Multimed Tools Appl (2018) 77:12073–12094
DOI 10.1007/s11042-017-4859-7

Effective and efficient similarity searching in motion capture data

Jan Sedmidubsky[1] · **Petr Elias**[1] · **Pavel Zezula**[1]

Received: 5 October 2016 / Revised: 9 May 2017 / Accepted: 22 May 2017 /
Published online: 30 May 2017

Abstract Motion capture data describe human movements in the form of spatio-temporal trajectories of skeleton joints. Intelligent management of such complex data is a challenging task for computers which requires an effective concept of motion similarity. However, evaluating the pair-wise similarity is a difficult problem as a single action can be performed by various actors in different ways, speeds or starting positions. Recent methods usually model the motion similarity by comparing customized features using distance-based functions or specialized machine-learning classifiers. By combining both these approaches, we transform the problem of comparing motions of variable sizes into the problem of comparing fixed-size vectors. Specifically, each rather-short motion is encoded into a compact visual representation from which a highly descriptive 4,096-dimensional feature vector is extracted using a fine-tuned deep convolutional neural network. The advantage is that the fixed-size features are compared by the Euclidean distance which enables efficient motion indexing by any metric-based index structure. Another advantage of the proposed approach is its tolerance towards an imprecise action segmentation, the variance in movement speed, and a lower data quality. All these properties together bring new possibilities for effective and efficient large-scale retrieval.

Keywords Motion capture data retrieval · Effective similarity measure · Efficient indexing · k-NN query · Motion image · Convolutional neural network · Fixed-size motion feature

✉ Jan Sedmidubsky
xsedmid@fi.muni.cz

[1] Masaryk University, Brno, Czech Republic

1 Introduction

Computer-aided analysis of motion capture data (shortly *motion data*) remain interdisciplinary challenges that link together information retrieval, human computer interaction, robotics, artificial intelligence, security, entertainment, gaming industry, medicine, and sport. Motion data can be captured from multiple sources, for example, from video data [10, 25, 26], accelerometers in mobile devices [30], or from optical sensors [33, 43]. In this paper, we primarily focus on data obtained from optical-based sensors that estimate 3D positions of human body joints in a frame-by-frame fashion. Measuring similarity between such spatio-temporal data is a difficult task as a single action, e.g., kicking, can be performed by various actors in different styles. The actions can also vary in their lengths, speeds of performances, or initial body configurations. Given the fact that the similarity is also application dependent, there is no established global method for human motion comparison [45].

To effectively compare two motion sequences, both of them should have comparable lengths. For example, a 5-minute exercise sequence cannot be considered globally similar to a 2-second action of jumping. To compare meaningfully long motions, various segmentation techniques [21, 22, 42, 47] are applied to divide long motion sequences into shorter actions. Such segmentation techniques can generate a huge number of different-length motions that need to be efficiently compared against a query motion.

In this paper, we introduce a new effective and efficient similarity method for searching in motion data. This method extracts highly descriptive 4,096-dimensional feature vectors for rather short motions of various lengths. The fixed-size vectors can be efficiently compared by the Euclidean distance and indexed to speed-up the retrieval process by orders of magnitude, which makes the proposed method suitable for large-scale similarity search.

2 Related work

The majority of tasks such as action [12, 16] and activity [27–29] recognition, subsequence searching [41, 42] and stream annotation [34, 56] require motion data to be firstly preprocessed in order to extract descriptive features. These motion features are then either compared for similarity by distance functions or processed by machine-learning techniques, typically to learn a classification model. The following subsections describe various types of known motion features and the ways of their comparison, outline techniques for efficient similarity searching, and summarize the contributions of our approach.

2.1 Motion features

Human motions are modeled using a simplified skeleton figure represented by joints that are virtually connected by bones. The positions of joints are estimated for each video frame in the form of 3D coordinates. These coordinates are simply used as features [3, 48] as they keep the absolute body viewpoint in a captured space. To become invariant of the absolute positioning, original coordinates are, e.g., relativized to the skeleton centric space [1, 8, 50] or transformed to joint angle rotations [16, 39, 41], that furthermore benefit from a compact representation and invariance towards the size of the human skeleton.

Multiple features can be combined together to focus on more aspects of motions simultaneously, such as turning, accelerating or moving horizontally [3, 9, 34]. On the other hand, the amount of features can be narrowed to lower performance demands while keeping

reasonable effectiveness. In [8, 15], information of only five and three joints is considered while still acquiring high accuracy in action recognition.

To reduce the feature space and simplify indexing, features are quantized into discrete classes [32]. For instance, Müller et al. [34] extract 39 feature values of various kinds and carefully quantize each of them into {0, 1}. A similar idea is used by Ijjina et al. [15] who quantize distances between specific pairs of joints based on predefined thresholds. However, finding such thresholds is difficult and highly domain-dependent. Liang et al. [24] avoid determining thresholds by feature quantization into a histogram of 84 spherical bins.

The most relevant approach to our work is introduced by Milovanovic et al. [31] who represent motions by visual features to reveal similar walking patterns. However, they omit data normalization which results in a poorer representation power of visual features, and thus rendering their method less effective. In our preliminary work [13], we show that data normalization is of high importance and brings better results for general action recognition.

2.2 Comparing motions based on similarity

Motion features can be compared for similarity by (1) distance-based functions to obtain a list of the most similar motions with respect to a query motion or (2) processed by learning-based methods such as neural networks to obtain the classification of the query motion.

Distance-based methods utilize a distance function defining the measure of (dis)similarity between any pair of motions. Similarity can be directly compared on the level of multi-dimensional spatio-temporal trajectories [44, 51]. The most widely used function is the Dynamic Time Warping (DTW) [2, 41] and its variants [18] that quantify how good is a match between a pair of time series. The drawback of DTW is its quadratic complexity and inability to discover semantics in the inherent variability of motions, unlike approaches involving machine learning. Among other distance-based measures, the Bhattacharyya distance is used to compare pose-level histograms [2], the Martin distance is applied in Linear Dynamical Systems [7], or the Euclidean distance compares rotation angles [14]. Such methods support indexing and do not incorporate a time-consuming training phase. They are convenient for a wide range of applications, such as efficient large-scale query-by-example retrieval [41], motion classification [7, 54] using a k-NN classifier, or cluster analysis [24].

Machine learning methods generally employ a training phase to learn a classification model from provided training data. Effectiveness of the model is proportional to the quality and amount of the provided data. Convolutional neural networks currently constitute the state-of-the-art in machine learning, for example, Du et al. [12] achieve the best results by classifying motion data using a deep hierarchical recurrent neural network. Neural networks have usually a complex architecture and high-performance demands, but can also be very efficient when a simple architecture is chosen, e.g., a single-hidden layer feed-forward neural network annotating motion streams nearly in real time [8]. Besides neural networks, Support Vector Machines (SVM) are also widely used [6, 16, 46] for classifying motions.

The more detailed analysis of distance-based and machine learning methods applied to motion data can be found in recent papers [38, 45].

2.3 Content-based searching

Searching is a very important operation for motion capture data. It requires a convenient feature representation and effective method for the pair-wise similarity comparison [52]. A

simple way of searching constitutes the evaluation of k-nearest neighbor queries using a sequential scan, e.g., to recognize the class of query action [2, 34].

Searching becomes challenging when dealing with large volumes of motion data. For instance, the task of subsequence matching [17, 49] requires long motion sequences to be partitioned into a large number of short motion parts that need to be efficiently compared with a short query motion. In general, partitioning techniques [5, 42, 47] can generate millions of motion instances making the sequential scan inapplicable. To significantly speed-up similarity searching in large databases, scalable metric-based index structures can be easily utilized due to their ability of being extensible [55]. In the field of motion data, the trie-based structure is used to efficiently access motion features in [17], M-Index is used to perform fast subsequence retrieval on key poses [41], or KD-Trees are utilized in [20]. Traditional memory index structures are hardly usable when motion data do not fit into main memory. In this paper, we also target the issue of scalability and employ a very efficient disk-oriented approach to approximately search a 20-million motion database in real time.

2.4 Our contribution

In this paper, we introduce a novel solution for searching in large volumes of motion data. The core is formed by an effective motion similarity measure that combines advantages of both distance-based and machine learning methods. The specific contributions of this paper are the following:

- *Image-based motion representation* – several variants of motion normalizations, such as skeleton positions, orientations and sizes, are proposed to transform motion data into images;
- *Motion similarity measure* – effective 4,096-dimensional feature vectors are extracted from motion images using a convolutional neural network and compared by the Euclidean distance;
- *Indexing and large-scale search* – an applied index structure enables real-time searching in a database containing 20 millions of short motions.

The proposed solution has a potential to be employed in a wide range of motion retrieval applications due to several positive properties such as indexability or tolerance towards different speed of execution or imprecise segmentation (i.e., to an added noise or partly occluded content). Furthermore, a thorough experimental evaluation in the terms of effectiveness and efficiency presents challenging results, compared to state-of-the-art methods.

3 Visualization-based similarity

We propose a new concept of encoding 3-dimensional joint trajectories into images. Such visualization is conveniently combined with computer vision methods for content-based image retrieval. Namely, a convolutional neural network [19] is used to detect and recognize key visual patterns in images. In particular, the last hidden layer of the network is used to extract a 4,096-dimensional feature vector from each image. These fixed-size features *effectively* represent original motions of variable lengths and can be *efficiently* compared for similarity by metric functions, such as the Euclidean distance. In addition, metric functions can be indexed using metric-based structures, such as PPP-Codes [35], to retrieve query-similar feature vectors very efficiently. The proposed *visualize-train-extract-index-retrieve*

concept is depicted in Fig. 1 and introduces a generalized view on comparing similarity in motion data, with a possible applicability in large-scale searching, filtering, classification or clustering. In the following subsections, we formally introduce motion data and describe the processes of motion visualization, feature extraction and indexing.

3.1 Motion data definition

Motion data are represented as trajectories of the specific body-joint positions in a 3D space. Every *motion* m (e.g., a simple gesture, action, or complex activity) is an ordered sequence $(p_1, \ldots, p_n)$ of *poses* p_i ($i \in [1, n]$), where $n \in \mathbb{N}$ denotes the motion length in terms of the number of frames. Each pose p_i represents the skeleton configuration in a given frame by an ordered sequence $(j_1, \ldots, j_l)$ of joint coordinates $j_i = (j_i[x], j_i[y], j_i[z])$, where $l \in \mathbb{N}$ represents the number of tracked joints. The motion data used in experiments recognize $l = 31$ different joints on the human body. Individual joints are visualized by a simplified skeleton and ordered within the kinematic tree in Fig. 2. For clarity, we denote pelvis, left hip joint and right hip joint by abbreviations j_{root}, j_{lhip} and j_{rhip}, respectively.

3.2 Motion data normalization

Normalization neutralizes differences in motions which are performed similarly but in different contexts, such as a different location, facing direction or by humans of different bodies [37]. Normalization is applied on the level of individual poses by changing their original coordinates throughout the whole motion m to acquire a *normalized* motion $\overline{m}$. We altogether provide $2 \cdot 2 \cdot 2$ variants of position, orientation and skeleton-size normalizations that can be combined together. Whether to apply the specific normalization or not always depends on requirements of a particular application.

3.2.1 Position normalization

Intuitively, position normalization aligns variously positioned motions from the real-world space into a specific location in some virtual space. This helps focus on the way *how* motions are performed rather than *where* they are performed. We can either force all the motions to start at the same initial position, or we can fix all their poses to that position:

– *First-pose position normalization* – the same starting position of root is assured for every motion by shifting the skeleton configuration of (1) the first pose so that its

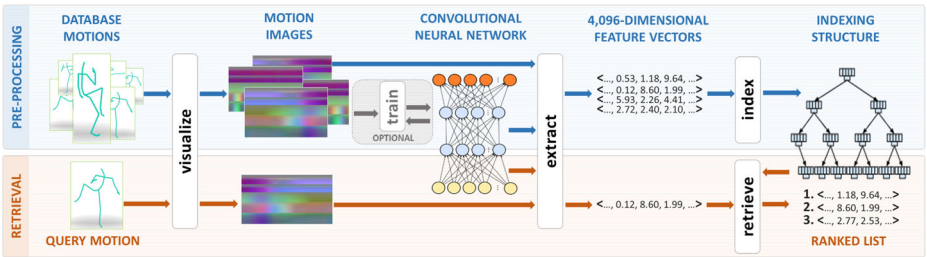

Fig. 1 Flowchart diagram that demonstrates encoding motion data into images, extracting their 4,096D feature vectors using a convolutional neural network, indexing the extracted features, and retrieving a ranked list of the most similar motions with respect to a query

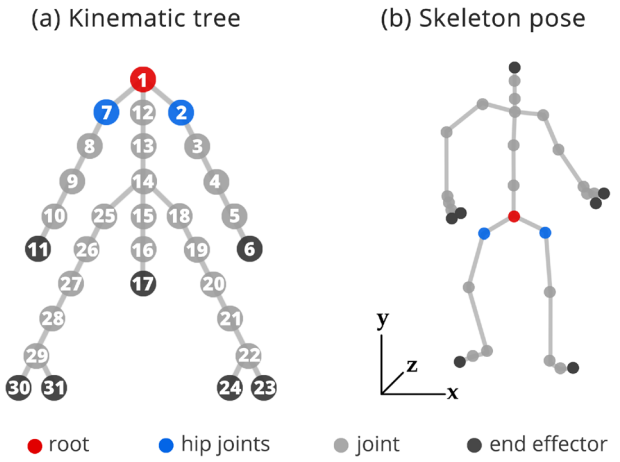

Fig. 2 The kinematic tree and corresponding skeleton model

root joint gets the position $(0, 0, 0)$ and (2) other poses relatively to the first one. This normalization can be useful, for example, when we need to compare long-jump performances.

– *All-poses position normalization* – actors are attached by their root to the origin $(0, 0, 0)$ throughout the whole motion in every single pose. The newly obtained space, referred to as the skeleton-centric position-invariant coordinate system, has an advantage of reducing the tracked-space size.

Publicly available motion capture datasets do not usually contain movement categories that can be directly distinguished by the absolute traveled distance (e.g., short jump and long jump). For this reason, it is practical to apply all-poses position normalization to reduce the size of the tracked space without losing much of original information.

3.2.2 Orientation normalization

Orientation normalization unifies the direction which the skeleton is facing. Such normalization assumes the y-axis is pointing upwards. All joints within a pose are rotated around the y-axis so that the subject faces the positive x-axis and the hips are placed parallel to the z-axis. The angle of rotation φ of a given pose is defined as:

$$\varphi = arctan\left(\frac{j_{lhip}[x] - j_{rhip}[x]}{j_{lhip}[z] - j_{rhip}[z]}\right).$$

To rotate the whole skeleton, each joint coordinate j_i is transformed in a way that the y-position remains unchanged and x- and z-positions are rotated by angle φ:

$$(\overline{j_i[x]}, \overline{j_i[y]}, \overline{j_i[z]}) = \begin{pmatrix} j_i[x] \\ j_i[y] \\ j_i[z] \end{pmatrix}^T \begin{pmatrix} cos(\varphi) & 0 & sin(\varphi) \\ 0 & 1 & 0 \\ -sin(\varphi) & 0 & cos(\varphi) \end{pmatrix}.$$

Similarly as in position normalization, we distinguish two different variants:

– *First-pose orientation normalization* – the skeleton is rotated in each pose according to fixed rotation angle φ computed in first motion pose p_1. The actor initiates the motion

facing the direction of the positive x-axis and, afterwards, the body orientation changes as in the original motion. This is useful, for example, when we want to distinguish turning left and right.

– *All-poses orientation normalization* – the skeleton is rotated in each pose individually according to its rotation angle. Since the skeleton is facing the same direction all the time, the turning aspect is not considered anymore. This can be convenient, for example, when considering jogging and jogging in circle as the same movement category.

An example of the visualization of position and orientation normalizations is illustrated in Fig. 3.

3.2.3 Skeleton size normalization

As people vary in sizes, joint trajectories for the very same movement performed by various actors can be significantly different. To be able to focus only on nuances in movement execution, it is vital to work with normalized skeletons. Each skeleton bone can be scaled to an average limb size over a given population.

– *Normalized skeleton* – starting from the root joint as parent joint j_{parent}, each coordinate of its child joint j_{child} (i.e., joint connected with the root by a bone) is adjusted according to the average bone length $b_{parent,child}$ connecting these joints as:

$$\overline{j_{child}} = (j_{child} + \alpha \cdot (j_{child} - j_{parent})),$$

where $\alpha = b_{parent,child} / \| j_{child} - j_{parent} \|$ is the ratio between the average and the actual bone length. Based on the kinematic tree (see Fig. 2), coordinates of other joints are recursively adjusted.

– *Original skeleton* – skeletons keep the original proportions of their limbs, which is useful when motion data are categorical with respect to a body or limb size.

3.3 Motion data visualization

We effectively represent each normalized motion as a static image, where pixel *colors* represent quantized positions of normalized joint coordinates. Seeing this image as a matrix of pixels, a single column represents the skeleton configuration at a given frame while a row illustrates how the position of a given joint changes in time. The color of each pixel is encoded using the three-compound RGB color space with 8 bits (i.e., 256 values) per channel to conveniently approximate (quantize) the nearly continuous space into 256^3 bins (i.e., 256 bins for each of x, y and z axis). The image is constructed by normalizing the motion,

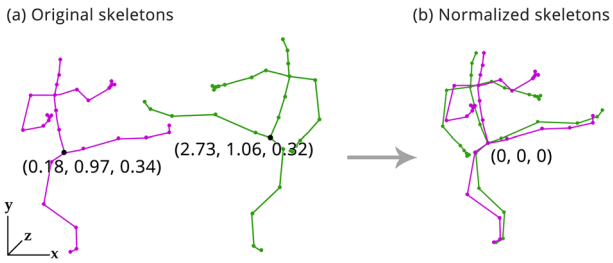

Fig. 3 Position and orientation normalizations of two similar skeleton configurations

quantizing each normalized joint coordinate into a given bin, transforming each pose into a vertical color *stripe* (i.e., the column of the image matrix) and concatenating all the stripes in order of poses. The whole transformation process of motion $m = (p_1, \ldots, p_n)$ is illustrated in Fig. 4 and consists of the following four steps.

1. *Motion normalization* – poses of motion m are optionally normalized in their position, orientation and size to obtain normalized motion $\overline{m} = (\overline{p_1}, \ldots, \overline{p_n})$ with normalized coordinates $(\overline{j[x]}, \overline{j[y]}, \overline{j[z]})$ of each joint j.
2. *Quantization of coordinates* – normalized coordinates in each pose are then quantized into the space $[0, 255]^3$ of 16 M bins. New quantized coordinate $(\widehat{j[x]}, \widehat{j[y]}, \widehat{j[z]})$ of joint j is calculated as:

$$(\widehat{j[x]}, \widehat{j[y]}, \widehat{j[z]}) = \left\lfloor \frac{255}{\overline{c_{max}} - \overline{c_{min}}} \cdot \begin{pmatrix} \overline{j[x]} - \overline{c_{min}} \\ \overline{j[y]} - \overline{c_{min}} \\ \overline{j[z]} - \overline{c_{min}} \end{pmatrix}^T \right\rfloor,$$

$$\overline{c_{min}} = min\left\{min\{\overline{j[x]}, \overline{j[y]}, \overline{j[z]}\} | \, \forall \overline{j} \in \overline{p}, \forall \overline{p} \in \overline{m}\right\},$$

   where $\overline{c_{min}}$ and $\overline{c_{max}}$ are global minimum and maximum values of normalized joint coordinates, respectively.
3. *Single pose visualization* – pose $\widehat{p} = \left(\widehat{j_1}, \ldots, \widehat{j_{31}}\right)$ with quantized coordinates is visualized as a vertical stripe image of 31 pixels, where each pixel $i \in [1, 31]$ is assigned RGB color $R = \widehat{j_i[x]}$, $G = \widehat{j_i[y]}$ and $B = \widehat{j_i[z]}$).
4. *Motion visualization* – stripe images of individual poses $(\widehat{p_1}, \ldots, \widehat{p_n})$ are concatenated in the same order to construct the motion image of $31 \times n$ pixels.

The proposed image representation compactly keeps most of the characteristics of the original motion. Since images encode motions in a quite accurate fashion, it is possible to reconstruct original motions with a small relative joint coordinates error. Assuming the all-poses-position normalization and average person height of 180 cm, relative joint coordinates can be reconstructed with the error up to 7 or 0.03 millimeters by using 8- or 16-bit color representation. The 8-bit representation is sufficient since its reconstruction error is comparable to the tracking error of nowadays motion capturing technologies.

Images of motions from different categories look diversely – this is especially practical for the future detection of visual patterns by a convolutional neural network. Examples of several images generated for distinct motion categories are visualized in Fig. 5. Motion images might be eventually helpful for humans to discover distinctive motion characteristics at the first sight (e.g., repetition of sub-motions or rapid movement changes). Moreover, the time component can be easily adjusted by resizing the images. However, the main advantages are that (1) such images can be effectively processed by neural networks that can learn

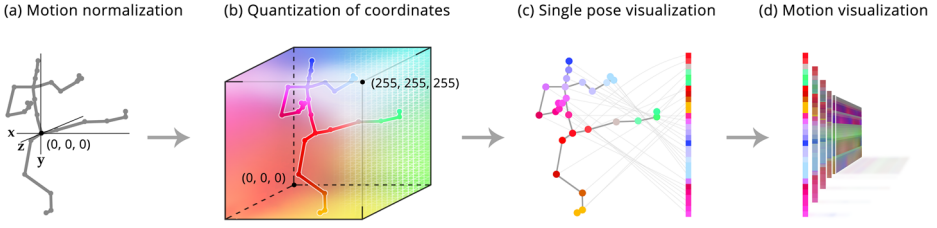

Fig. 4 Transformation of a single pose into a vertical stripe image (**a**–**c**) and concatenation of stripe images of all poses into a motion image (**d**)

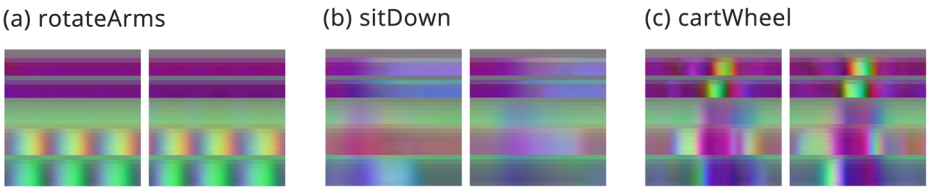

Fig. 5 Six motion images belonging to three categories. Repeating movement patterns can be also discovered by humans at the first sight, like in the first two images

and reveal visual patterns, (2) fixed-size features extracted by the neural network can be compared for similarity.

3.4 Extraction and comparison of fixed-size feature vectors

We compute similarity of motion images by extracting their 4,096-dimensional feature vectors that are compared by the Euclidean distance. These fixed-size vectors are extracted using a *reference model* of convolutional neural network.[1] This model is a replication of well-known Krizhevsky's [19] model that consists of 5 convolutional and 3 fully connected layers. It is trained on 1 M ImageNet photographs and classifies an input image of 256×256 pixels into one of 1 k categories.

We utilize the reference model, which performs very well on the domain of photographs, and *fine-tune* it to also recognize motion images with a high accuracy. In particular, we resize each motion image to the input size of 256×256 pixels and change the network layer of 1 K output neurons to 122 motion categories used in experiments. The advantage of pre-learnt filters (i.e., the weights initialized in lower levels of the network) trained on 1 M photographs is not lost as the learning phase affects more significantly the uppermost layers. Such tuning is done by feeding the network with application-specific motion images and their corresponding category labels.

3.4.1 Feature extraction

The response of the last hidden layer of the network is known as the DeCAF feature [11]. This feature represents a 4,096-dimensional vector of real numbers which carry great semantic information – the feature vectors compared by the Euclidean distance form the feature space that clusters similar images together. This is even true for feature vectors belonging to categories on which the network has never been explicitly trained. The specific results are reported within the experimental evaluation. The main advantage of feature vectors is their possibility to be compared by the Euclidean distance for similarity, which enables their indexing and utilization in a wide variety of applications.

To extract a 4,096-dimensional feature vector for each motion image, we utilize the GPU implementation, which further decreases extraction time by order of magnitude, compared to the CPU implementation. The extraction time using a single GPU card takes 25 ms on average.

Compared to normalization and visualization of motion data into images, the feature extraction process is the main bottleneck from the time complexity point of view. We can

[1] https://github.com/BVLC/caffe/tree/master/models/bvlc_reference_caffenet.

extract about 40 feature vectors from motion images per second using a single GPU card. Considering the data sampling frequency of 120 Hz and a sliding window of average length of 2 seconds (i.e., 240 frames), we are able to gradually shift the sliding window by 3 frames and still generate the feature vectors in real time. This demonstrates the possibility of our approach to be also usable in some real-time motion tracking applications.

3.5 Large-scale retrieval by metric-based indexing

We evaluate a k-nearest-neighbor (k-NN) query to search for the most similar motions. Having a query motion, we extract its 4,096-dimensional feature vector, compute the Euclidean distance between the query and all vectors stored in a database, and present the k most similar vectors as the query result. Using a single CPU (i7 960 at 3.2 GHz) we can approximately compute 250,000 Euclidean distances per second. By applying a sequential scan we can search in real time a database of only 250 K motions. To search in much larger databases in less than one second, we need to organize the database features within an appropriate index structure.

As the Euclidean distance is a metric function, we can use any metric-based search technique to index the database features – see [55] for a survey. We further focus on disk-oriented structures that do not require to keep the features in main memory. For example, the database of 20 M feature vectors occupies about 328 GB (approximately 16 KB per vector) and thus do not fit into main memory. On the other hand, if the features are stored on disk, the primary bottleneck is their reading from disk into main memory for each query.

To reduce the number of disk read operations by orders of magnitude, we decide to utilize a metric-based approximate search structure, called the PPP-Codes [35]. This index structure defines a mapping of the feature vectors onto small codes composed of pivot permutation prefixes from several pivot spaces. These codes are kept in memory; given a k-NN query, the PPP-Codes algorithm combines candidate sets from independent pivot spaces into a small but very accurate candidate set. Only vectors from this candidate set are read from the disk and refined. The technical details about this structure and search process can be found in [35].

4 Experimental evaluation of the similarity method

We thoroughly analyze the proposed similarity method on a search scenario by evaluating k-nearest neighbor queries from both effectiveness and efficiency points of view. We analyze the impact of different motion normalizations and variously fine-tuned neural network models on search effectiveness. We also analyze properties of the space generated by the proposed method and evaluate its sensitivity to noisy data. Then we analyze the search efficiency and compare the results against recent approaches.

4.1 Dataset

The search scenario is evaluated on the publicly available HDM05 [33] motion capture dataset. This dataset contains 324 motion sequences performed by 5 distinct actors. The dataset authors also provide the ground truth, which categorizes 2,345 manually segmented motions (i.e., semantically meaningful parts of motion sequences) into 130 categories of specific movement actions, such as the turn left, sit down on a chair, or clap with hands five times. The average action length is 2.17 seconds – it corresponds to about 260 frames with

the dataset sampling frequency of 120 Hz. We select the HDM05 dataset because it contains the largest number of 130 categories when compared to the others, such as CMU[2] (30 categories), NTU RGB+D [43] (60 categories), MSR [23] (20 categories) or MHAD [36] (11 categories). Moreover, it is characterized with a subtle categorization, containing for example separate classes for kicking with left or right leg, to the front or to the side. For these reasons, the HDM05 dataset is very challenging for evaluating the search scenario.

To evaluate k-nearest neighbor queries for higher values of k, we ignore 8 categories with less than 10 motion instances. Thus our resulting ground truth contains **122** categories with 2,328 motions in total. We denote such ground truth as **HDM05-122**.

4.2 Methodology

We evaluate the search accuracy by analyzing results of k-nearest neighbor (k-NN) queries. Firstly, the HDM05-122 ground truth has to be preprocessed in order to:

1. Generate motion images according to the specific normalization method;
2. Divide the generated motion images into *training* and *test* sets;
3. Fine-tune the neural network model by the training motion images;
4. Extract a 4,096-dimensional feature vector for both sets of training and test images based on the fine-tuned network model.

The test feature vectors are then used as query object arguments of k-NN queries that are evaluated with five settings of $k \in \{1, 3, 5, 10, cat_{max}\}$, where cat_{max} is a special case determined for each query independently as the maximum number of relevant objects to be possibly retrieved, i.e., cat_{max} value corresponds to the number of training objects that belong to the same category as the query object.

Each k-NN query is then evaluated by searching for the most similar training vectors. The *precision* of the query answer is traditionally calculated as a ratio between the number of retrieved vectors of the same category as the query object and the number of all retrieved vectors. Note that if $k = cat_{max}$, the precision corresponds to the same value as the recall, i.e., to the ratio of all relevant objects retrieved. The global search accuracy is then measured as an average precision over all k-NN queries.

To analyze the influence of fine-tuning, we consider four different portions of training data: 0 %, 50 %, 90 % and 100 %. In case of 50/90 %, the 2/10-*fold cross validation* procedure is adopted to fine-tune 2/10 instances of neural network models, respectively. On the other hand, special cases of 0 % and 100 % are evaluated on the basis of *leave-one-out* procedure. The case of 100 % of training data is used to fine-tune the network model on all HDM05-122 motion images, while the 0 % case uses the *not-tuned* network model, i.e., the reference model trained on ImageNet images (see Section 3.4). In all four cases, all the HDM05-122 motions are gradually used as test queries.

4.3 Effectiveness of normalization

A suitable selection of motion data normalization influences the search accuracy. Figure 6 presents precision values of 8 variants of normalizations by combining different settings of the position, orientation and skeleton normalization. The results are evaluated using k-NN queries on both 100 %-fine-tuned and not-tuned features (features extracted using

[2] http://mocap.cs.cmu.edu/.

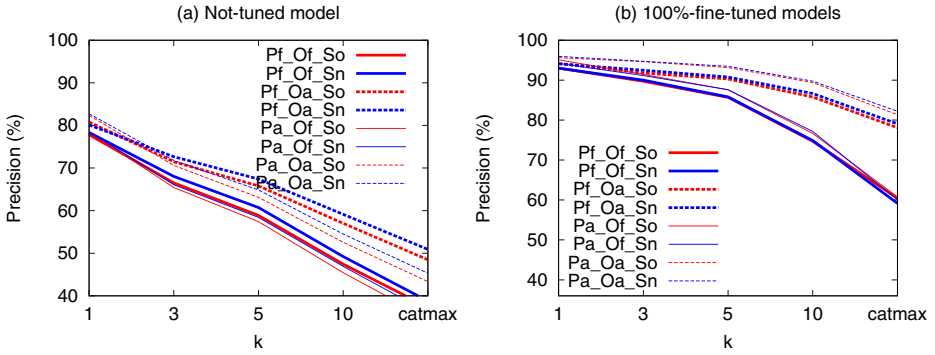

Fig. 6 Impact of different combinations of normalizations to the precision evaluated using k-NN queries on (**a**) not-tuned and (**b**) fine-tuned features. The first/all (Pf/Pa) pose position normalization is denoted by *thick/thin line*, first/all (Of/Oa) pose orientation normalization by *solid/dashed line*, and original/normalized (So/Sn) skeleton by *red/blue color*

the reference model). In particular, the original/normalized (So/Sn) skeleton is denoted by red/blue color, first/all (Of/Oa) pose orientation normalization by solid/dashed line, and first/all (Pf/Pa) pose position normalization by thick/thin line. The precision values are expected to have a decreasing trend with an increasing value of k. Focusing on the not-tuned model in Fig. 6a, we can observe that:

- The blue lines are always above the red lines of the same thickness and type (i.e., the same position and orientation normalization) which emphasizes the importance of *skeleton normalization* for the HDM05 dataset;
- All the dashed lines have a higher precision than the solid ones which results in recommendation to apply the *all-pose orientation* normalization all the time;
- The *first-pose position* normalization has slightly better results than centering all the poses, when fixing the skeleton and orientation normalization.

With a higher value of k, the normalization plays a more and more important role. For example, by fixing k to 10, the difference between the best (Pf_Oa_Sn) and worst (Pa_Of_So) combination of normalization is almost 14 %. A similar behavior can be also observed for the fine-tuned network models in Fig. 6b. The skeleton normalization is negligibly better than original skeleton proportions, while the orientation normalization in all poses helps a lot. The only difference is that centering all the poses achieves slightly better results than the first-pose position normalization. It is caused by the fact that all the centered poses utilizes a much smaller space in total and thus the color spectrum within the RGB cube is more utilized. A better color spectrum utilization has to be learned by the neural network, otherwise it is not so useful, as confirmed by experiments on the not-tuned model.

To sum up, the experiments show that absolute viewpoints and positions are rather superfluous and not decisive for general action recognition on the HDM05 dataset. It is the reason we decide to fix the combination of all-pose position, all-pose orientation, and normalized-skeleton normalization (i.e., Pa_Oa_Sn) for the rest of experiments.

4.4 Effectiveness of fine-tuning a neural network

Figure 7 presents the search accuracy of feature vectors that are extracted using network models on four different portions of training data. In case of 50 % and 90 % fine-tuned

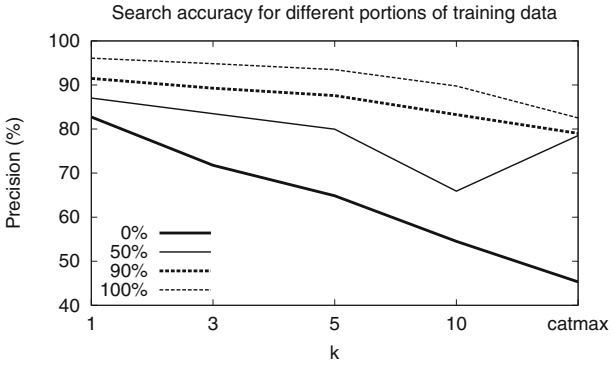

Fig. 7 Impact of different portions of training data to the search accuracy

models, the precision values are additionally averaged over 2 and 10 measurements (folds), respectively. As expected, the search accuracy increases as the neural network is fine-tuned on larger portions of data. On the other hand, differences between precisions of 50 %, 90 % and 100 % fine-tuned models are relatively small. For example, the difference between 50 % and 100 % fine-tuned models is only 9 % for $k = 1$. Note that a steeper precision decrease of the 50 % approach for $k = 10$ is caused by a limited number of relevant motions in the training set, which equals to 8.5 on average.

The great advantage of the proposed concept is its ability to extract highly-descriptive feature vectors even if the neural network is not fine-tuned at all, i.e., the reference model trained on a completely different domain of 1,000 image categories is used. Although no motion data are employed for training, the search accuracy surprisingly achieves high values, e.g., 82.7 % for $k = 1$. In such scenarios where training data are not known, machine-learning approaches such as [12] cannot be applied at all.

We also show that a fine-tuned network model is robust and can be used to extract descriptive feature vectors for different kinds of motion actions. In particular, we extract feature vectors using the existing 100 %-fine-tuned model but for 1,464 motions belonging to other 15 categories (this categorization is specified in [34]). The search accuracy achieves very high values of 93.9 %, 92.1 %, 90.7 %, 85.8 % and 60.2 % for $k = 1, 3, 5, 10$ and cat_{max}, respectively. These results demonstrate that a single network model can be utilized for various action categories unknown during the training phase.

4.5 Analysis of space generated by the similarity measure

We analyze the properties of a space generated by the feature vectors that are compared by the Euclidean distance. In particular, we compute the average intra and inter category distance for each HDM05-122 feature vector extracted using the 100 %-fine-tuned network model. The intra and inter category distances are then averaged over all objects in the same category and illustrated graphically in Fig. 8. The lower the intra-category and higher inter-category distance, the better clustering and also search accuracy. For example, the "cartwheelLHandStart1Reps" category is well recognizable because it has a very high inter-category distance even if having a quite high intra-category distance.

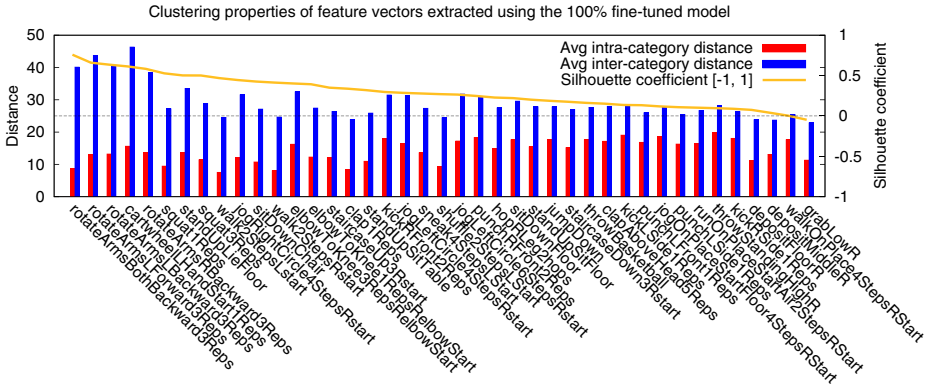

Fig. 8 The difference between average intra and inter cluster distances (*left y-axis*). The categories are sorted according to the silhouette coefficient (*right y-axis*). For better clarity, only a subset of 41 categories out of 122 is illustrated

To determine how well the feature vectors are clustered, we calculate the *silhouette coefficient* [40]. The silhouette coefficient refers to a method of interpretation and validation of consistency within clusters of data. The coefficient $\in [-1, 1]$ describes how similar a feature vector is to its own cluster (category) compared to other clusters. A value near 1 indicates that the feature vector is well matched to its own cluster and poorly matched to neighboring ones. A value near zero means that the vector is on the border of two clusters, while a value near -1 suggests moving the vector to another cluster. The silhouette coefficient is computed for each category by averaging coefficients of the vectors belonging to the given category and visualized by a yellow color in Fig. 8. The most important observation is that only 4 categories ("walkOnPlace2StepsRStart", "hitRHandHead", "grabLowR" and "grabHighR") out of 122 have the silhouette coefficient within interval $[-0.1, 0]$, i.e., lower than zero. This means that the other categories are quite well clustered with an average silhouette coefficient of 0.293, computed over all 122 categories.

4.6 Sensitivity to noisy data

In the following three subsections, we analyze how the proposed similarity method is tolerant towards noisy data by means of a decreased data quality, changed movement speed and slightly cropped motions.

4.6.1 Quality of motion data

We show the elasticity with respect to the input quality of motion data in the terms of different frame-per-second (fps) rates. We simulate a decreasing data quality by reducing the original 120-fps rate to the 60-, 40-, 20-, 12-, 6-, 3- and 2-fps rate. The inferior-quality motion of the i-fps rate is obtained by considering only each $(120/i)$-th frame of the original motion. We transform all 2,328 HDM05-122 motions to analyze each observed frame rate separately. We use both 0 % and 100 %-fine-tuned models to extract feature vectors from low fps rate motions which are evaluated by 1-NN queries. The results presented in Fig. 9a show how much a lower fps rate influences the search accuracy. These results imply that the proposed method is able to search motion data with almost the same

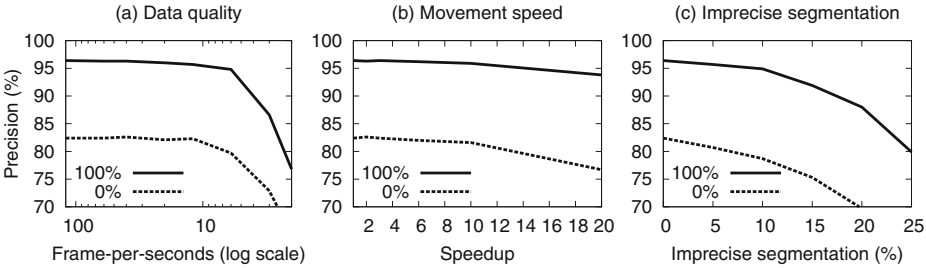

Fig. 9 Analysis of sensitivity towards a (**a**) decreasing data quality, (**b**) quicker movements and (**c**) imprecise segmentation with respect to the 1-NN precision (evaluated using the 0 % and 100 %-fine-tuned network models)

precision even if the original 120-fps rate is decreased 10 times. Both 0 % and 100 % scenarios follow the same trend line, which can be also observed in the following two experiments.

4.6.2 Different movement speed

To analyze the elasticity towards a different speed of performed actions, we simulate faster movements. The faster movements are created by decreasing the original 120-fps rate similarly as in the previous experiment but the accelerated motion is then evaluated against the original motions. For example, having an original 2-second action of kicking, we only consider every 2nd frame to simulate its two-times faster performance variant. For 2-, 3-, 6-, 10- and 20-faster motions, we evaluate the impact of a different speed to the precision in Fig. 9b. The results show about the same precision values for motions which are up to 10-times faster (i.e., 1-, 2-, 3-, 6- and 10-times faster). The precision only slightly drops for 20-times faster motions. This is practical since performing the same type of action as much as twenty times faster is quite unrealistic scenario. These results imply that the proposed method is very tolerant to a faster/slower movement performance.

4.6.3 Imprecise segmentation

The HDM05 dataset is already cut into parts that flawlessly correspond to well-segmented actions (see Section 4.1). However, when automated segmentation methods are used, the cuts can carry a certain degree of noise causing extra or missing parts in actions. In this scenario, we show how segmentation mistakes influence the search accuracy. We simulate an imprecise segmentation in 5 different scenarios by trimming each HDM05-122 motion by 5, 10, 15, 20 and 25 % of the original frames. For example, for the 10-% scenario and motion of 260 frames, a random part of $x \in [0, 26]$ frames is cut from the left side of the motion and the rest of $26 - x$ frames is cut from the right side. The segmentation error is distributed randomly on each of the sides of the motion but bounded in its total size. As in the previous experiment, the trimmed motions are used as query objects of 1-NN queries that are evaluated against the original HDM05-122 motions, excluding the exact match. The results presented in Fig. 9c demonstrate that our method is still effective in searching motions that lose as much as 10 % of their content due to the bad segmentation. The error of 20 % decreases the precision by 10–12 % which is not so big gap against the original accuracies. These results are particularly interesting as more than 50 action categories interfere

in cyclic movements with another category (e.g., "walkLeft2Steps" and "walkLeft3Steps" categories).

4.7 Indexing and scalability

We also evaluate efficiency of the proposed similarity method on a large-scale search scenario. To obtain much more than 2,345 motions, as specified by the HDM05-122 ground truth, we partition original long motion sequences into short motions. In particular, we generate 20 million motions of variable sizes by applying a segmentation technique [42] with various settings on all the original 324 HDM05 motion sequences. The generated motions are then preprocessed to extract 4,096-dimensional feature vectors, which takes about 6 days using a single graphics card. We store the extracted features on an SSD disk and index them by the PPP-Codes [35] (see Section 3.5) that is initialized by 256 randomly selected feature vectors as pivots.

In the retrieval phase, we measure an average search time to evaluate a single query on the 20 M database. We also control the level of PPP-Codes search approximation (i.e., tradeoff between search effectiveness and efficiency) by setting the maximum number of feature vectors that are accessed during the search process. Since the ground truth is not known for the generated 20 M motions, we evaluate search effectiveness by a *recall* metric. The recall determines the percentage of the same motions retrieved by the PPP-Codes with respect to the sequential scan for each k-NN query. The recall values and search times are then averaged for 1,000 randomly selected k-NN queries. Figure 10 shows values of k-NN recall (left vertical axis) and of search times (right axis) with respect to the absolute number of accessed vectors on the 20 M database. We can see that the PPP-Codes can access two orders of magnitude fewer motions than the sequential scan, while achieving a very high recall. For example, we can achieve the 96 % recall by accessing only 10,000 vectors (out of 20 M) in 800 ms for $k = 1$. Note that the search time is not influenced by the setting of k but only by the number of accessed feature vectors.

To sum up, assuming an average motion length of 260 frames as in the HDM05-122 ground truth, the 20 M motion database constitutes **1.4 years** of motion data which we can search in real time. We believe that our similarity method can be utilized in future motion retrieval technologies since it is both highly effective and scalable when equipped with an appropriate index structure, such as the PPP-Codes.

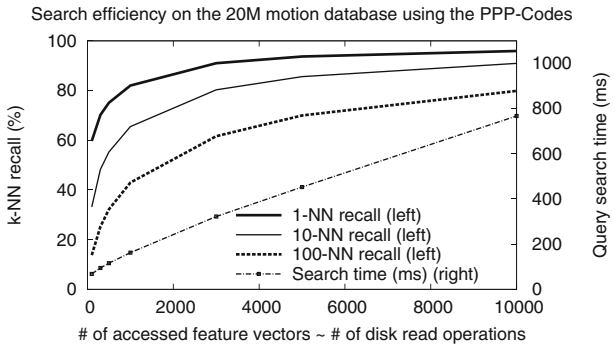

Fig. 10 k-NN recall and search times with respect to the number of accessed feature vectors stored on SSD disk using the PPP-Codes structure

4.8 Comparison with the state-of-the-art methods

We compare effectiveness of the proposed similarity method against state-of-the-art approaches in Table 1. The table depicts only the approaches reaching the highest achieved precision on different subsets of motion datasets. Since existing methods work almost perfectly on MHAD and CMU datasets with a smaller number of 11 and 30 categories, we primarily focus on the most challenging HDM05 dataset having up to 130 categories.

There is the only one approach of Elias et al. [13] that enables pair-wise motion comparison. They use a similar concept of motion images but compare them by MPEG-7 visual descriptors. We outperform this approach on the HDM05-14 dataset by increasing the search precision from 87.4 % to 94.3 %, even if the neural network is not fine-tuned on the HDM05-14 ground truth. All the other stated methods [9, 12] are purposely trained classifiers to achieve the highest possible accuracy just in the classification task. Even if our method is not a classifier, we simply implement it by searching for the nearest neighbor motion (1-NN query) and taking the category label of the retrieved motion as the classification result. We demonstrate that our approach is only slightly worse in classification effectiveness on the HDM05-65 dataset than the best classifiers [9, 12]. On the other hand, the best methods need 90 % of training data, while our approach achieves a high accuracy of 93.5 % with only 50 % of training data on the HDM05-65 dataset. In addition, we reach the 91.7 % accuracy with 90 % of training data by classifying the HDM05-122 motion set into 122 categories, which is almost two times more categories than it is considered in the state-of-the-art papers.

The main advantage of our similarity method is *efficiency* and *applicability* in large-scale retrieval due to the possibility of pair-wise motion comparison. Although the surveyed classifiers may achieve slightly higher effectiveness, they pay a little attention to efficiency,

Table 1 Classification rates of state-of-the-art approaches compared to our search-based 1-NN approach on selected subsets of four motion datasets: MHAD [36], MSR [23], CMU and HDM05 [33]

Dataset	# of categories	# of motions	Training motions	Accuracy	Approach
MHAD	11	659	58 %	100 %	Chaudhry et al. [7]
			58 %	100 %	Du et al. [12]
MSR	20	557	50 %	92.5 %	Vemulapalli et al. [46]
			50 %	94.5 %	Du et al. [12]
CMU-14	14	267	85 %	98.1 %	Wu et al. [53]
CMU-30	30	278	80 %	99.6 %	Kadu et al. [16]
HDM05-14	14	1,034	0 %	87.4 %	Elias et al. [13]
			0 %	94.3 %	Our approach
HDM05-65	65	2,345	90 %	95.6 %	Cho et al. [9]
			90 %	96.9 %	Du et al. [12]
			50 %	93.5 %	Our approach
			90 %	93.9 %	Our approach
HDM05-122	122	2,338	50 %	87.0 %	Our approach
			90 %	91.5 %	Our approach

The fourth column "Training motions" presents a percentage of motions (with respect to the number of dataset motions) which are used in the training phase

scalability and applicability issues. In particular, with respect to our approach, the surveyed classifiers:

- Cannot apply any indexing;
- Are infeasible to be used for (subsequence) searching or motion clustering;
- Can classify only samples from categories that have been available during the training phase.

Importantly, we clearly outperform related work in efficiency. We can index and search datasets comprising months of motion data in real time – such data volumes are not considered in related work at all. For instance, the approach in [4] needs about 36 seconds to search the 100 K motion database, which is more than three orders of magnitude slower than our approach. For best-performing classifiers [9, 12], it is even infeasible to process a database of 20 million motions in real time.

5 Conclusions

We present an effective and efficient method for similarity searching in motion capture data. The similarity is based on extracting 4,096-dimensional feature vectors for rather short motion sequences of variable lengths. These feature vectors are extracted from a compact visualization of normalized motions using a fine-tuned deep neural network. The machine-learning part enables to effectively learn intrinsic motion characteristics of the training data, while the distance-based comparison enables efficient indexing by any multi-dimensional or metric-based index structure.

We demonstrate that our search-based approach achieves competitive effectiveness results even if it is compared to specifically trained machine-learning classifiers. Moreover, we achieve the 91.5 % precision using 1-NN queries with 90 % of training data on the HDM05-122 ground truth with 122 categories, which is almost two times more categories than it is considered in the state-of-the-art papers. We can additionally achieve the 87.0 % precision by utilizing only 50 % of training data on the same HDM05-122 ground truth. We also demonstrate that the accuracy remains almost unchanged even if the data quality decreases ten times to 12 frames per second. Having the data quality of 120 fps, the method can recognize up to 10 times faster actions almost without any error. Moreover, only a slight inaccuracy occurs when recognizing badly-segmented actions up to 20 % of their length. On the other hand, high efficiency can be reached by indexing the fixed-size vectors using any metric-based index structure. For example, by employing the PPP-Codes structure, we can easily search a 20 M database of rather short motions in real time.

The main advantages against state-of-the-art approaches are that (1) feature vectors have a fixed size for motions having a variable length in order of seconds, which enables their efficient indexing, (2) features have a high descriptive power even for motion categories that have not been provided during the training phase, and (3) our approach is not just a simple classifier but can be also used for a wide range of applications, such as clustering and retrieval.

In the future, we plan to compress the feature vectors that are sparse in descriptive values and dense in zeros. We also intend to study how to utilize our approach for stream processing (e.g., for action detection and recognition) and for motion data containing imprecise joint coordinates.

Acknowledgements This research was supported by GBP103/12/G084.

References

1. Barnachon M, Bouakaz S, Boufama B, Guillou E (2013) A real-time system for motion retrieval and interpretation. Pattern Recogn Lett 34(15):1789–1798
2. Barnachon M, Bouakaz S, Boufama B, Guillou E (2014) Ongoing human action recognition with motion capture. Pattern Recogn 47(1):238–247
3. Baumann J, Wessel R, Krüger B., Weber A (2014) Action graph: a versatile data structure for action recognition. In: International conference on computer graphics theory and applications (GRAPP 2014). SCITEPRESS, pp 1–10
4. Beecks C, Hassani M, Obeloer F, Seidl T (2015) Efficient query processing in 3D motion capture databases via lower bound approximation of the gesture matching distance. In: 2015 IEEE International symposium on multimedia (ISM 2015), pp 148–153
5. Bouchard D, Badler N (2007) Semantic segmentation of motion capture using Laban movement analysis. Springer Berlin Heidelberg, Berlin Heidelberg, pp 37–44
6. Cai M, Zou B, Gao H, Song J (2014) Motion recognition for 3d human motion capture data using support vector machines with rejection determination. Multimed Tools Appl 70(2):1333–1362
7. Chaudhry R, Ofli F, Kurillo G, Bajcsy R, Vidal R (2013) Bio-inspired dynamic 3d discriminative skeletal features for human action recognition. In: Computer vision and pattern recognition workshops (CVPRW 2013), pp 471–478
8. Chen X, Koskela M (2013) Classification of RGB-D and motion capture sequences using extreme learning machine. Image Anal 640–651
9. Cho K, Chen X (2013) Classifying and visualizing motion capture sequences using deep neural networks. CoRR arXiv:abs/1306.3874
10. Cui J, Liu Y, Xu Y, Zhao H, Zha H (2013) Tracking generic human motion via fusion of low- and high-dimensional approaches. IEEE Trans Syst Man Cybern: Syst 43(4):996–1002
11. Donahue J, Jia Y, Vinyals O, Hoffman J, Zhang N, Tzeng E, Darrell T (2014) Decaf: a deep convolutional activation feature for generic visual recognition. In: International conference in machine learning (ICML 2014), pp I–647–I–655
12. Du Y, Wang W, Wang L (2015) Hierarchical recurrent neural network for skeleton based action recognition. In: International conference on computer vision and pattern recognition (CVPR 2015), pp 1110–1118
13. Elias P, Sedmidubsky J, Zezula P (2015) Motion images: an effective representation of motion capture data for similarity search. In: 8th International conference on similarity search and applications (SISAP 2015). Springer, pp 250–255
14. Huynh DQ (2009) Metrics for 3d rotations: comparison and analysis. J Math Imag Vis 35(2):155–164
15. Ijjina E, Mohan C (2015) Human action recognition based on motion capture information using fuzzy convolution neural networks. In: 8th International conference on advances in pattern recognition (ICAPR 2015), pp 1–6
16. Kadu H, Kuo CC (2014) Automatic human mocap data classification. IEEE Trans Multimed 16(8):2191–2202
17. Kapadia M, Chiang IK, Thomas T, Badler NI, Kider JT Jr (2013) Efficient motion retrieval in large motion databases. In: ACM SIGGRAPH Symposium on interactive 3D graphics and games (I3D 2013). ACM, New York, pp 19-28
18. Keogh E, Palpanas T, Zordan VB, Gunopulos D, Cardle M (2004) Indexing large human-motion databases. In: 30th International conference on very large data bases (VLDB 2004), VLDB 2004, pp 780–791. VLDB Endowment
19. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Pereira F, Burges C, Bottou L, Weinberger K (eds) Advances in neural information processing systems 25. Curran Associates Inc, pp 1097–1105
20. Krüger B, Tautges J, Weber A, Zinke A (2010) Fast local and global similarity searches in large motion capture databases. In: ACM SIGGRAPH/Eurographics symposium on computer animation, SCA 2010. Eurographics Association, pp 1–10
21. Lan R, Sun H (2015) Automated human motion segmentation via motion regularities. Vis Comput 31(1):35–53
22. Li M, Leung H (2016) Graph-based representation learning for automatic human motion segmentation. Multimed Tools Appl 75(15):9205–9224
23. Li W, Zhang Z, Liu Z (2010) Action recognition based on a bag of 3d points. In: Computer vision and pattern recognition workshops (CVPRW 2010), pp 9–14

24. Liang Y, Lu W, Liang W, Wang Y (2014) Action recognition using local joints structure and histograms of 3d joints. In: 10th International conference on computational intelligence and security (CIS 2014), pp 185–188
25. Liu Y, Zhang X, Cui J, Wu C, Aghajan H, Zha H (2010) Visual analysis of child-adult interactive behaviors in video sequences. In: 16th International conference on virtual systems and multimedia, pp 26–33
26. Liu Y, Cui J, Zhao H, Zha H (2012) Fusion of low-and high-dimensional approaches by trackers sampling for generic human motion tracking. In: 21st International conference on pattern recognition (ICPR 2012), pp 898–901
27. Liu Y, Nie L, Han L, Zhang L, Rosenblum DS (2016) Action2activity: recognizing complex activities from sensor data. CoRR arXiv:abs/1611.01872, 1–7
28. Liu Y, Nie L, Liu L, Rosenblum DS (2016) From action to activity: sensor-based activity recognition. Neurocomputing 181:108–115. Big data driven intelligent transportation systems
29. Liu L, Cheng L, Liu Y, Jia Y, Rosenblum DS (2016) Recognizing complex activities by a probabilistic interval-based model. In: AAAI, pp 1266–1272
30. Lu Y, Wei Y, Liu L, Zhong J, Sun L, Liu Y (2016) Towards unsupervised physical activity recognition using smartphone accelerometers. Multimed Tools Appl 1–19
31. Milovanovic M, Minovic M, Starcevic D (2013) Walking in colors: human gait recognition using kinect and cbir. IEEE MultiMed 20(4):28–36
32. Müller M, Röder T, Clausen M (2005) Efficient content-based retrieval of motion capture data. In: ACM SIGGRAPH. ACM, pp 677–685
33. Müller M, Röder T, Clausen M, Eberhardt B, Krüger B, Weber A (2007) Documentation Mocap Database HDM05. Tech. Rep. CG-2007-2 Universität Bonn
34. Müller M, Baak A, Seidel HP (2009) Efficient and robust annotation of motion capture data. In: ACM SIGGRAPH/Eurographics symposium on computer animation (SCA 2009). ACM Press, pp 17–26
35. Novak D, Zezula P (2014) Rank aggregation of candidate sets for efficient similarity search. In: 25th Int. Conference on database and expert systems applications (DEXA 2014), pp 42–58
36. Ofli F, Chaudhry R, Kurillo G, Vidal R, Bajcsy R (2013) Berkeley mhad: a comprehensive multimodal human action database. In: International workshop on applications of computer vision (WACV 2013), pp 53–60
37. Poppe R, Van Der Zee S, Heylen DJ, Taylor P (2014) Amab: automated measurement and analysis of body motion. Behav Res Methods 46(3):625–633
38. Presti LL, Cascia ML (2016) 3D skeleton-based human action classification: a survey. Pattern Recogn 53:130–147
39. Raptis M, Kirovski D, Hoppe H (2011) Real-time classification of dance gestures from skeleton animation. In: ACM SIGGRAPH/Eurographics symposium on computer animation (SCA 2011), SCA 2011. ACM, pp 147–156
40. Rousseeuw P (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math 20(1):53–65
41. Sedmidubsky J, Valcik J, Zezula P (2013) A key-pose similarity algorithm for motion data retrieval. In: Advanced concepts for intelligent vision systems (ACIVS 2013), LNCS, vol 8192. Springer, pp 669–681
42. Sedmidubsky J, Elias P, Zezula P (2016) Similarity searching in long sequences of motion capture data. In: 9th International conference on similarity search and applications (SISAP 2016). Springer, pp 271–285
43. Shahroudy A, Liu J, Ng TT, Wang G (2016) Ntu rgb+d: a large scale dataset for 3d human activity analysis. In: IEEE Conference on computer vision and pattern recognition (CVPR), pp 1010–1019
44. Trajcevski G, Ding H, Scheuermann P, Tamassia R, Vaccaro D (2007) Dynamics-aware similarity of moving objects trajectories. In: 15th Annual ACM international symposium on advances in geographic information systems, GIS '07. ACM, New York, pp 11:1–11:8
45. Valcik J, Sedmidubsky J, Zezula P (2016) Assessing similarity models for human-motion retrieval applications. Comput Anim Virt Worlds 27(5):484–500
46. Vemulapalli R, Arrate F, Chellappa R (2014) Human action recognition by representing 3d skeletons as points in a lie group. In: International conference on computer vision and pattern recognition (CVPR 2014), pp 588–595
47. Vögele A, Krüger B, Klein R (2014) Efficient unsupervised temporal segmentation of human motion. In: ACM Symposium on computer animation, pp 167–176
48. Wang JY, Lee HM (2009) Recognition of human actions using motion capture data and support vector machine. In: World Congress on software engineering (WCSE 2009), vol 1, pp 234–238
49. Wang Y, Neff M (2015) Deep signatures for indexing and retrieval in large motion databases. In: 8th ACM SIGGRAPH conference on motion in games. ACM, pp 37–45

50. Wang J, Liu Z, Wu Y, Yuan J (2012) Mining actionlet ensemble for action recognition with depth cameras. In: International conference on computer vision and pattern recognition (CVPR 2012). IEEE Computer Society, pp 1290–1297
51. Wang H, Su H, Zheng K, Sadiq S, Zhou X (2013) An effectiveness study on trajectory similarity measures. In: 24th Australasian database conference, ADC '13. Australian Computer Society, Inc., Darlinghurst, pp 13–22
52. Wang X, Chen L, Jing J, Zheng H (2016) Human motion capture data retrieval based on semantic thumbnail. Multimed Tools Appl 75(19):11,723–11,740
53. Wu S, Wang Z, Xia S (2009) Indexing and retrieval of human motion data by a hierarchical tree. In: 16th ACM Symposium on virtual reality software and technology (VRST 2009). ACM Press, New York, pp 207–214
54. Zanfir M, Leordeanu M, Sminchisescu C (2013) The moving pose: an efficient 3d kinematics descriptor for low-latency action recognition and detection. In: International conference on computer vision (ICCV 2013), pp 2752–2759
55. Zezula P, Amato G, Dohnal V, Batko M (2006) Similarity search: the metric space approach, advances in database systems, vol 32. Springer-Verlag
56. Zhao X, Li X, Pang C, Zhu X, Sheng QZ (2013) Online human gesture recognition from motion data streams. In: 21st International conference on multimedia (MM 2013). ACM, pp 23–32

Jan Sedmidubsky is a researcher of computer science at Masaryk University (Czech Republic) where he received the Ph.D. degree in 2011. His research activities are concentrated on efficient similarity search algorithms, face recognition and motion capture data processing.

Petr Elias received the M.Sc. degree at Masaryk University (Czech Republic). Currently, he is a Ph.D. student and focuses on handling motion capture data with a special interest on action recognition.

Pavel Zezula is a professor of computer science at Masaryk University (Czech Republic). His professional interests mainly focus on contentbased retrieval, large-scale similarity search, and big data analysis. He is a co-author of a famous metric-based similarity search structure MTree and book "Similarity Search: The Metric Space Approach".