In Proceedings of Motion In Games (MIG), Rennes, France, 2012
This version is the author's manuscript.

Realtime Performance Animation Using Sparse 3D Motion Sensors

Jongmin Kim, Yeongho Seol, and Jehee Lee

Seoul National University, Seoul 151-742, Korea
{kimjongmin,seolyeongho,jehee}@mrl.snu.ac.kr
http://mrl.snu.ac.kr

Abstract. This paper presents a realtime performance animation system that reproduces full-body character animation based on sparse 3D motion sensors on the performer. Producing faithful character animation from this setting is a mathematically ill-posed problem because input data from the sensors is not sufficient to determine the full degrees of freedom of a character. Given the input data from 3D motion sensors, we pick similar poses from the motion database and build an online local model that transforms the low-dimensional input signal into a high-dimensional character pose. Kernel CCA (Canonical Correlation Analysis)-based regression is employed as the model, which effectively covers a wide range of motion. Examples show that various human motions are naturally reproduced by our method.

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

1 Introduction

This paper presents a realtime performance animation system to control full-body animation based on sparse 3D motion sensors attached on the performer's body. We select 3D motion sensors as an input device because they are free from the occlusion: problem, a typical problem 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 consider a data-driven approach using motion capture data to predict full-body motion.

Utilizing motion capture data as prior knowledge is widely used in performance animation systems to convert low-dimensional data into high-dimensional 3D animation. Previously, a data-driven online animation system was addressed in [4, 14]. They set up an online linear local model with every frame from a precaptured motion database to handle various motions. Although their methods generated convincing results, reconstructing accurate motion sequences is challenging when the performed motion does not exist in the database.

We employ a nonlinear statistical model, kernel CCA-based regression [6], which is a capable of strong extrapolation. 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 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 the convincing recreation of the performer's input motions, with examples being boxing, walking, and table tennis. We evaluate the performance of our system with different sensor setups to find the most effective capture setup. We also show that our performance animation system can generate more accurate character animation than previous methods.

2 Related Work

Reproducing the human motion of the character 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 has been used to record the 3D movement of a user who wore a skin-tight garment with dozens of attachable optical markers. Badler and colleagues [1] employed four magnetic sensors and an inverse kinematics method to control the standing pose of a full-body character. Shin and colleagues [18] mapped the actions of a performer to a virtual character using a magnetic motion capture system. Yin and Pai [22] proposed an animation system that controls avatars with a foot pressure sensor pad. Lee and colleagues [11] used a vision-based technique to capture full-body motion from a single camera. Chai and Hodgins [4] 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. 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 equipment. While this hardware can provide an exciting game experience in a constrained environment, it is not applicable for general-purpose retargeting. Our sensor-based performance animation system shares the affordable cost advantage of those input devices while remaining applicable to a wider range of situations.

Slyper and Hodgins [19] created a system that searches a motion database with low-cost accelerometers and then plays the best match of motion sequences from the database. Tautges and colleagues [20] introduced a performance animation system for generating full-body character animation using four accelerometers. Liu and colleagues [14] also introduced an online animation system with a small number of inertial sensors as a performance animation interface. Our approach has a similar environment setup to the work of [14], however, we suggest an alternative data-driven approach for performance animation. We utilize ker-

nel CCA-based regression [6], which performs a strong extrapolation by applying a kernel trick method to the CCA-based regression [8].

Many motion synthesis studies have utilized statistical analysis from motion data. Brand and Hertzmann [2] employed a Gaussian mixture model to estimate the distribution of given motion data, inferring the desired poses by computing several parameters. Grochow and colleagues [7] applied the GPLVM (Gaussian Process Latent Variable Model) from motion examples and a set of kinematic constraints to create realistic human motion. Shin and Lee [17] proposed an interactive motion synthesis framework in the low-dimensional space. Liu and colleagues [13] exploited probabilistic PCA (Principle Components Analysis) to predict full-body human motions from a set of principle markers. Chai and Hodgins [3] constructed a statistical dynamic model to generate motion sequences satisfying given user's constraints. Wei and Chai [21] introduced a system that interactively controls a 3D human character using 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 local model to handle the relationship between 3D motion sensors and the human pose [6].

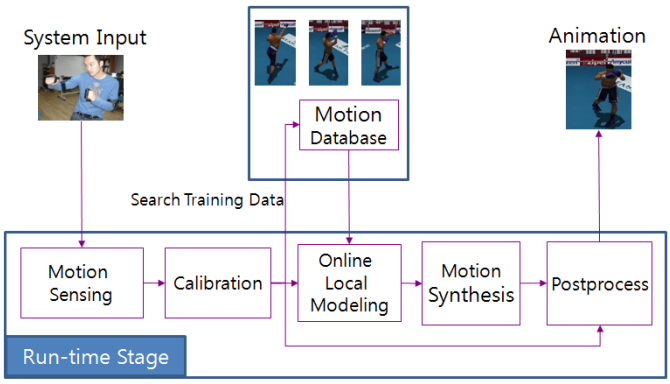

Fig. 1: Overall workflow of our performance animation.

3 Overview

Our system uses a motion database and five 3D motion sensors to reconstruct the full-body character pose. As the input data from the sensors do not provide enough information to restore an accurate full-body character, we additionally employ a 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 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 the kernel CCA-based regression. We then estimate the whole body character pose with the learned model. A motion post-processing step reduces noise in the retargeting result.

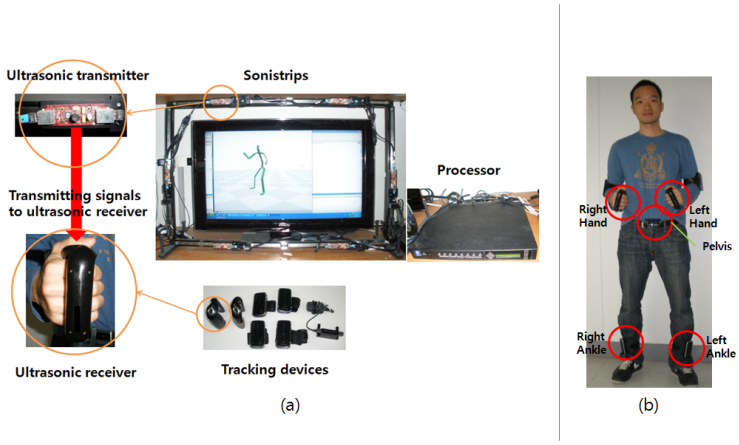

Fig. 2: 3D motion sensor setups: (a) each transmitter transmits the signal to the receiver(3D motion sensor); (b) 3D motion sensors are attached to the hands, pelvis, and ankles of the performer.

4 Sensor Data and Calibration

We use the InterSense IS-900 system [9], which offers robust realtime 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 rates values, respectively. Using the measured values, the processor computes 6-DOF tracking data, which is employed in the next stages. The performer wears sparse 3D motion sensors on the target joints, which here are the pelvis, right hand, left hand, right ankle, and left ankle (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 6-DOF tracking data from the 3D motion sensors needs to 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 the standard "T-pose" to calibrate the information from the 3D motion sensors. We transforms the position and orientations of the i-th 3D motion sensor in the world coordinates, $\mathbf{p}_s^i$ and $\mathbf{q}_s^i$, into corresponding values in the virtual coordinates, $\mathbf{p}_m^i$ and $\mathbf{q}_m^i$, as follows:

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

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 the the translational component of the i-th 3D motion sensor, respectively. $\mathbf{T}_q^i$ denotes the transformation quaternion for sensor orientation.

5 Motion Synthesis

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

5.1 Online Local Model

We suggest an online local modeling approach that can produce various types of motion sequences, of the 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 with 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, k poses which are similar to the current input from the motion sensors are chosen as the training data. We employ the position, velocity, and orientation of sensors for the subsequent query metric:

$$w_1 \sum_{i=1}^{n} \|\mathbf{T}_p^i \mathbf{p}_s^i + \mathbf{b}^i - \mathbf{p}_m^i\|^2 + w_2 \sum_{i=1}^{n} \|\mathbf{T}_p^i \mathbf{v}_s^i - \mathbf{v}_m^i\|^2 + w_3 \sum_{i=1}^{n} \|\mathbf{T}_q^i \mathbf{q}_s^i - \mathbf{q}_m^i\|^2 \tag{2}$$

where $\mathbf{v}_s^i$ and $\mathbf{v}_m^i$ denote the velocity of the i-th 3D motion sensor and the corresponding end-effector of the k-collected poses, respectively. We use $w_1 = 0.6$, $w_2 = 0.2$, and $w_3 = 0.2$ as the weight values for each respective term.

As the total number of motions in a 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 [16].

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 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 joint. 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 will be 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 is not sufficient to determine the full degrees of freedom of a character. We choose the 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 a strong extrapolation. Our system can faithfully generate reasonable animation results when a few similar motions exist.

CCA is a machine learning algorithms that can maximize the correlations between training inputs and outputs. It extracts the meaningful features of both training inputs and outputs to maximize the correlations between them. Once the correlation has been maximized with the transformed training data, it become possible to avoid over-fitting as associated with the regression process efficiently. We formulate this into a nonlinear problem while utilizing kernel method in CCA [15], as there is a very clear nonlinear relationship between the training input $\mathbf{x}$ and output $\mathbf{y}$. The following equation represents the transformation of the training input $\mathbf{x}$ 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 [15]. 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 [15]. 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 the problem using what is known as kernel trick method. kernel trick uses a kernel function with well-defined inner products of vectors in the feature space. kernel trick method can be applied to kernel CCA equation, as the equation contains the values of kernel for every training data [15]. 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 $\bar{\mathbf{x}} = \hat{\mathbf{X}}^T \tilde{\mathbf{x}}$, where $\tilde{\mathbf{x}} = \big(k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_m)\big)^T$. The training output data in the reduced dimension $\bar{\mathbf{y}}$ can be computed similarly.

We can obtain the matrix $\mathbf{A}$ that transforms the reduced input to the reduced output from the following equation:

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

We can also acquire the transformation matrix $\mathbf{B}$ that transforms the reduced output to the character pose $\mathbf{y}$ from 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 [5] for the process of these two regressions. The desired pose is estimated by multiplying $\mathbf{BA}\hat{\mathbf{X}}^T$ to 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 is prone to temporal jitter, a smoothing process needs to be considered if wanting to produce 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 is 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, $\mathbf{y}_{t-1}$, $\mathbf{y}_{t-2}$ are the reproduced poses in the previous two frames, and the mathematical notations from [10, 12] are used. We employed $\alpha = 0.3$ and $\beta = 0.4$ as the blending weights. Finally, analytical inverse kinematics is applied to the character to make its end-effectors accurately follow the position and orientation of motion sensors.

6 Experimental Results

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

Figure 6 and 7 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

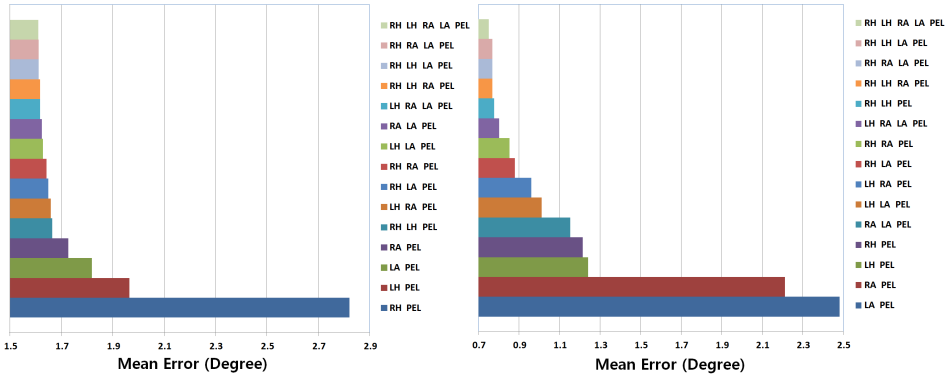

Fig. 3: Comparison of error reconstruction with different sensor setups: (left) walking motion; (right) boxing motion; RH: right hand, LH: left hand, LA: left ankle, RA: right ankle, PEL: pelvis

are generated by our method, such as jabs, straight, 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 the application of the inverse kinematics method 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 based on the variation of the sensor setups to select appropriate sensors. 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 combinations of sensor setups. For walking motions, the placement of the three sensors (RA, LA, PEL) gave a suitable result compared to when more than three sensors were used. For the boxing motion data, with three sensors, their placement (RH, LH, PEL) led to a well-generated result. In this experiment, we found that selecting the highest number of most movable joints as the system input for the each movement is important to minimize the number of sensors and reconstruct the desired motion sequences. As expected, the final motion can be generated more naturally as we add more sensors on the end-effectors onto the limbs. In this experiment, because walking motions are relatively insufficient to determine various types of walking, it is more challenging to generate the accurate results than boxing motion.

We compared the error of our algorithm with two baseline methods: inverse kinematics and an online weighted blending method. The inverse kinematics

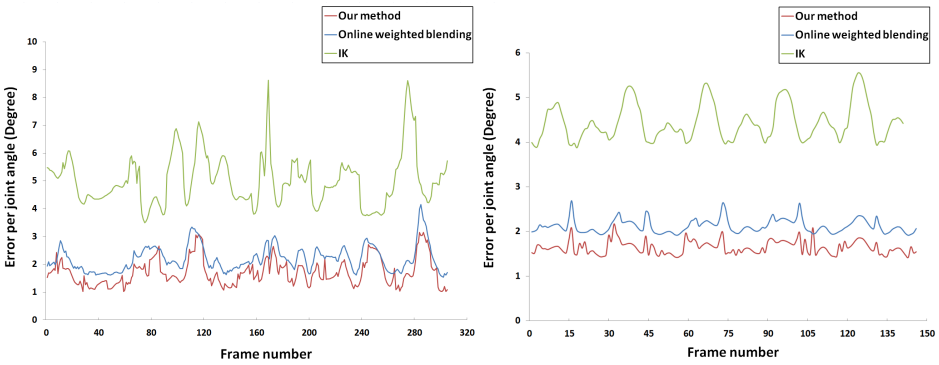

Fig. 4: Comparison with two baseline methods for reconstructing motion: (left) table tennis motion; (right) walking motion; our methods makes the smallest error from among all methods.

method simply matches all measured end-effector configurations to reproduce character poses. The online weighted blending method generates new poses on the fly via the weighted blending of nearby k-neighbor poses (Section 5.1). To test the extrapolation capacity of our method, we setup the motion database with extremely sparse motion examples (74 frames), as follows: table tennis, 31 frames of motion; walking, 43 frames of motion. The input motion data does not exist in the database. The graphs show that our method generates more accurate results than the two baseline methods. Figure 4 shows a frame-by-frame performance comparison of the given motion sequences.

The extrapolation capacity of kernel CCA-based regression method is evaluated by comparing our results with those from the online weighted blending method when the input motion sequences does not exist in the sparse dataset (table tennis, 31 frames of motion) as shown in Figure 5. Character poses are reconstructed without the application of the inverse kinematics method to the end-effectors. In the case of insufficient motion capture data, our method is able to generate reliable results due to its a strong extrapolation ability. In our experiments, the ground truth data is acquired from the motion capture data.

7 Discussion

This paper presents a robust realtime performance animation system based on kernel CCA-based regression framework. After an online local model is established, the 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 nearby poses of the current and previous frames' input motions and effective temporal interpolation produces temporally smooth results.

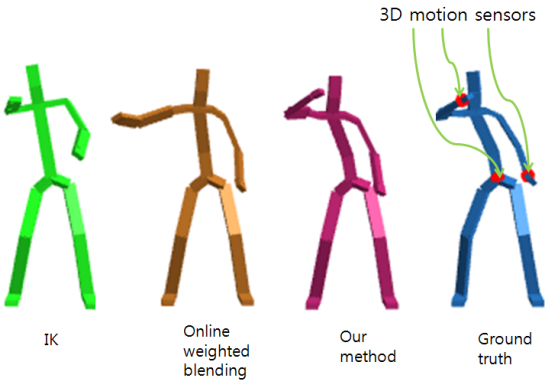

Fig. 5: Our kernel CCA-based regression method can generate desired poses more accurately than the online weighted blending method due to the strong extrapolation capability of kernel CCA-based regression (see e.g., the poses of the right arm and spine). The three red spheres on the rightmost character indicate the locations of the 3D motion sensors (right hand, left hand, and pelvis).

Temporal filtering works in realtime with a small delay. We demonstrated the robustness of our system by showing performance animation examples of boxing and table tennis.

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 markers using a temporal smoothing function in the motion post-processing step, markers that are not tracked for a long time cannot be approximated. In this case, we temporally stop the system and wait until reliable input data is provided.

While we concentrated on the 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 need to solve additional problems for this goal, such as accurate synchronization and the appropriate selection of a model, and 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).

Fig. 6: The boxing motion of the performer is transferred to the virtual character.

Fig. 7: Realistic table tennis motion is reproduced by our system.

References

1. Badler, N.I., Hollick, M.J., Granieri, J.P.: Real-time control of a virtual human using minimal sensors. Teleoperators and Virtual Environments 2, 82–86 (1993)
2. Brand, M., Hertzmann, A.: Style machines. ACM Transactions on Graphics (SIGGRAPH 2000) pp. 183–192
3. Chai, J., Hodgins, J.K.: Constraint-based motion optimization using a statistical dynamic model. ACM Transactions on Graphics (SIGGRAPH 2007)
4. Chai, J., Hodgins, J.K.: Performance animation from low-dimensional control signals. ACM Transactions on Graphics (SIGGRAPH 2005) pp. 686–696
5. Davis, T.: Umfpack version 5.6.1. http://www.cise.ufl.edu/research/sparse (2012)
6. Feng, W.W., Kim, B.U., Yu, Y.: Real-time data driven deformation using kernel canonical correlation analysis. ACM Transactions on Graphics (SIGGRAPH 2008) 27(3), 1–9
7. Grochow, K., Martin, S.L., Hertzmann, A., Popović, Z.: Style-based inverse kinematics. ACM Transactions on Graphics (SIGGRAPH 2004) pp. 522–531
8. Huang, H., Yin, K., Zhao, L., Qi, Y., Yu, Y., Tong, X.: Detail-preserving controllable deformation from sparse examples. IEEE Transactions on Visualization and Computer Graphics pp. 1215–1227 (2012)
9. InterSense: Is-900 system. http://www.intersense.com
10. Lee, J.: Representing rotations and orientations in geometric computing. IEEE Comput. Graph. Appl. pp. 75–83 (2008)
11. Lee, J., Chai, J., Reitsma, P.S.A., Hodgins, J.K., Pollard, N.S.: Interactive control of avatars animated with human motion data. ACM Transactions on Graphics (SIGGRAPH 2002) pp. 491–500
12. Lee, Y., Kim, S., Lee, J.: Data-driven biped control. ACM Trans. Graph. (SIGGRAPH 2010) 29, 129:1–129:8
13. Liu, G., Zhang, J., Wang, W., McMillan, L.: Human motion estimation from a reduced marker set. pp. 35–42. ACM Symposium on Interactive 3D graphics and games (2006)
14. Liu, H., Wei, X., Chai, J., Ha, I., Rhee, T.: Realtime human motion control with a small number of inertial sensors. In: Symposium on Interactive 3D Graphics and Games. pp. 133–140 (2011)
15. Melzer, T., Reiter, M., Bischof, H.: Appearance models based on kernel canonical correlation analysis. Pattern Recognition 36(9), 1961 – 1971 (2003)
16. Mount, D.M., Arya, S.: Library for approximate nearest neighbor searching. http://www.cs.umd.edu/ mount/ANN (2010)
17. Shin, H.J., Lee, J.: Motion synthesis and editing in low-dimensional spaces: Research articles. Comput. Animat. Virtual Worlds pp. 67 – 94 (2006)
18. Shin, H.J., Lee, J., Shin, S.Y., Gleicher, M.: Computer puppetry: An importance-based approach. ACM Transactions on Graphics pp. 67–94 (2001)
19. Slyper, R., Hodgins, J.: Action capture with accelerometers. pp. 193–199. ACM SIGGRAPH / Eurographics Symposium on Computer Animation (2008)
20. Tautges, J., Zinke, A., Krüger, B., Baumann, J., Weber, A., Helten, T., Müller, M., Seidel, H.P., Eberhardt, B.: Motion reconstruction using sparse accelerometer data. ACM Transactions on Graphics pp. 18:1–18:12 (2010)
21. Wei, X., Chai, J.: Intuitive interactive human character posing with millions of example poses. IEEE Computer Graphics and Applications pp. 78–88 (2009)
22. Yin, K., Pai, D.K.: Footsee: an interactive animation system. pp. 329–338. ACM SIGGRAPH / Eurographics Symposium on Computer Animation (2003)