Learning features combination for human action recognition from skeleton sequences

Diogo Carbonera Luvizon, Hedi Tabia, David Picard

Learning features combination for human action recognition from skeleton sequences

Diogo Carbonera Luvizon * Hedi Tabia David Picard
ETIS – UMR CNRS 8051 – ENSEA – Université Paris Seine / Université de Cergy-Pontoise
{diogo.luvizon, hedi.tabia, picard}@ensea.fr

Abstract

Human action recognition is a challenging task due to the complexity of human movements and to the variety among the same actions performed by distinct subjects. Recent technologies provide the skeletal representation of human body extracted in real time from depth maps, which is a high discriminant information for efficient action recognition. In this context, we present a new framework for human action recognition from skeleton sequences. We propose extracting sets of spatial and temporal local features from subgroups of joints, which are aggregated by a robust method based on the VLAD algorithm and a pool of clusters. Several feature vectors are then combined by a metric learning method inspired by the LMNN algorithm with the objective to improve the classification accuracy using the nonparametric k-NN classifier. We evaluated our method on three public datasets, including the MSR-Action3D, the UTKinect-Action3D, and the Florence 3D Actions dataset. As a result, the proposed framework performance overcomes the methods in the state of the art on all the experiments.

1 Introduction

Despite many efforts in the last years, automatic recognition of human actions is still a challenging task. Action recognition is broadly related to human behavior and to machine learning techniques, whereas its applications include improved human-computer interfaces, human-robot interaction and surveillance systems, among others. Activities involving temporal interactions between humans and objects could be handled by graphical models as described in [5]. In our work, we address specifically human actions, which are described as a well defined sequence of movements. Moreover, action recognition methods may be used as intermediate stages in systems capable of providing more complex interpretations such as human behaviour analysis and task recognition [13]. In action recognition frameworks, we can identify three major parts: action segmentation from video streams, modeling and representation of spatial and temporal structure of actions, and action classification. The first two parts are highly dependent on the quality of sensory data, while the classification stage has been proved difficult due to the variety and complexity of human body movements.

Many approaches for human action recognition are based on 2D video streams [26]. However, 2D color images are hard to be segmented and lack depth information. As an alternative to color cameras, depth sensors have been popularized by their low-cost and accessibility. Examples of affordable depth sensors are Microsoft's Kinect and Asus' Xtion, which allows to capture both RGB images and depth maps. The human poses, composed by skeleton joints, can be extracted from depth maps in real-time [19]. Skeleton joints are a high discriminant representation that allows efficient extraction of relevant information for action classification. Samples of RGB images with associated depth maps and skeleton joints from captured human actions are shown in Fig. 1.

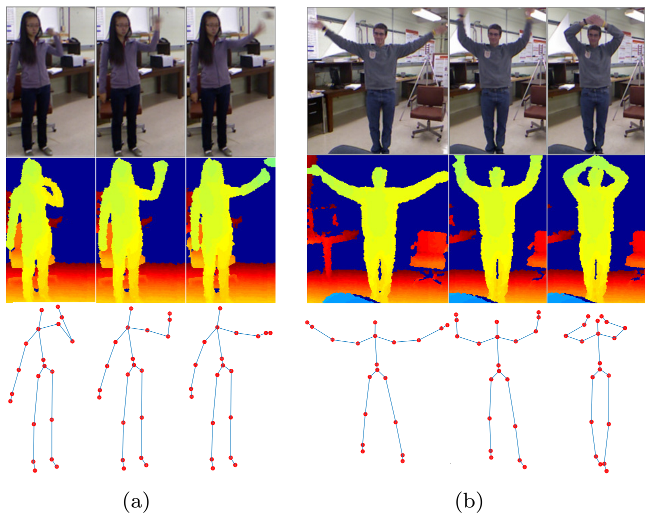

Figure 1: Samples of RGB images, depth maps, and their respective skeleton joints from the public UTKinect-Action3D dataset [27]. The images correspond to the actions *throw* (a) and *wave hands* (b).

Human action recognition methods still include some drawbacks, specially when representing the structure of actions. Many authors have proposed to extract spatial features from skeleton joints [23, 27], while others extract temporal information from sequences alignment [12] or by frequency analysis [24] of spatial features. It is known

*Corresponding author: Tel.: +33-013-073-6292; fax: +33-013-073-6627;

that both spatial and temporal information are fundamental, however, an early combination of distinct features may not effectively improve results [21]. Some authors have noticed that the relevance of each skeleton joint varies from one action to another. [24] empirically demonstrated the benefit of grouping certain joints into subgroups to construct more discriminant features. Also, human actions are often performed in very different durations, which demands a robust method to represent the variety of lengths of the input sequences.

actionlet

Considering the presented difficulties, we present the three main contributions of this work as follows. First, we bring traditional methods in the image classification domain to human action recognition, constructing a new framework able to combine distinct features in a straightforward pipeline. Second, we propose simple yet efficient spatial and temporal local features from subgroups of joints, aggregated into global representations, allowing further learning to extract relevant information for classification. Third, we demonstrate the individual contributions of each step of the proposed framework by extensive experiments, showing that all parts are important to the final results. With these contributions we are able to provide state-of-the-art performance at very high speed on three well know datasets. In addition, the source code of this work is publicly available [1].

This paper is organized as follows. Section 2 presents a review of related work. The proposed framework is described in section 3, followed by the experimental evaluation of our method in section 4. Finally, section 5 concludes this paper.

2 Related work

In this section, we focus on action recognition methods from depth sensors. Readers can refer to the survey from [15] for general vision-based approaches.

State-of-the-art methods for human action recognition using depth sensors are mostly based on depth maps, skeleton joints, or both [10]. Some of the advantages of depth sensors over color cameras are their invariance to lightning and color conditions. They can also be used to provide a 3D structure of the scene, which makes the segmentation step easier. Some methods use solely depth maps for action recognition [9, 11, 14, 16, 28]. However, these methods suffer from noisy depth maps and occlusions. To deal with multichannel RGB-D frames, [21] proposed a latter feature combination scheme, although it can be costly if several features are used, since their method needs one classifier and one weight coefficient for features individually. Using both depth maps and skeleton joints, [24] proposed the actionlet ensemble model. The actionlet features combine the relative 3D position of subgroups of skeleton joints and the local occupancy pattern (LOP) descriptor. To capture the temporal structure of actions, the authors employ the short time Fourier transform on concatenated features to compose a final feature vector.

[1]The Matlab® source code is publicly available at `https://github.com/dluvizon/harskel`

Based only on skeletons extracted from depth maps, [27] proposed a representation of human pose by the histogram of 3D joints (HOJ3D). They project the sequence of histograms using LDA and label each posture using the k-means algorithm. Each posture label from a sequence is fed into a discrete hidden Markov model (HMM) that gives the matching probability for each action. Their approach showed low accuracy in cross subject tests due to the high intra-class variance observed in the evaluated datasets. [29] showed that speed and acceleration of skeleton joints are also important for action classification by proposing the nonparametric moving pose (MP) descriptor. A dictionary learning algorithm was proposed by [12] in order to learn a sparse representation for the human body. Some works have proposed manifold based methods to analyse sequences of skeleton joints. [4] projected the temporal evolution of skeleton joints into a Riemannian manifold and proposed an elastic metric to compare different actions, and [20] projected one sequence as a point on a Grassmann manifold, proposing to learn the "Control Tangent" spaces for actions representation. [23] proposed a new skeletal representation that models the 3D geometric relationships between human joints using three-dimensional linear transformations. Applying a graph-based model for human action recognition, [8] proposed the Joint Spatial Graph (JSG), in which the vertices represent the human joints and the edges represent the relative variance among joint pairs with respect to a specific action. For each action, a JSG is constructed using a fixed number of joint pairs according to their importance. Two graphs are then compared by their structures, resulting in a similarity score.

Although recurrent neural networks (RNN) have succeed in the text and speech domains, a few works have presented satisfactory results using the already available skeletal features in differential RNN [22] and Long Short-Term Memory (LSTM) networks [31]. One of the major difficulties using neural networks on 3D action recognition is the relatively small number of training samples, which usually leads to strong overfitting.

From this related work we can conclude two important facts. First, both spatial and temporal information are relevant for action recognition. However, since they have different nature, it is not trivial to combine them. Second, certain joints are more discriminant for specific actions. In our work, we demonstrate that spatial and temporal features can be used together if they are combined correctly. Furthermore, the proposed approach relies on an aggregation method that preserves fundamental information from individual joints, allowing further efficient metric learning.

3 Proposed framework

The proposed approach for human action recognition rely on sequences of skeleton joints extracted from depth maps

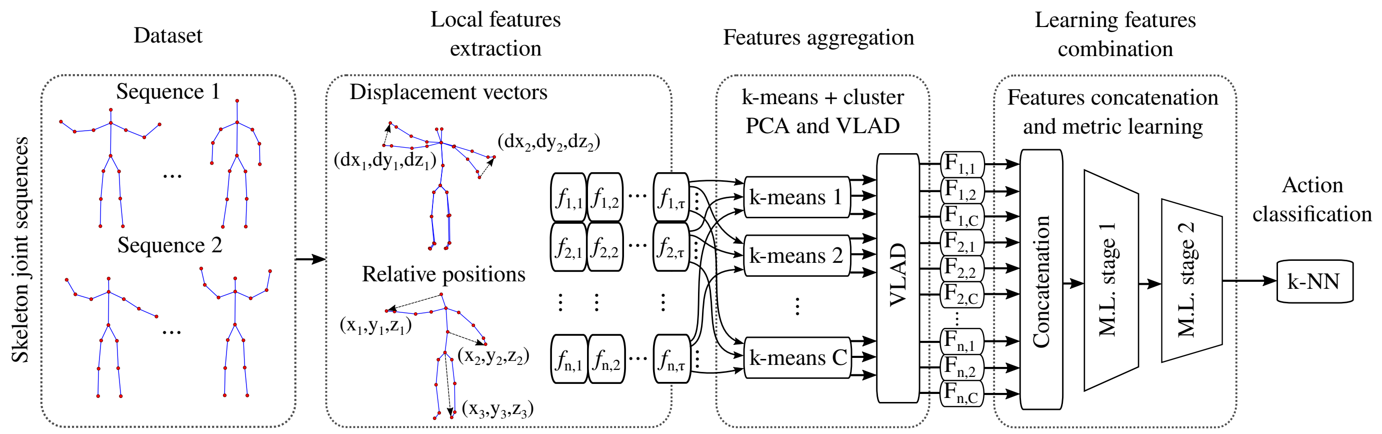

Figure 2: Overview of the proposed framework.

and can be divided into four stages, as outlined in Fig. 2. In the first stage we extract local features, which are then aggregated into global representations. In the third stage, all the resulting features are concatenated. Finally, the learning method extract the relevant information for the k-NN classification.

3.1 Local features extraction

The local features are extracted directly from sequences of skeleton joints and can be divided into two types, according to their physical interpretation. The first are the *displacement vectors* of joints, which represent the motion of specific body parts. Displacement vectors are 3D vectors taken from single joints with respect to the sequence of skeletons $s = \{1, 2, \dots, \tau\}$, defined as follows:

$$\upsilon_i^s = \frac{p_i^{s+1} - p_i^{s-1}}{\Delta T} \mid 1 < s < \tau, \qquad (1)$$

where p_i^s is the coordinate (x, y, z) of the joint i in the
time
sequence index s, ΔT is the time interval between two sequences $s+1$ and $s-1$, and τ is the number of skeletons (frames) in a given sequence. The second type of local features are formed by *relative position* of joints, which is a relevant information that describes the body position and has been successfully used by other authors [24, 12]. The relative position between two joints in the sequence index s is a 3D vector defined by the equation bellow:

$$\omega_{i,k}^s = p_i^s - p_k^s \mid i \neq k, \qquad (2)$$

where p_i^s and p_k^s are the coordinates (x, y, z) of different joints from the same skeleton.

The skeletal representation of human body is usually composed by a fixed number of joints. Two different layouts of human body representation are shown in Fig. 3. The basic layout is composed by 15 joints: *right hand, r. elbow, r. shoulder, head, neck, left shoulder, l. elbow, l. hand, spine, r. hip, r. knee, r. foot, l. hip, l. knee*, and *l. foot*. Another representation commonly used in some datasets includes the *r. wrist, l. wrist, center hip, r. ankle*,

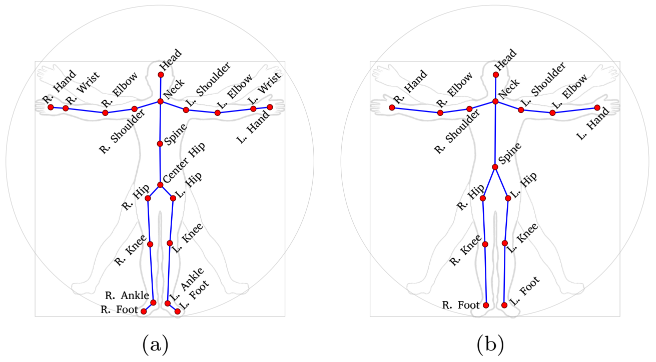

Figure 3: Human body representation by 20 (a) and 15 (b) skeleton joints.

Table 1: Features f_n composed by subgroups of displacement vectors.

Feature	Subgroup of joints (i)
f_1	*Head, r. hand, l. hand, r. foot,* and *l. foot*
f_2	*Neck, r. elbow, l. elbow, r. knee,* and *l. knee*
f_3	*Spine, r. shoulder, l. shoulder, r. hip,* and *l. hip*

and *l. ankle*, resulting in 20 joints. In order to build the proposed local features, we concatenate displacement vectors (from Eq. 1) or relative positions (from Eq. 2) taking subgroups of joints. Three features are composed by combination of displacement vectors and four are composed by combination of relative positions, respectively detailed by Tables 1 and 2. Specifically, the features f_1, f_2, and f_3 are composed by concatenation of displacement vectors of five joints. Similarly, the features f_4, f_5, f_6, and f_7 result from distinct relative positions concatenated.

The major objective of dividing skeletons into subgroups of joints is to provide smaller features to the clustering stage. When compared to all joints composing one single feature, we expect two improvements. First, smaller features tend to be better clustered, and second, it is preferable to use smaller but complementary groups of joints than reducing the feature space (by PCA, for example).

对关节点分组的原因

Table 2: Features f_n composed by subgroups of relative positions.

Feature	Subgroup of joints (i)	Relative to (k)
f_4	*Head, l. hand,* and *r. hand*	*Spine*
f_5	*Head, l, hand,* and *l. foot*	*R. hip*
f_6	*Head, r. hand,* and *r. foot*	*L. hip*
f_7	*L. hand* and *r. hand*	*Head*

This assumption is verified by our experiments, as presented in Sec. 4. Another important point is how the subgroups are chosen. In our method it is not practical to evaluate all the possible combinations of different joints into smaller groups. Therefore, we intuitively divided the joints of displacement vectors from the center to the extremities of the human body. Similarly, the relative positions were selected to represent the position of hands, feet and head with respect to the rest of the body. This approach was empirically validated as a satisfactory solution by our experiments.

3.2 Features aggregation

For each video frame, a set of local features are extracted by the local feature extraction method. Namely, the sequence of local features is represented by $f_{n,s}$ where $n = \{1, 2, \ldots, 7\}$ and $s = \{1, 2, \ldots, \tau\}$. The objective of the aggregation stage is to build fixed-size features for each sequence of local features, as depicted in Fig. 4.

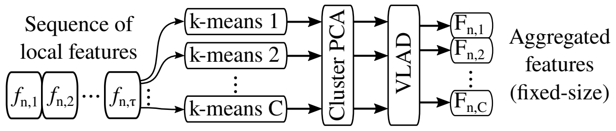

Figure 4: Diagram of the feature aggregation stage.

三步

The feature aggregation method can be divided into three steps. In the first step, for each subgroup n we compute C different k-means using different initializations. For each subgroup we then obtain C sets of k clusters represented by $\{\mu_{n,c,m}\}$, where $m = \{1, \ldots, k\}$ and $c = \{1, \ldots, C\}$. The next step consists in applying PCA to local features individually in each cluster. In this step, we apply PCA keeping all the components. Finally, in the third step, we use the vector of locally aggregated descriptors (VLAD), which is the non probabilistic version of the Fisher Vectors, proposed by [6]. Let

$$S_{n,c,m} = \left\{ f_{n,s} \middle| m = \arg\min_p \|f_{n,s} - \mu_{n,c,p}\| \right\} \tag{3}$$

be the set of local features of subgroup n from the initialization c in cluster m. Then the VLAD component m with respect to the initialization c is:

$$v_{n,c,m} = \sum_{f_{n,s} \in S_{n,c,m}} (f_{n,s} - \mu_{n,c,m}). \tag{4}$$

The VLAD representation of subgroup n and k-means c is simply the concatenation of all components $v_{n,c,m}$, as follows:

$$F_{n,c} = [v_{n,c,1}, \ldots, v_{n,c,k}]. \tag{5}$$

As proposed by [7], we apply power law normalization keeping the sign of each component x of the VLAD representation by doing $x \leftarrow sign(x)\sqrt{|x|}$. The PCA inside clusters followed by the power law normalization can be seen as a kind of "whitening" [3]. As noted by [17], whitening vectors is equivalent to replacing the Euclidean distance by the Mahalanobis distance.

Each feature $F_{n,c}$ is a vector which size depends only on the corresponding local feature size and the number of centers in the clustering algorithm. We do a flat concatenation of all features $F_{n,c}$ into a final feature vector here represented by $\vec{x}$. The key factor in the aggregation method is that the fundamental structure of local features are preserved while using multiple clustering representations. That fact allows the next stage to learn the best combination of features and clustering representations.

3.3 Learning features combination

As a result from the previews two stages, we have a feature vector $\vec{x}$ formed by aggregated features that depends on the number of local feature subgroups (n) and the number C of unique k-means initializations. The goal of the feature combination stage is to extract discriminant information that improves the action recognition accuracy, considering the nonparametric k nearest neighbor (k-NN) classifier. In this regard, we employ two stages of metric learning, resulting in a reduced final feature vector, which is used for classification.

The metric learning approach is inspired by the large margin nearest neighbor (LMNN) algorithm proposed by [25]. For simplicity, we define the squared distance (squared l2-norm) between two feature vectors in function of the linear transformation $\mathbf{L}$:

$$\mathcal{D}_{\mathbf{L}}(\vec{x_i}, \vec{x_j}) = \|\mathbf{L}(\vec{x_i} - \vec{x_j})\|^2 \tag{6}$$

3.3.1 Loss function

Considering the k-NN classifier, the loss function is a measurement of violations made by *impostor* samples and distancing among *target* neighbors. Specifically, given one feature vector $\vec{x_i}$, the target neighbors, here represented by $\vec{x_j}$, are those that we want to be closest to $\vec{x_i}$. On the other hand, the *impostors*, represented by $\vec{x_l}$, are those that are closer to $\vec{x_i}$ without being targets. This concept was previously introduced by [25]. The function loss can be represented by two forces: the *pull* and *push* components, trying to respectively pull the targets while pushing the impostors, defined as follows:

$$\varepsilon_{pull}(\mathbf{L}) = \sum_{j \rightarrow i} \mathcal{D}_{\mathbf{L}}(\vec{x_i}, \vec{x_j}) \tag{7}$$

$$\varepsilon_{push}(\mathbf{L}) = \sum_{i,j \to i} \sum_{i,l \not\to i} \lfloor \xi + \mathcal{D}_{\mathbf{L}}(\vec{x_i}, \vec{x_j}) - \mathcal{D}_{\mathbf{L}}(\vec{x_i}, \vec{x_l}) \rfloor \tag{8}$$

where ξ is the desired separation margin between targets and impostors. The notations $j \to i$ and $l \not\to i$ mean that j is the index of targets of sample i while l is the index of impostors of sample i.

如何扩展训练集以避免overfitting

As most of the datasets for human action recognition have a relatively small number of samples from each action, learning algorithms can be very prone to *overfitting* on such data. To cope with this, we added a regularization in the linear transformation $\mathbf{L}$, as presented in the global loss function:

$$\varepsilon(\mathbf{L}) = (1-\mu)\varepsilon_{pull}(\mathbf{L}) + \mu\varepsilon_{push}(\mathbf{L}) + \gamma\|\mathbf{L}^T\mathbf{L} - \mathbf{I}\|^2 \tag{9}$$

where μ is the ratio between "push" and "pull" components, γ is the regularization coefficient, and $\mathbf{I}$ is the identity matrix. The regularization term enforces that the equivalent metric $\mathbf{L}^T\mathbf{L}$ should remains close to the identity matrix.

3.3.2 Global optimization

The optimal transformation $\mathbf{L}^*$ that minimizes Eq. (9) can be found by solving the global optimization problem:

$$\mathbf{L}^* = \arg\min_{\mathbf{L}} \varepsilon(\mathbf{L}) \tag{10}$$

In order to solve Eq. (10) using the gradient descent approach, we compute the derivative term of ε in $\mathbf{L}$, as follows:

$$\begin{aligned} \frac{1}{2}\frac{\partial \varepsilon}{\partial \mathbf{L}} &= (1-\mu)\mathbf{L} \sum_{i,j \to i} (\vec{x_i} - \vec{x_j})(\vec{x_i} - \vec{x_j})^T \\ &+ \mu\mathbf{L} \sum_{i,j \to i} \sum_{i,l \not\to i} [(\vec{x_i} - \vec{x_j})(\vec{x_i} - \vec{x_j})^T \\ &- (\vec{x_i} - \vec{x_l})(\vec{x_i} - \vec{x_l})^T] + 2\gamma\mathbf{L}(\mathbf{L}^T\mathbf{L} - \mathbf{I}) \end{aligned} \tag{11}$$

Since the number of operations required to solve Eq. (11) can be significantly large even for small training datasets, we employ a minimization algorithm based on stochastic gradient descent [1]. Let us define $\mathbb{D}$ as the training dataset. In the SGD optimization, for each iteration, we randomly select a small subset from $\mathbb{D}$ defined as $\mathbb{S}$. Iterating over the samples in $\mathbb{S}$ i.e., the index i is restricted to samples in $\mathbb{S}$, we solve Eq. (11) taking targets and impostors from the whole dataset $\mathbb{D}$. A good initialization of $\mathbf{L}$ can be done by taking the eigenvectors of the covariance matrix of $\mathbb{D}$, which means to initialize $\mathbf{L}$ with the PCA on $\mathbb{D}$. This is a good initialization since we reduce the feature size in the metric learning stages and PCA is known to be a good dimension reduction technique. The optimization is performed until the maximum number of epochs (*MaxEpoch*) is reached or the gradient vanishes according to the threshold ϑ. The global SGD optimization is presented in Algorithm 1.

Algorithm 1 Global SGD optimization.

```
Require: Training dataset $\mathbb{D}$, $MaxEpoch$, vanishing
    value $\vartheta$
1: Do PCA on $\mathbb{D}$ to initialize $\mathbf{L}$
2: $Epoch \leftarrow 0$
3: repeat
4:     $\mathbb{S} \leftarrow$ Randomly select samples from $\mathbb{D}$
5:     $\mathbf{G} \leftarrow$ Solve Eq. (11) for the subset $\mathbb{S}$
6:     $\mathbf{L} \leftarrow \mathbf{L} - \eta\mathbf{G}$
7:     $Epoch \leftarrow Epoch + sizeof(\mathbb{S})/sizeof(\mathbb{D})$
8: until $(\|\mathbf{G}\| \geq \vartheta)$ and $(Epoch < MaxEpoch)$
9: return $\mathbf{L}$
```

As shown in Fig. 2, two stages of metric learning were used in our method. The first one aimed to reduce the feature size while performing a first separation between targets and impostors by learning the transformation $\mathbf{L_1}$. In this regard, we set $\mu = 0.9$ and $MaxEpoch = 2$. The second stage works on smaller features and learns the transformation $\mathbf{L_2}$ with $\mu = 0.5$ and $MaxEpoch = 50$. Each metric learning stage is individually optimized following the Algorithm 1, replacing $\mathbf{L}$ by $\mathbf{L_1}$ and $\mathbf{L_2}$, respectively at each etage. Namely, we first learn $\mathbf{L_1}$, which takes $\vec{x}$ as input, then in the second stage we learn $\mathbf{L_2}$ which input is $\mathbf{L_1}\vec{x}$. This approach allows a fast learning process in addition to avoiding overfitting, since the feature size is reduced in a few iterations and the more intensive learning stage is performed over fewer parameters. Finally, since all transformations are linear, in the testing evaluation we can use $\mathbf{L} = \mathbf{L_2}\mathbf{L_1}$ to represent the full learned transformation.

4 Experiments

We evaluated the accuracy of our method on three publicly available datasets. The MSR-Action3D [9] is the most common dataset for 3D human action recognition according to [30] and is composed by 10 subjects performing 20 actions chosen in the context of gaming, which include: *high wave, horizontal wave, hammer, hand catch, high throw, draw X, draw tick, draw circle, hand clamp, two hand clamp, two hand wave, side boxing, bend, forward kick, side kick, jogging, tennis swing, tennis serve, golf swing*, and *pick-up throw*. This dataset is challenging due to some pairs of actions very similar, for example: *hand catch* and *draw tick*, or *pick-up throw* and *bend*. The UTKinect-Action3D dataset [27] is composed by 10 subjects, of which nine are males and one is female including one left-handed, performing 10 actions: *walk, sit down, stand up, pick up, carry, throw, push, pull, wave*, and *clap hands*. Each subject perform actions in various views and the length of videos vary from 5 to 120 frames, resulting in significant variation among the recordings. The Florence 3D Actions dataset [18] is composed by 10 subjects performing 9 actions recorded in distinct environment conditions, which include: *wave, drink from a bottle, answer phone, clap, tight lace, sit down, stand up, read watch,* and *bow*. Since our feature extraction method requires only 15 skeleton joints, we averaged the joints from hands and wrist into a single joint *hand* for the datasets MSR-Action3D and UTKinect-Action3D, in which skeletons are composed by 20 joints. We apply the same process to foot and ankle, and to spine and center hip.

For all datasets, we use the already computed skeleton joints data and the same parameters in all the performed tests. We optimized the hyperparameters of our method using only the MSR-Action3D dataset split as proposed by [24]. Seven local feature subgroups were extracted as described in Sec. 3.1. In the feature aggregation stage, we use a pool of five unique k-means ($C = 5$), each one computing $k = 23$ clusters. After the feature concatenation stage, the resulting feature vectors are of size 8970. In both metric learning stages, we set $\gamma = 0.1$ and $\xi = 0.1$. In the SGD optimization, we solve the Eq. 11 by taking batches of 32 training samples. In the first metric learning stage we set the output dimension to 512, followed by the second stage with output dimension equal to 256, which is the final feature size. The final classification is a seven nearest neighbors voting.

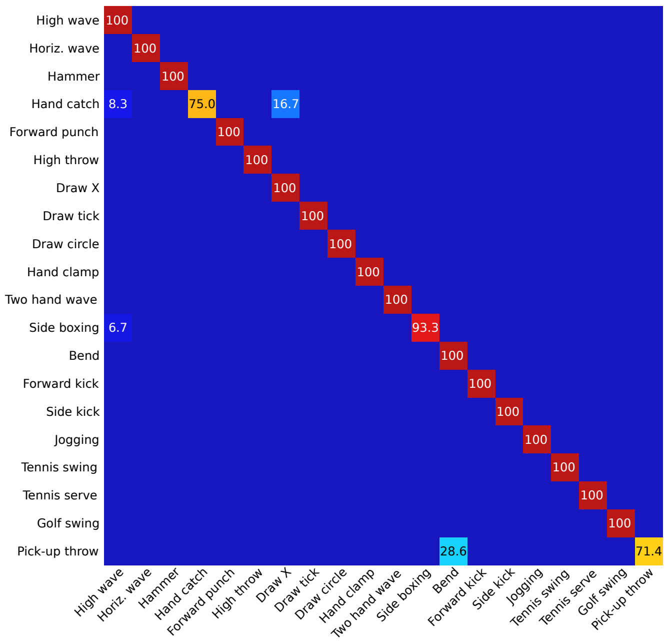

Figure 5: Confusion matrix for action classification on the MSR-Action3D dataset resulted from the proposed method.

4.1 Comparison with the state of the art

We compared our results with several methods in the state of the art on three distinct datasets, as presented in Table 3.

4.1.1 MSR-Action3D dataset

The MSR-Action3D dataset has been used by several works in many disparate ways. In our tests, we selected the two most relevant evaluation approaches on this dataset. The first approach we used is the cross-subject splitting proposed by [24], where subjects 1,3,5,7,9 are used for training and subjects 2,4,6,8,10 are used for testing. In that case, the accuracy of our method is 97.1%, which is the best result on this data as far as we know. Comparable results are shown in Table 3. As can be seen in the confusion matrix (Fig. 5) resulting from our method, several actions were classified without any mistake and only two actions presented classification accuracy lower than 93%. The second approach for evaluation we used was proposed by [14], where we report the average result among all possible 5-5 subject splits. We consider this approach the most relevant, since it reduces the possibility of effects from particular combinations. By this approach, our method achieved an average accuracy of 90.36% and a standard deviation of 2.45%, which is an improvement of 3.08% over the best method so far [4]. We reinforce that in both approaches results are reported in the cross-subject scenario. The results from other methods using the same assessment are reported in the third column of Table 3.

4.1.2 UTKinect-Action3D dataset

On this dataset, the authors [27] proposed to use the leave one actor out cross validation (LOOCV) scheme. Specifically, one actor is removed from training and used as testing. This process is repeated for all actors and the final result is the average accuracy of all runs. On our tests, we followed the same procedure and our method achieved on average 98.00% of accuracy with a standard deviation of 3.49%, as reported in the fourth column of Table 3. We consider the LOOCV scheme statistically more stable than the single cross-validation assessment employed by [32] and [8]. Therefore, our results are not comparable on this dataset.

4.1.3 Florence 3D Actions dataset

Similarly to the previews experiment, we evaluate the performance of our method on the Florence 3D Actins dataset using the LOOCV approach, as suggested by the authors [18]. On average, our method classified 94.39% of actions correctly. Our method exceeded the state-of-the-art approaches by a significant margin, as presented in the fifth column of Table 3.

4.2 Contribution of each method's stage

In this section, we discuss the influence of each part of our method, as tested on the MSR-Action3D dataset.

- First, using all joints together instead of our proposed subgroups leads to a performance decrease of 5.5%.

Table 3: Accuracy evaluation of our method compared to the state-of-the-art methods on three public datasets. Columns two and three present results on the MSR-Action3D dataset using the protocols proposed by [24] and [14], respectively. Columns four and five respectively show results on UTKinect-Action3D and Florence 3D Actions datasets.

Dataset / Method	MSR-Action3D protocol of [24]	MSR-Action3D protocol of [14]	UTKinect-Action3D	Florence 3D Actions
[24]	88.2%	—	—	—
[27]	—	—	90.92% ± 1.74%	—
[12]	96.7%	—	—	—
[14]	88.36%	82.15% ± 4.18%	—	—
[18]	—	—	—	82.0%
[21]	88.89%	—	—	—
[4]	92.1%	87.28% ± 2.41%	91.5%	87.04%
[11]	95.62%	—	—	—
[23]	89.48%	—	97.08%	90.88%
[28]	93.09%	—	—	—
[20]	91.21%	—	88.5%	—
[22]	92.03%	—	—	—
[8]	92.2%	—	—	—
[16]	—	86.5%	—	—
Our method	**97.1%**	**90.36% ± 2.45%**	**98.00% ± 3.49%**	**94.39%**

- If only displacement vectors or only relative positions are used, the classification accuracy drops by 4.1% and 17.2%, respectively.

- In the feature aggregation stage, if the PCA or the power law normalization is turned off, the performance decreases by 4.4% and 4.8%, respectively.

- Similarly, aggregating features with a single clustering initialization, i.e., setting $C = 1$, drops the performance by 2.5%.

- Replacing the proposed two stages of metric learning by features reduction with PCA, it means using PCA to reduce the feature size from 8970 to 512 and then using a single metric learning stage, the best performance decreases by 0.8%.

- Removing the regularization coefficient from Eq. 9 reduces the best performance by 0.4% and led to faster overfitting.

- Finally, replacing the k-NN classifier by SVM or neural network (MLP) drops the performance by 1.5% in the best case (see Table 4).

The conception of the proposed framework was reasoned that each part is optimally designed regarding the next stage in the pipeline. For instance, the local features extraction provides small features that can be clustered well, while avoiding early combination of distinct information. In the clustering stage, we use multiple initializations to increase the chances to have a better representation, which can be learned in the next stage. Similarly, the metric learning algorithm (LMNN) is optimal to increase the nearest neighborhood (k-NN) classifier accuracy.

The multiple clustering initialization is an important step in the feature aggregation method and goes beyond the improvement on classification accuracy of the proposed framework. As shown in Fig. 6, the probability of reaching better accuracy drastically increases after metric learning when using $C = 5$. This effect can also be observed by the standard deviation decreasing from 1.34% to 0.71%, respectively using $C = 1$ and $C = 5$. This fact is expected, since the metric learning can extract complementary information from different clustering representations. Additionally, the metric learning stage can be seen as the point of convergence where all the particular improvements are intensified, resulting in a final improvement of 12% as shown in Fig. 7, while drastically reducing the feature size.

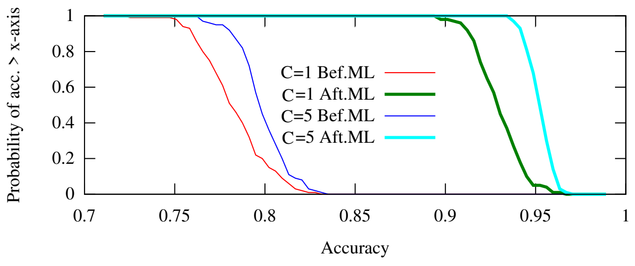

Figure 6: Probability of accuracy according to 100 random evaluations using $C = 1$ and $C = 5$ in the feature aggregation stage, before (Bef.ML) and after (Aft.ML) metric learning.

We evaluate the influence of the last stages by replacing the k-NN classifier by two well known classifiers, before and after the metric learning (LMNN) stage. First, we compared with a standard SVM [2] using the sigmoid

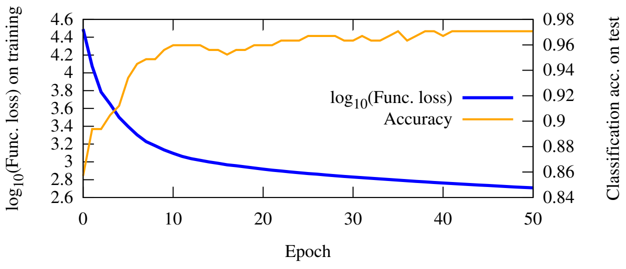

Figure 7: The learning curves: evolution of the function loss on training and the classification accuracy on testing samples.

kernel setting the parameters *gamma* and C respectively to 1 and 10. Second, we compared with a neural network (MLP) with two fully-connected layers, the first using *ReLu* activation and the second using *softmax* for classification. Table 4 shows that the k-NN classifier on learned features by the metric learning stage gives the best result, even though both SVM and neural network present results comparable to the state of the art, which demonstrates the robustness of the proposed features representation. These are expected results, since the objective function of the metric learning stage was specially designed to increase the k-NN classification accuracy.

Table 4: Classification accuracy of the proposed metric learning and classifier stages compared to SVM and neural network approaches. We evaluate the classifiers taking as input the aggregated features ($\vec{x}$) and the learned features after LMNN.

Classifier	Aggregated features	Learned features (LMNN)
SVM	93.4%	95.6%
Neural net.	93.8%	95.6%
k-NN	83.2%	**97.1%**

4.3 Computation time

The average testing runtime of the proposed method is presented in Table 5, which is faster than the computation time reported by [20]. The testing sequences were processed in a laptop machine with Intel® Core^{TM} i7-4710MQ processor, after training. One of the reasons which led to low computing time for action recognition is that the most complex part of our method is the metric learning feature combination. Once the training stage is finished, recognizing new sequences is a fast and straightforward process.

5 Conclusion

In this work, we presented a new framework for human action recognition using only skeleton joints extracted from depth maps. We proposed extracting sets of spatial and temporal local features from subgroups of joints. Local features are aggregated into several feature vectors by a robust method using the VLAD algorithm and a pool of clusters, providing a good representation for long and short actions. All the feature vectors are then efficiently combined by a metric learning method inspired by the LMNN algorithm, which is used to extract the most discriminant information from features with the objective to improve the accuracy of the k-NN classifier.

Extensive experiments with the proposed framework show that all the proposed steps contribute significantly to improve classification accuracy. We conclude that spatial and temporal information, as well as the multiple clustering representations, could be efficiently combined by the metric learning approach, resulting in a significant increase of performance. Moreover, the proposed method relies on a few external parameters and our experiments show that the method generalizes well, since its performance overcame all the results in the state of the art on three important datasets, using the same parameters in all evaluations.

Table 5: Average testing runtime of the proposed method on three datasets. The given computation time in milliseconds refers to one testing sequence.

Dataset / Stage	MSR-Action3D	UTKinect Action3D	Florence 3D Actions
Local features extraction (ms)	2.34	2.00	1.52
Features aggregation (ms)	4.59	4.92	4.14
Features combination (ms)	0.14	0.18	0.15
Classification k-NN (ms)	1.38	0.97	1.06
Average testing time (ms)	**8.45**	**8.07**	**6.87**

Acknowledgment

This work was partially supported by the Brazilian National Council for Scientific and Technological Development (CNPq) – Grant 233342/2014-1.

References

[1] Léon Bottou. Stochastic Learning. In Olivier Bousquet and Ulrike von Luxburg, editors, *Advanced Lectures on Machine Learning*, Lecture Notes in Artificial Intelligence, LNAI 3176, pages 146–168. Springer Verlag, Berlin, 2004.

[2] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–

27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[3] Jonathan Delhumeau, Philippe-Henri Gosselin, Hervé Jégou, and Patrick Pérez. Revisiting the VLAD image representation. In *ACM Multimedia*, Barcelona, Spain, October 2013.

[4] Maxime Devanne, Hazem Wannous, Stefano Berretti, Pietro Pala, Mohamed Daoudi, and Alberto Del Bimbo. 3D Human Action Recognition by Shape Analysis of Motion Trajectories on Riemannian Manifold. *IEEE Transactions on Cybernetics*, Aug 2014.

[5] Richard G Freedman, Hee-Tae Jung, and Shlomo Zilberstein. Temporal and object relations in unsupervised plan and activity recognition. In *AI for HRI: Papers from the AAAI 2015 Fall Symp*, pages 48–50, 2015.

[6] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3304–3311, 2010.

[7] Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Pérez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(9):1704–1716, Sep 2012.

[8] Meng Li and Howard Leung. Graph-based approach for 3D human skeletal action recognition. *Pattern Recognition Letters*, pages –, 2016.

[9] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Action Recognition Based on a Bag of 3D Points. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9–14, 2010.

[10] Bin Liang and Lihong Zheng. A Survey on Human Action Recognition Using Depth Sensors. In *International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8, 2015.

[11] C. Lu, J. Jia, and C. K. Tang. Range-Sample Depth Feature for Action Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 772–779, 2014.

[12] Jiajia Luo, Wei Wang, and Hairong Qi. Group Sparsity and Geometry Constrained Dictionary Learning for Action Recognition from Depth Maps. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1809–1816, 2013.

[13] V. Magnanimo, M. Saveriano, S. Rossi, and D. Lee. A Bayesian approach for task recognition and future human activity prediction. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 726–731, 2014.

[14] O. Oreifej and Zicheng Liu. HON4D: Histogram of Oriented 4D Normals for Activity Recognition from Depth Sequences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 716–723, June 2013.

[15] Ronald Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976 – 990, 2010.

[16] H. Rahmani, A. Mahmood, D. Huynh, and A. Mian. Histogram of Oriented Principal Components for Cross-View Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, PP(99):1–1, 2016.

[17] Bahjat Safadi, Nadia Derbas, and Georges Quénot. Descriptor Optimization for Multimedia Indexing and Retrieval. *Multimedia Tools and Applications (MTAP)*, 74(4):1267–1290, Feb 2015.

[18] L. Seidenari, V. Varano, S. Berretti, A. Del Bimbo, and P. Pala. Recognizing Actions from Depth Cameras as Weakly Aligned Multi-part Bag-of-Poses. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 479–485, 2013.

[19] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time Human Pose Recognition in Parts from Single Depth Images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '11, pages 1297–1304, 2011.

[20] Rim Slama, Hazem Wannous, Mohamed Daoudi, and Anuj Srivastava. Accurate 3D action recognition using learning on the Grassmann manifold. *Pattern Recognition*, 48(2):556 – 567, 2015.

[21] Q. D. Tran and N. Q. Ly. An effective fusion scheme of spatio-temporal features for human action recognition in RGB-D video. In *International Conference on Control, Automation and Information Sciences (ICCAIS)*, pages 246–251, 2013.

[22] Vivek Veeriah, Naifan Zhuang, and Guo-Jun Qi. Differential Recurrent Neural Networks for Action Recognition. In *IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[23] R. Vemulapalli, F. Arrate, and R. Chellappa. Human Action Recognition by Representing 3D Skeletons as Points in a Lie Group. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 588–595, 2014.

[24] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Mining Actionlet Ensemble for Action Recognition with Depth Cameras. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1290–1297, June 2012.

[25] K.Q. Weinberger and L.K. Saul. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *The Journal of Machine Learning Research (JMLR)*, 10:207–244, 2009.

[26] Daniel Weinland, Remi Ronfard, and Edmond Boyer. A Survey of Vision-based Methods for Action Representation, Segmentation and Recognition. *Computer Vision and Image Understanding*, 115(2):224–241, feb 2011.

[27] Lu Xia, Chia-Chih Chen, and J. K. Aggarwal. View Invariant Human Action Recognition Using Histograms of 3D Joints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20–27. IEEE, 2012.

[28] X. Yang and Y. Tian. Super Normal Vector for Activity Recognition Using Depth Sequences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 804–811, 2014.

[29] M. Zanfir, M. Leordeanu, and C. Sminchisescu. The Moving Pose: An Efficient 3D Kinematics Descriptor for Low-Latency Action Recognition and Detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2752–2759, 2013.

[30] Jing Zhang, Wanqing Li, Philip O. Ogunbona, Pichao Wang, and Chang Tang. RGB-D-based action recognition datasets: A survey. *Pattern Recognition*, 60:86 – 105, 2016.

[31] Wentao Zhu, Cuiling Lan, Junliang Xing, Wenjun Zeng, Yanghao Li, Li Shen, and Xiaohui Xie. Co-occurrence Feature Learning for Skeleton based Action Recognition using Regularized Deep LSTM Networks. In *Association for the Advancement of Artificial Intelligence (AAAI)*, February 2016.

[32] Y. Zhu, W. Chen, and G. Guo. Fusing Spatiotemporal Features and Joints for 3D Action Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, pages 486–491, 2013.