbol{x})$ is a mapping from $\boldsymbol{x}$ to basis functions and $\boldsymbol{w}$ is a column vector of weights.

Now consider a distribution over $\boldsymbol{w}$, rather than take it as a fixed vector, e.g: an isotropic Gaussian distribution

$$p(\boldsymbol{w}) = N(\boldsymbol{w}|0, \alpha^{-1}\boldsymbol{I}) \tag{3.38}$$

Then for every specific $\boldsymbol{x}_n$, $\boldsymbol{y}_n = \boldsymbol{w}^T \Phi(\boldsymbol{x}_n)$ is not a fixed value, but a random variable. Its expectation and covariance are:

$$E[\boldsymbol{y}] = E[\boldsymbol{w}^T \Phi(\boldsymbol{x}_n)] = \Phi(\boldsymbol{x}_n) E[\boldsymbol{w}] = 0 \tag{3.39}$$

$$Cov[\boldsymbol{y}] = E[(\boldsymbol{y} - E[\boldsymbol{y}])^2] = E[\Phi^T \boldsymbol{w}\boldsymbol{w}^T \Phi] = \Phi^T E[\boldsymbol{w}\boldsymbol{w}^T]\Phi = \frac{1}{\alpha}\Phi^T\Phi = \boldsymbol{K} \tag{3.40}$$

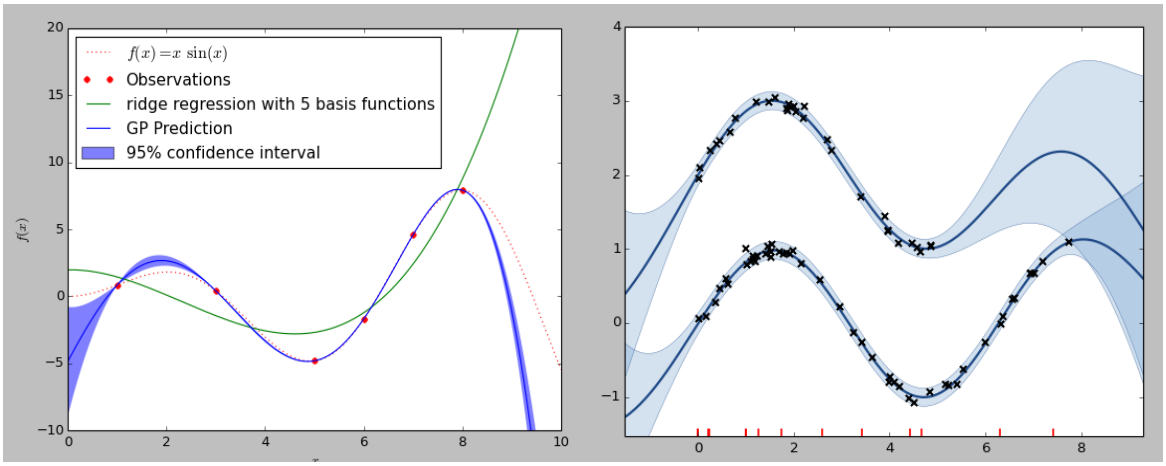
where the kernel $\boldsymbol{K}$ is a Gram matrix, and each element $K_{mn} = k(x_m, x_n) = \frac{1}{\alpha} < \Phi(x_m), \Phi(x_n) >$.

Here, we choose Gaussian radial basis function as our kernel function,:

$$k(\boldsymbol{x}, \boldsymbol{x}') = exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{x}')^T(\boldsymbol{x}-\boldsymbol{x}')}{2\sigma^2}\right) \tag{3.41}$$

In our transition model, the training set is $D = \{(\boldsymbol{y}_n, \boldsymbol{t}_n)|n = 1, \dots, N\}$, where $\boldsymbol{y}_n$ is a sample from node s_i, and t_n is a sample from node s_j. Our goal is to learn a regression funciotn $f(\cdot)$ that finds the predictive output $\boldsymbol{t}_{N+1}$ using a testing input $\boldsymbol{y}_{N+1}$. In order to achieve that, we need to take account of the noise on observed target values $\boldsymbol{t}_n$

$$\boldsymbol{t}_n = \boldsymbol{y}_n + \boldsymbol{\epsilon}_n \tag{3.42}$$



(a) (b)

Figure 3. 14: (a) Comparison of Gaussian process and ridge regression fitting a set of 2D points. Six samples (red dots) drawn from *f(x)* plotted as red dot line. The blue area shows 95% confidence interval of the distribution from Gaussian process, and the solid curve is the mean of functions. The green line is the function estimated from ridge regression. We can see that the result from GP is more closer to estimated function from ridge regression. (b) shows using GP to fit two sets of data. From the results we can see the area with more training samples has small variance in the distribution of GP, which indicates high confidence in prediction.

where $\boldsymbol{\epsilon}_n$ is a random noise variable whose value is chosen independently from normal distribution $N(0, \beta^{-1})$ for each observation.

$$p(\boldsymbol{t}_n|\boldsymbol{y}_n) = N(\boldsymbol{t}_n|\boldsymbol{y}_n, \beta^{-1}) \tag{3.43}$$

Because the noise is independent for each data point, the joint distribution of target values $\boldsymbol{t} = (t_1, \dots, t_n)^T$ conditioned on the value of $\boldsymbol{y} = (y_1, \dots, y_n)^T$ is an isotropic Gaussian:

$$p(\boldsymbol{t}|\boldsymbol{y}) = N(\boldsymbol{t}|\boldsymbol{y}, \beta^{-1} I_N) \tag{3.44}$$

And we know the prior distribution of $p(\boldsymbol{y})$:

$$p(\boldsymbol{y}) = N(\boldsymbol{y}|\boldsymbol{0}, \boldsymbol{K}) \tag{3.45}$$

So the marginal distribution of $\boldsymbol{t}$ is:

$$p(\boldsymbol{t}) = \int p(\boldsymbol{t}|\boldsymbol{y}) p(\boldsymbol{y}) d\boldsymbol{y} = \int N(\boldsymbol{t}|\boldsymbol{y}, \beta^{-1}\boldsymbol{I}_N)\, N(\boldsymbol{y}|\boldsymbol{0}, \boldsymbol{K}) d\boldsymbol{y} = N(t|\boldsymbol{0}, \boldsymbol{K} + \beta^{-1}\boldsymbol{I}_N) \tag{3.46}$$

Given a new test input y_{N+1}, the marginal distribution of $\boldsymbol{t}_{N+1} = (t_1, \dots, t_{N+1})^T$ is:

$$p(\boldsymbol{t}, t_{N+1}|\boldsymbol{y}, y_{N+1}) = N(\boldsymbol{0}, \begin{bmatrix} K(y_{N+1}, y_{N+1}) & K(y_{N+1}, \boldsymbol{y}) \\ K(\boldsymbol{y}, y_{N+1}) & K(\boldsymbol{y}, \boldsymbol{y}) + \beta^{-1} I_N \end{bmatrix}) \tag{3.47}$$

where vector $K(\boldsymbol{y}, y_{N+1})$ has element $k(y_n, y_{N+1})$ for $n = 1, \dots, N$

Since $\boldsymbol{t} = (t_1, \dots, t_n)^T$ is already known, this Gaussian can be conditioned on $\boldsymbol{t}$ to obtain the regression model for y_{N+1}, which is also a Gaussian distribution with mean μ and covariance Σ (for detailed derivation, see [34], chapter 2).

$$p(t_{N+1}|\boldsymbol{y}, \boldsymbol{t}, y_{N+1}) = p(t_{N+1}|y_{N+1}, D) = N(\mu(y_{N+1}), \Sigma(y_{N+1})) \tag{3.48}$$

$$\mu(y_{N+1}) = K(\boldsymbol{y}, y_{N+1})[K(\boldsymbol{y}, \boldsymbol{y}) + \beta^{-1} I_N]^{-1}\boldsymbol{t} \tag{3.49}$$

$$\Sigma(y_{N+1}) = K(y_{N+1}, y_{N+1}) - K(\boldsymbol{y}, y_{N+1})[K(\boldsymbol{y}, \boldsymbol{y}) + \beta^{-1} I_N]^{-1} K(y_{N+1}, \boldsymbol{y}) \tag{3.50}$$

If we replace t_{N+1}, y_{N+1} with of morphable parameters $\boldsymbol{s_i}, \boldsymbol{s_j}$, then the final form of our transition model is:

$$p(\boldsymbol{s}_j | \boldsymbol{s}_i, D) = N(\mu(\boldsymbol{s}_i), \Sigma(\boldsymbol{s}_i)) \tag{3.51}$$

where both mean $\mu(\boldsymbol{s}_i)$ and covariance $\Sigma(\boldsymbol{s}_i)$ are functions of the current morphable primitive parameters $\boldsymbol{s}_i$.

4 Implementation and Analysis

In this chapter, the implementation details of each aforementioned step in last chapter are presented, and a series of experimental results, which compares the performance of motion graphs++ with our modified approach on both markerless motion capture data and Vicon motion capture data. The structure of this chapter is organized as follows: Section 4.1 gives a description of the motion capture dataset used in our experiment. Section 4.2 presents the results of automatic motion decomposition. Section 4.3 compares the performance of weighted frame distance based motion alignment with unweighted approach. Section 4.4 compares the performance of weighted PCA with standard PCA in morphable model construction. Section 4.5 evaluates the synthesized motions from our morphable model.

4.1 Experimental Data

Our testing scenario is walking in a free space. We have two sets of motion capture data, one is from our low-cost markerless motion capture system, another is from a high-cost Vicon motion capture system. From visualization, the quality of captured motion from Vicon system is much better than markerless system, which can be regarded as noise free data. The details of the data for our experiments can be found in table 4.1. All experiments are run on a machine with 2.3 GHz Intel CPU, 8 GB RAM. And all methods mentioned in last chapter are implemented in Python programming language.

	Markerless system	Vicon system
Number of samples	145 (4-meter walking)	2 (4 km and 6 km walking)
Number of frames	23467	30993
Frame rate	50 /s	120 /s

Table 4. 1: Details of motion capture data

4.2 Automatic Motion Decomposition

Because we use keyframe based motion decomposition (cf. section 3.1.2), so the task to automatically decompose captured motion data into structurally similar segments is actually to find pre-defined keyframes from frame sequence automatically.. In this section, we apply feature based keyframe detection and learning based keyframe detection on markerless mocap data and Vicon mocap data. And for the walking motion data from our markerless mocap system, three contact transition keyframes are defined: right stance, left stance and double stance (cf. figure 3.4 (b))...

4.2.1 Feature based KeyFrame Detection

The idea of feature based approach is to select one or several dominant features to extract keyframes. So the first step is feature selection. In our experiment, the keyframes are defined by contact points, which are feet on the ground, such as left foot down, right foot up, so one

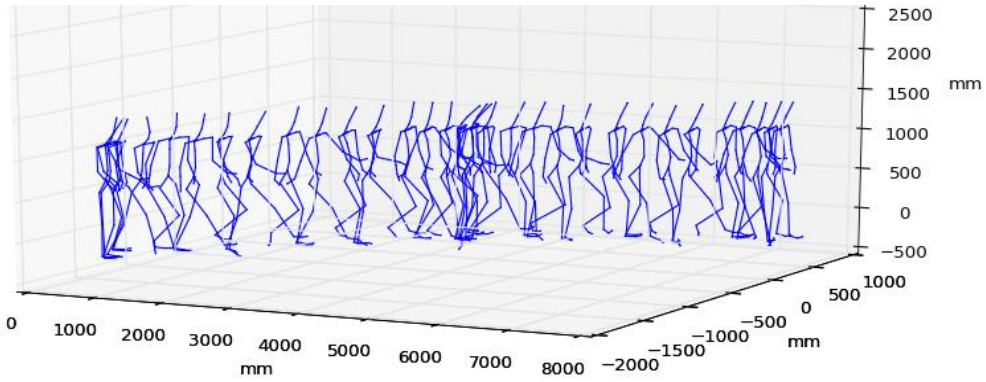


Figure 4. 1: An example of recorded walking data.

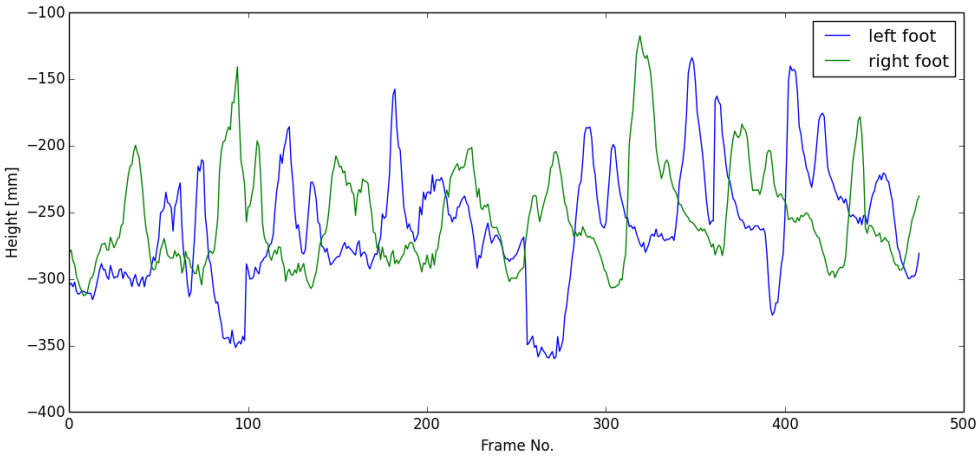


Figure 4. 2: Height of two feet joints of walking example in figure 4.2.

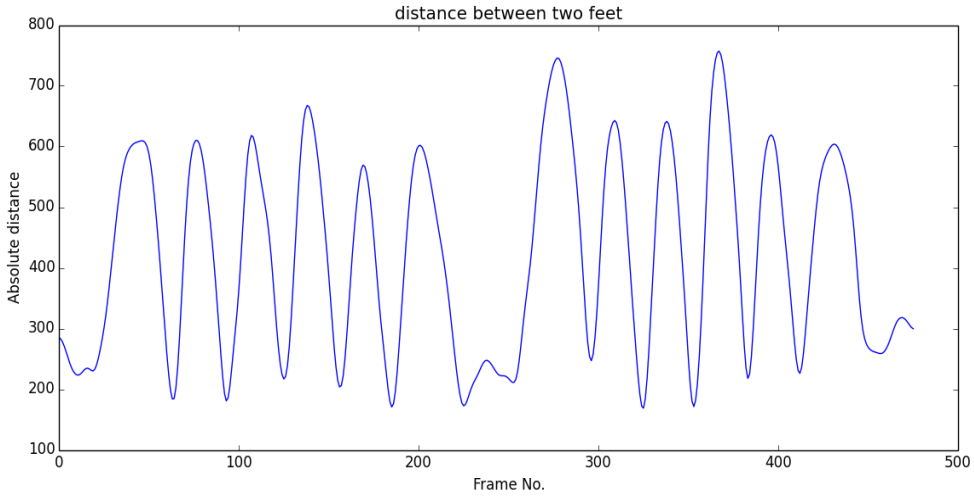


Figure 4. 3: Smoothed distance between two feet joints of walking example in figure 4.1.

feature should be enough to detect these keyframes. As there is no extra sensors to detect these points in our experiment, we use visual features from animation. The most simple feature which can be used as an indicator of key frames would be the height of feet. If the

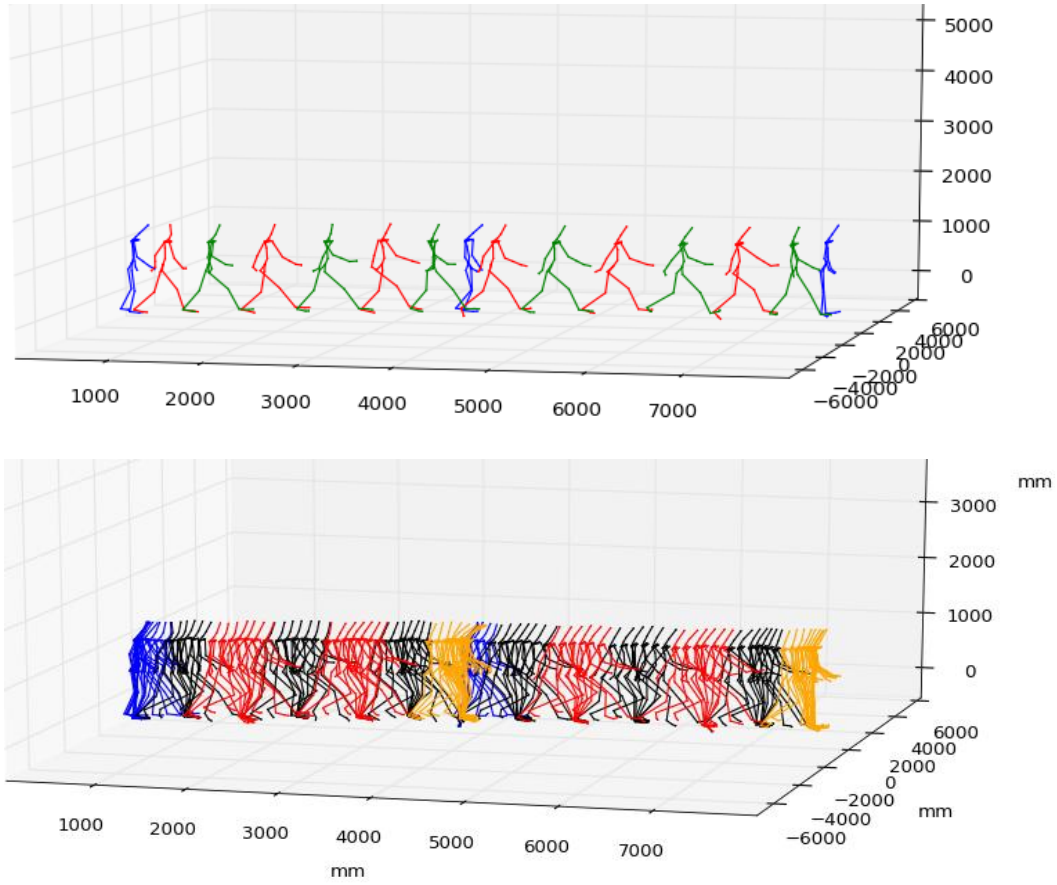


Figure 4. 4: (top) extracted keyframes using feature based approach; (bottom) motion decomposition based on extracted keyframes.

captured motion is well calibrated and very accurate, a zero height of foot means it is on the ground. However, real life is not so easy. Figure 4.1 gives an example of capture walking motion data, and figure 4.2 plots its corresponding height of feet. We can see that the data is quite noisy, and the values of local minimum for both curve vary a lot. This indicates that the feet position in each keyframe is not on the ground. So the height of feet is not a good choice to detect keyframes in our work. From observation, we find out that for our mocap data from markerless system, the contact transition between left stance and right stance usually happen when the distance between two feet reach maxima. However, it is a little bit tricky for double stance. Because double stance is quite similar to frames where two feet are close to each other (local minima in figure 4.3). We assume that if double stance is reached, the actor will stay in this phase for a while (at least a step size in our experiment), then double stance becomes low flat area rather than a low peak in distance curve (cf. figure 4.3). Based on these observation, the distance between two feet is a quite robust feature to detect keyframes for our data. Figure 4.3 plots the corresponding feet distance of walking example in figure 4.1.

The next step is to find pre-defined keyframes from frame sequence using selected feature. In our case, the left stances and right stances are indicated by peaks in figure 4.3. Double stances are indicated by a low value flat area, and we take the local minimum as the index of double

stance. In order to reduce the influence of noise and improve the extraction result, there are two parameters need to be set. One is step size over frames. In our experiment, we only consider the normal walking, which the actor does not vary his step size two much, and one key frame for one step. However, in feet distance measure, sometimes, there are two peaks where are very close to each other. We take this kind of cases as artifacts caused by capturing noise, and eliminate them by choosing a proper step size. Basically, we take the frame which has maximum feet distance within one step size as left stance or right stance. Another one is a threshold to tell which feet distance is a low value. If all the feet distance within one step size are below this threshold, then we believe that there will be a double stance, and we take the one with minimal distance as keyframe. It is better to set this threshold automatically, because different motion capture system or different actors have different scales. In our implementation, we take the average of maximum and minimum of feet distance. Figure 4.4 illustrates the result of applying feature based approach for the walking example in figure 4.1.

4.2.2 Learning based Keyframe Detection

For some simple motions, e.g. walking, feature based keyframe extraction can work well (cf. figure 4.4). However, feature based approach requires a good prior knowledge about the motion. And the motion usually should be able to be decomposed just by contact state transition keyframes. This is because it is always possible to detect contact state transition by some features, e.g. whether the contact points are reached or not. But for complex motions, e.g. the actions in assembly line, they could have some keyframes with highest visual content change, which could not be possible quantitative measured by several features, or requires a huge amount of features to describe them. For this kind of keyframes, we propose to use a learning based approach, which is more general and robust than feature based approach.

From our recorded motion capture data, we select 10 samples, and manually annotate all the key frames in each of selected samples. So for each kind of keyframe, we have a set of example frames. Then the system will automatically search the most similar frames to label frames in the rest of samples. Here, we use the frame distance metric defined in section 3.2.1, equation 3.1. And all the points in skeleton are treated equally, so $w_i = 1, i = 1, \dots, N$. For each test frame, it will be compared with all examples in one keyframe, and the minimum distance is the distance to this keyframe. Figure 4.5 plots frame distance of each frame in a test sample (figure 4.1) to three defined keyframes. In figure 4.5, the red and green curves are distance to right and left stance respectively. The value of these two curves periodically change and each local minimum is a good candidate for keyframe. The blue curve is the distance to double stance, which also changes periodically, but with a small magnitude. This is because when two feet are very close to each other when walking, the pose is similar to double stance. That's why we have lots of local minima in blue curve. So for learning based keyframe extraction, a good threshold is needed to only accept local minima below it. The simplest approach is to take an empirically determined threshold, which is applied in our work. Figure 4.6 illustrates the result of extracted keyframes of the test sample in figure 4.1 and the corresponding decomposed segments. For our test sample, both keyframe extraction approaches work well and give acceptable segmentation results. The evaluation of performance on our whole data set will be presented in the next section.

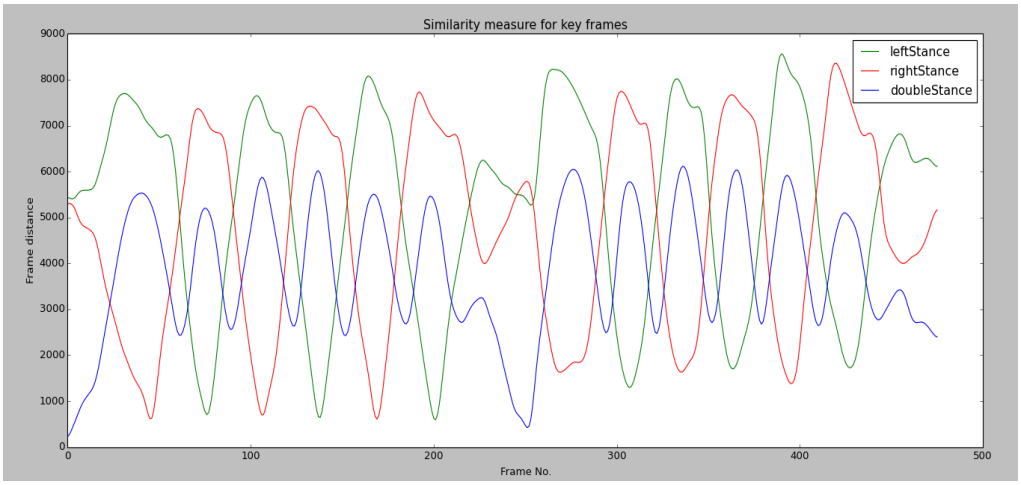


Figure 4. 5: Frame distance between frames in example motion in figure 4.1 and pre-defined keyframes.

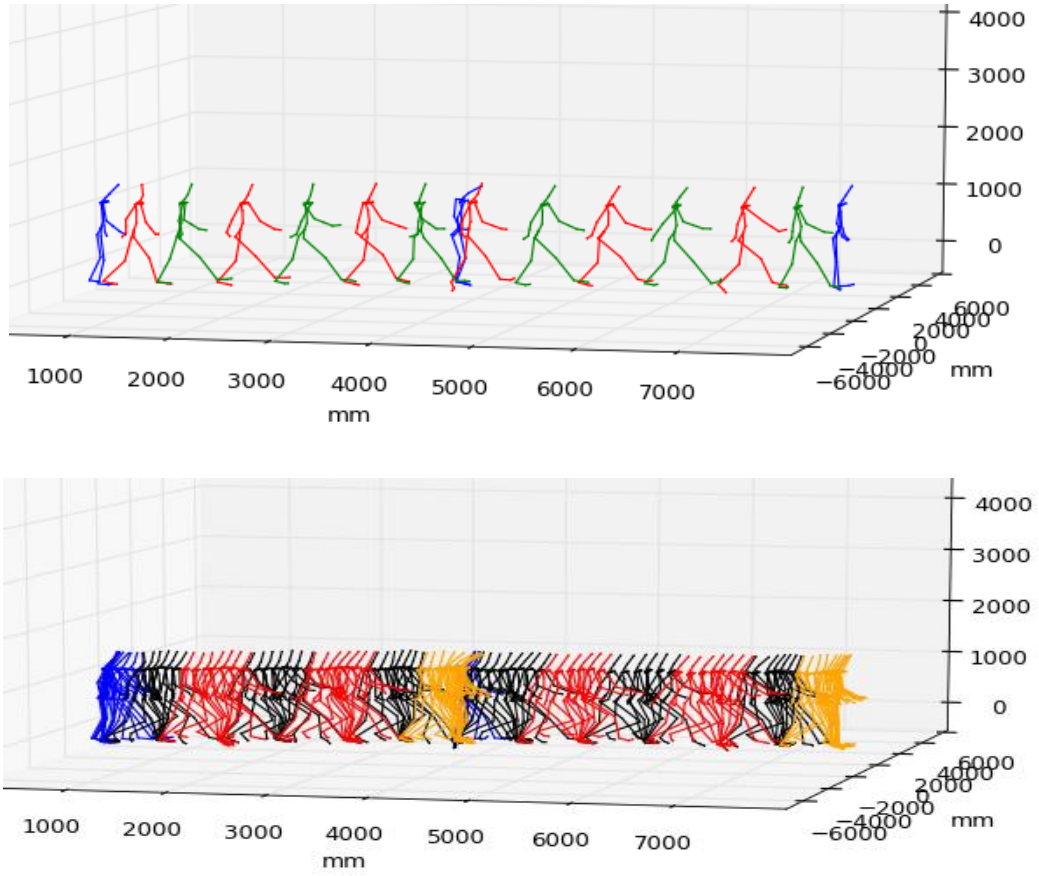


Figure 4. 6: Keyframe extraction and motion decomposition of example motion in figure 4.1. (top) extracted keyframes using learning based approach; (bottom) motion decomposition based on extracted keyframes.

4.2.3 Evaluation Methods

In this section, some evaluation criteria for evaluating the performance of keyframe detection are given. There are many existing methods to evaluate keyframe extraction algorithms[49]. In our work, we take some evaluation measures from information retrieval [50]. Table 4.2 gives the confusion matrix for our keyframe extraction problem.

	key frame	**not key frame**
extracted	true positives (tp)	false positives (fp)
not extracted	false negatives (fn)	true negatives (tn)

Table 4. 2: The confusion matrix for keyframe extraction

- true positive (tp): correctly extracted keyframe
- false negative (fn): keyframe, but not extracted
- false positive (fp): extracted, but not keyframe
- True negative (tn): not keyframe, and not extracted

In our experiment, we use manually labeled motion samples as training data. The correctly extracted keyframes do not need to have exactly the same index in frame sequence as our label result. Because two adjacent frames are very similar to each other in frame sequence and there is no guarantee that the results from manual labeling is exactly accurate. So we regard the frame which locals in the narrow neighbor of key frames as a keyframe as well. The following measures will be measured:

Precision (*P*): the fraction of extracted keyframes are correct:

$$P = \frac{tp}{tp+fp} \tag{4.1}$$

Extraction Rate (*R*): the fraction of keyframes are extracted correctly:

$$R = \frac{tp}{tp+fn} \tag{4.2}$$

Distortion (*d*): the average index distance from correct automatic extraction to manually labeling, which measure the variance of correctly extracted result.

$$d = \frac{\sum_{i=1}^{N} |I_a^i - I_m^i|}{N} \tag{4.3}$$

where I_a^i is *i*-th correctly automatically extracted keyframe, I_m^i is *i*-th corresponding manually labeled keyframe, *N* is the total number of manually labeled keyframes.

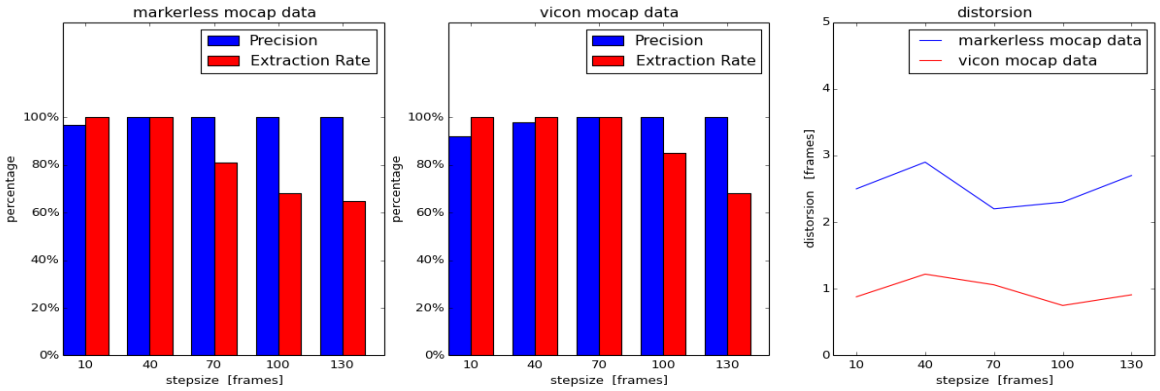


Figure 4. 7: Precision, extraction rate and distorsion results of left stance and right stance using feature based keyframe extraction for both markerless mocap data and vicon mocap data.

4.2.4 Experimental Results

First, we evaluate the performance of feature based keyframe detection with different stepsize. Figure 4.7 shows the results of measures defined in last section on both high quality Vicon mocap data and low quality markerless mocap data. Here, only two kinds of keyframes are considered: left stance and right stance, due to Vicon mocap data does not contain double stance. From the results, we can find that feature based approach works well on our test data (high quality and low quality both) if an appropriate stepsize is chosen. In addition, high quality Vicon data has much less distortion than low quality markerless data, which means that the automatic extracted results from Vicon data is more close to manually extracted. This is because Vicon data contains less noise, so the feature we select is more stable..

Next, we compares the performance of learning based keyframe detection with feature based approach with optimal stepsize on markerless mocap data. Figure 4.8 shows that in our walking test scenario, for the keyframes: right stance and left stance, both methods work well and feature based approach is a little bit better in precision and extraction rate. But for double stance, the results are worse than others and learning based approach performs better than feature based. This is because walking is a quite simple structure motion, and the feature used in our work is quite obvious for left and right stance (cf. figure 4.3). However, this is not the case for double stance. the feet distance itself is not a good feature for double stance, as we discussed before. That's why we need some auxiliary information to detect it, and the results are not as good as other keyframes. But this is not a problem for learning based approach, because it uses a more general approach and no prior knowledge about motion is required. Although double stance has a higher risk to be confused with frames where two feet meet each other in markerless mocap data (cf. figure 4.5), the results do not attenuate much. Another issue is time complexity, feature based approach is much faster than learning based approach in our experiment. To extract keyframes from all our test motions, it takes about 10 minutes for feature based approach. However, for learning based approach, it requires several hours.

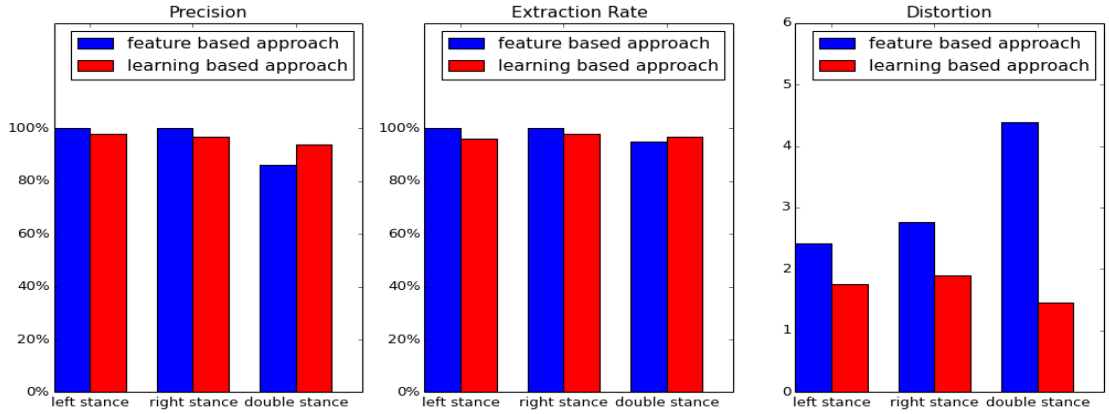


Figure 4. 8: Precision, extraction rate and distorsion results of feature based approach and learning based approach on markerless mocap data.

In conclusion, for some simple motions, such as walking, running and so on, which can be decomposed simply by contact transition keyframes, feature based approach should be enough for automatic keyframe extraction if the contact information can be described by one or two dominant features, especially for high quality motion capture data. However, for some complex motions, such as punching, working with screwdriver, which cannot be well segmented just by contact transition keyframes, learning based approach offers a more general and robust solution. For instance, some highest visual content change keyframes are even hard to define by language, the simple and effective solution is to store some examples, and let system to automatically find most similar frames in test data.

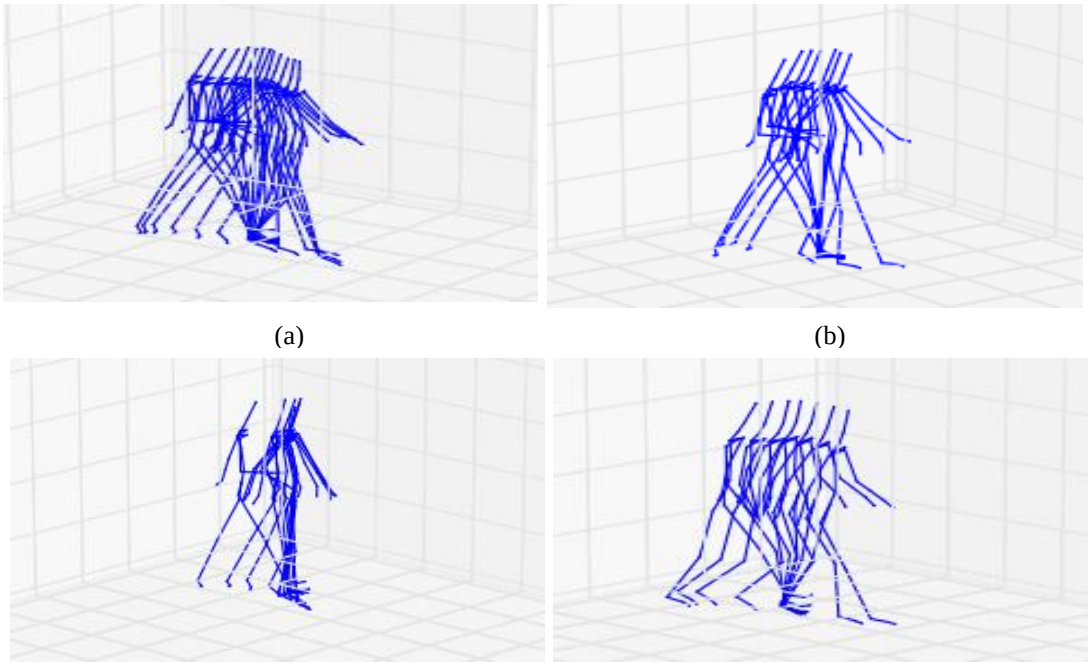
4.3 Motion Alignment Evaluation

After motion decomposition, although motion segments are structurally similar, they could be in different position and rotation, and have different length. The motivation of motion alignment is to make the motion segments within the same motion primitive similar to each other. Good quality of aligned motion is the key to success for our approach. In this section, we apply the motion alignment method described in section 3.2 to markerless mocap data, and compare the results from weighted frame distance with unweighted frame distance.

4.3.1 Evaluation Methods

The results of motion alignment is evaluated by two criteria, path difference between reference motion and aligned motion and motion velocity [51]. Path of aligned motion indicates how the dynamic time warping works. If each frame in test motion is translated and rotated correctly based on reference motion, the path of aligned motion should be very similar to reference motion. The path difference is measured by Eq. (4.4):

$$D(M_R, M_T) = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(x_{R_i} - x_{T_i})^2 + (y_{R_i} - y_{T_i})^2} \tag{4.4}$$



(a) (b)

(c) (d)

Figure 4. 9: Comparison of motion alignment with weighted frame distance and unweighted frame distance. (a) test motion, one step walking with a small angle rotation; (b) reference motion, one step straight line walking; (c) aligned motion using unweighted frame distance; (d) aligned motion using weighted frame distance.

where M_R denotes the reference motion, M_T denotes the test motion, (x_{R_i}, y_{R_i}) is the 2D root position of i-th frame in M_R, (x_{T_i}, y_{T_i}) is the 2D root position of i-th frame in M_T, and N is the total number of frames.

The natural-looking is measured by the velocity of aligned motion. From the observation, we noticed that the main reason for visual discontinuity of aligned motions is sharp changes in motion speed. So we use the variance of velocity of aligned motion as a measure of naturalness.

Figure 4.7 gives examples of results from both motion alignment using weighted frame distance and unweighted frame distance. (a) is a test motion which is going to be aligned to the reference motion in (b), which is one step of straight line walking. Here, we take a "hard" case that the test motion is not actually walking in a straight line. It has a rotation in the middle, which can be seen in figure 4.8. And it has some jitters in its hands and feet. The expected result should be as similar as reference motion and look natural as well. (c) and (d) give the results using different approaches.

Figure 4.8 (c) and (d) show the root path of different approaches. Weighted approach gives us a smoother root path than unweighted approach, which is more similar to reference motion. In unweighted result, there are some sharp changes in the root path, which indicate that they are not transform correctly, probably influenced by the noise in arms or legs. However, weighted approach is more robust against these noise, because the joints which are more likely to be

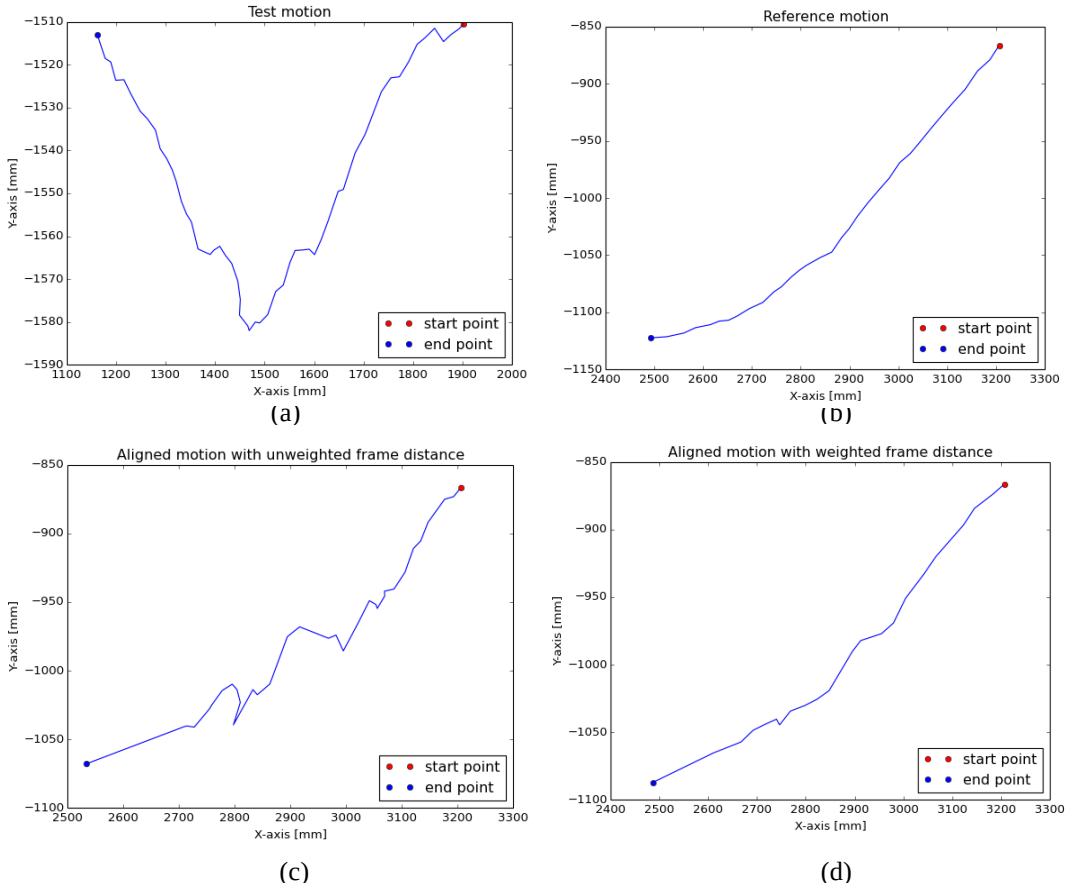


Figure 4. 10: A comparison of path of four walking examples in figure 4.9. (a) test motion; (b) reference motion; (c) aligned motion using unweighted frame distance; (d) aligned motion using weighted frame distance.

noisy have less influence to weighted frame similarity measure. Figure 4.9 (c) and (d) demonstrate that weighted approach also has a better performance in velocity of motion. Although the velocity of aligned motion using weighted frame distance measure has larger variance than the raw data, because the reference motion has less frames than test motion, it is much better than unweighted result. We can see that there is a big jump in unweighted result, which makes the result look unnatural.

	Data	Weighted frame distance	Unweighted frame distance
Average path difference[m]	Markerless data	0.218	0.496
	Vicon data	0.015	0.061
Average variance of velocity	Markerless data	140.329	686.414
	Vicon data	0.0009	0.0074

Table 4. 3: Evaluation results of motion alignment of motion primitive $M(s_3)$ (cf. figure 3.2).

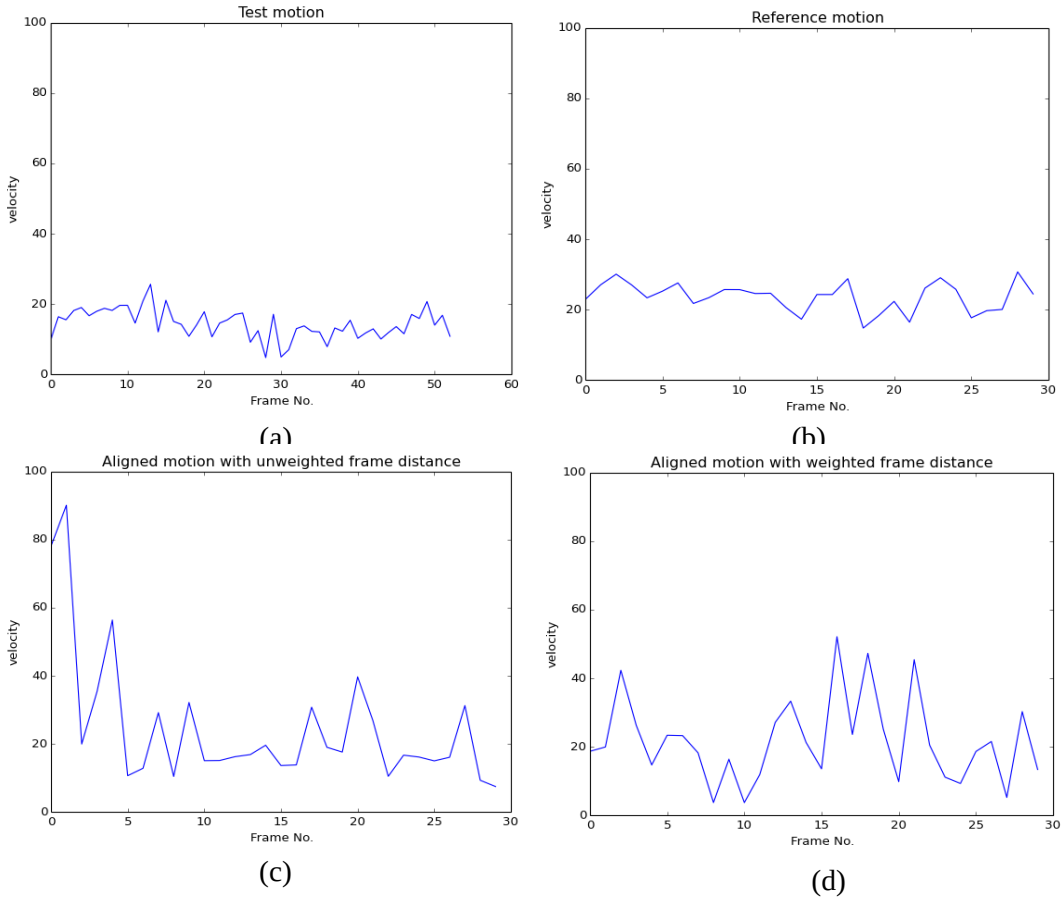


Figure 4. 11: A comparison of velocities of four walking examples in figure 4.9. (a) test motion; (b) reference motion; (c) aligned motion using unweighted frame distance; (d) aligned motion using weighted frame distance.

4.3.2 Experimental Results

Table 4.3 gives the results of average path distance from aligned motion to reference motion and average variance of velocity of motion segments in primitive $M(s_3)$, left first walking. The walking path of aligned motions using weighted frame distance measure are more closer to reference motion and their velocity have less average variance. And the results from Vicon data are much better than markerless data. Because Vicon data contains less noise, and it is exactly straight line walking, which is much easier for alignment. The results convince our assumption that embed the importance of different joint in our hierarchical motion data structure could reduce the influence of noise.

4.4 Dimension Reduction Evaluation

In section 3.3 we explain our motivation to use a weighted PCA rather than standard PCA, and theoretically show how weighted PCA could help our work. In this section, weighted PCA and standard PCA are applied to preprocessed motion data, and their performance in dimension reduction and statistical modeling are compared.

The assignment of weights is the same as what we did in motion alignment, which is inverse linearly propositional to the depth of the joint in our hierarchical skeleton structure (cf. figure 2.5). Figure 5.12 compares performance of weighted PCA and standard PCA on markerless mocap data. It demonstrates that under same number of eigenvectors, weighted PCA outperforms standard PCA in both minimizing weighted squared error between original data and projected data and keeping most variance of original data. And the less eigenvectors they use, the better results for weighted PCA than standard PCA. Figure 5.13 presents the same test on vicon mocap data. However, the performance of weighted PCA is almost the same as standard PCA. This is because the motion samples from vicon mocap data are nearly noise free and very similar to each other for samples within the same motion primitive. So the value of each element in eigenvectors of vicon mocap data is very small. Most of them are nearly zero. That's why rescaling eigenvectors does not influence the result much.

Decide the number of eigenvectors which the motion data is going to project is a trade-off between the accuracy and the compactness of morphable model. In our experiment, the number of eigenvectors is automatically decided by keeping 99% of the original variations. Figure 5.12 (b) and figure 5.13 (b) illustrate that weighted PCA requires less eigenvectors than standard PCA for keeping the same amount of variations of original data. Another observation is vicon mocap data requires less eigenvectors than markerless mocap data for keeping the same amount of variations, which also indicates the vicon mocap data is less noisy and well aligned.

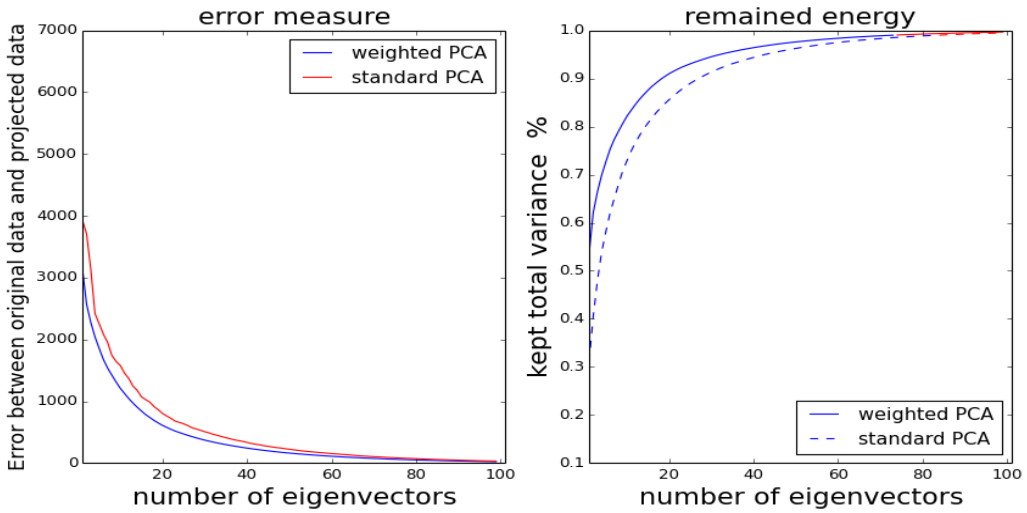
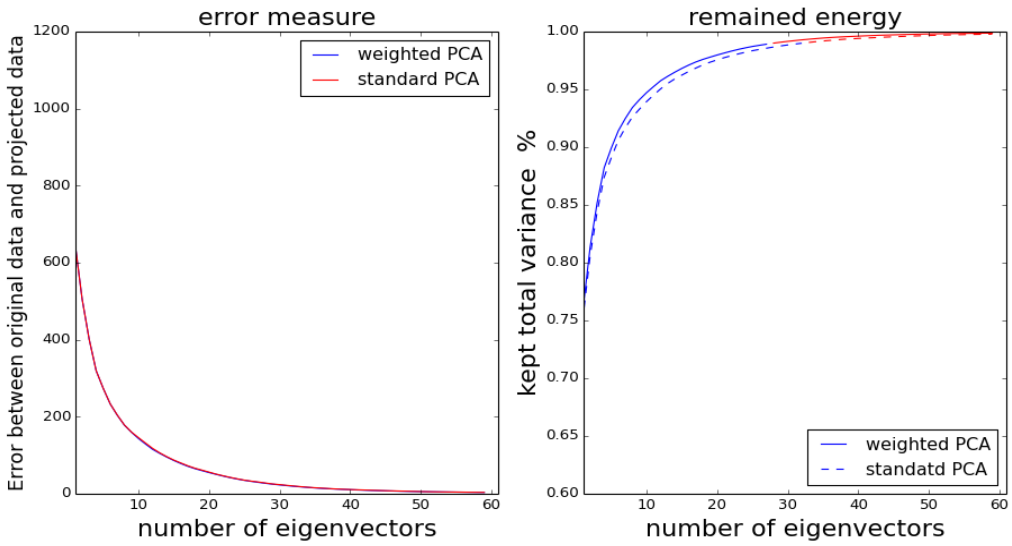


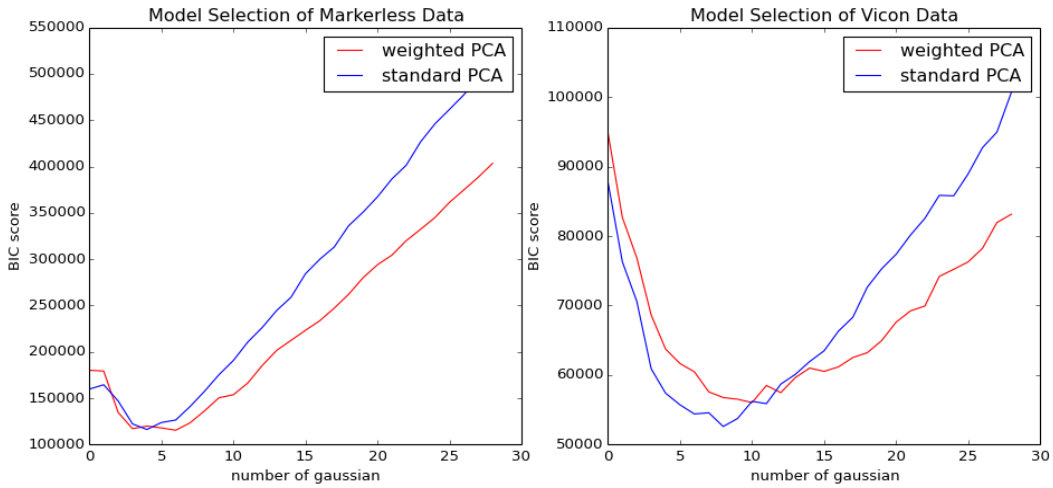
Figure 4. 12: Comparison of performance between weighted PCA and unweighted PCA on markerless mocap data. (a) weighted squared error between original data and projected data; (b) ratio of summative variance with different number of eigenvectors. The blue part of curves denotes the summative variance is below 99%, and the red part denotes the summative variance is above 99%.



(a) (b)

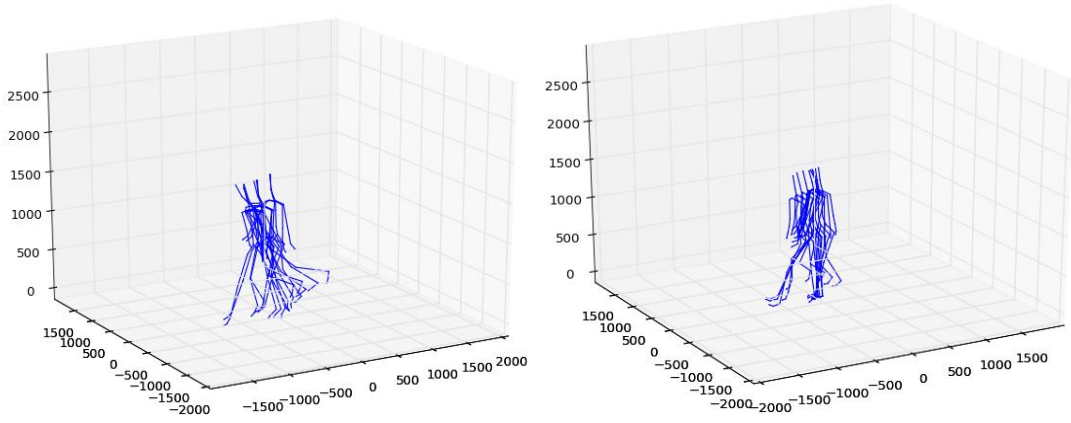
Figure 4. 13: Comparison of performance between weighted PCA and unweighted PCA on Vicon mocap data. (a) weighted squared error between original data and projected data; (b) ratio of summative variance with different number of eigenvectors. The blue part of curves denotes the summative variance is below 99%, and the red part denotes the summative variance is above 99%.

The statistical model of motion primitive can also benefit from weighted PCA. In our experiment, the features of training data are far more than number of training samples, which makes the model have a high risk to overfit training data. Although we use BIC to select models, which will penalize complex models heavily, there is still no guarantee for avoiding overfitting. Figure 5.14 plots the BIC score of Gaussian Mixture Model with different number of Gaussians for markerless data and vicon data by using weighted PCA and standard PCA.



(a) (b)

Figure 4. 14: BIC score of Gaussian Mixture Model with different number of Gaussians. (a) markerless mocap data, (b) Vicon mocap data.



(a) (b)

Figure 4. 15: Comparison of sampled motions from motion primitive $M(s_3)$ (cf. figure 4.2). (a) sample based on standard PCA; (b) sample based on weighted PCA.

We conclude that for our hierarchical motion data structure, weighted PCA can to reduce the influence of noise in dimension reduction. Experiments of generating new motion samples from statistical model demonstrate that weighted PCA can reduce artifacts and jitters in synthesized motion based on noisy data. Figure 5.15 gives an example of two sampled new motion segments from motion primitive $M(s_3)$ based on markerless mocap data. The sample generated by standard PCA has some noise on left leg, because the left foot supposes to be on the ground according to the definition of motion primitive $M(s_3)$. While for sample from weighted PCA fit this constraint well. However, for the high quality noise free data, e.g. Vicon mocap data, the performance of weighted PCA is almost the same as standard PCA.

4.5 Morphable Model Evaluation

The performance of morphable models are evaluated by the new motion segments sampled from morphable models. Our results are best seen by visualization. Figure 5.16 gives 16 examples of sampled motion segments from different motion primitives.

4.5.1 Evaluation Methods

The goal of morphable model is to generate physical correctness and natural-looking motions from motion primitives. The physical correctness means synthesized motion is physically valid and the prior knowledge about each motion primitive is embedded [64]. For instance, for our walking example, the contact point should stay on the ground. So the physical correctness is measured by computing the distance between the contact point of synthesized motion and corresponding contact plane (ground) in our experiment. Figure 4.17 shows the distance between the synthesized contact point and corresponding contact plane of each frame in synthesized motions in figure 4.16. The figures illustrate that the synthesized motions based on Vicon mocap data have much better performance in contact correctness, whose contact point in most frames are very close to contact plane.

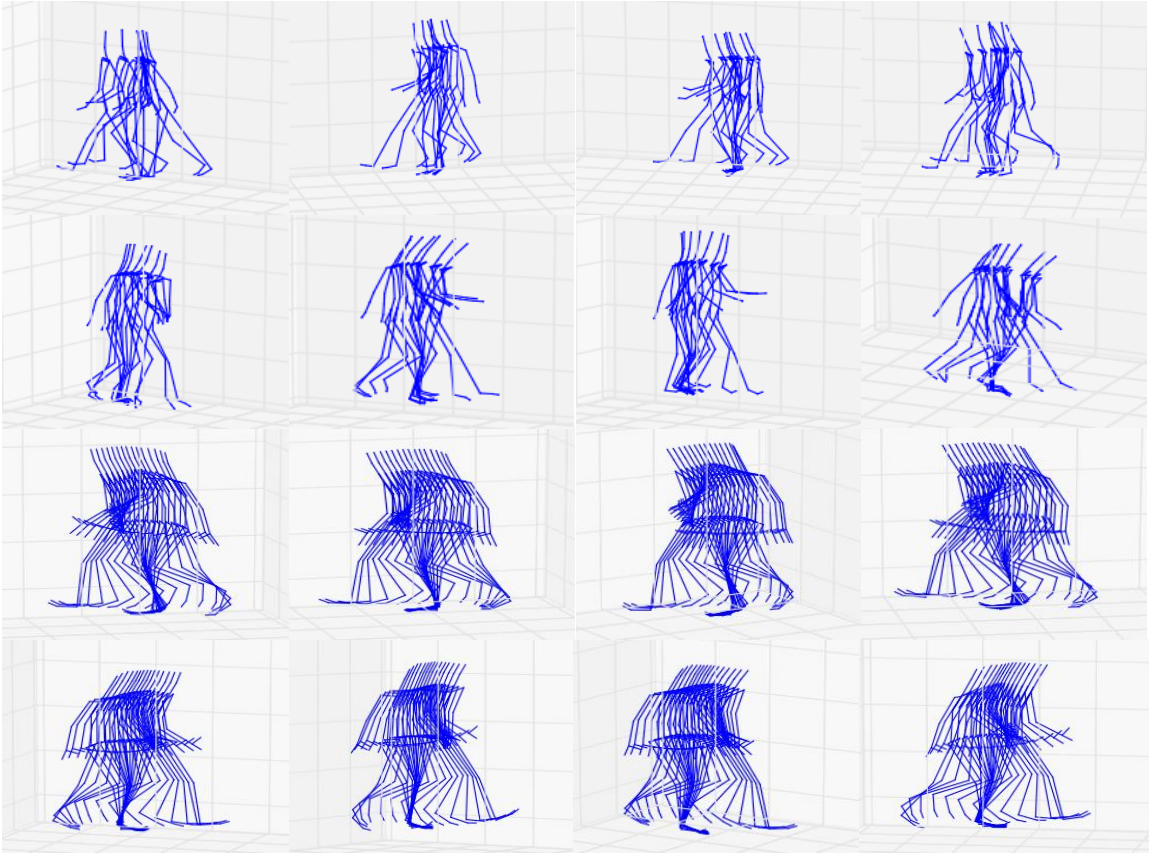


Figure 4. 16: Examples of sampled motion segments. First row: samples of motion primitive $M(s_3)$, trained by markerless mocap data; second row: samples of motion primitive $M(s_2)$, trained by markerless mocap data; third row: samples of motion primitive $M(s_3)$, trained by Vicon mocap data; last row: samples of motion primitive $M(s_2)$. trained by Vicon mocap data. For detail of motion primitives, see figure 3.2.

For the natural-looking of synthesized motion segment, It is hard to define a clear measure to judge how natural-looking of a motion is. Because the natural-looking of a motion depends on human knowledge of the motion. For example, it is very easy to notice any unnatural poses for walking motion, since people have seen walking every day. However, for some less familiar motions, e.g. ballet motions, it is not easy to tell some poses in the motion is natural-looking or not. Some researches evaluate the results of synthesized motion by user study [15, 65]. In our work, we find out that a natural-looking motion usually changes smoothly over frames. So we compute the forward difference of frames, and take the variance of forward difference as naturalness measure.

$$\Delta d_i = D(F_i, F_{i+1}) \tag{4.5}$$

$$naturalness = Var(\Delta \boldsymbol{d}) \tag{4.6}$$

where D is the frame distance defined by Eq. (3.1), F_i, F_{i+1} are *i*-th and *(i+1)*-th frames in frame sequence respectively.

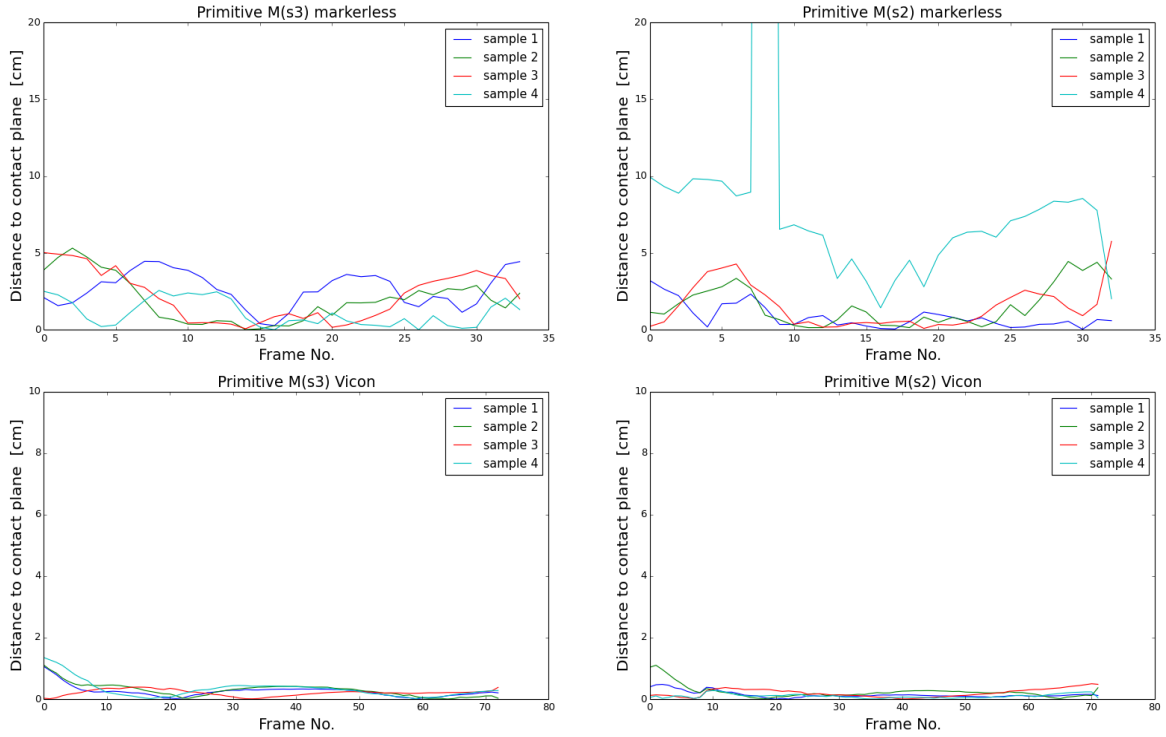


Figure 4. 17: Distance between contact point and corresponding contact plane of each frame of synthesized motion segments in figure 4.16. (up left): contact distance of 4 samples in first row; (up right): contact distance of 4 samples in second row; (bottom left) primitive $M(s_3)$ of walking from Vicon mocap data; (bottom right) primitive $M(s_2)$ of walking from Vicon mocap data.

In order to test this observation,. we compute the forward difference of frames of motion capture data. Figure 4.18 shows the forward difference of frames of 20 capture motion segments from two motion primitives: right first walking $M(s_2)$ and left first walking $M(s_3)$ for markerless data and Vicon data respectively. The figures support our assumption that the forward difference of Vicon data has very small variance. It is consistent with our observation that Vicon data has a very high quality. And the forward difference of markerless data has a much higher variance than Vicon data, which indicates the data is heavily noisy.

4.5.2 Experimental Results

As our morphable models are statistical models, we have evaluated the performance of morphable models on a set of samples from each morphable mode. In our experiment, 20 samples are taken from primitive $M(s_2)$ and primitive $M(s_3)$. The quality of synthesized motion is measured by the criteria from last section, and compared with original captured motion segments. Figure 4.20 shows the curves of distance between contact point and contact plane for pre-captured motions. The figures convinces that the quality of Vicon mocap data is better than markerless mocap data. Basically, the distance is very small and stable for most of frames, except for the first several frames. This could be caused by our motion decomposition approach. Because we use feet distance as feature to extract keyframes, so the contact-awareness is not guaranteed for each segments. Figure 4.19 shows the curves of distance between contact point and contact plane for synthesized motion. It illustrates the influence of

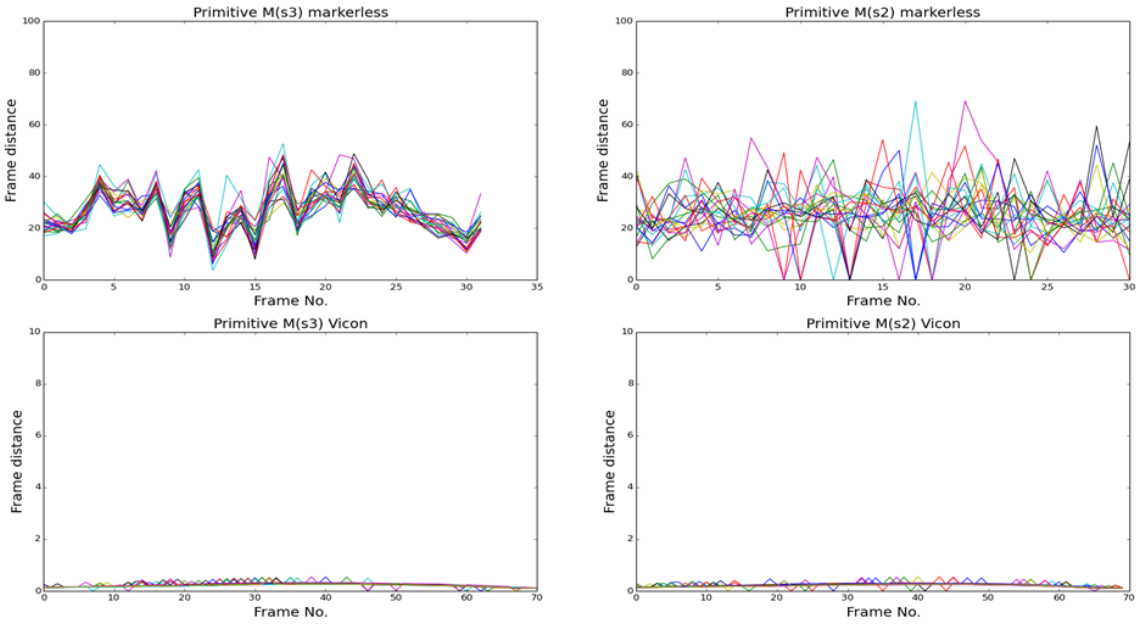


Figure 4. 18: The forward difference of 20 captured motion segments: (up left) samples from primitive $M(s_3)$ based on markerless data; (up right): samples from primitive $M(s_2)$ from based on markerless data; (bottom left): samples from primitive $M(s_3)$ based on Vicon data; (bottom right): samples from primitive $M(s_2)$ based on Vicon data.

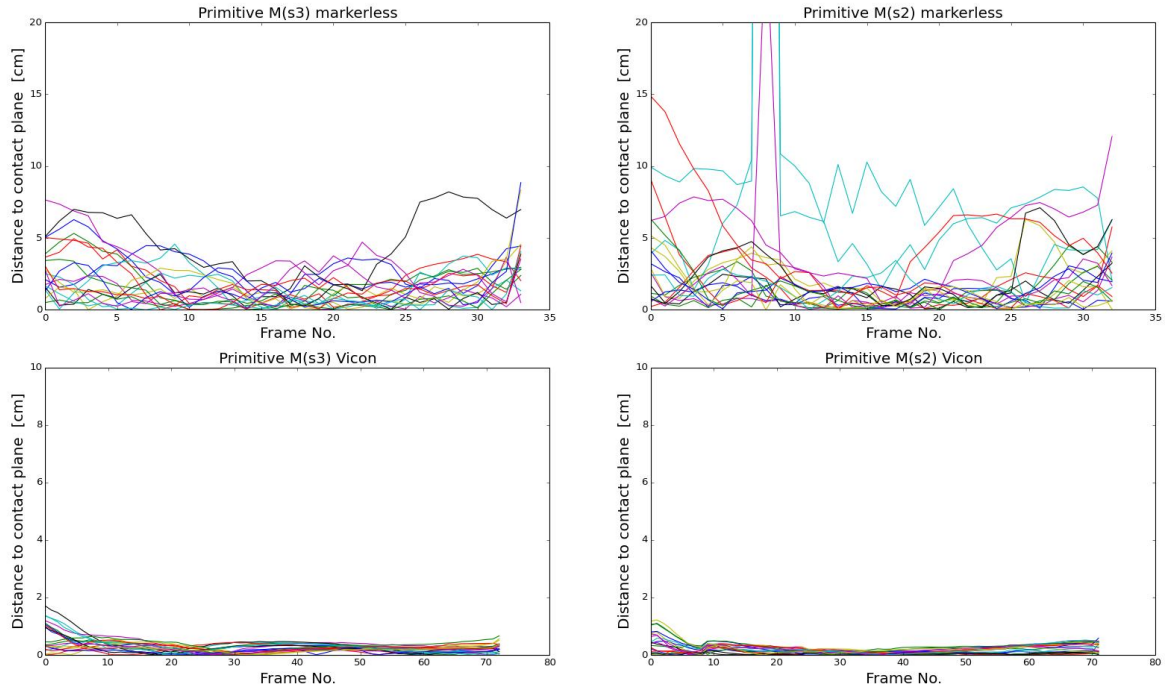


Figure 4. 19: Distance between contact point and corresponding contact plane of 20 sampled motion segments from primitive $M(s_3)$ and primitive $M(s_2)$ based on markerless and Vicon mocap data. (up left): primitive $M(s_3)$ trained by markerless data; (up right): primitive $M(s_2)$ trained by markerless data; (bottom left): primitive $M(s_3)$ trained by Vicon data; (bottom right): primitive $M(s_2)$ trained by Vicon data.

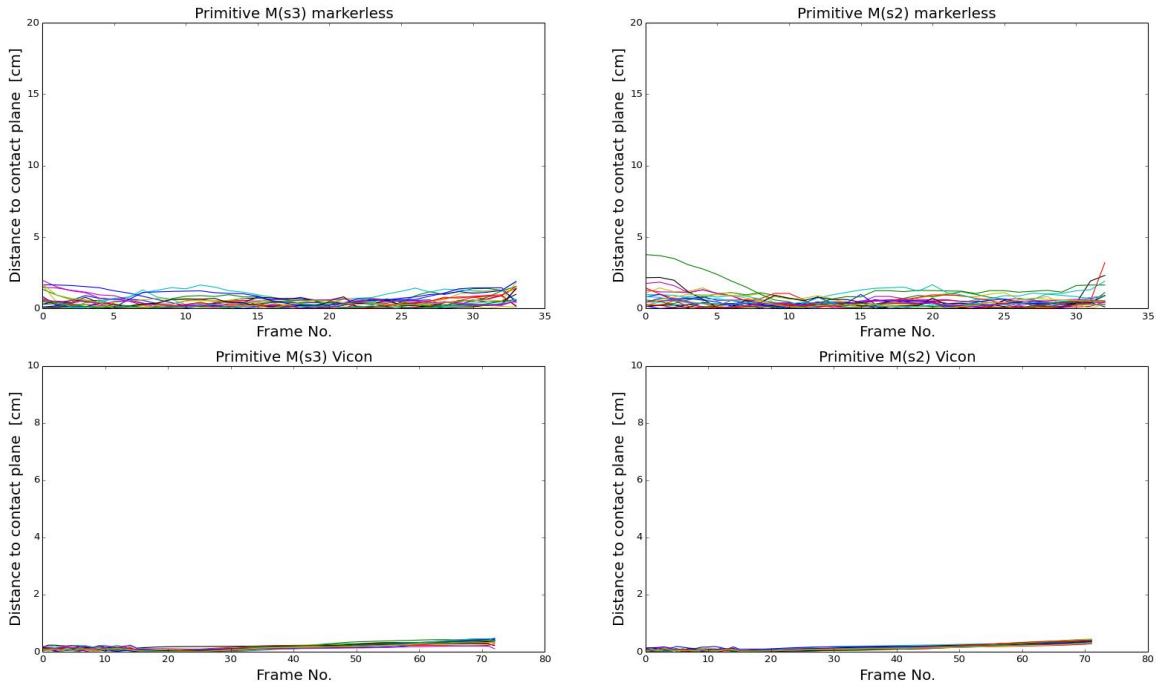


Figure 4. 20: Distance between synthesized contact point and corresponding contact plane of 20 original motion segments of primitive $M(s_3)$ and primitive $M(s_2)$ from markerless and Vicon mocap data. (up left): primitive $M(s_3)$ from markerless data; (up right): primitive $M(s_2)$ from markerless data; (bottom left): primitive $M(s_3)$ from Vicon data; (bottom right): primitive $M(s_2)$ from Vicon data.

the quality of motion capture data for statistical motion synthesis. From the results we can find that the physical correctness of synthesized samples from Vicon data is very similar to the training data in figure 4.20. The distance between contact point and corresponding contact plain is very small and stable. Table 4.4 computes average distance between contact point to contact plane for both pre-captured motions and synthesized motions. The quantitive result convinces the observations from figure 4.19 and 4.20, our statistical model is quite sensitive to the noise. The synthesized results from Vicon data only degrade a little in physical correctness. However, the results from markerless data degrade a lot.

	primitives	pre-captured motions	synthesized motions
markerless data	Right first walking	0.5193	2.9366
	Left first walking	0.4640	1.9152
Vicon data	Right first walking	0.1784	0.2558
	Left first walking	0.2556	0.2834

Table 4. 4: Evaluation results of physical correctness of markerless data and Vicon data.

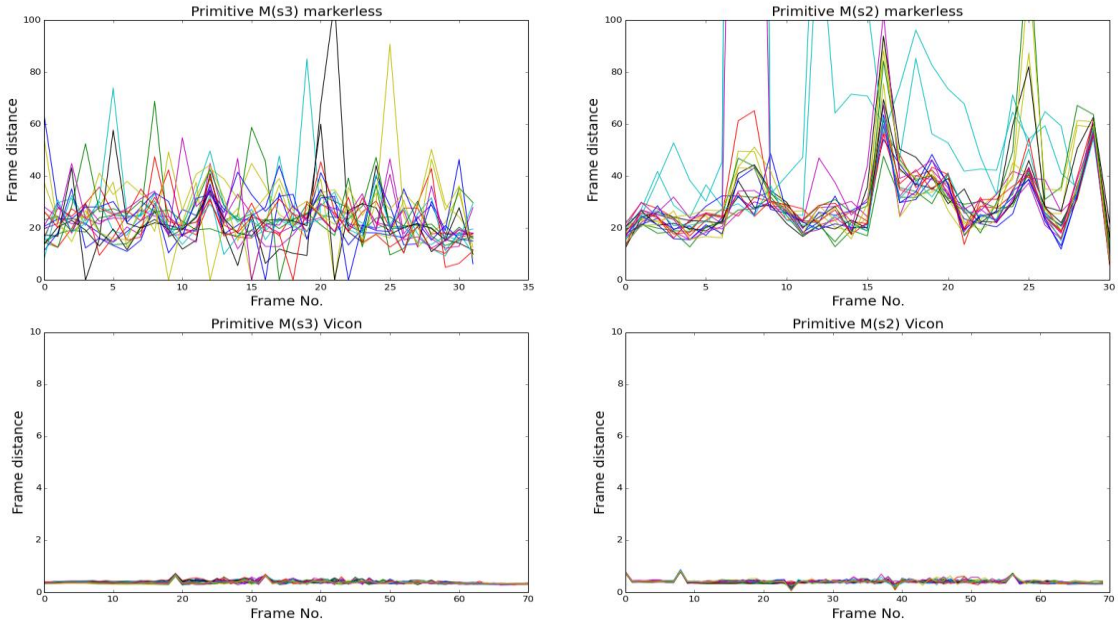


Figure 4. 21: The forward difference of 20 synthesized motion segments: (up left) primitive $M(s_3)$ of walking from markerless mocap data; (up right) primitive $M(s_2)$ of walking from markerless mocap data; (bottom left) primitive $M(s_3)$ of walking from Vicon mocap data; (bottom right) primitive $M(s_2)$ of walking from Vicon mocap data.

The results of naturalness measure for 20 synthesized motion segments and random samples from pre-capture data are given in table 4.5.

	primitives	pre-captured motions	synthesized motions
markerless data	Right first walking	84.2959544907	428.55970782
	Left first walking	66.3129244266	116.147046771
Vicon data	Right first walking	0.0405199047189	0.0451970445294
	Left first walking	0.0533480382759	0.0903985611321

Table 4. 5: Evaluation results of naturalness of markerless data and Vicon data.

From these observations, we conclude that the statistical morphable model demonstrates its ability to generate natural-looking and contact-awareness motions from high quality, noise free motion capture data, such as Vicon mocap data in our experiment. Some long walking examples are shown in figure 4.22. However, the performance of our statistical model is very sensitive to the noise in the input motion capture data. The quality of synthesized motions attenuate heavily with increase of the noise.

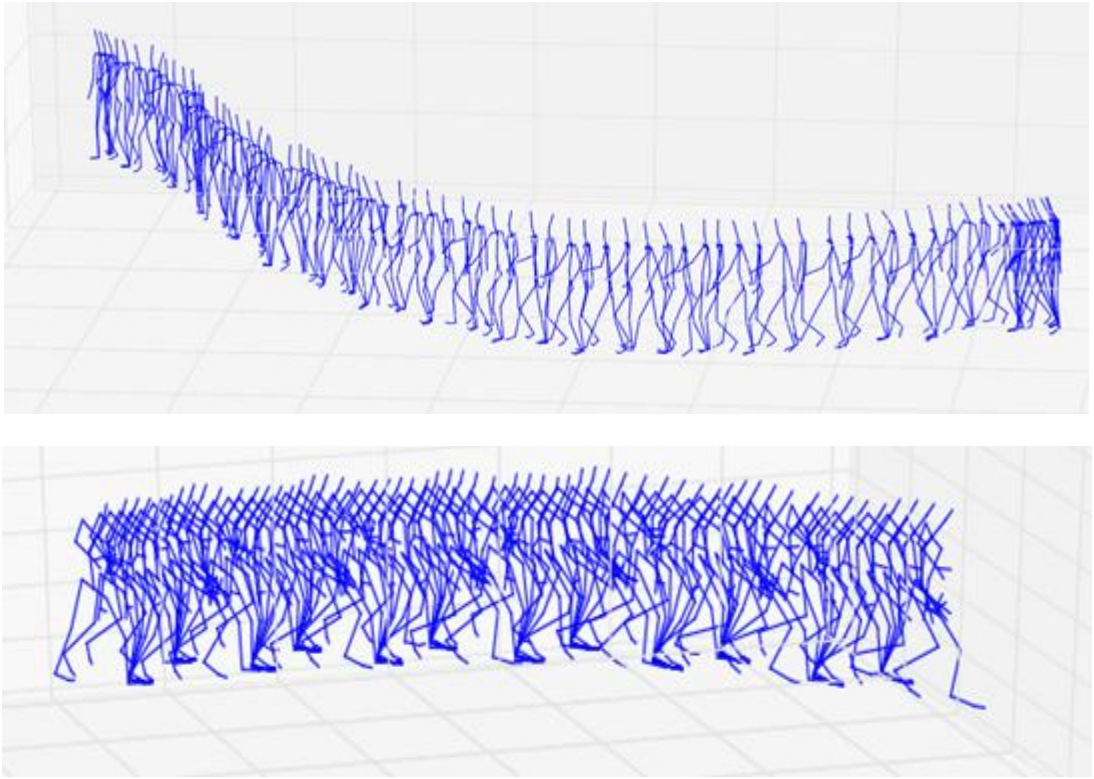


Figure 4. 22: Long walking motions generated by morphable models of walking. (up): synthesized motion from markerless data; (bottom): synthesized motion from Vicon data.

5 Summary and Future Work

In conclusion, although it is not novel to apply statistical analysis to computer graphics and computer vision, there is no generative statistical model for human motion synthesis, analysis and editing for practical usage in industry field to the best of our knowledge. In the thesis, we present our reengineering work for a novel generative human motion synthesis model -- motion graphs++ on our motion capture data. We set up a markerless motion capture system, and evaluate motion graphs++ methodology on our markerless mocap data and compare the result with a high quality system Vicon mocap data.

In addition, we modify some steps of the original algorithm [15] to make it fit our data better in the work. The motion capture data for our experiment is stored in BVH file, which employs a hierarchical human skeleton to represent motion capture data. So the value of each joint has the same scale in data representation, but has huge different impact in visualization. We find out that the motion graphs++ approach does not work well with our motion data representation if the data is very noisy. In order to overcome this problem, some modifications are employed in our work to fit our motion data representation. We use a weighted frame distance based dynamic time warping algorithm to align noisy motions. And weighted principal component analysis is applied in our work rather than standard PCA to address the different influence of joint's value. In addition, we find out that for structurally simple motion, such as running, walking and so on, using one or two dominant features to detect keyframes can provide efficient and good quality results. We test our approach on a simple motion -- free space walking and evaluate our results by measuring the physical correctness and naturalness of synthesized motions from morphable models we construct.

The experiment results demonstrate that these variants make our approach work better on our hierarchical motion capture data than original approach. And in our experiment, we show that the statistical approach for human motion synthesis is very sensitive to noise. The synthesized result from high quality motion capture data is far more better than noisy data. In addition, we believe that with the high quality motion capture data, the statistical motion synthesis approach is generative and scalable, and the synthesized motions are natural-looking and contact-awareness, which would be appealing for motion analysis, editing and synthesis in many applications.

This work is actually very preliminary step in thewhole project. Currently, we only evaluate the approach how statistical model use motion capture data to generate new variant motion, without any constraint conditions on that. And our test scenario is too simple to practical usage in assembly workshop. The next step of our work would be to generate variant motions under user's specified constraints and environmental constraints. In traditional motion synthesis approaches, these constraints are needed to be carefully analyzed for each motion and explicitly added to the corresponding joints in the skeleton of actor, which make the synthesis work not generative and hard to extend to different motions. One appealing property of our statistical approach is the model can be represented as a set of equations. So the constraint can be formulized as constraint term in our objective function derived from

statistical model. And the task of motion synthesis under constraints is converted to an optimization problem. The target motion will be the optimal parameters from the optimization problem. This conversion makes statistical motion synthesis become very generative and scalable. If users want to generate arbitrary valid motions from the model, they do not need to know anything about kinematic. Basically, what they need to do is present their constraints in supported forms.

And we also want to explore the scalable property of motion graphs++ as it is stated by authors. Currently, we only have a limit amount of motion capture data. The test of some simple motions like walking are obviously not enough for practical application. So we are going to test this approach on more different kinds of motions in real assembly workshop, such as working with screwdriver, moving box and so on.

Reference:

[1] Gausemeier, J.; Ebbesmeyer, P.; Eckes, R.: Virtual Production – Computer Model-Based Planning and Analyzing of Manufacturing Systems. Erschienen in: Dashchenko, A.I.: Reconfigurable Manufacturing Systems and Transformable Factories. Springer Berlin Heidelberg New York, ISBN-10 3-540-29391-4, 2006.

[2] Mandel, S.; Bär, T.; Fay, A.: Concept for Proactive Ramp-up Validation of Body-in-White Lines. 13th IEEE International Conference on Emerging Technologies and Factory Automation : IEEE, 2008. ISBN: 1424415063; 978-142441506-9, 2008.

[3] Nof S.Y., Wilhelm W.E., Warnecke H.J., Industrial Assembly, London, Weinheim, New York, Tokyo, Melbourne, Madras, 1997.

[4] Hartmann, B. et al. Reflective physical prototyping through integrated design, test, and analysis. In Proceedings of UIST, pp 299–308, 2006.

[5] Kulkarni, A., Kapoor, A., Iyer, M., Kosse, V. (2011). Virtual prototyping used as validation tool in automotive design. Proceedings of the 19th International Congress on Modelling and Simulation, Perth, Australia, 12–16 December 2011, p. 419-425, ISSN.

[6] Friedlaender, A. F., C. Winston, et al. (1983). "COSTS, TECHNOLOGY, AND PRODUCTIVITY IN THE U. S. AUTOMOBILE INDUSTRY." Bell Journal of Economics 14(1): 1-19.

[7] Y.Pang, A.Y.C. Nee, K. Youcef-Toumi, S.K. Ong and M.L. Yuan, Assembly Design and Evaluation in an Augmented Reality Environment. Singapore-MIT Alliance Symposium, (19-20 January 2005) Singapore.

[8] Zorriassatine F, Wykes C, Parkin R, Gindy N (2003) A survey of virtual prototyping techniques for mechanical product development. Proc Inst Mech Eng B J Eng Manuf 217 (4):513–530.

[9] McLean, C., 1993. Computer-Aided Manufacturing System Engineering, Proceedings of the IFIP TC5/WG5.7 International Conference on Advances in Production Systems, APMS '93.

[10] K. W. Lee, “CAD system for human-centered design,” Computer-Aided Design & Applications, vol. 3, no. 5, pp. 615-628, 2006.

[11] PERLIN, K., AND GOLDBERG, A. 1996. Improv: A system for scripting interactive actors in virtual worlds. In ACM SIGGRAPH.

[12] Manns, M.; Arteaga Martín, N. A.: Automated DHM Modeling for Integrated Alpha-Numeric and Geometric Assembly Planning. Proceedings of the 23rd CIRP Design Conference, Bochum, Germany, Springer-Verlag Berlin Heidelberg, 2011.

[13] Bruderlin A. and Williams L.: 'Motion signal processing', in Proceedings of the 22nd annual conference on Computer Graphics and Interactive Techniques, pp. 97-104, 1995.

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

[15] MIN, J., and CHAI, J. 2012. Motion Graphs++: A Compact Generative Model for Semantic Motion Analysis and Synthesis. In ACM Transactions on Graphics, 31(6)

[16] D.Winter. Biomechanics and Motor Control of Human Movement. Wiley, New York, 1990.

[17] J. Hodgins,W.Wooten, D. Brogan, and J. O'Brien. Animating human athletics. In Proceedings of ACM SIGGRAPH 95, Annual Conference Series, pages 71–78. ACM SIGGRAPH, August 1995.

[18] W.Wooten and J. Hodgins. Dynamic simulation of human diving. In Proceedings of Graphics Interface (GI'95), pages 1–9, May 1995.

[19] W. Wooten and J. Hodgins. Simulating leaping, tumbling, landing, and balancing humans. In IEEE International Conference on Robotics and Animation, volume 1, pages 656–662, 2000.

[20] BOWDEN, R. Learning non-linear Models of Shape and Motion , PhD Thesis, Dept Systems Engineering, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK. December 99.

[21] S. Theodore. Understanding animation blending. Game Developer, pages 30–35, May 2002.

[22] WILEY, D., AND HAHN, J. 1997. Interpolation synthesis of articulated figure motion. IEEE Computer Graphics and Application 17, 6, 39.45.

[23] R. Bowden. Learning statistical models of human motion. In IEEE Workshop on Human Modelling, Analysis, and Synthesis, CVPR 2000. IEEE Computer Society, June 2000.

[24] M. Mizuguchi, J. Buchanan, and T. Calvert. Data driven motion transitions for interactive games. In Eurographics 2001 Short Presentations, September 2001.

[25] KOVAR, L., AND GLEICHER, M. 2003. Flexible automatic motion blending with registration curves. In ACM SIGGRAPH/EUROGRAPH Symposium on Computer Animation. 214–224.

[26] K. Perlin. Real time responsive animation with personality. IEEE Transactions on Visualization and Computer Graphics, 1(1):5–15, March 1995.

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

[28] O. Arikan and D. A. Forsyth. Interactive motion generation from examples. ACM Transactions on Graphics, 21(3):483–490, 2002.

[29] J. Lee, J. Chai, P. Reitsma, J. Hodgins, and N. Pollard. Interactive control of avatars animated with human motion data. ACM Transactions on Graphics, 21(3):491–500, 2002.

[30] K. Perlin. Real time responsive animation with personality. IEEE Transactions on Visualization and Computer Graphics, 1(1):5–15, March 1995.

[31] K. Pullen and C. Bregler. Motion capture assisted animation: texturing and synthesis. ACM Transactions on Graphics, 22(3):501–508, 2003.

[32] C. Rose, B. Guenter, B. Bodenheimer, and M. Cohen. Efficient generation of motion transitions using spacetime constraints. In Proceedings of ACM SIGGRAPH 1996, Annual Conference Series, pages 147–154. ACM SIGGRAPH, August 1996.

[33] M. Brand and A. Hertzmann. Style machines. In Proceedings of ACM SIGGRAPH 2000, Annual Conference Series, pages 183–192. ACM SIGGRAPH, July 2000.

[34] A. Galata, N. Jognson, and D. Hogg. Learning variable-length markov models of behavior. Computer Vision and Image Understanding Journal, 81(3):398–413, 2001.

[35] Y. Li, T.Wang, and H.-Y. Shum. Motion texture: A two-level statistical model for character motion synthesis. ACM Transactions on Graphics, 21(3):465–472, 2002

[36] K. GROCHOW, S. L. MARTIN, A. HERTZMANN, AND Z. POPOVI C ,2004. Style-based inverse kinematics. In ACM Transactions on Graphics. 23(3):522–531.

[37] Elhayek, A.; Stoll, C.; Hasler, N.; Kim, K. I.; Seidel, H.-P.; Theobalt, C.: Spatio-temporal Motion Tracking with Unsynchronized Cameras. IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2010) IEEE, 2012.

[38] MEREDITH, M., MADDOCK, S. Motion capture file formats explained. Department of Computer Science, University of Sheffield.

[39] MYERS, C. S., AND RABINER, L. R., A comparative study of several dynamic time-warping algorithms for connected word recognition. In The Bell System Technical Journal. 60(7):1389-1409, 1981.

[40] RABINER, L., and SCHMIDT, C., Application of Dynamic Time Warping to Connected Digit Recognition. IEEE Transactions on acoustics, speech, and signal processing, VOL. ASSP-28, NO. 4, AUGUST, 1980.

[41] Müller, M., Information Retrieval for Music and Motion, Springer, 2007.

[42] Safonova, A., Hodgins, J.L., Pollard, N.S. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. ACM Transcations on Graphics 23(3), 524-521, 2004.

[43] Pearson, K. On lines and planes of closest fit to systems of points in space. Philosophical Magazine, 1901.

[44] C.M. Bishop. Pattern recognition and machine learning. Springer, 2006.

[45] Weakliem, David L. 1999. "A Critique of the Bayesian Information Criterion for Model Selection." Sociological Methods & Research 27:359-97.

[46] MUKAI, T., AND KURIYAMA, S. 2005. Geostatistical Motion Interpolation. In ACM Transactions on Graphics. 24(3):1062–1070.

[47] LAWRENCE, N. D., Gaussian Process Latent Variable Models for Visualization of High Dimensional Data. Proc. NIPS 16, 2004.

[48] RASMUSSEN, C. E., AND WILLIAMS, C. K. I., Gaussian Processes for Machine Learning. The MIT Press, 2006.

[49] IKEMOTO, L., ARIKAN, O., AND FORSYTH, D., Generalizing motion edits with gaussian processes. ACM Transactions on Graphics. 28(1):1–12, 2009.

[50] YE, Y., AND LIU, K., Synthesis of responsive motion using a dynamic model. Computer Graphics Forum (Proceedings of Eurographics), 2010.

[51] WEI, X., MIN, J., AND CHAI, J., Physically valid statistical models for human motion generation. ACM Trans. Graph. 30, 19:1–19:10, 2011.

[52] RASMUSSEN, C. E., AND WILLIAMS, C. K. I., Gaussian Processes for Machine Learning. The MIT Press, 2006.

[53] F. De la Torre, "A Least-Squares Framework for Component Analysis," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 34, no. 6, pp. 1041-1055, June 2012.

[54] M. Irani and P. Anandan, "Factorization with uncertainty," in European Conference on Computer Vision, 2000.

[55] M. J. Greenacre, Theory and Applications of Correspondence Analysis. London: Academic Press, 1984.

[56] Gabriel, K.R. Biplot display of multivariate matrices for inspection of data and diagnosis. In Interpreting Multivariate Data, ed. V. Barnett, 147–173. Chichester: Wiley, 1981.

[57] Rao, C.R.. The use and interpretation of principal component analysis in applied research. Sankhya A, 26, 329–358, 1964.

[58] I. T. Jolliffe, Principal Component Analysis. New York: Springer-Verlag, 1986.

[59] Sujatha C, Mudenagudi, U. "A Study on Keyframe Extraction Methods for Video Summary", IEEE Conference on Computational Intelligence and Communication Networks (CICN), 7-9 October 2011,Gwalior, India.

[60] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. Introduction to Information Retrieval. Cambridge Univ Press, 2008.

[61] Bachynskyi, M., Oulasvirta, A., Palmas, G., Weinkauf, T. Is motion-capture-based biomechanical simulation valid for HCI studies? Study and implications. Proc. CHI'14

[62] Mandel, S.; Bär, T.; Fay, A.: Concept for Proactive Ramp-up Validation of Body-in-White Lines. 13th IEEE International Conference on Emerging Technologies and Factory Automation : IEEE, 2008. ISBN: 1424415063; 978-142441506-9, 2008.

[63] P.S.A. Reitsma, N.S. Pollard, Evaluating motion graphs for character animation, ACM Transactions on Graphics (2007) 18.

[64] A. Safonova, J.K. Hodgins, Analyzing the physical correctness of interpolated human motion, in: ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 2005, pp. 171–180.

[65] Y. Zhao, L. Yu, A perceptual metric for evaluating quality of synthesized sequences in 3DV system. In: Proceedings of visual communications and image processing (VCIP), July 2010.