In Computer Animation and Virtual Worlds, 24(6), 565-576, 2013
This version is the author's manuscript.

Human Motion Reconstruction from Sparse 3D Motion Sensors Using Kernel CCA-Based Regression

Jongmin Kim, Yeongho Seol, and Jehee Lee

Movement Research Lab.,

School of Computer Science and Engineering,

Seoul National University

Seoul 151-742, Republic of Korea

Tel. (+82)2 880 1864 Fax. (+82)2 871 4912

email: {kimjongmin,seolyeongho,jehee}@mrl.snu.ac.kr

Abstract

This paper presents a real-time performance animation system that reproduces full-body character animation based on sparse 3D motion sensors on a performer. Producing

faithful character animation from this setting is a mathematically ill-posed problem because input data from the sensors are not sufficient to determine the full degrees of freedom of a character. Given the input data from 3D motion sensors, we select similar poses from a motion database and build an online local model that transforms the low-dimensional input signal into a high-dimensional character pose. A regression method based on kernel CCA (Canonical Correlation Analysis) is employed because it effectively handles a wide variety of motions. Examples show that various human motions are naturally reproduced by the proposed method.

Keywords: motion reconstruction, performance animation, motion capture, sensor, machine learning

1 Introduction

Recent advancements in gaming interfaces such as Nintendo's Wiimote and Sony's Move have improved the user's gaming experience by allowing various performance animation applications in a virtual environment. Users can play virtual sports, dance, and engage in other activities using their bodies as a controller. Such applications usually depend on the input data from sparse body parts (e.g., two acceleration sensors on each hand, tracking of a vision based marker) and therefore cannot accurately reproduce the full-body pose of a performer. Despite the various potential applications of human motion based animation, the reconstruction of an accurate full-body pose from a sparse input setting remains a challenging problem.

This paper presents a real-time performance animation system that can be used to realize full-body animation based on sparse 3D motion sensors attached onto a performer's body. We select 3D motion sensors as an input device because they are free from occlusion, a typical problem associated with optical motion capture systems. However, it is difficult to implement a performance animation system using a sparse set of 3D motion sensors because the system input is not sufficient to determine a high-dimensional full-body character pose. We therefore consider a data-driven approach using motion capture data to predict full-body motion.

The concept of utilizing motion capture data as prior knowledge has been widely exploited

in performance animation systems to transform low-dimensional data into high-dimensional character animation. Previously, data-driven online animation systems were addressed in [1, 2]. They set up an online linear local model for every frame using a pre-captured motion database to handle various motions. Although previous methods generated convincing results, reconstructing accurate motion sequences based on linear statistical models is limited by the intrinsic nonlinear relationship between 3D motion sensors and character animation.

We employ a nonlinear statistical model, kernel CCA-based regression [3], which fully explains the nonlinear relationship between 3D motion sensors and character pose. Our method also has an advantage in terms of its ability to express a wide range of character animation with an insufficient motion dataset. Kernel CCA-based regression learns the nonlinear structures of high-dimensional motion data and forms a connection between the user's input from 3D motion sensors and the full-body character pose. At run-time, the system promptly searches for motion examples that are similar to the input motion in the motion database. The collected examples are used to train kernel CCA-based regression as an online local model and the system input then determines the character pose using the learned model.

The resulting character animation is shown to be a convincing recreation of the performer's input motions with examples of boxing, walking, and table tennis. We evaluate the performance of our system with different sensor setups and query metrics to generate the most desired results. We also show that our performance animation system can generate more accurate character animation than previous methods.

The remainder of this paper is organized as follows. In section 2, we first discuss previous research, after which present a system overview in section 3. Section 4 discusses the sensor setups and the calibration process. In section 5, we describe how to reconstruct human motions using kernel CCA-based regression, and we present experimental results in Section 6. We conclude the paper in section 7.

2 Related Work

Reconstructing human motions of characters is an important issue in computer animation and has long been studied. Researchers have obtained raw human motion using magnets, optical devices, and video cameras. The Vicon motion capture system is used to record the 3D movement of a user who wears a skin-tight garment with dozens of attachable optical markers. Badler et al. [4] employed four magnetic sensors and an inverse kinematics to control the standing pose of a full-body character. Shin et al. [5] mapped the actions of a performer to a virtual character using a magnetic motion capture system. Yin and Pai [6] proposed an animation system that controls avatars with a foot pressure sensor pad. However, a problem arises when attempting to generate accurate upper-body motions with restricted foot pressure information. Lee et al. [7] used a vision-based technique to capture full-body motion from a single camera. Chai and Hodgins [1] reconstructed human motion using a small set of optical markers and two video cameras.

Low-cost hardware to capture human motion was recently developed for game interfaces.The Wiimote controller by Nintendo is a popular interface that uses an accelerometer and optical sensors to allow the system to recognize human gestures. Sony's play Station Move is another motion-sensing controller that tracks the three-dimensional position of the controller using inertial sensors. Microsoft Kinect can generate 3D avatars without attachable hardware. While this hardware can provide an exciting game experience in a constrained environment, it is not applicable for general-purpose motion retargeting. Our sensor-based performance animation system shares the affordable cost advantage of these input devices while remaining applicable to a wider range of situations.

Slyper and Hodgins [8] created a system that searches a motion database with low-cost accelerometers and then plays the best match of motion sequences from a database. Tautges et al. [9] introduced a performance animation system that generates full-body character animation using four accelerometers. Additionally, Liu et al. [2] introduced an online animation system with a small number of inertial sensors as a performance animation interface. Our approach has an environment setup similar to that in [2]; however, we suggest an alternative data-driven approach for performance animation. We utilize kernel CCA-based regression concept, introduced in [3], which achieves strong interpolation and extrapolation capabilities by applying a kernel trick method to CCA-based regression scheme presented in [10].

Many motion synthesis studies have utilized statistical analyses from motion capture data. Brand and Hertzmann [11] employed a Gaussian mixture model to estimate the distribution

of given motion data, inferring the desired poses by computing several parameters. Shin and Lee [12] proposed an interactive motion synthesis framework for use in low-dimensional space. Liu et al. [13] exploited a probabilistic PCA (Principle Component Analysis) to predict full-body human motions from a set of principle markers. Chai and Hodgins [14] constructed a statistical dynamic model to generate motion sequences satisfying given user constraints. Wei and Chai [15] introduced a system that interactively controls a 3D human character using a factor analysis with millions of motion examples. Our work is motivated by these successful statistical approaches. We consider the idea of constructing a nonlinear statistical model as presented in earlier work [3] to handle the nonlinear relationship between 3D motion sensors and the human pose.

Motion reproduction based on statistical models encounters difficulties when the size of the motion database becomes bigger. Grochow et al. [16] applied the GPLVM (Gaussian Process Latent Variable Model) using motion examples and a set of kinematic constraints to create realistic human motion. Although their approach can generate faithful animation results, it has difficulties when synthesizing high-quality animation with heterogeneous motions due to intrinsic problems associated with global modeling approaches. For faithful local modeling, an efficient retrieval method is required. Many researchers have studied local modeling approaches to address this problem. To construct a faithful local model, efficient retrieval of motion from the database is required. Müller [17] introduced the content-based motion retrieval method. Chai and Hodgins [1] proposed the approach of searching for similar pose

examples in a heterogeneous motion database using a precomputed neighborhood graph, and generates the resulting pose using a local model at every frame. In another work [9], Tautges et al. suggested a more efficient nearest-neighbor search method that describes a crucial temporal coherence for a large scale motion database. Their approach is based on an earlier study [18] that proposed a kd-tree-based search method to find exact neighborhood poses in a large motion database. For the retrieval of motions, we incorporate the kd-tree-based search and temporal features such as velocity and acceleration. With this approach, we retrieve similar pose examples to the performer's movement and use them to build a nonlinear local model for human motion reconstruction.

3 Overview

Our system uses a motion database from five 3D motion sensors to reconstruct full-body character poses. As the input data from the sensors do not provide enough information to restore an accurate full-body character, we also employ the motion database. Based on existing motion data, we build a prediction model that produces a full-body character pose.

Figure 1 shows the overall workflow of the proposed performance animation system. Given the input data from 3D sensors, calibration is performed to obtain consistent input data with respect to the motion capture database. Our system rapidly searches for pose examples that are similar to the given system input and builds kernel CCA-based regression model. We

then estimate the whole-body character pose with the learned model. A post-processing step reduces noise in the retargeting result.

4 Sensor Data and Calibration

We use the InterSense IS-900 system [19], which offers robust real-time tracking. Figure 2 shows our tracking system setup. Each tracking device (3D motion sensor) has a gyroscope and an accelerometer to measure the angular and acceleration rate values, respectively. Using the measured values, the processor computes 6-DOF tracking data, which are employed in the subsequent stages. The performer wears sparse 3D motion sensors on the target joints, which are the pelvis, right hand, left hand, right ankle, and left ankle in this study (Figure 2), and is allowed to move freely in front of the IS-900 processor within its maximum sensing area of 20 m^2. The system provides absolute positional accuracy of 2-5mm and orientation accuracy of 1°(yaw) and 0.4°(pitch, roll).

The 6-DOF tracking data from the 3D motion sensors must be transformed from the world coordinate frame to the virtual coordinate frame. At the beginning of every capture instance, we ask the performer to begin with a standard "T-pose" to calibrate the information from the 3D motion sensors. We transform the positions and orientations of the i-th 3D motion sensor in world coordinates, $\mathbf{p}_s^i$ and $\mathbf{q}_s^i$, into their corresponding values in virtual coordinates, $\tilde{\mathbf{p}}_s^i$

and $\tilde{\mathbf{q}}_s^i$, as follows:

$$\tilde{\mathbf{p}}_s^i = \mathbf{T}_p^i \mathbf{p}_s^i + \mathbf{b}^i, \;\; \tilde{\mathbf{q}}_s^i = \mathbf{T}_q^i \mathbf{q}_s^i, \tag{1}$$

where $\mathbf{T}_p^i \in \mathbb{R}^{3\times3}$ and $\mathbf{b}^i \in \mathbb{R}^{3\times1}$ denote the linear transformation matrix and translational component of the i-th 3D motion sensor, respectively. Here, $\mathbf{T}_q^i$ denotes the transformation of the quaternion to orient the sensor. More precise calibration can be achieved with a non-linear mapping model, although this significantly increases the overall number of calibration parameters. In contrast to the nonlinear model, our calibration process is easy-to-implement and showed a reasonable level of accuracy in experiments. With ten different subjects, we observed that orientation errors are negligibly small, within 2°, and position errors are within 3 cm.

5 Motion Synthesis

In this section, we describe the construction process of the online local model. Kernel CCA-based regression serves as the online local model due to its extrapolation capacity. Character animation is generated using the learned model and temporal noise is reduced in the post-processing stage.

5.1 Online Local Model

We suggest an online local modeling approach that can produce various types of motion sequences, i.e., types that cannot be easily generated by a global modeling approach. For a global analysis, the connection between 3D motion sensors and a certain pose is determined by processing the entire set of heterogeneous motion capture data. There is an important limitation, however, when using this method. Prediction errors associated with regression can increase dramatically such that the system can scarcely generate the desired motion sequences.

To build a local model at every frame, we first choose the training data for the model. In the motion database, the k-nearest poses of the current input from the pose are chosen as the training data. We employ the position, orientation, velocity, angular velocity, acceleration, and angular acceleration of sensors for the following query metrics:

$$\begin{aligned} & c_p \sum_{i=1}^{n} \|\tilde{\mathbf{p}}_s^i - \mathbf{p}_m^i\|^2 + c_q \sum_{i=1}^{n} \|\tilde{\mathbf{q}}_s^i - \mathbf{q}_m^i\|^2 + c_v \sum_{i=1}^{n} \|\tilde{\mathbf{v}}_s^i - \mathbf{v}_m^i\|^2 + \\ & c_w \sum_{i=1}^{n} \|\tilde{\mathbf{w}}_s^i - \mathbf{w}_m^i\|^2 + c_d \sum_{i=1}^{n} \|\tilde{\mathbf{d}}_s^i - \mathbf{d}_m^i\|^2 + c_e \sum_{i=1}^{n} \|\tilde{\mathbf{e}}_s^i - \mathbf{e}_m^i\|^2, \end{aligned} \tag{2}$$

where $\tilde{\mathbf{v}}_s^i$, $\tilde{\mathbf{w}}_s^i$, $\tilde{\mathbf{d}}_s^i$, and $\tilde{\mathbf{e}}_s^i$ represent the velocity, angular velocity, acceleration, and angular acceleration of the i-th 3D motion sensor in the calibration pose, respectively. Here, $\mathbf{v}_m^i$,$\mathbf{w}_m^i$,$\mathbf{d}_m^i$, and $\mathbf{e}_m^i$ represent the same types of data pertaining to the i-th end-effector of a pose in the motion database, respectively. c_p, c_q, c_v, c_w, c_d, and c_e are the weight values for each respective term. In Equation (2), temporal features (e.g, the velocity, acceleration,

angular velocity, and angular acceleration) are incorporated to better choose similar pose examples to the performer's movement. For example, we can correctly identify the direction of swing motion of the performer when playing table tennis by considering the temporal features.

As the total number of motions in the motion database grows, the search time increases accordingly. In order to reduce the computational cost at the search time, we store the positions, velocities, and orientations of n target joints in a kd-tree. At run-time, our system searches for k-nearest motion examples (k = 8 in our experiments) efficiently using the kd-tree [20].

Training data offer both input and output data for the construction of the online local model. From the collected poses, we extract the positions and orientations of n target joints, which are then used as the input of the model. The input vector $\mathbf{x} \in \mathbb{R}^{7n}$ consists of the xyz position and quaternion of the joints. All joint quaternions of the character $\mathbf{y} \in \mathbb{R}^{76}$ are also extracted so that they can be used as the output of the model. As a training model, we employ kernel CCA-based regression method, which is described in detail in the following section.

5.2 Kernel CCA-based Regression

Producing faithful character animation is a mathematically ill-posed problem because the input data from the sensors are not sufficient to determine the full degrees of freedom of

a character. We choose a data-driven nonlinear statistical approach, kernel CCA-based regression, to find a mapping function between the low-dimensional input data and a high-dimensional character pose. Compared to other linear models, kernel CCA is capable of extrapolation. Our system can faithfully generate reasonable animation results when a few similar motions exist.

CCA is a machine learning algorithm that maximizes the correlations between training inputs and outputs by extracting the meaningful features of both training inputs and outputs. Once the correlation has been maximized with the transformed training data, it becomes possible to efficiently avoid a large amount of error, which is typically associated with the regression process. We formulate this into a nonlinear problem while utilizing the kernel method in CCA [21], as there is a very clear nonlinear relationship between the training input $\mathbf{x}$ and output $\mathbf{y}$. The following equation represents the nonlinear transformation of the training input x to a vector $\phi(\mathbf{x})$ in the higher-dimensional feature space:

$$\phi : \mathbf{x} = (x_1, \ldots, x_n) \rightarrow \phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_N(\mathbf{x})) \quad (N > n)$$

Given m pairs of training data, $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ and $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m)$, the equation of kernel CCA with the correlation coefficient ρ is defined as follows:

$$\max_{(\hat{\mathbf{x}}, \hat{\mathbf{y}})} \rho = \max_{(\hat{\mathbf{x}}, \hat{\mathbf{y}})} \frac{\hat{\mathbf{x}}^T \mathbf{H} \mathbf{F} \hat{\mathbf{y}}}{\sqrt{\hat{\mathbf{x}}^T \mathbf{H}^2 \hat{\mathbf{x}} \hat{\mathbf{y}}^T \mathbf{F}^2 \hat{\mathbf{y}}}} \tag{3}$$

where $\mathbf{H} = \phi^T(\mathbf{X})\phi(\mathbf{X})$ and $\mathbf{F} = \phi^T(\mathbf{Y})\phi(\mathbf{Y})$ represent the nonlinear version of the training data in matrix form. The derivation of Equation (3) is described in [21]. The i-th basis of

$\phi(\mathbf{X})$ and $\phi(\mathbf{Y})$ can be represented as $\phi(\mathbf{X})\hat{\mathbf{x}}_i$ and $\phi(\mathbf{Y})\hat{\mathbf{y}}_i$, respectively, where $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{y}}_i$ are the i-th coefficient vectors with respect to the training data pairs [21]. From the total m basis sets, we select l basis sets to reduce the dimension of the input training data. The extracted sets, $\{\hat{\mathbf{x}}, \hat{\mathbf{y}}\}$ can be represented as matrices, $\hat{\mathbf{X}} = (\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, ..., \hat{\mathbf{x}}_l)$ and $\hat{\mathbf{Y}} = (\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, ..., \hat{\mathbf{y}}_l)$.

Although it is difficult to determine the transformation function ϕ explicitly, we can solve this problem using an approach known as the kernel trick. The kernel trick uses a kernel function with well-defined inner products of vectors in the feature space. The kernel trick method can be applied to kernel CCA (Equation (3)) as the equation contains the values of the kernel for every instance of the training data [21]. We utilize the Gaussian kernel function, $k(\mathbf{x}, \mathbf{x}_j) = \exp\left(\frac{-\|\mathbf{x}-\mathbf{x}_j\|^2}{2\sigma^2}\right)$. The standard deviation of the Gaussian kernel is automatically determined by the standard deviation of the training input data. The reduced training input data can be computed as follows: $\bar{\mathbf{x}} = \hat{\mathbf{X}}^T\tilde{\mathbf{x}}$, where $\tilde{\mathbf{x}} = \begin{pmatrix} k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_m) \end{pmatrix}^T$. The training output data in the reduced dimension $\bar{\mathbf{y}}$ can be computed similarly.

We can obtain the matrix $\mathbf{A}$, which transforms the reduced input to the reduced output with the equation below:

$$\min_{\mathbf{A}} \sum_{i=1}^{m} \|\mathbf{A}\bar{\mathbf{x}}_i - \bar{\mathbf{y}}_i\|^2 \tag{4}$$

We can also obtain the transformation matrix $\mathbf{B}$, which transforms the reduced output to the character pose $\mathbf{y}$ with the following equation:

$$\min_{\mathbf{B}} \sum_{i=1}^{m} \|\mathbf{B}\bar{\mathbf{y}}_i - \mathbf{y}_i\|^2 \tag{5}$$

We use the efficient LU solver [22] to process these two regressions. The desired pose is estimated by multiplying $\mathbf{B}\mathbf{A}\hat{\mathbf{X}}^T$ by the kernelized input vector $\tilde{\mathbf{x}}$.

5.3 Motion Post-processing

While the character poses produced in the previous section are plausible in each frame, they may be temporally inconsistent. Because data from motion sensors are prone to temporal jitter, a smoothing process needs to be applied in order to achieve high-quality animation. To generate a smooth animation result, two smoothing methods are jointly applied. In the first method, more input training data for the prediction model are used. We additionally use the nearest poses of the previous frame on top of those of the current frame to build the regression model. Second, we blend the resulting character poses of previous frames with the current character pose over all joints. The interpolating function is defined as follows:

$$\mathbf{y} = \mathbf{y}_t \otimes ((\mathbf{y}_{t-1} \otimes (\mathbf{y}_{t-2} \oslash \mathbf{y}_{t-1})^{\beta}) \oslash \mathbf{y}_t)^{\alpha}, \tag{6}$$

where $\mathbf{y}_t$ is the reconstructed pose in the current frame. Here, $\mathbf{y}_{t-1}$ and $\mathbf{y}_{t-2}$ denote the reproduced poses in the previous two frames. We use the mathematical notations introduced in [23, 24]. In our experiments, the blending weights were α = 0.3 and β = 0.4. Finally, analytical inverse kinematics is applied to the character so that its end-effectors will accurately follow the position and orientation of the 3D motion sensors.

6 Experimental Results

We present the performance animation results of various motions and evaluate our system with different sensor setups and query metrics, and finally compare our system with three previous performance animation approaches. Motion data examples of boxing (7,233 frames), table tennis (4,271 frames), and walking (1,317 frames) were employed in these experiments.

Figures 10 and 11 depict the performance animation results given the performer's motion of boxing and table tennis, respectively. Both motions are consistently reconstructed using our method. In the boxing example, various types of punches are generated by our method, such as jabs, straights, hooks, and uppercuts. For the table tennis sequence, plausible motions such as forward/backward attacks and side-walking are successfully generated. Our system is capable of approximately 80 fps on a standard PC with an Intel Core i7 CPU and 16GB of memory. We include an accompanying video of the full-animation sequences.

Only one sensor on the right hand and five sensors on the target joints of the performer were used, and a side-by-side comparison was made. With only one sensor, the root position of the character automatically follows the destination of the ball without application of the inverse kinematics to the feet. Our system generates a more natural motion sequence with five sensors than with only one sensor.

Regarding the walking and boxing motions, we observed the resulting errors according to

variation of the sensor setups to select appropriate sensors. In this experiment, we use c_p = 0.6, c_q = 0.2, c_v = 0.2, and set all other weights of the query metric to be zero. In Figure 3, the vertical axis of the graph represents all possible types of combinations while the horizontal axis shows the amount of error. We tested 15 possible sensor setup combinations. For walking motions, the placement of three sensors (RA, LA, and PEL) gave a suitable result compared to when more than three sensors were used. For the boxing motion data, with three sensors, their placement positions (RH, LH, and PEL) led to a well-generated result. In this experiment, we found that selecting the highest number of the most movable joints as the system input for each movement is important when seeking to minimize the number of sensors and reconstruct the desired motion sequences. As expected, more natural motion was generated when we added more sensors on the end-effectors on the limbs.

We evaluated the reconstruction error of different combinations of signals from 3D motion sensors to define the query metrics (Section 5.1). In Figure 4, the horizontal axis shows the amount of error while the vertical axis represents the eight possible combinations of features from 3D motion sensors for the query metrics. These results show that the combination of position and orientation can generate more accurate results than with the independent use of position or orientation. For instance, in order to correctly identify the performer's forward swing during table tennis, we need both position and orientation features. Even in the same position, the type of swing may be different depending on the orientation of the hand. We also considered the velocity, angular velocity, acceleration, and angular acceleration as

features for the query metrics to distinguish the direction of a given pose (e.g., forward and backward swings in table tennis). Based on these experiments, we exploit all 3D signals as query metrics to select the appropriate training poses. The weight of each term of the query metrics (Section 5.1) is determined to be proportional to the inverse of the total amount of error.

We compared the performance of our algorithm with three baseline methods: inverse kinematics, an online weighted blending method, and PCA-based regression. The inverse kinematics simply matches all measured end-effector configurations to reproduce character poses. The online weighted blending method generates new poses on the fly via weighted blending of nearby k-neighbor poses (Section 5.1). PCA-based regression reproduces the motion sequences by utilizing two-step regression (Section 5.2). To test the extrapolation capacity of our method, we established a motion database with extremely sparse motion examples of total 85 frames (45 frames from table tennis motion and 40 frames from boxing motion), and reconstructed a target test pose that is not in the motion database. The graphs show that our method generates more accurate results than the other three methods. Figure 5 shows a frame-by-frame performance comparison of the given motion sequences.

The extrapolation capacity of kernel CCA-based regression approach is evaluated by comparing our results with those from the baseline methods when the input motion sequences do not exist in the sparse dataset (table tennis, 40 frames of motion) as shown in Figure 6. All character poses are reconstructed without application of the inverse kinematics to the

end-effectors. In the event of insufficient motion capture data, our method can generate reliable results due to its extrapolation ability. Ground truth data are acquired from the motion capture data. Figure 7 shows the closest neighbor poses and the final synthesized pose.

In kernel CCA stage, we experimented with another type of kernel function. We chose the polynomial kernel, $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + c)^p$, where p represents the degree of the polynomial and c is a constant value. The difference between the Gaussian and polynomial kernel is not significant in the predicted results. In most cases, the Gaussian kernel works well in all examples while the polynomial kernel produces slightly worse results when p is smaller than five. Figure 8 describes a comparison of the reconstruction error that is obtained with the use of different types of kernels.

The reconstruction error was compared with different combinations of input signals for learning the prediction model: position, orientation, and a combination of the two. The combination of position and orientation generated the most accurate results among them, because increasing the number of independent features typically leads to better performance, up to a point where the dimension of the input does not suffer the "curse of dimensionality" [25]. Figure 8 represents the performance comparison for different combinations of sensor signals.

We evaluated the motion reconstruction performance with respect to different numbers of kernel CCA bases. Given a large number of bases for kernel CCA, our model generated an

over-fitting pose with more reconstruction errors. The most accurate pose was achieved with five bases. Figure 9 shows the reconstruction error for different number of bases.

The time complexity of our k-nearest search algorithm in d dimensions with n points and error bound e can be represented as $\mathcal{O}((c_{d,e} + kd)\log n)$ in [26], where $c_{d,e}$ is a constant term that is related to d and e. Our search algorithm efficiently finds training examples in large databases. The average query time (from 20 trials) was fast enough for real-time performance (Table 1). Theoretically, our system can handle up to a million of pre-recorded human poses.

7 Discussion

We have presented a robust real-time performance animation system based on kernel CCA-based regression framework. Kernel CCA suitably explains the nonlinear relationship between 3D motion sensors and character animation while generating more accurate character animation than previous statistical linear models. The improved performance of our approach was shown through an experimental evaluation.

To generate various types of motion sequences, we employ an online local model instead of a global one. After the online local model is established, our system generates new poses by means of simple matrix-vector multiplications; the matrix represents a linear transformation computed in a regression process and the vector represents the input data from 3D motion

sensors. In addition, combining the nearby poses of the input motions of the current and previous frames' with effective temporal interpolation produces temporally smooth results. Temporal filtering works in real-time with a small delay. We demonstrated the robustness of our system by showing performance animation examples of boxing, walking, and table tennis.

To achieve more accurate results in the post-processing stage, a nonlinear programming approach with several equality constraints would be required. However, the fully nonlinear method runs too slow for real-time application. Given these limitations, our system provides a good balance between performance and precision/generalization, thus making it practical for various situations.

Similar to most sensor-based performance animation systems, our system may miss the position of 3D motion sensors when a performer moves rapidly. Although we can successfully approximate short-term missed sensors using a temporal smoothing function in the post-processing step, markers that are not tracked for a long time cannot be approximated. In this case, we should temporally stop the system and wait until reliable input data are provided.

While we concentrated on a sensor-based performance animation system in this paper, our system can easily be extended by combining complementary types of input (e.g., Microsoft Kinect, the Vicon capture system, or stereo cameras). The use of multiple input sources would minimize the amount of missing data and improve the animation quality. We would need however to solve additional problems in order to achieve this goal, such as accurate

synchronization and the appropriate selection of a model. This would be an interesting future research direction.

Acknowledgements

We would like to thank all the members of the Movement Research Laboratory for their help. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Education, Science, and Technology (No. 2012-0001242 and No. 2012-0000789).

References

[1] Jinxiang Chai and Jessica K. Hodgins. Performance animation from low-dimensional control signals. *ACM Transactions on Graphics (SIGGRAPH 2005)*, pages 686–696.

[2] Huajun Liu, Xiaolin Wei, Jinxiang Chai, Inwoo Ha, and Taehyun Rhee. Realtime human motion control with a small number of inertial sensors. In *Symposium on Interactive 3D Graphics and Games*, pages 133–140, 2011.

[3] Wei-Wen Feng, Byung-Uck Kim, and Yizhou Yu. Real-time data driven deformation using kernel canonical correlation analysis. *ACM Transactions on Graphics (SIGGRAPH 2008)*, 27(3):1–9.

[4] Norman I. Badler, Michael J. Hollick, and John P. Granieri. Real-time control of a virtual human using minimal sensors. *Teleoperators and Virtual Environments*, 2:82–86, 1993.

[5] Hyun Joon Shin, Jehee Lee, Sung Yong Shin, and Michael Gleicher. Computer puppetry: An importance-based approach. *ACM Transactions on Graphics*, pages 67–94, 2001.

[6] KangKang Yin and Dinesh K. Pai. Footsee: an interactive animation system. ACM SIGGRAPH / Eurographics Symposium on Computer Animation, pages 329–338, 2003.

[7] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics (SIGGRAPH 2002)*, pages 491–500.

[8] Ronit Slyper and Jessica Hodgins. Action capture with accelerometers. ACM SIGGRAPH / Eurographics Symposium on Computer Animation, pages 193–199, 2008.

[9] Jochen Tautges, Arno Zinke, Björn Krüger, Jan Baumann, Andreas Weber, Thomas Helten, Meinard Müller, Hans-Peter Seidel, and Bernd Eberhardt. Motion reconstruction using sparse accelerometer data. *ACM Transactions on Graphics*, pages 18:1–18:12, 2010.

[10] Haoda Huang, KangKang Yin, Ling Zhao, Yue Qi, Yizhou Yu, and Xin Tong. Detail-

preserving controllable deformation from sparse examples. *IEEE Transactions on Visualization and Computer Graphics*, pages 1215–1227, 2012.

[11] Matthew Brand and Aaron Hertzmann. Style machines. *ACM Transactions on Graphics (SIGGRAPH 2000)*, pages 183–192.

[12] Hyun Joon Shin and Jehee Lee. Motion synthesis and editing in low-dimensional spaces: Research articles. *Comput. Animat. Virtual Worlds*, pages 67 – 94, 2006.

[13] Guodong Liu, Jingdan Zhang, Wei Wang, and Leonard McMillan. Human motion estimation from a reduced marker set. ACM Symposium on Interactive 3D graphics and games, pages 35–42, 2006.

[14] Jinxiang Chai and Jessica K. Hodgins. Constraint-based motion optimization using a statistical dynamic model. *ACM Transactions on Graphics (SIGGRAPH 2007)*.

[15] Xiaolin Wei and Jinxiang Chai. Intuitive interactive human character posing with millions of example poses. *IEEE Computer Graphics and Applications*, pages 78–88, 2009.

[16] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. *ACM Transactions on Graphics (SIGGRAPH 2004)*, pages 522–531.

[17] Meinard Müller, Tido Röder, and Michael Clausen. Efficient content-based retrieval of motion capture data. In *ACM Transactions on Graphics (TOG)*, pages 677–685, 2005.

[18] Björn Krüger, Jochen Tautges, Andreas Weber, and Arno Zinke. Fast local and global similarity searches in large motion capture databases. ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pages 1–10, 2010.

[19] InterSense. Is-900 system. http://www.intersense.com.

[20] David M. Mount and Sunil Arya. Library for approximate nearest neighbor searching. http://www.cs.umd.edu/ mount/ANN, 2010.

[21] Thomas Melzer, Michael Reiter, and Horst Bischof. Appearance models based on kernel canonical correlation analysis. *Pattern Recognition*, 36(9):1961 – 1971, 2003.

[22] Tim Davis. Umfpack version 5.6.1. http://www.cise.ufl.edu/research/sparse, 2012.

[23] Jehee Lee. Representing rotations and orientations in geometric computing. *IEEE Comput. Graph. Appl.*, pages 75–83, 2008.

[24] Yoonsang Lee, Sungeun Kim, and Jehee Lee. Data-driven biped control. *ACM Trans. Graph. (SIGGRAPH 2010)*, 29:129:1–129:8.

[25] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. Wiley-interscience, 2012.

[26] Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Wu. An optimal algorithm for approximate nearest neighbor searching. In *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 573–582, 1994.

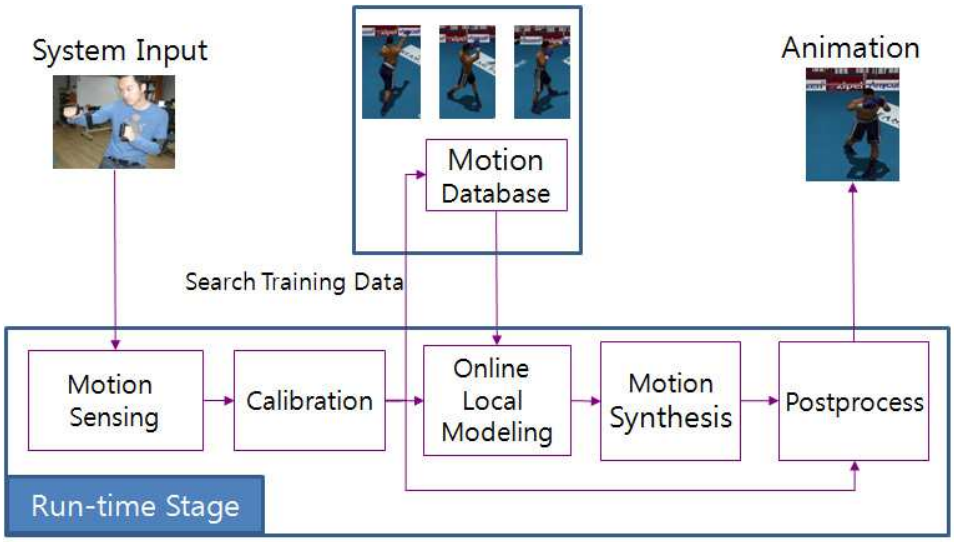

Figure 1: Overall workflow of our performance animation

	length	**#neighbors**	**query time**
(a)	136s	8	2ms
(b)	225s	8	3ms
(c)	389s	8	4ms
(d)	389s	16	7ms

Table 1: Length of motion database versus query time.

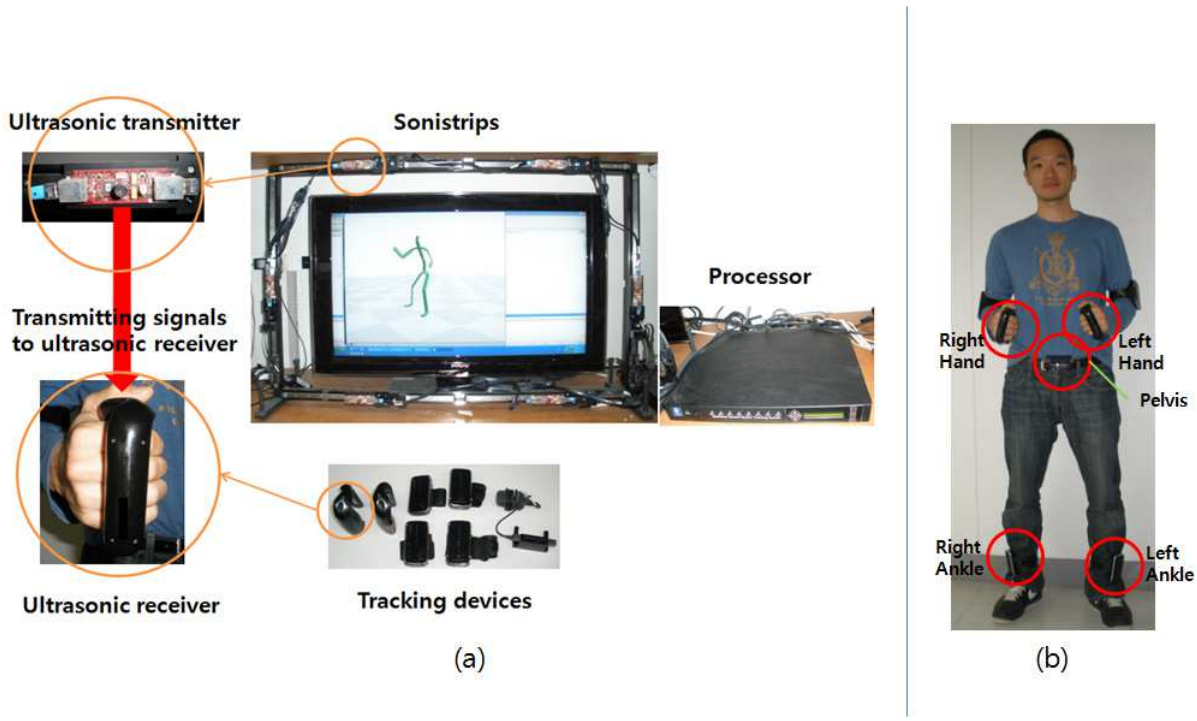

Figure 2: 3D motion sensor setups: (a) each transmitter transmits a signal to the receiver(3D motion sensor); (b) 3D motion sensors are attached to the hands, pelvis, and ankles of the performer.

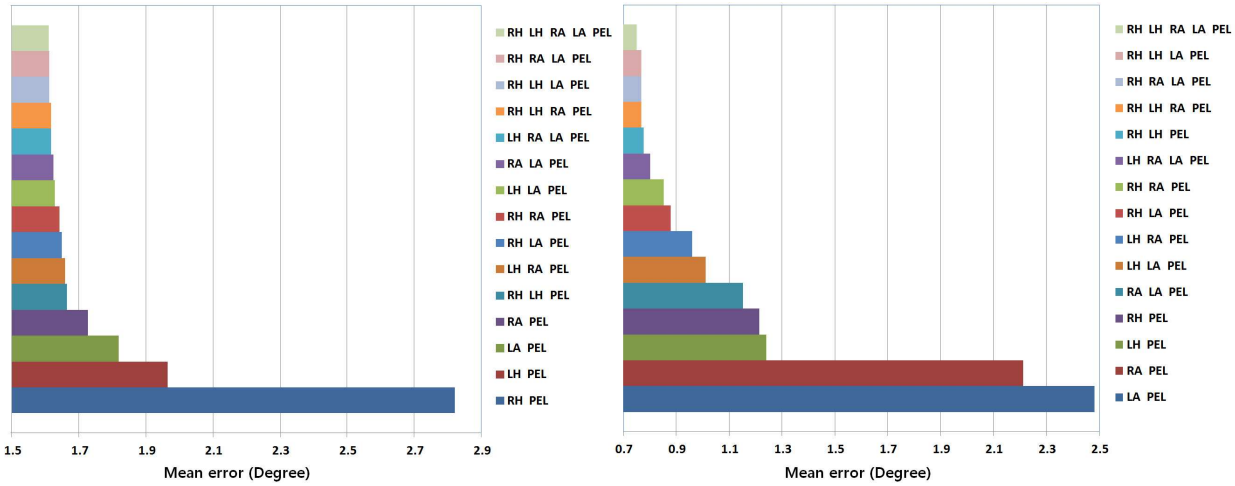

Figure 3: Comparison of reconstruction error with different sensor setups: (left) walking motion; (right) boxing motion; RH: right hand, LH: left hand, LA: left ankle, RA: right ankle, PEL: pelvis.

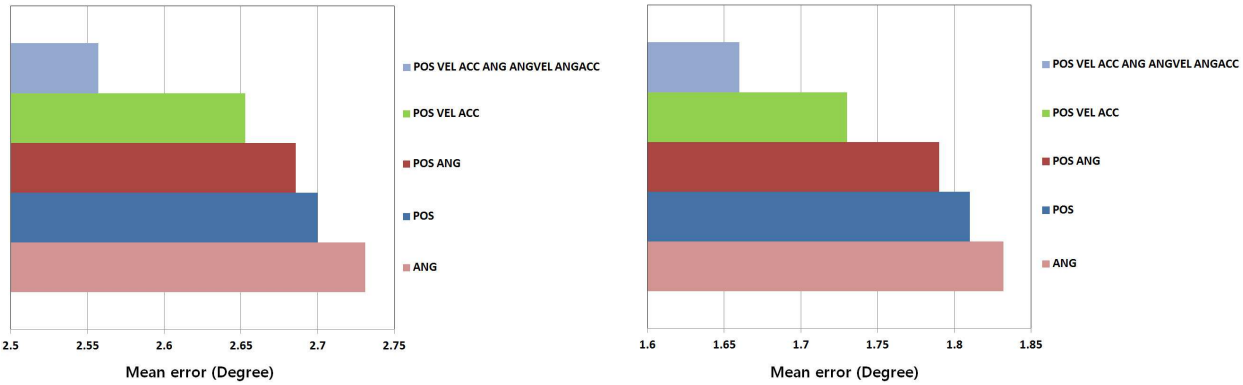

Figure 4: Comparison of reconstruction error with combinations of features for the query metrics: (left) table tennis motion; (right) walking motion; POS: position; VEL: velocity; ANG: orientation; ANGVEL: angular velocity; ACC: acceleration; ANGACC: angular acceleration.

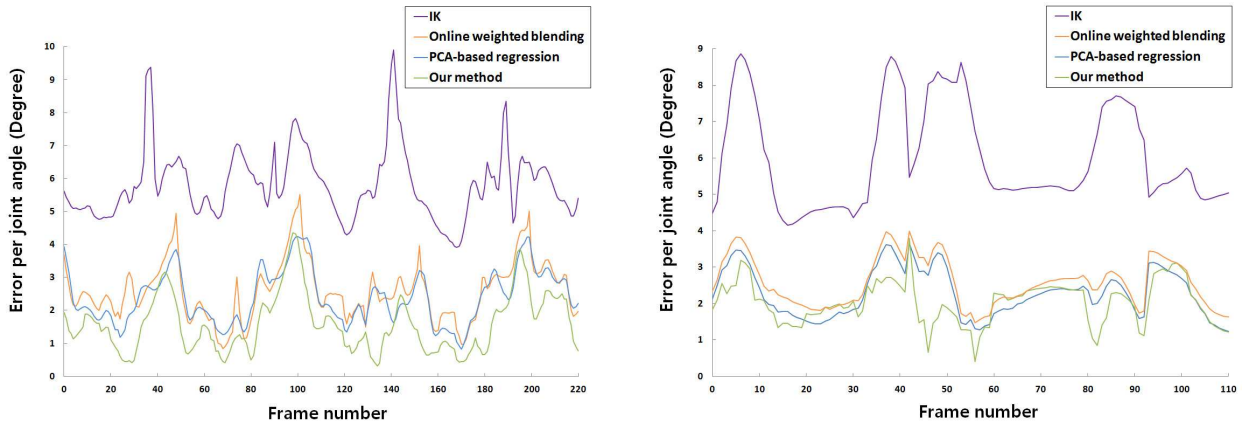

Figure 5: Comparison of the joint angle error in the reconstructed motion with three baseline methods: (left) table tennis motion; (right) boxing motion; our method makes the smallest amount of error among all methods.

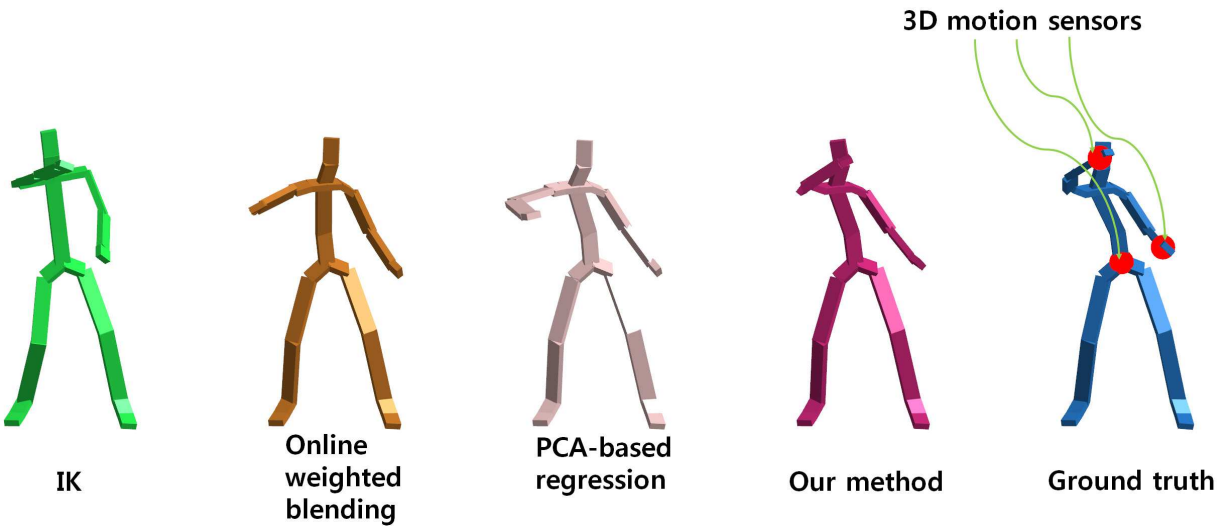

Figure 6: Our method can generate the desired pose more accurately than the other methods due to the extrapolation capability of kernel CCA-based regression (see the right arm and spine). Three red spheres on the rightmost character indicate the locations of the 3D motion sensors (right hand, left hand, and pelvis).

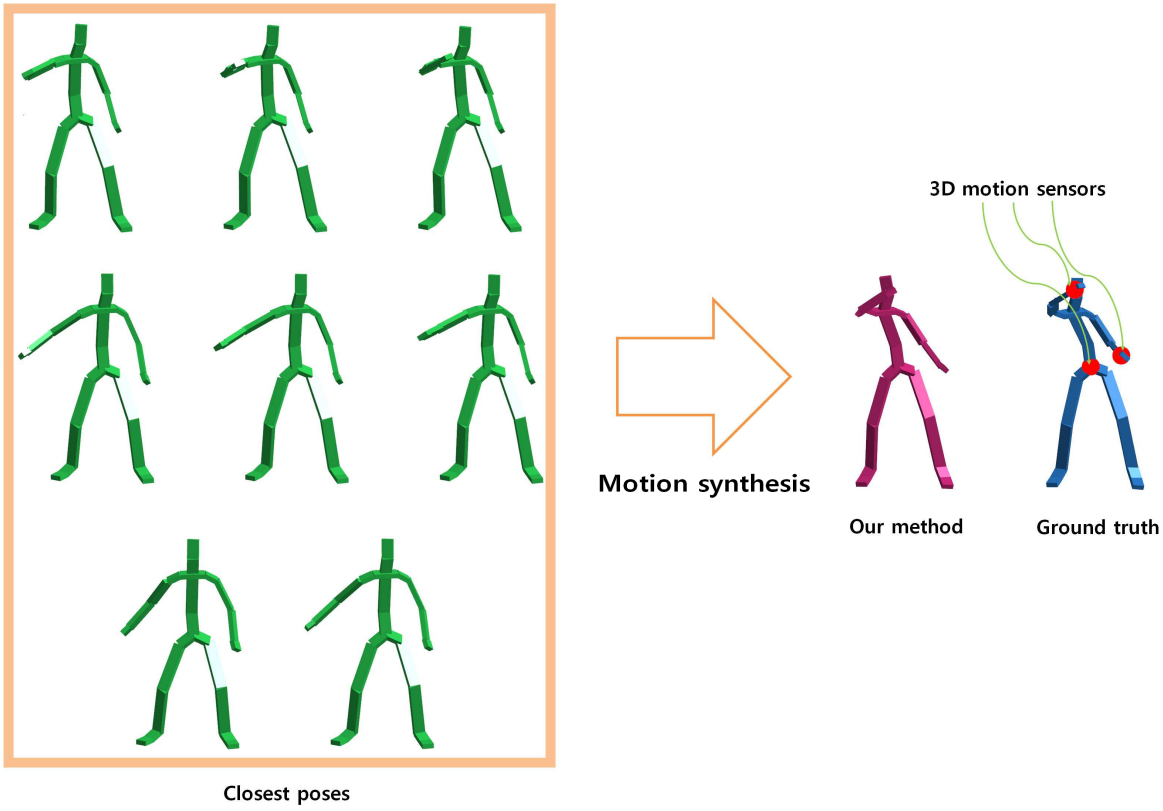

Figure 7: Visualizations of the eight nearest neighbors for the corresponding synthesized pose and ground truth. Given the extrapolation capability of our approach, we can generate a new pose that does not exist in the motion database.

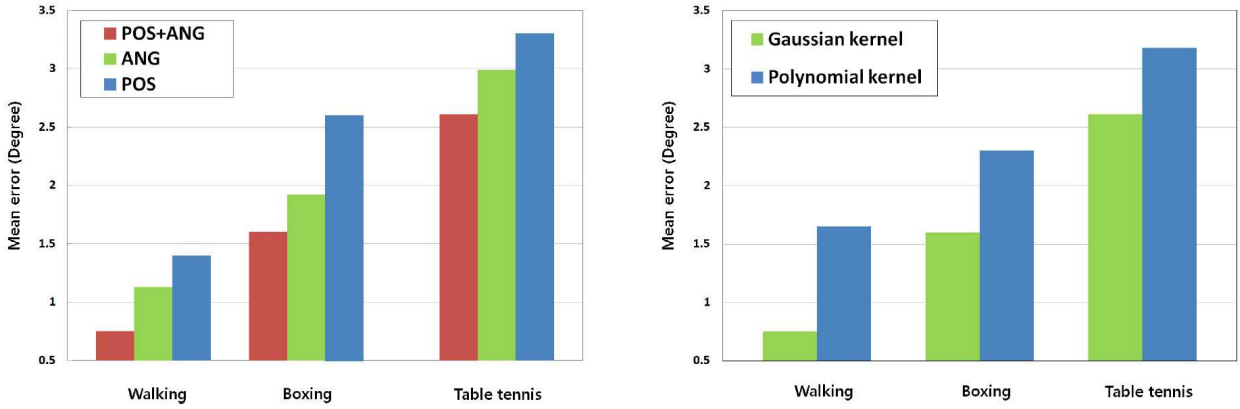

Figure 8: Comparison of reconstruction error for three different actions: (left) motion reconstruction error with different combinations of sensor signals; (right) motion reconstruction error with different types of kernel functions.

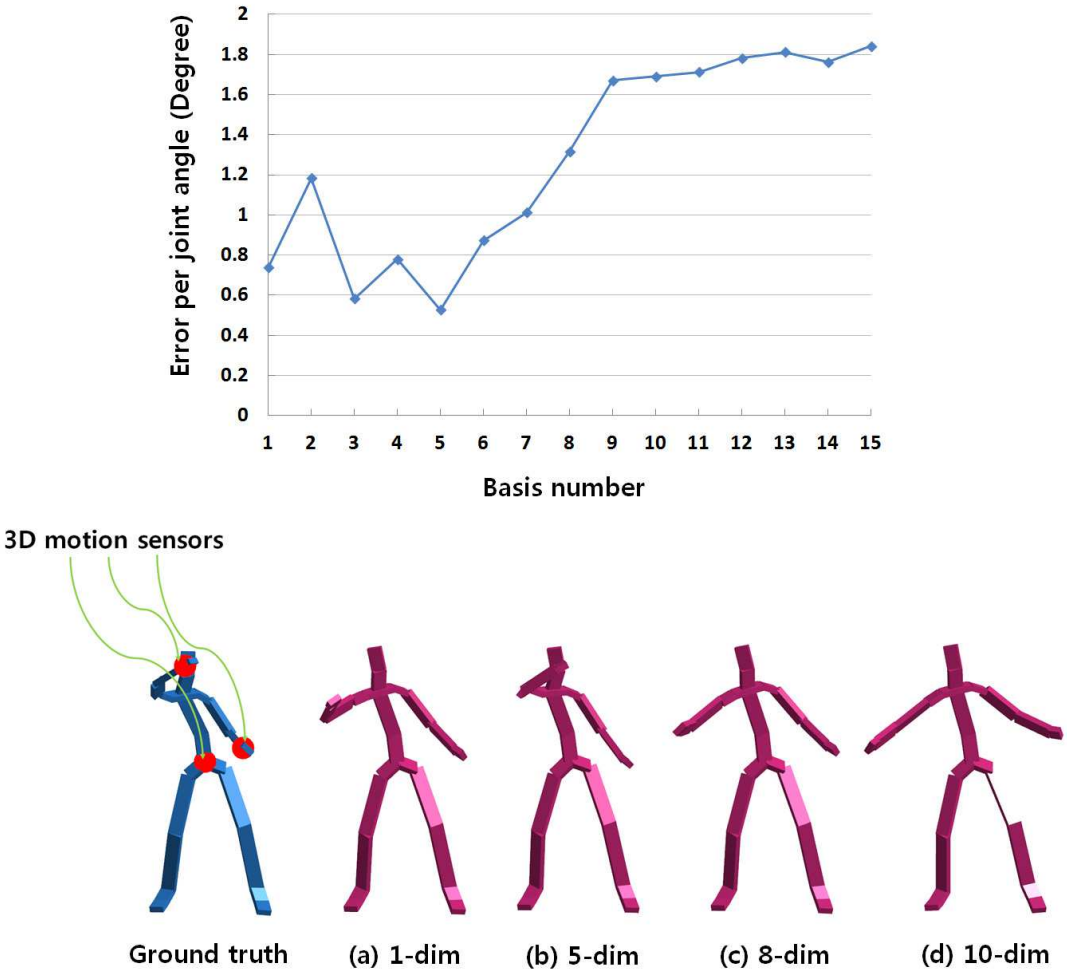

Figure 9: Motion reconstruction accuracy for different basis number. The pose prediction error does not decrease monotonically as the basis number increases. Our system showed an over-fitting pose only when more than five dimensions were used. The reconstructed poses with different number of bases are illustrated along with ground truth pose. In contrast to measuring reconstruction error, we do not apply inverse kinematics to each illustrated character's end-effectors. dim: dimension.

Figure 10: The boxing motion of the performer is transferred to the virtual character.

Figure 11: Realistic table tennis motion is reproduced by our system.