Vol.2017-CVIM-207 No.39
2017/5/10

Human Action Recognition-Based Summarization of User-Generated Sports Video

Antonio Tejero-de-Pablos[1,†1,a)] Yuta Nakashima[2] Tomokazu Sato[1] Naokazu Yokoya[1]

Abstract: A vast amount of sports videos are taken by users everyday. These videos are usually long and contain uninteresting parts, so they require summarization. Existing work in sports video summarization leverages various knowledge in application domains, *e.g.*, editing conventions, which are commonly found in broadcast video. However, user-generated videos normally lack any editing, and thus the existing work is ineffective. This thesis approaches the challenge of summarizing user-generated sports video by resorting to the field of human action recognition (HAR). We hypothesize that players' actions can be used as a novel source of semantics to elaborate summaries. We first propose an HAR method with flexible learning that deals with the trade-off between accuracy and flexibility. Then, we propose a user-generated sports video summarization method that extracts HAR features from players' actions. We model the interestingness of the original sequence and extract the highlights of the game.

Keywords: video summarization, human action recognition, sports video, user-generated video, RGB-D camera

1. Introduction

The widespread availability of commercial devices capable of video recording has lead to an ever-growing enormous collection of unedited and unstructured video data generated by users around the world [1], [2]. Among them, sports video appeals to large audiences, being one of the most popular themes. Nowadays users can take sports video with their own devices at public events, professional matches, etc. These user-generated videos are normally lengthy, with a lot of redundant and uninteresting parts, and therefore they require summarization for an easier review. Also, by reducing their size, we facilitate the distribution of the video through different online platforms (*e.g.*, social networks). Nevertheless, manually extracting video highlights, *i.e.*, the most interesting contents of the video, is a very time-consuming task. In order to tackle this problem, the field of automatic video summarization [3] studies techniques to automatically compact the content of a video to facilitate its storage, transmission, browsing, etc. Researchers have studied sports video summarization for decades, and they have proposed several methods for creating a summary with the interesting highlights of a sports game [4], [5], [6]. Most of these methods are specific for broadcast video, since it is edited following sport-specific conventions that are easily detected and can be used to find the highlights of the game. For example, television programs, which are recorded and edited by an expert, feature slow-motion replays, narration, superimposed text, and fixed camera angles that imply a free kick in soccer or a pitch in baseball [7]. Also, some sports like baseball and American football have a certain structure in a game itself [4], [6], which can be also used to extract the moments of greatest interest in a game, and create a summary. For example, in baseball, pitching and batting scenes intertwine in a way that is common to all broadcasts.

However, in contrast to broadcast video, user-generated sports video (UGSV) normally does not follow any convention, and the structure of the sport is not always well defined. The computer vision community has proposed several approaches to understand the content of unstructured video and user-generated video. These approaches range from the traditional clustering of video features that eliminates redundancy, to the most recent works that use deep neural networks to automatically learn features that allow modeling the interesting segments of the video [8], [9]. However, to the best of our knowledge the problem of summarizing UGSV has not been directly tackled to date. In order to approach this problem, we should rely in a source of features that does not depend on any editing convention and yet is also present in UGSV. We, as a novel approach for video summarization, propose to use the *players* as our source of features, more concretely, their *actions*.

In this thesis, we hypothesize that using human action recognition (HAR) techniques we can obtain a representation of the players' actions in a video by which we can model the interesting highlights. For example, a boxing scene showing a parry and an aggressive uppercut might be more interesting than a scene showing a feint or a failed attack. With this idea in mind, we propose a first methodology for which we recorded our own UGSV and HAR datasets using a commercial RGB-D camera. The 3D information provided us with accurate information on the movements of players, but the HAR dataset was not big enough to train current action classifiers. We then came across with challenge of designing a flexible action recognition method that could pro-

[1] Nara Institute of Science and Technology, Ikoma, Nara 630–0101, Japan
[2] Osaka University, Suita, Osaka 565–0871, Japan
[†1] Presently with University of Tokyo, Bunkyo, Tokyo 113–0033, Japan
[a)] antonio-t@mi.t.u-tokyo.ac.jp

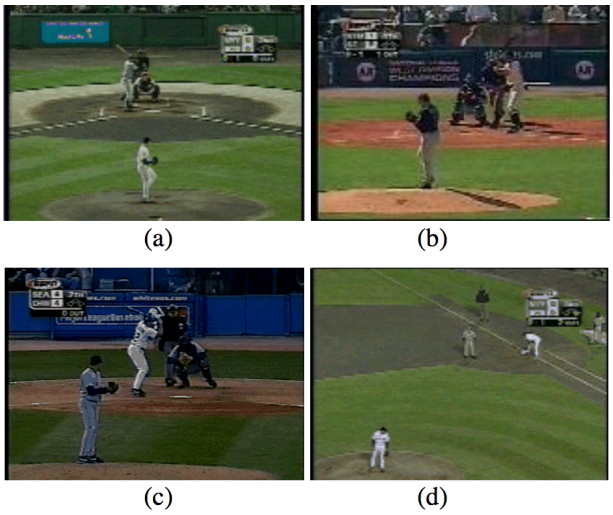

Fig. 1 (a)-(c) A typical start of a regular play - a pitching scene. Other types of starts include a base-stealing scene (d), which is also captured from a fixed camera angle. Obtained from [4].

vide state of the art accuracy without requiring too many training instances. Once we evaluated our method and proved that actions are an effective source of features to generate summaries of UGSV, we explored different ways for extracting features from players' actions. Motivated by the outstanding results of convolutional and recurrent neural networks, the latest fashion in image and video processing, we propose a deep learning-based approach for UGSV summarization that outperforms our previous method.

The contributions of this thesis can be summarized as follows:

- A novel HAR method focused on flexibility, that deals with the trade-off between the recognition accuracy and the computational cost of the learning process (Section 3).
- A novel method for UGSV summarization using a new source of semantics, *i.e.*, applying HAR to RGB-D video sequences of sports that consist of a series of actions (*e.g.*, tennis, boxing, and martial arts) (Section 4).
- An improved method for UGSV summarization, which uses a deep neural network based on a two-stream architecture. We generate summaries using holistic and body joint-based features extracted from players' motion (Section 5).

2. Related work

Video summarization has been studied by several research communities, such as computer vision and multimedia, and there are different ways to group existing methods. In this section, we introduce them in terms of the types of video (*i.e.*, broadcast sports video and user-generated video). This section also reviews existing work for action recognition, which in our method is the key technique for modeling highlights.

2.1 Broadcast sports video summarization

Summarization of sports video focuses on extracting interesting moments, or highlights, of a game. The major approach leverages editing conventions, such as those present in broadcast TV programs. Editing conventions are common to almost all videos of a specific sport and allow automatic methods to extract high-level semantics [7], [10]. Ekin *et al.* [5] proposed to summarize professional soccer games by leveraging the predefined camera angles of broadcast video to detect soccer field elements (*e.g.*, goal posts). Other works detect slow-motion replays to find key events in a game [11], and predefined camera motion patterns to find scenes where players score in basketball/soccer games [12].

Apart from editing conventions, the "play" structure of certain sports also provides high-level semantics for summarization [14]. "Plays" are defined according to the rules of the sport and can often be easily recognized in broadcast video (Fig. 1). For example, Li *et al.* [4] use shot classification to recognize "downs" in American football and they leverage its turn-based structure to generate a summary. Other methods leverage the metadata of edited sports videos [6], [15] because it contains high-level descriptions (*e.g.*, in a baseball video, "hits" may be annotated in the metadata with their timestamps). A downside of all the aforementioned methods is that we cannot apply them to sports video with no editing conventions, structures, and metadata. Also, since they are based on heuristics, they are difficult to generalize to different sports.

Just very few methods have used motion as a non-heuristic feature to generate the summary of a sports video. Mendi *et al.* [16] uses a simple optical flow-based approach for highlight extraction in rugby videos. In a similar fashion, [17] calculates the direction of the variations of the activity level in the color frames to represent how lively the scene changes, and then segment semantically relevant events in broadcast games of soccer, basketball, and tennis. The results are acceptable, but the lack of semantics makes them unable to capture the most interesting highlights of a sports game. In an attempt of performing a more precise semantic analysis, [18] used action recognition on tennis players' actions in combination with editing conventions. However, due to the difficulty of recognizing players' actions from the low-resolution RGB video, they were able to recognize only two classes, left swing and right swing.

2.2 User-generated video summarization

Sports video has a somewhat universal criterion on how interesting a "play" is (for example, a *homerun* in a baseball game should be an interesting play for most viewers). In contrast, user-generated video does not have a clear and universal criterion to identify interesting moments. Moreover, there are no editing conventions nor specific structures that can be used to grab high-level semantics [19]. For this reason, many video summarization methods for user-generated video are designed to reduce the redundancy of lengthy original video rather than finding interesting moments. Traditional methods uniformly sample frames [20] or apply clustering based on low-level features, such as color [8], to extract a brief synopsis of a lengthy video. Since these methods do not extract highlights of the video, researchers have proposed other types of summarization criteria such as important objects [21],attention [22], and interestingness [23].

Recent works use deep neural networks to automatically learn a criterion to model highlights. Yang *et al.* [9] extract features from already summarized videos to train a model for highlight detection. Otani *et al.* [24] use a set of both original videos and their summaries, which were generated via majority voting by annotators, to train a model to find video highlights. Video titles

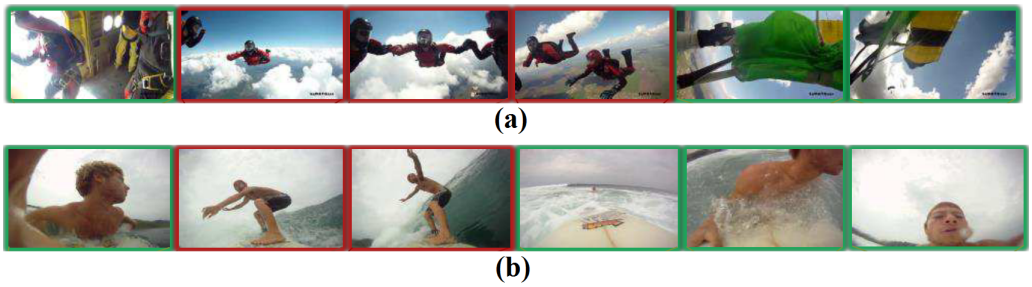

Fig. 2 Most previous work in user-generated video summarization extracts highlights (in red border) based on the general appearance of the scene. Obtained from [13].

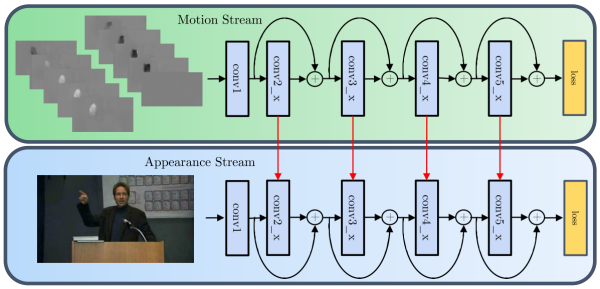

Fig. 3 Example of a two-stream CNN that separately captures appearance and motion. Obtained from [29].

[25] and descriptions [26] have been also used to learn a criterion to generate summaries. Some of these methods employ networks with CNNs and LSTMs, which require a large amount of data for training. Since such huge summarization datasets are not available for researchers, their models are built upon pre-trained networks such as VGG [27] and GoogLeNet [28].

2.3 Action recognition

The highlights extracted by user-generated video summarization methods are based in most cases on the general appearance of the scene (Fig. 2). However, in UGSV not only scenes look very similar, but, according to our hypothesis, the interestingness criterion is directly related to players' actions. This section reviews the state of the art in human action recognition.

Traditional approaches for action recognition use conventional classifiers and hand-crafted features to represent human movement in RGB video [30], [31]. However, the trend of deep neural networks also influenced the field of action recognition. One example are three-dimensional convolutional neural networks (3D CNNs), which are an extension of CNNs applied to images (2D CNNs). While 2D CNNs perform operations only spatially in a single image, 3D CNNs perform operations also temporally, preserving the temporal dependencies among the input video frames [32]. Le *et al.* [33] use a 3D CNN with independent subspace analysis (CNN-ISA) and a support vector machine (SVM) to recognize human actions from video. Also, Tran *et al.* [32] designed a CNN called C3D for extracting video features, which are then fed to an SVM for action recognition.

Another state-of-the-art CNN-based action recognition methods employ two types of streams, a spatial *appearance* stream and a temporal *motion* stream [29], [34]. As shown in Fig. 3, video is decomposed into spatial and temporal components, *i.e.*, into an RGB and optical flow representation of its frames, and fed into two separate CNNs. Each stream separately provides a score for each possible action, and the scores from two streams are later combined for making the final decision. This architecture is supported by the two-stream hypothesis of neuroscience, in which the human visual system would be composed of two different streams in the brain, the dorsal stream (spatial awareness and guidance of actions) and the ventral stream (object recognition and form representation) [35].

Besides RGB video, other work leverages depth maps obtained from commodity depth sensors (*e.g.*, Microsoft Kinect) to estimate human 3D pose for action recognition [36], [37]. The third dimension provides robustness to occlusions and variations in the camera viewpoint.

3. HAR for RGB-D video datasets with a reduced number of instances

Our first approach to sports video summarization is to recognize players' actions in order to model video highlights. However, since we could not find a video dataset containing players' actions specialized for the sport, we decided to create our own dataset for action recognition. As many other self-recorded datasets, ours did not have a number of instances big enough to train a very sophisticated method, although we still needed to recognize actions. For this, we introduce a novel action recognition approach that uses 3D joints estimated from depth maps in RGB-D video. The novelty of our method lies in its flexibility to learn new instances and its capability of recognizing actions even with a reduced number of learned instances. Several methods can take advantage of these benefits, such as applications that need to learn new actions in real-time, or like in our case, applications with a reduced number of training instances. Besides, the use of 3D positions assures more accuracy when recognizing actions, especially those perpendicular to the camera plane.

3.1 Flexible HAR applications

There are a range of HAR-based applications that require learning new actions in runtime. Applications such as customizable gesture interfaces [38], [39] and action databases, either for indexing or retrieval [40], [41], can benefit from such capability, since they are expected to be able to recognize a new type of action right after being input. This kind of applications also does not count with many learning instances [42]. Hence in this thesis we consider the flexibility of approach by two factors:

- Being able to recognize actions even with a very small num-

ber of training instances.

- Being able to learn a certain action class at runtime.

We consider that a method is capable of runtime learning if it does not perform any optimization of the classifier when learning a new action instance. The majority of the previously mentioned works rely on classifiers with a costly learning process that cannot be updated at runtime (*e.g.*, support vector machines) and therefore are not suitable for applications that require adaptive modification of the training model. On the other hand, methods that are capable of runtime learning (*e.g.*, nearest neighbors) allow this, but to the best of our knowledge they have not proved state-of-the-art accuracy yet.

3.2 Flexible HAR using masked 3D joint trajectories

Fig. 4 depicts an overview of our method, which takes a nearest neighbor-based approach to gain flexibility instead of learning a classifier for each action class. We first estimate the 3D joint positions using skeleton tracking from a series of depth map sequences using, *e.g.*, [43], and store them with their action labels as instances of a training dataset. One of the main issues that lead to failure in HAR is concerned with the estimation errors in the skeleton tracking, as stated in [44]. Fortunately, the joint position estimation algorithm provides a confidence value for each joint tracked in each frame. Our method uses it for both learning and recognition stages to alleviate the problem of erroneous skeleton tracking. Then, we prepare an AT for each given action class, which can be viewed as a model of a specific action. Each AT consists of a set of joint trajectories of the action instances belonging to that class along with the confidence values for each joint positions.

At the recognition stage our method tracks the joint trajectories of an unknown action instance in the same way as the learning process, and retrieves its closest instance from the ATs in the database. Since different instances of the same action can be subjected to temporal variations (especially different length and execution speed), we employ a DTW-based distance measure for template matching during the nearest neighbor-based classification.

3.2.1 Action templates learning

To generate an AT, we manually select a number of $J = 15$ different joints from the skeleton tracked in an action instance (*i.e.*, *head*, *neck*, *left shoulder*, *left elbow*, *left hand*, *right shoulder*, *right elbow*, *right hand*, *torso*, *left hip*, *left knee*, *left foot*, *right hip*, *right knee*, *right foot*). Let $\mathbf{p}'_{fj} = (x_{fj}, y_{fj}, z_{fj})^\top$ denote the 3D position of joint j at frame f. Since these positions are in the RGB-D sensor's coordinate system, they can vary from one action instance to another depending on the position of the actor relative to the sensor. For reducing this variability, we transform the joint coordinates so that a certain joint coincides with the origin to improve the robustness against viewpoint variations. In this work we choose the torso as the origin, thus denoting the transformed joint position as $\mathbf{p}_{fj} = \mathbf{p}'_{fj} - \mathbf{p}'_{f\text{torso}}$.

The joint trajectories of all the instances from a certain action class are then aggregated to form an AT. Along with them, the associated confidence values of the tracked positions offered by the joint estimation algorithm of the skeleton tracker [43] are also

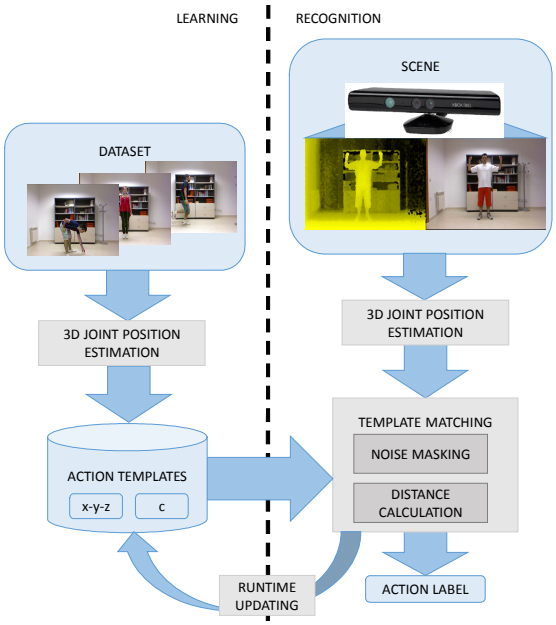

Fig. 4 Overview of our HAR method. The 3D joint positions (x,y,z) along with the confidence value (c) are tracked from the video source to build action templates for each action class. They are used to match new actions and updated at runtime.

included. Let m_i be the action class label for the joint trajectories of the instance i in the training dataset (m_i = *running*, for example), $P_i = \{\mathbf{p}^i_{fj} | f = 1, \ldots, F_i, j = 1, \ldots, J\}$ the corresponding joint trajectories, and $C_i = \{\mathbf{c}^i_{fj} | f = 1, \ldots, F_i, j = 1, \ldots, J\}$ their corresponding confidences, where F_i is the number of frames for action instance i. The AT for action class M is then a set of joint trajectories with their respective confidence values, *i.e.*,

$$A_M = \{(P_i, C_i) | i \text{ s.t. } m_i = M\}. \quad (1)$$

The learning process only requires the generation of ATs.

3.2.2 Action classification

Our recognition process calculates a distance measure to find in our ATs the action instance that is the nearest neighbor of the given unknown instance. Due to the variability in the execution of human actions, naive distance measures are not applicable. For this reason, we employ the use of a DTW-based distance measure, which does not require temporal alignment nor synchronization between a pair of sequences in different sizes [45].

Let $U = \{\mathbf{u}_{fj} | f = 1, \ldots, F_\text{U}, j = 1, \ldots, J\}$ be the joint trajectories of an unknown action instance, with F and J as the total number of frames of the action and the number of joints, respectively. Note that length F_U of an unknown action and length F_i of an action instance in an AT are generally different. The local distance between the positions of joint j in frame f of U and frame f' in P_i is defined as the Euclidean distance as follows:

$$e(\mathbf{u}_{fj}, \mathbf{p}^i_{f'j}) = \|\mathbf{u}_{fj} - \mathbf{p}^i_{f'j}\|_2. \quad (2)$$

Then, using confidence value c_{fj} generated during the tracking we apply a mask to the trajectory of each joint j for each frame f. If this value is smaller than a predefined threshold τ, we determine that that part of the trajectory is not useful for classification. Therefore we assign a binary weight to each point of a joint trajectory by

$$w_{fj} = \begin{cases} 1 & \text{if } c_{fj} \geq \tau \\ 0 & \text{otherwise} \end{cases} . \quad (3)$$

This weighting is applied to the joint positions of both U and P_i. This means only J' out of the J joints are used for frame f, where J' is the number of joints that are not masked ($J' \leq J$). Thus, we define the masked distance between all joint positions $\mathbf{u}_f$ and $\mathbf{p}^i_{f'}$ in frames f and f' as

$$d(\mathbf{u}_f, \mathbf{p}^i_{f'}) = \frac{1}{J'} \sum_{j=1}^{J} e(\mathbf{u}_{fj}, \mathbf{p}^i_{f'j}) w_{fj} w_{f'j}. \quad (4)$$

Using this distance, the DTW-based distance measure between U and P_i is defined as the minimum sum of the local distances over a warping path. Namely, letting $\mathbf{t}_n = (f_n, f'_n)$ be a pair of frames, f for the unknown action instance U and f' for the one in an AT, and $T = \{\mathbf{t}_n | n = 1, \ldots, N\}$ a warping path over which the sum is calculated, the DTW-based distance D is given by

$$D(U, P_i) = \min_T \sum_{(f_n, f'_n) \in T} d(\mathbf{u}_f, \mathbf{p}^i_{f'}) \quad (5)$$

$$\begin{aligned} \text{subject to} \quad & \mathbf{t}_1 = (1,1) \quad \text{and} \quad \mathbf{t}_N = (F_U, F_i) \\ & f_1 = 1 \leq f_2 \leq \cdots \leq f_N = F_U \\ & f'_1 = 1 \leq f'_2 \leq \cdots \leq f'_N = F_i \\ & \mathbf{t}_{n+1} - t_n \in \{(1,0), (0,1), (1,1)\}. \end{aligned} \quad (6)$$

Eq.(5) can be minimized by dynamic programming.

Since the nearest neighbor-based approach needs to compare the distances calculated for action instances of different length, a normalized version of this distance is calculated. The normalizing factor in this case is the length of the warping path T, that is

$$D'(U, P_i) = \frac{1}{N} D(U, P_i). \quad (7)$$

The action class m^* for the unknown action instance U is given as the one whose AT includes an action instance that gives the minimum distance with U, *i.e.*,

$$m^* = m_{i^*} \text{ where } i^* = \arg\min_i D'(U, P_i). \quad (8)$$

3.3 Experimental results

In order to evaluate our approach for generic HAR, we choose datasets containing heterogeneous actions [46] involving the whole body. Here we show the results obtained with the MSR-Action3D dataset (for further details on other datasets refer to [47]).

3.3.1 Implementation details

The recognition algorithm was implemented in Matlab, running in Windows 8 (64 bit), installed in a PC with an Intel Core i7 processor and 16 GB RAM. In addition, for the experiments, we used an empirically determined threshold value $\tau = 0.1$.

3.3.2 MSR-Action3D dataset

This dataset contains 20 different actions performed by up to 10 actors, and the same actor did the same action from one to three times. The actions are: (a) *high arm wave*, (b) *horizontal arm wave*, (c) *hammer*, (d) *hand catch*, (e) *forward punch*, (f) *high throw*, (g) *draw x*, (h) *draw tick*, (i) *draw circle*, (j) *hand clap*, (k) *two hand wave*, (l) *side-boxing*, (m) *bend*, (n) *forward kick*, (o) *side kick*, (p) *jogging*, (q) *tennis swing*, (r) *tennis serve*, (s) *golf swing*, (t) *pickup & throw*. The dataset was built using sequences captured with depth sensors at 15 fps. It provides the 3D position and the tracking confidence of 20 joints per frame, but we kept using 15 joints for our proposed method since we considered the extra five (wrists, ankles, and center hip) do not add much information to the model.

Table 1 Action subdivision of the MSR-Action3D dataset used in the experiments

Subset 1 (SS1)	Subset 2 (SS2)	Subset 3 (SS3)
Horizontal arm wave	High arm wave	High throw
Hammer	Hand catch	Forward kick
Forward punch	Draw x	Side kick
High throw	Draw tick	Jogging
Hand clap	Draw circle	Tennis swing
Bend	Two hand wave	Tennis serve
Tennis serve	Forward kick	Golf swing
Pickup & throw	Side boxing	Pickup & throw

We followed the evaluation methodology employed in previous works [14], [44], [48], [49], and divided the 555 instances into three groups as shown in Table 1. For each group, we conducted a cross-subject experiment in which the actions performed by actors 1, 3, 5, 7, and 9 were used for training and the ones from actors 2, 4, 6, 8, and 10 for testing. The overall recognition accuracy obtained in the experiment was 84.09%. The individual accuracy rates for SS1, SS2, and SS3 are 80%, 78.57%, and 93.69%, respectively. The first two subgroups were more erroneous than the third one. These results are shown in detail in Tables 2, 3, and 4.

Table 2 Confusion matrix (%) for the MSR-Action3D dataset (SS1)

	(b)	(c)	(e)	(f)	(j)	(m)	(r)	(t)
(b)	50	8.33	41.67					
(c)		75	25					
(e)			100					
(f)	18.18	9.09		72.73				
(j)					100			
(m)						46.67		53.33
(r)							100	
(t)							7.14	92.86

Table 3 Confusion matrix (%) for the MSR-Action3D dataset (SS2)

	(a)	(d)	(g)	(h)	(i)	(k)	(l)	(n)
(a)	83.33	8.33	8.33					
(d)	50	16.67	16.67				16.67	
(g)			92.31	7.69				
(h)	20			80				
(i)	26.67		13.33		60			
(k)						100		
(l)		6.66					86.68	6.66
(n)								100

Table 5, obtained partially from [44], shows the generalization performance of our method compared with other state-of-the-art methods that were evaluated against this dataset using the same configuration. The upper part of the table lists the methods that are capable of runtime learning (*e.g.*, NN), and the lower part of the table lists the ones that are not (*e.g.*, SVM). Our method's accuracy outperforms the other HAR methods that are capable of

Table 4 Confusion matrix (%) for MSR-Action3D dataset (SS3)

	(f)	(n)	(o)	(p)	(q)	(r)	(s)	(t)
(f)	81.82			18.18				
(n)		100						
(o)			90.91	9.09				
(p)				100				
(q)					100			
(r)						100		
(s)							100	
(t)					28.57			71.43

runtime learning by far, and is very close to the state-of-the-art methods. As Müller and Röder remarked in [40], recognizing new actions is hard when we generate templates containing noisy instances (MSR-Action3D contains a lot of tracking noise). However, when we apply the confidence value of the skeleton tracker to avoid using the faulty sections in the AT, the recognition performance of our method improves noticeably, as shown in Table 5.

Table 5 Recognition accuracy comparison for the MSR-Action3D dataset

Method	Accuracy	Type
Proposed method	**84.09%**	**Skeleton**
Proposed method (no noise masking)	**79.31%**	**Skeleton**
Rate-invariant Analysis (NN) [49]	63%	Skeleton
Dynamic Temporal Warping [40]	54%	Skeleton
MMTW [50]	92.57%	Skeleton
Joint Movement Similarities [51]	91.2%	Skeleton
HOPC [52]	90.9%	Depth
Rate-invariant Analysis (SVM) [49]	89%	Skeleton
HON4D [48]	88.36%	Depth
Mining Actionlet Ensemble [44]	88.2%	Skeleton
Histograms of 3D joints [36]	78.97%	Skeleton
Action Graph on Bag of 3D Points [14]	74.7%	Depth
Hidden Markov Model [53]	63%	Skeleton
Recurrent Neural Network [54]	42.5%	Skeleton

3.3.3 Flexible HAR

We also evaluated our proposed method's capability of learning new action instances in runtime. We assume a scenario of a customizable gesture interface for a certain application system, in which a command for the system is issued via the gesture interface whose backend is our HAR method. This scenario supposes that the gesture interface has a predefined set of gestures, each of which has a single instance of the corresponding gesture when initialized. The interface learns at runtime; if the interface fails in recognizing an input instance of a gesture correctly, the user specifies the correct label of the instance and the interface includes it to the corresponding AT.

To demonstrate the performance under this scenario, we used the action classes contained in each subset of the MSR-Action3D dataset instead of actual gestures (8 different action classes per subset). We used 20 action instances of each action class in the subset, and divided it into two groups: 10 for learning and 10 for testing. That is, for each subset we use a learning and testing groups of 80 action instances each. At the start, we generate the ATs with a single instance for each class, and then we feed the remaining instances in the learning group one by one (72 instances in total). If our HAR method fails to recognize one instance, it adds that instance to the corresponding AT. We evaluated the accuracy of the method using the test set after an instance in the learning group is input. We repeat this 100 times, randomizing the instances in the learning and testing groups, and the order of

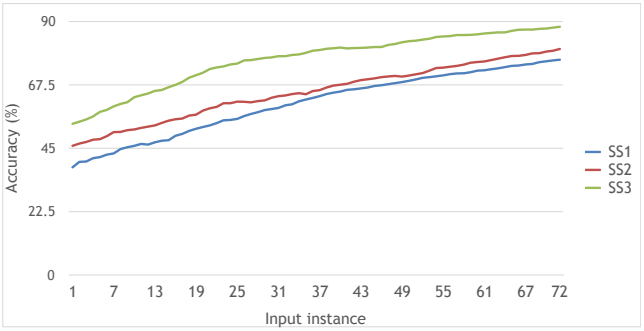

Fig. 5 Recognition accuracy during runtime learning. Horizontal axis: input instances, vertical axis: average recognition accuracy (percentage).

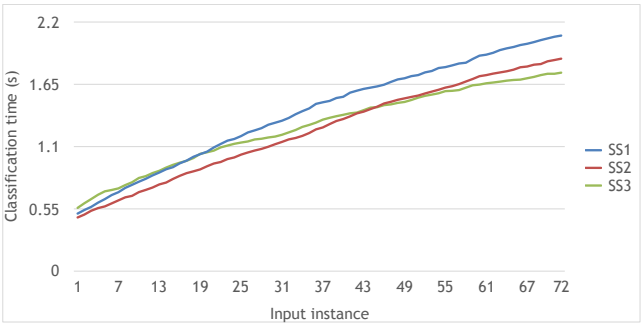

Fig. 6 Total number of instances in the ATs during runtime learning. Horizontal axis: input instances, vertical axis: average number of instances in the ATs.

the input learning instances. The recognition accuracy is the average of all repetitions. We also measured the time required for recognizing the instances in the test set, which is also averaged over the 100 repetitions.

Figure 5 shows the runtime accuracy of our method for each instance in the learning group evaluated against the test group. The final recognition accuracies achieved for subsets SS1, SS2, and SS3 are 75.12%, 79.06%, and 88% respectively, with 37, 35, and 27 instances on average added to the ATs respectively (see Figure 6). By comparing these results to the previous experiment, it can be noticed that our method is able to provide a similar accuracy generating ATs in runtime with less than half the action instances than the previous configuration. It is also remarkable that our method achieves accuracies around 50% with just a single instance per action class. Figure 7 shows the time in seconds spent in classifying one gesture using our implementation. It grows from 0.5 sec to about 2 sec almost linearly as the number of learned instances in our ATs grows.

3.3.4 Discussion

Our experimental results have shown that our approach can be successfully applied for HAR at runtime in depth video datasets with a reduced number of instances. Compared to many related works, we use raw 3D joint trajectories, thereby reducing the computational cost of learning. By applying DTW we gain robustness against variations in execution rates, which heavily affect HAR. Although this methodology is more sensitive to the noise present in the joint position estimation, we manage to effectively alleviate this problem by using the confidence values provided by the skeleton tracker itself. We achieved high recognition rates on a variety of actions, and outperformed other methods

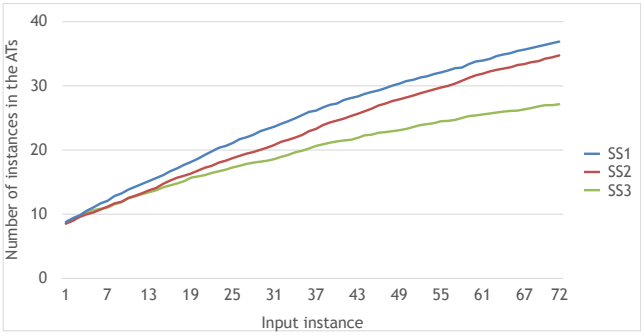

Fig. 7 Classification time for one instance during runtime learning. Horizontal axis: input instances, vertical axis: average classification time (seconds).

capable of runtime learning on the challenging MSR-Action3D dataset.

Compared to the state-of-the-art methods that are not capable of runtime learning, our performance is slightly inferior. We think the reason is that we do not rely on an intricate training phase in order to reduce the cost of learning a new action instance. Besides, our feature set consists of a small number of joint positions tracked in real time, with no other RGB/depth information. Basically, our method deals with a trade-off between flexibility and accuracy in order to allow for runtime learning. For example, Wang and Wu [50] also deal with variations in execution rate of actions using a human joint model. But contrary to our proposed method, their maximum margin temporal warping (MMTW) method relies on a costly SVM algorithm in order to extract the optimal template for the training dataset. Therefore, it can be considered that MMTW is not suitable for runtime learning of new action instances.

Besides, we have proved experimentally that our method offers a great flexibility that would allow users to provide some feedback on wrong classifications or even to add a new action category at runtime. To the best of our knowledge, this feature is not present in the other methods that, in spite of achieving a higher recognition rate in noisy conditions, suffer from a computationally expensive and intricate learning phase demanding a large amount of training data. Also, an AT can contain instances for several ways of performing the same action class (*e.g.*, drinking with your left hand or right hand, gesturing standing up or sitting, etc.), which provides robustness against variations in the way actions are executed. Another example of its flexibility is that, in case of performing action recognition of a specific body part, the number of trajectories used can be easily modified, generating customized ATs with just the joints of interest (hands, legs, etc.). Also, the joint positions contained in an AT itself can be used to reproduce the captured action, which is useful for animation purposes.

4. Summarization of UGSV based on HAR results

In Section 3 we introduced an action recognition method based in template matching of actions that can be applied to recognition problems that do not have a large number of training instances. It works with body joints in 3D estimated from the depth maps

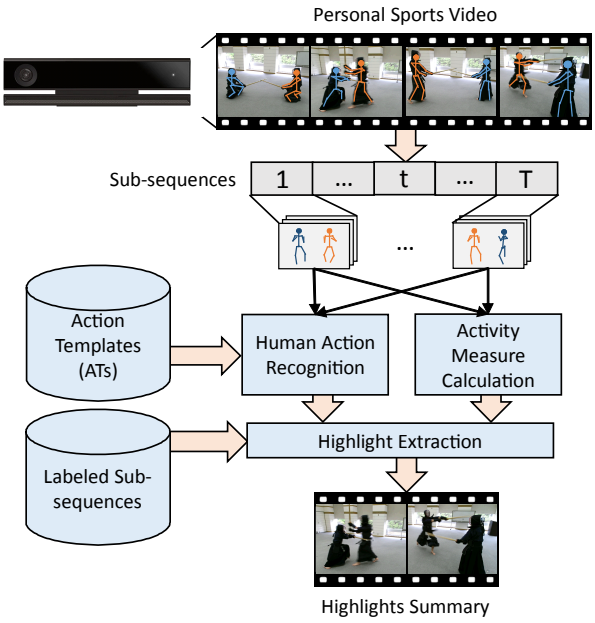

Fig. 8 Overview of our summarization method.

in RGB-D video. Our intention is to use the recognition results of this method to model the interesting parts of a UGSV. As explained in Section 2.1, for user-generated video we cannot use the same methods as other works in sports video summarization, so our novel idea is that the players, a constant element in a sports video of any kind, can be used as a source of features for summarization by recognizing their actions. In order to test our hypothesis, we recorded our own dataset of an example sport (Kendo, or Japanese fencing) using an RGB-D camera. The reason we used depth information is to ensure the actions of the players were properly recognized, so highlights can be modeled better. This section explains the methodology of this approach in detail and the first results ever in HAR-based summarization of UGSV.

4.1 Recognizing players' actions for UGSV summarization

Figure 8 depicts an overview of our method, which takes an RGB-D sports video sequence and generates a summary containing the highlights of the game. The sequence is firstly segmented into T uniform-length (*i.e.*, 3 seconds) sub-sequences. In order to exploit the inherent semantics of the video, we apply our action templates-based HAR method (Section 3) to each sub-sequence. In most sports, multiple players are involved in the game; therefore, HAR is also applied to each player to calculate the dissimilarity between the action of that player in each sub-sequence and each action instance in a predefined set of action classes. We use this dissimilarity and an activity measure, which quantifies the amount of motion in the sub-sequence, to model interesting sub-sequences that are to be included in the resulting highlights summary with a hidden Markov model with Gaussian mixture model emissions (GMM-HMM), which is trained with labeled sub-sequences. Finally the summary is extracted via skimming curve formulation [3] for a given time length L.

4.1.1 HAR via Action Templates

In order to calculate the dissimilarity between the players' actions in the t-th sub-sequence and each of the predefined actions,

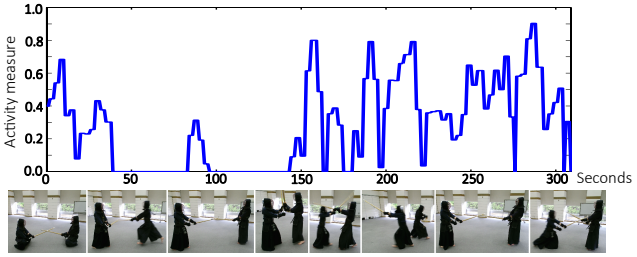

Fig. 9 Activity measure along the course of a Kendo game.

we apply HAR to each player p. From the depth maps in a sub-sequence, we obtain the skeleton (*i.e.*, a set of 3D joint positions) of each player using a skeleton tracker ([43], for example) to gain robustness to view variations with respect to both the camera locations and subject appearances. We use our Action Templates method for HAR, which calculates the distance between the sequence of skeletons of player p in a sub-sequence and each of the action templates (referred to as ATs) in an action dataset.

An AT is a set of action instances (sequences of skeletons) of a predefined action class specialized for the sport. To generate an AT, we extract the skeleton from a depth map sequence that contains one of the predefined actions. Skeleton trackers can also provide a confidence value for each estimated joint position. These positions are transformed to the player's coordinate system, whose origin is at one of the joints (*e.g.*, torso). The sequence of transformed skeletons along with the confidence values form the AT.

For the given t-th input sub-sequence, which may contain multiple players in unknown action classes, we apply a similar process to extract the players' skeletons and transform them into each player's coordinate system. We then calculate the distance between the sequence of skeletons for each player and each of the ATs. Since the duration of an action varies from instance to instance, we adopt dynamic time warping [45] to handle this. In this method, the confidence values are used to filter the noisy sections of the trajectories. Let N denote the number of the predefined actions classes and M the number of action instances per action class. Our HAR method generates a vector $\mathbf{d}_{tp}$ whose n-th element d^n_{tp} is given by $d^n_{tp} = \min_m d^{nm}_{tp}$, where d^{nm}_{tp} is the distance between player p's action in t-th sub-sequence and the m-th AT for the n-th action class ($m = 1, \ldots, M$ and $n = 1, \ldots, N$).

4.1.2 Activity measure

The HAR outputs may not reflect how sudden or prominent the actions are. In [17], they hypothesize that interesting highlights in sports video are characterized by certain patterns in the entropy of the intensities in RGB frames. For each sub-sequence, we use the activity measure of each player's motion based on the entropy of the motion of each joint. For this, we divide the 3D space of the player's coordinate system into V volumes and calculate the ratio r_v of the number of frames in the subsequence in which the joint j of player p fall into volume v. The entropy for joint j is given by

$$e_j = -\sum_{v=1}^{V} r_v \log(r_v). \tag{9}$$

We define the activity measure of a player as

$$a = \sum_{j=1}^{J} e_j, \tag{10}$$

where J is the total number of joints. Figure 9 shows the variation of a along time. The activity measure rises as sudden actions are executed successively, and decreases with repetitive motion (or lack of motion). Sections with zero activity are those where players were not recognized.

For sub-sequence t, we define a feature vector $\mathbf{f}_t^\top = (\mathbf{d}_{t1}^\top, a_{t1}, \mathbf{d}_{t2}^\top, a_{t2}, \ldots, \mathbf{d}_{tP}^\top, a_{tP})$, which is a concatenation of the HAR result $\mathbf{d}_{tp}$ and activity measure a_{tp} for all players, where P is the number of the players in the t-th sub-sequence and a_{tp} is the activity measure for player p.

4.1.3 Highlight extraction

In order to create the summary from the original sequence, we calculate the probability of each sub-sequence of being interesting/non-interesting based on the features, assuming that the segments that are labeled as interesting by users are the highlights of the game. We adopt a GMM-HMM [55] to model interesting/non-interesting segments because adjacent sub-sequences are expected to be highly correlated.

In our method, we assume that the emission probability $\Pr(\mathbf{f}_t|e)$ of $\mathbf{f}_t$ given e follows a Gaussian mixture model, where $e = 1$ indicates that the sub-sequence belongs to an interesting segment and $e = 0$ otherwise. Specifically, the emission probability is given by

$$\Pr(\mathbf{f}_t|e) = \sum_{k=1}^{K} w_{ek} \mathcal{N}(\mathbf{f}_t|\mu_{ek}, \mathbf{\Sigma}_{ek}), \tag{11}$$

where w_{ek}, μ_{ek}, and Σ_{ek} are the mixture weight, the mean, and the covariance matrix of the k-th mixture component for state e. Letting $F = \{\mathbf{f}_t | t = 1, \ldots, T\}$ and $\mathbf{e}^\top = (e_1, \ldots, e_T)$, the probability $\Pr(F_T, \mathbf{e})$ is given by

$$\Pr(F, \mathbf{e}) = \Pr(e_0) \prod_{t=1}^{T} \Pr(e_t|e_{t-1}) \prod_{t=1}^{T} \Pr(\mathbf{f}_t|\mathbf{e}_t, \phi), \tag{12}$$

where $\Pr(e_0)$ is the initial state probability. We can calculate the posterior probability $\Pr(e_t|F)$ using the forward-backward algorithm. Since we have labeled videos for training, the parameters for initial state probability $\Pr(e_1)$ and the transition probability $\Pr(e_t|e_{t-1})$ can be easily determined by counting, and the parameters for GMM (*i.e.*, w_{ek}, μ_{ek}, and Σ_{ek}) can be estimated using the EM algorithm [56].

Once the probabilities are obtained, we generate the summary using skimming curve formulation [3]. Given a certain summary length L in seconds, we apply thresholding to $\Pr(e_t|F)$ by reducing the threshold until we find a set of segments whose total length in seconds is the largest below L. We sort the extracted segments temporally to generate a video summary.

4.2 Experimental results

4.2.1 Implementation details

To evaluate our method, we chose Kendo as an example sport, which is a martial art featuring two players and a set of recognizable actions. Using a Microsoft Kinect v2 sensor, we recorded

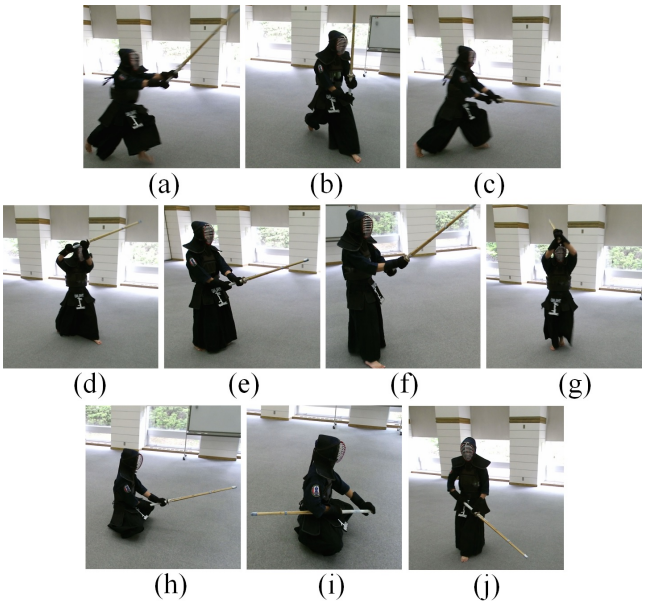

Fig. 10 Actions used in the dataset.

10 RGB-D videos (90 minutes in total), which contain 12 combats. The videos used in the experiments were taken close to the players (2m–4m) for depth map acquisition. We used [43] for skeleton tracking. Apart from these videos, we generated a dataset for HAR, which contains 200 action instances (10 action classes×4 actors×5 repetitions) of action classes (a) *men*, (b) *kote*, (c) *dou*, (d) *bougyo*, (e) *kamae*, (f) *tsubazeriai*, (g) *hikimen*, (h) *sonkyo*, (i) *osametou*, and (j) *aruki*. These actions consist of strikes in different body parts and defense positions. [*1] We evaluated the used HAR method with this dataset. Table 6 shows the recognition results for each action class. The high-speed of the actions and players' clothes hindered HAR, and similar actions were often mistaken. Its generalization performance is evaluated in [47] against the MSRAction3D dataset with the configuration used in [14] (for further details refer to [57]).

Table 6 Confusion matrix of our HAR method over the kendo dataset (%).

		Recognition results									
		(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)
Action classes	(a)	25	20	15	5	25			10		
	(b)		20	30	20	5	10				15
	(c)	15	10	50	5	10			5		5
	(d)	10		5	15	45	25				
	(e)	20			20	40	20				
	(f)	10			35	35	20				
	(g)	20		5				50		25	
	(h)	60	10						30		
	(i)	35				5		10	5	45	
	(j)	50	20	5						10	15

We asked 13 participants to evaluate our method. Since the interestingness of the extracted highlights can differ from one user to another, we grouped them into experienced (*E*) and non-experienced (*NE*) in Kendo, which would affect the results the most. Group *E* has 3 users and group *NE* has 10. In order to train the GMM-HMM for highlight extraction, 3 and 5 users from groups *E* and *NE* were employed as annotators, and assigned interesting/non-interesting labels to the sub-sequences in the 10 original videos. Each sub-sequence was judged to be interesting if two or more annotators labeled it as interesting. Whereas group *E* picked sub-sequences with very specific actions (*e.g.*, very fast strikes, decisive strikes, etc.), group *NE* picked a more general set of actions (*e.g.*, non-decisive strikes, feints, etc.), reaching about twice the number of sub-sequences than group *E*. Again in the LOO fashion, we trained the GMM-HMM with the labels of 9 videos to generate the summary of the remaining.

[*1] A description can be found at https://en.wikipedia.org/wiki/Kendo

4.2.2 GMM-HMM objective evaluation

We evaluated the performance of our trained GMM-HMM by thresholding $\Pr(e_t = 1|F) > 0.5$, and calculating precision (P), recall (R), and f-score (F) metrics for the extracted sub-sequences. Due to the limitations of the capturing device, in some parts of the original video, one or both players were not recognized. For this reason, we evaluated the performance under these conditions: all sub-sequences (A, B) and only the sub-sequences in which both players' skeleton is tracked (C, D). We also evaluated the difference in performance when the activity measure is used (A, C) or not (B, D). Table 7 shows the results. The best results correspond to the case where both players' skeletons were tracked and activity measure was used (C). The effect of including our activity measure is greater on group *E*'s results. Since group *E*'s annotations included more specific actions, it seems the activity measure helps to discern specific interesting actions among similar HAR results. When comparing groups *E* and *NE*, the latter's performance is higher since their annotations contain a broader set of actions.

Table 7 GMM-HMM performance.

	Annot. *E*			Annot. *NE*		
	P	R	F	P	R	F
(A)	0.41	0.44	0.42	0.62	0.76	0.68
(B)	0.39	0.42	0.41	0.62	0.75	0.68
(C)	**0.57**	**0.72**	**0.63**	**0.79**	**0.77**	**0.78**
(D)	0.49	0.64	0.56	0.77	0.75	0.76

4.2.3 Video summary objective evaluation

Our generated summaries are composed of sub-sequences with their estimated labels of interestingness. Human annotators expected that a set of consecutive sub-sequences with interest labels (referred to as highlights, hereinafter) contain an event in a certain granularity. Therefore, even a single missed sub-sequence in the set may distract viewers. For this, we objectively evaluated our method by modifying the definitions of precision and recall to take into account the completeness of the extracted highlights. We define the completeness criterion for an extracted highlight as the fraction of overlap with its associated highlight from the ground truth annotated by our participants. Associating extracted and ground truth highlights is not trivial, and we did this in a greedy manner, in which the total number of overlapping sub-sequences is maximized. We deemed an extracted highlight as a true positive (TP) if it covers over C% of the sub-sequences in the associated ground truth highlight. In this experiment, we thresholded $\Pr(e_t|F_T)$ in the range [0, 1] (instead of 0.5 as in section 4.1) to generate summaries of different lengths.

Figure 11 shows the recall-precision curves produced for C = 50%, 70%, 90%. Whereas almost all highlights with C = 70% reached also C = 90%, when reducing C to 50% the number of

Table 8 Survey results. Each cell consists of the mean ± standard deviation of the subjective scores.

		Summary type			Length			Video		
		Annot. E	Annot. NE	Clust.	20 s	30 s	40 s	(a)	(b)	(c)
Q1	Grp. E	**3.44±0.67**	3.04±0.72	1.89±0.69	3±0.7	**3.56±0.58**	3.17±0.81	**3.61±0.88**	3.11±0.58	3±0.56
	Grp. NE	**3.63±0.5**	**3.63±0.49**	2.26±0.78	3.58±0.46	**3.75±0.43**	3.57±0.61	**3.9±0.54**	3.75±0.35	3.25±0.33
Q2	Grp. E	**3.33±0.58**	3±0.33	1.37±0.35	2.89±0.62	**3.33±0.21**	3.28±0.49	**3.28±0.57**	**3.28±0.39**	2.94±0.49
	Grp. NE	**3.79±0.53**	3.78±0.3	1.88±0.55	3.53±0.5	**3.92±0.32**	3.9±0.36	**4.1±0.29**	3.8±0.24	3.45±0.45
Q3	Grp. E	**3.33±0.33**	3.11±0.58	1.33±0.29	3.11±0.66	**3.33±0.21**	3.22±0.5	**3.33±0.67**	3.22±0.46	3.11±0.27
	Grp. NE	3.57±0.54	**3.68±0.39**	1.92±0.49	3.38±0.48	**3.77±0.38**	3.72±0.49	**3.88±0.48**	3.65±0.31	3.33±0.45
Q4	Grp. E	4.41±0.57	**4.67±0.33**	2.22±0.58	4.44±0.69	**4.61±0.44**	4.56±0.27	**4.72±0.33**	4.61±0.44	4.28±0.57
	Grp. NE	3.6±0.34	**3.62±0.36**	2.27±0.35	3.47±0.41	**3.8±0.27**	3.57±0.29	**3.88±0.32**	3.52±0.25	3.43±0.3

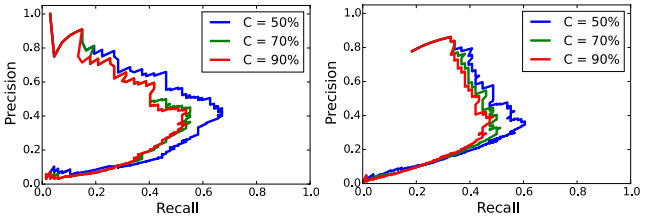

Fig. 11 Recall-precision curves for groups *E* (left) and *NE* (right)

TP increases significantly. We attribute the presence of incomplete segments to the transition probabilities of our GMM-HMM model, which are very low for the *non-interesting to interesting* transition and higher for the *interesting to non-interesting* one. This makes highlights start later and begin earlier than the annotated ground truth. When comparing groups *E* and *NE*, the latter's recall shows a higher and more constant number of TPs for different summary lengths, which is consistent with the results shown in section 4.1. We conclude that our method is able to detect very well certain highlights, but others remain incomplete.

4.3 Video summary subjective evaluation

We assessed the quality and usefulness of our video summaries from the users' point of view by means of a survey. All 13 participants watched the video summaries that, for $C = 70\%$, gave the (a) maximum, (b) median, and (c) minimum f-scores averaged for groups *E* and *NE* in the previous section, as well as their corresponding original video. We also used different summary lengths $L = 20$, 30, and 40 seconds, to see how the length affects viewers' perception. For comparison, besides the summaries created with groups *E* and *NE* annotations, we also evaluated video summaries based on the k-means clustering algorithm as a baseline, in which clustering was performed on our HAR features. As a result, every participant watched 27 summaries.

We asked participants (Q1) if each summary showed an entire action from beginning to end, (Q2) if each summary was interesting, (Q3) if the participant got an insight on the original video by watching the summary, and (Q4) if the summary was not redundant. Table 8 shows the results for each question. Answers are averaged for group *E* and *NE* separately and grouped by the summary type, length, and video. The latter two cover the answers for summaries created with annotations *E* and *NE* together. By looking at the first row, the answers to Q1 show that users were satisfied with the completeness of our summary. Q2 and Q3 also show the user's satisfaction, although group *E*'s rating is slightly lower than group *NE*'s. This is probably because the experienced participants wanted to see all interesting highlights in the summary, but some were missing. The inexperienced participants did not have such a firm predilection. In Q4, group *NE* found the summaries more redundant than group *E*, in a way that group *NE* preferred watching also non-active segments before the action starts for a better understanding of the context.

When comparing summary types, it can be observed that the clustering-based baseline has the lowest scores for all the questions. Overall, group *E* rated the summaries created with their annotations higher, except in Q4. For group *NE*, the difference between summaries generated with their annotations or with group *E*'s is not noticeable. Regarding length, 30 second summaries obtained the best evaluation for all questions and user groups. We consider the reason is that 20 second summaries contained some incomplete highlights that were filled in the 30 second ones, but in the 40 second summary, newly added highlights were incomplete. The summary for video (a) was ranked higher for all questions and both groups, which is coherent since it has the highest f-score.

5. Summarization of UGSV using deep action features

One major drawback of the work introduced in Section 4 is its classic approach: it uses a conventional classifier and handcrafted features for action recognition that require a HAR dataset. The recent trend of deep neural networks has demonstrated the power of feature learning, in which a neural network is trained in an end-to-end manner from its input to the top layers or at least partially from one of its layers to the top. Another interesting direction to extend our previous work is the use of different features. In Section 4, we only use body joint positions as a cue for action recognition. They provide a rich information on players' actions, but miss other potential cues for summarization in the appearance of the scene. At least, appearance is useful when the joint position estimation (*e.g.* [18]) fails.

5.1 Deep neural network for UGSV summarization using two motion streams

In this work, we formulate UGSV summarization as a problem of classifying a video segment in the original video into interesting (and thus included in the summary) or uninteresting. We design a two-stream neural network for this problem and train it in a supervised manner with ground truth labels provided by multiple annotators.

Figure 12 shows an overview of our method. It first divides the input video into video segments $S = \{s_t\}$, in which RGB frames may be accompanied by their corresponding depth maps. A video

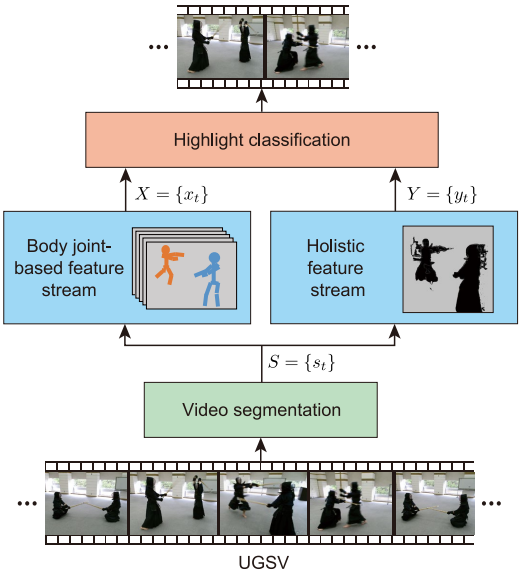

Fig. 12 We feed an LSTM with the body joint positions estimated from players on each frame x_t^f to model their temporal dependencies and extract a feature vector h_t. We also use these body joint positions to calculate an activity measure for all players a_t. Our body joint-based feature vector is the concatenation x_t.

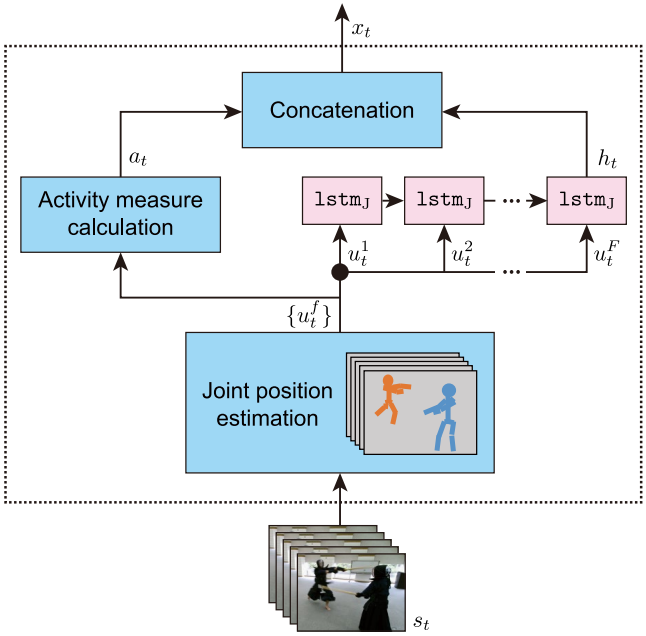

Fig. 13 We feed an LSTM with the body joint positions estimated from players on each frame x_t^f to model their temporal dependencies and extract a feature vector h_t. We also use these body joint positions to calculate an activity measure for all players a_t. Our body joint-based feature vector is the concatenation x_t.

segment s_t is then fed into our two-stream network. The body joint-based feature stream takes RGB frames (and depth maps) in s_t to obtain the body joint-based features x_t, and the holistic feature stream computes holistic features y_t from RGB frames. The former stream captures the players' motion in detail by estimating their body joint positions explicitly. The latter is to represent the entire frames in the video segment, which can be helpful to encode, *e.g.*, the spatial relationship between the players. Our features $X = \{x_t\}$ and $Y = \{y_t\}$ are then fed to the highlight classification block to find the highlights of the input video. This block takes into account the temporal dependencies among the video segments. Our highlight summaries are a concatenation of the segments classified as interesting.

5.2 Video segmentation

Various methods have been proposed to segment a video based on, *e.g.*, its content [17], but in our method we uniformly segment the original input video into multiple segments s_t, *i.e.*, $S = \{s_t | t = 1, \ldots, T\}$, where T is the number of the video segments in S and s_t is the video segment that contains frames from second $t-1$ to second $t+\tau-1$. Since most actions last only a very short period of time, we need short video segments for a finer labeling of highlights. We choose $\tau = 3$ seconds, so adjacent video segments overlap with each other by 2 seconds. Each segment s_t contains a different number of frames, especially when the input video is captured with an RGB-D camera (*e.g.*, Microsoft Kinect), due to automatic exposure control.

5.3 Body joint-based feature stream

For this stream (Fig. 13), in order to obtain a detailed representation of players' actions, we use a sequence of positions of the players' body joints (*e.g.*, head, elbow, etc.) that represent the movement of the players regardless of their appearance. In this work, we employ two types of joint representations, *i.e.*, 3D positions from depth maps or 2D positions from RGB frames.

In the case of 3D body joint positions, we use a skeleton tracker (*e.g.*, [43]) as in [57], which estimates the 3D positions from depth maps. The 3D positions are usually in the camera coordinate system, so they are view-dependent, which introduces extra variations. Therefore, we transform the 3D positions from the camera coordinate system to each player's coordinate system, whose origin is at one of the body joints (*e.g.* torso).

In the absence of depth maps, which is likely in current user-generated video, we can still estimate 2D body joint positions from RGB frames. Recent methods in human pose estimation leverage 2D CNNs to learn the spatial relationships among human body parts and estimate the 2D joint positions [58]. Such 2D positions are not as robust against view variations as 3D positions, but they can be extracted from RGB frames alone without using depth maps. Given the 2D body joint positions, we also transform them to positions relative to the player's coordinate system to make them translation invariant.

As we demonstrated in Section 4, the use of an activity measure works positively when extracting highlights. To calculate the activity measure a, we divide the volume (or plane for the 2D case) around a player into regions and calculate the ratio r_v of the number of frames in the video segment in which the joint j falls into region v. The activity measure a is defined as the entropy obtained based on r_v (Equations 9 and 10). We calculate the activity measure for each player in a segment $(a_{t1} \quad \cdots \quad a_{tQ})$.

We represent joint j of player q in frame f using 3D or 2D relative body joint positions u_{qj}^f in $\mathbb{R}^3$ or $\mathbb{R}^2$ (a row vector). Then, $u_t^f = (u_{11}^f \cdots u_{QJ}^f)_t$ is the concatenation of the body joints of all players in frame f for the video segment s_t, where Q and J are the numbers of players and joints. As shown in Fig. 13 we pass vectors u_t^1 to u_t^F through an LSTM to model the temporal depen-

dencies of the players' body joint positions in s_t. After feeding the last vector x_t^F, we take the hidden state vector h_t of the LSTM as a representation of $\{u_t^f\}$. We reset the state of the LSTM to all zeros before feeding the next video segment. We presume that the number of players Q does not change. However, some players can be out of the field-of-view of the camera. In that case, we pad the corresponding elements in u_t with zeros.

Our method represents a video segment s_t by concatenating the LSTM output and the activity measure of all players in one vector

$$x_t = (h_t \;\; a_t), \tag{13}$$

where a_t is the concatenation of $(a_{t1} \;\; \cdots \;\; a_{tQ})$ and a_{tq} is the activity measure of player q in s_t.

5.4 Holistic feature stream

This stream encodes a video segment s_t in a spatio-temporal representation. We rely on state-of-the-art 3D CNN over RGB frames. Training a 3D CNN from scratch requires thousands of videos [59], which are not available for our task. Recent work on deep neural networks for computer vision [13], [29], [32] show that the activations of an upper layer of a CNN can be used for other related tasks without requiring fine-tuning. Thus, we can instead use 3D CNN whose parameters are pre-trained with large-scale datasets to leverage a huge amount of labeled training data [60]. For example, since we consider that our UGSV summarization task is related to action recognition, we can use a publicly available dataset for action recognition, such as Sports-1M [59].

For this stream, we employ two types of holistic representations of video segments extracted using 3D CNNs, *i.e.*, CNN-ISA [33] or C3D [32]. CNN-ISA provides a representation robust to local translation (*e.g.*, small variations in players' or camera motion) while being selective to frequency, rotation and velocity of such motion. The details of CNN-ISA can be found in [33]. CNN-ISA achieves state-of-the-art performance in well-known datasets for action recognition, such as YouTube [61], Hollywood2 [62] and UCF sports [63]. On the other hand, C3D features provide a representation of objects, scenes and actions in a video. The network architecture and other details can be found in [32]. C3D pre-trained with the Sports-1M dataset achieves state-of-the-art performance on action recognition over the UCF101 dataset [64].

This stream represents video segment s_t using a holistic feature vector y_t, which is the output of one of the aforementioned 3D CNNs.

5.5 Highlight classification using LSTM

Figure 14 shows the network architecture designed to model highlights of UGSV using our features x_t and y_t. We again use an LSTM in order to model the temporal dependencies among video segments, and the network outputs the probability p_t that the video segment s_t is interesting. We first concatenate the features to form vector $z_t = (x_t \;\; y_t)$. Vector z_t then goes through a fully-connected layer to reduce its dimensionality.

We consider that video segments are temporally related to each other; *e.g.*, a skillful boxer first feints a punch before hitting to generate an opening in the defense. Existing work in video summarization uses LSTMs to extract video highlights [9], since it allows to model temporal dependencies across longer time periods than other methods [65]. We follow this idea and introduce a LSTM layer to our network. The hidden state of the LSTM from each time step goes through two fully-connected layers, resulting in a final softmax activation of two units, which correspond to "interesting" and "uninteresting."

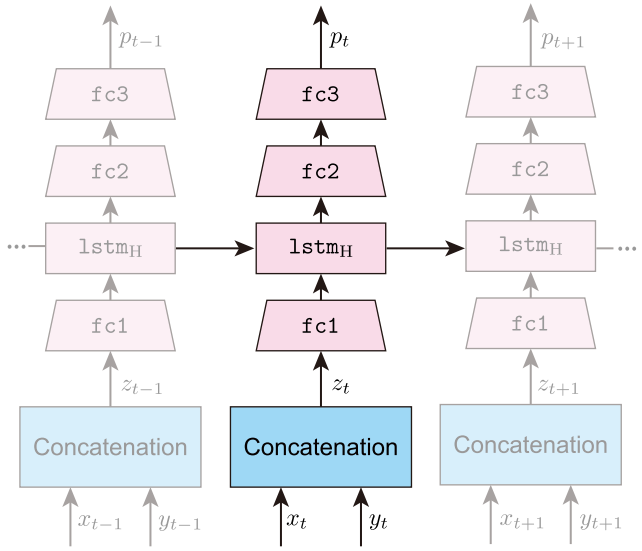

Fig. 14 Neural network architecture for highlight classification, which consists of a single LSTM layer and several fully-connected layers. We feed the body joint-based features x_t and holistic features y_t extracted from video segment s_t to calculate its probability p_t of being interesting.

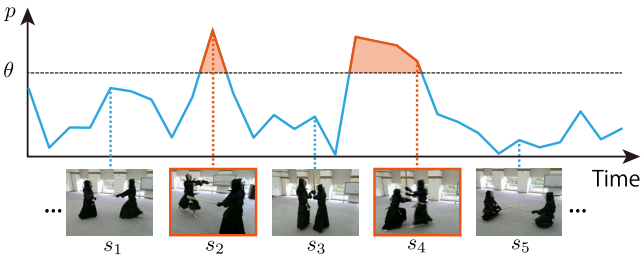

Fig. 15 We generate a summary by concatenating segments whose probability p_t of being highlight surpasses a certain threshold θ. The threshold is chosen to fit the summary length.

Our method provides the control over the length L of the output summary. Therefore, instead of hard decision, we deem the softmax activation of the unit corresponding to "interesting" as the probability p_t of the segment s_t being interesting and apply skimming curve formulation [3] to the sequence of probabilities by decreasing the threshold θ from 1 until it finds a set of segments whose total length is largest below L as shown in Fig. 15. The segments whose probability exceeds θ are concatenated to generate the output summary in the temporal order. In this way, a resulting summary may contain multiple consecutive interesting segments.

5.6 Network training

We use pre-trained CNN in the holistic features stream (*i.e.* CNN-ISA or C3D), whereas we train our LSTMs and fully-connected layers from scratch. That is, during training, the parameters in the holistic feature stream are fixed, and those in the

Table 9 Size of the learnable parameters in our network depending on the features used ($input \times output$). Feature vector sizes are detailed in Section 5.7.1)

	Body joint-based features only			Holistic features only		Body joint-based and holistic features	
	3D joints	2D joints	Action recognition	CNN-ISA	C3D	3D joints + CNN-ISA	2D joints + CNN-ISA
$\texttt{lstm}_J$	90×90	52×52	—	—	—	90×90	52×52
fc1	92×50	54×50	402×400	400×400	4096×400	492×400	454×400
$\texttt{lstm}_H$	50×50	50×50	400×400	400×400	400×400	400×400	400×400
fc2	50×20	50×20	400×100	400×100	400×100	400×100	400×100
fc3	20×2	20×2	100×2	100×2	100×2	100×2	100×2

body joint-based feature stream (*i.e.*, $\texttt{lstm}_J$) and highlight classification (*i.e.*, fc1, $\texttt{lstm}_H$, fc2, and fc3) are updated.

Our UGSV dataset contains video and ground truth labels $l_t \in \{0, 1\}$ for every 1 second, where $l_t = 1$ means that the period from t to $t + 1$ second of the video is "interesting" and $l_t = 0$ otherwise. We assign label l_t to s_t, which covers the frames in $t - 1$ to $t + 2$ since s_t captures the period from t to t_1 second in its center.

For training, we used cross-entropy loss ℓ defined as

$$\ell = \sum l_t \log p_t. \tag{14}$$

5.7 Experiments

In order to evaluate our method, we compare the performance when using different representation of the players' actions. More concretely, we evaluate body joint features only (3D or 2D), holistic motion features only (CNN-ISA or C3D), and the combination of both. Then, we study the completeness of the highlights of the generated summaries. For the subjective evaluation, we surveyed users with and without experience in the sport to study their opinion about our summaries.

5.7.1 Implementation details

We chose Kendo (Japanese fencing) again as an example sport for evaluation, which is a martial art featuring two players and a set of recognizable actions (*e.g.*, attacking and parrying). We extended the UGSV Kendo dataset used in Section 4, which contained 90 minutes of self-recorded Kendo matches divided in 10 RGB-D videos taken with a Microsoft Kinect v2, by adding 18 more self-recorded RGB-D Kendo videos. The total length of our videos is 246 minutes, with a framerate of about 20 fps.

Our body joint-based feature stream was configured for $Q = 2$ players, since Kendo is a two-player sport. We used the tracker in [43] for estimating $J = 15$ 3D body joint positions from depth maps, more specifically: *head*, *neck*, *torso*, *right shoulder*, *right elbow*, *right wrist*, *left shoulder*, *left elbow*, *left wrist*, *right hip*, *right knee*, *right ankle*, *left hip*, *left knee* and *left ankle*. For estimating the 2D positions of the players' joints from the RGB frames, we use the CNN-based method proposed by Linna *et al.* [66]. We initialized the network's parameters with Linna *et al.*'s human pose estimation dataset and fine-tuned it with our extended UGSV Kendo videos. This network provides $J = 13$ joints (same as the 3D case except *neck* and *torso*). Therefore, the size of vector u_t^f is $Q \times J \times 3 = 90$ in the case of 3D positions and $Q \times J \times 2 = 52$ in the case of 2D. Since we made the size of $\texttt{lstm}_J$ the same as the input, and the size of a_t is $Q = 2$, the feature vector for this stream is $x_t \in \mathbb{R}^{92}$ for 3D, or $x_t \in \mathbb{R}^{54}$ for 2D.

For the holistic feature stream, we used either the CNN-ISA [33] or C3D [32] network. Since our UGSV Kendo dataset is not big enough to train these CNNs from scratch, we used networks pre-trained with an action recognition dataset. CNN-ISA was trained in an unsupervised way with the Hollywood2 dataset consisting of 2859 videos [62]. For this network, we followed the configuration in [67] and used a vector quantization representation of the extracted features with a codebook size of 400, resulting in a feature vector $y_t \in \mathbb{R}^{400}$ for each segment s_t. C3D was trained with the Sports-1M dataset [59], which consists of 1.1 million videos of sports activities. We extracted C3D features as indicated in [32] by uniformly sub-sampling 16 frames out of around 60 frames in s_t (the number of frames in s_t may vary for different segments due to the variable framerate of Microsoft Kinect v2) and then extracting the activations from layer fc6 (*i.e.*, $y_t \in \mathbb{R}^{4096}$).

Our method was implemented in Chainer [68]. The learning rate is calculated by the adaptive moment estimation algorithm (Adam) [69] with $\alpha = 0.001$. We introduced sigmoid activation after our fully-connected layers. Table 9 summarizes the number of learnable parameters for each layer, which varies depending on the choice of features.

We invited 15 participants to our experiment and divided them in two groups, experienced (*E*, 5 people) and inexperienced (*NE*, 10 people), according to their experience in the target sport (*i.e.*, Kendo). We considered that the highlights that *E* and *NE* groups prefer would vary greatly from each other, and we wanted to evaluate how well our method adapts to their needs. Then, we asked them to manually annotate the highlights of our 28 videos. We obtained the ground truth labels of our videos for both *E* and *NE* groups separately, considering that each one-second period in video is interesting if 40% of the participants agreed. Due to group *E*'s technical knowledge of Kendo, their highlights contain very specific actions (*e.g.*, decisive strikes, counterattacks). On the other hand, group *NE* selected not only strikes but also more general actions (*e.g.*, parries, feints), so their labeled highlights are almost three times as long as group *E*'s.

We trained our network separately with each group's ground truth labels in the leave-one-out (LOO) fashion, *i.e.*, we used 27 videos for training and generated a summary of the remaining one for evaluation. The CNN for 2D pose estimation was trained independently before each experiment, fine-tuning it with the same 27 videos and estimating the joints of the video used for evaluation. Repeating this process for each video results in 28 experienced summaries and 28 inexperienced summaries. We generated summaries with the same length as the ground truth.

5.7.2 Evaluation by segment f-score

We evaluate the ability of our method to extract highlights in

terms of the f-score. In our method, a one-second period of video is a:

- true positive (TP) if in the summary and $l_t = 1$,
- false positive (FP) if in the summary but $l_t = 0$,
- false negative (FN) if not in the summary but $l_t = 1$, or
- true negative (TN) if not in the summary and $l_t = 0$.

The f-score is then defined as

$$\text{f-score} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{15}$$

Tables 10 and 11 show the f-scores for the summaries generated with the labels of both *E* and *NE* groups. Firstly, Table 10 compares the performance of the feature combinations described in Section 5.7.1. The upper part of the table presents the results of using body joint-based features only. The second part presents the results of using holistic features only. The third part shows the results of using the features from Sections 4.1.1 (including activity measure). We obtained the features by feeding a 3D body joint representation of players' actions to the action recognition method in Section 3, and taking the action classification results. Lastly, the lower part shows the results obtained from the combination of body joint-based and holistic features.

Table 10 F-score comparison of different combinations of features in our method.

Features	Group *E*	Group *NE*
3D joints	0.53	0.83
2D joints	0.45	0.77
CNN-ISA	0.5	0.79
C3D	0.27	0.60
Action recognition (Sec. 4)	0.48	0.76
3D joints + CNN-ISA	**0.58**	**0.85**
2D joints + CNN-ISA	0.57	0.81

Table 11 F-score comparison of out method (3D joints + CNN-ISA) with other UGSV summarization methods.

Method	Group *E*	Group *NE*
Our method	**0.58**	**0.85**
GMM-HMM (Sec. 4)	0.44	0.79
k-means clustering	0.28	0.61

Then, Table 11 compares our proposed architecture with our previous method in Section 4, which uses a Hidden Markov Model with Gaussian mixture emission (GMM-HMM) over the action recognition results mentioned in the previous paragraph, and *k*-means clustering. Such clustering-based method is widely accepted as a baseline for user-generated video summarization [70]. For our *k*-means clustering baseline, we cluster our video segments *S* based on the concatenation *3D joints and CNN-ISA* features and take each cluster centroid. We configured the number of clusters for each video so that the resulting summary length is equal to the ground truth's.

When using a single feature (*i.e.* 3D joins, 2D joints, CNN-ISA, C3D, or action recognition), 3D joints obtain the best performance. CNN-ISA, which uses RGB frames, obtains better results than C3D and even 2D joints. This implies that we can also obtain from RGB frames features that allow us to model UGSV highlights. The drop in performance found between 3D joints and 2D joints may indicate that view variations in the same pose affects negatively our body joint-based features stream. The action recognition feature had an intermediate performance. One reason can be that the action recognition feature is based on a classic approach for classification and some useful cues in 3D body joint positions degenerated in this process. From this result, the features that performed better for highlight classification are CNN-ISA holistic features and 3D body joint-based features.

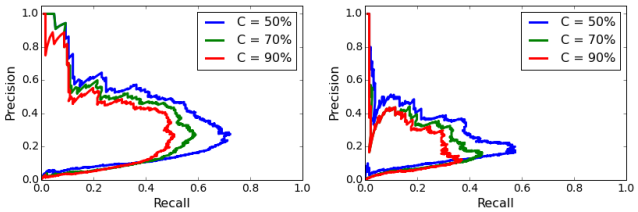

Fig. 16 Recall-precision curves for different completeness values (left: labels *E*, right: labels *NE*). The gap between the completeness $C = 50\%$ and $C = 70\%$ shows that a significant number of our highlights are missing at most half of the interesting segments.

Several state-of-the-art methods in action recognition tasks enjoy a boost in performance by combining handcrafted spatio-temporal features (*e.g.*, dense trajectories) and those learned via CNNs [29], [32]. This is also true in our case, where the combination of CNN-ISA with 3D joints achieves the best performance. The combination of CNN-ISA with 2D joints also provides a considerable boost in performance, especially for the experienced summaries. This confirms our hypothesis that a two-streams architecture also provides better results for UGSV summarization.

Finally, as shown in Table 11, our method outperformed both the previous work and the clustering-based baseline. While clustering allows to show a wider variety of scenes in the summary, this is not a good strategy for UGSV summarization, which follows a different criterion on interestingness. Our proposed method also outperforms the previous work, that used the classification results of an action recognition method and fed them to a GMMHMM for highlight modeling.

Thus, our method outperforms both the highlights model trained on action recognition results and also the feature representation based on action recognition results (Table 10). We can conclude that it is not necessary to explicitly recognize the players' actions for UGSV summarization; it might actually degrade the performance compared to directly using action recognition features.

5.7.3 Evaluation by highlight completeness

As in our previous method (Section 4.2.3), we also evaluated the completeness of our improved UGSV summarization method. We deemed an extracted highlight is a TP if its completeness *c* is greater than a certain percentage *C%*, and according to this we calculated precision and recall of our highlights as

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{16}$$

In this experiment, we moved the threshold θ from 0 to 1 over the probability p_t to generate the recall-precision curve of group *E* and *NE*.

Figure 16 shows the curves produced for $C = 50\%$, 70%, and 90%. We observe that reducing *C* to 50% increases the number of complete highlights significantly. We attribute the presence of

incomplete highlights to our highlight extraction; first the *high p segments* are extracted, and then the highlight is completed with *low p segments* as the threshold θ decreases. But before a highlight is completed, *high p segments* from other highlights are extracted and, in some cases, the *low p segments* are never extracted. In particular, the parts before and after an interesting technique normally correspond to *low p segments*, since they are not present in every ground truth highlight annotated by our participants.

Also, the reason there are more incomplete segments (less TP) in the *NE* summaries is that the inexperienced group annotated a larger number of highlights.

6. Conclusion

This thesis described a novel approach to user-generated sports video (UGSV) summarization. Due to the lack of editing and unstructured nature of user-generated video, we propose using a new source of features for summarization, *i.e.*, the players' actions. In order to recognize players' actions, we generated our own dataset, which has a reduced number of instances. We then also proposed a human action recognition (HAR) method that is effective even with small datasets, and used its recognition results to model the video highlights to generate a summary. A subjective and objective evaluation of our method showed that players' actions can be used to generate summaries adapted to different types of users (*i.e.*, experienced and inexperienced). We also concluded that explicitly recognizing the players' actions is not necessary in order to model the highlights. Thus, we improved our original summarization approach by proposing a deep learning-based method that extracts two types of video features related to the players' actions (body joint-based and holistic) and models the interesting highlights. This method outperformed our previous method and opened a way for future research in UGSV. Our method can be applied to sports that consist of a series of actions and can be recorded from a close distance for an accurate action recognition (*e.g.*, boxing, fencing, and table tennis).

References

[1] Tompkin, J., Kim, K. I., Kautz, J. and Theobalt, C.: Videoscapes: Exploring sparse, unstructured video collections, *ACM Transactions on Graphics*, Vol. 31, No. 4, pp. 68:1–68:12 (2012).

[2] Jiang, Y. G., Dai, Q., Mei, T., Rui, Y. and Chang, S. F.: Super fast event recognition in internet videos, *IEEE Transactions on Multimedia*, Vol. 17, No. 8, pp. 1174–1186 (2015).

[3] Truong, B. T. and Venkatesh, S.: Video abstraction: A systematic review and classification, *ACM Transactions on Multimedia Computing, Communications, and Applications*, Vol. 3, No. 1, pp. 1–37 (2007).

[4] Li, B. and Sezan, M. I.: Event detection and summarization in sports video, *Proc. IEEE Workshop on Content-based Access of Image and Video Libraries*, pp. 132–138 (2001).

[5] Ekin, A., Tekalp, A. M. and Mehrotra, R.: Automatic soccer video analysis and summarization, *IEEE Transactions on Image Processing*, Vol. 12, No. 7, pp. 796–807 (2003).

[6] Nitta, N., Takahashi, Y. and Babaguchi, N.: Automatic personalized video abstraction for sports videos using metadata, *Multimedia Tools and Applications*, Vol. 41, No. 1, pp. 1–25 (2009).

[7] Choi, J., Jeon, W. J. and Lee, S. C.: Spatio-temporal pyramid matching for sports videos, *Proc. ACM International Conference on Multimedia Information Retrieval*, pp. 291–297 (2008).

[8] Lienhart, R. W.: Dynamic video summarization of home video, *Proc. SPIE Electronic Imaging*, pp. 378–389 (1999).

[9] Yang, H., Wang, B., Lin, S., Wipf, D., Guo, M. and Guo, B.: Unsupervised extraction of video highlights via robust recurrent auto-encoders, *Proc. IEEE International Conference on Computer Vision*, pp. 4633–4641 (2015).

[10] Chen, M., Chen, S. C., Shyu, M. L. and Wickramaratna, K.: Semantic event detection via multimodal data mining, *IEEE Signal Processing Magazine*, Vol. 23, No. 2, pp. 38–46 (2006).

[11] Pan, H., Van-Beek, P. and Sezan, M. I.: Detection of slow-motion replay segments in sports video for highlights generation, *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 3, pp. 1649–1652 (2001).

[12] Xu, G., Ma, Y. F., Zhang, H. J. and Yang, S. Q.: An HMM-based framework for video semantic analysis, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 15, No. 11, pp. 1422–1433 (2005).

[13] Zeng, K. H., Chen, T. H., Niebles, J. C. and Sun, M.: Title generation for user generated videos, *Proc. European Conference on Computer Vision*, pp. 609–625 (2016).

[14] Li, W., Zhang, Z. and Liu, Z.: Action recognition based on a bag of 3D points, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9–14 (2010).

[15] Divakaran, A., Peker, K. A., Radhakrishnan, R., Xiong, Z. and Cabasson, R.: Video summarization using mpeg-7 motion activity and audio descriptors, *Video Mining*, pp. 91–121 (2003).

[16] Mendi, E., Clemente, H. B. and Bayrak, C.: Sports video summarization based on motion analysis, *Computers & Electrical Engineering*, Vol. 39, No. 3, pp. 790–796 (2013).

[17] Chen, C. Y., Wang, J. C., Wang, J. F. and Hu, Y. H.: Motion entropy feature and its applications to event-based segmentation of sports video, *EURASIP Journal on Advances in Signal Processing*, Vol. 2008, pp. 1–8 (2008).

[18] Zhu, G., Xu, C., Huang, Q., Gao, W. and Xing, L.: Player action recognition in broadcast tennis video with applications to semantic analysis of sports game, *Proc. ACM International Conference on Multimedia*, pp. 431–440 (2006).

[19] Hua, X. S., Lu, L. and Zhang, H. J.: Optimization-based automated home video editing system, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 14, No. 5, pp. 572–583 (2004).

[20] Mills, M., Cohen, J. and Wong, Y. Y.: A magnifier tool for video data, *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*, pp. 93–98 (1992).

[21] Lu, S., Wang, Z., Mei, T., Guan, G. and Feng, D. D.: A bag-of-importance model with locality-constrained coding based feature learning for video summarization, *IEEE Transactions on Multimedia*, Vol. 16, No. 6, pp. 1497–1509 (2014).

[22] Ma, Y. F., Hua, X. S., Lu, L. and Zhang, H.: A generic framework of user attention model and its application in video summarization, *IEEE Transactions on Multimedia*, Vol. 7, No. 5, pp. 907–919 (2005).

[23] Peng, W. T., Chu, W. T., Chang, C. H., Chou, C. N., Huang, W. J., Chang, W. Y. and Hung, Y. P.: Editing by viewing: Automatic home video summarization by viewing behavior analysis, *IEEE Transactions on Multimedia*, Vol. 13, No. 3, pp. 539–550 (2011).

[24] Otani, M., Nakashima, Y., Tomokazu, S. and Yokoya, N.: Video summarization using textual descriptions for authoring video blogs, *Multimedia Tools and Applications*, pp. 1–19 (2016).

[25] Song, Y., Vallmitjana, J., Stent, A. and Jaimes, A.: Tvsum: Summarizing web videos using titles, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5179–5187 (2015).

[26] Otani, M., Nakashima, Y., Rahtu, E., Heikkilä, J. and Yokoya, N.: Video summarization using deep semantic features, *arXiv preprint arXiv:1609.08758* (2016).

[27] Simonyan, K. and Zisserman, A.: Very deep convolutional networks for large-scale image recognition, *arXiv preprint arXiv:1409.1556*, pp. 1–14 (2014).

[28] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A.: Going deeper with convolutions, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9 (2015).

[29] Feichtenhofer, C., Pinz, A. and Wildes, R.: Spatiotemporal residual networks for video action recognition, *Proc. Conference and Workshop on Neural Information Processing Systems*, pp. 3468–3476 (2016).

[30] Chen, D. Y., Shih, S. W. and Liao, H. Y. M.: Human action recognition using 2-D spatio-temporal templates, *Proc. IEEE International Conference on Multimedia and Expo*, pp. 667–670 (2007).

[31] Calderara, S., Cucchiara, R. and Prati, A.: Action signature: A novel holistic representation for action recognition, *Proc. IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 121–128 (2008).

[32] Tran, D., Bourdev, L., Fergus, R., Torresani, L. and Paluri, M.: Learning spatiotemporal features with 3D convolutional networks, *2015 IEEE International Conference on Computer Vision*, pp. 4489–4497 (2015).

[33] Le, Q. V., Zou, W. Y., Yeung, S. Y. and Ng, A. Y.: Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3361–3368 (2011).

[34] Simonyan, K. and Zisserman, A.: Two-stream convolutional networks for action recognition in videos, *Advances in Neural Information Processing Systems*, pp. 568–576 (2014).

[35] Goodale, M. A. and Milner, A. D.: Separate visual pathways for perception and action, *Trends in Neurosciences*, Vol. 15, No. 1, pp. 20–25 (1992).

[36] Xia, L., Chen, C. C. and Aggarwal, J. K.: View invariant human action recognition using histograms of 3D joints, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 20–27 (2012).

[37] Martínez-Zarzuela, M., Díaz-Pernas, F. J., de Pablos, A. T., González-Ortega, D. and Antón-Rodríguez, M.: Action recognition system based on human body tracking with depth images, *Advances in Computer Science: an International Journal*, Vol. 3, No. 1, pp. 115–123 (2014).

[38] Liu, J., Zhong, L., Wickramasuriya, J. and Vasudevan, V.: uWave: Accelerometer-based personalized gesture recognition and its applications, *Pervasive and Mobile Computing*, Vol. 5, No. 6, pp. 657–675 (2009).

[39] Mistry, P., Maes, P. and Chang, L.: WUW-wear Ur world: A wearable gestural interface, *Proc. CHI Extended Abstracts on Human Factors in Computing Systems*, pp. 4111–4116 (2009).

[40] Müller, M. and Röder, T.: Motion templates for automatic classification and retrieval of motion capture data, *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 137–146 (2006).

[41] Müller, M., Baak, A. and Seidel, H. P.: Efficient and robust annotation of motion capture data, *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 17–26 (2009).

[42] Prekopcsák, Z., Halácsy, P. and Gáspár-Papanek, C.: Design and development of an everyday hand gesture interface, *Proc. ACM International Conference on Human Computer Interaction with Mobile Devices and Services*, pp. 479–480 (2008).

[43] Zhang, Z.: Microsoft Kinect sensor and its effect, *IEEE MultiMedia*, Vol. 19, No. 2, pp. 4–10 (2012).

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

[45] Rabiner, L. and Juang, B. H.: Fundamentals of speech recognition (1993).

[46] Chaquet, J. M., Carmona, E. J. and Fernández-Caballero, A.: A survey of video datasets for human action and activity recognition, *Computer Vision and Image Understanding*, Vol. 117, No. 6, pp. 633–659 (2013).

[47] de Pablos, A. T., Nakashima, Y., Yokoya, N., Díaz-Pernas, F. J. and Martínez-Zarzuela, M.: Flexible human action recognition in depth video sequences using masked joint trajectories, *EURASIP Journal on Image and Video Processing*, Vol. 2016, No. 1, pp. 1–12 (2016).

[48] Oreifej, O. and Liu, Z.: HON4D: Histogram of oriented 4D normals for activity recognition from depth sequences, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 716–723 (2013).

[49] Amor, B. B., Su, J. and Srivastava, A.: Action recognition using rate-invariant analysis of skeletal shape trajectories, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 38, No. 1, pp. 1–13 (2016).

[50] Wang, J. and Wu, Y.: Learning maximum margin temporal warping for action recognition, *Proc. IEEE International Conference on Computer Vision*, pp. 2688–2695 (2013).

[51] Pazhoumand-Dar, H., Lam, C. P. and Masek, M.: Joint movement similarities for robust 3D action recognition using skeletal data, *Journal of Visual Communication and Image Representation*, Vol. 30, pp. 10–21 (2015).

[52] Rahmani, H., Mahmood, A., Huynh, D. Q. and Mian, A.: HOPC: Histogram of oriented principal components of 3D pointclouds for action recognition, *Proc. European Conference on Computer Vision*, pp. 742–757 (2014).

[53] Lv, F. and Nevatia, R.: Recognition and segmentation of 3-D human action using HMM and multi-class AdaBoost, *Proc. European Conference on Computer Vision*, pp. 359–372 (2006).

[54] Martens, J. and Sutskever, I.: Learning recurrent neural networks with Hessian-free optimization, *Proc. International Conference on Machine Learning*, pp. 1033–1040 (2011).

[55] Bilmes, J. A.: A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models, *International Computer Science Institute*, Vol. 4, No. 510, pp. 1–126 (1998).

[56] Moon, T. K.: The expectation-maximization algorithm, *Signal processing magazine*, Vol. 13, No. 6, pp. 47–60 (1996).

[57] de Pablos, A. T., Nakashima, Y., Sato, T. and Yokoya, N.: Human action recognition-based video summarization for RGB-D personal sports video, *Proc. IEEE International Conference on Multimedia and Expo*, pp. 1–6 (2016).

[58] Wei, S. E., Ramakrishna, V., Kanade, T. and Sheikh, Y.: Convolutional pose machines, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3714–3722 (2016).

[59] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R. and Fei-Fei, L.: Large-scale video classification with convolutional neural networks, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1725–1732 (2014).

[60] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S. and Darrell, T.: Caffe: Convolutional architecture for fast feature embedding, *Proc. ACM International Conference on Multimedia*, pp. 675–678 (2014).

[61] Liu, J., Luo, J. and Shah, M.: Recognizing realistic actions from videos "in the wild", *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1996–2003 (2009).

[62] Marszalek, M., Laptev, I. and Schmid, C.: Actions in context, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2929–2936 (2009).

[63] Rodriguez, M. D., Ahmed, J. and Shah, M.: Action MACH a spatio-temporal Maximum Average Correlation Height filter for action recognition, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8 (2008).

[64] Soomro, K., Zamir, A. R. and Shah, M.: UCF101: A dataset of 101 human actions classes from videos in the wild, *CRCV-TR-12-01*, pp. 1–6 (2012).

[65] Ng, J. Y.-H., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R. and Toderici, G.: Beyond short snippets: Deep networks for video classification, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4694–4702 (2015).

[66] Linna, M., Kannala, J. and Rahtu, E.: Real-time human pose estimation from video with convolutional neural networks, *arXiv preprint arXiv:1609.07420*, pp. 1–16 (2016).

[67] Wang, H., Ullah, M. M., Klaser, A., Laptev, I. and Schmid, C.: Evaluation of local spatio-temporal features for action recognition, *Proc. British Machine Vision Conference*, pp. 124–1 (2009).

[68] Tokui, S., Oono, K., Hido, S. and Clayton, J.: Chainer: A next-generation open source framework for deep learning, *Proc. Conference and Workshop on Neural Information Processing Systems*, pp. 1–6 (2015).

[69] Kingma, D. P. and Ba, J. L.: Adam: A method for stochastic optimization, *Proc. International Conference on Learning Representations*, pp. 1–13 (2015).

[70] Cong, Y., Yuan, J. and Luo, J.: Towards scalable summarization of consumer videos via sparse dictionary selection, *IEEE Transactions on Multimedia*, Vol. 14, No. 1, pp. 66–75 (2012).