Classification of RGB-D and Motion Capture Sequences Using Extreme Learning Machine

Xi Chen and Markus Koskela

Department of Information and Computer Science
Aalto University School of Science
P.O. Box 15400, 00076 Aalto, Finland
{xi.chen,markus.koskela}@aalto.fi

Abstract. In this paper we present a robust motion recognition framework for both motion capture and RGB-D sensor data. We extract four different types of features and apply a temporal difference operation to form the final feature vector for each frame in the motion sequences. The frames are classified with the extreme learning machine, and the final class of an action is obtained by majority voting. We test our framework with both motion capture and Kinect data and compare the results of different features. The experiments show that our approach can accurately classify actions with both sources of data. For 40 actions of motion capture data, we achieve 92.7% classification accuracy with real-time performance.

1 Introduction

Human motion capture (mocap) represents the human skeleton with the 3D positions of each joint or with the angular and translative distances between the joints [7]. Mocap and its classification are important in fields like filmmaking, computer animation, sports science, robotics, surveillance systems and other applications. The recognition of motion provides important information to various intelligent systems, and can also enable human–computer interaction without any physical contact. The methods for the classification of motion capture data are not only beneficial to the mocap community but can also be used for the classification of other media data which generates similar skeleton models during an intermediate process, such as video [4] or depth data [13].

The classification of mocap sequences faces the difficulties of interclass similarity between different actions and intraclass variety of different performance instances of the same actions. For example, a "jump" performed by different actors can have very different styles, and even the same actor does not perform the action exactly the same way each time. Different instance of the same action can also vary e.g. in speed and the number of repetitions of the gestures.

Motion sequence analysis has been revigorated by the proliferation of RGB cameras with depth sensors (RGB-D), such as the Microsoft Kinect, which are nowadays widely used in computer vision due to their low cost and portability. The depth information provided by RGB-D cameras enriches the sensing of the environment, making the analysis of the 3D world considerably easier than with 2D images and e.g. stereo vision approaches. The classification of motion sequences recorded by RGB-D sensors is raising increasing attention in the research community [2]. Several algorithms have been developed for extracting the human skeleton from the RGB-D images [13,16]. These algorithms provide analogous data to mocap, and similar methods for motion classification can thus be applied to both mocap and RGB-D data.

In this paper, we propose a method to classify multiple actions with high accuracy and low computation requirements. We use several different kinds of features, propose the time difference concept to be used in the feature generation, and apply extreme learning machines (ELM) for multiclass classification. We examine the efficiency of the proposed method with a large number of experiments conducted with both motion capture and RGB-D sensor data.

2 Related Work

The methods for classification, segmentation and retrieval of motion capture data are not isolated, e.g. the methods used for classification can often be used in retrieval. In general, the classification methods vary in two main components: feature selection and the used classification algorithm.

Müller et al. [8] proposed relational motion features to describe geometric relations between the key points of a pose. A motion template was learned for each motion class by dynamic time warping. In [15], the 3D coordinates of markers forming the feature vector were used to represent each frame in a motion. Each frame was classified with a SVM, and the recognition result for each motion was the average of the frame-wise results. The algorithm was able to recognize 21 motions with an average accuracy of 92.3%. In [1], the relative rotation angles of bones were used as features. For each bone, a learned feature vector was used as a model vector and each bone's angle sequences were compared to the model vectors of the corresponding bones. The method was used to classify between 4 different motions. In [10], a flexible dictionary of action primitives was built from time series data of limb positions by K-means clustering. Each action was represented as a concatenation of histograms of each primitive and then classified with a SVM. The algorithm was tested by 5 motions with mean classification accuracy of 97.40%. In [12,6], the (x, y, z) positions of seven markers were adopted as features, and the gestures were classified online by sparse coding of a hierarchical Self-Organizing Map. The average accuracy for 6 gestures was 80%. Vieira [14] used distance matrices as features for each posture. The matrix was vectorized and reduced in dimensionality with PCA. Each motion was described by an action graph which consists of salient postures as nodes with each motion modeled by a path in the graph. The average accuracy for 10 motions was 90.86%.

In our paper, instead of extracting complicated features such as in [11], we use either the 3D data directly or calculate Euclidean distances from it and and utilize the information contained in the temporal order to form feature vectors for each frame. These are then classified by ELM, which can directly do multi-class classification and most of the parameters in the model are assigned

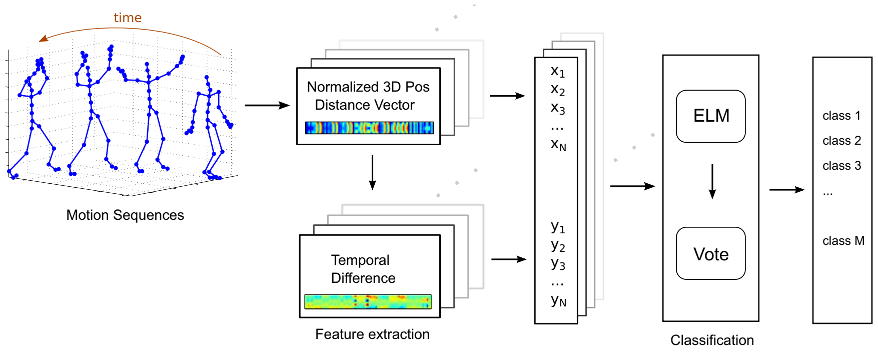


Fig. 1. An overview of the proposed classification framework

randomly which makes the learning extremely fast. The final classification result for a sequence is obtained by majority voting, which makes the system robust to inaccurate and noisy data.

3 Method

In this section, we describe the proposed framework for recognizing multiple actions from motion capture and RGB-D sensor data with high resolution and low computational requirements. We will first describe the feature extraction for the motion sequences and then introduce the extreme learning machine.

Our recognition framework begins with feature extraction. We experiment with several different kinds of features extracted from the skeletal data. Then, a temporal difference vector is calculated on each kind of feature and concentated with the original feature. The selection of which joints from the full skeleton to use directly influences the dimensionality of the features.

Every frame in each motion sequence is classified by the extreme learning machine, and the frame-wise classification results are counted, and the class with the highest number of votes is the the final classification result for that sequence. A graphical overview of the classification framework can be seen in Fig. 1.

3.1 Feature Extraction

Skeletal features can be designed or extracted in various ways. Here, we try to discover features that are simple to calculate and easy to obtain but still contain sufficient distinguishing information for classification. We introduce four kinds of features for skeletal data originating either from mocap or from RGB-D sensors.

Normalized 3D Position (POS). The original data generated by mocap is usually either in ASF/ACM or C3D format. Both formats express skeletons using translations and rotations. The 3D coordinates of the joints can be obtained

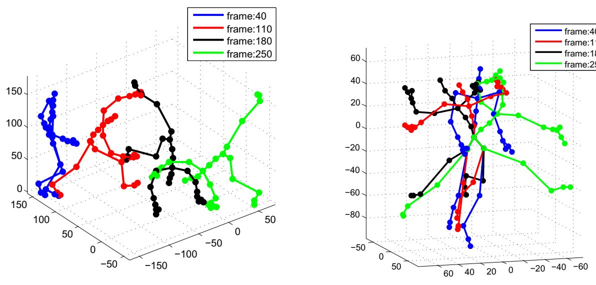


Fig. 2. A "cartwheel" action in original (left) and root (right) coordinates

through several levels of hierachical calculations of rotations and translations. As the first step, a transformation to a common coordinate system is needed.

For the mocap skeleton, the transformed coordinates of joints can be obtained by setting the rotation matrix $\mathbf{R}_1$ and the translation vector $\mathbf{t}_1$ of the root (joint 1 in the HDM05 skeleton in Fig. 4) to identity and zero, i.e. $\mathbf{R}_1 = \mathbf{I}$ and $\mathbf{t}_1 = [0\ 0\ 0]^T$. After this transformation, the root of the skeleton is at origin, and the left and right hip are at positions relative to the root at coordinates fixed during calibration, which are independent to the performed posture. That is, for all postures of a given actor, the coordinates of the root, left hip and right hip remain fixed after the transformation. For example, Fig 2 shows sampled frames from a "cartwheel" action in the original and root coordinates.

For RGB-D skeletons, only the 3D coordinates of the joints are typically available, i.e. no similar hierarchical format of bone structure and rotation and translation data exist. Therefore, we have to transform the joints to a common coordinate system using the original 3D positions of the joints. We start by translating the root of the skeleton to the origin by subtracting the coordinates of the root from all the joints.

Next, we select any skeleton as the standard skeleton, and use its coordinates as the common basis that the other skeletons in the sequence will be transformed to. Let us assume that after the translation to the origin, the left and 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 $\mathbf{R}$. $\mathbf{p}_{lhip}$ and $\mathbf{p}_{rhip}$ form a plane c that can be represented by the origin of the coordinates and the norm vector $\mathbf{n}_c$, which can be calculated by

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

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

$$\begin{gathered} \mathbf{n}_c \cdot (\mathbf{R}\mathbf{p}_{lhip}) = 0 \\ \mathbf{n}_c \cdot (\mathbf{R}\mathbf{p}_{rhip}) = 0 \\ \min_{\mathbf{R}}(\|\mathbf{R}\mathbf{p}_{lhip} - \mathbf{p}^0_{lhip}\| + \|\mathbf{R}\mathbf{p}_{rhip} - \mathbf{p}^0_{rhip}\|) \end{gathered} \tag{2}$$

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

Finally, due to the varying size of actors, the skeleton should be normalized so that the sum of the distances of the connected joints equals to one.

Pairwise Distance Vector (PW). As the name implies, the pairwise distance vector feature is composed of all distances between the joints, and the vector is normalized so that the sum of all elements equals to one. The elements of the feature vector can be calculated by

$$\frac{\|\mathbf{p}_i - \mathbf{p}_j\|}{\sum_{i \neq j} \|\mathbf{p}_i - \mathbf{p}_j\|}\,, \quad i \neq j \tag{3}$$

where $\mathbf{p}_i$ and $\mathbf{p}_j$ are the 3D positions of joints i and j.

Centroid Distance Vector (CEN). In this feature, we consider the centroid of the triangle formed by the neck and hips as the centroid of the body. The elements of the feature vector consist of the distances between joints and the centroid, normalized by the sum of the distances,

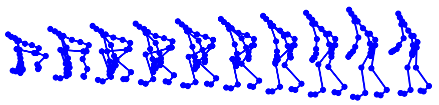
$$\frac{\|\mathbf{p}_i - \mathbf{p}_{cen}\|}{\sum_i \|\mathbf{p}_i - \mathbf{p}_{cen}\|}\,, \quad \forall i \tag{4}$$

where $\mathbf{p}_{cen} = \frac{1}{3}(\mathbf{p}_{neck} + \mathbf{p}_{lhip} + \mathbf{p}_{rhip})$.

Key Joints Distance Vector (KEY). Similar to the centroid distance vector, the key joints distance vector is calculated as the distances between a set of key joints and the other joints. We use in this paper the following three key joints: *head*, *root* and left knee (*lknee*);

$$\begin{gathered}\left\{\frac{\|\mathbf{p}_i - \mathbf{p}_{head}\|}{\sum_i \|\mathbf{p}_i - \mathbf{p}_{head}\|}\,, \quad \frac{\|\mathbf{p}_j - \mathbf{p}_{root}\|}{\sum_j \|\mathbf{p}_j - \mathbf{p}_{root}\|}\,, \quad \frac{\|\mathbf{p}_k - \mathbf{p}_{lknee}\|}{\sum_k \|\mathbf{p}_k - \mathbf{p}_{lknee}\|}\right\} \\ \forall i,j,k: \quad i \notin \{head\}, \quad j \notin \{head,\ root\}, \quad k \notin \{head,\ root,\ lknee\}\ . \end{gathered} \tag{5}$$

Temporal Difference of Feature Vectors (TDFV). The above feature vectors represent the uniqueness of each posture. In general, actions are composed of multiple postures, and some actions can even be kinematically inverse to each other. Fig. 3 shows two such actions: *StandUpKnee* and *SitDownKnee*. Here, for each posture in one action there is an almost identical counterpart in the other one, and a frame-wise classification method can easily misclassify such frames. Therefore, we take the temporal order of the postures into account by calculating a feature expressing temporal difference information.


(a) *StandUpKnee*

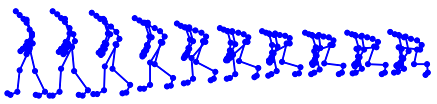

(b) *SitDownKnee*

Fig. 3. One pair of inverse actions

Let us assume the feature of the ith posture calculated by one of the above methods is $\mathbf{f}_d^i$. The *temporal difference of feature vectors* feature $\mathbf{f}_{td}^i$ can be calculated as

$$\mathbf{f}_{td}^i = \begin{cases} \mathbf{f}_d^i & 1 \leq i < m \\ \mathbf{f}_d^i - \mathbf{f}_d^{i-m+1} & m \leq i \leq N \end{cases} \tag{6}$$

where m is the temporal offset parameter, $1 < m < N$. By concatenating the feature vector $\mathbf{f}_d$ and the temporal distance feature vector $\mathbf{f}_{td}$, the final feature of a posture can be written as $\mathbf{f}^i = [(\mathbf{f}_d^i)^T \ (\mathbf{f}_{td}^i)^T]^T$.

Dimensionality of the Features. The dimensionalities of the feature vectors $\mathbf{f}_d$ described in this section are

• normalized 3D position:	$3n$
• pairwise distance:	$n(n-1)/2$
• centroid distance:	n
• key joints distance:	$n_{kj}n - \sum_{i=0}^{n_o} i$

where the numbers of joints in the used set and of key joints in the skeleton are n and n_{kj}, respectively, and n_o is the number of overlapping joints between key joints and the set of used joints. $\mathbf{f}_{td}$ has the same dimensionality as the original feature vector, so the dimensionality of the final feature vector $\mathbf{f}$ is doubled.

The commonly used mocap databases include the CMU Graphics Lab Motion Capture Database [3] and the Motion Capture Database HDM05 [9]. Both databases provide 31 joints for their skeletons with the same ordering, whereas e.g. the skeletons extracted from Kinect with the PrimeSense Natural Interaction Middleware (NiTE) have 15 joints. According to [3] some joints are not real captured data, and some are relatively noisy. For these reasons, and also to be able to perform comparisons between mocap and Kinect data, we simplify the mocap skeletons from 31 to 15 joints corresponding to the NiTE skeleton format shown in Fig. 4.

3.2 Extreme Learning Machine

Extreme learning machine (ELM) [5] 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.

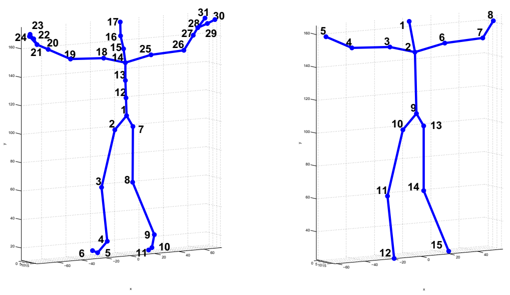


Fig. 4. Original skeleton of HDM05 (left) and the simplified skeleton (right)

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

$$\mathbf{y}_i = f(\mathbf{x}_i) = \sum_{j=1}^{L} \boldsymbol{\gamma}_j g(\boldsymbol{\omega}_j \cdot \mathbf{x}_i + b_j) \ , \tag{7}$$

where $g(\cdot)$ is a nonlinear activation function, $\boldsymbol{\gamma}_j \in \mathbb{R}^m$ are the output weights, $\boldsymbol{\omega}_j \in \mathbb{R}^n$ is the weight vector connecting the input layer to the ith hidden neuron and b_j is the bias of the ith hidden neuron. Both $\boldsymbol{\omega}_j$ and b_j are assigned randomly during the learning process. With $\mathbf{Y} = [\mathbf{y}_1 \ \mathbf{y}_2 \cdots \mathbf{y}_P]^T \in \mathbb{R}^{P \times m}$ and $\boldsymbol{\Gamma} = [\boldsymbol{\gamma}_1 \ \boldsymbol{\gamma}_2 \cdots \boldsymbol{\gamma}_L]^T \in \mathbb{R}^{L \times m}$, Eq. (7) can be written compactly as

$$\mathbf{H}\boldsymbol{\Gamma} = \mathbf{Y} \tag{8}$$

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

$$\mathbf{H} = \begin{bmatrix} g(\boldsymbol{\omega}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\boldsymbol{\omega}_L \cdot \mathbf{x}_1 + b_L) \\ \vdots & \ddots & \vdots \\ g(\boldsymbol{\omega}_1 \cdot \mathbf{x}_P + b_1) & \cdots & g(\boldsymbol{\omega}_L \cdot \mathbf{x}_P + b_L) \end{bmatrix}_{P \times L} . \tag{9}$$

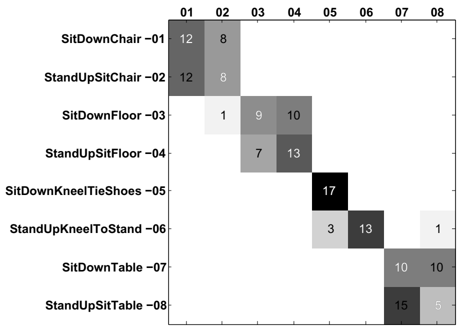
The solution of Eq. (8) is given [5] as

$$\boldsymbol{\Gamma} = \mathbf{H}^{\dagger}\mathbf{T} \tag{10}$$

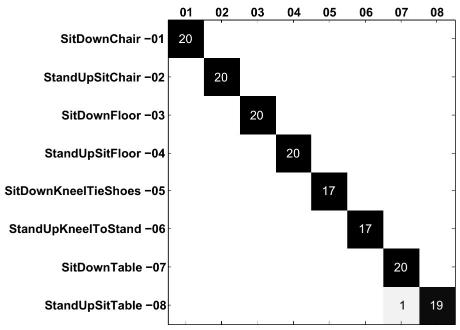
where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of matrix $\mathbf{H}$.

4 Experiments

In this section, we provide an experimental evaluation of our proposed method. To be able to make comparisons to other published methods, we use the public mocap database HDM05 [9], which contains roughly 70 motion classes with 10 to 50 motion sequences for each class performed by 5 actors. We also record additional data with the Kinect sensor that corresponds to a subset of HDM05. In all experiments, we use the ELM with $L = 500$ hidden neurons.



(a) Using only distance features $\mathbf{f}_d$



(b) Using concatenated features $\mathbf{f}$

Fig. 5. Confusion matrix for four pairs of inverse actions

4.1 Effect of TDFV Feature

In order to test the effect of the TDFV feature, we first select four pairs of actions from HDM05 database. In each pair, the actions are similar but in reverse chronological order. All motion sequences of these actions are used in the experiments, totaling 154 sequences. We use the pairwise distance of all joints and either $\mathbf{f}_d$ or $\mathbf{f}$ as our feature. In both cases, we use 10-fold cross validation.

The confusion matrices for the experiments are shown in Fig. 5. We can see that with $\mathbf{f}_d$, i.e. only the frame-wise distance vectors, the actions in each pair are often misclassified to each other, whereas with $\mathbf{f}$, that is, after concatenating with the TDFV feature, the actions are clearly distinguished from each other. The average accuracy is increased from 56.49% to 99.35%. Based on these results, we use the concatenated features in all subsequent experiments in this paper.

4.2 Temporal Parameter in TDFV

In Eq. (6), the temporal parameter m represents the offset in frames in the motion sequence. To observe the influence of different offset values, we calculate TDFV features for different values of m corresponding to time offsets ranging from 0.1s to 0.8s with a 0.1s interval. with the same motion sequences as in Sec. 4.1. The average classification accuracies are plotted in Fig. 6. We can observe that the accuracy is above 96% in all cases, and when the time offset is 0.3s, we get the highest classification accuracy. Therefore in all further experiments we use values $m = 0.3f_s$, where f_s is the frame rate of the sequence.

4.3 Comparing Features with Mocap and Kinect Skeletons

In [14], the distance matrix is used as features and an action graph is used to classify between 10 classes of motions from the HDM05 database. In order to be able to compare results, we use the same data to test our method. In total, 156 motion sequences from 10 classes are used in 10-fold cross validation. We

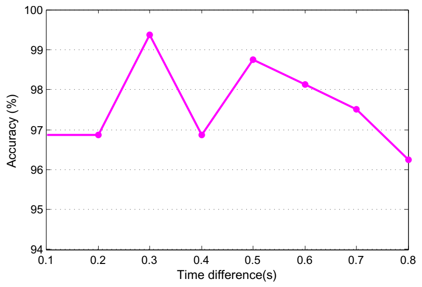


Fig. 6. Classification accuracy vs. time difference in TDFV

select two sets of joints: all 15 joints in the simplified skeleton and a reduced set of five joints (head, hands and feet). The four features described in Sec. 3.1 are extracted for both sets of joints.

We also recorded the same motions with a Kinect device. Example RGB-D images and the corresponding skeleton are shown in Fig. 7. In our data, there are five actors similarly as in the mocap data, and each of the actors performed an equal number of motions as in the original data[1].

The classification accuracies are shown in Table 1. The 10 used actions are indicated by numbers and the corresponding actions can be found in Table 2. The dimensionality of the features $\mathbf{f}_d$ is shown in parentheses. We can see that, for the mocap skeletons, ELM with each of the features and either set of joints (15 or 5 joints) can classify the 10 classes with a minimum of 98.8% average accuracy, which is higher than reported in [14]. For the same motions with Kinect the accuracy is lower, which can be explaned by the unstability and noisiness of the Kinect skeletons. An example is shown in Fig. 7 where the right knee of the skeleton is not bent as in the corresponding RGB-D image. With five joints, the dimensionalities of the features are significantly lower than with 15 joints, but the classification accuracies remain almost as high. For Kinect, the normalized 3D position feature requires a relatively complicated normalization computation which is not needed with the other three features, and the accuracy is only about 1–4% higher than with the pairwise and key joints distances. Therefore, the pairwise and key joints distance are also a good choices with Kinect skeletons.

4.4 Classification of 40 Actions with Mocap Data

In Sec. 4.3, all four features were shown to provide very high accuracies in classification of 10 classes with mocap data. Therefore, we now test the performance of the features on a larger dataset. In HDM05, the actions can be categorized into two groups: stationary and mobile actions. Our features extract the relative joint relations without the absolute movement. In this experiment we therefore use the stationary actions and a small number of mobile actions such as *jog* and *walk*, which contain different relative joint relations to the stationary actions. In

[1] The used data is available at `http://research.ics.aalto.fi/cbir/data/`

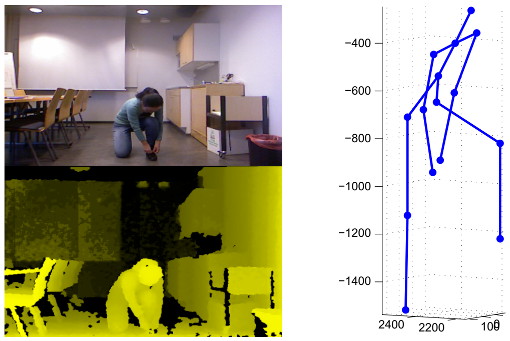


Fig. 7. RGB and depth images (left), and the corresponding skeleton (right)

total, we use 40 actions and 790 sequences. We use the four features with the set of five joints (head, hands, and feet), again with 10-fold cross validation.

The results are shown in Table 2 where N_s is the number of motion sequences in each class, and N_f is the total number of frames for that class. We can observe that the classification accuracy with the centroid distance vector drops significantly, while the normalized 3D position remains above 92% accuracy. It is slightly better than the key joints distance vector, which is then 4% better than the pairwise distance vector. The accuracies correspond to feature dimensionalities, in that the normalized 3D position has 15 dimensions, the key joints distance vector 14, the pairwise distance vector 10, while the centroid distance vector has only 5 dimensions. The experiment shows that the normalized 3D position and the key joints distance vector with the TDFV features are informative features and scale well into a large dataset.

The experiments were conducted using Intel Core i5 CPU at 3.3 GHz and 8 GB of memory. The four features require almost an equal amount of time both in training and in testing. For training with about 200 000 frames, the ELM algorithm took on average around 22 seconds. In testing, including the data preprocessing and the majority voting, it took on average 0.04 seconds to classify one sequence. The average number of frames in a sequence was 265. The experiments were performed using non-optimized Matlab code.

5 Conclusions and Future Work

In this paper, we use four features—normalized 3D position, pairwise distance, centroid distance, and key joint distance—and propose the concept of temporal difference of feature vectors. Concatenated together, these two form a feature vector for each frame in a motion sequence. We then apply extreme learning machines to classify each frame, and the action class with the highest number of votes is selected as the class of the motion sequence. We test our framework on both Kinect and mocap data.

The experiments show that with only the head, hands and feet, the classification can be almost as accurate as with the whole 15-joint skeleton. We compare

Table 1. Classification of 10 action classes with both Kinect and mocap data

Ac-tion	Kinect (%)								Mocap (%)								[14] (%)
	15 joints				5 joints				15 joints				5 joints				
	pos (45)	pw (105)	cen (15)	key (39)	pos (15)	pw (10)	cen (5)	key (14)	pos (45)	pw (105)	cen (15)	key (39)	pos (15)	pw (10)	cen (5)	key (14)	
1	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	85.7
2	100	95.0	90.0	95.0	100	95.0	75.0	90	100	100	100	100	100	95.0	100	100	87.9
3	100	95.0	70.0	95.0	100	95.0	75	90	100	100	100	100	100	100	95.0	100	100
4	100	100	100	100	100	100	100	100	100	100	93.3	93.3	100	100	100	100	90.0
5	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	88.1
6	92.3	84.6	76.9	84.6	84.6	92.3	76.9	76.9	100	100	100	100	100	100	100	100	88.9
7	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	94.1
8	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	89.1
9	75.0	87.5	56.3	75.0	81.3	50.0	50.0	68.8	100	100	94.1	100	94.1	100	100	100	93.3
10	93.8	87.5	37.5	75.0	81.3	87.5	56.3	81.3	100	100	100	100	100	100	100	100	91.4
Av.	96.1	95.0	83.1	92.5	94.7	92.0	83.3	90.7	100	100	98.8	99.3	99.4	99.5	99.5	100	90.9

Table 2. Classification of 40 action classes with mocap data

Action	N_s	N_f	pos	pw	cen	key
cartwheelLHandS	21	8627	100	100	100	100
clapAboveHead(1)	14	6102	100	100	100	100
depositLowR	28	7767	100	75.0	100	100
elbowToKnLeS (7)	13	5756	100	100	100	100
hitRHandHead	13	2943	84.6	92.3	7.69	92.3
hopBothLegs	36	3462	61.1	91.7	41.7	91.7
hopLLeg	41	3080	100	100	95.1	100
hopRLeg	42	3107	100	100	100	100
jogLeftCircleRS	17	4142	100	94.1	100	100
JumpingDown	14	3952	92.9	7.14	76.5	92.9
jumpingJack (6)	13	5589	100	100	0	100
kickLFront (5)	14	6422	78.6	78.6	0	78.6
kickLSide	26	6063	76.9	88.5	92.9	88.5
kickRFront	15	6728	100	86.7	53.9	86.7
kickRSide	15	7020	93.3	100	80.0	66.7
punchLFront	15	5924	80.0	73.3	67.7	86.7
punchLSide	15	5324	86.7	66.7	53.3	26.7
punchRFront (4)	15	6450	93.3	86.7	60.0	73.3
punchRSide	14	5140	85.7	85.7	28.6	78.6
rotateArmsBBack	16	5111	100	100	100	100
rotateArmsLBack	16	1725	93.8	93.8	43.8	100
rotateArmsRBack	16	1685	100	56.3	43.8	100
sitDownChair (2)	20	6377	90.0	70.0	100	90
sitDownFloor (3)	20	8154	95.0	100	80.0	100
sitDownKnTS(10)	17	10978	100	100	100	100
sitDownTable	20	5411	85.0	60.0	35.0	85.0
skierLstart	30	4240	100	100	90.0	100
squat (8)	13	7619	100	100	100	100
staircaseDownRS	15	3338	100	100	86.7	100
standUpKnTS (9)	17	3094	100	100	82.4	100
standUpSitChair	20	5919	90.0	85.0	100	100
standUpSitFloor	20	8060	90.0	100	95.0	95.0
standUpSitTable	20	5000	85.0	65.0	30.0	70.0
throwBasketball	14	5710	78.6	92.9	0	78.6
throwSitHighR	14	4192	100	100	78.6	100
throwStandingLR	14	4957	100	85.7	0	100
turnLeft	30	5882	76.7	43.3	40.0	80.0
turnRight	30	5908	93.3	86.7	70.0	86.7
walkLstart	31	4818	96.8	93.6	83.9	96.8
walkRightCrossF	16	5369	100	100	93.8	100
Average			92.7	86.5	66.9	91.1

the classification of the same actions with mocap and Kinect skeletons with 10 action classes. With mocap data, we get almost 100% accuracy in each case, and even though the Kinect skeleton is much more unstable and inaccurate, we still get accuracies reaching 96%. With 40 action classes and mocap data, our method achieves an accuracy of 92.7% with a 30-dimensional feature vector. In all cases, the low time requirements for both feature extraction and classification make online action classification possible.

The proposed features describe the relative positions between joints and do not contain the absolute traces of the movement. Therefore, they are only effective in distinguishing between stationary actions that mainly express relative body movement. The are not well-suited for distinguishing mobile actions, e.g. between running in place and running forward. In the future, we will extend the features in order to be able to classify also motions with absolute movement.

Acknowledgments. The authors would like to thank the actors for helping to collect the Kinect data used in the experiments. 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.

References

1. Adistambha, K., Ritz, C., Burnett, I.: Motion classification using dynamic time warping. In: IEEE 10th Workshop on Multimedia Signal Processing (2008)
2. CHALEARN: Gesture challenge, `http://www.kaggle.com/c/GestureChallenge`
3. CMU: Carnegie-mellon mocap database (2003), `http://mocap.cs.cmu.edu`
4. Ferrari, V., Marin-Jimenez, M., Zisserman, A.: Progressive search space reduction for human pose estimation. In: Proc. CVPR (June 2008)
5. Huang, G., Zhou, H., Ding, X., Zhang, R.: Extreme learning machine for regression and multiclass classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 42(2), 513–529 (2012)
6. Kawashima, M., Shimada, A., Taniguchi, R.-I.: Early recognition of gesture patterns using sparse code of self-organizing map. In: Príncipe, J.C., Miikkulainen, R. (eds.) WSOM 2009. LNCS, vol. 5629, pp. 116–123. Springer, Heidelberg (2009)
7. Menache, A.: Understanding motion capture for computer animation and video games. Morgan Kaufmann Pub. (2000)
8. Müller, M., Roder, T.: Motion templates for automatic classification and retrieval of motion capture data. In: Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Computer Animation, Vienna, Austria, vol. 2, pp. 137–146 (2006)
9. Müller, M., Röder, T., Clausen, M., Eberhardt, B., Krüger, B., Weber, A.: Documentation mocap database HDM05. Tech. Rep. CG-2007-2, U. Bonn (June 2007)
10. Raptis, M., Wnuk, K., Soatto, S., et al.: Flexible dictionaries for action classification. In: Proc. MLVMA 2008 (2008)
11. Raptis, M., Kirovski, D., Hoppe, H.: Real-time classification of dance gestures from skeleton animation. In: Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 147–156. ACM (2011)
12. Shimada, A., Taniguchi, R.: Gesture recognition using sparse code of hierarchical som. In: Proc. ICPR (2008)
13. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from single depth images. In: Proc. Computer Vision and Pattern Recognition (June 2011)
14. Vieira, A., Lewiner, T., Schwartz, W., Campos, M.: Distance matrices as invariant features for classifying MoCap data. In: 21st International Conference on Pattern Recognition (ICPR), Tsukuba, Japan (2012)
15. Wang, J., Lee, H.: Recognition of human actions using motion capture data and support vector machine. In: Proc. WCSE, vol. 1, pp. 234–238. IEEE (2009)
16. Xia, L., Chen, C.C., Aggarwal, J.K.: 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)