Data-driven Motion Estimation with Low-Cost Sensors

Liguang Xie[1], Mithilesh Kumar[1], Yong Cao[1],Denis Gracanin[1], Francis Quek[1]

[1] Computer Science Department
Virginia Polytechnic Institute and State University, United States
yongcao@vt.edu

Keywords: Performance animation, accelerometer, motion synthesis, data-driven, local modeling

Abstract

Motion capture can produce high quality data for motion generation. However, that professional motion capture is expensive, and imposes restrictions on the capturing environment. We propose a motion estimation framework that utilizes a small set of low-cost, 3D acceleration sensors. We use a data-driven approach to synthesize realistic human motion comparable in quality to the motion captured by the professional motion capture systems. We employ eight 3D accelerometers — four Nintendo© Wii controllers — attached to a performer's body to capture motion data. The collected data is used to synthesize high quality motion from a statistical model learned from a high quality motion capture database. The proposed system is inexpensive and is easy to setup.

1 Introduction

Compared to manual animation authoring solutions, automatic motion generation techniques can produce a large amount of high quality motion clips with little effort. Motion capture is one of these techniques and has been widely adopted by the animation industry. However, motion capture systems are costly and require a long setup time. This places significant constraints for its use. Recently, the availability of low-cost motion sensors, such as 3D accelerometers and gyroscopes, promises to put ubiquitous motion generation systems within the reach of the animation community.

In this paper, we propose a low cost motion estimation and synthesis framework. A prototype system is built on a small number of Nintendo© Wii™controllers that are easy to attach to human body. Using Wii controllers as input devices, we are able to generate high quality motion data using our motion estimation framework. The system we developed is easy to set up and imposes little or no restriction on the data acquisition environment. We aim to make our motion synthesis framework as convenient as video capture systems and make it applicable to a wide range of applications, ranging from video game interfaces, animated chat-rooms to interactive character control in virtual environments (VEs).

We estimate full body motion in two phases. During the first, data collection phase, we collect motion data from a "profesional" performer using a commercially available professional motion capture system. At the same time we also capture 3D acceleration data from eight sensors (four Wii controllers) attached to the performer's body. This one-to-one mapped, time synchronized data is used to create a large, high quality motion capture database. In the second phase, we capture motion from a 'regular user' using only the attached accelerometer sensors. We then estimate the corresponding motion using a local linear model created from the motion capture database. The proposed local linear model can estimate high quality motion from low-dimensional noisy accelerometer data. Our local modeling approach also enables us to scale the database to incorporate large amounts of motions without performance degradation.

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 acceleration sensors.

The remainder of the paper is organized as follows. Section 2 provides the background and describes the related work in this area. Section 3 explains the system architecture while Sections 4 and 5 provide the detailed description of our approach. Section 6 shows the results and demonstrates the accuracy of our approach. Section 7 summarizes the paper and discusses the limitations. Section 8 discusses future work to address the current limitations.

2 Background

There is a variety of user interfaces to control the motion of characters in 3D VEs, especially for gaming. The input devices include mouses, keyboards, joysticks and other devices such as vision based tracking systems (e.g., ©Sony EyeToy) and inertial/acceleration sensors (e.g., Wii™Controller). Such user interfaces provide immediate and direct control signals with limited number of degrees of freedom. Therefore, it is difficult to provide performance-driven control for complex human motions.

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 [9] 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.

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. [6] 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. Oore et al. [7] use six degree-of-freedom tracking devices to interactively control the locomotive animations. Dontcheva et al. [5] also use a tangible user interface in their puppetry system to control the motions divided into several different layers.

Recently, Vlasic et al. [8] 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 system is still high and, due to the necessary post-processing time, it is not a real-time system.

3 System Overview

We describe a novel approach that uses low cost 3D acceleration sensors to synthesize full body motion. The high quality motion is synthesized from the input sensor data using a statistic model learned from a motion capture database. There are two major phases of our approach – data collection and motion synthesis.

Data collection: 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 pre-processed 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.

Motion synthesis: Figure 1 describes this phase. During the motion synthesis phase, the user performs actions using only 3D acceleration sensors (Wii controllers) attached to the body. Using this sensor data, we synthesize high quality motion data and the statistical model built from the motion database captured in the previous step. In order to build the model, we first segment the database into clusters using a Gaussian Mixture clustering algorithm. We then construct a Radial Basis Function (RBF) interpolation model for each cluster. 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. All newly generated poses are post-processed to generate a smooth animation sequence.

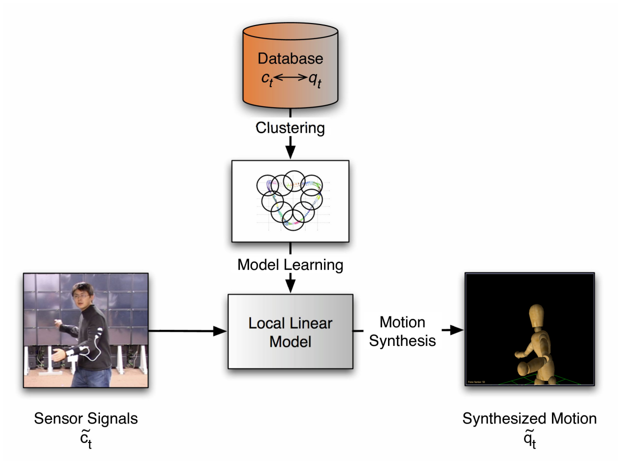

Figure 1: *Motion query and synthesis.*

4 Data Collection and Preprocessing

In this Section we describe our data capture, synchronization and pre-processing procedures.

4.1 Data Capture and Representation

We perform an off-line motion capture session to create a motion database which consists of two types of synchronized data. One is high quality motion capture data, acquired using a Vicon optical motion capture system. The other type is accelerometer sensor data from Wii controllers. The database contains 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).

We use a system with 8 Vicon MX series cameras for high quality motion capture at a frame rate of 60 frames per second. Simultaneously, data are captured using low cost 3D accelerometers (Wii controllers) with a range of $\pm 3g$ and built-in *Bluetooth*® interface for data transmission at a peak rate of 100 frames per second, where a data frame consists of acceleration values from all the sensors. The interface based on these wireless sensors is cheap, easy to set-up and, unlike a vision based system, does not suffer from occlusion. The data obtained from the motion performance recording is pre-processed to control the frame rate and provide noise reduction due to the wireless environment.

Figure 2 shows a total of 45 retro-reflective markers and eight accelerometers attached to the performer's body. The sensors are attached to the arms and legs since they provide most of the movements for a majority of human actions. Each sensor transmits its 3D acceleration signal to the data collection computer where the data is converted into sensor frames, synchronized with motion data frames and stored into the database. There are N frames of data in the database, which

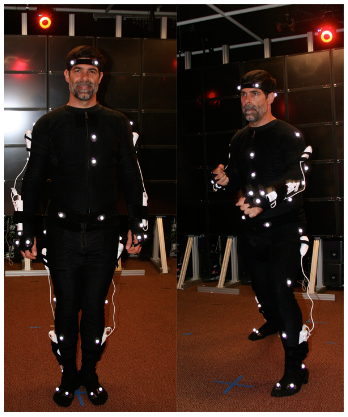

Figure 2: *Data collection: an optical motion capture system and a 3D acceleration sensor based data acquisition system are used in parallel. There are 45 retro-reflective markers and eight sensors (four Wii^{TM}Nintendo controllers) attached to the performer.*

can be represented as

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

where $\mathbf{c}_t$ is a 24-dimensional frame of sensor data representing the 3D acceleration measures of four or eight sensors on the body at time t. $\mathbf{q}_t$ is a frame of optical motion capture data and it represents a pose at time t. The pose is represented in the form of local joint rotations in the quaternion format. Note that there is one-to-one correspondance between $\mathbf{c}_t$ and $\mathbf{q}_t$.

4.2 Data Synchronization

In order to map the motion data $\mathbf{q}_t$ to the sensor data $\mathbf{c}_t$, it is necessary to synchronize data from all the sensors and optical motion capture system. This is critical since the sensors transmit independently of each other using the wireless medium. Moreover, data from each sensor is received with variable frame-rate owing to packet loss in the wireless environment. To resolve 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 after every frame period and one sensor frame is constructed with the currently 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 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.

4.3 Data Pre-Processing

Before storing the captured data into the database, we pre-process the data 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 value. Noise in optical motion capture data due to marker occlusion is removed in the post-processing step of the data capture stage. This noise reduction is crucial for the output data quality.

Noise in the sensor data is mainly because of the wireless environment. We use sensitive Bluetooth receivers to maintain a high bandwidth. It is common to find a few arbitrary values that are beyond the range of values we expect from the sensors. These values are automatically detected and replaced by quantities that are estimated from the neighboring data.

5 Model Learning and Motion Synthesis

In this section, we describe how to build the model from the database and how to synthesize novel motions from the model.

5.1 Local Linear Model

Local linear models are often used to deal with a large database that cannot be accurately represented by a linear model. The idea is to cluster the whole database into smaller, compact datasets via a Gaussian mixture model. We then build a linear model for each dataset. Globally, the model is non-linear but the local models are linear and have simple linear solutions.

5.1.1 Gaussian Mixture Clustering

In our approach we model the motion database using a Gaussian mixture model. The model is defined as follows:

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

where K is the number of clusters and $\theta = \{\pi_j, \mu_j, \Sigma_j\}$ are the model parameters. π_j is the mixing weight satisfying $0 \leq \pi_j \leq 1$ and $\Sigma_{j=1}^{K} \pi_j = 1$. μ_j and Σ_j are the mean and covariance of the mixture cluster j. We use Bouman's algorithm [2] to estimate the number of clusters which best fit the motion capture dataset Q containing optical motion capture data $\mathbf{q}_t$. The estimation is based on the Rissenen order identification criteria known as minimum description length [2]. Given the number of clusters K, we perform unsupervised classification on Q and get the model parameter sets θ. Following the clustering result of Q, we classify the sensor dataset $C = \{\mathbf{c}_t\}$ into K clusters by one-to-one mapping from $\mathbf{c}_t$ to $\mathbf{q}_t$ during data synchronization. Figure 3 shows an example plot of motion capture dataset clustering and sensor dataset clustering for tennis forehand action. In our database, the total 5,082 frames are classified into 81 clusters.

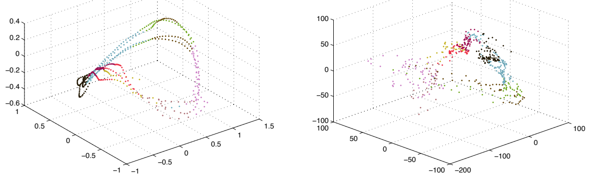
Figure 3: *Example of Motion and Sensor Dataset Clustering: A set of data points with the same color represent one cluster. The dataset are plotted in the top three components of PCA space of motion capture data (left figure) and sensor data (right figure), respectively.*

5.1.2 Principal Component Analysis

For each cluster Q_j, where $1 \leq j \leq K$ and $Q_j = \{\mathbf{q}_i^j | i = 1, 2, .., n_j\}$ (n_j is the size of j^{th}cluster), we first apply Principal Component Analysis (PCA) to reduce the dimensionality of motion capture data Q_j. In our case we reduce the dimensionality from 72 to 7. This choice of lower dimension is an optimum value that reduces computational cost of model learning and synthesis while still capturing 99% of the motion variance. The reduced dimension data $\mathbf{r}_i^j$ is produced by PCA using Equation 2:

$$\mathbf{r}_i^j = (\mathbf{q}_i^j - \bar{\mathbf{q}}_j)A_j^{-1}, \tag{2}$$

where $\bar{\mathbf{q}}_j$ is the mean value for cluster j, A_j is the transfer matrix built from the eigenvectors corresponding to the largest eigenvalues of the covariance matrix of the data.

5.1.3 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) [3] 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 3 as:

$$\tilde{\mathbf{q}}_t = F_j(\tilde{\mathbf{c}}_t) = \bar{\mathbf{q}}_j + A_j \sum_{i=1}^{p_j} \mathbf{w}_{ij} \phi(||\tilde{\mathbf{c}}_t - \mathbf{c}_i^j||), \tag{3}$$

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, multiquadratic, or thin plate spline. We chose the Gaussian function,

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

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 [3] 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 3).

5.2 Motion Synthesis with Interpolation Model

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 3 to synthesize new motion data frame $\tilde{\mathbf{q}}_t$.

During the post-processing step, we smoothen the synthesized motion data $\tilde{\mathbf{q}}_t, 1 \leq t \leq N$ using a low pass filter.

6 Results

We tested the performance of the system with two subjects performing various actions. Figure 4 shows the results for four synthesized actions, tennis forehand, tennis backhand, basketball shot and middle block from Karate. The results clearly show that the synthesized motion precisely captured the poses of the subjects (compared to the video footage of the motion capture session).

We perform an end-to-end evaluation to measure the accuracy of our system. During capture sessions we also recorded the high quality motions using an optical motion capture system. The recorded high quality motions are used as ground truth that can be compared against the synthesized motion frame by frame. The recorded motions and the synthesize motions are both converted from quaternion data to joint angle data for error calculation. We then use the normalized Root Mean Square (RMS) distance e to quantitatively measure the difference. The unit of e is degree of freedom per angle. e is defined as below:

$$e = RMS(\tilde{\mathbf{q}}_k, \mathbf{q}_k) = \sqrt{\frac{\sum_{i=1}^{n}(\tilde{\mathbf{q}}_{k,i} - \mathbf{q}_{k,i})^2}{n}}, \tag{4}$$

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 $\mathbf{q}_k$. Table 1 shows the RMS distances for four synthesized motions. Figure 5 shows a comparison of one of the synthesized motions with the corresponding ground-truth motion. (Please refer to the demonstration video for more comparisons). 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. In terms of computational expense, our system is efficient. The motion synthesis implemented in *Matlab*® is at a rate of about 0.019 seconds/frame, compared with real time at a rate of 0.016 seconds/frame (for 60 frames per second animation).

Actions	**Frame Number**	**Average RMS**	**Processing Time**
Basketball shot	302	0.37	5.78 sec.
Tennis Forehand	256	0.18	4.90 sec.
Tennis Backhand	206	0.26	3.94 sec.
Middle Block	160	0.53	3.07 sec.

Table 1: *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.*

7 Conclusion

We present a framework for estimating full body motion using a small set of low cost inertial sensors. Our two step approach involves data collection and motion synthesis. Data collection is performed in the studio and produces a database of time synchronized high quality motion capture data and sensor data. Prior to motion synthesis, we cluster the database and create local linear models that enable us to convert the non-linear global database into clusters of linear data. We then apply linear interpolation or optimization technique for pose estimation. We have shown the effectiveness of our framework by performing an End-to-End evaluation.

The type of animation that can be synthesized using our system is only limited by the size of the database and sensitivity of the sensors. The examples close to the motion to be synthesized must be present in the database. However, our design is scalable and can handle larger databases without performance degradation. Using a sufficient number of examples we can synthesize a large variety of human motions.

8 Future Work

In our current work we have focused on a motion estimation system that requires data to be processed offline and used later. We plan to improve the performance of our motion capture framework and support interactive applications by using real-time optimization algorithms. Real-time support will allow performers to view the results during capture session and hence make quick adaptations.

Like most of the linear model based motion synthesis, our approach suffers from the smoothness problem during motion synthesis. This problem results partly from the noise in the sensor data and partly from possible discontinuity between two clusters. We plan to improve the smoothness of synthesized motion by using on-line motion blending techniques.

A major goal of our research is to use minimum number of sensors without compromising the quality. We believe that it is possible to generate visually indistinguishable animation with a reduced number of sensors, if the motion to be acquired is within a small set of movements (e.g. golf swing). For such a scenario, we would like to design an effective database and synthesize full body motion with as little as 1 sensor.

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] C. A. Bouman. Cluster: An unsupervised algorithm for modeling gaussian mixtures. Available from http://www.ece.purdue.edu/~bouman, April 1997.

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

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

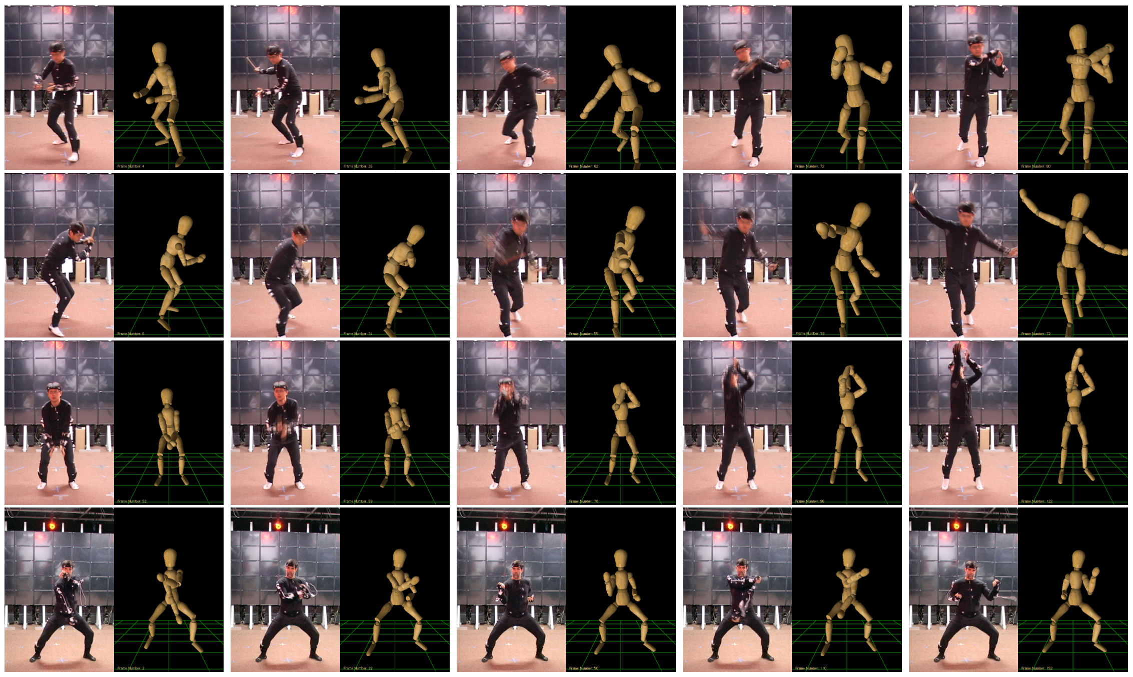

Figure 4: *Four different actions (one in each row) synthesized by our system. Each frame shows on the left side the actual pose and on the right side the synthesized pose.*

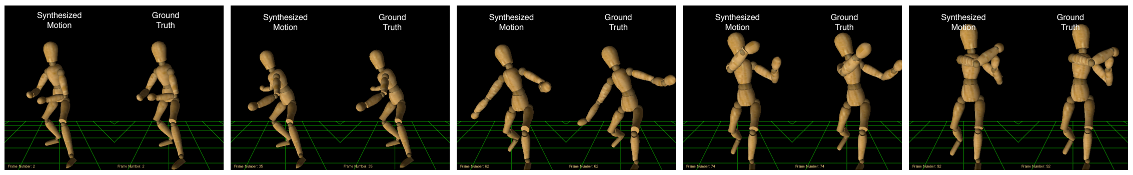

Figure 5: *Synthesized motion compared to the ground truth. Each frame shows on the left side the synthesized motion and on the right side the ground truth.*