Noname manuscript No.
(will be inserted by the editor)

Representing Human Motion with FADE and U-FADE: an Efficient Frequency-Domain Approach

Pietro Falco · Matteo Saveriano · Dharmil Shah · Dongheui Lee

Received: date / Accepted: date

Abstract In this work, we present FADE, a frequency-based descriptor to encode human motion. FADE is simple, and provides high compression rate and low computational complexity. In order to reduce space and time complexity, we exploit the biomechanical property that human motion is bounded in frequency. FADE and U-FADE can be used in combination with both supervised and unsupervised learning approaches in order to classify and cluster human actions, respectively. We present also a branch of FADE, called Uncompressed FADE (U-FADE). U-FADE performs well in combination with some unsupervised algorithms such as Spectral Clustering (SC), paying the price of a reduced compression rate. Also, U-FADE performs in general better than FADE well with small datasets. We tested our descriptors with well-known, public motion databases, such as HDM05, Berkeley MHAD, and MSR. Moreover, we compared FADE and U-FADE with diverse state of the art approaches.

P. Falco was with the Department of Human-Centered Assistive Robotics, Technical University of Munich, Munich 80333, Germany. He is now with the Department of Automation Solutions, ABB Corporate Research, Västeras 72178, Sweden. E-mail: pietro.falco@se.abb.com
M. Saveriano was with the Department of Human-Centered Assistive Robotics, Technical University of Munich, Munich 80333, Germany. He is now with German Aerospace Center, Institute of Robotics and Mechatronics, Münchener Str. 20, 82234 Wessling, Germany. E-mail: matteo.saveriano@dlr.de
Dharmil Shah was with the Department of Human-Centered Assistive Robotics, Technical University of Munich, Munich 80333, Germany.
Dongheui Lee is with Department of Human-Centered Assistive Robotics, Technical University of Munich, Munich 80333, Germany and with German Aerospace Center, Institute of Robotics and Mechatronics, Münchener Str. 20, 82234 Wessling, Germany. E-mail: dhlee@tum.de

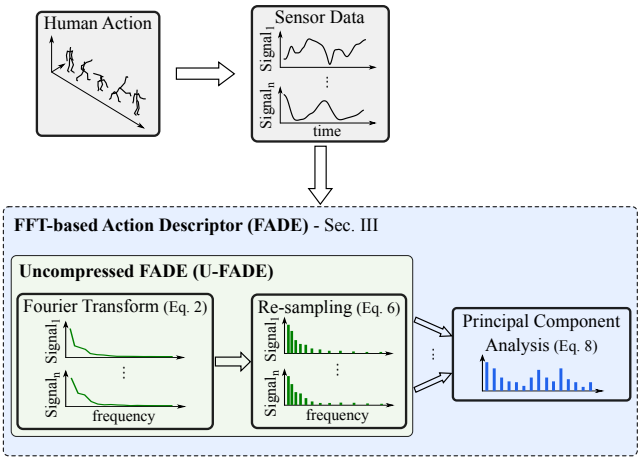


Fig. 1 Overview of FADE approach

Keywords Human Action Recognition · Motion Analysis · Descriptors for Human Motion

1 Introduction

In the last decade, human-robot interaction and cooperation gained an intensive interest within the scientific community. An important step to achieve an effective interaction between humans and robots consists in understanding human actions and, consequently, human intentions. Human behavior, in fact, is difficult to model and predict. Machine learning methods have the potential to play a key role in this research area. Motion analysis and in particular action recognition are topics of great interest in computer vision and human movement science. The robotics community can benefit greatly by the results achieved within such communities. However, the methods for human movement interpretation designed for robotics have additional re-

quirements that are domain-specific. In particular, descriptors for robotic applications require

1. compatibility with multiple sensor modalities,
2. low computational complexity,
3. high compression rate, and
4. robustness to noise.

The first requirement is essential in modern robotics. Since we want to provide robots with the capability to operate in unstructured environments, the sensors adopted to observe humans are diverse. As a consequence, also the state variables to represent human motion can be potentially of many types. For example, in environments with generous light conditions, the robot can observe humans with a simple RGB-D camera. With such a perception modality, it is convenient to represent human body configuration in terms of joint Cartesian positions. In different scenarios, for example in outdoor environments or in environments with variable light conditions, humans can use wearable devices like accelerometers or inertial measurement units. In this case, the motion is generally represented through joint angles. An effective descriptor has to work with both the representations. Since FADE is based on frequency properties of human motion, it easily adapts to different data input. In this work, we test FADE and U-FADE with both joint angles and joint positions.

Autonomous systems often have limited computational power and the energy consumption has to be limited as much as possible. Data-efficient methods, which do not require powerful GPU and are suitable for frequent retrain, are a key feature for the success of such a technology.

The third requirement, i.e., achieving an high compression rate, is important since a high-dimensional representation requires higher computational cost for the classification (or clustering) step. Also, representations with high space complexity are more easily exposed to the curse of dimensionality.

Robustness to noise, the fourth requirement, is clearly a feature that algorithms for modern robotics have to present. When the environment is unstructured, noise of weakly-known nature and the possibility of missing data have to be taken into account for a successful system design.

In this work, we discuss how FADE and U-FADE perform at the light of these four requirements. We compare our approach with state of the art methods by using as criteria these four features. This work extends the approach proposed in (Shah et al, 2016), in which preliminary results are presented only on joint angles as input and with a minimal number of comparisons with existing approaches. In the extended journal version, we include several experiments on three public datasets, the comparison with several well-known approaches, the joint 3D-position input mode, and more insights on how to tune the parameters of the descriptors. With the extended set of experiments, we show in which applications and in which conditions FADE and U-FADE can be a valid alternative to other approaches in the literature.

The rest of the paper is structured as follows. In Section 2 we mention papers in the literature concerning action recognition. Section 3 presents the algorithm to encode human motion with FADE and U-FADE. The experimental results on three public motion dataset, as well as comparison with state-of-the-art approaches, are shown in Section 4. Section 5 reports conclusions and future research directions.

2 Related Work

Kulic et al. proposed an unsupervised method based on Hidden Markov Models (Rabiner, 1989) to represent and cluster actions (Kulić et al, 2008, 2011). This approach introduces the possibility to learn a number of actions in an unsupervised incremental fashion. It was further extended in human-robot interaction and human-robot collaboration (Lee et al, 2009) (Medina et al, 2011). The approach presented in (Cavallo and Falco, 2014) leverages Singular Value Decomposition (SVD) to represent human manipulation actions, Euclidean distance to measure the similarity among actions, k-means and k-NN for off-line and online clustering respectively. HMM-based models are adopted for grasping and manipulation gesture recognition using joint angles and fingertip force data in (Di Benedetto et al, 2016) and (Schmidts et al, 2011). A template-based approach to recognize actions (Leightley et al, 2014) uses a small set of a-priori known actions called templates. To align observed actions with the example actions, the dynamic time warping is adopted (Sakoe and Chiba, 1978). In (Pervez et al, 2017), observed actions were aligned without the preprocessing of dynamic time warping, but during the EM algorithm.

The work presented in (Wang et al, 2012) considers frequency domain, and exploits ensembles models learnt to represent each action and to capture the intra-class variance. The method shows promising results in dealing with data from depth cameras. The approach is supervised and it uses a Support Vector Machine (SVM) training method (Bishop, 2006). Compared with our approach, the method proposed in (Wang et al, 2012) adopts a different descriptor, which is based on pairwise joint relative positions and it only uses input data based on joint Cartesian position. It clearly shows how

information in the frequency domain can be valuable in human action recognition. A frequency-based approach is also adopted in (Wang et al, 2015) for segmentation of human repetitive actions. In (Wang et al, 2016) the action recognition is performed by mining a set of key-pose-motifs.

Ofli et al. suggested the SMIJ (Sequence of the Most Informative Joint) for action recognition (Ofli et al, 2013), which is based on ranking the informative joints involved in an action (Ofli et al, 2014). In particular, the set of joints that present the maximum variance during the motion are considered most informative. The approach was tested on 16 actions in the HDM05 database and on 11 actions in the Berkeley MHAD database (Ofli et al, 2013). On 11 actions of HDM05, the authors reach 84% accuracy with a supervised learning approach. In (Falco et al, 2017), a descriptor is proposed that is based on motion coordination. In (Evangelidis et al, 2014), a simple features based on skeletal joint quads are introduced, which achieve an interesting balance between accuracy and computational cost.

In (Le Naour et al, 2012) the authors proposed a representation that exploits the pair-wise joint-to-joint distances in the skeletal model. Afterwards, the dimensionality is reduced by Principal Component Analysis (PCA). The descriptor is associated to a 2-NN method to classify the actions.

One of the main limitations of the state-of-the-art approaches is the scalability to a large number of actions and classes (Chen and Koskela, 2015). The scalability is difficult to achieve because of diverse problems: potential complexity in the representation of actions, potential complexity in computing distances between actions, difficulty to differentiate actions in presence of a high numbers of classes, and heavy curse of dimensionality in the classification process. Deep learning approaches are potentially interesting to tackle these challenges. In (Chen and Koskela, 2015) a method is proposed to alleviate this problem. The method leverages a skeleton-based action descriptor and it is tested with extreme learning machines (Huang et al, 2006) and SVM for the classification step. The descriptor is defined as skeleton-based (or model-based) because it requires the knowledge of the skeletal model of the performer to obtain a user-independent normalized representation. Using the same skeleton-based features, in (Cho and Chen, 2014) a convolutional neural network is proposed to classify motion capture sequences, while in (Du et al, 2015) a hierarchical recurrent neural network (RNN) is adopted. Deep Long Short-Term Memory (LSTM) networks for skeleton based action recognition are used in (Zhu et al, 2016) and (Liu et al, 2016). In (Mahasseni and Todorovic, 2016), LSTM is exploited to improve action recognition in video by providing 3D human-skeleton sequences as an additional modality in training data. Despite the good performance in terms of accuracy, deep learning methods require a big amount of training data and long training time.

3 Representing Motion with FADE and U-FADE

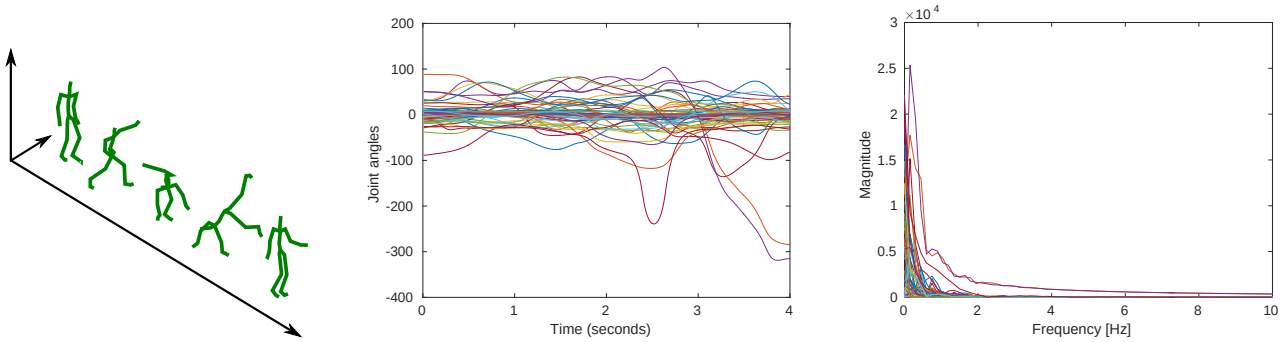
In this section, we describe the algorithms to represent human motion with FADE and U-FADE. In order to successfully derive a motion descriptor that is independent from position and orientation of the world reference frame, we need to leverage invariant representations of whole body motions. In the scientific community, there are two well-known types of human motion representations, which can be used for full-body motion. The first is represented by joint angles, the second consists in Cartesian joint positions. In principle, FADE can be used with both the representations. In practical applications, representations based on joint angles can be used mainly with wearable devices based on accelerometers or IMU. Also, most motion capture systems allow to compute joint angles using inverse kinematic from a skeletal model. Representations based on joint positions are common in the computer vision community. Such representations are particular common in works that exploit low-cost RGB-D cameras and motion capture systems. This representation, however, requires a skeletal model, which is not always available in robotics applications and it is not natively invariant to position and orientation of the world frame.

3.1 Encoding Algorithms

We define the FADE action representation as the function $f : \mathcal{A} \rightarrow \mathbb{R}^m$, where $\mathcal{A}$ is a set of human actions. In this work, each action $A \in \mathcal{A}$ is expressed with a matrix $\boldsymbol{A}_t$ of the form

$$\boldsymbol{A}_t = \begin{pmatrix} x_1(t_1) & x_2(t_1) & \dots & x_J(t_1) \\ x_1(t_2) & x_2(t_2) & \dots & x_J(t_2) \\ \dots & \dots & \dots & \dots \\ x_1(t_N) & x_2(t_N) & \dots & x_J(t_N) \end{pmatrix}. \quad (1)$$

The vector $\boldsymbol{x}_k = [x_1(t_k), \dots, x_J(t_k)]$ contains the components of the chosen invariant representation at the discrete time frame k. In our work, J is the dimension of the chosen representation. In particular, if we choose joint angles, J is the number of angular signals that describe the human body configuration. If we choose joint Cartesian positions, we have $J = 3 \times n_{joints}$, where n_{joints} is the number of joints of the skeletal model. The



(a) Visualization (b) Joint angles in the time domain (c) FFT of the joint angular values

Fig. 2 Example of visualization, time domain signals, and frequency domain signals of a HDM05 *cartwheel* action

number of joints is multiplied by 3 since we express the position of each joint with a point in the 3D space. In Eq. (1) we use the symbol $\boldsymbol{A}_t$ to indicate that the matrix contains values in the time domain. The first step of FADE is to compute the Discrete Fourier Transform (DFT) of the time-domain signals. In order to compute the DFT, we leverage the Fast Fourier Transform algorithm. For more details about the FFT, refer to (Walker, 1996). Applying the FFT algorithm we have:

$$\boldsymbol{A}_f = \mathrm{FFT}(\boldsymbol{A}_\mathrm{t}). \tag{2}$$

The columns of the matrix $\boldsymbol{A}_f$ contains the FFT of the columns of the matrix $\boldsymbol{A}_t$

$$\boldsymbol{A}_f = \begin{pmatrix} x_1(f_1) & x_2(f_1) & \dots & x_J(f_1) \\ x_1(f_2) & x_2(f_2) & \dots & x_J(f_2) \\ \dots & \dots & \dots & \dots \\ x_1(f_N) & x_2(f_N) & \dots & x_J(f_N) \end{pmatrix}. \tag{3}$$

The matrix $\boldsymbol{A}_f$ has the same size as the matrix $\boldsymbol{A}_t$. We have that

$$\Delta f = f_{i+1} - f_i = f_s/N \,, \forall i \in \{1, 2, ..., N-1\}, \tag{4}$$

where Δf is the resolution in the frequency domain, f_s is the sampling frequency and N is the number of samples in the time domain. Since the human motion does not contain significant frequency components beyond 10 Hz (Forestier and Nougier, 1998), we can remove the values above the threshold $f_{th} = 10$ Hz, considering the left-open interval $(0,\ 10]$ Hz. After this step, we describe the action with the following matrix:

$$\boldsymbol{A}_f^{th} = \begin{pmatrix} x_1(f_1) & x_2(f_1) & \dots & x_J(f_1) \\ x_1(f_2) & x_2(f_2) & \dots & x_J(f_2) \\ \dots & \dots & \dots & \dots \\ x_1(f_{th}) & x_2(f_{th}) & \dots & x_J(f_{th}) \end{pmatrix}. \tag{5}$$

This is a significant step in FADE and U-FADE, since it allows us to reduce significantly the number of points and to identify a constant interval that contains all the significant information about the motion. As it can be seen in Eq. (4), the resolution in the frequency domain depends on the number of samples in the time domain N. As a consequence, each action will present a different resolution. To solve the problem, we resample the data in the frequency domain with a linear interpolation methods (Dyn et al, 1990), obtaining to a resolution in the frequency domain $\tilde{\Delta f}$ that is constant and equal for all the actions. After computing the sampling point in the frequency domain we obtain the matrix

$$\boldsymbol{A}_{\tilde{f}} = \begin{pmatrix} x_1(\tilde{f}_1) & x_2(\tilde{f}_1) & \dots & x_J(\tilde{f}_1) \\ x_1(\tilde{f}_2) & x_2(\tilde{f}_2) & \dots & x_J(\tilde{f}_2) \\ \dots & \dots & \dots & \dots \\ x_1(\tilde{f}_{th}) & x_2(\tilde{f}_{th}) & \dots & x_J(\tilde{f}_{th}) \end{pmatrix}. \tag{6}$$

The set of all the selected sampling frequencies is denoted as Ω_K, where K is the cardinality of the set Ω_K, i.e. the number of points we sample in the frequency domain. We have that

$$K = \frac{f_{th}}{\tilde{\Delta f}}. \tag{7}$$

In order to compute FADE, we apply the Principal Component Analysis (PCA) on the matrix $\boldsymbol{A}_{\tilde{f}}$, with the aim of maximizing the compression of our descriptor:

$$\boldsymbol{V}_{\tilde{f}} = \mathrm{PCA}(\boldsymbol{A}_{\tilde{f}}). \tag{8}$$

The matrix $\boldsymbol{V}_{\tilde{f}}$ contains the PCA coefficients and has dimension $J \times J$, where J is the number of joints. To derive the FADE representation, we choose the first column of the matrix $\boldsymbol{V}_{\tilde{f}}$ and denote it as $\boldsymbol{v}$. The vector $\boldsymbol{v}$ is then the FADE representation of the action $\boldsymbol{A}_t$ and we can use the notation: $\boldsymbol{v} = \mathrm{FADE}(\boldsymbol{A}_\mathrm{t})$. In this work, we have chosen $K = 500$. This value offers a good trade-off between accuracy and computational time. The dimension of the FADE descriptor is then $J \times 1$. A sequential description of the representation procedure is described in Algorithm 1. We have chosen PCA for compressing our descriptor because (i) it

is a very mature and well-known technique, (ii) it is very easy to find optimized software implementations in most programming languages, (iii) it can be applied for both supervised and unsupervised learning approaches, since PCA does not require data labels, and (iv) it does not require a training phase like for example neural autoencoders (Hinton and Salakhutdinov, 2006).

To derive U-FADE, instead of performing the PCA in Eq. (8), we simply reshape the matrix $\boldsymbol{A}_{\tilde{f}}$ into a $J \cdot K$ column vector, where K is the number of points in the frequency domain. In particular, the U-FADE descriptor has the following structure:

$$\boldsymbol{v}_U = [x_1(\tilde{f}_1), ..., x_1(\tilde{f}_{th}), ..., x_J(\tilde{f}_1), ..., x_J(\tilde{f}_{th})]^{\mathrm{T}} \quad (9)$$

In U-FADE, hence, there is no compression but only a reshape operation. Algorithm 2 reports the procedure to derive U-FADE.

In Fig 2(a), an example of the action *cartwheel* is shown. Fig. 2(b) shows the joint angular signals of the same action in the time domain. In Fig. 2(c) the FFT of the matrix $\boldsymbol{A}_t$ relative to the action cartwheel is computed. Each signal depicted in Figures 2(b) and 2(c) is associated to a color and represents one of the 56 joint angles as a function of the time and of the frequency respectively. From Fig. 2(c) it is evident how the significant information about a complex action like cartwheel is all contained in the range $0 - 5$ Hz. For any action or automatically segmented primitive of any time duration, the information useful to discriminate an action is contained always in a very limited interval of frequencies.

3.2 Analysis of Time and Space Complexity

The asymptotic complexity of FADE and U-FADE as a function of the time-frame number is $O(n \log n)$. To derive this complexity, we consider the steps of Algorithm 1. In line 2 we compute the FFT, whose $O(.)$ complexity is $O(n \log n)$. In line 3 we resample the signal in the frequency domain stopping at f_{th}. Using linear interpolation for resampling the complexity is $O(n)$. Line 4 computes the PCA of the matrix $\mathbf{A}_{\tilde{f}}$. Since the dimension of $\mathbf{A}_{\tilde{f}}$ is fixed and does not depend on n, we have $O(1)$ time complexity. The total cost is then $O(n \log n) + O(n) + O(1)$, that is asymptotically equal to $O(n \log n)$. The dimensionality is only J for FADE and $K \times J$ for U-FADE. The number of frequency domain points K is a parameter and does not depend on the number of frames in the time domain. As a consequence, after fixing the number of joints, both FADE and U-FADE have $O(1)$ space complexity.

Algorithm 1 FADE Action Representation

1: $\boldsymbol{v}$ = FADE(ActionMatrix $\boldsymbol{A}_{\mathrm{t}}$)
2: $\boldsymbol{A}_f$ =FFT($\boldsymbol{A}_t$)
3: $\boldsymbol{A}_{\tilde{f}}$ =resample($\boldsymbol{A}_f, \Omega_K$)
4: $\boldsymbol{V}_{\tilde{f}}$ =PCA($\boldsymbol{A}_{\tilde{f}}$)
5: $\boldsymbol{v} = \boldsymbol{V}_{\tilde{f}}(:, 1)$ //select the first column
6: **return** $\boldsymbol{v}$

Algorithm 2 U-FADE Action Representation

1: $\boldsymbol{v}_U$ = UFADE(ActionMatrix $\boldsymbol{A}_{\mathrm{t}}$)
2: $\boldsymbol{A}_f$ = FFT($\boldsymbol{A}_{\mathrm{t}}$)
3: $\boldsymbol{A}_{\tilde{f}}$ =resample($\boldsymbol{A}_f, \Omega_K$)
4: $\boldsymbol{v}_U$ =reshape($\boldsymbol{A}_{\tilde{f}}, K \cdot J, 1$)
5: **return** $\boldsymbol{v}$

3.3 Parameter Tuning

In order to give an example on how K and f_{th} affect the performance of the recognition pipeline, in Fig. 3 we plot the accuracy of FADE, combined with 1-NN and Manhattan distance, evaluated on the whole HDM05 dataset split into 80 classes.. The accuracy is defined as the ratio between the number of test inputs correctly classified and the total number of test inputs. We estimated the accuracy with a 10-fold cross-validation method. It is possible to notice that $f_{th} = 5$ Hz, $f_{th} = 10$ Hz, and $f_{th} = 15$ Hz show similar performance. As expected, frequencies beyond 10 Hz do not add valuable information. Reducing the frequency threshold to 2 Hz, the accuracy decreases for every value of K. In terms of number of points K, the accuracy increases significantly

Table 1 Time and space complexity of FADE, U-FADE, and SVD.

Representation	Time Complexity	Space Complexity
FADE	$O(n \log n)$	$O(1)$
U-FADE	$O(n \log n)$	$O(1)$
SVD	$O(n^2)$	$O(1)$

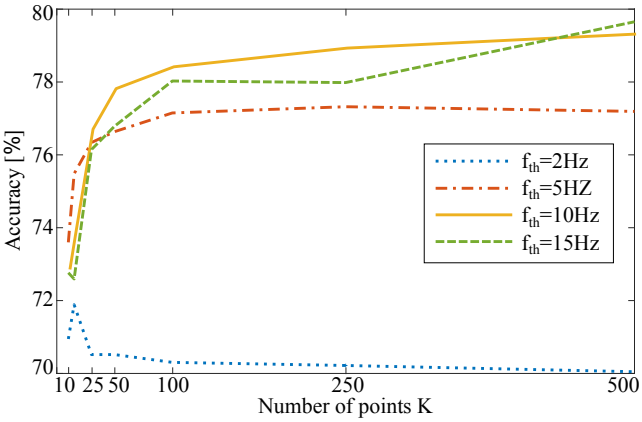


Fig. 3 Accuracy of FADE with 1-NN as a function of number of points K for different values of f_{th}

before a threshold. Increasing the number of points K after a certain threshold, the improvement of the accuracy becomes minor. For FADE, the threshold is around $K = 500$. The number of points affects the computational cost. When limited computational power is available, it can be important to choose a trade-off between accuracy and computational cost.

We performed the same experiment with U-FADE. In Fig. 4, the accuracy of U-FADE is shown on the whole HDM05 dataset as a function of K. It is evident that with the same value of K, increasing the parameter f_{th} to more than 10 Hz, we obtain slightly worse performance since we use the the same number of points to capture more frequencies. With frequency less than 10Hz, the accuracy starts decreasing, since we can loose information about human motion. Differently from FADE, U-FADE has a maximum for $K = 25$ and $f_{th} = 10$ Hz. When the number of points K is too small, we loose information. On the other hand, when $K > 30$, the size of the descriptor increases with no significant gain in information and in discrimination capability, also due to the curse of dimensionality. Hence, the accuracy slightly decreases and the increased size requires a higher computational cost for classification. Compared to FADE, U-FADE is less compressed and the size depends on the choice of the parameter K, while for FADE the size is constant with K. However, U-FADE performs well with a smaller value of the parameter K.

To give more detailed guidelines on the parameter choice, we evaluated the robustness of FADE and U-FADE to additive Gaussian noise for different values of K with $f_{th} = 10$ Hz. For each action of our test set, described by the matrix A_t, we add noise as

$$\hat{A}_t = A_t + \text{rand}(T, J; 0, \sigma^2), \tag{10}$$

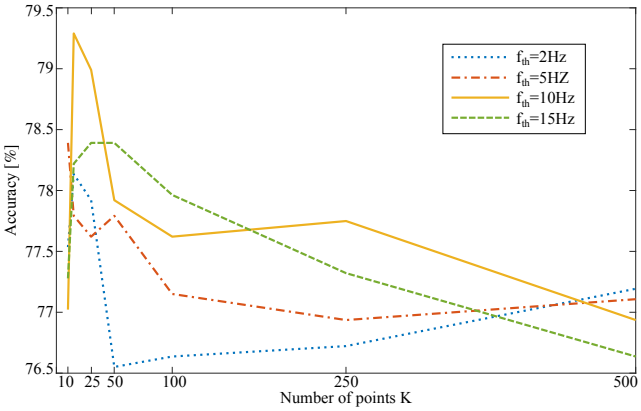


Fig. 4 Accuracy of U-FADE with 1-NN as a function of number of points K for different values of the frequency threshold f_{th}

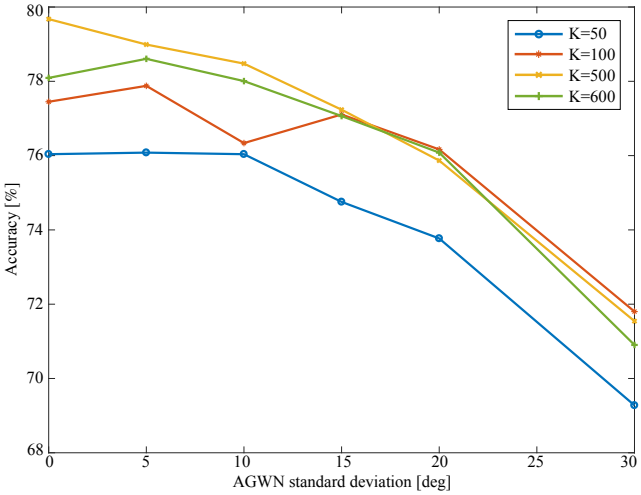


Fig. 5 Accuracy of FADE as a function of the standard deviation on the whole HDM05 using joint angles as representation and 1-NN to classify actions

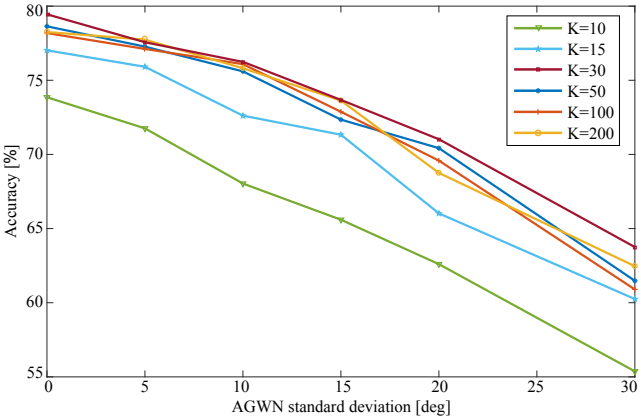


Fig. 6 Accuracy of U-FADE as a function of the standard deviation on the whole HDM05 using joint angles as representation and 1-NN to classify actions

where rand(T, J) is a function which returns a $T \times J$ matrix of real numbers sampled from a Gaussian stochastic process with 0-mean, variance σ^2, and impulsive autocorrelation, i.e., Additive Gaussian White Noise (AGWN). In Figures 5 and 6, the accuracy of FADE and U-FADE as a function of the standard deviation is reported, considering different values of K. The standard deviation range is $[0, 30]$deg. Concerning FADE, with a standard deviation $\sigma = 10$ deg the accuracy reduces by 1% with $K = 500$ and it remains practically constant for $K = 50$ and $K = 600$. With $\sigma = 10$ deg, we can see that the accuracy starts decreasing for each tested value of K. We can see that also with a very high, unrealistic noise standard deviation of 30 deg the accuracy on the whole data set, using FADE, decreases by 6% for all K values. Approximately, we loose 0.25% accuracy for standard deviation unit. As a consequence, the choice $K = 500$ shows also a good robustness to

noise. Concerning U-FADE, we approximatively loose 0.5% accuracy for each degree of noise standard deviation and for all values of K. In this work, we choose $K = 25$ since this value shows good accuracy and robustness to noise. Both FADE and U-FADE show excellent noise robustness. However, FADE shows better performance than U-FADE in terms of robustness to very high noise levels. Concerning high-frequency noise, FADE and U-FADE are intrinsically robust since they do not use frequencies above 10 Hz.

3.4 Discussion on FADE and U-FADE

It is of interest to underline the main differences between FADE and U-FADE. In terms of time and space complexity, they both have $O(n \log n)$ and $O(1)$, respectively. As it can be noticed in Algorithms 1 and 2, the difference between the two descriptors is in line 4: in FADE we compute the PCA of the matrix $\boldsymbol{A}_{\tilde{f}}$ (Algorithm 1, line 4) and extract only the first eigenvector (line 5). In line 4 of U-FADE algorithm, instead, we rearrange the matrix $\boldsymbol{A}_{\tilde{f}}$ into a single vector to obtain the descriptor. Even though the big-O time complexity is the same as U-FADE, computing the PCA in FADE algorithm is more computationally expensive than rearranging $\boldsymbol{A}_{\tilde{f}}$ into a vector in U-FADE. Therefore, in terms of encoding time, FADE is slower than U-FADE. In terms of memory requirement, FADE performs significantly better than U-FADE. In fact, FADE has J elements while U-FADE has $K \cdot J$ elements. In Sec. 3.3, we showed that a suitable choice is $K = 25$, which means that U-FADE typically needs 25 times more memory than FADE. For FADE, the best value of this parameter is $K = 500$ in our tests. Choosing a high value of K in FADE does not increase the required memory but only the encoding time. Once the encoding operation is completed, U-FADE needs more time for classification. In fact, the classification time of most state of the art learning algorithms increases with the dimension of the descriptor, i.e., the number of features. In terms of informativeness, U-FADE provides no loss of information for frequencies minor than 10 Hz and allows distinguishing the contribution of single joints. With FADE, distinguishing the contribution of single joints is not straightforward, because the vector $\boldsymbol{v}$ is a linear combination of the columns of $\boldsymbol{A}_{\tilde{f}}$.

4 Experimental Result

In this section, we present the experimental results to show the performance of FADE and U-FADE evaluated with different public datasets, namely HMD05[1], MHAD (Ofli et al, 2013), and MSR Action3D (Li et al, 2010). FADE and U-FADE are compared with well-known approaches to action recognition. Moreover, experiments have been performed to test the robustness of FADE and U-FADE in presence of noisy signals. The classification algorithms we tested with FADE and U-FADE are 1-Nearest Neighbor (1-NN) and Support Vector Machine (SVM). These are very simple and well-known classifiers. Even though more complex classification algorithms could be adopted, we intend to keep the classification module as simple as possible and to achieve competitive performance due to a smart descriptor selection. The main features of the adopted datasets are described below, and are summarized in Table 2.

Table 2 Datasets characteristics.

Dataset	Subjects	Classes	Sequences	f_s[Hz]
HDM05	5	80	2337	120
R-HDM05	5	16	401	120
MHAD	12	11	659	480
MSR	10	20	567	15

HDM05 Dataset

The HDM05 Motion Capture Database is a freely available dataset of human whole-body actions. It contains 80 action classes with 10 - 50 action sequences per class, performed by 5 different actors. In total, the dataset has 2337 action sequences and the frame rate is 120 Hz. The database provides motion data in format of both joint 3D positions and joint angles. In order to prove the generality of our approach, we tested the descriptors with both joint angle-based data and joint position-based data.

Berkeley MHAD Dataset

The Berkeley MHAD (Multimodal Human Action Data set) is constituted by 11 action classes performed by 12 subjects. All the subjects performed 5 repetitions of each action class, obtaining 659 action sequences in total. The 3D positions of active LED markers are measured with an optical motion capture system. The sampling frequency for this dataset is $f_s = 480$ Hz. The number of joints is $n_{joints} = 35$ and for each frame $J = 35 \times 3$ Cartesian joint trajectories can be extracted.

[1] Müller M, Röder T, Clausen M, Eberhardt B, Krüger B, Weber A (2007) Documentation mocap database hdm05

MSR Action3D Dataset

The MSR Action3D dataset is constituted by 20 action classes performed by 10 subjects, and captured with a RGB-D Kinect camera. Each subject repeated each action 2 or 3 times, for a total of 567 action sequences. The sampling rate f_s of this dataset is 15 Hz, and the Cartesian positions of 20 skeleton joints are captured. This dataset is characterized by a low sampling rate, low accuracy of the joint Cartesian positions, and by the presence of noise and missing frames.

4.1 Cross-validation on the HDM05, MHAD, and MSR datasets

First, we tested FADE and U-FADE on the HDM05, MSR, MHAD datasets coss-validating the results with a classical 10-fold method. This test is, in our opinion, very important to assess the performance of our descriptors. In fact, a test on the whole dataset does not allow us to remove lower-quality actions or to select particular subsets which are more convenient. In the literature, there are only few papers that evaluate the results on these entire public available action sets. In particular, (Chen and Koskela, 2015) and (Cho and Chen, 2014) are frame-by-frame classification approaches tested on the whole HDM05 dataset with 40 and 65 action classes respectively. Recently, in (Du et al, 2015) the entire HDM05 is tested with 65 action classes in order to be directly compared with (Cho and Chen, 2014). Both methods are based on deep learning and achieve an excellent accuracy paying the price of having a more complex classification procedure. In Table 3, the performance of FADE and U-FADE are reported in terms of accuracy, training time, and one-action testing time. The training time is given by the time to compute the descriptors for all the actions and the time to train the classifier. The actions are encoded with a MATLAB script running on a quadcore 2.4 GHz CPU. The SVM classifier is trained with a Python script.

On the HDM05 dataset with 790 actions grouped into 40 classes (Chen's HDM05 subset), FADE reaches 95.0% accuracy with SVM and U-FADE 95.6%. The approach presented in (Chen and Koskela, 2015) achieves 96.5% accuracy with SVM and 95.8% with ELM. In terms of accuracy and using the same classification algorithm, Chen's features preform slightly better than FADE and U-FADE. However, in terms of training time FADE performs much better since it requires 1.3 s to encode the entire HDM05 dataset and train the SVM classifier, while the approach in (Chen and Koskela, 2015) requires about 21 minutes. In terms of testing time, i.e. the time to evaluate one action, FADE and U-FADE with SVM take 0.58 ms and 0.74 ms respectively. The approach in (Chen and Koskela, 2015) with SVM takes 2500 ms in C++, and the same approach with Extreme Learning Machine (ELM) takes 34 ms with a C++ implementation. On the HDM05 dataset split in 65 action classes, we compared our descriptors with (Cho and Chen, 2014) and (Du et al, 2015). The first is a frame-by-frame classification approach which uses a deep neural network to obtain a final decision on the sequence. The second uses a recurrent neural network. The first reaches 95.6% accuracy while the second achieves 96.5% accuracy. FADE combined with a SVM achieves 92.0% accuracy with a training time of 8.9 s. U-FADE with SVM achieves 90.2% accuracy with a training time of 6.3 s. Using a compiled language such as C++ instead of Python, we expect much faster results.

Concerning MSR, we have not found any approach that tests the full dataset with a cross-validation approach. The results of the 10-fold cross-validation on the entire MSR are 90.1% with U-FADE combined with 1-NN and 95.5% with SVM. FADE reaches 90.0% accuracy with 1-NN and 90.3% with SVM. In general, for small-size datasets, U-FADE performs better than FADE. For larger datasets, the ratio between computational efficiency and computational cost is higher in FADE. In terms of a rough estimation of the computational cost, we measured a time of 1.7 s to encode all the actions and train a SVM classifier on the whole MSR dataset with FADE. With U-FADE, it takes 0.17 s.

Experiments on the whole MHAD are in (Chen and Koskela, 2015) and (Ofli et al, 2013). In particular, Chen's appraoch reaches 99.6% accuracy and Ofli's approach reaches 74.7% accuracy. U-FADE with SVM presents an accuracy of 99.3% with an encoding and training time of 7.1 s. In particular, 7.0 s are for encoding all the actions and 0.1 s is for training the SVM. In (Chen and Koskela, 2015) and (Ofli et al, 2013) there is no discussion concerning the training time. We expect higher computational cost in Chen's approach due to the frame-by-frame classification method and similar performance as FADE in Ofli's approach. The latter, in fact, adopts very simple and compressed features based on properties of human motion, instead of using a complex classification algorithm. This is the same philosophy we adopt with FADE and U-FADE. Concerning the computational time of the approach discussed in (Du et al, 2015), in the discussion section the authors report that the training time for the entire MHAD is about 50 s per epoch. To reach an accuracy higher than 98%, the RNN required 30 epochs. The estimated training time for the entire MHAD is then about 300 s with a C++ implementation. From this rough consideration on computational cost, we can claim that FADE and U-FADE

Table 3 Accuracy, training time, and one-action testing time of FADE and U-FADE on different complete datasets compared with well-known state of the art approaches

Method	Dataset	Classes	Actions	Acc. (%)	Train time[s]	Test time[ms]
FADE+SVM	HDM05	40	790	95.0	2.6	**0.58**
U-FADE+SVM	HDM05	40	790	95.6	**1.3**	0.74
(Chen and Koskela, 2015) + SVM	HDM05	40	790	96.5	1300 (C++)	2500 (C++)
(Chen and Koskela, 2015) + ELM	HDM05	40	790	95.8	31 (C++)	4.7 (C++)
FADE + SVM	HDM05	65	2337	92.0	8.9	0.57
U-FADE +SVM	HDM05	65	2337	90.2	6.3	0.76
HBRNN-L (Du et al, 2015)	HDM05	65	2337	**96.6**	-	-
(Cho and Chen, 2014)	HDM05	65	2337	95.6	-	-
FADE + 1NN	MSR	20	567	90.0	1.7	0.15
FADE + SVM	MSR	20	567	90.3	1.7	0.58
U-FADE + 1-NN	MSR	20	567	90.1	0.17	1.8
U-FADE + SVM	MSR	20	567	**93.5**	0.17	0.58
FADE + 1NN	MHAD	11	659	98.3	10.5	0.15
FADE + SVM	MHAD	11	659	98.6	10.6	0.58
U-FADE + 1-NN	MHAD	11	659	98.9	7.0	1.8
U-FADE + SVM	MHAD	11	659	99.3	7.1	0.74
(Chen and Koskela, 2015) + ELM	MHAD	11	659	**99.6**	-	-
(Ofli et al, 2013) + 1-NN	MHAD	11	659	74.8	-	-
(Ofli et al, 2013) + SVM	MHAD	11	659	79.9	-	-

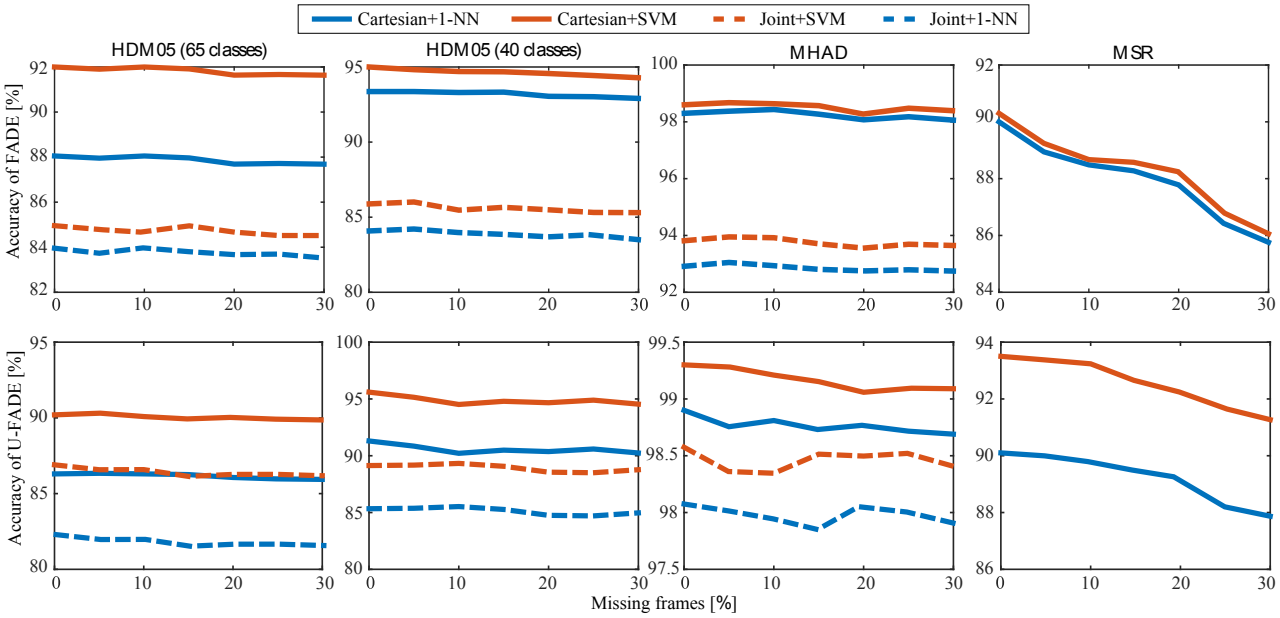


Fig. 7 Accuracy of FADE (upper row) and U-FADE (lower row), averaged on 10 executions, in presence of missing frames with all the considered datasets

are particularly suitable in applications where the classifier needs to be trained frequently. The cost to pay with respect to complex classification algorithms based on deep learning is a slight reduction (from a minimum of 0.2% to a maximum of 5%) of the accuracy.

4.2 Robustness to noise and missing frames

The robustness to FADE and U-FADE to missing frames is shown in Fig. 7 both for Cartesian and joint angle data. Missing frames are created by removing frames at random time instants. This test represents the performance of FADE and U-FADE in presence of sensors or communication channel with heavy data loss. It is possible to notice that for HDM05 and MHAD the accuracy remains practically constant even with 30% missing frames. In the MSR, FADE loses 4% accuracy when 30% of the frames are missing, while U-FADE loses only 2% accuracy with both 1-NN and SVM. Our guess is that the robustness of MSR is slightly lower with respect to HHD05 and MHAD cases because for

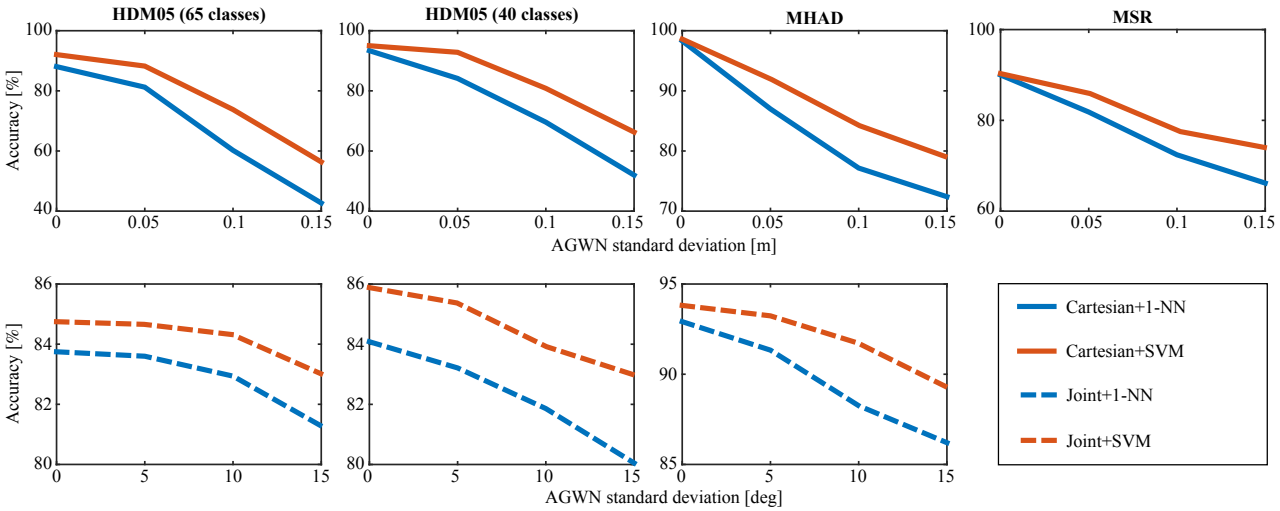


Fig. 8 Accuracy of FADE, averaged on 10 executions, in presence of AGWN with all the considered datasets

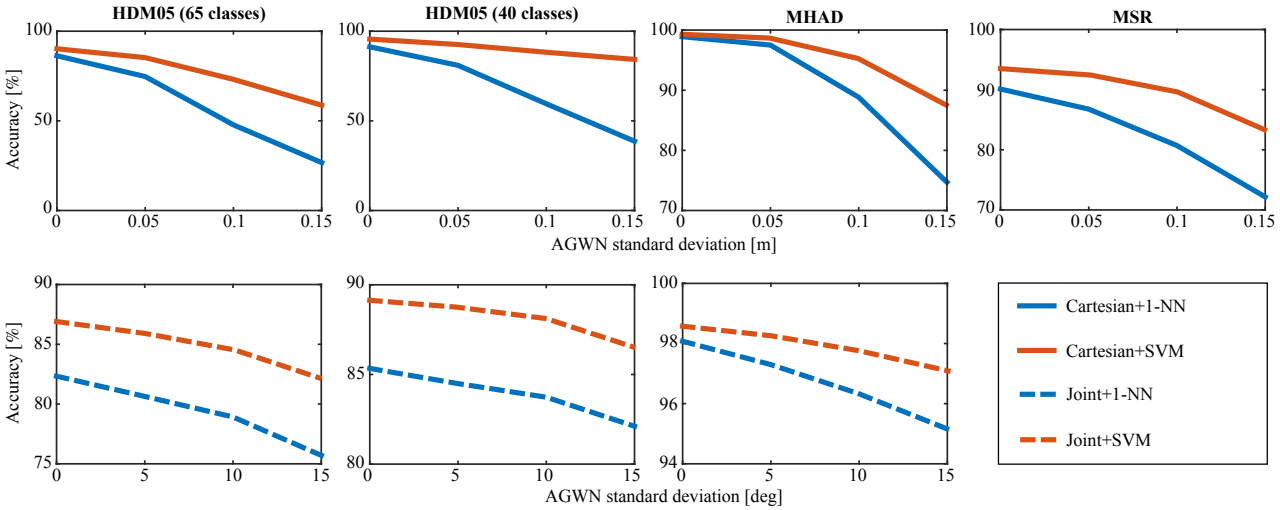


Fig. 9 Accuracy of U-FADE, averaged on 10 executions, in presence of AGWN with all the considered datasets

MSR the sampling rate is rather low(15 Hz), as reported in Table 2. We can conclude from this analysis that both FADE and U-FADE are very robust to missing frames and, hence, they can be used successfully with low-quality sensors or communication channels. Moreover, we notice that U-FADE is slightly more robust to missing frames than FADE.

Figures 8 and 9 report the robustness to noise for all the datasets. We added additive gaussian white noise to the signals with different standard deviations, ranging from 0 cm to 15 cm in case of Cartesian joint positions and from 0 deg to 15 deg in the case of joint angles. It is worth emphasizing that the noise is added to the original acquisition noise, which already corrupts the signals. We can notice that, with HDM05 and Cartesian joint positions, and a noise standard deviation of 5 cm, the accuracy drop is 3% using SVM and 4% using 1-NN as a classification method. With the MHAD dataset and the MSR dataset, the accuracy loss is around 5% for 5 cm standard deviation when using the SVM classifier and 10% with 1-NN classification. In case of U-FADE, the robustness to noise is slightly better with the HDM05 dataset. However, with MHAD and MSR the accuracy is significantly higher with respect to FADE, especially using the SVM classifier.

For sake of speculation, we tested also unreasonable standard deviations of 10 cm and 15 cm. We see that even in those extreme cases, FADE and U-FADE with SVM do not lose more than 30% accuracy. From this experiments we conclude that both FADE and U-FADE show good robustness to noise and missing frames. Combining FADE and U-FADE with SVM gives better performance than 1-NN not only in terms of accuracy, but also in terms of robustness to noise and missing frames.

In the case of joint angles, we applied AGWN with standard deviation between 0 deg and 15 deg. We see

the same trend in the performance as in the Cartesian position case. However, a maximum standard deviation of 15 deg is more realistic than 15 cm. Hence, the maximum drop in performance is much lower than in the previous case. In fact, the drop of accuracy with 15 deg is between 2% and 5% also in the worse cases.

4.3 Experiments with HDM05

The HDM05 contains two types of data: joint angles and joint Cartesian positions. In the literature, the joint Cartesian positions seem to perform in general better. Also, methods that work with joint angles, such as (Ofli et al, 2014), adopt as input joint Cartesian position and then compute joint angles. We evaluated the performance of FADE and U-FADE with both joint angles and joint Cartesian positions with both supervised and unsupervised learning approaches. In order to show the performance of our approach on datasets of different sizes, it was evaluated on distinct sets:

1. Joint angle-based data
   - *Cho's HDM05 dataset* We used all the actions and the classes in the HDM05 dataset without removing lower quality actions. The actions were performed by five different actors. The dataset consists of 2337 actions split in 65 classes according to the class distribution adopted in (Cho and Chen, 2014).
2. Joint position-based data
   - *Cho's HDM05 dataset* 2337 actions of HDM005 split in 65 classes according to the class distribution adopted in (Cho and Chen, 2014).
   - *Chen's HDM05 dataset* 790 actions of HDM005 split in 40 classes according to the class distribution adopted in (Chen and Koskela, 2015).
   - *R-HDM05* It is a subset of HDM05 composed by 16 action classes. We consider this action subset to compare our approach with diverse approaches such as (Ofli et al, 2014), and (Bissacco et al, 2001).
   - *R2-HDM05* It is a subset of HDM05 composed by 11 action classes, used to compare our approach with Evangelidis et al (2014)

For each dataset, the performance of FADE and U-FADE is evaluated with respect to supervised and unsupervised methods. A first performance evaluation of FADE and U-FADE on HDM05 joint angle data is reported in (Shah et al, 2016). The baseline of our comparison for action classification is the Singular Value Decomposition approach in (Cavallo and Falco, 2014). This approach is used for online classification of manipulation actions with force and position data. However, the SVD compression can be applied also as a descriptor for full-body actions. It is interesting comparing the frequency domain compression performed in FADE with a similar compression method performed directly in the time domain. It is possible to notice how FADE is more data-efficient, has a lower time complexity, and higher accuracy.

4.3.1 Joint-Angle Data

The results on the HDM05, with Chen's class organization, are shown in Table 4. This dataset challenges the scalability of action representation and classification methods, since it contains a large number of action classes and action sequences. As shown in Table 4, FADE with 1-NN obtains a recognition rate of 83.7% and with SVM it achieves an accuracy of almost 84.7%. Considering its simplicity and compression rate, FADE shows good properties of scalability with supervised learning approaches. The results obtained with unsupervised learning are reported in Table 5. We evaluated the performance of FADE and U-FADE combined with K-means (KM), Spectral Clustering (SC), and Agglomerative Clustering (AC) (Bishop, 2006). The accuracy of unsupervised approaches is computed with the Clustering Accuracy (CA) metrics in (Xu et al, 2003). Given an action A_i, let α_i be the cluster label estimated by the clustering algorithm and let λ_i be the label provided by the motion dataset, i.e., the true label. The clustering accuracy CA (Xu et al, 2003) is defined as

$$\mathrm{CA} = \frac{\sum_{i=1}^{N_a} \delta(l_i, \mathrm{map}(\lambda_i))}{N_a}, \tag{11}$$

where N_a is the number of actions in the test set, and $\delta(x, y)$ is the Kronecker delta function. It is equal to one if $x = y$, and it equals zero otherwise. The function $\mathrm{map}(\lambda_i)$ maps each cluster label λ_i to the equivalent label from the motion dataset. Such a function can be implemented by using the Kuhn-Munkres algorithm (Lovász and Plummer, 2009). Unsupervised approaches are less scalable and suffer more with high compression rate. With FADE, both KM and SC achieve 37% accuracy. U-FADE combined with SC performs almost 10% better than FADE with SC, getting 47% accuracy. With k-means U-FADE reaches 40% accuracy, while U-FADE combined with AC shows 39% accuracy.

4.3.2 Joint Cartesian-position Data

In order to make the data invariant to the world frame, we adopted a coordinate transformation process. This procedure allows us to observe the joint Cartesian trajectories in a frame fixed to the torso. Additionally, it

Table 4 Classification Accuracy on HDM05 using joint positions (P. Acc.) and joint angles (A. Acc.)

Method	Classif.	P. Acc.(%)	A. Acc.(%)
FADE	1-NN	88.0	83.7
FADE	SVM	**92.0**	84.7
U-FADE	1-NN	86.3	82.3
U-FADE	SVM	90.2	**86.9**
SVD	1-NN	87.1	77.8
SVD	SVM	88.0	77.9

Table 5 Clustering accuracy on HDM05 using joint positions (P. CA) and joint angles (A. CA)

Method	Clustering	P. CA(%)	A. CA (%)
FADE	SC	43.6	37.5
FADE	KM	43.4	37.0
FADE	AC	44.3	36.2
U-FADE	SC	**51.2**	**46.6**
U-FADE	KM	42.4	40.4
U-FADE	AC	45.9	39.2
SVD	SC	38.9	34.0
SVD	KM	39.6	33.3
SVD	AC	39.9	34.4

is possible to compute the skeleton model normalization according to the procedure presented in (Chen and Koskela, 2015), which normalizes the skeletal model of the subject such that the lengths of all the bones sum up to 1.

4.3.3 Whole HDM05 Action Set

The results are reported in Table 4. Using FADE with 1-NN results in an accuracy of 89.7%. With SVM, it increases to 92.0%. For unsupervised learning, U-FADE with SC gets an accuracy of 51.22%. For big datasets, the performance of unsupervised methods can appear rather weak. However, unsupervised methods have a great potential for robotic applications. Such methods, indeed, can allow robots to observe and cluster human actions without any human contribution.

4.3.4 R-HDM05 Action Set

After evaluating FADE and U-FADE on HDM05, we show the performance on the reduced set, which we called R-HDM05. This subset is constituted by 401 action sequences grouped into the 16 action classes: *depositFloorR* (1), *elbowToKnee1RepsLelbowStart* (2), *grabHighR* (3), *hopBothLegs1hops* (4), *jogOnPlaceStartAir2StepsLStart* (5), *jumpDown* (6), *jumpingJack1Rep* (7), *kickLFront1Reps* (8), *lieDownFloor* (9), *rotateArmsBothBackward1Reps* (10), *sitDown Chair* (11), *sneak2 Steps LStart* (12), *squat1Reps* (13), *standUpKneelToStand* (14), *throwBasketball* (15), *throwFarR* (16). With this dataset,

Table 6 The best classification results for different action descriptors obtained for the R-HDM05 dataset.

Descriptor	Accuracy (%)
FADE + 1-NN	93.3
FADE + SVM	91.2
U-FADE + 1-NN	**97.0**
U-FADE + SVM	94.9
SMIJ (Ofli et al, 2014) + 1-NN	91.5
SMIJ (Ofli et al, 2014) + SVM	89.2
HMIJ (Ofli et al, 2014) + 1-NN	73.5
HMIJ (Ofli et al, 2014) + SVM	78.5
HMW (Ofli et al, 2014) + 1-NN	77.4
HMW (Ofli et al, 2014) + SVM	79.4
LDSP (Bissacco et al, 2001) + 1-NN	67.8
LDSP (Bissacco et al, 2001) + SVM	70.6

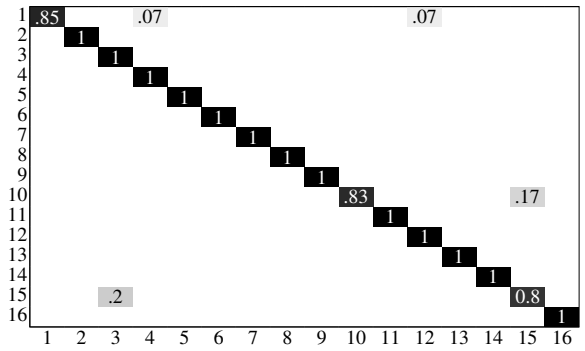


Fig. 10 Confusion matrix of U-FADE with R-HDM05.

U-FADE achieves the best performance with 97.0% accuracy. The accuracy of FADE is 93.2%. We compared the performance of the proposed descriptors with Sequence of Most Informative Joints (SMIJ), Histogram of Most Informative Joints (HMIJ), Histogram of Motion Words (HMW), proposed in (Ofli et al, 2014). Also, we compared the performance also with Linear dynamical system parameters (LDSP), which is a classical method to analyze human motion. SMIJ gets 91.5% accuracy with 1-NN and 89.2% with SVM.

The confusion matrix of U-FADE with SVM is shown in Fig. 10. U-FADE gets 100% recognition accuracy with all the actions except *depositFloorR* (1), *rotateArmsBothBackward1Reps* (10), and *throwBasketball* (15). Concerning the action *depositFloorR*, U-FADE shows 85% accuracy since it is confused with a rate of 7% with the action *hopBothLegs1hops* (4) and in 7% of the cases with the action *sneak 2StepsLStart* (12). The action *rotateArmsBothBackward1Reps* (10) achieves 83% accuracy and in 17% of the cases is confused with *throwBasketball* (15). Finally, in 20% of cases, *throwBasketball* is confused with *grabHighR* (3). Concerning unsupervised learning, FADE reaches a clustering accuracy of 63.9% with KM, 63.0% with AC, and 17.4% with SC. U-FADE achieves a cluster accuracy of 62.7%, 51.9%, and 68.7% with KM, AC, and SC respectively. On this

Table 7 The best classification results for different action descriptors obtained for the R2-HDM05 dataset.

Descriptor	Accuracy (%)
FADE + 1-NN	94.9
FADE + SVM	96.9
U-FADE + 1-NN	**97.6**
U-FADE + SVM	96.4
(Evangelidis et al, 2014)	93.9

reduced dataset, the performance of clustering methods increases significantly, even though still inferior to the performance of supervised approaches.

4.3.5 R2-HDM05 Action Set

This subset is used by the skeletal quad approach in (Evangelidis et al, 2014). This is an interesting approach for our work because, like FADE and U-FADE, it aims at finding simple features for time-efficient and data-efficient classification. R2 is a subset of R-HDM05. In particular, the subset is constituted by the actions {(1), (2), (3), (4), (5), (8), (9), (10), (12), (13), (15)}. Table 7 reports the results in terms of accuracy of FADE, U-FADE, and the approach in (Evangelidis et al, 2014). U-FADE reaches 97.6% accuracy with 1-NN e 96.4 accuracy with SVM. The skeletal quad approach reaches 93.9% on this dataset. FADE shows 94.9% with 1-NN and 96.6% with SVM. Hence, on this dataset, both FADE and U-FADE performs better in terms of accuracy. With clustering approaches, FADE gets an accuracy of 69.1%, 68.6%, and 17.4% with KM, AC, and SC respectively. U-FADE gets 64.1%, 53.8%, and 68.0% with KM, AC, and SC respectively.

4.4 MSR Action3D Dataset

In order to have more comparisons between our method and other approaches in the literature, we adopt the test protocol proposed in (Li et al, 2010). In this protocol, the action classes of the MSR dataset are split into 3 Action Sets (AS) with 8 classes: AS1, AS2, and AS3. For each action set, three tests (T1, T2, and T3) are performed. In the first test case, T1, one demonstration of each user is used for training and the other two demonstration for testing. In T2, two demonstrations of each user are used for training and one demonstration for testing. In T3, all action sequences performed by half of the users are used for training and the rest for testing. T3 is then a cross-subject test, while in T1 and T2 data from all the subjects are considered for the training. The actions of the database are split in action sets in the following way:

- AS1: *horizontalArmWave (1), hammer (2), forwardPunch (3), highThrow (4), handClap (5), bend (6), tennisServe (7), pickUpThrow (8)*
- AS2: *highArmWave (1), handCatch (2), drawX (3), drawTick (4), drawCircle (5), twoHandWave (6), sideBoxing (7) ,forwardKick (8)*
- AS3: *highThrow (1), forwardKick (2), sideKick (3), jogging (4), tennisSwing (5), tennisServe (6), golfSwing (7) pickUpThrow (8)*

Using these action sets and action tests, we can compare our method with diverse approaches presented in the literature. Most approaches, such as (Gowayyed et al, 2013), (Vemulapalli et al, 2014), (Evangelidis et al, 2014), (Ofli et al, 2014), implemented only a part of the proposed tests. Other approaches like (Xia et al, 2012), (Li et al, 2010), (Chen et al, 2013) are tested adopting the complete protocol. Comparing the experimental results in Table 8, there is no ultimate method that outperforms all the others on this dataset. On this dataset, U-FADE performs slightly better than FADE. This happens in general for small-sized datasets.

In Figures 11(a), 11(b), 11(c), the confusion matrices of U-FADE with SVM are reported relatively to Action Set 1 (AS1). On AS1, U-FADE combined with SVM achieves 94.7% on T1, 100% on T2 and 92.4% on T3. On this action set, (Xia et al, 2012) performs better than (Li et al, 2010), (Chen et al, 2013) and U-FADE on T1 with 92% accuracy. On the test T2, UFADE reaches 100% accuracy and performing better than all the other methods, while on T3 U-FADE reaches 92.7% accuracy. On the test T3, it is possible to make comparisons with more methods. In particular, U-FADE performs better than (Xia et al, 2012), (Li et al, 2010), (Evangelidis et al, 2014), (Gowayyed et al, 2013). Also, (Du et al, 2015) performs slightly better than U-FADE (0.5% better), while (Vemulapalli et al, 2014) achieves 95.3% accuracy (2.5% better) and (Chen et al, 2013) achieve 96.2% accuracy, which is the best accuracy on T3 for the first Action Set.

The confusion matrices relative to AS2 are shown in Figures 11(d), 11(e), 11(f). On the AS2, U-FADE with SVM gets 82.0% in the test T1. Compared to (Xia et al, 2012), (Chen et al, 2013), and (Li et al, 2010), U-FADE shows worse performance in this particular case. To investigate in more detail the reason, we can consider the confusion matrix reported in Fig. 11(d). As it can be seen from the confusion matrix in Fig. 11(d), the actions that mostly contribute to drop the accuracy are *handCatch* (2) with 65% accuracy, *drawTick* (4) with 70% accuracy, and *sideBoxing* (7), with 71% accuracy. *handCatch* (2) is confused especially with *drawX* (3) in 10% of cases and with *sideBoxing* (7) in 12% of cases. In 6% of cases, (2) is

confused with *drawTick* (4), and emphdrawCircle (5). *drawTick* (4) is confused with *highArmWave*(1) in 15% of cases, and with *drawCircle*, *sideBoxing* and *drawX* in 5% of cases. The action *sideBoxing* (7), is confused with *drawCircle* in 24% of cases. From analyzing the confusion matrix we can conclude that the actions *sideBoxing*, *drawCircle*, *drawX*, *drawTick* have a rather similar spectrum and in some cases are confused. A plausible explanation is that using only the spectrum, we loose information on the directionality of the motion. This way, actions like stand-up and sit-down are encoded with similar descriptors. In applications where the directionality of the motion is important, we can include the mean velocities of the joints in the descriptor, paying the price of a slightly higher computational cost, but without increasing the $O(.)$ time and space complexity. On the same action set and on Test 2, U-FADE performs slightly better than (Li et al, 2010) and it performs worse than (Chen et al, 2013) and (Xia et al, 2012). However, on test T3, U-FADE performs better than most approaches such as (Xia et al, 2012), (Li et al, 2010), (Chen et al, 2013), (Gowayyed et al, 2013), (Vemulapalli et al, 2014), (Evangelidis et al, 2014). The only method which in this case perform better is (Du et al, 2015).

Concerning the Action Set 3, U-FADE achieves 97.5% accuracy on T1, 98.9% accuracy on T2 and 91.9% on T3. The confusion matrices relative to this action set are reported in Figures 11(g), 11(h), 11(i). On T1 and T2, U-FADE performs better than (Li et al, 2010) and (Xia et al, 2012), but it is slightly outperformed by (Chen et al, 2013). On T3, U-FADE has the same performance as (Chen et al, 2013), but it performs worse than (Evangelidis et al, 2014), (Du et al, 2015) and (Vemulapalli et al, 2014). On the AS3, in (Ofli et al, 2014), an accuracy of 47.1% is reported. It is also interesting to compare U-FADE with the frequency-based approach proposed in (Wang et al, 2012). The average value for this approach is 88.1% using the proposed classification method based actionlet ensembles, while U-FADE reaches 91.6% with a standard SVM classifier.

From these experiments we can conclude that U-FADE performs better than other simple and data-efficient descriptors, and also in terms of accuracy is competitive with descriptors that adopt more complex classifiers and features. The unsupervised approaches, i.e., KM, AC, and SC show accuracy of 52.2%, 32.7%, and 25.4%, respectively, for FADE; they show and accuracy of 45.8%, 11.7%, and 46.0, respectively, for U-FADE.

4.5 Berkeley MHAD Dataset

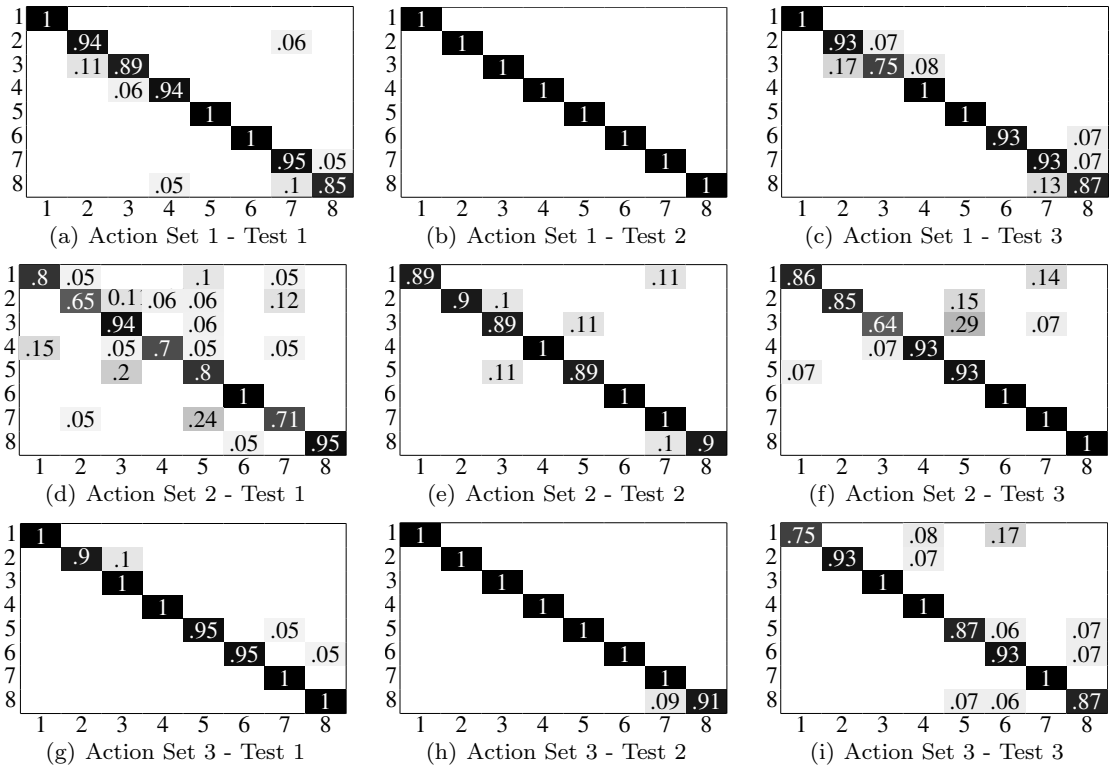
The comparison between FADE, U-FADE, and diverse state of the approaches on the classes of the MHAD database is reported in Table 9. In this case, FADE achieves 91.6% accuracy, U-FADE achieves 93.8%. The RNN-based method proposed in (Du et al, 2015) reaches 100% accuracy on this test set, while SMIJ with 95.1%. In this experiment, 7 subjects are chosen for training (384 action sequences) and 5 (275 action sequences) for testing, according to the cross-subject validation protocol adopted in (Ofli et al, 2014). MHAD is constituted by 11 action classes: *jumping* (1), *jumping jacks* (2), *bending* (3), *punching* (4), *waving two hands* (5), *waving one hand* (6), *clapping* (7), *throwing* (8), *sit down* (9), *stand up* (10), *sit down/stand up* (11). The confusion matrix is shown in Fig. 12. For most action classes, the accuracy of FADE is 100%, except for actions *jumping jacks* (2), *bending* (3), *punching* (4), *sit down/stand up* (11). The most critical action is *jumping jacks* (2), which achieves 64% accuracy. In 24% of cases is confused with *throwing* (8), while in 12% of cases is confused with *waving two hands* (5). The action *throwing* (8) is confused with a 4% rate with *waving two hands* (5) and *clapping* (7), and in 8% of the cases with *waving one hand* (6). The action *sit down/stand up* (11) is confused with *stand up* (10) in 12% of the cases. On this dataset U-FADE performs slightly worse than SMIJ and it performs better than MHIJ, HMW, and LDSP. Also, U-FADE is less accurate than approach proposed in (Du et al, 2015), which is based on hierarchical RNN. Nevertheless, this approach uses a more complex classifier and requires higher computational power. With the whole dataset, FADE with KM reaches 32.2% accuracy, 57.0% accuracy with AC, and 25.4% with SC. U-FADE achieves an accuracy of 56.0%, 43.7%, and 59.5% with KM, AC, and SC respectively.

5 Conclusion and Future Work

In this paper, we presented Frequency-based Action Descriptor (FADE) and Uncompressed FADE (U-FADE). We tested the performance of FADE and U-FADE with three publicly available datasets, i.e. HDM05, MHAD, and MSR Action3D. Due to its strong compression, FADE is suitable for classifying several actions with very reduced training and test times. U-FADE is less compressed but it performs better than FADE on small-sized actions sets and with some unsupervised clustering algorithm such as Spectral Clustering. Despite their simplicity, FADE and U-FADE present an accuracy comparable with most existing approaches that

Table 8 Recognition results on the MSR Action3D dataset.

	Test 1				Test 2				Test 3			
	AS1	AS2	AS3	Average	AS1	AS2	AS3	Average	AS1	AS2	AS3	Average
U-FADE + SVM	94.7%	82.0%	97.5%	91.4%	**100%**	93.3%	98.9%	97.4%	92.7%	90.2%	91.9%	91.6%
(Xia et al, 2012)	**98.47%**	**96.67%**	93.47%	96.2%	98.61%	97.92%	94.93%	97.15%	87.98%	85.40%	63.46%	78.97%
(Li et al, 2010)	89.5%	89.0%	96.3%	91.6%	93.4%	92.9%	96.3%	94.2%	72.9%	71.9%	79.2%	74.7%
(Chen et al, 2013)	97.3%	96.1%	**98.7%**	**97.4%**	98.6%	**98.7%**	**100%**	**99.1%**	**96.2%**	83.2%	92.0%	90.47%
(Gowayyed et al, 2013)	-	-	-	-	-	-	-	-	92.39%	90.18%	91.43%	91.26%
(Vemulapalli et al, 2014)	-	-	-	-	-	-	-	-	95.29%	83.87%	**98.22%**	92.46
(Du et al, 2015)	-	-	-	-	-	-	-	-	93.33%	**94.64%**	95.50%	**94.49%**
(Evangelidis et al, 2014)	-	-	-	-	-	-	-	-	88.4%	86.6%	94.5%	89.8%
(Wang et al, 2012)	-	-	-	-	-	-	-	-	-	-	-	88.2%
(Ofli et al, 2014)	-	-	-	-	-	-	-	-	-	-	-	47.1%



(a) Action Set 1 - Test 1 (b) Action Set 1 - Test 2 (c) Action Set 1 - Test 3

(d) Action Set 2 - Test 1 (e) Action Set 2 - Test 2 (f) Action Set 2 - Test 3

(g) Action Set 3 - Test 1 (h) Action Set 3 - Test 2 (i) Action Set 3 - Test 3

Fig. 11 Confusion matrices for U-FADE with SVM evaluated on the MSR Action3D dataset using the protocol suggested in (Li et al, 2010).

Table 9 Cross-subject for Berkley MHAD

Descriptor	Accuracy (%)
FADE + 1-NN	84.3
FADE + SVM	91.6
U-FADE + 1-NN	92.4
U-FADE + SVM	93.8
SMIJ (Ofli et al, 2014)+1-NN	93.1
SMIJ (Ofli et al, 2014)+SVM	95.4
HMIJ (Ofli et al, 2014)+1-NN	81.6
HMW (Ofli et al, 2014)+SVM	83.1
LDSP (Bissacco et al, 2001)	92.8
(Du et al, 2015)	100

use more complex classifiers. Due to their low computational cost, FADE and U-FADE shine for applications in which the classifier needs to be frequently retrained. Combining FADE with SVM, the training time for the whole HDM05 is 1 s with a python script, loosing 2% to 4% accuracy with respect to methods based on deep networks, which requires much longer training time. We tested FADE and U-FADE with data sets of different sizes, in order to prove that the descriptor can be used for both large training sets with thousands of actions and for small training set with hundreds of actions. However, we noticed that in most cases U-FADE performs better than FADE in small-size datasets.

Emergent deep learning techniques such as (Cho and Chen, 2014) have a good potential to achieve a high

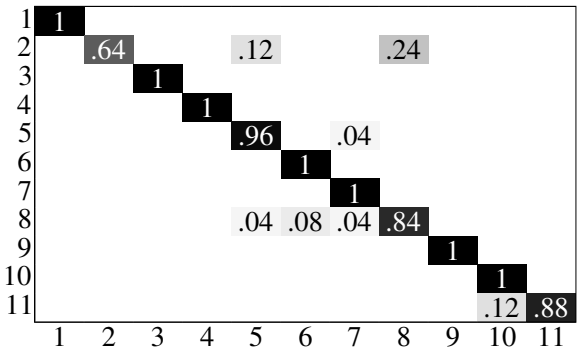


Fig. 12 Confusion matrix of U-FADE with SVM evaluated on the MHAD dataset

accuracy with large datasets, paying the price of high computational cost, especially in the training phase. However, such approaches do not perform well when only few examples are available. FADE and U-FADE offer a balance for good performance with both small and large datasets, by exploiting prior knowledge on human biomechanics. FADE and U-FADE have a $O(1)$ space complexity and a $O(n \log n)$ time complexity. Hence, they scale well when the dimensionality of the problem increases. In the literature, most action recognition methods use supervised learning approach. On the contrary, we have tested our descriptor also with diverse clustering methods. We found out that FADE and especially U-FADE work well with clustering approaches when the size of the dataset is small (few hundred of actions).

In the light of the diverse experiments and comparisons performed in this work, we suggest approaches based on FADE and U-FADE in applications where a low computational cost and robustness to noise are important issues. Also, FADE and U-FADE perform well when we want good performance on both small and large datasets, and when we seek a good trade-off between accuracy and computational cost. For this reasons, the proposed descriptors are good solutions for autonomous robots with limited computation power and energy storage capabilities. However, when powerful CPUs (and GPUs) are available without relevant power consumption issues, a large set of training data is available, and there is no need to retrain often the classification algorithm, methods based on deep networks and RNN can be a very solid option.

We believe that, for robotic applications, developing descriptors suitable for unsupervised learning has a particularly important value. However, still much has to be understood on descriptors for unsupervised learning that can achieve good performance also with large datasets. A future research direction will consist in finding extensions of FADE and U-FADE that perform better with unsupervised learning on both small and large action sets. Also, run-time action recognition methods leveraging the online FFT will be investigated. We used FADE and U-FADE only for classifying actions that are already segmented with a state of the art approach. As next step, we will use the concepts of FADE and U-FADE also for segmenting data streaming into distinct behaviors. Another direction will consist in combining the proposed descriptor with motion invariant representations such as (Soloperto et al, 2015),(De Schutter et al, 2011), Lee et al (2017) and (Saveriano and Lee, 2013).

Acknowledgements This work has been supported by the Marie Curie Action LEACON, EU project 659265, and by the Technical University of Munich, International Graduate School of Science and Engineering (IGSSE).

References

Bishop CM (2006) Pattern Recognition and Machine Learning. Springer-Verlag New York, Inc.

Bissacco A, Chiuso A, Ma Y, Soatto S (2001) Recognition of human gaits. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol 2, pp II–52

Cavallo A, Falco P (2014) Online segmentation and classification of manipulation actions from the observation of kinetostatic data. Human-Machine Systems, IEEE Transactions on 44(2):256–269

Chen C, Liu K, Kehtarnavaz N (2013) Real-time human action recognition based on depth motion maps. Journal of real-time image processing pp 1–9

Chen X, Koskela M (2015) Skeleton-based action recognition with extreme learning machines. Neurocomputing 149:387–396

Cho K, Chen X (2014) Classifying and visualizing motion capture sequences using deep neural networks. In: International Conference on Computer Vision Theory and Applications (VISAPP, vol 2, pp 122–130

De Schutter J, Di Lello E, De Schutter JF, Matthysen R, Benoit T, De Laet T (2011) Recognition of 6 dof rigid body motion trajectories using a coordinate-free representation. In: IEEE International Conference on Robotics and Automation, pp 2071–2078

Di Benedetto A, Palmieri FA, Cavallo A, Falco P (2016) A hidden markov model-based approach to grasping hand gestures classification. In: Advances in Neural Networks, Springer, pp 415–423

Du Y, Wang W, Wang L (2015) Hierarchical recurrent neural network for skeleton based action recognition.

In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 1110–1118

Dyn N, Levin D, Rippa S (1990) Data dependent triangulations for piecewise linear interpolation. IMA journal of numerical analysis 10(1):137–154

Evangelidis G, Singh G, Horaud R (2014) Skeletal quads: Human action recognition using joint quadruples. In: International Conference on Pattern Recognition

Falco P, Saveriano M, Hasany EG, Kirk NH, Lee D (2017) A human action descriptor based on motion coordination. IEEE Robotics and Automation Letters 2(2):811–818

Forestier N, Nougier V (1998) The effects of muscular fatigue on the coordination of a multijoint movement in human. Neuroscience letters 252(3):187–190

Gowayyed MA, Torki M, Hussein ME, El-Saban M (2013) Histogram of oriented displacements (hod): Describing trajectories of human joints for action recognition. In: International Joint Conference on Artificial Intelligence

Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507

Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1):489–501

Kulić D, Takano W, Nakamura Y (2008) Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains. The International Journal of Robotics Research 27(7):761–784

Kulić D, Ott C, Lee D, Ishikawa J, Nakamura Y (2011) Incremental learning of full body motion primitives and their sequencing through human motion observation. The International Journal of Robotics Research

Le Naour T, Courty N, Gibet S (2012) Fast motion retrieval with the distance input space. In: Motion in Games, Springer, pp 362–365

Lee D, Ott C, Nakamura Y (2009) Mimetic communication with impedance control for physical human-robot interaction. In: IEEE International Conference on Robotics and Automation, 2009., pp 1535–1542

Lee D, Soloperto R, Saveriano M (2017) Bidirectional invariant representation of rigid body motions and its application to gesture recognition and reproduction. Autonomous Robots pp 1–21

Leightley D, Li B, McPhee JS, Yap MH, Darby J (2014) Exemplar-based human action recognition with template matching from a stream of motion capture. In: Image Analysis and Recognition, Springer, pp 12–20

Li W, Zhang Z, Liu Z (2010) Action recognition based on a bag of 3d points. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops, pp 9–14

Liu J, Shahroudy A, Xu D, Wang G (2016) Spatio-temporal lstm with trust gates for 3d human action recognition. In: European Conference on Computer Vision, Springer, pp 816–833

Lovász L, Plummer MD (2009) Matching theory, vol 367. American Mathematical Soc.

Mahasseni B, Todorovic S (2016) Regularizing long short term memory with 3d human-skeleton sequences for action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 3054–3062

Medina JR, Lawitzky M, Mörtl A, Lee D, Hirche S (2011) An experience-driven robotic assistant acquiring human knowledge to improve haptic cooperation. In: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, IEEE, pp 2416–2422

Ofli F, Chaudhry R, Kurillo G, Vidal R, Bajcsy R (2013) Berkeley mhad: A comprehensive multimodal human action database. In: IEEE Workshop on Applications of Computer Vision, pp 53–60

Ofli F, Chaudhry R, Kurillo G, Vidal R, Bajcsy R (2014) Sequence of the most informative joints (smij): A new representation for human skeletal action recognition. Journal of Visual Communication and Image Representation 25(1):24–38

Pervez A, Ali A, Ryu JH, Lee D (2017) Novel learning from demonstration approach for repetitive teleoperation tasks. In: World Haptics Conference (WHC), pp 60–65

Rabiner LR (1989) A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2):257–286

Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. Transactions on Acoustics, Speech, and Signal Processing pp 43–49

Saveriano M, Lee D (2013) Invariant representation for user independent motion recognition. In: IEEE International Symposium on Robot and Human Interactive Communication, pp 650–655

Schmidts AM, Lee D, Peer A (2011) Imitation learning of human grasping skills from motion and force data. In: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, IEEE, pp 1002–1007

Shah D, Falco P, Saveriano M, Lee D (2016) Encoding human actions with a frequency domain approach. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 5304–5311

Soloperto R, Saveriano M, Lee D (2015) A bidirectional invariant representation of motion for gesture recognition and reproduction. In: IEEE International Conference on Robotics and Automation (ICRA), pp 6146–6152

Vemulapalli R, Arrate F, Chellappa R (2014) Human action recognition by representing 3d skeletons as points in a lie group. In: IEEE Conference on Computer Vision and Pattern Recognition, pp 588–595

Walker JS (1996) Fast fourier transforms, vol 24. CRC press

Wang C, Wang Y, Yuille AL (2016) Mining 3d key-pose-motifs for action recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 2639–2647

Wang J, Liu Z, Wu Y, Yuan J (2012) Mining actionlet ensemble for action recognition with depth cameras. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 1290–1297

Wang Q, Kurillo G, Ofli F, Bajcsy R (2015) Unsupervised temporal segmentation of repetitive human actions based on kinematic modeling and frequency analysis. In: 2015 International Conference on 3D Vision, pp 562–570, DOI 10.1109/3DV.2015.69

Xia L, Chen CC, Aggarwal J (2012) View invariant human action recognition using histograms of 3d joints. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pp 20–27

Xu W, Liu X, Gong Y (2003) Document clustering based on non-negative matrix factorization. In: ACM SIGIR conference on Research and Development in Informaion Retrieval, pp 267–273

Zhu W, Lan C, Xing J, Zeng W, Li Y, Shen L, Xie X, et al (2016) Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks. In: AAAI, vol 2, p 8