CuDi3D: Curvilinear displacement based approach for online 3D action detection

Said Yacine Boulahia, Eric Anquetil, Franck Multon, Richard Kulpa

CuDi3D: Curvilinear Displacement based approach for online 3D action detection

Said Yacine Boulahia[a,**], Eric Anquetil[a], Franck Multon[b], Richard Kulpa[b]

[a] *Univ Rennes, INSA de Rennes, IRISA. France*
[b] *Univ Rennes, Inria, M2S. France*

Abstract

Being able to interactively detect and recognize 3D actions based on skeleton data, in unsegmented streams, has become an important computer vision topic. It raises three scientific problems in relation with variability. The first one is the temporal variability that occurs when subjects perform gestures with different speeds. The second one is the inter-class spatial variability, which refers to disparities between the displacement amounts induced by different classes (i.e. long vs. short movements). The last one is the intra-class spatial variability caused by differences in style and gesture amplitude. In this paper, we design an original approach that better considers these three issues. To address temporal variability we introduce the notion of curvilinear segmentation. It consists in extracting features, not on temporally-based sliding windows, but on trajectory segments for which the cumulated displacement equals a class-based amount. Second, to tackle inter-class spatial variability, we define several competing classifiers with their dedicated curvilinear windows. Last, we address intra-class spatial variability by designing a fusion system that takes the decisions and confidence scores of every competing classifier into account. Extensive experiments on four challenging skeleton-based datasets demonstrate the relevance of the proposed approach for action recognition and online action detection.

Keywords:
Online action recognition, Skeleton-based approach, Human action detection, Curvilinear displacement, Online segmentation, Skeleton data stream

1. Introduction

Online action detection (OAD) became recently a major computer vision concern [1, 2]. Different from offline action recognition, which focuses on labelling the action after it is fully observed, OAD intends to temporally identify the action, as early as possible, in an unsegmented stream. Addressing such problem is particularly relevant for enhancing human-machine interaction. This supposes a real-time understanding of what a subject is about to do in order to allow the machine returning an appropriate feedback.

Despite the increasing number of works addressing OAD [3, 4, 5, 6, 7], it remains a complex research problem due to three main issues. A first issue is the *temporal variability* which occurs when subjects perform gestures with different speeds and with pauses inserted before, within or after the action. This variability prevents from knowing beforehand the full duration of a potential ongoing action. The second issue, referred to as *inter-class spatial variability*, is due to the fact that different gesture classes are likely to result in different amount of displacements (Figure 1). If this second issue is not addressed, a detection decision would be made with an insufficient tra-

*Corresponding author: Tel.: +33-299-842-291;
Email address: said-yacine.boulahia@irisa.fr (Said Yacine Boulahia)

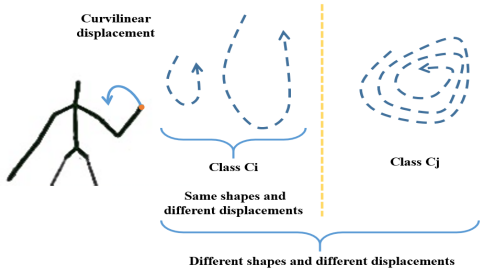

Figure 1: Illustration with a single joint trajectory of *intra-class spatial variability* within a class Ci (left) and *inter-class spatial variability* between Ci and Cj (right).

jectory information quantity, which affects the detection performance. The last issue is the *intra-class spatial variability* which is an intrinsic property of human motion. In fact, this variability often results in samples with different amount of displacements while belonging to the same action class (Figure 1). This is mainly due to differences in performance style inherent to each subject leading to different amplitudes of the same gesture class. In some applications, capturing such *intra-class variabilities* might be desirable as it brings additional information and could allow for different interpretations of the same class of gesture. However, in this paper, these variabilities are undesirable and thus must be neutralized.

Approaches proposed so far addressed the OAD problem from a partial view, without explicitly dissociating and efficiently considering temporal as well as spatial inter- and intra-class variabilities. Therefore, in this paper, the aim is to propose an original skeleton-based approach designed as a step by step solution which addresses transparently the three OAD issues. In particular, to tackle *temporal variability* we introduce the concept of curvilinear segmentation. It consists in dynamically defining windows depending on the amount of information (i.e. motion) available in the unsegmented flow. The amount of information is in fact the cumulated length of the (curvilinear) trajectory produced by skeleton joints which we refer to as curvilinear displacement. The window size is thus indexed on the curvilinear displacement flow instead of the usual temporal flow so as to capture consistent motion segments. Moreover, to handle *inter-class spatial variability*, several specialized classifiers with their dedicated curvilinear windows are trained and put into competition. Last, *intra-class spatial variability* is addressed by means of a fusion system, modeled as a continuously updated histogram, which takes classifiers decision and confidence score into account.

The remainder of this paper is organized as follows: Section 2 reviews recent related work in the literature. Section 3 contains an overview of the proposed approach, named **CuDi3D**. This approach is then detailed in three stages, one for each identified variability, presented in Section 4. In particular, to tackle *temporal variability* we introduce the curvilinear segmentation concept. We then explain how multiple classifiers are designed to address *inter-class spatial variability*. Last, we detail the decision fusion system which allows for addressing *intra-class spatial variability*. In Section 5 we explain how the proposed **CuDi3D** approach is adapted to be used for pre-segmented action recognition. Various results obtained over four skeleton-based datasets according to different protocols are presented in Section 6 and compared to a total of eighteen other methods. An overall discussion of the OAD task is presented in Section 7. Finally, in Section 8 conclusions and future research directions are drawn.

2. Related work

Most of previous OAD methods follow a temporal sliding window design. These methods can mainly be distinguished based on how the sizes of their temporal windows are defined. Early OAD methods used fixed-size temporal sliding windows. In [8], a temporal sliding window of 35 frames was used along with a Random Forest classifier for performing the action detection task on the streaming data. Zhao et al. [9, 10] employed also a 35-frame temporal sliding window along with the so-called Structured Streaming Skeleton (SSS) features. SSS consists in minimum DTW distances between all the scanned subsequences (ending at the current frame) and a template in a pair-joints based dictionary. Moreover, the moving pose descriptor proposed by Zanfir et al. [11] captured the positions, speeds and accelerations of skeleton joints over short fixed-size temporal sliding windows. In [7, 12], the motion sequence located in the temporal sliding window is first split into sub-sequences called Motion Segments (MS) which are then modelled via a Dynamic Naive Bayes classifier for identifying the ongoing action, if any. Last, Bloom et al. [6, 13, 14, 15] proposed to

decompose the motion according to a hierarchical body model and then consider the motion of the low level body parts. Authors relied also on a fixed-size temporal sliding window.

However, due to *temporal variability* between different subjects, even when performing the same gesture class, there is no global temporal size of the sliding window that would be adequate. An alternative approach consists in sequentially constructing a larger window by combining an initial temporal window of minimal size with the incoming new frames. This was done by Huang et al. [16] who proposed the Sequential Max-Margin Event Detectors (SMMED). It consists in sequentially discarding the unlikely classes on consecutive segments of increased size, until a reliable class label can be identified from the remaining classes individually. A similar approach based on multinomial Naive Bayes model was introduced in [17]. The issue with such methods is that determining the minimum window size is non-trivial, while simultaneously having a great impact on detection performance [18].

Sharaf et al. [5] proposed to put into competition three temporal sliding windows of different sizes in order to accommodate for actions of different temporal scales within the same class. Authors analyzed the training data in order to select three sizes as a minimum, medium and maximum number of frames along which features were built. A similar multi-level approach but more computationally efficient was presented in [19, 20]. Authors proposed to identify at each position (frame) the subinterval ending at this position with the maximum sum. If this sum exceeded a given threshold, then the system decided that an action is being performed. But one may wonder why using only three temporal scales when there should be almost as many temporal scales as there are samples.

Moreover, by using temporal sliding windows with either fixed-size or increasing size, these approaches implicitly assume that the duration of actions could be known beforehand. Such solutions are in fact partially satisfactory given that real-life gestures differ greatly in terms of motion nature and displacement amounts. This is all the more valid since that execution time is different from one subject to another even when performing the same gesture.

Besides, the system latency is an important aspect in designing interactive, action-based interfaces and should be considered when conceiving OAD approaches. High latency causes in fact system feedback to lag behind user actions and thus significantly degrades the interactivity of the user experience. Therefore, several approaches were proposed so as to reliably recognize actions with minimal latency [4, 21, 3, 22, 23, 24]. Ellis et al. [3] proposed new algorithms for reducing latency for both pre-segmented and online action classification tasks. They used a latency-aware learning formulation to train a logistic regression based classifier that automatically determines distinctive canonical poses from data and uses these to robustly recognize actions in the presence of ambiguous poses. Kviatkovsky et al. [4] also proposed to reduce latency by representing a spatio-temporal volume using a temporal extension of the covariance descriptor in an exceptionally low dimensional (15D) space. Authors showed that their reduced descriptor is well suited for online scenarios where only incomplete spatio-temporal volumes are available.

Recently, deep learning approaches have been proposed for action recognition using skeleton data. Most of these approaches consider only the problem of pre-segmented action recognition [25, 26, 27, 28, 29, 30, 31] while only a few of them address the more challenging problem of online action detection [32]. In these architectures, early layers learn features from unlabelled streams, in contrast to selecting hand-crafted features, while later layers may perform feature transformation and finally classification. One of the first architectures was proposed by Du et al. [27] who divided the human skeleton into five parts according to human physical structure, and then separately fed them to five corresponding bidirectionally recurrently connected subnets (BRNNs). Authors claimed that the stacked BRNNs are supposed to extract the spatial and temporal features of the skeleton sequences. Another RNN-based architecture was proposed by Zhu et al. [28] in which the skeleton is taken as the input at each time slot and a novel regularization scheme to learn the co-occurrence features of skeleton joints is introduced. Besides action recognition, online action detection was considered in the work of Li et al. [32] who employed an end-to-end RNN with a joint classification and regression optimization objective function so as to accurately localize the start and end points of actions.

The benefit of deep learning is that the features can be automatically selected without the use of prior knowl-

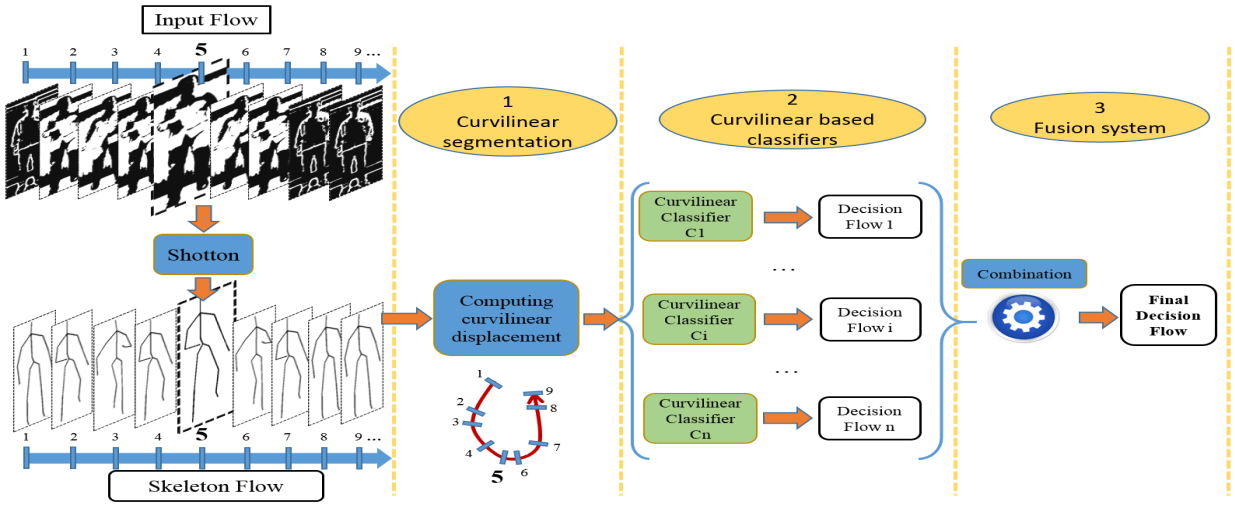

Figure 2: Overview of the proposed OAD approach using skeleton data extracted from depth maps. The approach is composed of three stages, such that at each stage an OAD issue is addressed. *Temporal variability* is considered at the first stage. *Inter-class spatial variability* is tackled at the second stage. *Intra-class spatial variability* is addressed at the last stage.

edge. These methods have achieved comparable or even better accuracy than engineered features for offline action recognition. Nevertheless, deep learning approaches require large amounts of training data, which may not always be available. Besides, due to their opaque structure, it is difficult to understand how such approaches address the highlighted variabilities and, when they fail, how to identify what is causing these failures.

Therefore, the step by step approach we propose intends to address the OAD problem in a transparent manner by focusing on *temporal variabilities*, *inter-class* and *intra-class spatial variabilities* while ensuring real-time efficiency. In fact, our approach improves the search of temporal relationship inside a motion by focusing and guiding this search on sub-segments of pre-defined curvilinear sizes. This is better than the RNN/LSTM based approaches whose search of such relationships is fully automatic and does not take into account the specificities of the modelled actions during this search. Besides, in our approach, the classifier's task is alleviated as it is provided with homogeneous segments. This is supposed to make it more efficient as it only focuses on the classification. The proposed approach is detailed in the following sections.

3. Overview

In this paper, we investigate online action detection using skeleton data extracted from depth maps [34] for three main reasons. Firstly, a skeleton-based representation does not only capture the essential structure of a subject in an easily understandable way but is also robust to variations in viewpoint and illumination [32]. Secondly, the skeleton compactness (25*3=75 positions per frame) allows for a more efficient processing of these motion data, and makes their use more adapted for a real-time evaluation [33]. Finally, skeleton data allow for a straightforward tracking of a subject and a high level understanding of his/her movements since these data consist in trajectories of the body joints [35, 36].

The overall OAD approach is illustrated in Figure 2. This approach is designed according to three stages so that to address the three identified OAD issues, namely *temporal variability*, *inter-class* and *intra-class spatial variabilities*.

In particular, to address *temporal variability* we introduce the novel concept of curvilinear segmentation (Stage 1 in Figure 2). Unlike previous methods, features extracted over the curvilinear windows during training and testing are consistent as they represent similar motions re-

gardless of execution speed.

Furthermore, *inter-class spatial variability* refers to the fact that different action classes rely on different curvilinear displacement quantities. To tackle this second issue, several specialized classifiers with dedicated curvilinear windows compete to detect and recognize the ongoing action in the unsegmented flow (Stage 2 in Figure 2). Each classifier is trained to detect and recognize all gestures whose cumulated curvilinear displacement exceeds the minimal length required by the classifier.

Finally, *intra-class spatial variability* is addressed at the decision combination step (Stage 3 in Figure 2). At this step, the final decision is made by a fusion system taking each classifier decision and confidence score into account, modeled as a continuously updated decision histogram. The approach, that we refer to as **CuDi3D**, is detailed in Section 4.

4. Proposed approach: CuDi3D

In this Section, we first introduce the concept of curvilinear segmentation. We then detail the training step of the classifiers based on a curvilinear window. Here, we briefly present the local features used, namely HIF3D [37]. Last, we explain the decision process that allows to output the final decision flow.

4.1. Curvilinear segmentation

First of all, a pre-processing is operated on the input skeleton data in order to tackle the anthropometric variability. In fact, different subjects are likely to have different morphological properties which may lead to bad recognition performance if not considered. In this preprocessing, only a subset of joints is used which enhances the real-time running of the proposed approach. In particular, raw human body is represented by a set of 3D joints forming a hierarchy of subparts (such as the arm, the trunk, etc.) expressed relatively to their parent. Previous studies [37, 38, 39, 40] showed that only a subset of these joints, namely shoulders, elbows, wrists, hips, knees and ankles, is sufficient for characterizing a wide range of motions.

To ensure effective independence from the subjects' morphology, coordinates of the retained joints should be normalized such that only adimensional data are stored.

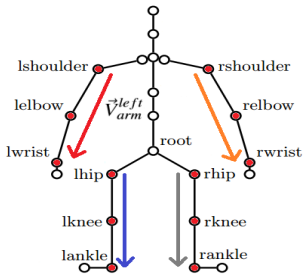

Figure 3: Selected joints and the associated morphology-independent vectors.

For instance to get the morphology independent vector of the left-up body, noted $\vec{V}_{LeftArm}(t)$ in Figure 3, we first compute the vector that connects the two end joints of the left-up body part which are Left Shoulder j^t_{LSh} and Left Wrist j^t_{LWr}, at time t. This vector is then normalized by the total arm length. The used formula is given in Eq. 1, where j^t_{LEl} refers to the Left Elbow joint position at time t.

$$\vec{V}_{LeftArm}(t) = \frac{\overrightarrow{j^t_{LSh} j^t_{LWr}}}{\|\overrightarrow{j^t_{LSh} j^t_{LEl}}\| + \|\overrightarrow{j^t_{LEl} j^t_{LWr}}\|} \quad (1)$$

In this paper, we proceed similarly such that, at each frame, the considered input data are the trajectories of each of the four normalized vectors associated with movement of arms and legs as illustrated in Figure 3.

To illustrate the impact of this process, we provide in Figure 4 the left arm trajectories, before and after normalization, of three different subjects while performing the "raise arms" action. We additionally report in Table 1 the size of each subject's arm and the lengths of the related trajectories before and after normalization.

From the illustrations in Figure 4, it is interesting to note that the overall shape of the pattern is preserved after normalization. Such normalization allows in fact to standardize the joints trajectories while preserving the characteristic information related to the action. Moreover, we can note that the normalized trajectories are smoother than the raw ones (before normalization). This could be explained by the fact that the normalization is performed frame by frame based on the real, and probably noisy, measures of the limbs positions. This allows in fact a mutual neutralization of the noisy measures. Furthermore,

from Table 1, we can note that the normalization process allows to significantly attenuate the variation between the lengths of the trajectories. This is particularly interesting as the subjects have different morphological properties, as shown in the second column of Table 1.

The *temporal variability* is the first OAD issue to be addressed. To this end, we propose a novel concept consisting in the curvilinear segmentation. This concept consists in dynamically defining windows depending on the amount of information (i.e. motion) available in the unsegmented flow. The window size is constrained by the curvilinear displacement instead of the usual temporal flow to capture the amount of available data for detection and recognition.

The metric used to measure the amount of information is the curvilinear displacement of joints. We therefore define a function $CuDi(F_S, F_E)$ that computes the curvilinear displacement for a given motion segment, starting at frame F_S and ending at F_E, as follows:

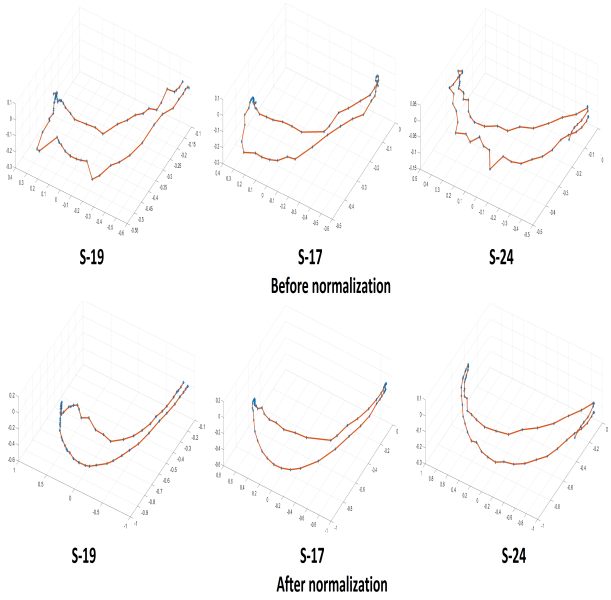

Figure 4: Illustration of the left arm trajectory, before and after normalization, while performing a "raise arms" action from the MSRC-12 dataset[8]. These samples were performed by subjects S-19, S-17 and S-24.

Subject ID	Arm's size (m)	Length before (m)	Length after
S-19	0.54	1.79	2.79
S-17	0.51	1.40	2.70
S-24	0.47	1.32	2.64

Table 1: Morphology and displacement information of "raise arms" action samples from the MSRC-12 dataset[8] performed by three different subjects. The displacements of the left arm are reported before and after normalization. m refers to meter.

$$CuDi(F_S, F_E) = \sum_{i=F_S}^{i=F_E} d_i^{Avg} \tag{2}$$

where d_i^{Avg} is the instantaneous average displacement, computed for each frame i as reported in Equation 3:

$$d_i^{Avg} = \sqrt{(d_i^{LeftArm})^2 + (d_i^{RightArm})^2 + (d_i^{LeftLeg})^2 + (d_i^{RightLeg})^2} \tag{3}$$

where the $d_i^{LeftArm}$, $d_i^{RightArm}$, $d_i^{LeftLeg}$ and $d_i^{RightLeg}$ are the displacement values of each of the four body parts, computed as:

$$d_i^j = \sqrt{(x_i^j - x_{i-1}^j)^2 + (y_i^j - y_{i-1}^j)^2 + (z_i^j - z_{i-1}^j)^2} \tag{4}$$

with $j \in \{LeftArm, RightArm, LeftLeg, RightLeg\}$ and $x_i^j, y_i^j, z_i^j, x_{i-1}^j, y_{i-1}^j, z_{i-1}^j$ the 3D coordinates of the corresponding vectors respectively in the current frame i and the previous frame $i-1$.

Based on the metric defined in Equation 2, we define a curvilinear window as being a sliding window whose size is continuously updated such that it encompasses, at each frame, a specific curvilinear displacement. For instance, at a frame F_t, the curvilinear window should encompass the motion segment ending at this frame F_t and starting at the frame F_S. The frame F_S is determined such that:

$$S = \max_{s_i} CuDi(F_{s_i}, F_t) \geq \rho \quad ; 1 \leq s_i < t \tag{5}$$

with ρ a curvilinear displacement threshold specific to each classifier. This threshold bounds the sliding window at each instant and is computed using the ground truth start and end of each action sample (see Equation 6).

In Figure 5, we illustrate the difference between the

curvilinear window and the usual temporal sliding window on an input flow. In particular, we show how these two type of windows operate on two samples of the same action class that are performed at different speeds.

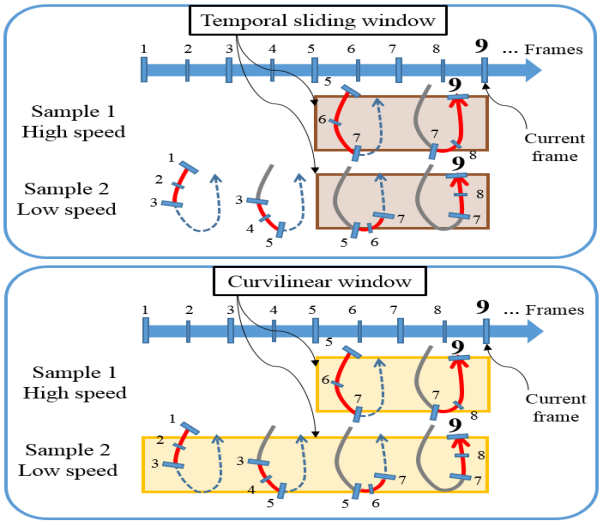

Figure 5: Illustration of the difference between the curvilinear window and the usual temporal sliding window. We consider the motion extracted with these two windows at frame 9 for two samples of the same class which were performed at different speeds.

The curvilinear window encompasses motion segments with the same curvilinear displacement for both samples regardless of their execution speed. In fact, only the curvilinear displacement controls the window size and not the number of frames needed to perform it. On the contrary, the usual temporal sliding window always considers the same number of frames regardless of the resulting curvilinear displacement. In this illustration, one can notice that the usual temporal sliding window does not always consider the adequate motion part as the size of such window is temporally-based. The curvilinear window thus allows the extraction of features over consistent motion segments which would better tackle the *temporal variability*. The curvilinear window is then employed to process input data and to train curvilinear-based classifiers as explained in the following section.

4.2. Curvilinear-based classifiers

The second issue considered in our study is the *inter-class spatial variability*. This refers to the fact that, in the general case, different action classes would produce different curvilinear displacements. A single curvilinear window would thus allow for tackling the *temporal variability* of a single action class, but not the *inter-class spatial variability* between different action classes. We therefore propose to use as many classifiers as there are curvilinear displacements.

We refer to the different classifiers as Ci, and the different gesture classes as Gi with $1 \leq i \leq n$ and n the total number of gesture classes (n is also the total number of classifiers). In general, there will be as many classifiers as there are gesture classes as we hypothesize that all gesture classes have different curvilinear displacements. In practice, the number of classifiers could be reduced if some gesture classes have similar curvilinear displacements which would decrease the processing time. Each such classifier is trained to recognize all the gesture classes [1] and not only the class to which it is associated. Therefore, the output of each such classifier is a gesture label, i.e. one of the multiple class labels, and is not only a binary output. In the following, we consider the general case in which there are as many classifiers as there are gesture classes.

In order to train a given classifier Ci, we first compute the curvilinear displacement quantity $CuDi_i$ associated with the gesture class Gi . It consists in the average of all the curvilinear displacements of the samples belonging to the gesture class Gi (Equation 6).

$$CuDi_i = \frac{1}{K} \times \sum_{j=1}^{j=K} CuDi(F_{S_j}, F_{E_j}) \tag{6}$$

where K is the total number of samples belonging to the gesture class Gi and F_{S_j} and F_{E_j} are the ground truth start and end of the j^{th} sample, respectively.

The curvilinear displacement $CuDi(F_{S_j}, F_{E_j})$ of each gesture sample j of the training set is computed between the ground truth start F_{S_j} and end F_{E_j} by means of Equation 2. As we target real-time detection, the ground truth end corresponds in fact to the concept of action point introduced in the seminal work of [41]. It refers not to the effective end of an action but to a temporal anchor

[1] These gestures should at least produce as much curvilinear displacement as required by the considered classifier.

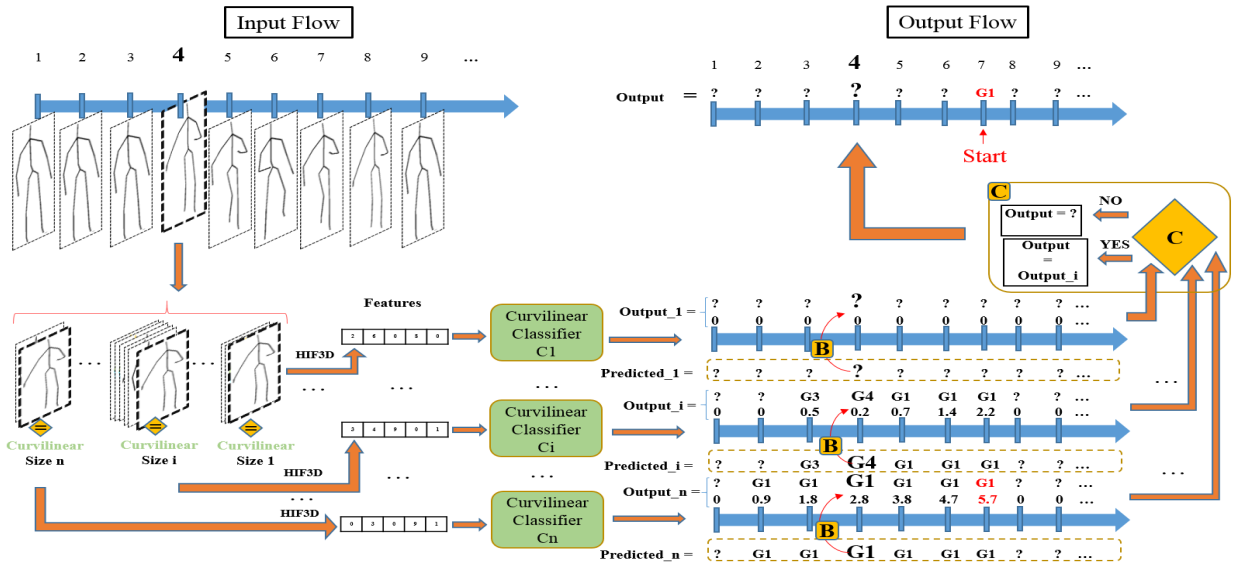

Figure 6: Overall system composed of curvilinear classifiers, one for each curvilinear displacement threshold. Different curvilinear classifiers extract features over different curvilinear windows. Besides, blocs '**B**' and '**C**', have in charge to process the raw predicted classes and to combine the decisions of the different classifiers respectively. '**?**' means that no class is predicted yet, while G1, ..., Gn are predicted classes and C1, ..., Cn are classifiers. *Predicted_i* and *Output_i* are the raw flow and the processed flow, respectively. *Output_i* flow consists in the *Predicted_i* flow plus the cumulated scores at each frame.

at which the presence of the action is clear. This action point can be uniquely identified for all samples of that action class.

Next, by means of the curvilinear window of curvilinear size $CuDi_i$, the training samples of all action classes are parsed between their ground truth start and action point. During this process, local features are extracted according to this curvilinear window to build the training set of the classifier *Ci*. As far as local features are concerned, we chose the state-of-the-art HIF3D feature-set [37]. HIF3D was initially conceived for recognizing pre-segmented skeleton-based actions and is very compact (89 features) which makes it suitable for real-time purposes. In fact, these features were inspired by recent achievements in hand-drawn patterns modelling. They were designed based on an efficient and compact handwriting feature-set. For more details about HIF3D features we refer the reader to [37].

It is beyond the scope of this paper to focus on the classification step. We therefore employ linear SVMs trained in an one-versus-all fashion in order to build multi-class classifiers. We optimised the SVM parameters by means of a cross-validation on training sets. Note that linear SVMs is a widespread classifier within the computer vision community as it allows for good generalization even from few examples, while avoiding over-fitting. To implement the classifier, we make use of the LIBSVM library [42].

4.3. Decision process

The decision process refers to how the decisions of all classifiers are combined during the testing step to detect the final class. During this process, the third OAD issue, namely *intra-class spatial variability*, is considered. This variability refers to the fact that samples belonging to the same action class could result in different amounts of displacements. One main reason is the disparities between the performance style of each subject.

This process is illustrated in Figure 6. In fact, during the testing step, the input skeleton stream is processed with several curvilinear windows of different curvilinear sizes. A given classifier *Ci* is launched only if the cumulated curvilinear displacement exceeds the curvilinear

size $CuDi_i$ of its associated curvilinear window. The outputs of all classifiers are then processed and combined based on a fusion system to get the final class. The fusion system takes the classifiers decision, referred to as *Predicted_i* in Figure 6, and confidence score into account.

In particular, the fusion system is mainly composed of two types of decision blocs namely **B** and **C** (Figure 6). Each classifier *Ci* is associated with a bloc **B** which is used to process the classifier's raw prediction flow, namely *Predicted_i*. Each bloc **B** is based on a local histogram which has as many entries as there are classes to predict. This local histogram cumulates the score of each class predicted by the classifier *Ci*. The output of each bloc **B** is a processed flow named *Output_i*. Furthermore, these processed flows are fused in bloc **C** to get the final output flow (Figure 6). Bloc **C** is based on a global histogram which is continuously updated by the output of each bloc **B**. This global histogram has as many entries as there are classifiers. In the following, we first explain how a given bloc **B** processes the raw prediction flow provided by a classifier *Ci*. Then, we consider the functioning of bloc **C** which fuses all the processed decision flows.

4.3.1. Local histogram based processing

As illustrated in Figure 6 (left), each classifier *Ci* receives a feature vector built over a different curvilinear window ending at the current frame. The raw result of each classifier, noted *predicted_i*, can either be the symbol ? or one of the gesture classes (G1, G2, ...). The symbol ? stands for the fact that no decision has yet been taken, as the cumulated curvilinear displacement has not yet reached the minimum curvilinear size of the window classifier. The predicted result *predicted_i* is then sent to bloc **B** of classifier *Ci* to update its local histogram noted His_i. This histogram has as many entries as there are classes and is used to record the cumulated confidence score of each potential class.

Formally, the j^{th} entry ($1 \leq j \leq n$) of a histogram His_i associated with classifier *Ci* is updated at each frame according to the following formula:

$$His_i(j) = \begin{cases} His_i(j) + \beta, & \text{if } j = Predicted\_i \\ His_i(j) - \gamma, & \text{otherwise} \end{cases} \quad (7)$$

Where β and γ are not parameters but difference of scores obtained by each classifier. In fact, at each prediction, each classifier outputs the confidence score of the predicted class but also that of all the other classes. Confidence scores range between 0 and 1 such that the closer to 1 the more confident the classifier is. β equals to the difference between the score of the currently predicted class, i.e. *Predicted_i*, and the score of the secondly ranked predicted class by the classifier *Ci*. γ corresponds to the difference between the score of *Predicted_i* and the score of the j^{th} entry of the histogram. n is the total number of classes.

The main goal behind this update procedure is to be more robust to confusions between two classes. In fact, if the best two classes predicted by a classifier have scores close to each other, i.e. score difference is too low, it is likely that the classifier will confuse these two classes and will not have enough information yet to decide of the ongoing action. Therefore the associated local histogram should grow slower by adding the difference between these two classes instead of adding for instance the confidence score of the most-likely class.

An illustration of how this local histogram is updated is provided in Figure 7. In this illustration, we assume that there are three gesture classes namely G1, G2 and G3 and we consider the functioning of the local histogram during four frames (4, 5, 6, and 7). The classifier predicted G1, G2, G1 and G1 at frame 4, 5, 6 and 7 respectively. At frame 5 for instance, the cumulated score of G2 (middle bar) is increased, while that of all other classes is decreased.

Thus, at each instant, the output of the bloc **B** associated with classifier *Ci* is the predicted class *Predicted_i* along with its histogram score, namely $His_i(j = Predicted\_i)$. This flow is then passed to bloc **C** which fuses decisions of local histograms and outputs the final class. The functioning of bloc **C** is presented in the following Section.

4.3.2. Fusion of decisions

At this step, the goal is to fuse the classifiers output flow to get the final class. A global histogram is used, noted *His_Global*, which is composed of as many entries as there are classifiers. Each entry *i* corresponds to the ongoing class *Predicted_i* predicted by a classifier *Ci* at the current frame along with its cumulated score,

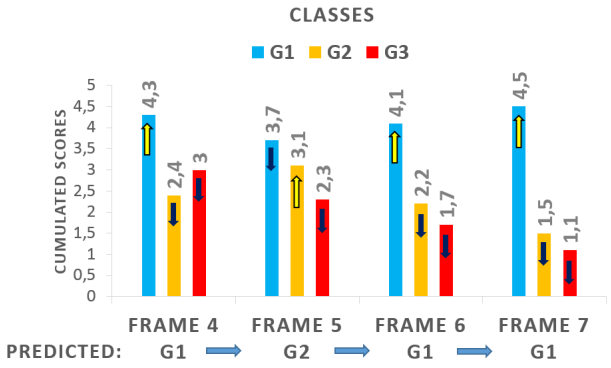

Figure 7: Illustration of the local histogram functioning with three classes at frames 4, 5, 6 and 7. ↑ and ↓ symbolize an increase and decrease of the score, respectively.

i.e. $His_i(j = Predicted\_i)$. The basic idea of the decision fusion is to ensure a compromise between the decision confidence and the detection latency. We therefore compute an experimentally-determined threshold matrix, noted *ThreshMat*, which is used inside bloc **C**. *ThreshMat* is $m \times n$ matrix where m is the number of classifiers and n is the number of classes. We consider here the general case where there are as many classifiers as there are classes, i.e. $m = n$.

Furthermore, the way the values of *ThreshMat* are determined is decisive to ensure the balance between reducing detection latency and increasing robustness to false positives. To this end we first compute two matrices: *PrecocityMat* and *ConfusionMat*. *PrecocityMat* is composed of cumulated confidence thresholds that a given classifier could ideally reach for each class. In the contrary *ConfusionMat* contains minimum threshold values that a classifier score should exceed to avoid confusing different classes.

In practice, to get the values of both matrices we discard the testing set and only consider the training set. We apply on that training set a cross validation procedure such as to get a new training set and a validation set. In particular, we split the main training set into 10 folds, train each classifier on 9 folds and then validate on the remaining fold. We repeat this operation 10 times and at each iteration we simulate the detection process between the start and the action point instant of each sample of the validation fold. We report then in the $(i, j)^{th}$ element of *PrecocityMat* matrix the average of the scores cumulated until the action point by classifier *Ci* for class *Gj* if the predicted class *Gj* corresponds to the ground truth label of the processed sequence. At the same time we update the *ConfusionMat* matrix such that its $(i, j)^{th}$ element is the average of the scores cumulated by classifier *Ci* for class *Gj* if this predicted class *Gj* is wrong (i.e. different from ground truth label).

Last, each element $\theta_{i,j}$ of *ThreshMat* matrix is obtained as follow:

$$\theta_{i,j} = \frac{PrecocityMat(i,j) + ConfusionMat(i,j)}{2} \quad (8)$$

On the one hand, this is supposed to avoid too early decisions due to confusion with other classes when the previously observed frames are not sufficient to allow classifiers deciding about the ongoing action. On the other hand this procedure aims at reducing latency by emitting decisions when a classifier cumulates enough score and becomes confident about its decision. It is worth noting that this procedure allows for computing thresholds independently from the actions to recognize and could thus be applied to any type of datasets.

During the decision fusion, a class *Gj* outputted by a classifier *Ci* is considered as the final class if the cumulated score by this classifier *Ci* for this class *Gj* exceeds the threshold $\theta_{i,j} = ThreshMat(i, j)$. Thus any classifier trained with a specific curvilinear window size is able to predict any class if this classifier is sufficiently confident about its decision. By proceeding in this manner, we provide a solution to tackle *intra-class spatial variability* as a classifier can detect the occurrence of an action other than its basic action class, i.e. the one corresponding to its curvilinear size. We summarize the overall decision process in Equation 9.

$$Output = \begin{cases} \mathbf{Gj} \quad , & \text{if} \quad \exists\,(i,j) \quad 1 \le i,j \le n \quad \& \\ & His\_Global(i) \ge \theta_{i,j} \quad \& \\ & Output\_i = Gj \\ \mathbf{?} \quad , & \text{otherwise} \end{cases} \quad (9)$$

Thus, the system outputs ? as long as the local decisions are not made, which corresponds to no action cur-

rently detected. Moreover, if the fusion results in two or more potential action classes, then the decision is postponed until there is only one class left. At the end of the decision process, all histograms are reinitialized to zeros, as are the cumulated curvilinear displacements for each classifier.

An illustration of how the global histogram of bloc **C** operates is provided in Figure 8. Following the illustration provided in Figure 7, we consider three classifiers C1, C2 and C3 corresponding to the curvilinear size of three classes: G1, G2 and G3 respectively. In Figure 8, we provide the global histogram state at frame 7. The class predicted by classifiers C1 and C2 is G1, with different cumulated scores. The class predicted by classifier C3 is G2.

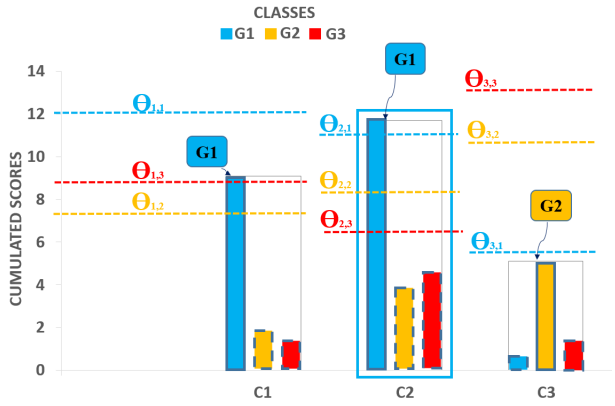

Figure 8: Illustration of the global histogram functioning at frame 7 with three classifiers C1, C2 and C3 which can predict either class G1, G2 or G3.

The classifier $C2$ is the one that succeeded in detecting the occurrence of the action class $G1$. In fact, the score cumulated by classifier $C2$ for class $G1$ is the only score that exceeds the associated threshold, i.e. $\theta_{2,1}$. Thus the final predicted class is $G1$.

5. Offline action recognition

In this Section, we explain how the proposed **CuDi3D** approach is adapted to be used for pre-segmented action recognition also known as offline action recognition. A key difference compared to previous offline approaches is that the testing sequence is processed progressively, i.e. frame by frame, as in an online setting. However, in this case the task is alleviated given that the start and end instants are known beforehand. Consequently some changes are made to the approach so as to take advantage of this new information.

Regarding the training step, curvilinear classifiers as presented in Sections 4.1 and 4.2 are trained in the same way. Besides, we add a new classifier which is trained with the whole pre-segmented samples of all classes. This additional classifier would be used if no curvilinear classifier is launched as explained below. Furthermore the confidence thresholds are no more learned. During the testing step, the decision process is updated and is illustrated in Figure 9.

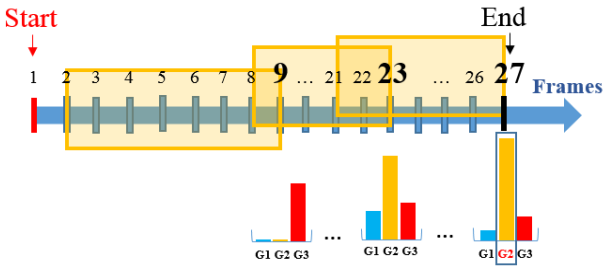

Figure 9: Adapted decision process for offline action recognition.

In particular, at each frame of the testing sequence only a single classifier is launched, instead of several classifiers in parallel, since we know exactly the start of the ongoing action. Starting from the beginning of the testing sequence, the cumulated curvilinear displacement is computed at each frame F_t and if that displacement exceeds the curvilinear size $CuDi_i$ of a given classifier Ci then this classifier is activated. Among the activated classifiers Ci at the current frame F_t we only consider the classifier Cj which corresponds to the largest curvilinear window $CuDij$ (Eq. 10).

$$CuDi_j = \max_{F_t} CuDi_i \tag{10}$$

HIF3D Features are then only extracted according to the corresponding curvilinear window $CuDi_j$. The goal of using at each frame a single activated classifier, the one with the largest curvilinear size, is twofold. On the one hand, this allows us to consider at each frame the largest motion segment so as to take into account as much infor-

mation as possible. On the other hand, it allows us to only consider an appropriate motion segment (and not necessarily the whole sequence) which ensures consistency between this motion segment and those used for training curvilinear classifiers.

Furthermore, as the testing sequence is progressively processed, a histogram is updated at each frame with the output decision and confidence score (Figure 9). In fact, for offline action recognition a single histogram is used, which is similar to local histograms presented in Section 4.3.1. Different from the online setting, this histogram is global as it is updated with the output of any classifier. At each time a class Gi is detected by the classifier actually used Cj, the histogram value corresponding to the i^{th} entry is increased by the difference between the score of the currently predicted class, i.e. Gi, and the score of the secondly ranked predicted class by the classifier Cj. Simultaneously, the score of all other classes is decreased according to Equation 7. At the end of the processed sequence, the final class is the one associated with the highest cumulated score in the histogram.

Moreover, if no classifier was launched due to insufficient curvilinear displacement of the actual action sample, then features are extracted over all the sequence and the additional classifier, presented above, is used to get the final class. It should be noted, though, that this additional classifier is only marginally used. For instance, in the experiment conducted on the HDM05 dataset (section 6.1), it is used to classify 99 samples out of 2337 samples, which corresponds to 4.23% of the whole dataset.

6. Experimental evaluation and discussion

In this section we first evaluate the **CuDi3D** approach in the context of offline action recognition. Even though the main goal of our approach is to address the OAD problem, this first and the second experiments aim at providing an idea about the performance of our approach compared to state-of-the-art approaches, specifically deep learning based methods, for **offline action recognition**. This first experiment is conducted using the Motion Capture Dataset HDM05 [43] and the second experiment is conducted by means of the MSRC-12 dataset[8].

In the second part, we present the results of our approach in the context of **online action detection (OAD)**. In particular, we conduct an extensive experimental study with the proposed approach and provide a comparison with twelve state-of-the-art OAD algorithms [19, 8, 5, 9, 20, 15, 44, 14, 32, 16, 17, 7] on three skeleton-based datasets, MSRC-12 [8], G3D [44] and MAD [16].

6.1. HDM05 dataset

The HDM05 dataset contains 2,337 skeleton sequences performed by 5 actors [43]. For fair comparison, we use the same protocol as first used by Cho and Chen [25] (65 classes, 10-fold cross validation each fold containing 234 sequences from all classes). Similar to OAD scheme a pre-processing is first applied and HIF3D features [37], used as local descriptors, are then extracted according to a temporal partitioning of two levels (Level = 2) as suggested in [37]. We train 66 (65 curvilinear + 1) SVM [42] classifiers which can recognize all classes, but only one of them (the one with the largest curvilinear size) is launched at each frame. Each curvilinear classifier is based on a curvilinear sliding window whose size corresponds to one of the 65 classes. The additional classifier is trained on the whole sequences regardless of the curvilinear size of each sequence. We optimized SVM parameters by means of a cross-validation on training sets.

Table 2 shows the results in terms of average accuracy. Overall, it has to be noted that on the HDM05 dataset our adapted **CuDi3D** approach improves over previous deep learning approaches using skeleton data and achieves a score of **99.4%**. First, it achieves better results than the Multi-layer Perception model [25]. This suggests that using several specialized classifiers exhibits better motion modelling than using a single model. Second, compared to the hierarchical RNNs [27] that were trained separately on five parts of human skeleton, it appears that considering the whole skeleton and dividing instead the motion into several curvilinear segments is likely to better model motions. Last, our approach appears to improve over all deep LSTMs architectures proposed in [28] which could be due to two specificities of our approach. The fact that our approach processes a motion progressively, i.e. frame by frame, and cumulates the scores along this processing, increases the robustness of the final decision. This decision, which is not necessarily the last observed action class, is in fact made based on multiple detection by several classifiers at different spatio-temporal levels. Furthermore, our approach improves the search of temporal relationship inside a motion by focusing and guiding this

Methods	Acc. (%)
Deep Multi-layer Perceptron, Cho and Chen [25]	95.6
Deep Hierarchical RNN, Du et al. [27]	96.9
Deep LSTM, Zhu et al. [28]	96.8
Deep LSTM + Co-occurrence, Zhu et al. [28]	97.0
Deep LSTM + Simple Dropout, Zhu et al. [28]	97.2
Deep LSTM + In-depth Dropout, Zhu et al. [28]	97.3
Deep LSTM + Co-occurrence + In-depth Dropout, Zhu et al. [28]	97.3
CuDi3D + SVMs + Level = 2	**99.4**

Table 2: Offline action recognition, HDM05: Comparison of the CuDi3D performance with state-of-the-art results on **HDM05 Dataset**.

search on sub-segments of pre-defined curvilinear sizes. This is better than deep LSTMs whose search of such relationships is fully automatic and does not take into account the specificities of the modelled actions to guide this search.

6.2. MSRC-12 dataset

The Microsoft Research Cambridge-12 dataset (MSRC-12) contains sequences of **skeleton data**, represented as 20 joint locations [8]. It includes 12 gestures performed by 30 subjects for a total of 594 sequences (about 50 sequences per class), where a single gesture is performed several times along a sequence. In this **skeleton-based** dataset, participants were provided with instruction to perform the 12 gestures using five displaying modalities including: Images, Text, Video, Images + Text, and Video + Text. The dataset is annotated with action points, which is the time point marking the completion of a gesture.

Experiments on this dataset are conducted in four steps: first, we evaluate our approach on pre-segmented actions according to the protocol proposed in [3] in order to comfort the results of Section 6.1. Secondly, we compare the proposed approach to previous OAD methods using the leave-subjects-out protocol as suggested in many previous papers [8]. Thirdly, we isolate the contribution of curvilinear sliding windows by substituting in our approach the curvilinear windows by regular temporal sliding windows while keeping the other components unchanged. Last, we provide results of other variants of our approach by varying either the number of curvilinear windows or their curvilinear sizes.

The first experiment allows us to compare the performance of our adapted approach to the results reported in [3, 45], in which a 4-fold cross validation experiment was conducted on the MSRC-12 dataset in the context of pre-segmented action recognition. All results are reported in Table 3.

Methods	Acc. (%)
Regression-based classifier, Ellis et al. [3]	88.7
Cov3DJ + SVM + Level = 1, Hussein et al. [45]	89.6
Cov3DJ + SVM + Level = 2, Hussein et al. [45]	90.9
Cov3DJ + SVM + Level = 3, Hussein et al. [45]	91.2
CuDi3D + SVMs + Level = 2	**96.3**

Table 3: Offline action recognition, MSRC-12: Pre-segmented action recognition accuracy results on the **MSRC-12** dataset according to a 4-fold cross validation protocol.

The classification rate reported by Ellis et al. [3] is 88.7%, while the best configuration of Hussein et al. [45] achieved 91.2% accuracy. The proposed approach achieves a higher score namely **96.3%**. In addition to the previous conclusions drawn in Section 6.1, it appears that modelling an action by extracting well designed features is better than only relying on a comparison between the most discriminative canonical poses as suggested by Ellis et al. [3]. In fact, features are able to bring out discriminant information which is often hidden within a pose or a set of poses and not accessible via a simple comparison of these poses. Furthermore, results also show that it is better to use a reduced set of features along with multiple classifiers, which can process a sequence at different spatio-temporal levels, rather than extracting a great number of features but on a reduced number of motion sub-segments as done by Hussein et al. [45].

In the second experiment we evaluate the proposed **CuDi3D** approach for online action detection using a leave-subjects-out protocol. This is the most often used OAD protocol on MSRC-12 dataset and consists in splitting the dataset into 10 disjoint sets. In each set, a group of subjects is removed from the full dataset to obtain the minimum set that contains performances of all gestures. At each iteration, nine sets are used for training and one set is used for testing. The performance of the system is measured in terms of precision and recall. Precision indicates how often a gesture is actually present when the system claims it is. Recall measures how many true ges-

tures are recognized by the system. For each action class, we combine both measures to calculate a single F_{score} defined as:

$$F_{score} = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (11)$$

Because the system detects multiple actions, we used the mean F_{score} over all actions. We finally obtain five F_{score}, one for each modality. Each F_{score} is an average over 10 repetitions and 12 gestures. As the MSRC-12 dataset is annotated with action points, a positive detection is counted if the detection is triggered before 10 frames from the action point. This is a latency of 0.333 seconds as in [8, 9, 5, 19, 20].

Thanks to [45], the MSRC-12 has also been annotated with start/end annotations. We therefore computed the curvilinear size of each classifier. We then built 12 curvilinear classifiers by parsing at each time all the training set with a curvilinear window constrained by the corresponding curvilinear size. Our results, along with those obtained by previous skeleton OAD approaches, are reported in Table 4.

As shown in Table 4, the best performance on this dataset was previously achieved by a simple yet effective OAD approach proposed by Meshry et al. [20]. Their approach, refereed to as ELS, parses the input skeleton flow with a conventional temporal sliding window and searches for the sub-interval with the maximum classifier score. In our approach the goal consists also in looking for a motion segment which best fits a pre-trained segment. Nevertheless, our approach improves over the ELS method given that it operates with curvilinear sliding windows which allows us to better handle the *temporal variabilities*. This allows in fact to reduce false positives as the approach does not emit decisions when no motion is occurring. Besides, the approach always considers motion segments which are consistent with what has been used for training, which increases the recognition ability. These specificities allow to achieve a state-of-the-art performance of **77.1**%. Furthermore, it is important to notice that the gap between the results of our approach and the best ones achieved by previous approaches is very wide according to almost all modalities, except with the *Text* modality (ranging from 2.3% using Images + Text to 12% using Video). In fact, using different modalities increases *inter-* and *intra-class variabilities* and the achieved results suggest that our approach addresses better these variabilities.

For a more detailed view of the detection latency using the **CuDi3D** approach, we provide in Figure 10 a curve that reports the cumulated F_{score} according to the detection instants measured as distances from ground truth action points. As suggested in the testing protocol, only the detections occurring before action point $+10 frames$ are taken into account, which corresponds to a latency of $\Delta = 333ms$. We report however the results for detection occurring beyond this limit in order to show whether the overall detection performance is improved if the latency constraint is relaxed.

One can first notice that most detection occurred around the action point instants, testifying the great precision of our approach and its ability to tackle the execution rate variations. This could be mainly due to the spatial constraints included in the proposed approach. Moreover,

Methods	ELS [19]	RF [8]	RTMS [5]	SSS [9]	ELS [20]	**CuDi3D**
Video + Text	0.645 ± 0.149	0.679 ± 0.035	0.713 ± 0.105	0.707 ± 0.170	0.790 ± 0.133	**0.854** ± 0.067
Images + Text	0.581 ± 0.134	0.563 ± 0.045	0.656 ± 0.122	0.730 ± 0.148	0.711 ± 0.228	**0.753** ± 0.088
Text	0.437 ± 0.170	0.479 ± 0.104	0.521 ± 0.072	**0.713** ± 0.191	0.622 ± 0.246	0.673 ± 0.102
Video	0.580 ± 0.189	0.627 ± 0.052	0.635 ± 0.075	0.557 ± 0.291	0.726 ± 0.225	**0.845** ± 0.079
Images	0.497 ± 0.122	0.549 ± 0.102	0.596 ± 0.103	0.666 ± 0.194	0.670 ± 0.254	**0.731** ± 0.122
Overall	0.548	0.579	0.624	0.675	0.704	**0.771**

Table 4: OAD, MSRC-12: Experimental results for MSRC-12 dataset according to the leave-subjects-out protocol at a latency of $\Delta = 333ms$. Mean F_{score} and its standard deviation is reported for each instruction modality. ELS = Efficient Linear Search; RF = Random Forests; RTMS = Real-Time Multi-Scale; SSS = Structured Streaming Skeleton.

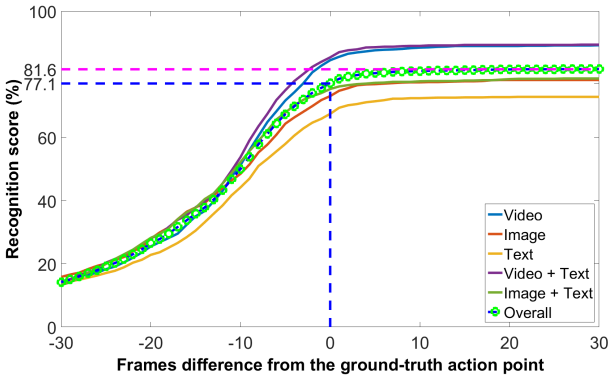

Figure 10: OAD, MSRC-12: Cumulative curve of the detection scores according to the distance from the action point instant. 0 corresponds to the fact that the used action point is the ground truth one.

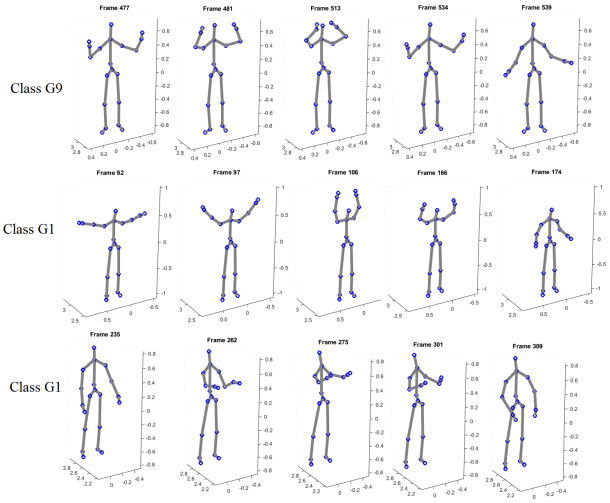

Figure 11: OAD, MSRC-12: Illustration of some failure cases. The first row contains frames of a class G9. The second and the third rows contain frames of the same class G1. In each row the middle frame corresponds to the action point instant of the gesture.

we can see that the detection score achieved according to the leave-subjects-out protocol, namely 77.1%, can be improved if the upper bound limit constraint is relaxed. In fact, the protocol as first proposed in [8] considers that an action detected beyond the action +10 frames is a false positive. However, due to the above mentioned execution rate variations, some approaches could be penalized. We therefore provide as a complementary information the maximum detection score, namely **81.6%**, that our approach is likely to obtain if such constraint is relaxed.

Furthermore, to illustrate some recognition difficulties, two failure cases are presented in Figure 11 (first and bottom rows). To first illustrate the *inter-class variability* that causes a recognition failure, we selected some frames of a class referenced as G9 (first row) and a similar yet different class referenced as G1 (middle row). Even though these classes are labelled differently one can visually see that these two motions are almost similar. Such similarities between two different classes makes it hard to early distinguish between these two classes. Secondly, some frames of another sample of class G1 are reported in the bottom row. These frames are performed by a different subject than those presented in middle row. In this case, one can see that there is a substantial difference between the two samples that belong to the same class. This *intra-class variability* makes it more challenging to recognize correctly such samples in an unsegmented case.

The third experiment aims at highlighting the interest of curvilinear sliding windows compared to conventional temporal sliding windows. We therefore reproduced the same OAD experiment presented previously using a variant of our approach in which curvilinear windows are substituted with temporal sliding windows while keeping all other components unchanged. We used as many temporal sliding windows as there are classes and the same procedure is followed to determine their temporal sizes and also the confidence thresholds $\theta_{i,j}$. Results are reported in Table 5. We also report in this table the **CuDi3D** results to allow comparison.

Methods	Temporal CuDi3D	**CuDi3D**
Video + Text	0.822 ± 0.080	**0.854** ± 0.067
Images + Text	0.713 ± 0.101	**0.753** ± 0.088
Text	**0.679** ± 0.124	0.673 ± 0.102
Video	0.773 ± 0.077	**0.845** ± 0.079
Images	0.664 ± 0.132	**0.731** ± 0.122
Overall	0.730	**0.771**

Table 5: OAD, MSRC-12: Experimental results of a temporal version of CuDi3D and our approach CuDi3D for MSRC-12 dataset according to the leave-subjects-out protocol at a latency of $\Delta = 333ms$.

Based on results reported in Table 5 two main conclu-

sions could be drawn. On the one hand, it appears clearly that the use of curvilinear windows instead of temporal ones allows a significant improvement of detection performances, namely 4%. In particular it suggests that curvilinear windows tackle more efficiently the problem of execution speed variations than using several temporally-based windows as first proposed in [5]. On the other hand, we note that the temporal version of **CuDi3D** is still better than previous temporally-based approaches which suggests that the proposed decision process, including the use of several specialized classifiers, is also an important component and a promising contribution.

In the last experiment conducted on the MSRC-12 dataset we intend to measure the impact of two factors on the approach performance: the number of curvilinear classifiers and their curvilinear sizes. To study the impact of the first factor, we conceived three variants of the **CuDi3D** approach such that: CuDi3D-Avg uses a single curvilinear classifier whose size equals the average of all curvilinear sizes, CuDi3D-Min uses a single curvilinear classifier whose size equals the minimum of all curvilinear sizes and last CuDi3D-Three uses three curvilinear classifiers whose sizes are respectively the minimum, the average and the maximum of all curvilinear sizes. For all these variants the remaining components are unchanged. Results according to a leave-subjects-out protocol are reported in Table 6.

Methods	CuDi3D-Avg	CuDi3D-Min	CuDi3D-Three
Video + Text	0.606 ± 0.101	0.859 ± 0.054	**0.862** ± 0.052
Images + Text	0.556 ± 0.079	0.716 ± 0.059	**0.746** ± 0.075
Text	0.557 ± 0.077	0.633 ± 0.113	**0.651** ± 0.098
Video	0.588 ± 0.075	0.806 ± 0.074	**0.814** ± 0.074
Images	0.535 ± 0.098	0.725 ± 0.100	**0.739** ± 0.099
Overall	0.568	0.748	**0.762**

Table 6: OAD, MSRC-12: Experimental results of three variants of CuDi3D namely CuDi3D-Avg, CuDi3D-Min and CuDi3D-Three obtained on MSRC-12 dataset according to the leave-subjects-out protocol at a latency of $\Delta = 333ms$.

From Table 6 it first appears that the choice of adequate curvilinear sizes is a crucial point. In fact, while both CuDi3D-Avg and CuDi3D-Min approaches uses a single classifier, there is a great difference between their performances. The lower performance of CuDi3D-Avg approach is due to the fact that the used classifier is not launched if the cumulated curvilinear displacement does not exceed the average curvilinear size. Since many actions have curvilinear sizes beyond the average size, these actions are not detected, which causes the low performance. On the contrary, CuDi3D-Min approach uses the shortest curvilinear sliding window and even though it lies on a single classifier it achieves high performance.

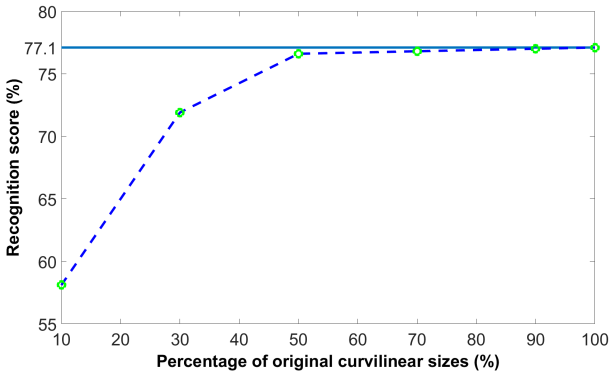

Figure 12: OAD, MSRC-12: Variations of the overall performance achieved with CuDi3D approach depending on the percentage of the curvilinear sizes.

Furthermore, results achieved with the CuDi3D-Three approach show that increasing the number of classifiers from one to three improves the overall performance, but remains lower than the original **CuDi3D** approach composed of all possible curvilinear classifiers. This demonstrates that using several curvilinear classifiers of different curvilinear sizes allows to better handle the spatial variabilities. We also think that the difference in performances achieved by the various approaches and that of **CuDi3D** approach would be greater if the spatial variabilities were more pronounced in the MSRC-12 dataset.

Regarding the second factor impacting the performance of the proposed approach, namely curvilinear sizes, we present in Figure 12 the results of our approach when the curvilinear size of all classifiers is varied from 10 to 100% of their original value (100% corresponds to the original **CuDi3D** approach). As explained in Section 4.2, the curvilinear size of each classifier is determined by averaging the curvilinear displacements of the associated gesture class.

From Figure 12 we can conclude that increasing the curvilinear sizes such that they get closer to the sizes used

in our approach (i.e. 100%) improves the overall performance. This is consistent with the fact that the original curvilinear sizes are computed between the ground-truth start and the action point instant. In fact, reducing these sizes increases the risk of confusing gestures as classifiers start predicting far before the moment at which the action is identifiable.

6.3. Gaming action (G3D) dataset

Further experiments were conducted on the gaming action G3D dataset. G3D aims at allowing the development of new action detection and recognition algorithms for video games [44]. It is a set of 20 gaming actions performed by 10 subjects who were given basic instructions as to how to perform the action. The collected actions are grouped into seven categories: fighting, golf, tennis, bowling, FPS, driving, and miscellaneous actions. Most sequences contain multiple actions in a controlled indoor environment with a fixed camera. Each sequence is repeated three times by each subject.

As in [15, 5, 33, 14] we first followed a leave-subjects-out protocol, in which one subject from the full dataset is removed to get the test set while the larger remaining samples are used for building our model. We reported an average score over 10 runs. Among the seven action categories available in this dataset, we retained the "Fighting" action sequences for which previous approaches reported their results. This category contains 5 classes: kick left, kick right, punch left, punch right and defend. Table 7 summarizes the achieved scores.

As reported, the proposed approach outperforms previous methods and achieves a state-of-the-art score of **98.9%** according to the leave-subjects-out protocol. It is noticeable that the results achieved on this dataset are particularly high given that each class of action activates different body parts inducing a very low *inter-class similarity*.

Methods	**F_score**
DFS + AdaBoost, Bloom et al. [15]	0.919
RTMS + SVM, Sharaf et al. [5]	0.921
RF + ST, Baek et al. [33]	0.948
CAM + DTW, Bloom et al. [14]	0.978
CuDi3D + SVM	**0.989**

Table 7: OAD, G3D: Online action detection results on the Fighting category of the G3D dataset according to the leave-subjects-out protocol. DFS = Dynamic Feature Selection; RTMS = Real-Time Multi-Scale; RF + ST = Random Forests using Spatio-Temporal Contexts; CAM = Clustered Action Manifolds.

A second experiment was then conducted on the G3D dataset. Following [44] we split data by subjects such that the first 5 subjects were used for training and the remaining 5 subjects for testing. The results are shown in Table 8.

Methods	**F_score**
AdaBoost, Bloom et al. [44]	0.585
SVM + SW, Li et al. [32]	0.767
Deep RNN + SW, Li et al. [32]	0.833
Deep CA-RNN, Li et al. [32]	0.940
Deep JCR-RNN, Li et al. [32]	0.962
CuDi3D + SVM	**0.989**

Table 8: OAD, G3D: Results according to the fixed split proposed in [44] on the Fighting category of the G3D dataset. SW = Sliding Window; CA-RNN = Classification Alone Recurrent Neural Network; JCR-RNN = Joint Classification-Regression Recurrent Neural Network.

These results are consistent with previous experiments. Specifically, the proposed **CuDi3D** approach performs better than state-of-the-art methods on the G3D dataset with a detection score of **98.7%** according to this fixed split protocol. Compared to previous sliding windows approaches with fixed sizes such as the ones proposed in [44, 32], the obtained results confirm the superiority of our approach which is much more adapted for handling *temporal variabilities*. These results are particularly important as they testify the superiority of our approach against more complex systems like the deep RNN based ones presented in [32].

6.4. Multi-modal Action Detection (MAD) dataset

We finally evaluate the **CuDi3D** approach on the Multi-Modal Action Detection (MAD) dataset [16]. MAD contains 40 long sequences of 20 subjects (2 sequences per subject) performing 35 actions continuously in each sequence. The length of each sequence is around 2-4 minutes (4000-7000 frames). Skeleton data, consisting in 3D coordinates of 20 joints per frame, are provided along with RGB video (240 × 320) and 3D depth maps (240 × 320). However, only skeleton data are used in our experiments. This dataset is also annotated with start/end annotations.

Similarly to previous approaches, a five-fold-cross-validation over the 20 subjects is used as evaluation protocol (4 subjects per fold). In each iteration, the labelled segments of four folds are used to learn the per class curvilinear displacement thresholds and to train the different spatial classifiers. The remaining sequence in the one fold is used for event detection. For instance, in the first cross-validation, the sequences of the 1st-4th subjects are used for testing ($4 \times 2 = 8$ sequences), and the sequences of the 5th-20th subjects are used for training ($16 \times 2 = 32$ sequences).

The **CuDi3D** approach is evaluated by means of the precision and recall metrics. An action is considered as correctly detected if it overlaps with 50% of the segments of the ground truth action. We compare these two measures along with the F_{score} for existing state-of-the-art OAD methods, namely: MSO-SVM [16], SMMED [16], Naive Bayes (NB) [17] and Motion Segments (MS) [7]. Results among the five folds are reported in Table 9.

Methods	MSO-SVM [16]	SMMED [16]	NB [17]	MS [7]	**CuDi3D**
Precision	0.286	0.574	0.761	0.721	**0.852**
Recall	0.514	0.592	0.736	0.797	**0.820**
F_score	0.368	0.583	0.748	0.757	**0.836**

Table 9: OAD, MAD: Comparison of the **CuDi3D** approach with previous OAD methods on the MAD dataset according to the five-folds protocol. MSO-SVM = Multiclass Structured Output SVM; SMMED = Sequential Max-Margin Event Detectors; NB = Naive Bayes; MS = Motion Segments.

First and foremost, results show that our approach improves over existing skeleton-based OAD methods on the MAD dataset and achieves a new state-of-the-art performance, namely an overall F_{score} of **0.836** which corresponds to an improvement of 7.9%. This is particularly important given that the testing protocol aims at both detecting and segmenting input sequences and also because the MAD dataset is composed of many actions (35 classes) some of which are very similar. Compared to the most efficient method of Devanne et al. [7], which decomposes an action into temporal segments representing elementary motions, our approach seems to be more efficient regarding both precision and recall measures. On the one hand, the higher precision score, which reflects a low number of false detections, might be due to the novel concept of curvilinear sliding window which allows to only consider sequence segments where motion effectively takes place, consequently reducing false positives. On the other hand, the higher recall score, which stands for a high ratio of true detections, is attributable to the decision process which suitably combines the local decision of all classifiers based on a set of thresholds which are automatically adapted and tuned for each classifier and each class.

Figure 13 shows the event-based detection results on the first sequence of the first testing subject. Besides detecting correctly most of the actions in the sequence, the proposed approach succeeds in detecting the beginning and the end of the actions very closely to the ground truth annotations. This might be due to the spatial constraints included in our approach such that each action is characterized by a given curvilinear displacement. However, one can notice that for this sequence, our approach missed a detection at the end of the sequence. It is in fact due to the absence of an idle pause between those two consecutive actions while the detection of an idle pause is supposed to end the ongoing action.

6.5. Real-time operation

The proposed approach is not only able to perform efficient online action detection, but also to do this in real-time. There are two parameters affecting this running-time, namely the number of classifiers and the levels over which features are extracted. Our experiment was conducted on the MSRC-12 dataset during which we used two extraction levels and twelve classifiers. The average running-time per frame for our C++ implementation was measured to be around 1.5ms per frame. The running-

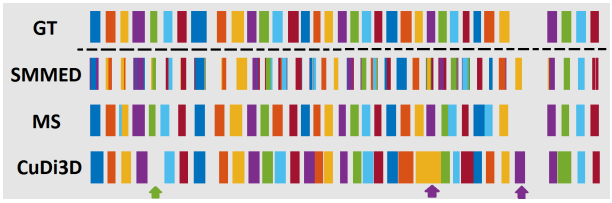

Figure 13: OAD, MAD: Action detection results (sequence-1 of subject-1) for the SMMED method [16] (second row), the MS method (third row) and the proposed **CuDi3D** method (fourth row) in comparison to the ground truth (first row). Each class has a specific colour.

time was measured on a machine with a 2.6 GHz Intel quad-core Core-i7 processor and 16 GB RAM.

In order to have an order of magnitude, we provide the computational cost of some state-of-the-art approaches in Table 10 on the same dataset. The running-time of the ELS approach proposed by Meshry et al. [20] was measured on a machine with 2.2 GHz Intel quad-core Core-i7 processor and 12 GB RAM. The running-time of the RTMS approach [5] used a machine with a memory of 16 GB and a 2.8 GHz Intel quad-core Core-i7 processor. Baek et al. [33] measured the running-time using the 6-core 2.8 GHz CPU with 32 GB of RAM.

ELS [20]	RTMS [5]	RF + ST [33]	**CuDi3D**
10.7 ms	2.7 ms	1.1 ms	1.5 ms

Table 10: OAD, MSRC-12: Comparison of average processing times per frame in milliseconds for different online action detection approaches based on skeleton data. Refer to Table 4 and Table 7 for acronyms.

7. OAD discussion

Before concluding, we would like in this section to discuss three aspects that emerged during our study of the online action detection (OAD) task based on skeleton data. The first aspect concerns the existing benchmark datasets. In fact, most existing action datasets focus on the action recognition tasks for segmented videos. There is a lack of standard large-scale benchmarks that are dedicated to OAD tasks which would allow a more in-depth study. Besides, when evaluating our approach on existing OAD datasets we noted a lack of clarity regarding the followed protocol and the used metrics. In fact, most of the released datasets are unaccompanied by a set of standard evaluation protocols. Thus, if this problem is not noticed by different authors, the validation method used in each work might differ and a direct comparison of their results cannot be made.

Secondly, we think that action detection would be more interesting if subjects could interact with objects around them. It would be also interesting to consider interaction between multiple users. This would broaden the scope of application of existing OAD methods and increase the interaction possibilities offered to users.

Last, we noted a significant difference of performance when evaluating our approach using skeleton data extracted from depth maps, such as MSRC-12 dataset [8], and when using data collected with motion capture systems, such as HDM05 dataset [43]. While using skeleton data allows the development of real-time detection approaches, data inferred from depth maps could be more or less noisy, which is likely to affect the approach performance. Such noisy data are mainly due to parameters such as the distance from the camera or the complexity of the performed gesture. This is even worse if subjects interact with objects as the captured skeletons are likely to collapse. Thus, a method that would keep on tracking subjects even though they are partially occluded would allow the emergence of very efficient skeleton-based OAD approaches.

8. Conclusion and future work

In this paper, we propose an original skeleton-based action detection approach, which better considers the OAD temporal and spatial issues. In particular, regarding the *temporal aspect*, we introduced the concept of curvilinear segmentation. Features extracted according to a curvilinear window encompass motion segments of fixed trajectory length regardless of the time a subject spends at performing this motion. This is fundamentally different from approaches proposed so far as they used temporal sliding windows such that representations are extracted over segments of similar duration. Next, to tackle *inter-class spatial variability*, we propose to launch several curvilinear classifiers to parse the input stream with different curvilinear windows. Last, we propose to tackle *intra-class spatial variability* by means of a fusion framework in which the local decisions and confidence scores are combined. The goal is to detect ongoing actions as early as possible while reducing possible confusions between classes. We also proposed an adapted version of our OAD approach to address the offline action recognition problem.

Experiments conducted on four challenging skeleton-based datasets show the efficiency of our approach in comparison to both existing OAD approaches and existing pre-segmented action recognition approaches. The proposed approach is all the more interesting in that the needed parameters, in particular the confidence thresholds, are automatically optimized from the training data.

Our future work will probably focus on the early recognition task and on action prediction.

References

[1] T. Vieira, R. Faugeroux, D. Martínez, T. Lewiner, Online human moves recognition through discriminative key poses and speed-aware action graphs, Machine Vision and Applications 28 (2017) 185–200.

[2] J. F.-S. Lin, M. Karg, D. Kulić, Movement primitive segmentation for human motion modeling: A framework for analysis, IEEE Transactions on Human-Machine Systems 46 (2016) 325–339.

[3] C. Ellis, S. Z. Masood, M. F. Tappen, J. J. LaViola, R. Sukthankar, Exploring the trade-off between accuracy and observational latency in action recognition, International Journal of Computer Vision 101 (2013) 420–436.

[4] I. Kviatkovsky, E. Rivlin, I. Shimshoni, Online action recognition using covariance of shape and motion, Computer Vision and Image Understanding 129 (2014) 15–26.

[5] A. Sharaf, M. Torki, M. E. Hussein, M. El-Saban, Real-time multi-scale action detection from 3d skeleton data, in: Proceedings of the IEEE Winter Conference on Applications of Computer Vision, 2015, pp. 998–1005.

[6] V. Bloom, V. Argyriou, D. Makris, Linear latent low dimensional space for online early action recognition and prediction, Pattern Recognition 72 (2017) 532–547.

[7] M. Devanne, S. Berretti, P. Pala, H. Wannous, M. Daoudi, A. Del Bimbo, Motion segment decomposition of rgb-d sequences for human behavior understanding, Pattern Recognition 61 (2017) 222–233.

[8] S. Fothergill, H. Mentis, P. Kohli, S. Nowozin, Instructing people for training gestural interactive systems, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2012, pp. 1737–1746.

[9] X. Zhao, X. Li, C. Pang, X. Zhu, Q. Z. Sheng, Online human gesture recognition from motion data streams, in: Proceedings of the 21st ACM international conference on Multimedia, 2013, pp. 23–32.

[10] X. Zhao, X. Li, C. Pang, Q. Z. Sheng, S. Wang, M. Ye, Structured streaming skeleton–a new feature for online human gesture recognition, ACM Transactions on Multimedia Computing, Communications, and Applications 11 (2014) 22.

[11] M. Zanfir, M. Leordeanu, C. Sminchisescu, The moving pose: An efficient 3d kinematics descriptor for low-latency action recognition and detection, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 2752–2759.

[12] M. Devanne, H. Wannous, P. Pala, S. Berretti, M. Daoudi, A. Del Bimbo, Combined shape analysis of human poses and motion units for action segmentation and recognition, in: Proceedings of the 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition, 2015, volume 7, pp. 1–6.

[13] V. Bloom, V. Argyriou, D. Makris, Hierarchical transfer learning for online recognition of compound actions, Computer Vision and Image Understanding 144 (2016) 62–72.

[14] V. Bloom, D. Makris, V. Argyriou, Clustered spatio-temporal manifolds for online action recognition, in: Proceedings of the 22nd IEEE International Conference on Pattern Recognition, 2014, pp. 3963–3968.

[15] V. Bloom, V. Argyriou, D. Makris, Dynamic feature selection for online action recognition, in: Proceedings of the 4th International Workshop on Human Behavior Understanding, 2013, pp. 64–76.

[16] D. Huang, S. Yao, Y. Wang, F. De La Torre, Sequential max-margin event detectors, in: Proceedings of the European conference on computer vision, 2014, pp. 410–424.

[17] H. J. Escalante, E. F. Morales, L. E. Sucar, A naive bayes baseline for early gesture recognition, Pattern Recognition Letters 73 (2016) 91–99.

[18] R. De Geest, E. Gavves, A. Ghodrati, Z. Li, C. Snoek, T. Tuytelaars, Online action detection, in: Proceedings of the European Conference on Computer Vision, 2016, pp. 269–284.

[19] M. Meshry, M. E. Hussein, M. Torki, Linear-time online action detection from 3d skeletal data using bags of gesturelets, arXiv preprint arXiv:1502.01228 (2015).

[20] M. Meshry, M. E. Hussein, M. Torki, Linear-time online action detection from 3d skeletal data using bags of gesturelets, in: Proceedings of the IEEE Winter Conference on Applications of Computer Vision, 2016, pp. 1–9.

[21] X. Liang, Q. Li, X. Zhang, S. Zhang, W. Geng, Performance-driven motion choreographing with accelerometers, Computer Animation and Virtual Worlds 20 (2009) 89–99.

[22] M. Oshita, Motion-capture-based avatar control framework in third-person view virtual environments, in: Proceedings of the 3rd ACM SIGCHI international conference on Advances in computer entertainment technology, 2006, p. 2.

[23] S. Jörg, A. Normoyle, A. Safonova, How responsiveness affects players' perception in digital games, in: Proceedings of the ACM Symposium on Applied Perception, 2012, pp. 33–38.

[24] A. Normoyle, G. Guerrero, S. Jörg, Player perception of delays and jitter in character responsiveness, in: Proceedings of the ACM Symposium on Applied Perception, 2014, pp. 117–124.

[25] K. Cho, X. Chen, Classifying and visualizing motion capture sequences using deep neural networks, in: Proceedings of the IEEE International Conference on Computer Vision Theory and Applications, 2014, volume 2, pp. 122–130.

[26] J. Liu, A. Shahroudy, D. Xu, G. Wang, Spatio-temporal lstm with trust gates for 3d human action recognition, in: Proceedings of the European Conference on Computer Vision, 2016, pp. 816–833.

[27] Y. Du, W. Wang, L. Wang, Hierarchical recurrent neural network for skeleton based action recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1110–1118.

[28] W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, X. Xie, et al., Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks., in: Proceedings of the 30th AAAI Conference on Artificial Intelligence, 2016, volume 2, p. 8.

[29] A. Shahroudy, J. Liu, T.-T. Ng, G. Wang, Ntu rgb+ d: A large scale dataset for 3d human activity analysis, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1010–1019.

[30] Q. Ke, M. Bennamoun, S. An, F. Sohel, F. Boussaid, A new representation of skeleton sequences for 3d action recognition, arXiv preprint arXiv:1703.03492 (2017).

[31] S. Song, C. Lan, J. Xing, W. Zeng, J. Liu, An end-to-end spatio-temporal attention model for human action recognition from skeleton data., in: Proceedings of the 31st AAAI Conference on Artificial Intelligence, 2017, pp. 4263–4270.

[32] Y. Li, C. Lan, J. Xing, W. Zeng, C. Yuan, J. Liu, Online human action detection using joint classification-regression recurrent neural networks, in: Proceedings of the European Conference on Computer Vision, 2016, pp. 203–220.

[33] S. Baek, K. I. Kim, T.-K. Kim, Real-time online action detection forests using spatio-temporal contexts, in: Proceedings of the IEEE Winter Conference on Applications of Computer Vision, 2017, pp. 158–167.

[34] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, R. Moore, Real-time human pose recognition in parts from single depth images, Communications of the ACM 56 (2013) 116–124.

[35] L. L. Presti, M. La Cascia, 3d skeleton-based human action classification: a survey, Pattern Recognition 53 (2016) 130–147.

[36] F. Han, B. Reily, W. Hoff, H. Zhang, Space-time representation of people based on 3d skeletal data: A review, Computer Vision and Image Understanding 158 (2017) 85–105.

[37] S. Y. Boulahia, E. Anquetil, R. Kulpa, F. Multon, Hif3d: Handwriting-inspired features for 3d skeleton-based action recognition, in: Proceedings of the 23rd IEEE International Conference on Pattern Recognition, 2016, pp. 985–990.

[38] A. Sorel, R. Kulpa, E. Badier, F. Multon, Dealing with variability when recognizing user's performance in natural 3d gesture interfaces, International Journal of Pattern Recognition and Artificial Intelligence 27 (2013) 1350023.

[39] S. Y. Boulahia, E. Anquetil, R. Kulpa, F. Multon, 3d multistroke mapping (3dmm): Transfer of hand-drawn pattern representation for skeleton-based gesture recognition, in: Proceedings of the 12th IEEE International Conference on Automatic Face & Gesture Recognition, 2017, pp. 462–467.

[40] R. Kulpa, F. Multon, B. Arnaldi, Morphology-independent representation of motions for interactive human-like animation, in: Computer Graphics Forum, 2005, volume 24, pp. 343–351.

[41] S. Nowozin, J. Shotton, Action points: A representation for low-latency online human action recognition, Microsoft Research Cambridge, Tech. Rep. MSR-TR-2012-68 (2012).

[42] C.-C. Chang, C.-J. Lin, Libsvm: a library for support vector machines, ACM Transactions on Intelligent Systems and Technology 2 (2011) 27.

[43] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, A. Weber, Mocap database hdm05, Institut für Informatik II, Universität Bonn 2 (2007) 7.

[44] V. Bloom, D. Makris, V. Argyriou, G3d: A gaming action dataset and real time action recognition evaluation framework, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2012, pp. 7–12.

[45] M. E. Hussein, M. Torki, M. A. Gowayyed, M. El-Saban, Human action recognition using a temporal hierarchy of covariance descriptors on 3d joint locations, in: Proceedings of the International Joint Conference on Artificial Intelligence, 2013, volume 13, pp. 2466–2472.