3D Res (2017) 8:25
DOI 10.1007/s13319-017-0136-9

3DR EXPRESS

Learning Motion Features for Example-Based Finger Motion Estimation for Virtual Characters

Christos Mousas · Christos-Nikolaos Anagnostopoulos

Received: 8 May 2017 / Revised: 6 July 2017 / Accepted: 11 July 2017

Abstract This paper presents a methodology for estimating the motion of a character's fingers based on the use of motion features provided by a virtual character's hand. In the presented methodology, firstly, the motion data is segmented into discrete phases. Then, a number of motion features are computed for each motion segment of a character's hand. The motion features are pre-processed using restricted Boltzmann machines, and by using the different variations of semantically similar finger gestures in a support vector machine learning mechanism, the optimal weights for each feature assigned to a metric are computed. The advantages of the presented methodology in comparison to previous solutions are the following: First, we automate the computation of optimal weights that are assigned to each motion feature counted in our metric. Second, the presented methodology achieves an increase (about 17%) in correctly estimated finger gestures in comparison to a previous method.

Keywords Finger motion · Motion estimation · Character animation · Motion features · Features preprocessing · Metric learning

1 Introduction

In character animation research, a variety of methodologies have been proposed to synthesize the motion of a virtual character as naturally as possible. However, the realism of the character's motion is not only related to its general representation (i.e., the full-body motion of a character), but it also depends on the details that appear on the character's body. For instance, facial expressions and finger motion enhance the appearance of a motion sequence. The aforementioned notations have been expressed in various perceptual evaluation studies, such as [1–4].

If highly realistic motion data is required, the motion of the fingers should be captured simultaneously with the full-body motion of a performer. With motion capture systems, it is too difficult to capture the full-body motion of an actor while also capturing facial expressions and finger gesture details. This is especially true when the actor that is captured performs locomotion tasks (i.e., moving within the capturing area space). For that reason, three basic

Electronic supplementary material The online version of this article (doi:10.1007/s13319-017-0136-9) contains supplementary material, which is available to authorized users.

C. Mousas (✉)
Graphics and Entertainment Technology Lab, Department of Computer Science, Southern Illinois University, Carbondale, IL 62901, USA
e-mail: christos@cs.siu.edu

C.-N. Anagnostopoulos
Department of Cultural Technology and Communication, University of the Aegean, 81100 Mytilene, Greece
e-mail: canag@ct.aegean.gr

Published online: 18 July 2017

methodologies could be used to add such details onto the character's body. First, by using key-frame techniques, it is possible to animate the motion of the fingers. Second, a reduced number of markers that are placed on the performer's hand could be captured and then, by using motion reconstruction techniques [5, 6], the motion of the fingers could be reconstructed [4, 7–9]. Finally, by capturing collections of finger gestures along with the motion of a performer's hand (with the actor remaining static), it is possible to estimate the finger gestures using parameters provided by the character's hand, as shown in [10, 11]. The advantage of the last technique is mainly its ability to estimate and consequently synthesize the required details automatically, without the need to animate such details using key-frame techniques. Hence, considering that ten or more characters may evolve in a film or videogame, example-based techniques could speed up the synthesis of such details.

Considering the aforementioned difficulties with capturing the motion of a character's fingers and the time-consuming process of animating finger gestures, this paper presents a methodology for estimating finger motion by using as parameters a variety of motion features associated with sample finger gestures contained in a database. In the presented method, a number of motion features computed by a character's hand are considered. The motion features are pre-processed using restricted Boltzmann machines (RBMs). RBM pre-processing performs a transformation of the feature space based on an unsupervised learning step. The aforementioned unsupervised training does not optimize the similarity between features, since the similarity ground truth is not included. However, RBMs help to transform the feature space in such a way that machine learning is benefited. Thus, by transforming the feature space in a suitable representation, a better adaptation of the model is achieved when the similarity ground truth data is entered.

In the presented methodology, the weighted Euclidean distance is used to model the inverse similarity of two segments. The optimization of the function is achieved by using a Support Vector Machine (SVM) to adapt weighted distances based on sample variations of semantically similar finger gestures. To compute the required optimal weights, the finger gesture dataset provided in [12] was used. The "large dataset" consists of eight different gesture types with eight different variations for each gesture. Generally, this dataset is ideal for cross-validations. This dataset has an additional important role: since there are variations of similar gestures, it is possible to use these repetitions to determine the optimal weights automatically.

To illustrate the efficiency of the presented methodology, upon having computed the optimal weights by using the large dataset, the same weights were applied to estimate the motion of a character's finger by using different datasets provided in [12]. These datasets contain freeform motions, instructing directions, and conversation-related and debate-related finger gestures. Finally, it should be noted that all the aforementioned datasets provide the full-body motion of a character that includes finger motion.

In summary, the advantages of the proposed solution are the following. It:

- proposes the use of a number of character's hand motion features that can be computed;
- shows that RBM pre-processing enhances the estimation rate;
- automates the weight optimization of a metric (compared to [10]) by using the SVM.

The remainder of this paper is organized as follows: Sect. 2 outlines related work on finger animation techniques. Sect. 3 covers motion data pre-processing. The finger motion estimation methodology is presented in Sect. 4. Implementation details are given in Sect. 5. The evaluations of the proposed methodology are presented in Sect. 6. Finally, conclusions are drawn and potential future work is discussed is Sect. 7.

2 Related Work

In previous years, a number of different approaches have been proposed to synthesize finger motion sequences by simplifying the rules of the motion synthesis process. A comprehensive article describing the majority of finger animation techniques that have been developed over the years can be found in [13, 14].

Among previously published work, methods that automatically synthesize hand motions, known as "hand-over animation" have already been proposed. The hand-over animation process relies on the ability to add finger and hand motion capture data to the pre-existing full-body motion capture data. Hence, solutions such as those proposed by Kang et al. [7],

Wheatland et al. [8], and Mousas et al. [9] are responsible for adding the required motions of the character's hand even when a reduced number of markers are used for the motion estimation process.

Other methodologies have introduced the automatic addition of valid character finger motion. In Jin and Hahn [15], the pose space distance from the character's motion capture data is analyzed and a stepwise searching algorithm is used to find the key poses to synthesize the hand motion. Jörg et al. [10] proposed another interesting solution for finger motion synthesis. This approach is based on the ability to synthesize a character's finger motions by assigning weight variables to the wrist's position and orientation, which are used as the input parameters to search the segment that best matches the character's gesture. Even though this solution provides the ability to synthesize perceptually valid finger motions, the basic disadvantage lies in the capability to estimate all the required motions correctly. This is due to the fact that their system uses only spatial information provided by the character's wrist. The aforementioned methodology was extended by Bitan et al. [16] to handle additional constraints, such as interaction between objects or other virtual characters as well as by Mousas and Anagnostopoulos [17] that implemented a real-time prediction algorithm using hidden Markov models. Another data-driven solution for synthesizing a character's finger motion was proposed by Majkowska et al. [18]. In this methodology, finger motion and body motion are captured separately in a pre-processing stage. Then, during the composition process, these motion sequences are assembled using spatial and temporal alignment methodologies, and the motion correlation is found. In general, this technique takes advantage of motion transplantation techniques [19–21]. These methods have been previously used to combine different body parts' motions with a new motion sequence in an attempt to maintain a natural final generated motion.

There are a number of methods for synthesizing a character's hand motion that are constrained to specific tasks. This is due to the difficulty of predicting a valid human finger motion. Even when using a whole-body motion, it is difficult to predict the exact actions of the hand, which are highly dependent on the fingers. Hence, other solutions, such as the one proposed by Ye and Liu [22], are responsible for generating the motion of the finger based on wrist movements and the handled object's specified motion constraints. Specific manipulation strategies are assigned to the fingers, while the character's wrists are used to predict the evolution of a specified action. Similar methods for synthesizing finger-object interactions by using physics-related and kinematics-related solutions have also been presented in [23] and [24]. Moreover, data-driven solutions for hand-object interactions have also been proposed, such as the methodology presented in [25].

Furthermore, finger motion synthesis techniques when virtual characters interact with musical instruments have also been proposed. One of the most recent approaches proposed by Zhu et al. [26] assigns a specific action to the fingers in conjunction with the ability to execute information from a MIDI input and from predefined parameters of a piano performance. Their system generates a valid motion sequence for the virtual character, where the fingers play an active role in the motion synthesis process. Similarly, the solution proposed by ElKoura and Singh [27] generates finger motion for specific tasks, such as playing musical instruments.

Alternatively, physics-based approaches have been used to generate finger motions, particularly for manipulation tasks, such as the solutions proposed by Liu [28], Pollard and Zordan [29], Andrews and Kry [30], and Kyota and Saito [31]. Other works use sensors to measure forces [32] in an attempt to generate the correct finger motion. In addition, Neff and Seidel [33] derived synthesized human hand motions by using relaxed hand shapes and physics-based parameters retrieved from video recordings. Another interesting solution for animating detailed and anatomically valid hand and finger motion was proposed by Tsang et al. [34]. The main disadvantage of all previously mentioned solutions is the computation cost that is required to generate the desired motion of the fingers. Finally, Zhao et al. [35] proposed a physics-based motion control system for real-time synthesis of human grasping. Once given an object to be grasped, their system automatically computes physics-based motion control that advances the simulation in order to achieve realistic manipulation of the object.

Lastly, as shown in Jörg et al. [10], information provided by the wrist (position and orientation) could be useful to the data-driven finger motion prediction process. In the presented solution, a weighted distance metric is enhanced by a variety of motion features of a

character's hand. It is shown that by pre-processing the motion features, it is possible to enhance the estimation rate of the metric. Finally, compared to Jörg et al. [10], the optimization of the distance function is performed automatically by SVM. These are the basic advantages of the proposed approach comparing to the solution proposed by Jörg et al. [10], which is the closest to the presented one.

3 Pre-Processing Motion Data

In the following three sub-sections, we present the motion segmentation method that is used, the collection of motion features that are computed for each motion segment, and finally the labeling process of motion segments.

3.1 Motion Segmentation

A gesture is divided into different phases, according to McNeil [36] and Kendon [37], which are: the preparation, in which the arm moves away from its rest position, the pre-stoke hold, the stroke, the post-stoke hold, and the retraction, where the arm moves back into the resting position. As mentioned in [10], it is not possible to identify all these phases. Hence, they adopted a simple motion segmentation process, which uses the speed of the wrist as a parameter in order to split the motion data into two phases: the motion phase that contains high-speed sequences, and the hold phase that contains low-speed motions.

In this paper, we consider segmenting the motion data into additional phases (compared to [10] and [38]), which will be able to describe the distance and speed phases of the character's wrist simultaneously. The proposed segmentation process splits the motion data based on the maximum and minimum speed as well as on the maximum and minimum distance, which results in five phases. Specifically, it is calculated that each segment is characterized by its speed phases (increase, decrease, static) as well as by its wrist position phases (increase, decrease, static).

Given a motion segment s_i that is represented by the distance $(d_1, \ldots, d_N)$ and the speed $(u_1, \ldots, u_N)$ of the wrist, where N denotes the total number of frames of the segment s_i, our segmentation method retrieves segments that are categorized into the following five phases:

- P_1: the distance and the speed of the wrist increase simultaneously ($d_1 < d_N$, $u_1 < u_N$),
- P_2: the distance of the wrist increases and the speed of the wrist decreases ($d_1 < d_N$, $u_1 > u_N$),
- P_3: the distance and the speed of the wrist remain static ($d_1 \approx d_N$, $u_1 \approx u_N$),
- P_4: the distance of the wrist decreases and the speed of the wrist increases ($d_1 > d_N$, $u_1 < u_N$), and
- P_5: the distance and the speed of the wrist decrease simultaneously ($d_1 > d_N$, $u_1 > u_N$).

A simple graph that illustrates the segmentation process used in the presented methodology is shown in Fig. 1. It should be noted that these five phases describe the ideal segmentation of a single gesture. It should further be noted that when dealing with freeform gestures (such as those included in the direction or in the conversation dataset provided in [12]), some of the phases might not appear very often.

3.2 Collecting Motion Features

Human motion can be characterized by its features. These features are responsible for describing the motion in either a spatial or a temporal manner. Generally, a variety of motion features can be computed through different body parts to analyze human motion and to estimate motion similarities. A variety of either spatial or temporal or spatiotemporal motion features have been used for the activity recognition process, as they have been examined in a variety of motion similarity solutions [39–41]. Additionally, even though a number of motion features for

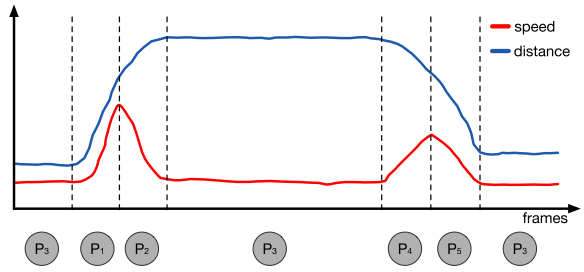

Fig. 1 The segmentation process used in the presented methodology. As we can see, it is possible to distinguish the five phases of the finger gestures. The *red line* indicates the speed of the wrist and the *blue line* indicates the distance of the wrist. (Color figure online)

gesture recognition have been proposed in the past, such as [42] and [43], less attention has been given to collecting motion features that could make it feasible to associate the motion of the hand with finger gestures.

According to [10], the parameters that are provided by the local position and orientation of the character's wrist can be quite beneficial for the finger motion estimation process. As it is also shown in the aforementioned work, information about a gesture could also be given by using the position and rotation of both the shoulder and the elbow. However, a gesture is not only related to the aforementioned two features. Thus, we tried to collect a variety of motion features provided by a character's hand that could be associated with the motion of the fingers. The collected motion features, which were used in the presented methodology, are illustrated in Table 1. It should be noted that all these features are computed with respect to the character's root position, orientation, velocity, and acceleration.

3.3 Labeling Segments

Once the motion data is segmented, each of the motion segments is labeled. The labeling process is quite important since it automates the model training process. With the labeling process, the system automatically determines semantically similar segments in order to pre-process their features. Moreover, according to the labeling process, the system is able to search segments that belong to specific phases. A segment s_i of a gesture is labeled according to its gesture type, t (i.e., attention, big, ds, ok, pp, small, turn, wipe), as well as according to its phase, p (i.e., any of the five phases presented in Sect. 3). Thus, a single segment of a gesture is represented as $s^{t,p}$.

4 Similarity Learning

This section presents the methodology used to train our system the motion feature similarities, such as estimating the most valid finger motion. First, we briefly introduce the RBM that is used to pre-process the computed motion features. Next, the similarity model and the model training procedure are explained. Finally, an optimization of the metric using SVM is presented.

Table 1 The collected motion features that were used in the presented methodology in order to estimate the motion of a character's fingers

Features	f_i
Spatial distance of wrist's trajectory	f_1
Spatial distance of elbow's trajectory	f_2
Spatial distance of shoulder's trajectory	f_3
Orientation distance of wrist trajectory	f_4
Orientation distance of elbow trajectory	f_5
Orientation distance of shoulder trajectory	f_6
Average rotation of wrist	f_7
Average rotation of elbow	f_8
Average rotation of shoulder	f_9
Wrist mean velocity	f_{10}
Wrist velocity distance	f_{11}
Wrist mean acceleration	f_{12}
Wrist acceleration distance	f_{13}
Wrist mean angular velocity	f_{14}
Wrist angular velocity distance	f_{15}
Wrist mean angular acceleration	f_{16}
Wrist angular acceleration distance	f_{17}
Wrist spatial extent	f_{18}
Wrist orientation extent	f_{19}
Mean wrist trajectory curvature	f_{20}
Wrist trajectory curvature distance	f_{21}
Wrist mean height	f_{22}
Wrist height distance	f_{23}
Square of above features	$f_{24} - f_{46}$
Exponential of above features	$f_{47} - f_{69}$

4.1 RBM Feature Space Transformation

As mentioned, RBMs were used to pre-process the motion features, such as allowing the system to learn the similarity of motion segments efficiently. An RBM network [44] is a two-layer system of connections between visible and hidden layers. The RBM is represented by the weights $W = w_{i,j}$ with size $n \times m$ that are associated with the connection between hidden h_j and visible v_i units, as well as bias weights a_i and b_j for the visible and the hidden units, respectively. Given these, the energy of a configuration $E(v, h)$ is defined as:

$$E(v,h) = -\sum_{ij} v_i w_{ij} h_j - \sum_i v_i w_i - \sum_{ij} b_i h_j \qquad (1)$$

to represent the joint distribution:

$$P(v,h) = \frac{1}{Z} e^{-E(v,h)} \tag{2}$$

where Z denotes the partition function and is represented by:

$$Z = \sum_{v,h} e^{-E(v,h)} \tag{3}$$

where in the above equations v and h denote the states of the visible and hidden layer, respectively. A simple example that illustrates the RBM network is shown in Fig. 2.

In an RBM network, the units that correspond to each layer are characterized by their independence from one another. A state of a hidden unit is only dependent on the state of a visible unit and vice versa. The visible and hidden units are represented by the following probabilities:

$$P(h_j = 1|v) = \sigma\left(\sum_i v_i w_{ij} + b_j\right) \tag{4}$$

$$P(v_j = 1|h) = \sigma\left(\sum_j h_i w_{ij} + a_j\right) \tag{5}$$

where

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

denotes the logistic sigmoid function.

In an unsupervised RBM training process, it is required to minimize its log-likelihood $\hat{\ell}$ when a set of independent and identically distributed (i.i.d.) samples $V = \{v^1, \ldots, v^n\}$ are given, such as:

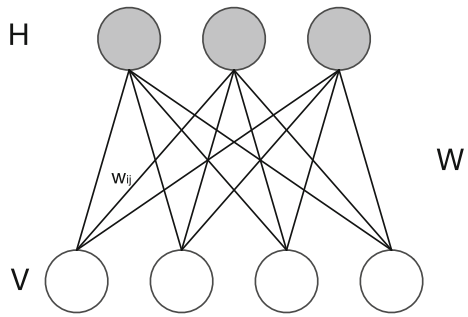

Fig. 2 The restricted Boltzmann machine network

$$\hat{\ell} = \frac{1}{N} \ln(L(\theta|V)) = \frac{1}{N} \sum_k \ln P(v^k|\theta) \tag{7}$$

with $\theta = \{W, a, b\}$.

The minimization of log-likelihood can be achieved by using gradient ascent optimization techniques. By computing the partition function Z, it is possible to estimate the exact gradient from the log-likelihood. The gradient ascent requires an expectation of data, which is sampled from the model as:

$$w_{ij} = w_{ij} + \eta(\langle v_i p(h_j|v)\rangle_0 - \langle v_i h_j\rangle_\infty) \tag{8}$$

where $\langle\cdot\rangle_0$ denotes the average of the data's distribution, and $\langle\cdot\rangle_\infty$ denotes the average with respect to the distribution from the model, and η denotes the learning rate.

Generally, Markov Chain Monte Carlo methods [45] could solve this problem by sampling the state of the model. This methodology is relatively slow: as it is unstable, this model needs to follow an unsupervised pre-sampling process before generating valid samples. This difficulty can be overcome by using the Contrastive Divergence algorithm [46]. This method shows that even with a small number of pre-sampling steps (e.g., with only one step), the learning step minimizes the divergence between the data's and the model's distributions. Hence, according to Eq. 8, the minimization is performed as follows:

$$w_{ij} = w_{ij} + \eta(\langle v_i h_j\rangle_0 - \langle v_i h_j\rangle_n) \tag{9}$$

Based on the aforementioned process, when a visible layer is set to its initial feature values, the hidden layer stands for a non-linear transformed feature space. Taking advantage of this, the finger gestures are estimated based on a similarity learning method presented in the following sub-sections. Finally, it should be noted that as a hyper-parameter, the hidden layer's units define the dimensionality of the new feature space.

4.2 Similarity Model

In the presented methodology, the computed similarity model is based on a weighted Euclidean metric similar to the one proposed in [10]. The presented metric uses a variety of features as well as automatic optimization. Thus, in contrast to [10], the proposed method automates the similarity learning process. The

weighted distance of two motion segments is used as the inverse indicator for those segments. As mentioned, each motion segment is labeled according to its type and phase. This allows the system to search specific gesture types as well as specific gesture phases. In our method, we define the distance between two feature vectors $x, y \in R^N$ as:

$$d(x,y) = \sqrt{\sum a_i d_i} \text{ where } d_i = (x_i - y_i)^2 \quad (10)$$

Specifically, d_i computes the distances between the features. At each distance $(x_i - y_i)^2$, a weight a_i is assigned. It should be noted that $d(x, y)$ only qualifies as a metric if and only $a_i > 0 \forall i$. The aforementioned metric is quite suitable for computing the similarity between features. However, it lacks properties such as non-negativity and identity of points with distance equal to zero. It should be noted that Stober and Nrnberger [47] showed that by using a specific function to compute the distances d, which is possible only if the interpretation of a x_i in the feature vector is clearly defined, the results and the stability of the model could be improved.

4.3 Model Training

To train the model based on similarity from the input data, we used the gesture dataset (eight gesture types $\times$ eight repetitions). As mentioned, during the preprocessing of the motion data, each segment is labeled by its phase as well as by its gesture type. The labeling process is quite important, since it allows the system to automatically estimate the weights a_i as presented in Eq. 10. In order to fulfill the similarity constraints of the labeling process, the weights a_i are predicted from the gesture dataset.

Eight variations are responsible for describing a single gesture type. For simplicity's sake, in the remainder of the paper, we denote a reference motion segment that belongs to a specific gesture type and specific phase as r, any other motion segment that belongs to the same gesture type and phase as f, and the remainder of motion segments that belong to the same phase provided by different gesture types as g.

Given a constraint $h(r, f, g)$, the system determines whether segment r should be more similar to segment f than to any other segment g by:

$$h(r,f,g) = \begin{cases} true & \text{if phase and type of } r \text{ is similar} \\ & \text{to phase and type of } f \\ false & \text{otherwise} \end{cases}$$

The aforementioned constraint provides the ability to compute the optimal weights for each feature of a gesture.

Given the distance computed with Eq. 10, it is possible to presume constraints in the similarity model. The aforementioned Boolean function is indirectly optimized by the weighted sum of differences between distances over the training samples. This is achieved by:

$$h(r,f,g) = \sum_i a_i (d_i(r,g) - d_i(r,f)) > 0 \quad (11)$$

We assume that the training data is represented as $D = \{d_k | k = 1, \ldots, M\}$ and $d_k = \{r_k, f_k, g_k | h(r_k, f_k, g_k) = true\}$, where $\{r_k, f_k, g_k\} \in R^N$ represent the feature vectors. Given a set of D, a weight vector $a \in R^N$ must be found that maximizes the following function:

$$f(a) = \frac{1}{M} \sum_k^M \sum_i a_i (d_i(r,g) - d_i(r,f)) \quad (12)$$

Generally, the function $f(a)$ in Eq. 12 is linear. This means that its optimal may be found at large values of a without applying constraints. In the presented method, we used the SVM method to compute the optimal weights as presented in the following subsection.

4.4 Metric Learning with SVM

To compute the optimal weights, we used the SVM for distance metric learning as introduced by Schultz and Joachims [48]. In the presented methodology, to learn a weighted distance measure with SVMs, the classifier is optimized to produce a vector of weights that fulfills the distance constraints. Here, for each constraint $h(r, f, g)$, a feature distance difference vector $\delta^{(r,f,g)} \in R^N$ is constructed where:

$$\begin{aligned} \delta_i^{(r,f,g)} &= d_i(r,g) - d_i(r,f) \\ &= (r_i - g_i)^2 - (r_i - f_i)^2 \end{aligned} \quad (13)$$

The optimization process is performed as follows:

$$\min_{a,\xi} G(a) = \frac{1}{2} a^T a + c \sum_{(r,f,g)} \xi_{(r,f,g)} \quad (14)$$

subject to:

$$\forall (r,f,g) \quad a^T \delta^{(r,f,g)} > 1 - \xi_{(r,f,g)}$$
$$\xi_{(r,f,g)} \geq 0$$
$$a_i \geq 0$$

where c determines a trade-off between regularization and the enforcement of constraints. The slack variables $\xi_{(s_i^{p,t}, s_j^{p,t}, s_D^{p,t})}$ allow for some constraints to be violated while adding a penalty value to the optimization result.

5 Implementation

The presented methodology was implemented in the MATLAB 2014b release by using the MATRBM Library provided in [49] and the SVM-Light [50]. By using the gesture dataset, the system requires only a few seconds to compute the optimal weights for each segment phase. Given a new motion sequence that does not include the motion of the fingers (with regard to the finger gesture estimation process), the system first segments the motion data as presented in Sect. 3.1. Then, for each motion segment, the motion features that describe each segment are computed and assigned to a feature vector. Given the phase of the input motion segment that is retrieved according to the phase segmentation constraints and by using the computed optimal weights for a given phase, the system thoroughly searches all segments contained in our dataset according to Eq. 10 and retrieves the motion segment that is characterized by the minimum $d(x, y)$ score. This motion is then applied to the character's fingers. Finally, since the synthesis of a continuous motion sequence is required, the estimated segments are synthesized using the standard motion graph [51] methodology.

6 Evaluations and Results

This section presents the evaluations that were conducted, along with the associated results. For the evaluation process, the "large dataset" that contains eight gesture types, each with eight repetitions, was used.

6.1 RBM Feature Pre-processing

In this sub-section, we evaluate how the estimation rate when pre-processing the motion features is affected by the use of RBM-based transformation. This evaluation process is divided into a variety of steps. In the first step, the parameters of the RBM are explored. In the second step, combinations between parameters are evaluated using a grid search of the selected values. Then, the estimation rate of each tested model is used to set its parameters for the final evaluation. The tested and chosen values that were selected are shown in Table 2. It should be noted that using training accuracy for the model selection process is very challenging. For that reason, strict regularization for training a model should be carefully applied.

Figure 3 shows the performances of the learning algorithms with respect to the number of hidden units in the RBM pre-processing. For this evaluation process, the configurations of the RBM were assigned to the chosen values presented in Table 2, except for the number of hidden units where its values are tested. As we can see in Fig. 3, the method achieves its highest estimation rate when 750 hidden units are used. The estimation rate at 50 units is very low. It reaches a rate equal to 72%. Similarly for the 100 units, it reaches a rate equal to 75.5%. Using 750 hidden values, the performance of the method reaches 97%.

6.2 Ground-Truth Estimation Rate

In order to understand the efficiency of the presented methodology, the leave-one-out cross-validation method was used. For the validation process, we computed the correct estimation rate by either using the RBM pre-processing step or not. It should be noted that the gesture dataset was used for the validation process, since it contains repetitions of similar gesture types. By keeping a motion segment out of the database, the system was trained with the rest of the motion segments. Then, by using the excluded motion segment, the gesture type was estimated. The class

Table 2 Tested and chosen values that were used for the RBM grid search

The chosen values were used in the final experiments

Parameters	Hiden values	Chosen values
Number of hidden units (*hidNum*)	50, 100, 250, 500, 750, 1000, 1500, 2000	750
Learning rate (*lrate1*)	0.02, 0.05, 0.1, 0.25, 0.5	0.05
Learning rate (*lrate2*)	0.02, 0.05, 0.1, 0.25, 0.5	0.1
Momentum	0.05, 0.1, 0.25	0.1
Cost	0.00002, 0.00005, 0.0001, 0.001 0.01	0.00002

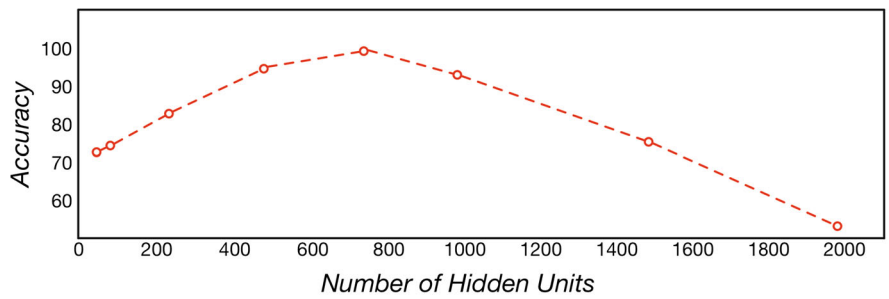

Fig. 3 Performance of the methodology with different values of RBM hidden units

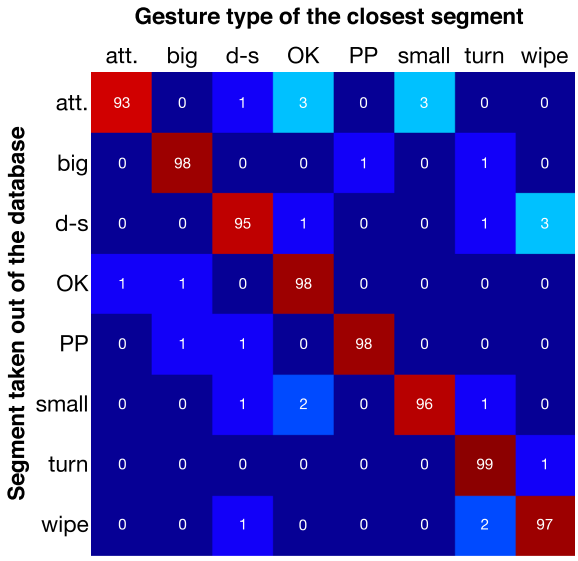

Fig. 4 The class confusion matrix shows the estimation rate resulting from the presented methodology. The estimation rate without the RBM feature pre-processing step is shown within parentheses

confusion matrix (Fig. 4) represents the results that were obtained from the evaluation process. On average, by using the eight different gesture types, the presented methodology achieves a correct estimation rate of 97% when motion features are pre-processed using the RBM and 88% when the RBM was not used to pre-process the motion features. Examples of correctly estimated gestures are shown in Fig. 5.

According to the results obtained from this evaluation process, the following should be mentioned. The presented methodology is able to estimate the finger gesture at a high rate of accuracy when using the pre-processing step. The estimation rate is increased by 9% when pre-processing the motion features. When comparing the estimation rate with the one resulting from the methodology proposed by Jörg et al. [10], it should be noted that by pre-processing the motion features with RBM, the presented approach provides better results. There is a 17% increase when the motion features are pre-processed using RBM. Generally, in [10], the optimization is performed manually and the metric uses only two (position and orientation of the wrist) features. We assume that this is the basic disadvantage of such a solution.

6.3 Additional Examples

The presented methodology is able to compute the optimal weights by analyzing the features of motion segments that belong to a similar gesture type and phase. By using the computed optimal weights provided by the gesture dataset, the presented methodology must be able to provide reasonable results when different datasets are used. For this reason, the computed optimal weights were applied to different

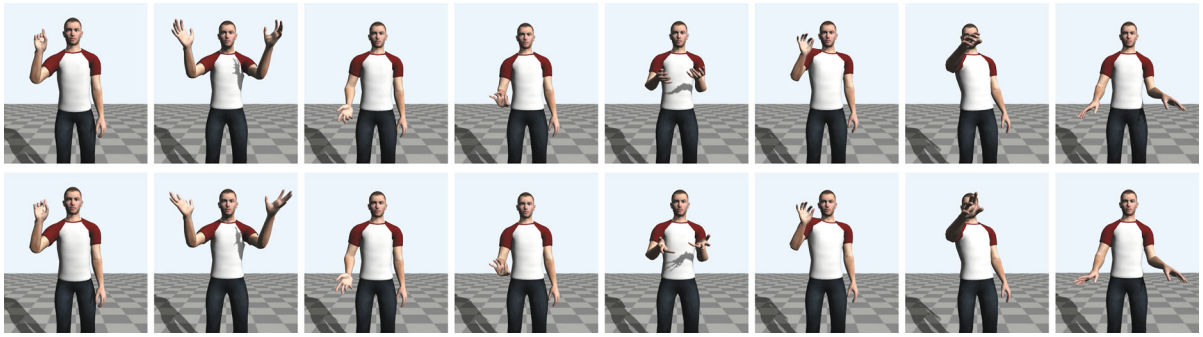

Fig. 5 Examples of correctly estimated finger gestures. The input gesture that was used as a reference is on the *upper row* and the estimated gesture based on the proposed methodology is on the *lower row*. From *left* to *right* attention, big, doubt shrug, OK, palm presentation, small, turn, and wipe

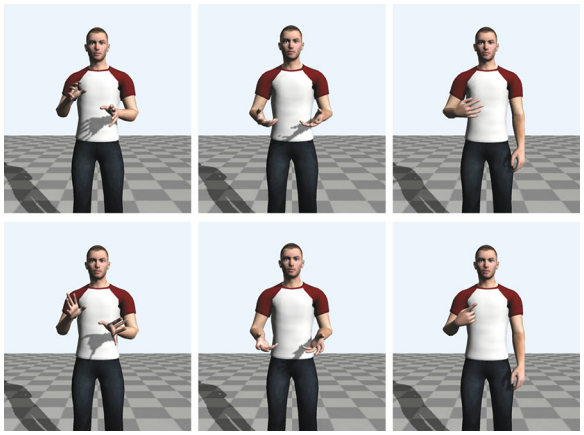

Fig. 6 The ground truth data (*upper row*) and estimated finger gestures with the presented methodology (*lower row*) synthesized with the debate dataset

Fig. 7 The ground truth data (*upper row*) and estimated finger gestures with the presented methodology (*lower row*) synthesized with the directions dataset

datasets. The resulting motion sequences are presented in the accompanying video. Moreover, Fig. 6 illustrates sample gestures synthesized using different datasets (Fig. 7).

7 Conclusions and Future Work

In this paper, a methodology for estimating and synthesizing the motion of a character's fingers

through learning motion features was presented. Firstly, a number of motion features from a character's hand are computed and then assigned to a feature vector. In this paper, we proposed a two-step process. In the first step, the motion features are pre-processed using RBM, and in the second step, we used SVM to optimize a distance function. The experiments presented have shown that RBM feature transformation improves the results of similarity learning.

By comparing the presented methodology to a previous solution, we find that a better estimation rate can be achieved. The finger motion estimation process remains a challenge for us. The current training process was performed using a small dataset. In our future work, we would like to extend the dataset used by adding more gesture types, as well as to explore alternative deep learning techniques to predict finger motion. Additionally, the exploration of motion features is an open challenge for us. We would also like to compare the presented set of features with different feature sets that have been used for full-body motion classification, such as by using Laban Motion Analysis (LMA) [52] features to associate the body emotion with the finger motion. Finally, correlation analysis between finger and hand motion can also provide results that would be used later to understand the relation between the motion of the hand and the motion of the fingers, which is an additional direction that we would like to investigate in our future work.

References

1. Courgeon, M., Buisine, S., & Martin, J. C. (2009). Impact of expressive wrinkles on perception of a virtual character's facial expressions of emotions. In Z. Ruttkay, M. Kipp, A. Nijholt & H. H. Vilhjálmsson (Eds.), *Intelligent Virtual Agents* (pp. 201–214). Berlin, Heidelberg: Springer.
2. Clavel, C., Plessier, J., Martin, J. C., Ach, L., & Morel, B. (2009). Combining facial and postural expressions of emotions in a virtual character. In Z. Ruttkay, M. Kipp, A. Nijholt & H. H. Vilhjálmsson (Eds.), *Intelligent virtual agents* (pp. 287–300). Berlin, Heidelberg: Springer.
3. Jörg, S., Hodgins, J. K., & O'Sullivan, C. (2010). The perception of finger motions. In *Proceedings of the 7th Symposium on Applied Perception in Graphics and Visualization, APGV 2010, Los Angeles, California* (pp. 129–133).
4. Hoyet, L., Ryall, K., McDonnell, R., & O'Sullivan, C. (2012). Sleight of hand: Perception of finger motion from reduced marker sets. In itACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (pp. 79–86). ACM.
5. Mousas, C., Newbury, P., & Anagnostopoulos, C.-N. (2014). Evaluating the covariance matrix constraints for data-driven statistical human motion reconstruction. In *Spring Conference on Computer Graphics* (pp. 99–106).
6. Mousas, C., Newbury, P., & Anagnostopoulos, C.-N. (2014). Data-driven motion reconstruction using local regression models. In *IFIP International Conference on Artificial Intelligence Applications and Innovations* (pp. 364–374). Berlin, Heidelberg: Springer.
7. Kang, C., Wheatland, N., Neff, M., & Zordan, V. B. (2012). Automatic hand-over animation for free-hand motions from low resolution input. In *Motion in Games—5th International Conference, MIG 2012, Rennes, France, Proceedings* (pp. 244–253).
8. Wheatland, N., Jörg, S., & Zordan, V. B. (2013). Automatic hand-over animation using principle component analysis. In *Motion in Games, MIG '13, Dublin, Ireland* (pp. 197–202).
9. Mousas, C., Newbury, P., & Anagnostopoulos, C. N. (2014). Efficient hand-over motion reconstruction. In *International Conferences in Central Europe on Computer Graphics, Visualization and Computer Vision* (pp. 111–120).
10. Jörg, S., Hodgins, J. K., & Safonova, Alla. (2012). Data-driven finger motion synthesis for gesturing characters. *ACM Transactions on Graphics*, *31*(6), 189.
11. Mousas, C., Anagnostopoulos, C.-N., & Newbury, P. (2015). Finger motion estimation and synthesis for gesturing characters. In *Spring Conference on Computer Graphics* (pp. 97–104) ACM.
12. Jörg, S., Hodgins, J., & O'Sullivan, C. (2017). Finger motion database. http://people.cs.clemson.edu/sjoerg/fms.html.
13. Wheatland, N., Wang, Y., Song, H., Neff, M., Zordan, V. B., & Jörg, S. (2015). State of the art in hand and finger modeling and animation. *Computer Graphics Forum*, *34*(2), 735–760.
14. Jörg, S. (2016). Data-driven hand animation synthesis. In B. Müller, S. I. Wolf, G.-P. Brueggemann, Z. Deng, A. McIntosh, F. Miller, & W. Scott Selbie (Eds.), *Handbook of Human Motion*. Berlin: Springer.
15. Jin, G., & Hahn, J. K. (2005). Adding hand motion to the motion capture based character animation. In *Advances in Visual Computing, First International Symposium, ISVC 2005, Lake Tahoe, NV, USA, Proceedings* (pp. 17–24).
16. Bitan, M., Jörg, S., & Kraus, S. (2016). Data-driven finger motion synthesis with interactions. Eurographics Association: In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.
17. Mousas, C., & Anagnostopoulos, C.-N. (2017) Real-time performance-driven finger motion synthesis. *Computers & Graphics, 65*, 1–11.
18. Majkowska, A., Zordan, V. B., & Faloutsos, P. (2006). Automatic splicing for hand and body animations. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2006, Vienna, Austria*, (pp. 309–316).
19. van Basten, B., & Egges, A. (2012). Motion transplantation techniques: A survey. *IEEE Computer Graphics and Applications*, *32*(3), 16–23.
20. Mousas, C., Newbury, P., & Anagnostopoulos, C. N. (2013). Splicing of concurrent upper-body motion spaces with locomotion. *Procedia Computer Science*, *25*, 348–359.

21. Mousas, C., & Newbury, P. (2012). Real-time motion synthesis for multiple goal-directed tasks using motion layers. In *Virtual Reality Interaction and Physical Simulation* (pp 79–85). Eurographics Association.
22. Ye, Y., & Liu, C. K. (2012). Synthesis of detailed hand manipulations using contact sampling. *ACM Transactions on Graphics*, *31*(4), 41:1–41:10.
23. Bai, Y., Liu, & C. K. (2014). Dexterous manipulation using both palm and fingers. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China* (pp. 1560–1565).
24. Wei, X. K., Zhang, P., & Chai, J. (2012). Accurate realtime full-body motion capture using a single depth camera. *ACM Transactions on Graphics*, *31*(6), 188.
25. Hamer, H., Gall, J., Urtasun, R., & Van Gool, L. (2011). Data-driven animation of hand-object interactions. In *IEEE International Conference on Automatic Face and Gesture Recognition* (pp. 360–367).
26. Zhu, Yuanfeng. (2013). Ajay Sundar Ramakrishnan, Bernd Hamann, and Michael Neff. A system for automatic animation of piano performances. *Journal of Visualization and Computer Animation*, *24*(5), 445–457.
27. ElKoura, G., & Singh, K. (2003). Handrix: Animating the human hand. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (pp. 110–119).
28. Liu, C. K. (2009). Dextrous manipulation from a grasping pose. *ACM Transactions on Graphics*, *28*(3), 59.
29. Pollard, N. S., & Zordan, V. B. (2005). Physically based grasping control from example. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2005, Los Angeles, CA, USA* (pp. 11–318).
30. Andrews, S., & Kry, P. G. (2013). Goal directed multi-finger manipulation: Control policies and analysis. *Computers & Graphics*, *37*(7), 830–839.
31. Kyota, F., & Saito, S. (2012). Fast grasp synthesis for various shaped object. *Computer Graphics Forum*, *31*(2), 765–774.
32. Kry, P. G., & Pai, D. K. (2006). Interaction capture and synthesis. *ACM Transactions on Graphics*, *25*(3), 872–880.
33. Neff, M., Seidel, H.-P. (2006). Modeling relaxed hand shape for character animation. In *Articulated Motion and Deformable Objects, 4th International Conference, AMDO 2006, Port d'Andratx, Mallorca, Spain, July 11–14, 2006, Proceedings* (pp. 262–270).
34. Tsang, W., Singh, K., & Fiume, E. (2005). Helping hand: An anatomically accurate inverse dynamics solution for unconstrained hand motion. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2005, Los Angeles, CA, USA* (pp. 319–328).
35. Zhao, W., Zhang, J., Min, J., & Chai, J. (2013). Robust realtime physics-based motion control for human grasping. *ACM Transactions on Graphics*, *32*(6), 207.
36. McNeill, D. (1992). *Hand and mind: What gestures reveal about thought*. Chicago, IL: University of Chicago Press.
37. Kendon, A. (2004). *Gesture: Visible Action as Utterance*. Cambridge: Cambridge University Press.
38. Mousas, C., Newbury, P., & Anagnostopoulos, C. N. (2014). Analyzing and segmenting finger gestures in meaningful phases. In *International Conference on Computer Graphics, Imaging and Visualization*, (pp. 89–94).
39. Seol, Y., O'Sullivan, C., & Lee, J. (2013). Creature features: Online motion puppetry for non-human characters. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (pp. 213–221). ACM.
40. Raptis, M., Kirovski, D., & Hoppe, H. (2011). Real-time classification of dance gestures from skeleton animation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (pp. 147–156). ACM.
41. Jin, Y., & Prabhakaran, B. (2008). Semantic quantization of 3d human motion capture data through spatial-temporal feature extraction. In *International Conference on Multimedia Modeling*. Berlin, Heidelberg: Springer.
42. Caridakis, G., Karpouzis, K., Drosopoulos, A., & Kollias, S. (2010). Somm: Self organizing markov map for gesture recognition. *Pattern Recognition Letters*, *31*(1), 52–59.
43. Elmezain, M., Al-Hamadi, A., Appenrodt, J., & Michaelis, B. (2008). A hidden markov model-based continuous gesture recognition system for hand motion trajectory. In *International Conference on Pattern Recognition* (pp. 1–4).
44. Smolensky, P. (1986). *Information Processing in Dynamical Systems: Foundations of Harmony Theory, Volume 1 of Parallel Distributed Processing*. Cambridge, MA: MIT Press.
45. Winkler, G. (2012). *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods: A Mathematical Introduction* (Vol. 27). Berlin: Springer Science & Business Media.
46. Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, *14*(8), 1771–1800.
47. Stober, S., & Nürnberger, A. (2013). An experimental comparison of similarity adaptation approaches. In *International Workshop on Adaptive Multimedia Retrieval* (pp. 96–113). Berlin, Heidelberg: Springer.
48. Schultz, M., & Joachims, T. (2003). Learning a distance metric from relative comparisons. *Neural Information Processing Systems*, *1*, 2.
49. Karpathy, A. Code for training Restricted Boltzmann Machines (RBM) and Deep Belief Networks in MATLAB from https://code.google.com/p/matrbm/.
50. Joachims, T. (2008). Svm-lights from http://svmlight.joachims.org/.
51. Kovar, L., Gleicher, M., & Pighin, F. (2002). Motion graphs. *ACM Transactions on Graphics*, *21*(3), 473–482.
52. Aristidou, A., Charalambous, P., & Chrysanthou, Y. (2015). Emotion analysis and classification: Understanding the performers' emotions using the lma entities. *Computer Graphics Forum*, *34*(6), 262–276.