International Journal of Semantic Computing Vol. 15, No. 2 (2021) 189–213 ©World Scientific Publishing Company DOI: 10.1142/S1793351X21400031



Efficient Retrieval of Human Motion Episodes Based on Indexed Motion-Word Representations

Petra Budikova^{*}, Jan Sedmidubsky[†], Jan Horvath[‡] and Pavel Zezula[§]

> Masaryk University Botanicka 68a, 60200 Brno, Czechia *budikova@fi.muni.cz †xsedmid@fi.muni.cz ‡433736@mail.muni.cz \$zezula@fi.muni.cz

With the increasing availability of human motion data captured in the form of 2D or 3D skeleton sequences, more complex motion recordings need to be processed. In this paper, we focus on similarity-based indexing and efficient retrieval of motion episodes — medium-sized skeleton sequences that consist of multiple semantic actions and correspond to some logical motion unit (e.g. a figure skating performance). As a first step toward efficient retrieval, we apply the motion-word technique to transform spatio-temporal skeleton sequences into compact text-like documents. Based on these documents, we introduce a two-phase retrieval scheme that first finds a set of candidate query results and then re-ranks these candidates with more expensive application-specific methods. We further index the motion-word documents using inverted files, which allows us to retrieve the candidate documents in an efficient and scalable manner. We also propose additional query-reduction techniques that accelerate both the retrieval phases by removing semantically irrelevant parts of the motion query. Experimental evaluation is used to analyze the effects of the individual proposed techniques on the retrieval efficiency and effectiveness.

Keywords: Human motion data; motion episodes; text-based processing; indexing.

1. Introduction

Human motion can be described by a sequence of skeleton poses, where each pose keeps 2D or 3D coordinates of important body joints in a specific time moment. Such spatio-temporal data have enormous application potential in many fields, e.g. in sports to automatically assess a figure-skating performance or detect fouls during a football game without emotions of human referees; in healthcare to remotely evaluate the progress in rehabilitation exercising or to discover movement disorders as indicators for choosing suitable treatments; in security to detect potential threats like a running group of people; or in computer animation to find previously captured animations relevant for building a new movie scene. The motion data have traditionally been captured by specialized hardware technologies, but recent pose-estimation software tools allow us to obtain 2D or even 3D joint positions also from ordinary video data [1, 2]. As a result, we expect an explosion of 2D/3D human skeleton data in the near future, which introduces new challenges to motion data processing.

To make the acquired skeleton data accessible and reusable, we need software tools that can efficiently process various types of user requests. Let us consider a specific example of skeleton data from a figure skating competition: the competition consists of the performances of individual skaters, which are composed of many skating elements (jumps, spins, etc.). Users may be interested in recognizing the type of element performed in some manually selected motion segment, finding all occurrences of a given element in the recording, or finding all performances with similar choreographies. In the first two cases, the query focuses on short atomic motion segments commonly denoted as *actions*, which are the most studied motion type action classification, detection, and retrieval are the objective of many recent research works [3–7]. However, the third example query needs to compare whole skating performances, which are significantly longer and more structured than typical actions. This type of motions, which we denote as *motion episodes*, is natural and occurs in many application domains, but has not received much attention yet.

1.1. Motion episodes

A motion episode, originally introduced in [8], is a complex human activity that consists of multiple actions. Typical examples include a skating performance, dancing performance, workout routine, or a fighting sequence. To search in motion episodes, we can formulate queries that consider mutual similarity of the whole episodes or the occurrences of individual actions and their combinations. Using the figure skating example, a user may search for performances with similar choreographies, performances containing the same skating elements, performances containing at least three very difficult elements, etc.

If the semantic partitioning of episodes is available, i.e. we know which actions are present in the episode and where, all episode search tasks can be solved easily using standard text search on action labels. However, in most cases, we have no information about semantic partitioning of the episode, which is provided as a plain sequence of skeleton poses. Then we have to estimate the semantic similarity of episodes using low-level motion features.

1.2. Our contributions

In this paper, we focus on two fundamental episode-processing tasks: (i) *episode* matching — measuring the mutual similarity of two unsegmented motion episodes and (ii) *episode indexing* — organizing episode data so that they can be efficiently searched using the query-by-example paradigm. The episode matching idea was originally introduced in our work [8] and comprises the identification of significant motion *patterns* that are shared between semantically related episodes.



Fig. 1. Order-sensitive (OS) and order-free (OF) matching among episodes E_1, \ldots, E_4 . Shared actions are illustrated by patterns of the same color.

We distinguish two distinct subtypes of episode matching: *order-free matching*, which measures the similarity of episodes by the amount of shared patterns, and *order-sensitive matching*, which takes into account also the ordering of the shared patterns. The difference between order-free (OF) and order-sensitive (OS) episode matching is depicted in Fig. 1.

We further extend the episode-matching idea by proposing an indexing scheme that enables efficient retrieval of query-relevant episodes. The proposed solution is application-independent and does not require any training data. The main paper highlights can be summarized as follows:

- We analyze the applicability of standard methods developed for short actions and discuss their limitations for processing episode data.
- We introduce an episode matching technique that transforms episode data into a text-like representation of motion words and compares them using information retrieval methods.
- We present a two-phase retrieval model that achieves a high result quality and supports both order-free and order-sensitive episode matching.
- We speed-up both phases of the retrieval model by (i) indexing the motion-word representations using inverted files and applying mature text-processing techniques on the constructed files and (ii) simplifying the episode representations using query-reduction techniques.

2. Problem Analysis

On our way to finding a solution for the episode matching, we first look for inspiration in several related fields of motion data processing. We identify two possible approaches, namely the pose-based and segment-based processing, that will be discussed in detail in the following sections. We also outline the methodology of experimental evaluation and describe the data we use.

2.1. Related work

As mentioned earlier, if the semantic segmentation of actions is known, the episode matching and searching task can be simply solved by text retrieval on the level of action labels. However, a general-purpose semantic segmentation without prior knowledge of training data is hardly achievable. Although there are some solutions [9, 10], they typically discover only the most frequent motion patterns.

Episode matching can also be seen as similar to matching of short actions, for which many deep-learning solutions exist [11]. The action characteristics are often learned using convolutional [12, 13], graph-convolutional [14, 15] or Long Short-Term Memory (LSTM) [16, 17] networks, which can be equipped with different attention-based mechanisms [11]. The widely used LSTM networks are convenient for modeling the temporal dimension of skeleton data but fail in learning dependencies in the spatial domain. On the other hand, the CNNs are successful in learning local spatial characteristics but can hardly learn any latent correlation related to all the joints and further require the fixed-size input, which leads to deformation of at least the temporal dimension of skeleton sequences. Modeling actions in the form of a graph in combination with RNNs or CNNs seems to be effective for learning spatiotemporal dependencies; however, a suitable transformation of skeleton data into some content-preserving graph-like representation is still challenging. Any of the trained neural-network models can be used for extraction of high-dimensional deep features (e.g. 4,096D features in [18]), which can be then compared using the Manhattan or Euclidean distance functions to quantify the similarity between two motion actions. The same approach is in principle applicable to motion episodes; however, the episode data are supposed to be much longer and internally structured as compared to short actions, especially in the case of OF-matching. Therefore, it is questionable whether existing deep-learning solutions can be used for extracting effective episode features.

Another research direction that we can benefit from deals with general representations of skeleton data. Recent papers [19, 20] propose to partition an unsegmented motion into a series of many short segments that are quantized into low-level features in an unsupervised way. This seems to be a promising approach for identifying motion patterns that are shared between two episodes. However, we need to find efficient ways of indexing and comparing the segment-feature sequences, since both the Dynamic Time Warping used in [20] and Earth-Mover's distance in [19] are too expensive and do not support indexing.

2.2. Selected approaches

Unsegmented motion episodes can be principally processed in two modes: we can either work with individual poses or partition each episode into segments that become the basic processing units. We examine both these approaches.

Learning deep features based on the *pose-based* motion representation achieves very good results in recognition of short actions, so it is interesting to find out how a neural network can deal with longer and much more structured episodes. For episode data, we adopt the standard deep-learning principle based on training a neural network on the classification task [11]. Let us emphasize that this approach is limited to situations where labeled episode data are available in advance. Potentially, it would also be possible to apply an unsupervised deep-feature learning, such as the triplet-loss approach used in [21]. However, this would require the existence of some effective matching function for determining the training triplets of similar and dissimilar episodes. As such episode-matching functions are not available yet, we evaluate the supervised-learning approach only.

Nevertheless, we believe that *segment-based* processing is better suited for complex motions. In particular, the segmented data offer more possibilities for efficient data management, and both order-free and order-sensitive matching can be straightforwardly implemented on top of segment-level matching. Consequently, we adopt the segment-based representation [20] to transform episodes into text-like documents, introduce algorithms for efficient episode matching and develop the indexing scheme for scalable episode retrieval.

2.3. Methodology of experimental evaluation

To evaluate both *effectiveness* and *efficiency* aspects of episode retrieval, we consider the k-nearest neighbor (kNN) search scenario and measure the result precision and query processing time. In the rest of this section, we introduce the datasets we use and detail the evaluation metrics.

2.3.1. Data

The test data are taken from the well-known HDM05 dataset [22], which was primarily created for action classification but also contains data useful for episode processing. In particular, the HDM05 dataset provides 241 medium-length skeleton sequences that can be treated as episodes. A single episode takes roughly 40 s on average and is associated with exactly one of 17 different *scenarios* that determine what should be performed (see Table 1 for illustration). The scenarios define both the set of actions to be performed and their ordering. Each scenario is performed in about 14–16 episodes, with the maximum of 19 episodes and the extreme of only 1 episode for one scenario. For interest, the individual scenarios are composed of 130 different kinds of actions. Each episode is captured as a sequence of 3D skeleton poses with 31 tracked joints and the 120 frame-per-second (FPS) rate. As recommended e.g. in [23], we pre-process the dataset by downsampling the episodes to the 12 FPS rate and applying the position, orientation, and skeleton-size normalization.

2.3.2. Datasets

The episode-scenario relationship provides a straightforward way to define a ground truth (GT) for retrieval of similar episodes: for a given query episode, the GT contains all the remaining episodes from the same scenario. However, it is necessary to realize that the HDM05 episodes are constructed in such a way that all episodes from a given scenario always contain the same actions in the same order. If we assume that

Order	Performed actions				
1	Walk 5 steps				
2	Turn around (right)				
3	Walk 5 steps (ducked)				
4	Walk 5 steps (backward)				
5	Walk 5 steps (sideways — right, feet cross front/back)				
6	3 double steps (sideways — left, no feet cross)				
7	3 double steps (sideways — right, feet cross only front)				
8	Walk 5 steps (happily)				
9	Turn around (left)				
10	Walk 5 steps (sadly)				
11	Turn around (right)				
12	Walk 5 steps (creep)				
13	Turn around				
14	Walk 5 steps (shuffle)				
	,				

Table 1. Example of a single HDM05 scenario. Each episode performing this scenario contains these 14 actions in the same order.

episode E_1 in Fig. 1 is the query, the HDM05 dataset only contains episodes of type E_2 and E_4 but never E_3 . Consequently, there is no difference between the results of order-sensitive (OS) and order-free (OF) matching on this data, which makes it impossible to evaluate the order-sensitivity of different episode matching techniques. Therefore, we use the HDM05 data to create the following two test datasets:

- Original dataset it contains the HDM05 episodes and scenarios, and the GT for each query contains the episodes that belong to the same scenario as the query. This allows us to directly compare both OS and OF methods in terms of their ability to identify motion patterns shared between episodes.
- *Mixed dataset* it is created by adjusting the original data as follows: half of the episodes from each scenario are cut in half and the order of the two halves is switched. We remember for each episode the original scenario identifier and whether it is switched. Clearly, the OS and OF matching should return different results, so we need two types of GT: (i) *order-free GT* for a given query contains all episodes belonging to the same scenario as the query, whereas (ii) *order-sensitive GT* contains only half of the respective scenario episodes: either the switched ones if the query is switched or the non-switched ones.

2.3.3. Measuring effectiveness and efficiency

The original and mixed datasets are used to measure the precision of episode matching in the context of kNN queries. For each query episode, we evaluate kNN queries with different values of k, including a special value k^* which corresponds to the number of GT episodes for this query. Each query is evaluated by computing the distance between the query episode and all the other episodes, excluding the exact match. The query precision is computed as the ratio between the number of correctly

identified episodes and the result size k. We ignore one scenario that contains only one episode, so 240 queries are evaluated in total. The overall search precision for a given k is determined as the average precision over the 240 queries.

Retrieval efficiency is quantified by the actual time needed to evaluate all the 240 queries. All the experimental runs are executed on the same hardware to be directly comparable.

3. Pose-Based Episode Matching

In the pose-based approach, we treat episodes as simple sequences of poses and match them as a whole. As a baseline, we implement a naive solution that finds a pose-level alignment of two episodes using the Dynamic Time Warping (DTW). Then, inspired by the success of pose-based methods in action recognition, we train a neural network to extract deep features from the episodes and measure the episode similarity by the Euclidean distance between the deep features.

3.1. Pose-level alignment using DTW

A motion episode $E = (P_1, \ldots, P_l)$ is defined as a sequence of skeleton poses P_i $(1 \le i \le l)$, where each pose $P_i \in \mathbb{R}^{j\cdot 3}$ represents the 3D skeleton configuration estimated in time moment *i* and consists of *xyz*-coordinates of *j* tracked *joints*. The similarity of episodes *E* and *E'* can be naively determined by finding a pose-level alignment of their 3D skeleton sequences using DTW [24]. The DTW approach requires a measure of distance between individual sequence items, i.e. the 3D skeleton poses $P \in E$ and $P' \in E'$ in our case. We define such distance $dist^P(P, P')$ as the sum of the Euclidean distances between the corresponding 3D joint coordinates:

$$dist^{P}(P, P') = \sum_{j=1}^{31} ||P^{j} - P'^{j}||,$$

where P^{j} is the 3D coordinate of the *j*th joint in pose *P*.

Since DTW respects the temporal order of poses, it is clearly suitable only for order-sensitive episode matching. It is also highly inefficient, as it has quadratic complexity and is hardly indexable because it does not satisfy the triangle inequality. On the other hand, DTW utilizes all information contained in the skeleton sequences and gives us a precision baseline for further experiments.

3.2. Deep LSTM features

The DTW approach finds the best pose-level matching but cannot identify any semantic relations between similar episodes. On the other hand, deep neural networks, such as convolutional, recurrent, or graph convolutional architectures, along with different attention-based mechanisms, are known to be successful in identifying semantic relations between short actions [25]. Among many possibilities, we choose

the bidirectional Long-Short Term Memory (Bi-LSTM) network because it focuses on long-term dependencies between motion patterns, which are characteristic for the motion episodes.

We train the Bi-LSTM network for classification of episodes into the predefined set of scenarios. It must be emphasized that this approach is limited to situations when labeled training data — i.e. episodes with scenario labels — are available. During the training phase, the sequence of episode poses is gradually embedded into individual LSTM cells. When the model is trained, deep episode features from the last hidden network layer are extracted and compared by the Euclidean distance to determine their similarity. To achieve a reasonable trade-off between the feature descriptiveness and training time, we have fixed the hidden layer to 1024 dimensions (i.e. 512 dimensions in each of the two network directions), resulting in a 1024D deep feature vector for each episode.

3.3. Experimental evaluation

The pose-level matching is evaluated in retrieval tasks over the *original dataset*, since we are mainly interested in the ability of individual methods to identify shared motion patterns. For the naive solution, we compute the DTW distance between each query episode and each other episode on the level of the 3D skeleton poses. In the case of the LSTM features, we compute the Euclidean distances of the 1024D features extracted from each episode. Since the Bi-LSTM network requires training and test data for learning the model, we split the 240 dataset episodes into two folds and apply the twofold cross-validation procedure. This results in two independent experiments whose kNN precision results are finally averaged.

The precision results for both the naive and LSTM solutions are depicted in Fig. 2. We can observe that the LSTM results are worse for lower values of k, whereas for k^* the LSTM solution is slightly better (76.96% versus 74.84%). From the efficiency point of view, there is a major difference between these two techniques — the



Fig. 2. Pose-based episode searching on the original dataset.

average query processing time is 22.5 s for the naive solution and only 4.2 ms for the LSTM approach.

3.3.1. Discussion

Both the naive and LSTM solutions return quite precise results, but the real applicability of both methods is limited. The naive solution is very sensitive to spatiotemporal variances in the skeleton data and is not usable for order-free matching as it respects the temporal order of poses. Moreover, the retrieval time is not acceptable even for small datasets. The LSTM network can learn the episode semantics to a certain level, but the achieved results are not so superior to the naive solution as in the action recognition task [26]. Additionally, the convergence of the LSTM training is much slower on episode data than on simple actions [26]. Both these issues are probably caused by the limited amount of training data and the complexity of episodes. It is also important to remember that the evaluation uses the *original dataset*, which should be rather easy to learn for the Bi-LSTM network, as the ordering of semantic actions is fixed for each scenario. The *mixed dataset* would be even more challenging and would need significantly more training data to achieve reasonable results.

4. Segment-Based Episode Matching

The segment-based motion processing is an orthogonal approach to the pose-based matching. Each episode is partitioned into a sequence of short segments, which are compared by a segment-level similarity measure. The episode similarity is then evaluated on top of the segment-level matching.

To process segments efficiently, we quantize the segment space into so-called *motion words* (MWs), which have been proposed as compact similarity-preserving representations of skeleton data [20]. The quantization significantly reduces the amount of data to be processed as well as the complexity of similarity evaluations. The MW-based processing should also be able to suppress noisy details in the 3D skeleton data and better identify the motion patterns. Moreover, the MW technique produces text-like representations of motion data, which can be combined with mature text-retrieval methods to implement efficient episode search approaches.

In this section, we first introduce the motion word technique. Then we discuss two approaches to MW-based episode matching — sequence alignment and bag-of-words processing — and compare their ability to identify shared motion patterns on the simple *original dataset*. Finally, we merge these two approaches into a two-phase retrieval that combines the strengths of both techniques and supports both order-sensitive and order-free matching, which is verified by experiments on the more challenging *mixed dataset*.

4.1. Transforming episodes into motion-word documents

We synthetically partition each episode into a sequence of short and overlapping segments and then quantize the segment space into MWs using the technique originally proposed in [20]. The quantization itself can be implemented in several ways, as discussed below. Each MW then provides a concise summary of the skeleton data in a given segment. The whole process is done completely in an unsupervised way, which makes it widely applicable.

Two MWs are compared by a Boolean-valued MW matching function that determines whether these MWs are considered equal: $match^{MW}$: MW × MW → $\{0, 1\}$. The objective of the MW approach is to provide a similarity-preserving transformation from segments to MWs, so that similar segment pairs are with a high probability mapped to matching MWs and dissimilar segment pairs to non-matching MWs. In Fig. 3, we illustrate two types of MWs that are considered in this work: hard MWs and multi-overlay MWs.

4.1.1. Hard MWs

The most straightforward method, denoted as *hard quantization*, computes a single clustering of the segment space, associates each cluster with a one-dimensional identifier, and represents a given segment by the identifier of the respective cluster. The cluster identifier is denoted as the *hard* MW. The matching function on hard MWs returns 1 only if the two MWs are identical. Using the hard MWs, it is possible to transform a 3D skeleton sequence into a sequence of one-dimensional identifiers that can be readily processed by standard text-retrieval techniques. However, the hard quantization suffers quite significantly from a so-called *border problem*: segments that are close in the segment space may be assigned to different clusters and thus considered non-matching in the MW space (e.g. segments s_1 and s_2 in the left part of Fig. 3).



Fig. 3. Transformation of 3D skeleton data into documents of hard or multi-overlay motion words.

4.1.2. Multi-overlay MWs

A promising technique of suppressing the border problem is called *multi-overlay* quantization. The segments are clustered repeatedly, using different techniques or parameter settings, thus creating $n \in \mathbb{N}$ independent *clustering overlays*. A single overlay may incorrectly separate a pair of similar segments, but it is less probable that the same pair is separated by the other independent overlays. The *multi-overlay* MW is defined as an *n*-tuple of cluster identifiers that are assigned to a given segment in individual overlays. The matching function on multi-overlay MWs is parametrized by a variable m and returns 1 if the two multi-overlay MWs agree on at least m positions of the respective *n*-tuples $(m, n \in \mathbb{N}; m \leq n)$ — see the right part of Fig. 3 where the segments s_1 and s_2 are already considered as matching. In general, the multi-overlay MWs achieve a better search quality than hard MWs. However, the multi-overlay MW matching function requires non-trivial adaptations of the text-retrieval methods, which results in less efficient processing.

4.1.3. Representing episodes by MW documents

As depicted in Fig. 3, we apply the MW concept to transform each 3D skeleton episode into a *document* of either hard or multi-overlay MWs. In particular, each episode is cut into 80-frame segments that are shifted by 16 frames. On average, there are 290 such segments per episode. The segment size is much smaller than the size of average semantic action (80 frames versus 260 frames) occurring in episode data, so each action should be covered by a specific subsequence of MWs. To define hard MWs, we use a single clustering of the segment space using the k-means algorithm with k = 350, so there are 350 different hard MWs. The multi-overlay MWs are created by five independent runs of the k-means algorithm with k = 350, resulting in the total number of almost 30,000 distinct multi-overlay MWs. For comparing two multioverlay MWs, we set m = 1 and n = 5, i.e. two multi-overlay MWs are considered matching whenever they are quantized to the same cluster in at least one overlay.

4.2. Sequence alignment on motion-word documents

To compare two episode documents, we can again utilize the DTW sequence alignment. The item-to-item distance function within the DTW is implemented by the specific MW matching function returning two discrete values: 0 or 1. Clearly, the DTW alignment on MWs is again suitable only for order-sensitive matching. We experiment with both hard and multi-overlay MWs and compare their effectiveness and efficiency to the pose-based episode alignment.

4.2.1. Experimental evaluation

We again evaluate each kNN query on the *original dataset* using a sequential scan of all the episode documents. The average retrieval precision with hard and



Fig. 4. Episode searching using DTW sequence alignment with 3D skeleton data, hard MWs and multioverlay MWs on the *original dataset*.

multi-overlay MW representations is depicted in Fig. 4. We can observe that the DTW sequence alignment with hard MWs is clearly worse than the baseline solution with 3D skeleton data. However, with multi-overlay MWs, we are able to achieve a slightly better result than the baseline. This conforms with the findings of [20]: the precision of hard MWs is limited by the border problem, while the multi-overlay MWs are able to suppress it. At the same time, the less-detailed multi-overlay MWs seem to be better suited for episode matching than the original 3D skeletons that force the DTW measure to penalize even slight variations of the same motion pattern.

In terms of processing costs, both MW approaches are about $35 \times$ more efficient than the naive solution. In particular, 0.63 s per query on average is needed for k^*NN search with hard MWs, and 0.67 s with multi-overlay MWs.

4.2.2. Discussion

Motion words provide a compact episode representation, which significantly reduces the data dimensionality and simplifies the distance computations. The utilization of MWs for DTW episode matching allows us to reduce the query processing costs by an order of magnitude. In terms of result quality, the sequence alignment of hard MWs gives worse results than the baseline approach on skeleton data due to the quantization error. On the other hand, multi-overlay MWs can suppress the border problem and are able to detect similar motion patterns better than the original 3D skeleton sequences. However, the properties of the DTW distance function still prevent efficient indexing.

4.3. Bag-of-words processing on hard-MW documents

To achieve fast and scalable MW-based retrieval, we need to employ some indexable distance measure. As mentioned earlier, hard-MW documents have similar properties as standard text, for which efficient indexing models are known. Consequently, we adopt the time-proven *bag-of-MWs* approach, where the original MW sequence is reduced to a set of words, disregarding their ordering. This set is encoded as a sparse vector, where the vector dimension equals the number of available words and the non-zero vector elements correspond to words contained in the respective sequence. In addition, the significance of individual words can be adjusted by assigning weights to the corresponding non-zero vector elements. The dissimilarity of two set vectors is computed by the standard Cosine distance. The bag-of-MWs model does not respect the order of MWs within each episode, so it performs order-free matching. However, the order sensitivity of candidate results can be checked in a post-processing phase.

Noticeably, there is a significant difference between the indexability of hard and multi-overlay MWs. For the hard MWs, the Cosine distance can be evaluated very efficiently and mature indexing techniques such as inverted files can be directly applied to scale to large datasets. However, the comparisons of multi-overlay MWs are more costly and the text indexing cannot be directly applied. Therefore, we only consider hard MWs in this section.

The key factor in applying the bag-of-words model on MW sequences is appropriate weighting of individual MWs. In text retrieval, two statistics are used to estimate the importance of a given term: term frequency (TF) and inverse document frequency (IDF). The term frequency of term t within document D equals the number of times t occurs in D, thus giving more importance to terms that occur repeatedly in a given document. The inverse document frequency of t is computed as a logarithmically scaled inverse fraction of documents containing this term, and its purpose is to suppress the influence of globally frequent words and boost the rare ones.

While we want to capitalize on best-practices of text processing, we need to keep in mind that the MW sequences are different from standard text in several aspects. The episode MW sequences are shorter than a typical text document and the distribution of words is likely to be different. Moreover, the vocabulary of hard MWs is much smaller than natural language vocabulary. And finally, the motion words suffer from the border problem, which may cause problems with the frequency counting. Therefore, we experiment with several variations of the TF-IDF weighting scheme:

- No weighting the weight coefficient of each MW in any episode is 1;
- *TF weighting* the weight of MW *w* within episode *E* is defined by the frequency of *w* within *E*;
- *IDF weighting* the weight of MW *w* is the same across all episodes and is equal to the inverse document frequency of *w*;
- TF-IDF weighting the weight of MW w within episode E is computed as TF · IDF in order to balance the global and local importance of w.

4.3.1. Experimental evaluation

The precision of Cosine-based retrieval with different weighting schemes is again evaluated on the *original dataset*. In Fig. 5, we can observe that the best results are



Fig. 5. Episode searching using the Cosine distance on hard MWs (hMW) with different weighting schemes on the *original dataset*.

achieved with the IDF weighting and slightly outperform the baseline solution that compares raw skeleton data by DTW. The second best option is the equal weighting of all MWs. Whenever the TF is involved, the results are worse. This is probably caused by a combination of two factors. First, the occurrence of very common words that should be treated as stopwords is likely bigger in motion episodes than in text documents. MWs representing non-significant motion parts such as simple standing can be repeated many times in a row, which does not happen with text words. Second, due to the border problem, it is more difficult to identify the common motion words and appropriately decrease their IDF to balance the high TF.

The average processing time of the bag-of-words query is about 65 ms, which is an order of magnitude faster than hard MWs with DTW. Compared to the baseline solution, we can achieve the same precision about 350 times faster.

4.3.2. Discussion

When we represent motion episodes by bags of hard MWs, we obtain indexable data that can be efficiently searched using the Cosine distance. Even with a sequential scan implementation, this representation reduces query processing costs by two orders of magnitude as compared to the naive solution. Moreover, the IDF-weighted retrieval is able to provide slightly more precise result than the baseline. This tells us that the bag-of-words retrieval is less influenced by the border problem than the DTW alignment, and the IDF weighting can successfully identify the most important patterns shared between matching episodes.

4.4. Two-phase episode retrieval

By applying the bag-of-words model on hard MWs, we can build an efficient and scalable search system with a reasonable precision. We also know that better quality can be achieved using the multi-overlay MWs, but their processing is more costly. This leads us to design a two-phase retrieval model that combines the strengths of both MW types and can also be used for both order-sensitive and order-free retrieval.

In particular, we first efficiently identify a set of *candidate documents* C, using the Cosine distance on IDF-weighted sets of hard MWs. Then, we *refine* the candidate set by comparing each candidate document to the query document using more expensive computations with multi-overlay MWs and return the k best-ranked episode documents as the query result. To allow both order-sensitive and order-free matching, we define two refinement methods:

- (1) The *DTW refinement* treats the candidate documents as sequences of multioverlay MWs and compares them by the DTW sequence alignment, which is suitable for order-sensitive matching.
- (2) The *Cosine refinement* treats the candidates as sets of multi-overlay MWs with IDF weighting and compares them by the Cosine distance, which reflects the needs of order-free matching.

While the DTW distance can be straightforwardly applied to multi-overlay MWs, for the Cosine measure we need to introduce some modifications. The standard Cosine score of a document D with respect to a query Q is computed as the (weighted and normalized) number of words that appear in both D and Q. However, the multioverlay MW representation works with the concept of matching, where two nonidentical MWs can be considered mutually relevant. Clearly, the MW-matching needs to be incorporated into the Cosine similarity evaluation. In an ideal case, the Cosine similarity of multi-overlay Q and D should take into account the number of words from Q that have some match in D, and the number of words from D having a match in Q (a single query MW can be matched by multiple different MWs in D, and vice-versa, a single MW in D may match multiple different MWs in Q). However, the evaluation of matches is computationally expensive, so we only consider the number of matched query MWs in the multi-overlay-MW Cosine similarity. To allow IDFweighted Cosine scores, we also need to determine the IDF of each multi-overlay MW w. This is computed as a logarithmically scaled inverse fraction of episodes containing any MW that matches w.

An important parameter of the two-phase search architecture is the size of the candidate set C, which determines how many episodes are subject to the more precise matching. Increasing the size of C gives a better chance of locating relevant episodes at the price of a linear increase of the refinement costs. We set the size of C as $\lceil k \cdot f \rceil$, where k is the required number of nearest neighbors to be returned as the query result, and $f \in \mathbb{R}^+$ is a *filtering factor*.

4.4.1. Experimental evaluation

We evaluate the performance of the two-phase retrieval model on the order-free and order-sensitive scenarios, using the challenging *mixed dataset*. We assess the



Fig. 6. Order-free searching on the *mixed dataset* using different combinations of hard and multi-overlay MWs and different sizes of candidate sets.

usefulness of the DTW and Cosine refinements and analyze the influence of the filtering factor f on the trade-off between the result quality and retrieval costs.

Figure 6 shows the results of selected approaches on the *order-free* scenario. As expected, solutions that employ the DTW alignment in any phase are not suitable for the order-free searching: the quality of a single-phase DTW search on hard MWs is very low, and the two-phase retrieval with DTW refinement provides worse results that the simple Cosine-based search on hard MWs. However, the two-phase retrieval with the Cosine refinement over multi-overlay MWs achieves better search quality than the simple hard-MW Cosine search: the average result precision of the k^*NN search improves from 77.54% to 87.53% for f = 3.

In Fig. 7, we depict the results of the same set of techniques on the *order-sensitive* scenario. We again observe the expected behavior — the DTW refinements that take into account sequence ordering significantly outperform the bag-of-words Co-sine refinements. Still, there are some interesting facts to be seen. The precision of bag-of-words solutions drops to half in comparison to order-free matching



Fig. 7. Order-sensitive searching on the *mixed dataset* using different combinations of hard and multioverlay MWs and different sizes of candidate sets. We only show the results for $k \leq 5$ and k^* , as the ordersensitive ground truth contains only half as many matching episodes for each query as the order-free ground truth.

experiments, because half of the episodes that are identified by matching patterns do not have them in the required order. Using the costly and non-scalable DTW alignment of the whole dataset provides the best result for k = 1, but with the growing result size the precision quickly falls down. This is caused by the border problem of hard MWs and better result quality could be achieved with DTW alignment of multi-overlay MW sequences, but the costs would be unacceptable. However, when we utilize the Cosine-based search on hard MWs for candidate filtering and then apply the DTW refinement on multi-overlay MW sequences, we obtain an efficient solution that achieves a good k^* precision — in particular, the precision of 75.74% is achieved with the DTW refinement and f = 3. The bag-ofwords candidate filtering thus works well even for the order-sensitive matching.

We also evaluate the influence of different candidate set sizes, which are determined by the filtering factor f. With the Cosine refinement, the setting f = 2.0improves the k^* precision of order-free retrieval from 77.54% to 86.45% at the cost of increasing the average query evaluation time from $65 \,\mathrm{ms}$ to $115 \,\mathrm{ms}$. The shift from f = 2.0 to f = 3.0 contributes to a slightly better precision (from 86.45% to 87.53%) but increases the average refinement time from $50 \,\mathrm{ms}$ to $79 \,\mathrm{ms}$, since the refinement costs grow linearly with the candidate set size. Further increasing the filtering factor has negligible effects on the result quality. A similar behavior can be observed for the DTW refinement and order-sensitive retrieval: there is a large improvement of search quality between no refinement and refinement with f = 2, for f = 3 the precision further grows (from 71.32% to 75.74%), but for f > 3 the search quality does not change significantly. The DTW refinement is slightly less efficient than the Cosine refinement, because the DTW computes the alignment of the complete MW sequences whereas Cosine only deals with the distinct MWs. For the same settings of order-free searching and f = 2, the Cosine refinement requires 50 ms on average and DTW refinement 88 ms.

4.4.2. Discussion

In the two-phase episode matching, a majority of dataset episodes are filtered out by the efficient and scalable Cosine distance on sets of hard MWs. Only the most promising candidates are submitted to the refinement phase that requires more expensive similarity evaluations. The refinement step significantly improves the result quality with an acceptable increase of processing costs. The precision-cost trade-off is determined by the candidate set size and can be adjusted depending on the preferences of a target application.

Noticeably, the two-phase order-sensitive matching with DTW refinement on multi-overlay MWs achieves better precision than the single-phase DTW alignment of multi-overlay MW sequences reported in Sec. 4.2. This supports our theory that the bag-of-words candidate filtering is well-suited for identification of matching patterns and filters out many irrelevant episodes that are confusing the DTW alignment. To complete our analysis, we have also evaluated a search of the whole dataset using multi-overlay MWs with the Cosine distance, which has been discarded earlier for high computation costs. The result precision is very similar to the twophase retrieval with Cosine refinement, which again confirms that the hard MWbased filtering phase works very well.

As a final remark, let us mention that the two-phase retrieval works for ordersensitive matching as long as the dataset is sufficiently diverse. In an extreme case, we could have a dataset of episodes that all contain the same patterns and differ only in the ordering, then the bag-of-words filtering would be useless. However, we do not expect such episodes to appear in real applications.

5. Efficient and Scalable Retrieval of Motion Episodes

The two-phase episode retrieval is able to provide relevant results and reduce the sequential-scan times by several orders of magnitude. However, its main advantage lies in the possibility of applying efficient and scalable indexing techniques for the candidate retrieval. In this section, we first describe how the hard-MW documents are indexed using the standard inverted files index. Then, we discuss the scalability of the candidate retrieval. This is to some extent solved by the text retrieval techniques, but the artificial MW vocabulary offers a new possibility of optimizing the vocabulary size for a given dataset. Finally, we demonstrate that the efficiency of both candidate retrieval and candidate set refinement can be further increased by reducing the query size.

5.1. Indexing motion episodes

The hard MW documents can be straightforwardly indexed by the standard inverted file structure [27]. In particular, for each MW from a given vocabulary, we create a posting list (PL) of all documents that contain this MW. Since our experiments show that considering text-frequencies is not helpful for episode retrieval, we only keep the document identifiers in the posting lists, which produces very compact PLs. To be able to use the IDF weighting, we further compute and store the IDF for each MW. Finally, we also store the magnitude of each document D in the bag-of-MWs representation, which is computed as the sum of IDF weights of all MWs in D. For our test database of 241 episodes, the posting lists together with the IDF and document magnitude tables fit into 60 kB of memory, while the original skeleton data occupy about 1 GB.

To answer a Cosine-based similarity query, the query episode is first transformed into a bag-of-words representation: $Q = \{q_1, \ldots, q_n\}$, where q_i are individual MWs from a given vocabulary. Then, we take the PLs for q_1, \ldots, q_n and gradually merge them, computing the score of each encountered document D_i as $Cosine_score(D_i) =$ $IDF(q_{j_1}) + IDF(q_{j_2}) + \cdots + IDF(q_{j_n})$, where q_{j_1}, \ldots, q_{j_n} are such elements of Q that have D_i in their posting list. Finally, the Cosine score is normalized by the sum of magnitudes of both Q and D_i .

Candid	late search	Refine	ement			${\rm Costs}~({\rm ms})$	
Data	Method	Data	Method	Precision $(\%)$	Phase I	Phase II	Overall
Skeletons	DTW			74.84	22,500		22,500
hMWs	DTW			58.61	630		630
moMWs	DTW		_	77.36	670		670
hMWs	Cosine			77.54	65		65
hMWs	Cosine	moMWs	DTW	81.06	65	88	153
hMWs	Cosine	moMWs	Cosine	86.45	65	50	115
hMWs	Cosine-index	moMWs	DTW	81.06	1.21	88	89
hMWs	Cosine-index	moMWs	Cosine	86.45	1.21	50	51

Table 2. Comparison of selected episode processing methods for k^*NN search over the *original dataset*. Cosine-based methods are used with IDF weights; the two-phase retrieval is used with filtering factor f = 2.

5.1.1. Experimental evaluation

To evaluate the effect of episode indexing and other techniques discussed in this section, we again use the *original dataset*. Let us remember that in this dataset, all episodes within one GT category have the same ordering of actions, so both order-free and order-sensitive methods are meaningfully applicable. We are now not much interested in the actual search precision of individual methods, which has been discussed in detail in previous sections; instead, Table 2 analyzes the development of query processing costs for different candidate retrieval methods.

We can observe that the introduction of indexed searching does not change the precision of retrieval results, because the Cosine score of each document is computed in the same way as with sequential scan. However, the retrieval efficiency is significantly improved, because the indexed search can skip documents that have zero overlap with the query. As reported in Table 2, the average candidate search time is reduced from 65 ms to 1.21 ms. We can also note that most of the query processing time is now spent on query refinement, which is evaluated over the multi-overlay MWs that require more costly processing. We discuss this issue in more detail in Sec. 5.3 and propose query reduction strategies that allow us to reduce the refinement phase costs.

5.2. Scalability of candidate retrieval

The retrieval of candidate episodes over the hard-MW inverted-file index is very fast, but our test dataset is small and does not allow us to demonstrate the scalability of our solution. To better understand the efficiency of the proposed retrieval scheme and its ability to scale to large episode collections, we now examine the candidate retrieval costs in more detail, identify potential scalability bottlenecks and discuss how these can be mitigated.

The costs of evaluating a query over the inverted file index are determined by two factors: (i) the number of posting lists to be accessed and merged and (ii) the length of individual PLs. The number of posting lists that need to be processed is equal to the number of distinct query words and is independent of the database size, so it does not pose a scalability issue. On the other hand, the length of individual posting lists is directly proportional to the database size and the vocabulary size. When the vocabulary size is fixed, the posting lists inevitably grow with the database size and their processing becomes more and more expensive, eventually exceeding the acceptable response time. This problem is well known in standard information retrieval, which solves it by applying approximate searching. More specifically, each posting list is kept sorted using some universal measure of documents' importance, and only a portion of the highest-ranking items from each list is accessed during retrieval [27]. A similar approach could also be applied to our posting lists, but the motion word technique offers an alternative way of dealing with long posting lists. Let us remember that the size of MW vocabulary is chosen by the system designer and should reflect the size and density of the dataset. Therefore, for static datasets, the vocabulary size can always be chosen so that the sizes of posting lists are manageable. For dynamically growing datasets, we could expand the vocabulary and rebuild the index whenever the posting lists exceed some size threshold or combine this strategy with the above-mentioned approximate retrieval.

5.3. Efficiency of candidate selection and refinement

As mentioned in the previous section, the episode searching times also depend on the size of the query. In indexed searching, the query size determines the number of PLs to be accessed, and in the refinement phase, it determines the costs of each candidate's similarity evaluation. The query size can have a significant influence on the overall search efficiency in case the query is large, which indeed happens in the case of episode processing. In contrast to classic information retrieval where users issue a few keywords that express their information need, in motion retrieval users describe their interest by an example object, i.e. the query episode. After the transformation into the MW representation, we obtain a long sequence of query MWs — for instance, a 40-s episode produces a sequence of 290 MWs. However, not all of the MWs are equally important for determining episode similarity, so a suitable query reduction can decrease the query processing costs without notable effect on search quality.

Let us first consider the Cosine similarity measure, which is used for both candidate retrieval and candidate refinement. The Cosine similarity works over the bagof-words representation, which only contains distinct query MWs, but its size is still considerable — in our dataset, we have on average 54 distinct hard MWs per episode and 206 distinct multi-overlay MWs (the multi-overlay vocabulary is significantly larger than the hard MW vocabulary). Such a large query is very likely to contain words with low IDF values that have little impact on the Cosine scores but increase the processing costs because they have matches in many documents. We denote such words as *motion stopwords* and optimize the Cosine-based similarity searching by removing some of the low-IDF stopwords from the query. We considered two possible ways of reducing the query size: (i) removing all MWs with IDF below a certain threshold or (ii) removing a fixed ratio of low-IDF MWs from each episode. The first option mirrors the stopword handling in text retrieval, but it assumes the existence of global motion stopwords, which may not be correct for motion retrieval. Indeed, there may be a motion episode composed of very frequent motions such as standing and sitting, which could be entirely eliminated if all low-IDF MWs were removed. Therefore, we choose the second approach, which looks for the most distinctive MWs in the context of a given episode.

The query size is also highly important for the DTW-based similarity evaluation, because the DTW processing costs are quadratic with respect to the episode length. In this case, we cannot apply the elimination of low-IDF MWs, because the DTW performs time-aware sequence alignment and does not consider MW weights. However, we can reduce the query in a different way. Let us remember that the MWs are created using overlapping segmentation of the original skeleton sequence. The overlaps are used to minimize the effect of segmentation time-shift and are important especially for the hard MWs which suffer from frequent mismatches of semantically related segments. However, for the multi-overlay quantization, the densely overlapping segments are not so vital, because the multi-overlay MW-matching function can identify related segments even if they are quantized to different partitions. Consequently, we reduce the query size for DTW by regular downsampling of the MW sequences.

5.3.1. Experimental evaluation

The effects of query reduction are again evaluated by experiments over the *original dataset*. We perform three separate set of experiments to analyze the influence of the query reduction on the Cosine-based candidate selection, Cosine refinement and the DTW refinement.

In Table 3, we can observe the effects of removing a fixed ratio of low-IDF hard MWs from each query in the candidate search phase. Eliminating up to 60% of the least important MWs does not significantly influence the retrieval precision, while the query processing costs gradually decrease. Removing up to 40% of least

Query size ratio (%)	Query size	Phase I costs (ms)	Precision			
			hMW+Cosine and moMW+DTW (%)	hMW+Cosine and moMW+Cosine (%)		
100	54	1.21	81.06	86.45		
80	43	0.98	81.26	86.57		
60	32	0.89	81.31	86.54		
50	27	0.86	81.53	86.45		
40	22	0.83	80.91	85.33		
20	11	0.77	77.97	82.30		

Table 3. Precision and costs of two-phase episode retrieval with reduced hard-MW queries in the candidate retrieval phase. Only a given percentage of query MWs with the highest IDF is used for querying. Experimental settings: k^*NN retrieval over the *original dataset*, f = 2.

210 P. Budikova et al.

Table 4. Precision and costs of two-phase episode retrieval with reduced queries in the Cosine refinement phase. Only a given percentage of episode MWs with the highest IDF is used for the refinement. No query reduction is used for the candidate retrieval. Experimental settings: k^*NN retrieval over the original dataset, f = 2.

Query size		Query redu	iction	Query and data reduction		
ratio (%)	Query size	Phase II costs (ms)	Precision (%)	Phase II costs (ms)	Precision (%)	
100	206	50.89	86.45	50.89	86.45	
80	165	40.57	86.42	38.23	86.44	
60	124	30.02	86.39	23.65	86.74	
50	103	25.42	86.29	17.51	86.84	
40	82	19.93	86.07	13.45	86.91	
20	41	9.98	83.91	5.16	85.00	

important MWs actually slightly increases the precision, which indicates that these MWs were not relevant for the identification of related episodes. It is also important to remark that the reduction is only applied to the query episodes, whereas the episodes in the database remain intact. This allows us to choose the query reduction parameters in time of querying, according to user preferences.

Table 4 summarizes the effects of bag-of-words reduction for the Cosine refinement over the multi-overlay MWs. There are two possibilities how to apply the reduction in the refinement phase — we can either remove the low-IDF MWs only from the query or from both the query and the candidate episodes that are compared to it. In contrast to the candidate retrieval where the query-time reduction of all searched episodes would not be feasible, the small set of candidates can be easily reduced as well. We can see that removing up to 60% of the low-IDF words from both the data and query does not harm the search precision and saves nearly 75% of the candidate refinement time. Reducing both the query and the candidate episodes produces better results than just the query-side reduction, which can be again attributed to the elimination of semantically irrelevant MWs. The processing time optimization is more substantial here than in the candidate retrieval phase, because the evaluation of similarity over the multi-overlay MWs is more costly.

The influence of downsampling query reduction for the DTW refinement is analyzed in Table 5. The DTW distance measure computes a time-aware alignment of the two multi-overlay MW sequences, so it is necessary to reduce both the query and

Table 5. Precision and costs of two-phase episode retrieval with reduced queries in the DTW refinement phase. No query reduction is used for the candidate retrieval. Experimental settings: k^*NN retrieval over the *original dataset*, f = 2.

Query and data downsampling	Query length	Phase II costs (ms)	Precision (%)
1×	290	88.05	81.06
$2 \times$	145	22.47	79.93
3 imes	97	10.20	79.17
$4 \times$	73	5.65	77.19
5×	58	4.73	76.08

the candidate episodes. We can observe that the search quality decreases here even for the lowest downsampling rate, but the worsening of precision is small while the time savings are substantial.

In all the studied cases, we could see that the reduction of query size to about 50% improves the efficiency of query processing without notably decreasing the retrieval quality. We have also tested the scenario with 50% reduced queries in both retrieval phases and obtained very good results — for the Cosine refinement the result precision is 86.82% (0.37% better than the retrieval with full queries) and for the DTW refinement 80.02% (1.04% worse than the retrieval with full queries). From the efficiency point of view, the query reduction is definitely useful for the refinement phase where a lot of the expensive similarity computations over multi-overlay MWs can be eliminated. The candidate selection over indexed hard MWs is already very fast, but for larger data collections the optimizations of candidate selection may also become important.

6. Conclusions

Episode retrieval is a new motion processing task that deals with similarity-based matching of skeleton recordings that capture complex human activities. In some cases, the episodes can be processed by state-of-the-art neural network models, but this approach requires costly training and is only suitable for some types of episode matching scenarios. Therefore, we propose to solve this task using a transformation of the input 3D skeleton data into compact motion-word sequences that can be processed by mature text-retrieval techniques. In particular, we first identify candidate sequences using the efficient and scalable Cosine distance on IDF-weighted sets of simple motion words. The candidates are then re-ranked using more complex motion words and a distance measure suitable for either order-free or order-sensitive matching. By using the compact data representation, inverted-file indexing, and query reduction techniques, we achieve the average processing time of about 20 ms per query, which is sufficient for real-time searching. Compared to the naive skeleton sequence alignment, our approach reduces the query processing time by three orders of magnitude. At the same time, we also increase the result precision because our bag-of-words candidate selection is better suited for identification of matching patterns than the sequence alignment. The whole approach does not require any training data and can be used in a wide range of applications. In the future, we plan to apply the proposed approach on large-scale collections of episodes from different sports domains, which will be obtained from video-data using state-of-the-art pose-estimation tools [1, 2].

Acknowledgments

This research has been supported by the Czech Science Foundation (project no. GA19-02033S).

References

- K. Sun, B. Xiao, D. Liu and J. Wang, Deep high-resolution representation learning for human pose estimation, in *Int. Conf. Computer Vision and Pattern Recognition*, 2019, pp. 5693–5703.
- [2] S. Kreiss, L. Bertoni and A. Alahi, PifPaf: Composite fields for human pose estimation, in IEEE Conf. Computer Vision and Pattern Recognition, 2019, pp. 11977–11986.
- [3] Q. Ke, M. Bennamoun, H. Rahmani, S. An, F. Sohel and F. Boussaid, Learning latent global network for skeleton-based action prediction, *IEEE Trans. Image Process* 29 (2020) 959–970.
- [4] S. Song, C. Lan, J. Xing, W. Zeng and J. Liu, Spatio-temporal attention-based LSTM networks for 3D action recognition and detection, *IEEE Trans. Image Process.* 27(7) (2018) 3459–3471.
- [5] C. Beecks, M. Hassani, B. Brenger, J. Hinnell, D. Schüller, I. Mittelberg and T. Seidl, Efficient query processing in 3D motion capture gesture databases, *Int. J. Semantic Comput.* 10(1) (2016) 5–25.
- [6] J. Sedmidubsky, P. Elias and P. Zezula, Searching for variable-speed motions in long sequences of motion capture data, *Inform. Syst.* 80 (2019) 148–158.
- [7] M. Ruta, F. Scioscia, M. Di Summa, S. Ieva, E. Di Sciascio and M. Sacco, Semantic matchmaking for Kinect-based posture and gesture recognition, *Int. J. Semant. Comput.* 8(4) (2014) 491–514.
- [8] P. Budikova, J. Sedmidubsky, J. Horvath and P. Zezula, Towards scalable retrieval of human motion episodes, in *IEEE Int. Symp. Multimedia*, 2020, pp. 49–56.
- [9] B. Krüger, A. Vögele, T. Willig, A. Yao, R. Klein and A. Weber, Efficient unsupervised temporal segmentation of motion data, *IEEE Trans. Multimed.* 19(4) (2017) 797–812.
- [10] X. Yu, W. Liu and W. Xing, Behavioral segmentation for human motion capture data based on graph cut method, J. Visual Lang. Comput. 43 (2017) 50–59.
- [11] B. Ren, M. Liu, R. Ding and H. Liu, A survey on 3D skeleton-based action recognition using learning method (2020).
- [12] T. Huynh-The, C.-H. Hua, N. A. Tu and D.-S. Kim, Learning geometric features with dual-stream CNN for 3D action recognition, in *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2020, pp. 2353–2357.
- [13] P. Wang, W. Li, C. Li and Y. Hou, Action recognition based on joint trajectory maps with convolutional neural networks, *Knowl.-Based Syst.* 158 (2018) 43–53.
- [14] H. Yang, Y. Gu, J. Zhu, K. Hu and X. Zhang, PGCN-TCA: Pseudo graph convolutional network with temporal and channel-wise attention for skeleton-based action recognition, *IEEE Access* 8 (2020) 10040–10047.
- [15] S. Yan, Y. Xiong and D. Lin, Spatial temporal graph convolutional networks for skeleton-based action recognition, in *Proc. 32nd AAAI Conf. Artificial Intelligence*, 2018, pp. 7444–7452.
- [16] H. Wang and L. Wang, Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks, in *IEEE Conf. Computer Vision and Pattern Recognition*, 2017, pp. 3633–3642.
- [17] J. Liu, G. Wang, L. Duan, P. Hu and A. C. Kot, Skeleton based human action recognition with global context-aware attention LSTM networks, *IEEE Trans. Image Pro*cess. 27(4) (2018) 1586–1599.
- [18] J. Sedmidubsky, P. Elias and P. Zezula, Effective and efficient similarity searching in motion capture data, *Multimed. Tools Appl.* 77(10) (2018) 12073–12094.
- [19] A. Aristidou, D. Cohen-Or, J. K. Hodgins, Y. Chrysanthou and A. Shamir, Deep motifs and motion signatures, ACM Trans. Graph. 37(6) (2018) 187:1–187:13.

- [20] J. Sedmidubsky, P. Budikova, V. Dohnal and P. Zezula, Motion words: A text-like representation of 3D skeleton sequences, in 42nd European Conf. Information Retrieval, 2020, pp. 527–541.
- [21] D. Dotti, E. Ghaleb and S. Asteriadis, Temporal triplet mining for personality recognition, in 15th IEEE Int. Conf. Automatic Face and Gesture Recognition, 2020, pp. 171–178.
- [22] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger and A. Weber, Documentation Mocap Database HDM05, Tech. Rep. CG-2007-2, Universität Bonn, 2007.
- [23] F. Carrara, P. Elias, J. Sedmidubsky and P. Zezula, Lstm-based real-time action detection and prediction in human motion streams, *Multimed. Tools Appl.* 78(19) (2019) 27309–27331.
- [24] M. Müller, Dynamic Time Warping, in Information Retrieval for Music and Motion (Springer, Berlin, Heidelberg, Berlin, Heidelberg, 2007), pp. 69–84.
- [25] J. Wang, Y. Chen, S. Hao, X. Peng and L. Hu, Deep learning for sensor-based activity recognition: A survey, *Pattern Recognit. Lett.* **119** (2019) 3–11.
- [26] J. Sedmidubsky and P. Zezula, Augmenting spatio-temporal human motion data for effective 3D action recognition, in 21st IEEE Int. Symp. Multimedia, 2019, pp. 204–207.
- [27] R. Baeza-Yates and B. A. Ribeiro-Neto, Modern Information Retrieval The Concepts and Technology Behind Search, 2nd edn. (Pearson Education Ltd., Harlow, England, 2011).