A Nonlinear Manifold Learning Framework for Real-time Motion Estimation using Low-cost Sensors

Liguang Xie, Bing Fang, Yong Cao, Francis Quek
Center for Human Computer Interaction
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24060
Email: { xie, fangb, yongcao, quek }@vt.edu

***Abstract*—We propose a real-time motion synthesis framework to control the animation of 3D avatar in real-time. Instead of relying on motion capture device as the control signal, we use low-cost and ubiquitously available 3D accelerometer sensors. The framework is developed under a data-driven fashion, which includes two steps: model learning from existing high quality motion database, and motion synthesis from the control signal. In the model learning step, we apply a non-linear manifold learning method to establish a high dimensional motion model which learned from a large motion capture database. Then, by taking 3D accelerometer sensor signal as input, we are able to synthesize high-quality motion from the motion model we learned from the previous step. The system is performing in real-time, which make it available to a wide range of interactive applications, such as character control in 3D virtual environments and occupational training.**

I. Introduction

Real-time character animation technique is critical in a number of interactive applications, such as avatar control in 3D Virtual Environments (VE) and occupational training. As one of the most popular automatic animation recording and playback techniques, optical motion capture systems, such as VICON, provide a real-time performance capture solution for character animation. Because its high-fidelity data quality, motion capture systems have been adopted as a standard animation generation tool in animated-film and video game industry. However, high cost and complex configuration place significant constraints for its ubiquitous application and extension. In this paper, we propose a low-cost motion synthesis framework to control the animation of 3D avatar in real-time. Our framework takes control signals from low-cost 3D accelerometer sensors as input and generate high quality human motion in real-time by applying a statistical model learned from a pre-recorded motion database. Our system is low cost and easy to set up since our proposed system only needs four Nintendo Wii controllers.

The main focus of our approach lies on the model learning from a prerecorded human motion database which it is critical for the quality of synthesized motion. The challenge of this research is to generate detailed and high dimensional motion data (more than 80 dimensions) from a noisy and lower dimensional sensor signals (less than 25 dimensions). We adopt a data-driven approach that build a statistical mapping model between low dimensional input signal and high dimensional motion data by learning from a pre-captured sensor and motion database. In our approach, the database consists of motions from various sport movements, such as tennis, basketball and golf etc. Global linear models have difficulties handling this type of heterogeneous motion database [3], [13], [17]. To solve the limitation of linear models, we learn a strong nonlinear reduced-dimension models. We use *Locally Linear Embedding* (LLE) to learn the nonlinear manifold in high dimensional motion data. Our result shows that the nonlinear manifold leaning model has better performance than linear models.

We evaluate our system by comparing the synthesized results with ground truth which is simultaneously captured by an optical motion capture system. The evaluation shows that our system can accurately estimate full body motion using a small number of low-cost and noisy accelerometer sensors.

The remainding paper is organized as follows. Section II provides the background and describes the related work in this area. Section III explains the system architecture while Sections IV provides the detailed description of our approach. Section V shows the experimental results and demonstrates the accuracy of our approach numerically and visually. Section VI concludes the paper, discusses the limitations and future work to address the current limitations.

II. Background

Our proposed work can be categorized into the research of performance capture for human motion, where human performance can be recorded by sensors and re-generated in the form of 3D animated avatars. In this section, we will describe the existing work in this area, and then explain how we are motivated to use nonlinear manifold learning methods for this research.

A. Performance capture for human motion

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

1) Optical systems: Optical systems utilize image data captured from a number of cameras to track special markers attached to a subject, or to recognize surface features identified dynamically for each particular subject. Marker-based optical systems can generate high-fidelity animation data with subtle details of human motions. These systems perform best in the applications that mostly play back the original motions, e.g., animated movies. However, most popular optical motion capture systems, such as VICON or Motion Analysis, are costly and, then, can not be widely deployed for interactive motion capture and motion control.

In order to lower the cost of performance capture, researchers explored the possibility of using standard video camera, compared with expensive professional camera. Chai et al. [4] implemented a vision based system that requires only two inexpensive video cameras. Using only six markers attached to a body, the system can synthesize a wide variety of human movement without a long suit-up time. The synthesized motions are very detailed because a data-driven approach is used to query a high quality motion capture database. Similarly, Liu et al. [10] applied a linear regression model to estimate human motions from a reduced marker set. However, these systems require a restrictive motion capture environment and suffer from the occlusion problem of a vision based tracking system.

Recently, Aguiar et al. [6] presented a markerless approach for video-based performance capture. This approach require 8 cameras of high resolution and produces feature-rich human motion, comprising of high-quality geometry, life-like motion data and surface texture of recorded subjects. This multi-view stereo approach support people wearing a wide variety of everyday apparel and performing fast motion. As the feature-rich output format, this method supplements and exceeds the capabilities of marker-based optical capturing systems. However, it is still a high cost approach and is limited by the capture environment.

2) Non-optical systems: Non-optical systems use acoustic, inertial, magnetic sensors or combinations of these sensors, which are usually low-cost. The sensors signals, providing an digital representation of the motion, are used as control signals to synthesize human motion.

Badler et al. [1] proposed a system that reconstructs full-body motion using four magnetic sensors and a real-time inverse-kinematic algorithm to control a standing character in a VE. The system introduced a data-driven approach to address the kinematic redundancy problem. Another system developed by Yin and Pai [19] synthesizes full-body motion within one second by using a foot pressure sensor. However, it can only generate a small range of behaviors and cannot produce motion for complex upper body movements. Oore et al. [11] use six degree-of-freedom tracking devices to interactively control the locomotive animations. Dontcheva et al. [7] also use a tangible user interface in their puppetry system to control the motions divided into several different layers.

Recently, Vlasic et al. [16] combined accelerometer, inertial and acoustic sensors to capture high-fidelity motions that are comparable to the motions captured from marker based vision systems. The system removes the restriction of constrained motion capture environments, and allows the user to be tracked almost “everywhere”. However, the cost of the systems are still high and, due to the necessary post-processing time, it is not a real-time system.

Safonova et al. [13] utilize an existing motion capture database and proposed a optimization algorithm in low-dimensional spaces to synthesize human motion. Principle Component Analysis (PCA) is used on motion with similar behavior to reduce data dimensionality thus the subject behavior has to be specific. The limitation results from dimensionality reduction technique used because for a global linear method, such as PCA, it is hard to model a heterogeneous database which is possibly nonlinear. Global linear models might be appropriate for Safonova’s application which is synthesis of motion without a continuous control signals, however, it is not a best choice for us. Similarly, Carvalho et al. [3] presented a low-dimensional optimization framework that used a Prioritized Inverse Kinematics (PIK) strategy. Two local motion models, i.e. PCA and Probabilistic PCA, are used to reduce dimensionality and their performances are compared for solving the optimization problem. However, the approach is only limited to specific behavior, i.e., golf swing, suffering from the model problem.

B. Nonlinear dimension reduction for motion synthesis

In order to estimate and synthesize high-quality human motion from noisy input signal, we adopted a widely used data-driven approach where a statistical model is learned from motion data. In our previous work [17], the statistical model we used is a piece-wise linear model which results from a global clustering and linear model learning for each local cluster. However, we find the approach is limited by the clustering result when operates on linearly dimension-reducted (e.g. by PCA) data.

To solve the limitation of linear dimensionality reduction, a new class of nonlinear dimensionality reduction technique have been developed. These algorithms are designed to explore the nonlinear structure for high dimensional data. Isomap algorithm [15] is a nonlinear generalization of MDS [5]. Isomap is designed to preserve the geodesic distance between pairs of multivariate data point, instead of simply taking Euclidean

distance. The geodesic distance can present the distances along the manifold. Roweis et al. [12] and Saul et al. [14] proposed LLE, an unsupervised learning algorithm that computes low dimensional embedding with neighborhood relationship preserving of high dimensional data. LLE is able to discover nonlinear structure in high dimensional data by optimally preserving the local configurations of nearest neighbors. The advantage of LLE over linear dimension reduction technique, such as PCA and MDS, is that LLE is able to correctly detect the nonlinear structure and project the multivariate data into a single global coordinate system of low dimension.

Yeasin et al. [18] discussed the performances of several linear and nonlinear dimensionality reduction techniques in classifying universal facial expressions, i.e., PCA, Non-negative Matrix Factorization (NMF) and LLE. Their results show LLE has highest recognition accuracy.

Elgammal et al. [8] employed LLE to find the low dimensional embeddings of silhouette manifold. Given sequences of silhouette from monocular uncalibrated camera, a sequence of human 3D poses are produced by RBF interpolations from the silhouette manifold to 3D pose in body joint space. Likewise, Jaeggli et al. [9] proposed a body pose estimation system using video sequence as input. The pose is synthesized from a statical model of 3D pose, dynamics, activity transition and silhouette using sparse kernel regressors. Both of the approaches are offline. Our approach is partly similar to Elgammal's, however, our goal is different. We focus on real-time animation driven by accelerometer sensors.

III. System Overview

We describe a novel approach that uses low-cost 3D accelerometer sensors for full-body motion synthesis. The high-quality motion is synthesized using a statistic model learned from a motion capture database. There are three major phases of our approach – data collection, model learning and motion synthesis. Figure 1 shows the work flow of our system.

Data collection and preprocessing: We first perform a series of off-line motion capture sessions using simultaneously an optical motion capture system and accelerometer sensors (Wii controllers). Both motion capture data and sensor data are preprocessed to reduce noise. We then synchronize the motion data with the sensor data in order to get a precise frame-to-frame mapping. All the data is then stored in a database for motion synthesis. The part will be explained in the Section V.

Model learning: To build a statistical model of the captured motion database, we use *local linear embedding* (LLE) to learn a nonlinear manifold in a reduced dimensional space. We then use a *Gaussian Mixture Clustering* (GMM) algorithm to segment the low dimensional nonlinear manifold data into a number of clusters. The clustering result provides us a better understanding of the structure of motion data in terms of data feature similarity. Finally we build a *Radial Basis Function* (RBF) interpolation model for each cluster. The local

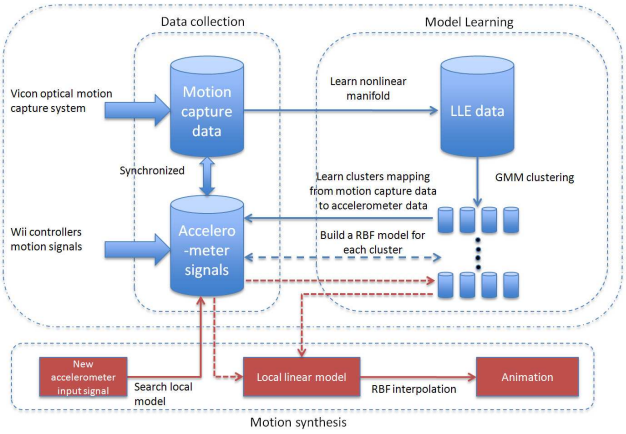

Fig. 1: *Our system work flow*

linear models enable us to learn the nonlinear manifold in the database.

Motion synthesis: During the motion synthesis phase, the user performs actions using only 3D acceleration sensors (Wii controllers) attached to the body. Using this sensor data as input, we synthesize high quality motion data by the local linear model built from the motion database captured in the previous phase. For each frame of the input sensor data, we apply the RBF interpolation function of the cluster associated with the input data. The result is a 3D pose with the same quality as the one of the motion capture data.

IV. Approach

In this section, we describe how we learn our model first, and then we discuss how to use the learned model for motion synthesis.

A. Model Learning

In our data-driven framework, a variety of dataset can be applied. Our current database consists of two types of synchronized data: one is obtained from a high quality motion capture system (Vicon), and the other from low-cost sensors with less accurate signals (accelerometers). These two types of data should be synchronized before modeling since the frame rates of device signal from various resources are usually different. The synchronized database is represented as

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

where N is the total number of frames and $\mathbf{c}_t$ is a d dimensional data of sensors representing the low quality signal at time t. $\mathbf{q}_t$ is the D dimensional data of high quality data at time t. $d = 24$ and $D = 88$ in our current database.

Once the database is synchronized, it is ready to be used to learn the local models. To improve the performance of the signal recognition and reduce computation cost, we perform

classification on the high quality dataset, which means our model describe the relationship between the low-dimensional database and the high-dimensional database with the same clustering information. Since the two data sets (high quality and less accurate data set) are synchronized before, the two data sets have exactly the same classification. In our previous work [17], we demonstrate that we need to perform clustering algorithm on the high quality motion capture dataset because of the accuracy reason. However, the high quality data is usually quite high dimension and contains a bunch of redundant features which do not help recognition. If the classification is applied directly onto the original high dimensional dataset, the performance is lower than our expectation. Thus we employ dimensionality reduction strategy to remove the redundant features in the original database. We use PCA in our previous work [17] but we found that linear dimensionality reduction technique are not the best way to represent our database for the complex nonlinear structure. Since *LLE* is a excellent way for nonlinear manifold learning, therefore, we use *LLE* for better representation of our data.

1) Non-linear Dimension Reduction: LLE is introduced for the problem of nonlinear dimensionality reduction. Given N input vectors $\{x_1,x_2,\ldots,x_N\},x_i \in R^d$, LLE calculates new vectors $\{y_1,y_2,\ldots,y_N\},y_i \in R^m$, and $m \leq d$. Normally, LLE consists three steps in its algorithm:

- Find the k nearest neighbors for each x_i.
- Measure reconstruction error resulting from the approximation of each point by its neighbors and calculate the reconstruction weights w_{ij} which minimize the error.
- Compute the low-embedding vectors y_i by minimizing the embedding function with the reconstruction weights w_{ij}.

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

In the second stage, we calculate reconstruction error by:

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

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

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

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

This embedding cost function is calculated based on the previous locally linear reconstruction errors, and the weight w_{ij} is fixed when optimizing y_i. In this procedure, the high-dimentional vector data x_i is one-to-one mapping to the low-dimentional vector y_i. So, we can perform a simpler clustering algorithm on this low-dimentional data set, and get the original data set the same clusters as this low-dimentional data.

2) Database Clustering and Clusters Mapping: In our clustering algorithm, we model the high quality motion database by the Gaussian mixture model (GMM), which is defined as:

$$p(x|\theta) = \sum_{j=1}^{K}\pi_j\eta(x|\mu_j,\Sigma_j). \tag{3}$$

When we perform LLE on the original high-quality database, we employ the GMM on the processed database to obtain the cluster set $C_i(i = 1,\ldots,k)$. So, we have k different clusters for the original database.

Since the high- and low-quality databases are synchronized when capturing, we apply the same clustering result to the low quality database, which means for any C_i, it has the same members in high and low quality database, which represented as Q_j. And we have k different clusters in the low quality database too. This cluster distribution information is the model we learnt for the motion synthesis.

3) Radial Basis Function: Now we can build a local linear model for each cluster. For the j^{th} cluster, we can build a local linear model using Radial Basis Functions (RBF) [2] to learn the mapping function from $\mathbf{c}_i^j$ to $\mathbf{q}_i^j$, where $\mathbf{c}_i^j \in C_j = \{\mathbf{c}_i^j | i = 1,2,..,p_j\}$, $\mathbf{q}_i^j \in Q_j = \{\mathbf{q}_i^j | i = 1,2,..,p_j\}$ and p_j is the number of frames in cluster j. Given a new input sensor data point $\tilde{\mathbf{c}}_t$ at the time frame t, if this data is classified as the j^{th} cluster, the mapping function can be expressed using Equation 4 as:

$$\tilde{\mathbf{q}}_t = F_j(\tilde{\mathbf{c}}_t) = \bar{\mathbf{q}}_j + A_j\sum_{i=1}^{p_j}\mathbf{w}_{ij}\phi(||\tilde{\mathbf{c}}_t - \mathbf{c}_i^j||), \tag{4}$$

where $\tilde{\mathbf{q}}_t$ is the high quality pose we want to synthesize, $\mathbf{w}_{ij}$ are unknown weights, $||\cdot||$ denotes a metric – in our case Euclidian distance, and $\phi()$ is a continuous kernel function.

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

$$\phi(r) = e^{-r^2/\sigma^2}, \tag{5}$$

because it is non-linear and provides good results when applied locally. The width σ, determines the amount of area covered by Gaussian function on the data. Since data points are not uniformly distributed in the data space, in order to improve quality of output we implemented a dynamic σ [2] dependent on the density of local data.

By using the local cluster data $\{\mathbf{c}_i^j, \mathbf{q}_i^j\}$, we can solve for unknown weights $\mathbf{w}_{ij}$ to complete the local linear model (Equation 4).

B. Motion Synthesis

1) Cluster Detection: When receiving a on-line low quality signal s, which has the same d dimension as the low quality database, we classify this signal by existing clusters in low quality database first. As we use GMM to do classification, we present our clusters by their mean values $\mu_i (i = 1, \ldots, k)$. We compute the likelihood based on the Euclidean distance:

$$p_i = \frac{1}{\sqrt{\sum_{j=1}^{d} (s_j - \mu_{ij})^2}}. \quad (6)$$

Because of the Gaussian distribution of our model, we rank the likelihood and choose the top three clusters as our interpolation database.

2) Radial Basis Function Interpolation: Given the new input sensor data $\tilde{\mathbf{c}}_t, 1 \leq t \leq N$, with N frames, we apply the local linear model learned from the previous step to synthesize the new high quality motion $\tilde{\mathbf{q}}_t, 1 \leq t \leq N$.

For the input sensor data $\tilde{\mathbf{c}}_t$ at frame t, we identify the cluster it belongs to by calculating the closest distance against the mean values of all clusters in sensor data, $\bar{\mathbf{c}}_j, 1 \leq j \leq K$. If it is classified as cluster j, we use RBF mapping function $F_j()$ defined in Equation 4 to synthesize new motion data frame $\tilde{\mathbf{q}}_t$.

3) Estimation and Smoothing: Because the low-quality input has its limitation of accuracy, we employ a least square method for estimation. Suppose we have historical series of motion data m_{i-1}. We fit the data to a cure:

$$f(x) = ax^2 + bx + c. \quad (7)$$

Then, we use this curve to estimate current motion m_i. Meanwhile, we get the current motion synthesis signal s_i. We fuse these two signals with sum-to-one weights:

$$\tilde{\mathbf{q}}_i = w_{detect} m_i + w_{estimate} s_i, \quad (8)$$

while $w_{detect} + w_{estimate} = 1$. The fusion result reduces the error and smoothes our motion results at the same time.

V. Experiment and Numerical Comparison

In this section, we explain the process for our database construction during the experiment. We then give our results relying on the database and show visual and quantitative accuracy of our system. Compared with the linear approach in our previous paper [17], the improved results show the benefit of nonlinear manifold learning.

A. Data Collection and Preprocessing

In this section we describe data used in our system, including data capture, synchronization and pre-processing.

Fig. 2: *Data collection: an optical motion capture system and a 3D acceleration sensor based data acquisition system are used in parallel. There are 45 retro-reflective markers and eight sensors (four Wii™ Nintendo controllers) attached to the performer.*

1) Data Capture and Representation: As we discussed in the previous section, we use the Vicon optical motion capture system to capture the high quality database, and Wii controllers are used for low quality motion capture database. In our Vicon system, we have 8 Vicon MX series cameras at a capturing rate of 60 frames per second, and we place 45 retro-reflective markers on the performer to collect motion signals. The Wii controllers play a role of 3D accelerometers with a range of interface for data transmission at a peak rate of 100 frames per second, and we attach 4 Wii remotes and 4 Wii nunchucks to the performer.

In our database, there are five different types of full-body motions: tennis forehand (1121 frames), tennis backhand (1004 frames), basketball shot (1300 frames), golf swing (1201 frames), and karate middle block (456 frames). In this procedure, there are totally 5082 frames of data in our database, represented as

$$\{(\mathbf{c}_t, \mathbf{q}_t) | t = 1, \ldots, 5082\}$$

where $\mathbf{c}_t$ is a 24-dimensional data of sensors representing the 3D acceleration measures of the four pairs of sensors on the body at time t. $\mathbf{q}_t$ is the 88-dimensinal data of optical motion capture data and it represents a pose at time t.

2) Data Synchronization: It is a necessity to synchronize the data from the accelerometers to the optical motion capture system to obtain the mapping between $\mathbf{c}_{ti}$ and $\mathbf{q}_{tj}$. This is critical because the independent transmission of each sensor. Moreover, data from each sensor is received with variable frame-rate owing to packet loss in the wireless environment. To solve these two issues, all data received from the sensors are marked with the sensor ID and placed in a common buffer. A snapshot of the buffer is taken each frame and the data of sensor is constructed with the most recently received data. After a snapshot is taken, the buffer is overwritten if new data

arrives. Typically, there should be only one sensor frame in the buffer when taking snapshots. However, if the terminal failed to receive data from any sensor during this time period (or the buffer is empty), then the previously received frame is considered again. If there are more than one sensor frame, we use the average of all frames in the buffer. This way the sensors can be synchronized at a constant frame-rate of 60Hz to match the frame rate of the motion capture system.

The next step is to synchronize the motion capture data with the sensor data. For this step, we ask a performer to strike fists before and after performing any action. We use this striking event to synchronize the sensor data and motion capture data, by aligning the spike in the sensor readings with the frame in motion capture data when two fists touch each other.

3) Data Pre-Processing: Before the synchronized data can be used as our animation database, we need to do a pre-processing for model learning and motion synthesis. We use quaternions for joint rotation representation so that congruent angles (e.g. 0° and 360°) are represented using the same numerical values. Noise in optical motion capture data due to marker occlusion is removed, and the synchronized accelerometer sensor's data is removed at the same time, since this noise reduction is crucial for the animation quality.

Noise also happens in the sensor data because of the wireless environment we use. To maintain a high bandwidth, we use Bluetooth receivers which are very sensitive, and it is not unusual to find a few arbitrary values beyond the range of what we expect to get from the sensors. This kind of noise is automatically detected and replace by quantities which are estimated (by least square function) from the neighboring data. Even using this automatic method, the outlier in a motion still exists. We also need to remove the noise as well as its synthesized optical motion data.

B. Result and evaluation

Relying on the database built in the data collection, we test the performance of our system with two subjects performing various actions. The sensors signals are used as input in our system to produce our on-line animation. The synthesized motion are visually compared with the recorded video. Figure 4 shows the results for two synthesized actions, tennis forehand and backhand. The results clearly show that our system can clearly capture the poses of the subjects with the sensor signals as control signals.

We also perform an end-to-end evaluation to numerically measure the accuracy of our system. During the subjects were doing actions with Wii controllers, their high quality motions were recorded using a Vicon optical motion capture system. As ground truth motions , the recorded high quality motions are compared against the synthesized motion frame by frame. We then use the normalized Root Mean Square (RMS) distance e to quantitatively measure the difference. e, as defined below, is able to measure the error of degree of freedom per angle.

$$e = RMS(\tilde{\mathbf{q}}_k, \mathbf{q}_k) = \sqrt{\frac{\sum_{i=1}^{n}(\tilde{\mathbf{q}}_{k,i} - \mathbf{q}_{k,i})^2}{n}}, \qquad (9)$$

where k is the frame index, $\tilde{\mathbf{q}}_k$ is the synthesize motion, $\mathbf{q}_k$ is the ground truth motion and $\mathbf{q}_{k,i}$ is the i^{th} dimension of $\mathbf{q}_k$. The unite of e is degree of freedom per angle. Figure 5 shows a numerical comparison of the synthesized motions with the corresponding ground-truth motion. The results of visual and quantitative comparisons show that our low cost system generates motions with the quality equivalent to that of an expensive optical motion capture systems.

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

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

C. Numerical Comparison

The performance of the system rely on the ability to represent human motion in a low-dimensional space. Without this low dimensional representation, the clustering algorithm has difficulty to cluster high dimensional data. Our previous work [17] depends on PCA to reduce data dimensionality, however, as we discussed in Section II, a global linear method, such as PCA and MDS, it is hard to model a heterogeneous database which is possibly nonlinear. In comparison, nonlinear dimensionality reduction, e.g. LLE and Isomap, compute low dimensional embedding with neighborhood relationship preserving of high dimensional data.

In this section, we compare the performance of our nonlinear manifold learning algorithm with our previous work [17] using linear models. Figure 3 plots the RMS errors of synthesized motion for both methods. It shows that our method creates more accurate results than the previous work and demonstrates LLE is a more appropriate technique to understand our data.

VI. Discussion

In this paper, we present a nonlinear manifold learning framework to control the animation of 3D avatar in real-time. We utilize low-cost 3D accelerometer sensors as control signal and a high quality database consisting of prerecorded human motion and sensors data. Our data-driven approach includes two steps: offline model learning from existing high quality motion database, and online motion synthesis from the control signal. The model learning step enable us to explore the nonlinear structure of motion data and build piecewise models on variety of clusters with different features. The learned models are later used to produce a realistic motion in real-time. We have shown the effectiveness of our framework by performing an End-to-End evaluation.

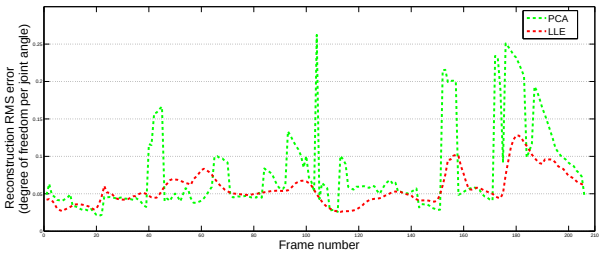

Fig. 3: *Comparison of construction RMS errors of synthesized tennis backhand motions from low-dimensional control signals. The average RMS error for PCA is 0.078 degree/joint per frame while average error of LLE is 0.057 degree/joint per frame. All the test motions are not in the database.*

In our experiments, we find our approach can't guarantee 100 percentage of correct cluster detection. This problem results partly from the noise in the wireless sensor signal. Another possibility is that our database is not dense enough that data in a cluster can't form a perfect Gaussian distribution. In addition, the sparse database also result in limitation of type of behaviors that can be synthesized. However, our design is scalable and can handle larger databases without performance degradation. Using a sufficient number of examples we can synthesize a larger variety of human motions.

Like most of the piecewise model based motion synthesis, our approach suffers from the non-smoothness problem during motion synthesis. Our algorithm mainly learn spatial knowledge for the models while ignore the temporal relationship among data pointers. To Learn sufficient knowledge from the database should help us improve the smoothness of synthesized motion.

ACKNOWLEDGMENTS

We would like to thank Shawn Bohner and Ruth Bohner for their professional performance during motion capture session.

REFERENCES

[1] N. I. Badler, M. J. Hollick, and J. P. Granieri. Real-time control of a virtual human using minimal sensors. *Presence*, 2(1):82–86, 1993.

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

[3] S. R. Carvalho, R. Boulic, and D. Thalmann. Interactive low-dimensional human motion synthesis by combining motion models and pik. *Computer Animation and Virtual Worlds*, 18(4-5):493–503, 2007.

[4] J. Chai and J. K. Hodgins. Performance animation from low-dimensional control signals. *ACM Trans. Graph.*, 24(3):686–696, 2005.

[5] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Chapman & Hall, London, 1994.

[6] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun. Performance capture from sparse multi-view video. In *Proceedings of ACM SIGGRAPH 2003*, pages 1–10, New York, NY, USA, 2008. ACM.

[7] M. Dontcheva, G. Yngve, and Z. Popović. Layered acting for character animation. In *Proceedings of ACM SIGGRAPH 2003*, pages 409–416, New York, NY, USA, 2003. ACM.

[8] A. Elgammal and C.-S. Lee. Inferring 3d body pose from silhouettes using activity manifold learning. *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, 2:II–681–II–688 Vol.2, June-2 July 2004.

[9] T. Jaeggli, E. Koller-Meier, and L. V. Gool. Multi-activity tracking in lle body pose space. In A. Elgammal, B. Rosenhahn, and R. Klette, editors, *Human Motion - Understanding, Modeling, Capture and Animation, Proceedings*, volume 4814 of *Lecture Notes in Computer Science*, pages 42–57. Springer-Verlag Berlin, Berlin, 2007.

[10] G. Liu, J. Zhang, W. Wang, and L. McMillan. Human motion estimation from a reduced marker set. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 35–42, New York, NY, USA, 2006. ACM.

[11] S. Oore, D. Terzopoulos, and G. Hinton. A desktop input device and interface for interactive 3D character animation. In *Proceedings of Graphics Interface 2002*, pages 133–140, May 2002.

[12] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–+, 2000.

[13] A. Safonova, J. K. Hodgins, and N. S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *Acm Transactions on Graphics*, 23(3):514–521, 2004.

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

[15] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–+, 2000.

[16] D. Vlasic, R. Adelsberger, G. Vannucci, J. Barnwell, M. Gross, W. Matusik, and J. Popović. Practical motion capture in everyday surroundings. *ACM Transactions on Graphics*, 26(3):35, 2007.

[17] L. Xie, M. Kumar, Y. Cao, D. Gracanin, and F. Quek. Data-driven motion estimation with low-cost sensors. *The 5th IET Visual Information Engineering 2008 Conference*, 290(5500):600–606, 2008.

[18] M. Yeasin and B.Bullot. *Comparison of linear and non-linear data projection techniques in recognizing universal facial expressions*. Proceedings of the International Joint Conference on Neural Networks. Ieee, New York, 2005.

[19] K. Yin and D. K. Pai. Footsee: an interactive animation system. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 329–338, Aire-la-Ville, Switzerland, 2003. Eurographics Association.

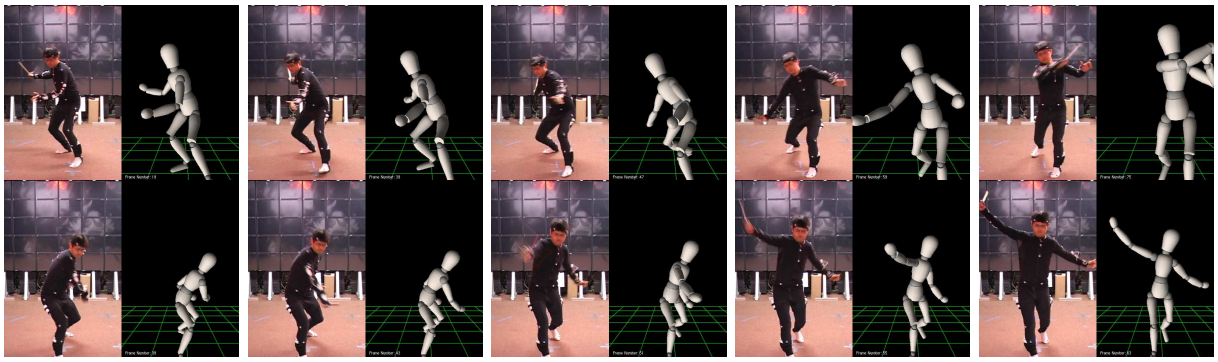

Fig. 4: *Four different actions (one in each row) synthesized by our system. Each frame shows on the left side the actual pose and on the right side the synthesized pose.*

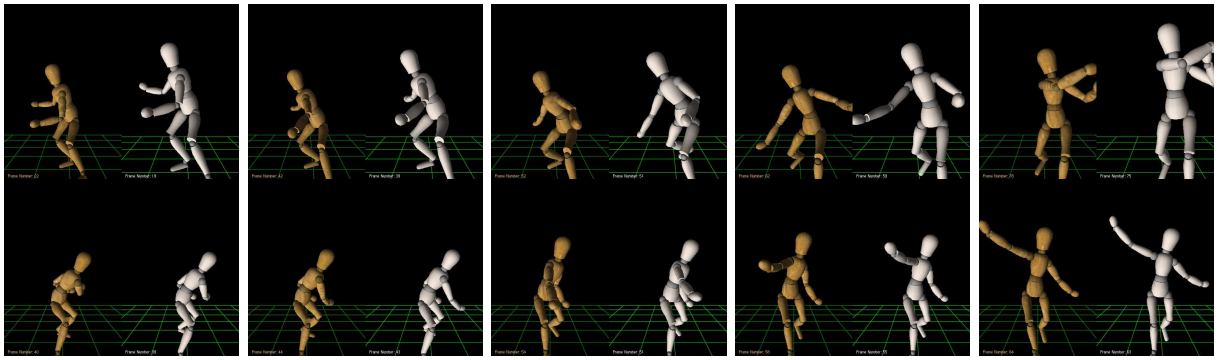

Fig. 5: *motion compared to the ground truth. Each frame shows on the left side the ground truth motion and on the right side the synthesized motion.*