Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Skeleton-based action recognition with extreme learning machines

Xi Chen*, Markus Koskela

Department of Information and Computer Science, Aalto University, School of Science, P.O. Box 15400, FI-00076 Aalto, Finland

ARTICLE INFO

Article history:
Received 9 August 2013
Received in revised form
11 October 2013
Accepted 23 October 2013
Available online 10 September 2014

Keywords:
Action and gesture recognition
Extreme learning machine
Support vector machine
Motion capture
RGB-D
HDM05

ABSTRACT

Action and gesture recognition from motion capture and RGB-D camera sequences has recently emerged as a renowned and challenging research topic. The current methods can usually be applied only to small datasets with a dozen or so different actions, and the systems often require large amounts of time to train the models and to classify new sequences. In this paper, we first extract simple but effective frame-level features from the skeletal data and build a recognition system based on the extreme learning machine. We then propose three modeling methods for post-processing the classification outputs to obtain the recognition results on the action sequence level. We test the proposed method on three public datasets ranging from 11 to 40 action classes. For all datasets, the method can classify the sequences with accuracies reaching 96–99% and with the average classification time for one sequence on a single computer core around 4 ms. Fast training and testing and the high accuracy make the proposed method readily applicable for online recognition applications.

1. Introduction

The recognition of human actions and gestures has raised considerable interest and has became a widely studied research topic during the last two decades [1–3]. The general aim in the field is to provide automated analysis of various kinds of human activities. Starting from either video, motion capture, depth data, or some combination of these modalities, many action and gesture recognition methods have been developed and applied in various applications, such as surveillance, human–computer interfaces, gaming, and analysis of sign language. Especially during the last 3 years, as the revolutionary low-cost RGB-D (RGB and depth) camera sensors such as the Microsoft Kinect have prevailed in consumer markets, a lot of attention has been drawn also towards research in this area. In this paper, we focus on action and gesture recognition of both motion capture and Kinect data using a generic approach that can be applied similarly to both data sources.

Motion capture (mocap) systems capture human motion with high frequency and accuracy by calculating the joints' coordinates and the angular information of the human skeleton using a system setup consisting of multiple calibrated cameras in a dedicated space [4]. Mocap motion classification is directly related to the recognition of human actions and gestures, and can also assist in tasks like the retrieval of similar motions from existing collections of motion data. Motion capture is often used in fields such as filmmaking, computer animation, sports science, and game development. On the other hand, the recently introduced commodity RGB-D sensors provide depth information along with the standard RGB video and are now widely used e.g. in gaming, human–computer interfaces, and robotics due to their portability and low cost. Several algorithms have been developed to extract the human skeleton model from the depth frames in real-time [5,6]. Essentially, these algorithms classify a large 3D point cloud into about a dozen human skeleton joint coordinates. These algorithms provide analogous, albeit noisier, data to mocap, and this enables the classification methodology developed for mocap skeletons to be applied for RGB-D data as well. In this paper, we apply our proposed method for three public datasets of mocap and Kinect data to classify between 11 and 40 different actions and several hundreds of instances of action sequence data.

An action or a gesture recognition system should be independent of the identity of the actor of the actions, the speed of the performance, and the inherent variance in the realizations of the action instances. The system should be able to support a large number of actions and gestures with high recognition accuracy and, especially in the case of online recognition systems, with fast classification performance. Our approach proposed in this paper is to first extract features from the raw skeletal data of each frame, classify the actions on the frame level, and then build a model of the whole action sequence by aggregating the frame-wise results to get the final classification result for the sequence. We normalize the skeletal joints' coordinates in orientation and scale and then calculate a temporal difference of features to form the final frame-wise features. Each frame-level feature is labeled as the same as

* Corresponding author.
E-mail address: xi.chen@aalto.fi (X. Chen).

http://dx.doi.org/10.1016/j.neucom.2013.10.046
0925-2312/© 2014 Elsevier B.V. All rights reserved.

the action or gesture which it belongs to and is classified with the extreme learning machine (ELM) [7]. ELM is a single-hidden layer feed-forward neural network whose input weights and first hidden layer biases do not need to be learned but are assigned randomly, which makes the learning and classification extremely fast and particularly suitable for online applications. We test the proposed method on multiple databases of Kinect, mocap and even accelerometer data, and observe even almost 100% classification accuracies with a few milliseconds required to classify a single motion sequence.

Our proposed method is different in many aspects over the previously published methods, such as [8–11]. First, we classify a motion initially on the frame level and only make the final classification decision considering the whole sequence, instead of building a feature representation on the sequence level, where one would have to deal with the different lengths of sequences or manually set up a window length and easily result in an extremely high dimensional feature. Second, the proposed method can classify motion sequences with arbitrary speed and length. We do not need to adjust the speed of the motion instances or to fit all sequences into a fixed length. Third, we use ELM for the classification, which provides a very high classification accuracy and, at the same time, the training of the model and the classification are very fast compared to many other non-linear classification methods. In particular, this makes online recognition possible with the proposed method. Moreover, our features are simple and computationally light to calculate. In many previous works, the proposed methods are applied to small datasets with a few different actions, whereas we show results with up to 40 classes while retaining high recognition accuracy. Our method can be applied to other types of data besides mocap and RGB-D skeletal data. We test our system in this paper also with accelerometer data.

This work is a continuation of our previous work in [12]. We continue to use the same dataset and present improved results due to the proposed posterior modeling of the classifier outputs on the sequence level. In this paper, we also compare the extreme learning machine with other classification algorithms, and present results with two other public datasets.

The rest of the paper is organized as follows. In Section 2, we review previous work related to action and gesture recognition from mocap and RGB-D data. We introduce our feature extraction and classification framework in Section 3, including the ELM algorithm and the three posterior modeling methods of the classifier outputs. In Section 4, we provide experiments on three public datasets. The experimental results demonstrate the effectiveness of our proposed framework with comparisons to other published results. Finally, we conclude the paper and discuss possibilities for future work in Section 5.

2. Related work

In the past decades, a lot of research has been conducted on motion classification for motion capture data. However, especially during the last few years, as RGB-D cameras such as the Microsoft Kinect have appeared and became popular, gesture recognition from RGB-D data has prevailed in the research community. Motion classification is also closely related to motion retrieval [13,14], where similar motions are retrieved from large databases in order to be able to reuse the expensive motion data. This section presents a brief review of existing approaches to motion classification and recognition based on either mocap data or skeleton models extracted from an RGB-D sensor.

In action and gesture recognition, the distinctiveness of the used features strongly influences the recognition effectiveness. There are many kinds of features that can be extracted from mocap and RGB-D skeletal data. The most often used ones can be categorized into two groups: features based on the angular information of the joints and features based on the 3D coordinates of the joints. In different recognition systems, the features are also structured in various ways to form the representation of the motion sequences.

In [15], the *x*-, *y*-, and *z*-axis rotations for selected bones are recorded, and all dimensions of the angular value from the whole motion sequence are used to form the feature vector. In [16], the 3D angles of 20 joints are used but all the angles from a single frame are combined as the feature vector for that frame. The whole motion sequence is represented by a matrix of all feature vectors from each frame in the sequence. In [17], the angles from the vertical axis of the hip-center to the rest of the joints are measured, resulting in three angular values for each joint. In total, all 3D angles from 19 joints are combined together as the feature vector for each frame. In [9], the joint angles from 21 joints in the skeleton for each frame are also calculated. Each sequence is partitioned into temporal windows. The variance of the joint angles in the temporal windows is calculated as a local temporal feature descriptor. Each sequence is then represented by a set of feature descriptors. Finally, all features are quantized into 20 codewords, and each sequence is represented by a codeword histogram. In [18], a sophisticated motion feature is designed, which fits the full torso with a single frame of reference. Based on the reference frame, a pair of angles is calculated for each joint and finally the motion sequence is represented by a 19D vector.

The 3D joint positions are often used as features; either the *x*,*y*,*z* coordinates are used directly without any post processing [8,19,20], or they are normalized to be invariant to orientation and scale [11,21,22]. The joint positions can also be used to calculate the distances between each joint, forming a distance matrix as the feature for each frame [10], or just to calculate the distance from one joint to the others as the feature for that joint [21]. In [23], the distances from four joints to the body centroid are calculated and with a time stamp these form a 13D feature vector. All feature vectors in the sequence are then concatenated together, and their log-covariance is calculated to represent the sequence. In addition to the distances between the joints, the velocities of the joints can also be calculated based on the joint coordinates [24]. In [3], the authors propose the motion template as a feature. The feature vector of each frame is a 39D boolean vector, describing the geometric relations between certain joints of the human body. Each motion sequence is represented by a feature matrix where each column is a frame-wise feature vector.

Different action and gesture recognition systems use different classification algorithms, which are also often directly related to the extracted features. Due to the characteristics of mocap data, the extracted features are often time series signals, and dynamic time warping (DTW) is widely applied to calculate distances between motion sequences. In [3], DTW is used to form a motion template as a representation of an action. When a motion sequence needs to be classified as a certain action or some motion sequences representing a certain action need to be retrieved from a database, the subsequence variant of DTW is used to calculate the distances between the feature matrices of the sequences and the motion template of each action. Similarly, [16] uses open-ended DTW to obtain the distance of joint angle vectors between the sequence and each action. In [15], DTW is used to calculate the distances between the angle vectors of each bone and the codebook representation of each action.

Alternatively to the DTW-based distance comparisons, another commonly used method is to build a dictionary or codebook via clustering from the training data. The motion sequences are then represented as histograms or collections of keywords and classified with supervised classifiers. SVM is a popular option, as e.g. in [20]. In [8], the features are extracted from sub-windows around extrema points and clustered with *k*-means to form action primitive dictionaries. The original time series signal forms a histogram from the dictionary. The histograms from each dimension are stacked to create a single descriptor for each sequence, which is classified with a multi-class χ^2-kernel SVM. Besides SVMs, some other classifiers have also been used in action and gesture recognition systems. In [17], the feature vectors are divided into 60 clusters by a Gaussian mixture model. Each sequence is represented as a sequence of clusterings, and the motion streams are segmented and recognized by a threshold model with a conditional random field. Nearest neighbor classifiers are used in [23]. In [19], a three-layer hierarchical self-organizing map is used to learn the gestures, with the first layer learning the posture, the second layer learning the gesture elements, and third layer learning the gestures.

Another schema often used with the dictionary-based approaches is to build a graph for classifying the motion sequences. In [10], the distance vectors are clustered into sets of salient postures. An action graph is built by the salient postures as nodes and each action is modeled as a path in the graph. Ref. [22] decomposes the features to get the action segments, which form a graph. Each sequence is then represented by the transitions between action segments from the starting node to the end node. Classifying a sequence corresponds to finding a path in the graph with a score associated at each step. The scores are accumulated to determine whether an action was performed. In [24], a finite state machine is used, and each sequence is coded by one of several states. The classification is carried out by a tree-based search to match the stream of states.

Extreme learning machine (ELM) [7] has been successfully used for solving many classification problems. For example, in [25], ELM was used to recognize human activities from video data. Even though ELM is already very fast to train and test, parallelized ELM with GPUs for large-scale data [26] can make ELM more feasible for large datasets. In our work, ELM is, as far as the authors are aware, for the first time applied to human action and gesture recognition from mocap and RGB-D skeletal data. The experiments show that in our system ELM performs very favorably against other commonly used classifiers when considering both the accuracy and time requirements.

3. System framework

3.1. System overview

A graphical overview of the proposed recognition system can be seen in Fig. 1. The system can be divided into two parts: feature extraction (Section 3.2) and classification (Section 3.3). After the ELM classification stage (Section 3.3.1), there are three alternative posterior models of the classifier outputs on the sequence level: the frame-wise voting model, the sequence histogram model, and the sequence probability model. Of these, the sequence histogram model is based on the frame-wise voting model. The models are described in Section 3.3.2.

3.2. Feature extraction

The distinctiveness of the features significantly influences the effectiveness of the classification system. On the other hand, simpler features make feature extraction easier, reducing the complexity and speeding-up the computation, enabling the possibility of online recognition. On this premise, we extract straightforward features that do not require any complex calculations.

3.2.1. Normalized 3D joint positions

The output from a motion capture system contains very rich raw information about the motion sequences, such as the structure relations of the joints and translation and rotation information. Intuitively, the 3D coordinates of the joints are more sensible to humans and they can easily be calculated from the raw data. However, the joint coordinates depend on the used coordinate system which is different in every recording environment, and, even in the same coordinate system, multiple instances of the same posture will likely have different coordinates due to translation and rotation. The same posture done by two performers can also have different coordinates e.g. due to the different body sizes. We thus first need to register the data into a common coordinate system to make the joints' coordinates comparable to each other.

The skeleton structures also differ from each other in different motion captures or RGB-D skeleton datasets. For example, in the CMU Graphics Lab Motion Capture Database [27] and the Motion Capture Database HDM05 [28] the skeletons contain 31 joints, the Microsoft Kinect SDK [29] gives 20 joints, and the PrimeSense Natural Interaction Middleware (NiTE) [30] provides 15 joints for the RGB-D skeletons. The skeleton structures from these systems are illustrated in Fig. 2. However, even though the number of joints differ, the essential joints are available in all skeletons. These include the hands, feet, elbows, knees and especially the root, which we use as the key joint for normalizing the orientation and translation of the skeletons. The root joints are marked in the skeletons in Fig. 2.

The public mocap datasets are often available in ASF/ACM and BVH formats [31]. To obtain a common coordinate system, different formats require different calculations. For ASF/ACM, the rotation matrix $\mathbf{R}_1$ and the translation vector $\mathbf{t}_1$ of the root should be set to identity and zero, i.e. $\mathbf{R}_1 = \mathbf{I}$ and $\mathbf{t}_1 = [0\ \ 0\ \ 0]^T$. For BVH, we first set the rotation matrix of the root to identity, calculate the coordinates of the joints, and translate the root joint to the

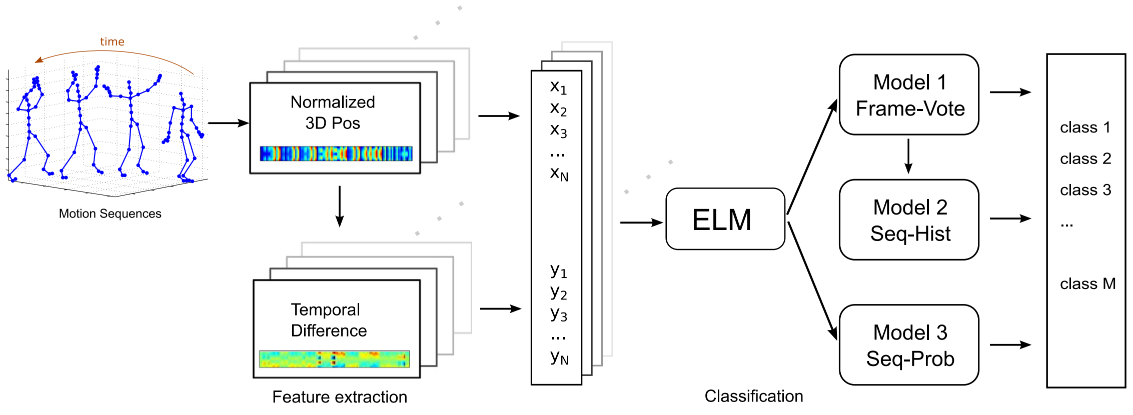


Fig. 1. An overview of the gesture recognition framework.

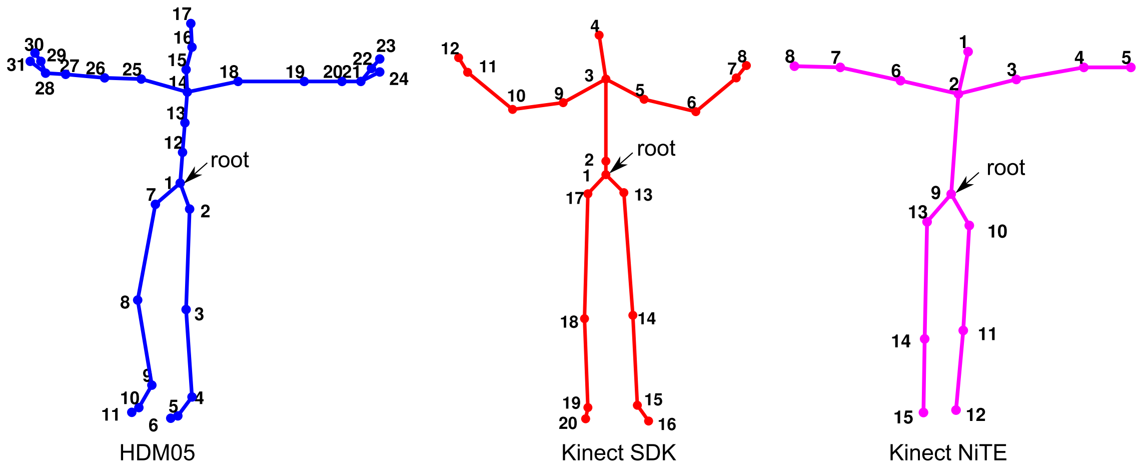


Fig. 2. Skeleton models from different sources.

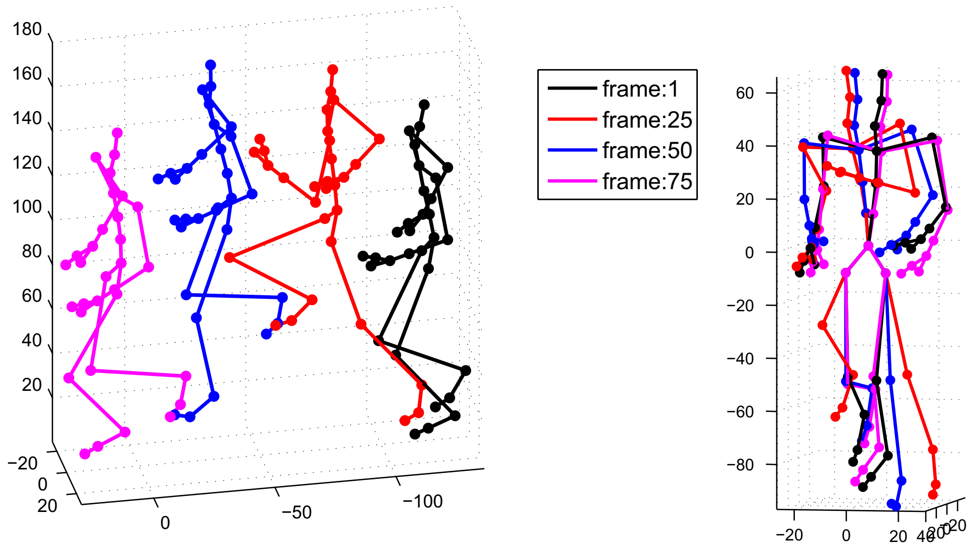


Fig. 3. A *HopLeftLeg* action in original (left) and root (right) coordinates.

origin. These transformations rotate all skeletons into the same orientation in the coordinate system and translate the root of the skeleton to the origin so that the hips coordinates of the same actor will overlap regardless of the posture. For example, Fig. 3 shows sampled frames from a *HopLeftLeg* action from the HDM05 database in the original and root coordinates.

For skeletons extracted from RGB-D sensor output, usually only the 3D coordinates of the joints are available. In order to transform the joints into the same coordinate system, we randomly select one skeleton as the standard skeleton, and use its coordinates as the common basis, then transform all the other skeletons in the sequence to it.

We start by translating the root of the skeleton to the origin by subtracting the coordinates of the root from all the joints. Then let us assume that the left and the right hip of the standard skeleton are $\mathbf{p}^0_{lhip}$ and $\mathbf{p}^0_{rhip}$, and of the skeleton to be transformed $\mathbf{p}_{lhip}$ and $\mathbf{p}_{rhip}$. The rotation matrix which transforms the joints to the common coordinates is $\mathbf{R}$. Now, $\mathbf{p}^0_{lhip}$ and $\mathbf{p}^0_{rhip}$ form a plane c that can be represented by the origin of the coordinates and the norm vector $\mathbf{n}_c$

$$\mathbf{n}_c = \mathbf{p}^0_{lhip} \times \mathbf{p}^0_{rhip}. \tag{1}$$

After the rotation, the left and the right hip have to lie on the plane c, and we minimize the sum of the distances between the transformed hips to the standard hips, which can be expressed as

$$\mathbf{n}_c \cdot (\mathbf{R}\mathbf{p}_{lhip}) = 0$$

$$\mathbf{n}_c \cdot (\mathbf{R}\mathbf{p}_{rhip}) = 0$$

$$\min_{\mathbf{R}}(\|\mathbf{R}\mathbf{p}_{lhip} - \mathbf{p}^0_{lhip}\| + \|\mathbf{R}\mathbf{p}_{rhip} - \mathbf{p}^0_{rhip}\|). \tag{2}$$

The transformed 3D positions of the whole skeleton can then be obtained by multiplying all joints with $\mathbf{R}$.

Finally, the coordinates need to be normalized to be invariant to the size of the actors. The skeletons are thus normalized so that the sum of the distances of the connected joints is equal to one.

3.2.2. *Temporal difference of feature vectors*

Some distinct actions or gestures can be very similar to each other on the frame level. For example, Fig. 4 shows the actions *StandUpFloor* and *SitDownFloor* from the HDM05 dataset (Section 4.1). They contain almost identical frames but reversed in time. Therefore we calculate a feature expressing temporal difference information with a fixed temporal offset within an action sequence.

Let us assume the features of the kth frame in a sequence with K frames calculated as described in the previous section are

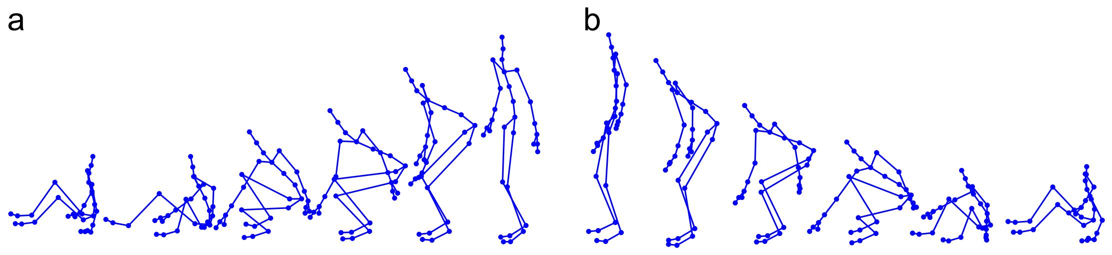


Fig. 4. One pair of inverse actions. (a) *StandUpFloor* and (b) *SitDownFloor*.

$\mathbf{x}_k^d$. The *temporal difference of feature vectors* $\mathbf{x}_k^{td}$ can be calculated as

$$\mathbf{x}_k^{td} = \begin{cases} \mathbf{x}_k^d & 1 \le k < k' \\ \dfrac{\mathbf{x}_k^d - \mathbf{x}_{k-k'+1}^d}{\| \mathbf{x}_k^d - \mathbf{x}_{k-k'+1}^d \|} & k' \le k \le K \end{cases} \tag{3}$$

where k' is the temporal offset parameter, $1 < k' < K$.

The final features $\mathbf{x}$ of a frame are the concatenation of the feature vector $\mathbf{x}^d$ and the temporal distance feature vector $\mathbf{x}^{td}$, $\mathbf{x} = [(\mathbf{x}^d)^T \; (\mathbf{x}^{td})^T]^T$. The dimensionality of the feature depends on the number of selected joints. Assuming that n_j joints are selected, the dimensionality of $\mathbf{x}$ is $n = 3 \cdot 2 \cdot n_j$.

3.3. Learning system

The proposed learning system can be divided into two steps, the classification of the frame-wise features and the posterior modeling of the classifier outputs on the sequence level. Let us assume that there are M actions $\mathcal{A} = \{A_1, \dots, A_M\}$ and let us define $y_m \in \{0, 1\}$, $1 \le m \le M$. If y_m is one, then the sequence belongs to the action A_m, otherwise it does not. The row vector $\mathbf{y} = [y_1 \; \dots \; y_m \; \dots \; y_M]$ indicates the action that the sequence belongs to. For each action there are multiple motion sequence examples, and each sequence s is represented by the features of its frames, calculated as described in Section 3.2. That is, $s = \{\mathbf{x}_1, \dots, \mathbf{x}_k, \dots, \mathbf{x}_K\}$, where K is the number of frames. Now, all $(\mathbf{x}_k, \mathbf{y})$ form a training input–output pairs for the classifier. In our system, we use extreme learning machine to classify the features on the frame level.

3.3.1. Extreme learning machine

Extreme learning machine (ELM) [32] belongs to the class of single-hidden layer feed-forward neural networks. In ELM, the input weights and first hidden layer biases do not need to be learned but are assigned randomly, which makes the learning extremely fast.

Given P samples $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^P$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $\mathbf{y}_i \in \mathbb{R}^M$, the standard ELM model with L hidden neurons can be represented as

$$\mathbf{y}_i = f(\mathbf{x}_i) = \sum_{j=1}^{L} \boldsymbol{\beta}_j g(\boldsymbol{\omega}_j \cdot \mathbf{x}_i + b_j), \tag{4}$$

where $g(\cdot)$ is a nonlinear activation function, $\boldsymbol{\beta}_j \in \mathbb{R}^M$ are the output weights, $\boldsymbol{\omega}_j \in \mathbb{R}^n$ is the weight vector connecting the input layer to the jth hidden neuron and b_j is the bias of the jth hidden neuron. Both $\boldsymbol{\omega}_j$ and b_j are assigned randomly during the learning process. With $\mathbf{Y} = [\mathbf{y}_1^T \; \mathbf{y}_2^T \cdots \mathbf{y}_P^T]^T \in \mathbb{R}^{P \times M}$ and $\boldsymbol{\beta} = [\boldsymbol{\beta}_1^T \; \boldsymbol{\beta}_2^T \cdots \boldsymbol{\beta}_L^T]^T \in \mathbb{R}^{L \times M}$, Eq. (4) can be written compactly as

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{Y}, \tag{5}$$

where the hidden layer output matrix $\mathbf{H}$ is

$$\mathbf{H} = \begin{bmatrix} g(\boldsymbol{\omega}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\boldsymbol{\omega}_L \cdot \mathbf{x}_1 + b_L) \\ \vdots & \ddots & \vdots \\ g(\boldsymbol{\omega}_1 \cdot \mathbf{x}_P + b_1) & \cdots & g(\boldsymbol{\omega}_L \cdot \mathbf{x}_P + b_L) \end{bmatrix}_{P \times L}. \tag{6}$$

Assuming that $L < P$, there is generally no $\boldsymbol{\beta}$ that would be an exact solution to (5). The smallest-norm least squares solution is given [32] as

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^\dagger \mathbf{Y}, \tag{7}$$

where $\mathbf{H}^\dagger$ is the Moore–Penrose generalized inverse of matrix $\mathbf{H}$.

3.3.2. Post-learning models

Given a test sequence $t = \{\mathbf{x}_1, \dots, \mathbf{x}_q, \dots, \mathbf{x}_Q\}$, ELM gives the output (Eq. (4)) of each class for each frame as $\hat{c}_{q,m}$. The whole output of the ELM for t can therefore be written in a matrix form as

$$\boldsymbol{\Omega} = \begin{bmatrix} \hat{\mathbf{c}}_1 \\ \hat{\mathbf{c}}_2 \\ \vdots \\ \hat{\mathbf{c}}_q \\ \vdots \\ \hat{\mathbf{c}}_Q \end{bmatrix} = \begin{bmatrix} \hat{c}_{1,1} & \hat{c}_{1,2} & \dots & \hat{c}_{1,m} & \dots & \hat{c}_{1,M} \\ \hat{c}_{2,1} & \hat{c}_{2,2} & \dots & \hat{c}_{2,m} & \dots & \hat{c}_{2,M} \\ \vdots & & & \vdots & & \vdots \\ \hat{c}_{q,1} & \hat{c}_{q,2} & \dots & \hat{c}_{q,m} & \dots & \hat{c}_{q,M} \\ \vdots & & & \vdots & & \vdots \\ \hat{c}_{Q,1} & \hat{c}_{Q,2} & \dots & \hat{c}_{Q,m} & \dots & \hat{c}_{Q,M} \end{bmatrix}. \tag{8}$$

After obtaining the output matrix $\boldsymbol{\Omega}$, we calculate the final classification $\hat{\mathbf{y}} = [\hat{y}_1 \; \dots \; \hat{y}_m \; \dots \; \hat{y}_M]$, where $\hat{y}_m \in \{0, 1\}$, $1 \le m \le M$, of the sequence t by three different modeling methods. We have $\sum_{i=1}^M \hat{y}_i = 1$, and if $\hat{y}_m = 1$, t is classified to the class A_m.

Frame-wise voting model: In the frame-wise voting method, each frame is first classified to the action class which has the largest output value for that frame. Let us define $\hat{f}_{q,m} \in \{0, 1\}$ as a binary variable indicating whether the qth frame of t was classified to action A_m. Each frame is classified as

$$\hat{f}_{q,m} = \begin{cases} 1 & \text{if } m = m' \quad \text{where } m' = \arg\max_i(\hat{c}_{q,1}, \dots, \hat{c}_{q,i}, \dots, \hat{c}_{q,M}) \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

After each frame is classified, the final class of the whole sequence t is determined by majority voting, that is, as the class which has the largest number of frames. Let us define u_m as the number of frames classified as action A_m

$$u_m = \sum_{q=1}^{Q} \hat{f}_{q,m}. \tag{10}$$

The sequence t is then classified as

$$\hat{y}_m = \begin{cases} 1 & \text{if} \quad m = m' \quad \text{where } m' = \arg\max_i(u_1, \dots, u_i, \dots, u_M) \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Sequence histogram model: In this model, we begin by normalizing u_m by the total number of frames Q. A sequence s can then be expressed by a normalized histogram as

$$\mathbf{h}_s = \frac{1}{Q}[u_1 \; \dots \; u_m \; \dots \; u_M]. \tag{12}$$

For the L_m training sequences $\{s_m^1, \dots, s_m^j, \dots, s_m^{L_m}\}$ belonging to the action A_m, the corresponding histograms are $H_m = \{\mathbf{h}_m^1, \dots, \mathbf{h}_m^j, \dots, \mathbf{h}_m^{L_m}\}$. By averaging the histograms in H_m, the class A_m can be represented by the histogram

$$\mathbf{h}_m = \frac{\sum_{j=1}^{L_m} \mathbf{h}_m^j}{L_m}. \tag{13}$$

After building a histogram model for all M actions based on the training data, given a testing sequence t, a normalized histogram $\mathbf{h}_t$ can be obtained in a similar way. The distance between $\mathbf{h}_t$ and $\mathbf{h}_m$ can be calculated as

$$d_m = \|\mathbf{h}_t - \mathbf{h}_m\|_1. \tag{14}$$

By calculating the distances between the testing sequence histogram with all the action histograms, the testing sequence is classified to the action with the minimal distance as

$$\hat{y}_m = \begin{cases} 1 & \text{if} \quad m = m' \quad \text{where } m' = \arg\min_i (d_1, \dots, d_i, \dots, d_M) \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Sequence probability model: If the sequence t belongs to an action A_m, every frame in the sequence also belongs to A_m. We can therefore use the joint probability of all frames in a sequence to determine the action of the sequence. We convert the outputs $\hat{c}_{q,m}$ into probabilities with the logistic sigmoid function

$$p(\hat{y}_m = 1|\mathbf{x}_q) = \frac{1}{1+\exp(-\hat{c}_{q,m})}. \tag{16}$$

The joint probability of the sequence is

$$p(\hat{y}_m = 1|t) = p(\hat{y}_m = 1|\mathbf{x}_1, \dots, \mathbf{x}_q, \dots, \mathbf{x}_Q). \tag{17}$$

We assume temporal independence among the frames in a sequence, which means that the class of each frame depends only on the features of that frame. Then, Eq. (17) can be simplified into

$$p(\hat{y}_m = 1|t) = \prod_{q=1}^{Q} p(\hat{y}_m = 1|\mathbf{x}_q). \tag{18}$$

We can now classify the sequence t as

$$\hat{y}_m = \begin{cases} 1 & \text{if} \quad m = m' \quad \text{where } m' = \arg\max_i p(\hat{y}_i = 1|t) \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

4. Experiments

In this section we evaluate our system on three public datasets and compare our results with existing methods in the literature. All the experiments are conducted on an Intel(R) Xeon(R) CPU at 3.3 GHz and 16 GB of memory.

4.1. HDM05 dataset

The HDM05 dataset [28] is one of the largest motion capture datasets publicly available. The included motion sequences are clearly organized and contain large numbers of different actions with multiple samples for each action. Due to the large number of motion sequences in the dataset, researchers often use only certain subsets of the actions [10]. We continue to use the same set of 40 actions as in our previous work [12] to test the new post-learning modeling methods and compare the extreme learning machine with other classifiers.

The actions in HDM05 can be divided into two categories: stationary and locomotive. In stationary actions, the actors do not need to move, for example, as in *clapping hands*. On the other hand, locomotion requires the actors to move around in the space. Typical examples include *jogging* and *walking*. Our subset[1] contains most of the stationary actions and a few locomotions. In total, the subset contains 40 actions, 790 motion sequences, and over 217 000 frames.

[1] The list of the included actions is given in Appendix A.

4.1.1. The selection of the number of hidden neurons

For the extreme learning machine, the number of hidden neurons is the only parameter we need to tune. We use 90% of the data for training and 10% for testing, and try hidden layer sizes ranging from 50 to 1000 neurons with an interval of 50 neurons. The classification accuracies are shown in Fig. 5a, and the corresponding training and testing times in Fig. 5b.

We can observe from Fig. 5a that the frame-level recognition accuracy of Eq. (9) of the training data keeps increasing as the number of hidden neurons increases, but above 400 neurons the frame accuracy of testing data increases only slowly. After applying the frame-wise voting (Section 3.3.2), the accuracy of the motion sequence recognition increases quickly at the beginning and then reaches a plateau with little fluctuation, caused by the randomness of ELM parameter generation and the voting schema. On the other hand, the training and testing times shown in Fig. 5b increase approximately linearly with the number of hidden neurons. For around 200 000 training samples, the training time of ELM with 1000 hidden neurons is about 45 s. Taking the accuracy, timing, and computational complexity into consideration, we select 750 as the number of hidden neurons for this dataset.

4.1.2. Comparison between the three modeling methods

In Section 3.3.2, we proposed three post-learning modeling methods to obtain the final class of an action sequence. We now compare the effectiveness of these methods. After the classification of each frame, we use the same frame-level results with all three models. The experiments are conducted with 10-fold cross-validation and with 750 hidden neurons. The accuracy of the methods can be seen in Fig. 6a. The sequence probability model gives the highest accuracy, slightly outperforming the sequence histogram model. The frame-wise voting model is clearly inferior. Therefore, in the further experiments we will primarily use the sequence probability model.

4.1.3. Comparison with other classifiers

In order to assess the effectiveness of ELM as the classifier in our system, we also experiment with other classification algorithms and record the obtained classification accuracies and training and testing times. We use three popular classifiers: first, *logistic regression*, which is popular due to its low computational complexity. Second, a *linear SVM with an approximate feature map* [33,34], which has reached considerable popularity in the last couple of years. It simplifies the calculation of additive non-linear kernels (such as the χ^2 kernel) by approximation and with a linear SVM, which results in much faster training and testing than with the original non-linear SVM. Third, we also use *RBF-kernel SVM* [35], which has shown its powerfulness in many classification problems. We use the LIBSVM [36] and LIBLINEAR [37] libraries, and apply the sequence probability model.

The classification results are shown in Fig. 6b. Both ELM and RBF-kernel SVM reach over 96% in accuracy, with SVM resulting in 0.7% higher accuracy than ELM. The accuracies of the other two classifiers are much lower.

Table 1 shows the training and testing times for all the classifiers. The average testing time for one motion sequence by ELM is 4.7 ms whereas the testing time with RBF-kernel SVM is more than 500 times greater. The training dataset contains about 200 000 frames, and it takes about 30 s for ELM to train the model and more than 21 min for the kernel SVM. The testing times for the other two methods are slightly smaller than with ELM, and their training is a bit slower.

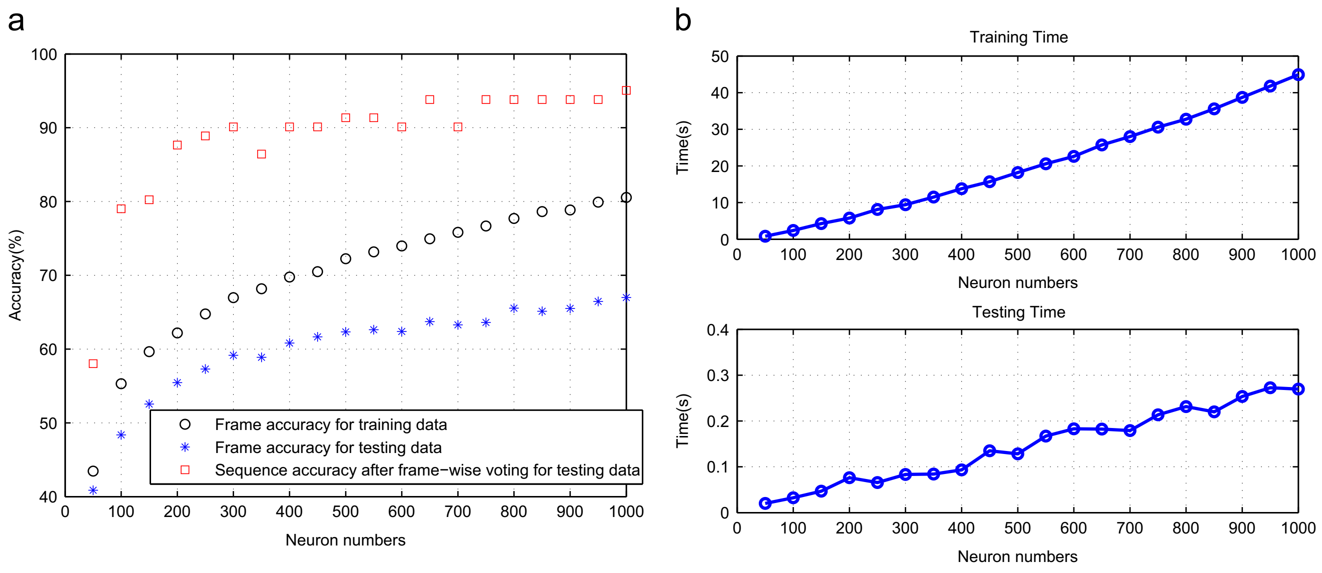


Fig. 5. The relationship between the number of neurons versus classification accuracy and training and testing times. (a) Accuracy and (b) training and testing times.

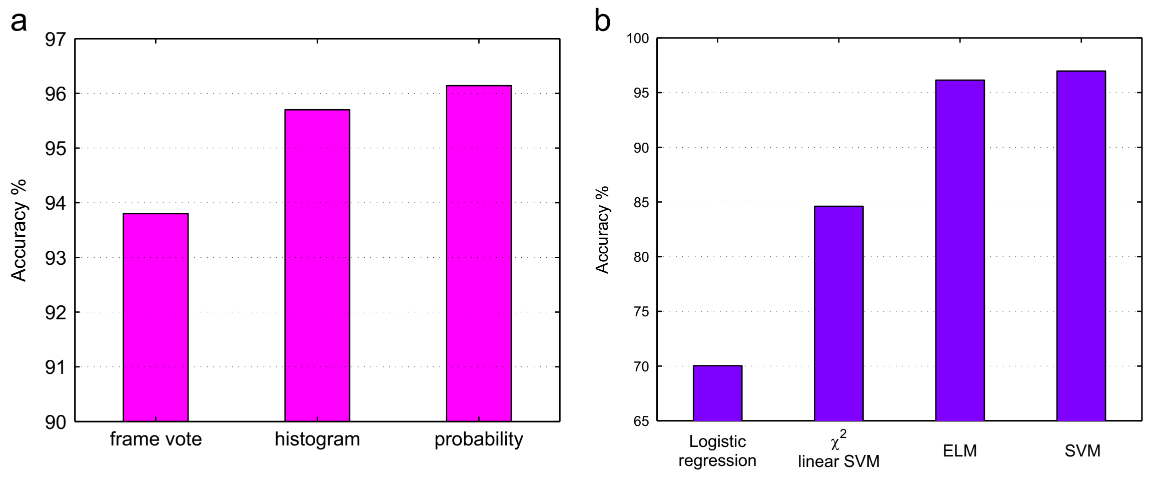


Fig. 6. Comparisons of post-learning modeling methods and classifiers. (a) Accuraciesof used sequence modeling method and (b) Accuracies of different classification.

Table 1
Training and testing times for different classifiers.

Classifier	Logistic regression	χ^2 linear SVM	ELM	SVM
Testing time (ms)	0.88	2.7	4.7	2500
Training time (s)	43	110	31	1300

4.2. *Microsoft Research Cambridge-12 Kinect gesture dataset*

The Microsoft Research Cambridge-12 (MSRC-12) Kinect gesture dataset was collected for the evaluation of different instructions to performers for training gestural systems [38]. In the dataset, five different kinds of instructions were given to the performers for conducting the same kind of actions. In total, 12 actions were collected for each instruction. The description of the actions can be seen in Table 2.

Due to the five different instruction types, the whole dataset can be divided into five groups correspondingly. We use the same data group as in the paper [17]; in this group, the performers were given the instructions from the videos played on the screen in front of them. The gestures were performed by a total of 30 people, and the same gesture was performed repeatedly and recorded as a stream into one sequence. The data collector also specified the frame index to mark the middle points of the actions in the sequence. We cut the sequences containing multiple instances of the same action into multiple files based on the middle point index, so that each file is a motion sequence containing one single instance of an action.

In these experiments, we use features from 15 joints corresponding to the NiTE Kinect skeleton (Section 3.2.1), 500 hidden neurons in the ELM, the sequence probability model, and 10-fold cross validation. The results are shown in Table 3. In [17], a conditional random field threshold model was used to segment and recognize the actions. We therefore only compare for the substitution errors which are equal to the classification error for the motions and which are represented as *S* in the table. *N* is the number of motion sequences for each action, totalling in 1223 motions in the used data group. The average accuracy of our method is 99.3%, the training time is less than 11 s, and the average testing time for a single motion sequence is 3 ms.

In [38], the authors analyzed the correctness of the performances given with different instructions using random forest classifiers and other methods, and concluded that the instructions given in both video and textual form (Video+Text) were superior. Therefore we also use the data group collected based on the Video+Text instructions, which contains 1210 motion sequences. For this data group we get an overall accuracy of 99.8% which is

Table 2
MSRC-12 Kinect gesture dataset and example motions.

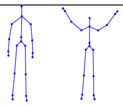
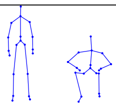
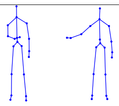
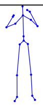
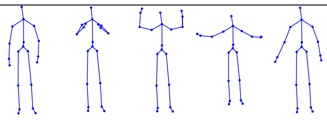
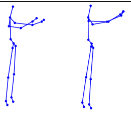
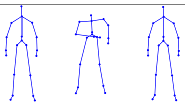
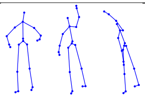
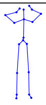
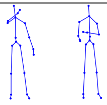
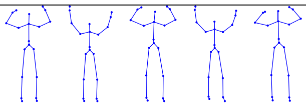
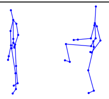
Metaphoric gestures	Main frames	Iconic gestures	Main frames
Start music\ raise volume (G1)		Crouch or hide(G2)	
Navigate to next menu(G3)		Put on night vision googles(G4)	
Wind up the music(G5)		Shoot with a pistol(G6)	
Take a bow to end the session(G7)		Throw an object such as a grenade(G8)	
Protest the music(G9)		Change weapon(G10)	
Lay down the tempo of a song(G11)		Kick to attack an enemy(G12)	

Table 3
Gesture recognition results for MSRC-12.

Gestures	N	S [17]	Our	Gestures	N	S [17]	Our	Gestures	N	S [17]	Our
G1	101	6	0	G5	101	8	6	G9	105	9	0
G2	102	0	0	G6	92	11	0	G10	103	3	1
G3	101	3	1	G7	103	1	1	G11	106	9	0
G4	101	10	0	G8	103	4	0	G12	105	0	0

Table 4
Gesture recognition results for MHAD.

Method (accuracy)	1-NN [9]	3-NN [9]	K-SVM [9]	Our
Motion capture (%)	74.8	75.6	79.9	99.5
Accelerometer (%)	79.2	81.8	85.4	90.7

slightly better than with the video-based instructions. Our results therefore also confirm the conclusions made in [38].

4.3. *Berkeley Multimodal Human Action Database*

The Berkeley Multimodal Human Action Database (MHAD) [9] contains data from a motion capture system, stereo cameras, depth sensors, accelerometers, and microphones. It contains 11 actions performed by 12 actors, yielding around 660 motion sequences and corresponding to about 82 min of recording time. The names of the actions can be seen in Fig. 7. Out of these 11 actions, the last one is a combination of other two actions, *sitdown* and *standup*. Using the sequence probability model and ELM with 500 hidden neurons, the confusion matrix is shown in Fig. 7a. We can see that the actions 9 and 10 are all classified as action 11, a combination of the other two actions. Therefore, for this dataset we use the sequence histogram model. We first train the ELM model using data from the first 10 actions in the training data, as they are independent from each other. Then we feed the training data from all 11 actions into the ELM model, and build the histogram model for each action. The confusion matrix obtained by this method is shown in Fig. 7b.

Recently, due to the popularity and wide use of smart phones, motion recognition based on accelerometer data is also gaining more attention [39,40]. In the MHAD dataset, there are six accelerometers attached to the actors' bodies, with each sensor providing 3D acceleration values. We use also this dataset to classify the actions with the proposed method. We combine the 3D acceleration values of the accelerometers together to form a 18D feature and normalize it to the range $[-1 \quad 1]$, making it conceptually correspond to our normalized 3D joint position feature (Section 3.2.1). We calculate the temporal difference feature (Section 3.2.2) similarly and combine these features together to form a 36D feature vector, and use the sequence histogram model.

The average recognition accuracies with the sequence histogram model are shown in Table 4. The table contains also three other methods using the same dataset from [9]. We can see that for motion capture data we get almost 100% accuracy, with an average classification time for one motion sequence of about 4 ms. For accelerometer data, our system also reaches above 90% in classification accuracy.

5. Conclusions and future work

In this work, we build an action and a gesture recognition system for motion capture and RGB-D skeleton data. We extract

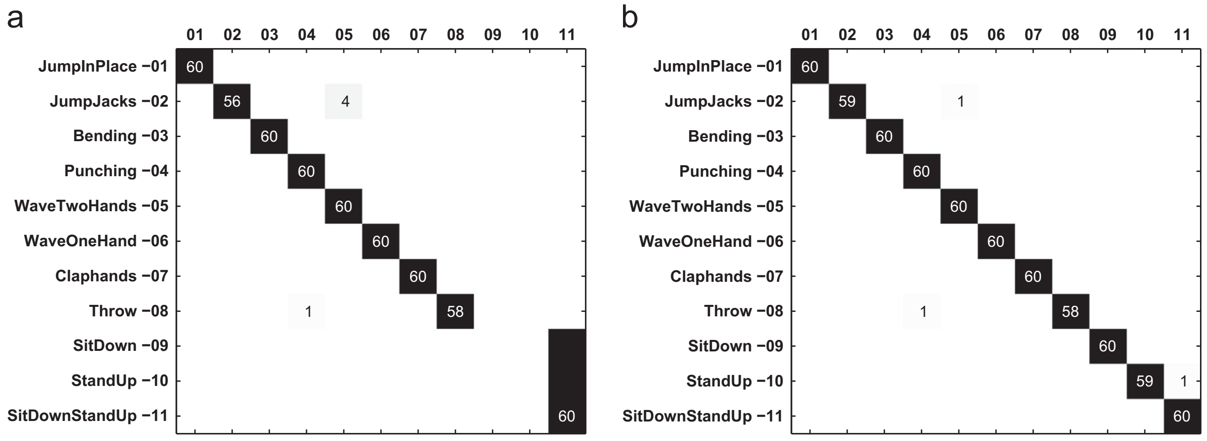


Fig. 7. Confusion matrices for the MHAD experiments. (a) Sequence probability model and (b) sequence histogram model.

simple features that are easy to obtain but still provide distinctive information about the posture. We use extreme learning machine as the classifier, as it results in a high classification accuracy and also has fast training and testing times. Our system can therefore be readily used for online applications.

We compare ELM with logistic regression, approximative kernel map expansion with linear SVM, and RBF-kernel SVM. We observe that ELM can provide a good combination of high accuracy and modest time requirements. Non-linear SVMs can obtain slightly higher recognition accuracies but with considerably higher computational complexity.

After the frame-wise classification, we propose three methods for posterior modeling of the classifier outputs to give a final result for the motion sequence. Experiments show that the sequence probability model generally provides the best classification accuracy, but the sequence histogram model can be used especially for learning combinations of basic actions. We test the proposed methods on three public databases, where they can reach almost 100% accuracy and take about 4 ms to classify a sequence, which indicate that the system is very robust and can be used in real applications.

In the future, we will continue by developing an automatic segmentation method for actions, so that when the actor is performing continuously, we will be able to detect the beginning and ending of the actions. In addition, instead of running the recognizer after the whole action has been performed, we will extend the system to predicting the actions during performance, which will provide further valuable information on online applications.

Acknowledgments

The authors would like to thank Mark van Heeswijk and KyungHyun Cho for productive discussions about this work. This work was funded by Aalto MIDE programme (Project UI-ART) and grants Multimodally Grounded Language Technology (254104) and Finnish Center of Excellence in Computational Inference Research COIN (251170) of the Academy of Finland.

Appendix A. The class names of the 40 actions from the HDM05 dataset

(1) cartwheelLHandStart1Reps
(2) clapAboveHead5Reps
(3) depositLowR
(4) elbowToKneeLelbowStart
(5) hitRHandHead
(6) hopBothLegs1hops
(7) hopLLeg1hops
(8) hopRLeg1hops
(9) jogLeftCircle4StepsRstart
(10) JumpingDown
(11) jumpingJack3Reps
(12) kickLFront2Reps
(13) kickLSide1Reps
(14) kickRFront2Reps
(15) kickRSide2Reps
(16) punchLFront2Reps
(17) punchLSide2Reps
(18) punchRFront2Reps
(19) punchRSide2Reps
(20) rotateArmsBothBackward
(21) rotateArmsLBackward
(22) rotateArmsRBackward
(23) sitDownChair
(24) sitDownFloor
(25) sitDownKneelTieShoes
(26) sitDownTable
(27) skier1RepsLstart
(28) squat3Reps
(29) staircaseDown3Rstart
(30) standUpKneelToStand
(31) standUpSitChair
(32) standUpSitFloor
(33) standUpSitTable
(34) throwBasketball
(35) throwSittingHighR
(36) throwStandingLowR
(37) turnLeft
(38) turnRight
(39) walk2SetpsLstart
(40) walkRightCrossFront2Steps

References

[1] S. Mitra, T. Acharya, Gesture recognition: a survey, IEEE Trans. Syst. Man Cybern. Part C 37 (May) (2007) 311–324.
[2] W. Li, Z. Zhang, Z. Liu, Expandable data-driven graphical modeling of human actions based on salient postures, IEEE Trans. Circuits Syst. Video Technol. 18 (11) (2008) 1499–1510.
[3] M. Müller, T. Röder, Motion templates for automatic classification and retrieval of motion capture data, in: Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Computer Animation, Vienna, Austria, 2006, pp. 137–146.

[4] A. Menache, Understanding Motion Capture for Computer Animation and Video Games, Morgan Kaufmann Publishers, San Francisco, California, 2000.
[5] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, A. Blake, Real-time human pose recognition in parts from single depth images, in: Proceedings of the Computer Vision and Pattern Recognition, June 2011.
[6] L. Xia, C.-C. Chen, J.K. Aggarwal, Human detection using depth information by Kinect, in: Workshop on Human Activity Understanding from 3D Data in Conjunction with CVPR (HAU3D), Colorado Springs, USA, 2011.
[7] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1) (2006) 489–501.
[8] M. Raptis, K. Wnuk, S. Soatto, et al., Flexible dictionaries for action classification, in: The First International Workshop on Machine Learning for Vision-based Motion Analysis-MLVMA'08, 2008.
[9] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, R. Bajcsy, Berkeley MHAD: a comprehensive multimodal human action database, in: 2013 IEEE Workshop on Applications of Computer Vision (WACV), pp. 53–60 (January).
[10] A. Vieira, T. Lewiner, W. Schwartz, M. Campos, Distance matrices as invariant features for classifying MoCap data, in: 21st International Conference on Pattern Recognition (ICPR), Tsukuba, Japan, 2012.
[11] X. Zhao, X. Li, C. Pang, S. Wang, Human action recognition based on semi-supervised discriminant analysis with global constraint, Neurocomputing 105 (0) (2013) 45–50.
[12] X. Chen, M. Koskela, Classification of RGB-D and motion capture sequences using extreme learning machine, in: Proceedings of the 18th Scandinavian Conference on Image Analysis, Lecture Notes in Computer Science, Espoo, Finland, Springer-Verlag, vol. 7944, June 2013.
[13] Y. Lin, Efficient motion search in large motion capture databases, Adv. Vis. Comput. (2006) 151–160.
[14] B. Krüger, J. Tautges, A. Weber, A. Zinke, Fast local and global similarity searches in large motion capture databases, in: Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association, 2010, pp. 1–10.
[15] K. Adistambha, C. Ritz, I. Burnett, Motion classification using dynamic time warping, in: IEEE 10th Workshop on Multimedia Signal Processing, 2008.
[16] L. Deng, H. Leung, N. Gu, Y. Yang, Automated recognition of sequential patterns in captured motion streams, in: Web-Age Information Management, Springer, Berlin, Heidelberg 2010, pp. 250–261.
[17] H. Chung, H.-D. Yang, Conditional random field-based gesture recognition with depth information, Opt. Eng. 52 (1) (2013) 017201-1–017201-7.
[18] M. Raptis, D. Kirovski, H. Hoppe, Real-time classification of dance gestures from skeleton animation, in: Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, ACM, Vancouver, Canada, 2011, pp. 147–156.
[19] A. Shimada, R. Taniguchi, Gesture recognition using sparse code of hierarchical SOM, in: Nineteenth International Conference on Pattern Recognition (ICPR), IEEE, 2008, Florida, USA, pp. 1–4.
[20] J.-Y. Wang, H.-M. Lee, Recognition of human actions using motion capture data and support vector machine, in: WRI World Congress on Software Engineering, WCSE'09, vol. 1, IEEE, 2009, Xiamen, China, pp. 234–238.
[21] J. Wang, Z. Liu, Y. Wu, J. Yuan, Mining actionlet ensemble for action recognition with depth cameras, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2012, Providence, RI, USA, pp. 1290–1297.
[22] M. Barnachon, S. Bouakaz, B. Boufama, E. Guillou, A real-time system for motion retrieval and interpretation, Pattern Recogn. Lett. (2013).
[23] K. Lai, J. Konrad, P. Ishwar, A gesture-driven computer interface using kinect, in: Proceedings of IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI), New Mexico, USA, 2012, pp. 185–188.
[24] A. Ramey, V. Gonzalez-Pacheco, M.A. Salichs, Integration of a low-cost rgb-d sensor in a social robot for gesture recognition, in: Sixth ACM/IEEE International Conference on Human–Robot Interaction (HRI), ACM, 2011, Lausanne, Switzerland, pp. 229–230.
[25] R. Minhas, A. Baradarani, S. Seifzadeh, Q. Jonathan Wu, Human action recognition using extreme learning machine based on visual vocabularies, Neurocomputing 73 (10) (2010) 1906–1917.
[26] M. van Heeswijk, Y. Miche, E. Oja, A. Lendasse, GPU-accelerated and parallelized ELM ensembles for large-scale regression, Neurocomputing 74 (16) (2011) 2430–2437.
[27] CMU, Carnegie–Mellon Mocap Database, ⟨http://mocap.cs.cmu.edu⟩, 2003.
[28] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, A. Weber, Documentation Mocap Database HDM05, Technical Report CG-2007-2, University of Bonn, June 2007.
[29] Microsoft, Kinect for Windows SDK, ⟨http://www.microsoft.com/en-us/kinectforwindows/⟩.
[30] OpenNI, Open-source SDK for 3D sensors ⟨http://www.openni.org/⟩.
[31] M. Meredith, S. Maddock, Motion Capture File Formats Explained, Department of Computer Science, University of Sheffield, United Kingdom, 2001.
[32] G. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 42 (2) (2012) 513–529.
[33] A. Vedaldi, A. Zisserman, Efficient additive kernels via explicit feature maps, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2010, San Francisco, CA, United States pp. 3539–3546.
[34] M. Sjöberg, M. Koskela, S. Ishikawa, J. Laaksonen, Large-scale visual concept detection with explicit kernel maps and power mean SVM, in: Proceedings of the ACM International Conference on Multimedia Retrieval (ICMR2013), Dallas, TX, USA, ACM, April 2013, pp. 239–246.
[35] J.A. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (3) (1999) 293–300.
[36] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. (TIST) 2 (3) (2011) 27.
[37] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, LIBLINEAR: a library for large linear classification, J. Mach. Learn. Res. 9 (2008) 1871–1874.
[38] S. Fothergill, H. Mentis, P. Kohli, S. Nowozin, Instructing people for training gestural interactive systems, in: Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems, ACM, 2012, Austin, Texas, USA, pp. 1737–1746.
[39] R. Slyper, J.K. Hodgins, Action capture with accelerometers, in: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association, 2008, Dublin, Ireland, pp. 193–199.
[40] H. Hachiya, M. Sugiyama, N. Ueda, Importance-weighted least-squares probabilistic classifier for covariate shift adaptation with application to human activity recognition, Neurocomputing 80 (2012) 93–101.

Xi Chen is a Ph.D. candidate at the Department of Information and Computer Science, Aalto University School of Science. She received her M.Sc. degree in precision measuring technology and instruments in 2007 from Tianjin University, China. She was the recipient of Erasmus Mundus scholarship and attained double M.Sc. degrees in Space Technology from Luleå University of Technology, Sweden, and in Space Robotics and Automation from Helsinki University of Technology, Finland, in 2009. Her current research interests are in computer vision and machine learning.

Markus Koskela received his M.Sc. degree in engineering physics and mathematics in 1999 and Doctor of Science in Technology in computer science in 2003, both from Helsinki University of Technology, Finland. He is presently a senior researcher at the Department of Information and Computer Science, Aalto University School of Science, Finland, and at the Center of Excellence in Computational Inference Research (COIN) of the Academy of Finland. His research interests are in computer vision, image and video indexing, multimedia understanding, and machine learning. He is the Chairman of the Pattern Recognition Society of Finland, and a member of IEEE, ACM, and IAPR.