Jiang ZF, Li W, Huang Y *et al*. PESTA: An elastic motion capture data retrieval method. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 38(4): 867-884 July 2023. DOI: 10.1007/s11390-023-3140-y

PESTA: An Elastic Motion Capture Data Retrieval Method

Zi-Fei Jiang[1] (蒋子飞), Wei Li[1] (李　伟), Yan Huang[1] (黄　艳), Yi-Long Yin[1] (尹义龙)
C.-C. Jay Kuo[2] (郭宗杰), *Fellow, IEEE*, and Jing-Liang Peng[3, 4, *] (彭京亮), *Member, IEEE*

[1] *School of Software, Shandong University, Jinan 250101, China*

[2] *Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California Los Angeles 90089, U.S.A.*

[3] *Shandong Provincial Key Laboratory of Network Based Intelligent Computing, Jinan 250022, China*

[4] *School of Information Science and Engineering, University of Jinan, Jinan 250022, China*

E-mail: jiangzifei@mail.sdu.edu.cn; wli@sdu.edu.cn; yan.h@sdu.edu.cn; ylyin@sdu.edu.cn; cckuo@sipi.usc.edu
ise_pengjl@ujn.edu.cn

Received February 1, 2023; accepted April 26, 2023.

Abstract Prevalent use of motion capture (MoCap) produces large volumes of data and MoCap data retrieval becomes crucial for efficient data reuse. MoCap clips may not be neatly segmented and labeled, increasing the difficulty of retrieval. In order to effectively retrieve such data, we propose an elastic content-based retrieval scheme via unsupervised posture encoding and strided temporal alignment (PESTA) in this work. It retrieves similarities at the sub-sequence level, achieves robustness against singular frames and enables control of tradeoff between precision and efficiency. It firstly learns a dictionary of encoded postures utilizing unsupervised adversarial autoencoder techniques and, based on which, compactly symbolizes any MoCap sequence. Secondly, it conducts strided temporal alignment to align a query sequence to repository sequences to retrieve the best-matching sub-sequences from the repository. Further, it extends to find matches for multiple sub-queries in a long query at sharply promoted efficiency and minutely sacrificed precision. Outstanding performance of the proposed scheme is well demonstrated by experiments on two public MoCap datasets and one MoCap dataset captured by ourselves.

Keywords motion capture (MoCap), content-based retrieval, adversarial autoencoder, temporal alignment

1 Introduction

Motion capture (MoCap) is widely employed for film production, video game development, computer-assisted medical diagnosis and so forth. Prevalent use of motion capture is producing large volumes of MoCap data. With a large database (or repository) of MoCap sequences at hand, it is often crucial to retrieve the right sequences or sub-sequences for synthesis or analysis. Correspondingly, content-based MoCap data retrieval has been intensively researched in recent years.

Existing content-based MoCap data retrieval methods can be classified in different ways. They can be classified into two categories, one that retrieves whole MoCap sequences[1–21] and the other that retrieves sub-MoCap-sequences (or sub-sequences for short) similar to the query[22–33]. In general, the former runs with higher efficiency by matching at a coarser granularity without temporally aligning the retrieved sequences or sub-sequences and the query, while the latter runs less efficiently due to the similarity search at a finer granularity. Existing algorithms may also be classified into supervised[14–17, 20, 21, 27] and unsupervised[1–13, 18, 19, 22–26, 28–33] ones. The former, though leading to high performance as reported, usually require huge data labeling efforts, are sensitive to labeling ambiguities, may not generalize well as limited by the scope of labelling, and may not be readily applicable for sub-sequence similarity search.

Regular Paper
This work was supported by Shandong Provincial Natural Science Foundation of China under Grant No. ZR2022MF294.
*Corresponding Author

We are motivated to make a generic scheme for MoCap data retrieval that works on MoCap clips, either segmented and labeled or not, and locates similarities, either holistic or partial, between MoCap clips. Therefore, we take an approach of unsupervised sub-sequence retrieval. Specifically, we propose a novel elastic scheme for content-based MoCap data retrieval via unsupervised posture encoding and strided temporal alignment (PESTA). The core of the scheme learns a dictionary of encoded postures using unsupervised neural network based adversarial autoencoder (AAE) techniques[34], symbolizes each MoCap sequence based on the learned dictionary, and temporarily aligns a given query to the repository sequences to find sub-sequence similarities. Further, it extends for efficient retrieval by multiple sub-queries inside a long query. The most remarkable contributions of the proposed scheme are listed as follows.

1) *Unsupervised Dictionary-Based Motion Abstraction.* The proposed scheme learns a posture dictionary using the neural network of unsupervised AAE techniques and, based on which, converts each raw MoCap sequence into a compact string of symbols. This abstract motion representation serves as the basis for the similarity search.

2) *Robust Sequence Alignment Method.* Distinctive from most existing sub-sequence retrieval methods, our proposed strided temporal alignment (STA) method relaxes the constraint of continuity and is resilient to intermittently occurring singular frames in the alignment.

3) *Elastic Framework Configuration.* The proposed framework is elastic in several aspects. It may be used for both sub-sequence retrieval and whole-sequence retrieval. It supports both holistic queries and sub-queries (with acceleration) for the retrieval. Flexible tradeoff between precision and speed of retrieval may be achieved by adjusting the posture dictionary size. Further, it is easy to extend the repository as no data labelling is required.

We provide a detailed introduction to PESTA in this paper as follows. In Section 2, we survey the related work on content-based MoCap data retrieval. In Section 3, we formulate the problem we address. In Sections 4–7, we explain how PESTA operates for different kinds of MoCap data retrievals. In Section 8, we demonstrate the effectiveness of PESTA through experiments. In Section 9, we summarize our findings and suggest future work.

Code and data of this work[①] will be posted to facilitate further research in this field.

2 Related Work

In recent years, many algorithms have been proposed for content-based MoCap data retrieval[35]. We divide them into two categories: one that retrieves whole MoCap sequences and the other that retrieves sub-MoCap-sequences similar to the query.

2.1 Whole-Sequence Retrieval

Liu *et al.*[1] and Deng *et al.*[2] used hierarchical structures for indexing motions or fusing similarities. Liu *et al.*[1] made elastic matching to compare the key frame sets of two MoCap sequences, while Deng *et al.*[2] turned a body part's motion into a string of motion pattern indices and used fast string matching for similarity measurement. Lv *et al.*[3] also employed a tree structure, named minimal motion spanning tree, to construct a motion signature for each MoCap sequence, which statistically describes its high-level and low-level morphological and kinematic characteristics.

Some algorithms employ matrix factorization to derive motion descriptors. Jin and Prabhakaran[4] applied singular value decomposition to motion data matrices and described a motion by the histogram of the semantically quantized singular values. Sun *et al.*[5] conducted a low-rank subspace decomposition on a motion sequence volume representation to obtain three rank-1 tensors for motion description. Zhu *et al.*[6] performed sparse decomposition in the quaternion space and used the dictionary component to describe a motion. Wang *et al.*[7] transformed six feature matrices of the whole body and body parts to eigenspaces to obtain the eigenvectors for motion description.

Some algorithms match motions based on key motion frames' or segments' features. Xiao *et al.*[8] constructed a weighted bipartite graph of two key frame sets and measured their similarity by weighted graph matching. Qi *et al.*[9] trained a Gaussian mixture model for every motion class and used them with the sparse coding for retrieval. Sedmidubsky *et al.*[10] obtained a dictionary of motion words (MWs) by clustering on fixed-length motion segments, represented

[①]https://github.com/iSunnii/umr, July 2023.

any MoCap sequence by a sequence of MWs and used mature text-retrieval models to complete the MoCap data retrieval.

Some algorithms use energy features to describe human motions. Li *et al.*[11] proposed to use a motion energy image to represent each MoCap sequence. Feng *et al.*[12] represented a motion by potential energy and kinetic energy of a human torso. They also used entropy to measure the intensity of different joints in a motion to find the most remarkable ones. Xiao *et al.*[13] combined the earth movers distance with the quaternion description to measure the similarity between a query motion and the motions in the database.

Recently, some deep neural network based algorithms have been proposed for motion classification and retrieval. Deep[14–16] and recurrent[17] neural networks constitute the best-performing methods in recognizing actions. Though highly effective, these algorithms are inherently designed for classifying actions and hardly employed to quantify pair-wise similarity, which is required for the task of data retrieval. Some exceptions[18–21] exist where discriminative feature vectors are extracted for similarity comparison. Holden *et al.*[18] used a deep convolutional autoencoder to learn a human motion manifold from 160-frame motion clips and represented each 160-frame motion clip by its latent feature. For each MoCap sequence, Wang and Neff[19] extracted a 160-bit feature called deep signature by a deep autoencoder, and Sedmidubsky *et al.*[20] extracted a 4 096-dimensional motion feature vector by a fine-tuned deep convolutional neural network (DCNN). Lv *et al.*[21] extracted a hash code for each motion clip by an adapted VGG16 network with a hash layer added.

Among the algorithms reviewed in this subsection, the deep neural network based ones[14–17, 20, 21] take a supervised approach, while the rest unsupervised. These supervised methods[14–17, 20, 21] require huge data labeling efforts, are sensitive to labeling ambiguities, are not directly applicable to sub-sequence similarity search, and may not generalize well as limited by the scope of labelling.

2.2 Sub-Sequence Retrieval

As a general technique for aligning two temporal sequences, dynamic time warping (DTW) has also been adapted for sub-sequence retrieval by some algorithms. The sub-sequence retrieval algorithms not using and using DTW and its variants are reviewed in the following two paragraphs, respectively.

Sakamoto *et al.*[22] built a motion map using the self-organizing map (SOM) technique, with each node indexing to the corresponding postures in the motion files. It is used to retrieve sub-sequences from the repository corresponding to a user-specified node sequence. Krüger *et al.*[23] found repository poses similar to those in a query with the help of a kd-tree structure, constructed a lazy neighborhood graph and found end-constrained paths in it for the result. Choi *et al.*[24] rendered all repository motions into stick figures and allowed a user to draw stick figures to specify a query. All the files that contain stick figures similar to and in the same order as the user-drawn ones are retrieved. Kapadia *et al.*[25] described motions around key frames by motion keys and used a trie-based data structure to map user-specified key sequences to motions. Xiao *et al.*[26] allowed the user to specify the query by sketching five-stroke human postures. They devised a two-dimensional geometric posture descriptor and conducted a posture-by-posture motion matching. Sedmidubsky *et al.*[27] made multi-level segmentation of long MoCap sequences and conducted the retrieval on the segment level, based on the segments' motion features as extracted by a fine-tuned DCNN.

Kovar and Gleicher[28] pre-constructed a match web using 1D minimum heuristics and dynamic programming (DTW in essense) for each pair of repository MoCap sequences and used the match webs for efficient intra-repository query. Müller *et al.*[29] and Müller and Röder[30] all used boolean geometric relation features for motion description. The former work indexes the motion repository by inverted lists, computes coarse matches on the segment level and refines the alignment by DTW; the latter learns a motion template for each motion class and matches a motion class' motion template to repository feature matrices using sub-sequence DTW. Forbes and Fiume[31] projected the motion data matrix into the weighted principal component analysis space for reduced dimensionality and used an adapted DTW method to find matching sub-sequences. Wu *et al.*[32] learned a two-dimensional (2D) SOM of key postures and represented each MoCap sequence with a string of indices into the SOM. Motifs are derived to help quickly locate candidate matches and the Smith-Waterman (SW) algorithm, an adapted DTW algorithm, is used for string matching. Gupta *et al.*[33] used a video clip as the query. They used dense trajectories and 2D relational pose features to describe each video or MoCap frame, and conducted sub-sequence DTW for the retrieval. Compared with the non-DTW based approaches, the DTW-based ones are based on finer modelings of frame-to-frame distances and make finer and more flexible temporal alignment in general.

The sub-sequence retrieval algorithms reviewed in this subsection can effectively deal with unsegmented MoCap clips that may have highly variant lengths. Among them, the one by Wu *et al.*[32] is the most related to ours, as it is also an unsupervised, automatic and generic method for MoCap-to-Mocap data retrieval, while the rest have different orientations and/or extra requirements. Krüger *et al.*[23] quickly identified just the very closest matches without paying much attention to general precison and recall ratios. Gupta *et al.*[33] used a video clip as a query and conducted video-to-MoCap data retrieval. Some methods require user interaction in query specification[22, 24–26], feature selection[29] or characteristic point specification[31]. Kovar and Gleicher[28] and Müller and Röder[30] required motion types to be known. Sedmidubsky *et al.*[27] required data labelling to make a supervised learning.

3 Problem Statement

In order to facilitate a quick reference, we firstly list in Table 1 the symbols that will be consistently used later in the text.

Table 1. Symbols Used in This Work

Symbol	Meaning
$\boldsymbol{Q}$	Query (a MoCap sequence)
$\boldsymbol{E}$	Symbolized query
B	MoCap database or MoCap repository
$\boldsymbol{M}$	Matrix containing all the raw repository MoCap sequences
$\boldsymbol{S}$	Symbolized representation of the whole MoCap repository
$\boldsymbol{B}$	Raw MoCap sequence
$\boldsymbol{G}$	Symbolized MoCap sequence

We formally define the core problem of sub-MoCap-sequence retrieval as follows. For a query, $\boldsymbol{Q}=(\boldsymbol{f}_1,\boldsymbol{f}_2,\ldots,\boldsymbol{f}_q)$, and a MoCap database, $B=\{\boldsymbol{B}_1,\boldsymbol{B}_2,\ldots,\boldsymbol{B}_b\}$, where $\boldsymbol{f}_i\in\mathbb{R}^{h\times 1}$, $i\in[1,q]$, represents the i-th frame (represented by an h-dimensional vector or an $h\times 1$ matrix) in the query and $\boldsymbol{B}_j\in\mathbb{R}^{h\times n_j}$, $j\in[1,b]$, represents the j-th repository MoCap sequence of n_j frames, the maximum set of sub-sequences, $R=\{\boldsymbol{R}_1,\boldsymbol{R}_2,\ldots,\boldsymbol{R}_r\}$, is to be found from B such that, $\forall i\in[1,r],\exists j\in[1,b],\boldsymbol{R}_i\sqsubseteq\boldsymbol{B}_j$ and $d(\boldsymbol{Q},\boldsymbol{R}_i)<U_d$ with $\sqsubseteq$ standing for the sub-sequence relationship, $d(.)$ being a distance metric and U_d a distance threshold. For the convenience of experimental evaluation, we set r flexibly and form R by the r sub-sequences from B with the smallest distances to $\boldsymbol{Q}$. For the convenience of description, we use a matrix, $\boldsymbol{M}=(\boldsymbol{B}_1,\boldsymbol{B}_2,\ldots,\boldsymbol{B}_b)=(\boldsymbol{p}_1,\boldsymbol{p}_2,\ldots,\boldsymbol{p}_n)\in\mathbb{R}^{h\times n}, n=\sum_{i=1}^{b}\times n_i$, to record the raw data of all the MoCap sequences in the database, and may interpret a column matrix as a vector and vice versa later in the text.

4 Overview of PESTA

In our design of the MoCap data retrieval solution, we take the strategy of posture symbolization and string matching. We make this choice as it enables flexible control of the granularity of motion encoding and is suited for both holistic and partial similarity search.

Our proposed solution to the core unsupervised sub-MoCap-sequence retrieval problem is composed of two stages: the preprocessing stage and the retrieval stage, as shown in Fig.1. In the preprocessing stage, a symbolization process is conducted on all the repository MoCap sequences. It firstly employs the unsupervised adversarial autoencoding technique to learn a dictionary of (encoded) postures. Based on the dictionary, it then represents each repository motion with a concise symbolic sequence. In the retrieval stage, the query is firstly converted to a concise symbolic sequence based on the pre-learned posture dictionary, which is then aligned to the symbolic repository sequences to retrieve similar symbolic sub-sequences and the corresponding raw MoCap sub-sequences.

Further, the core scheme is extended for efficient retrieval by multiple sub-queries inside a long query. The extended scheme takes a segment-by-segment approach to quickly form the quasi-optimal temporal alignments between the sub-queries and the repository sequences.

Note that the strategy of posture or segment symbolization and string matching has been adopted by related work (e.g., [2, 10, 32]) as well. However, our algorithm distinguishes in the following major aspects: 1) descriptive and discriminative posture codes learned by AAE, 2) deep reduction of temporal redundancy through repetitive symbol pattern detection, 3) flexible and robust temporal alignment be-

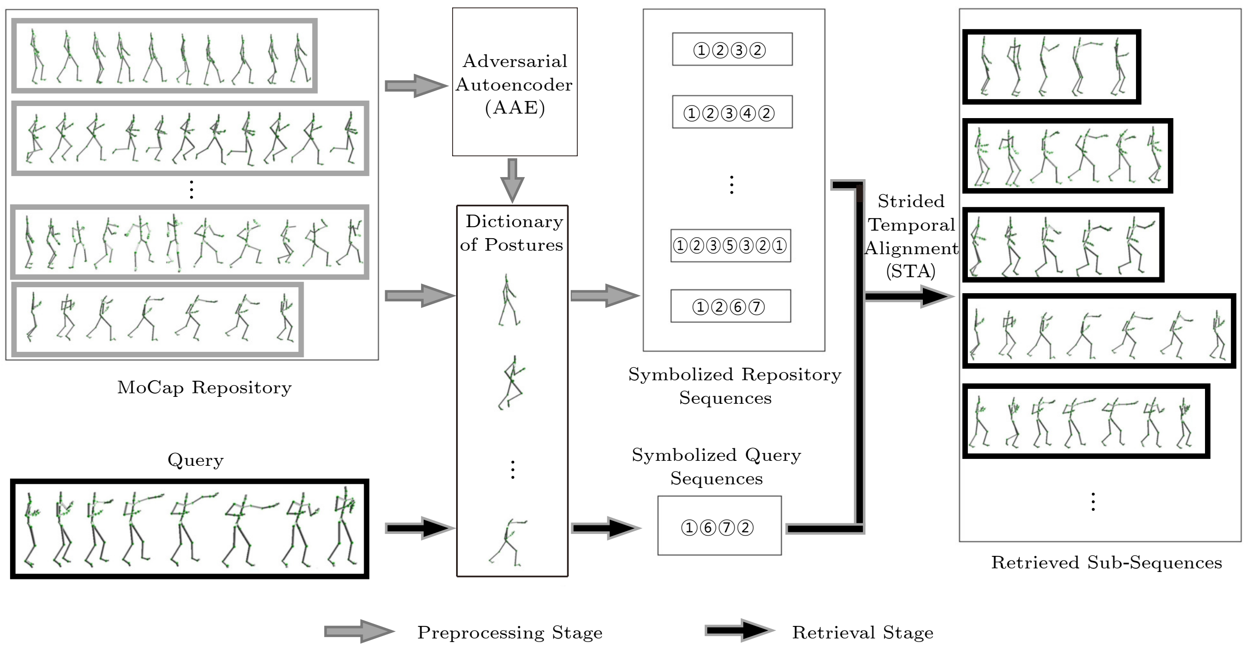


Fig.1. Flowchart of the proposed core scheme. In the preprocessing stage, a dictionary of (encoded) postures is learned by the unsupervised adversarial autoencoding technique and, based on which, the repository motions are represented with symbolic sequences. In the retrieval stage, the query is converted to a symbolic sequence and then aligned to the symbolic repository sequences to retrieve similar symbolic sub-sequences.

tween sequences, and 4) a novel solution to efficient retrieval by multiple sub-queries inside a long query.

Details of the algorithmic components are provided in Section 5–Section 7.

5 Motion Symbolization Based on Posture Encoding

5.1 Overall Strategy

The raw MoCap data usually have high dimensionalities and contain vast spatial and temporal redundancies[36], which need to be reduced for compact and discriminative motion description. Of various approaches taken by the existing MoCap data retrieval algorithms for the redundancy reduction, one (as taken by Deng *et al.*[2], Wu *et al.*[32] and Sedmidubsky *et al.*[10]) is of our particular interest. It firstly generates a dictionary of postures or motion segments and then turns each raw MoCap sequence into a sequence of keys according to the dictionary, which leads to reduced spatial redundancy. For the dictionary construction, all the methods operate in the raw posture or motion space but not other spaces that may better capture discriminative features of the data samples. Regarding the reduction of temporal redundancy in a symbolic sequence, these methods make no special effort or simply reduce consecutive occurrences of a single symbol into one.

In this work, we also adopt the approach of dictionary-based MoCap sequence symbolization but make more effective reduction of spatial and temporal redundancies. Specifically, for spatial redundancy reduction, we make a compact and descriptive dictionary of encoded postures, deeply exploiting the discriminative features of the postures by AAE; for temporal redundancy reduction, we reduce consecutive occurrences of the same symbol pattern into one, employing a repetitive symbol pattern detection technique.

5.2 Spatial Redundancy Reduction

A raw MoCap frame contains each joint's orientation with respect to its parent. An orientation may be represented by a rotation in the form of Euler angles. However, the Euler angle representation is not unique, considering that different choices and orderings of rotation axes may be adopted and angles 2π apart around the same axis correspond to the same orientation. By contrast, an orientation can be represented by either of two unit quaternions with opposite signs. Always picking the one with the first element being nonnegative, we reach a unique quaternion representation, which is adopted in this work.

Concatenating the quaternions of all J joints in the skeleton, we form a long vector, $\boldsymbol{f} \in \mathbb{R}^{h\times 1}$, $h = 4J$, to represent a raw MoCap frame.

In order to reduce the spatial redundancies existent in MoCap frames, we learn a compact dictionary of encoded postures in an unsupervised fashion and represent a MoCap frame by a symbol (or key) pointing to an atom in the dictionary with the closest posture. In essence, unsupervised dictionary learning is a process that clusters the non-labeled training samples and computes a representative (or named head) for each cluster. While a variety of techniques (e.g., K-means clustering[2], K-medoids clustering[10] and SOM[32]) have been used for learning dictionaries of postures or motion segments, we choose to use the unsupervised AAE technique[34], as reasoned below.

In general, an autoencoder is capable of extracting a descriptive and discriminative representation of the input. Makhzani *et al.*[34] enhanced a probabilistic autoencoder architecture by integrating GAN[37] to regularize the distribution of the latent codes. Further, Makhzani *et al.*[34] showed an extended AAE network for unsupervised clustering that exhibited excellent performance in disentangling class and style information. Both the code regularization and class and style separation enhance the basic autoencoder's power in making descriptive and discriminative codes. Therefore, we choose it for our posture clustering. Our choice is supported by experiments as well in the ablation study (see Subsection 8.4) that shows the sharp advantage of the extended AAE over K-means and SOM.

The extended AAE architecture (or AAE for short) for unsupervised clustering is diagramed in Fig.2(a). Assume that the dataset contains m classes of samples and we are reducing the dimensionality of each sample to n. For an input $\boldsymbol{x}$, the encoder predicts an m-dimensional discrete class variable $\boldsymbol{y}$ and an n-dimensional continuous latent variable $\boldsymbol{z}$ according to $q(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x})$, and the decoder reconstructs an output $\boldsymbol{x}'$ that is similar to $\boldsymbol{x}$ with $\boldsymbol{y}$ and $\boldsymbol{z}$. Two adversarial networks impose the categorial distribution $Cat(\boldsymbol{y})$ on $\boldsymbol{y}$ and the Gaussian distribution $N(\boldsymbol{z}|0, \boldsymbol{I})$ on $\boldsymbol{z}$, respectively. As a result, the class variable $\boldsymbol{y}$ corresponding to the categorial distribution does not carry any style information and the style variable $\boldsymbol{z}$ is ensured to be a continuous Gaussian variable. The encoder, the decoder and the two discriminators each have two layers of 1 000 hidden units. Each layer is with the ReLU activation function except that the second layer of the encoder is with the Tanh activation function.

Specifically, we make the spatial redundancy reduction in the following way. Firstly, we use a training posture set, $T = \{\boldsymbol{t}_1, \boldsymbol{t}_2, \dots, \boldsymbol{t}_u\}$, to train the unsupervised clustering AAE. During the training, the m cluster heads in the dimensionality-reduced space are automatically learned, which form our dictionary of encoded postures, $U = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_m\}$. For each pair of atoms, $\boldsymbol{u}_p$ and $\boldsymbol{u}_q$, in U, we compute their Euclidean distance, $d_{p,q} = |\boldsymbol{u}_p - \boldsymbol{u}_q|_2$ to form a posture distance matrix $\boldsymbol{D} = [d_{p,\ q}]_{m\times m}$ which will be used later for temporal alignment of symbolized MoCap sequences. With the AAE trained, we symbolize all the repository MoCap sequences, as part of the preprocessing, and any query given later. Denoting a MoCap sequence as $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_C) \in \mathbb{R}^{h\times C}$, the trained AAE predicts the class variable of $\boldsymbol{x}_c$ $(1 \leqslant c \leqslant C)$ that indicates the most probable class, the k-th $(1 \leqslant k \leqslant m)$ class with k being the key of $\boldsymbol{u}_k$

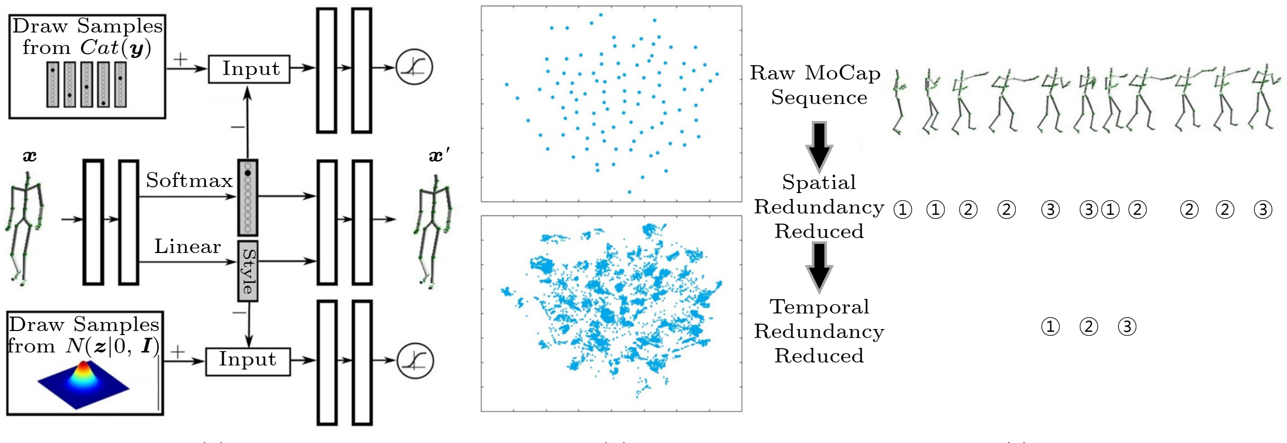

Draw Samples from $Cat(\boldsymbol{y})$
+
Input
−
$\boldsymbol{x}$
Softmax
$\boldsymbol{x}'$
Linear
Style
−
Draw Samples from $N(\boldsymbol{z}|0, \boldsymbol{I})$
+
Input
Raw MoCap Sequence
Spatial Redundancy Reduced
① ① ② ② ③ ③ ① ② ② ② ③
Temporal Redundancy Reduced
① ② ③
(a)
(b)
(c)

Fig.2. Process of motion symbolization. (a) Extended AAE architecture for unsupervised posture clustering. (b) 2D projections of cluster heads (top) and encoded training posture samples (bottom). (c) Spatial and temporal redundancy reduction.

in U and used as the symbol of y_c. As a result, $\boldsymbol{X}$ is converted into a symbolic sequence $\boldsymbol{Y}=(y_1,y_2,\dots,y_C)$.

In this work, we train the AAE with $m=400$ for all the algorithm evaluations in Section 8 except the test of elasticity on the dictionary size in Subsection 8.5.1. All the MoCap sequences in a repository are quarterly downsampled in the temporal dimension with the sampled postures randomly divided into two parts, three quarters for the training set and one quarter for the validation set. The autoencoder and the two adversarial networks are trained jointly with the Adam optimizer. We use the same loss function as Makhzani *et al.*[34]. The networks are trained for 5 000 epochs, using a gradient clipping value of 5.0, a momentum value of 0.1, a learning rate of 10^{-4} and a batch size of 64.

We illustrate in Fig.2(b) the clustering result on our first experimental dataset (see Subsection 8.1), where the top and the bottom parts visualize the 2D projections of the cluster heads and the encoded training posture samples, respectively, using the t-SNE (t-distributed stochastic neighbor embedding) technique. Here we use $m=200$ for the ease of illustration. We observe in Fig.2(b) that the encoded postures show an evident clustering tendency and the cluster heads are well apart from each other.

5.3 Temporal Redundancy Reduction

In a symbolized MoCap sequence resulting from the spatial redundancy reduction, we often observe segments of identical symbols, e.g., ①①①, mostly due to dense temporal sampling of the MoCap system. We remove this type of redundancy by keeping only one symbol out of each segment of this kind. In the thus-simplified symbolic sequence, we continue to search for segments of repetitive symbol patterns, e.g., ①②③①②③, mostly due to cyclic motions of the subject with a symbol pattern signifying a semantic behavioral unit, e.g., a walking cycle. We remove this type of redundancy by keeping only one instance of the symbol pattern out of each segment of this kind. The segments of identical symbols can be easily detected through a one-pass traversal of the symbolic sequence. For detecting segments of repetitive symbol patterns, we employ the suffix array technique[38]. Again it is worth noting that the related work[2, 10, 32] makes no special effort to reduce repetitive multi-symbol patterns.

In Fig.2(c), we create a brief example to illustrate the effects of spatial and temporal redundancy reductions on a raw MoCap sequence, according to a learned posture dictionary. A real example is further given in Fig.3, showing that a 341-frame raw cartwheel MoCap sequence is sharply reduced to an eight-symbol sequence by the proposed method. Fig.3(a) shows the frame samples of the raw MoCap data, with the corresponding dictionary keys marked below; Fig.3(b) presents the symbolized final sequence, shown by both dictionary keys and postures reconstructed from the corresponding dictionary atoms; Fig.3(c) shows that, in a symbolized MoCap sequence, one symbol usually corresponds to a long (43-frame for this case) segment in the raw MoCap sequence. Fig.3 demonstrates visually that AAE can effectively cluster similar postures while summarizing the whole MoCap sequence.

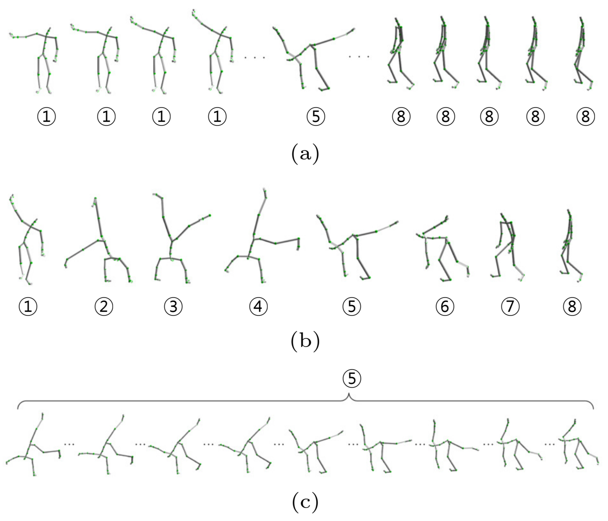


Fig.3. A real example of a 341-frame cartwheel MoCap sequence symbolized to an eight-symbol sequence. (a) Symbolized 341 raw frames. (b) Symbolized final sequence with eight symbols. (c) 43 raw frames corresponding to one symbol.

6 Strided Temporal Alignment

As formulated in Section 3, the MoCap database is denoted as $B=\{\boldsymbol{B}_1,\boldsymbol{B}_2,\dots,\boldsymbol{B}_b\}$, where $\boldsymbol{B}_i\in\mathbb{R}^{h\times n_i}$, $i\in[1,b]$, represents the i-th repository MoCap sequence of n_i frames each represented by an $h\times 1$ matrix. All MoCap sequences in the repository are concatenated into a long sequence, $\boldsymbol{M}=(\boldsymbol{B}_1,\boldsymbol{B}_2,\dots,\boldsymbol{B}_b)=(\boldsymbol{p}_1,\boldsymbol{p}_2,\dots,\boldsymbol{p}_n)\in\mathbb{R}^{h\times n}, n=\sum_{i=1}^{b}n_i$. Symbolizing $\boldsymbol{B}_i\in\mathbb{R}^{h\times n_i}$ into $\boldsymbol{G}_i\in\mathbb{N}_+{}^{1\times c_i}$ and concatenating $\boldsymbol{G}_i$, $1\leqslant i\leqslant b$, we obtain a symbolic representation of the whole repository, $\boldsymbol{S}=(\boldsymbol{G}_1,\boldsymbol{G}_2,\dots,\boldsymbol{G}_b)=(s_1,s_2,\dots,s_Y)\in\mathbb{N}_+{}^{1\times Y}, Y=\sum_{i=1}^{b}c_i$. In addition, we store necessary index correspondences between $\boldsymbol{S}$ and $\boldsymbol{M}$ that enable mapping any sub-sequence in $\boldsymbol{S}$ to one in $\boldsymbol{M}$.

Given a query, $\boldsymbol{Q}=(\boldsymbol{f}_1,\boldsymbol{f}_2,\dots,\boldsymbol{f}_q)$, it is symbolized to $\boldsymbol{E}=(e_1,e_2,\dots,e_X)\in\mathbb{N}_+^{1\times X}$. We propose to make a strided temporal alignment of $\boldsymbol{E}$ to $\boldsymbol{S}$ to extract sub-sequences of $\boldsymbol{S}$ that are most similar to $\boldsymbol{E}$ and, correspondingly, sub-sequences of $\boldsymbol{M}$ that are most similar to $\boldsymbol{Q}$.

Specifically, we use dynamic programming to compute a weight matrix, $\boldsymbol{W}\in\mathbb{R}^{X\times Y}$, by

$$\boldsymbol{W}(x,y)=\begin{cases}\boldsymbol{D}(e_x,s_y), \text{ if } x=1,\\ \min\limits_{(m,n)\in A}\Big(\boldsymbol{W}(m,n)+\boldsymbol{D}(e_x,s_y)\times p\Big),\\ \qquad\qquad \text{if } x\in(1,X],\end{cases}$$

where

$$A=\{[\max(1,x-w),x]\times[\max(1,y-w),y]-(x,y)\},$$

and w is named stride corresponding to a $(w+1)\times(w+1)$ local search window and $p=\max(x-m,y-n)$ is the penalty factor. During the iterative process, the optimal $(m,\ n)$ tuple is recorded as a backward pointer with each $\boldsymbol{W}(x,y),\ x\in(1,X]$. Any element in the last row, $\boldsymbol{W}(X,y)$, represents the distance between $\boldsymbol{E}$ and a best-matching sub-sequence, $\boldsymbol{V}$, from $\boldsymbol{S}$ that ends at the y-th symbol in $\boldsymbol{S}$. Following the backward pointers, we obtain an optimal aligning path through all the rows of $\boldsymbol{W}$. The indices of the columns that this path crosses give $\boldsymbol{V}$. Denoting the raw MoCap sub-sequence in $\boldsymbol{M}$ corresponding to $\boldsymbol{V}$ as $\boldsymbol{T}$, we define $d(\boldsymbol{Q},\boldsymbol{T})=\boldsymbol{W}(X,y)$.

In order to retrieve sub-sequences of $\boldsymbol{S}$ that are the most similar to $\boldsymbol{E}$, we traverse the elements in the last row of $\boldsymbol{W}$ in the ascending order of their weights. Starting from each element, we trace backward to the first row of $\boldsymbol{W}$ to obtain a similar sub-sequence. During the search, we invalidate a similar sub-sequence if it overlaps with any of the valid similar sub-sequence(s) obtained so far. In case a thus-identified similar sub-sequence crosses the boundary between two adjacent symbolized MoCap sequences, $\boldsymbol{G}_i$ and $\boldsymbol{G}_{i+1}$, $1\leqslant i<b$, it may be either kept or discarded, depending on the users' need. The search stops when a desired number of valid similar sub-sequences have been retrieved. With a similar sub-sequence located inside $\boldsymbol{S}$, it is easy to locate the corresponding sub-sequence in $\boldsymbol{M}$ based on the pre-stored index correspondences between $\boldsymbol{S}$ and $\boldsymbol{M}$.

As reviewed in Subsection 2.2, DTW and its variants have been used for MoCap data retrieval. There frequently exist singular frames in a MoCap sequence, resulting from sensory noises, computing errors or brief minor motions mixed inside major ones. However, the common DTW-based methods do not deal with this situation well. We show a simple example of STA-based and DTW-based temporal alignments in Fig.4 to illustrate the difference between the two methods. The symbolized postures of the query and a repository sequence are presented on the left and the bottom, respectively. The postures marked by solid boxes correspond to singular frames in the sequences. For convenience of illustration, we show the pairwise posture distance matrix but not the weight matrix in Fig.4, where a lighter shade means a smaller distance. The dashed and the dotted paths show the alignments between the query and two sub-repository-sequences, respectively, using STA, and the solid path shows the alignment between the query and the whole repository sequence using DTW. The dashed and the solid boxes on the paths indicate the compared neighbors by two methods, respectively, when computing the weight of the element in the lower right corner of each box. For this example, $w=3$ is used.

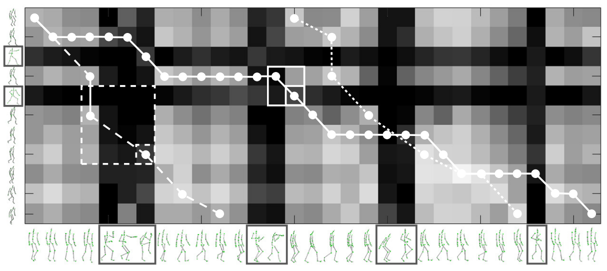


Fig.4. Example of the STA-based and the DTW-based temporal alignments.

When computing $\boldsymbol{W}(x,y)$, DTW-based methods search for the minimum only from at most three direct neighbors in $\boldsymbol{W}$ that have been computed, as illustrated in Fig.4 by the solid box on the path. As a result, one or more segments of singular frames inside may override the match between two sequences that are largely similar. By contrast, the proposed STA method conducts the minimum search in a local $(w+1)\times(w+1)$ window when computing $\boldsymbol{W}(x,y)$, $x\in(1,X]$, as illustrated in Fig.4 by the dashed box on the path. Therefore, it may bypass up to $w-1$ symbolized atoms that do not locally match well, making it robust against intermittently occurring singular frames inside overall similar sequences or sub-sequences.

From Fig.4, we observe that STA successfully strides over the singular frames in both the query and the repository sequence, while the standard DTW does not. As a result, the weight of the solid path is significantly higher than those of the other two paths, which may mistakenly indicate a mismatch between the query and the repository sequence. In other words, STA more effectively captures partial similarities of the repository sequence to the query, with robustness against intermittent singular frame occurrences.

7 Retrieval by Sub-Queries in a Long Query

There are often scenarios where we have a long query and wish to find similarities from the repository to various portions of the long query (or named sub-queries). Multiple sub-queries may be interactively specified by the user or automatically specified by a window sampling process (e.g., in the scanning for certain motions). In either way, there are often significant overlaps between sub-queries.

We assume that all the MoCap sequences in the repository and the query are symbolized, and may omit "symbolized" in the following to simplify the description. For a given query, $\boldsymbol{E}$, and sub-queries $\boldsymbol{SE}_1, \boldsymbol{SE}_2, \ldots, \boldsymbol{SE}_Z$, we aim to retrieve from the repository, $\boldsymbol{S}$, the set of similar sub-sequences, $\boldsymbol{V}_i$, for any sub-query, $\boldsymbol{SE}_i$, $1 \leqslant i \leqslant Z$. A straightforward approach is to treat each sub-query as a new query, and conduct the STA on it. When sub-queries overlap, however, temporal alignments are repetitively computed on each overlapping segment. In order to avoid such repetitive computation, we propose a segment-by-segment temporal alignment approach that divides the query into equal-length segments, computes the weight matrix by STA at most once for each segment and, for any given sub-query, uses the segments' weight matrices that are already computed to quickly extract the matching sub-sequences from the repository.

Specifically, we divide $\boldsymbol{E}$ into K equal-length segments, $\boldsymbol{E}_i$, $1 \leqslant i \leqslant K$, and denote their weight matrices when each aligned to $\boldsymbol{S}$ using STA by $\boldsymbol{W}_i$, $1 \leqslant i \leqslant K$. If the number of frames in $\boldsymbol{E}$ is not a multiple of K, we make it so by duplicating the last frame of $\boldsymbol{E}$ at the end. Then we conduct the retrieval for any given sub-query. For sub-query $\boldsymbol{SE}_i$, we get the minimum set of query segments, $\boldsymbol{E}_j$, $l_i \leqslant j \leqslant u_i$, that jointly cover it. If any of $\boldsymbol{W}_j$, $l_i \leqslant j \leqslant u_i$, is not computed yet, we compute and store it. Next, we initiate Y paths from the bottom row of $\boldsymbol{W}_{u_i}$ and extend them till the top row of $\boldsymbol{W}_{l_i}$.

The path extension process is illustrated in Fig.5 where the weight matrices, $\boldsymbol{W}_j$, $l_i \leqslant j \leqslant u_i$, are shown in grid structures with a lighter shade meaning a smaller weight. We initiate one path at each element in the bottom row of $\boldsymbol{W}_{u_i}$, and extend it by reversely following the optimal aligning path in $\boldsymbol{W}_{u_i}$, with one example shown in Fig.5(a). The cost of the current path is set to the initial element's weight. If $u_i = l_i$, the path extension is done. Otherwise, we extend each path further as shown by the example in Fig.5(b). Close to the top element of the current path, we select from a local $1 \times (w+1)$ ($w = 2$ for this example) window in the bottom row of $\boldsymbol{W}_{u_i-1}$ (black-framed in Fig.5(b)) the element with the smallest weight, c, append it to the current path and further extend the path by reversely following the optimal aligning path in $\boldsymbol{W}_{u_i-1}$. The current path is updated to the extended path and its cost is increased by c. This process is iterated until each path grows to the top row of $\boldsymbol{W}_{l_i}$, as shown by the example in Fig.5(c).

Finally, the paths with the smallest costs are selected. The portions of these paths corresponding to the frames of $\boldsymbol{SE}_i$ give the matching sub-sequences in $\boldsymbol{S}$, which in turn give the matching sub-sequences in $\boldsymbol{M}$ by the pre-stored index correspondences between $\boldsymbol{S}$ and $\boldsymbol{M}$.

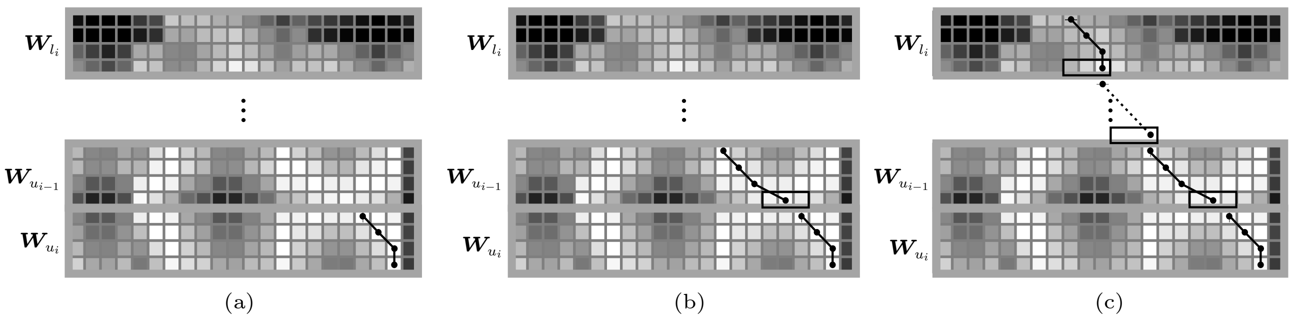


Fig.5. Path extension process for retrieval by a sub-query. (a) Path extension process inside the segment of the sub-query. (b) Path extension process between the segments of the sub-query. (c) Extension process of the whole path.

8 Experiments

In the experiments, we carefully design both the datasets and the experimental procedures to test the versatility and elasticity of the proposed algorithm, as detailed below.

8.1 Data and Platform

We compose three datasets for experiments. From the first to the third, the average length of the MoCap sequence, the average number of motion types in a MoCap sequence, and the fineness of labelling all increase. They are so designed for a full-range capability test of the proposed algorithm.

First Dataset. This is a homogeneous dataset with MoCap sequences taken from the HDM05 cut database[39], each containing a single type of motion with a label. This dataset includes 10 motion types, i.e., cartwheel, elbow-to-knee, hop, jog, jumping jack, kick, shuffle, squat, throw-sitting and walk, and 20 MoCap sequences for each motion type. In total, there are 200 MoCap sequences and frames in this dataset. The motions are performed by five actors/actresses, and each type may consist of different subtypes in terms of the number of the cycles, starting hand/foot, ending position, etc.

Second Dataset. This is a subset of the CMU MoCap database②. Since the database itself does not provide fine category annotations, we use the data and settings same to Gupta *et al.*'s[33], i.e., we use the same 2 000 shortest of the 2 549 sequences from the CMU database and the same annotations made by Gupta *et al.*[33] on these sequences. There are 1 900 416 frames in this dataset, corresponding to about 4.5 hours human motions. The annotations of eight motion types, i.e., down, get up, turn, walk, punch, kick, pick up and throw overhead, are used in the experiment. Note that a MoCap sequence may contain more than one type of motion, and a MoCap sequence may contain none of the eight motion types.

Third Dataset. This is a dataset captured by ourselves using a Vicon optical motion capture system. In contrast to the first and the second datasets, the third one consists of longer MoCap sequences each containing more types of motion. Besides, a complete frame-level labeling is made on the third dataset. In total, it consists of 45 long MoCap sequences of 12 motion types, which are captured from nine actors/actresses. Each long MoCap sequence is a mixture of an unfixed number of motion types, and one type of motion may appear for multiple times with different periods in the sequence. When acting a motion sequence, an actor/actress acts a randomly selected subset of motion types in a randomly determined order. The 12 motion types and their total times of occurrences in the 45 MoCap sequences are shown in Table 2. On the average, each sequence consists of eight types of motion. In total, there are 103 849 frames in this dataset.

Table 2. Motion Types and the Total Number of Occurrences of Each Motion Type in the Third Dataset

Motion Type	Total Number of Occurrences
Boxing	21
Tie	23
Leg press	22
Dribble	38
Jump shot	20
Arm raise	22
Arm wave	29
Crab walk	14
Cross step	13
Run circle	11
Rotate	13
Walk circle	14

All the experiments in this subsection are conducted on an iMac platform with an Intel i5-4570R @ 2.7 GHz CPU, 8 GB memory and 500 GB hard disk.

8.2 Evaluation Metrics

The metrics we use to evaluate the proposed algorithm include precision at n (P@n), precision-recall curve (P-R curve), and confusion matrix. For each query, the fraction of relevant samples in the result set gives the precision, while the fraction of all relevant samples that have been returned gives the recall. If the result set has a size of n, the precision gives P@n. By varying n, we obtain a P-R curve of this query. When $n = N$, with N being the size of the database, the average precision, AP, of this query is computed by

$$AP = \frac{1}{R}\sum_{j=1}^{N} I_j \times \frac{R_j}{j},$$

where R is the number of relevant samples in the

②http://mocap.cs.cmu.edu, March 2023.

database, $I_j = 1$ if the j-th ranking sample of the result set is relevant and $I_j = 0$ otherwise, and R_j is the number of relevant samples in the j top-ranking samples of the result set. By averaging the statistics of every query in a motion class, we obtain the P@n, P-R and mean average precision (MAP) statistics for their class. A confusion matrix is a square matrix where each element corresponds to a specific motion class for both rows and columns. The values within the matrix represent the frequence of motion sequences from the row class being returned as relevant to the column class. In the confusion matrices used in this paper, larger values are represented with darker grayscale.

8.3 Full Algorithm Evaluations

In this subsection, we evaluate our proposed algorithm with other unsupervised MoCap data retrieval algorithms on the three datasets. For the evaluation, we conduct three experiments, each emphasizing the capability of our proposed method in a different aspect. As the stride, $w = 3$ is used in the first two experiments and $w = 2$ is used in the third. In STA, a larger w leads to higher robustness against singular frames but more computation at the same time. Therefore, we empirically set the value of w to balance robustness and efficiency. The three experiments are detailed below.

8.3.1 Whole-Sequence Retrieval

This experiment is conducted on the first and the second dataset.

On the first dataset, we compare our proposed method (PESTA) with the method using motion signatures[3] (abbreviated as "MS"), the method using weighted graphs[8] (abbreviated as "WG"), the method using motion segment dictionaries[10] (abbreviated as "MW"), the method using deep signatures[19] (abbreviated as "DS"), and the method using SOMs[32] (abbreviated as "SOM"). Among the algorithms reviewed in Section 2, MS[3], WG[8], MW[10] and SOM[32] are the latest non-deep-learning based methods while DS[19] is one latest deep learning based method suited for unsupervised whole-sequence MoCap-to-MoCap data retrieval. We implement WG, DS and MW by ourselves and SOM with the help of its authors, and use the implementation of MS (unsupervised version) provided by the original paper. For MW, we implement the multi-overlay version and use the 1/5 settings in [10] for best retrieval performance.

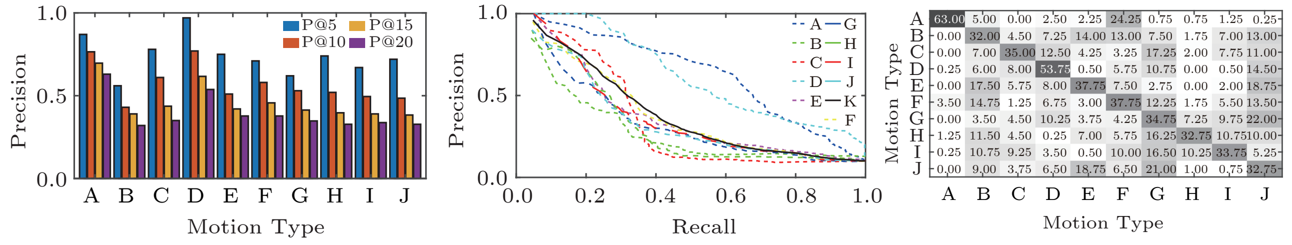
We also implement and test another latest deep learning based approach by Holden *et al.*[18] for unsupervised whole-sequence retrieval. The performance is quite low. Reasons may include that our implementation differs in details like joint sampling that are not fully specified by Holden *et al.*[18], and that their approach is not optimized for motion sequences with highly varied frame lengths and motion cycles as present in our first dataset. Since the test dataset is not specified in their paper, we choose not to show this part of experimental results to avoid biased evaluation.

When testing an algorithm, we use each MoCap sequence in the dataset as the query for once to retrieve similar sequences from the repository. When testing PESTA, for a given query, we return the whole repository sequences that contain the most similar sub-sequences to the query as the result. We use the performance statistics gathered from all the retrievals to draw P@n $(n = 5, 10, 15, 20)$ statistics and precision-recall curves, and use the top 20 retrieving results in each retrieval to draw the confusion matrices. The P@n statistics, P-R curves and confusion matrices of the algorithms are shown in Fig.6, which show a clear performance advantage of PESTA over the others.

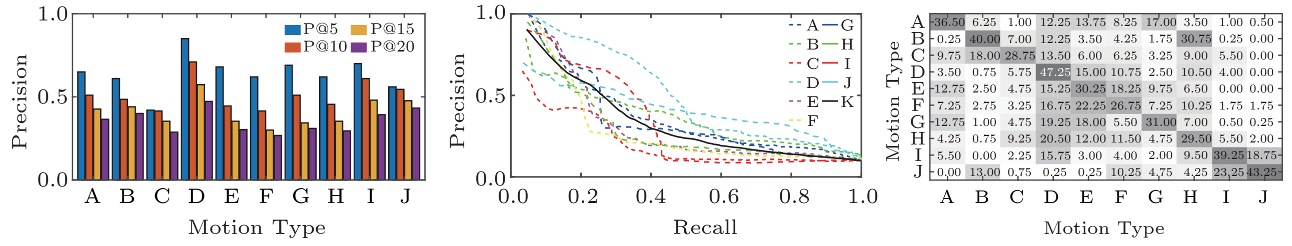
On the second dataset, we compare PESTA with WG and SOM. We do not make comparison with MS, DS and MW as they are originally designed for segmented MoCap sequences, each including a single motion type, while a MoCap sequence in the second dataset may include more than one motion type. In this experiment, we manually extract 10 examples from the dataset for each motion type as the queries. Table 3 shows the per-type and the overall MAP statistics of the algorithms. We clearly observe that PESTA outperforms WG and SOM in most of the motion types.

8.3.2 Sub-Sequence Retrieval by Holistic Queries

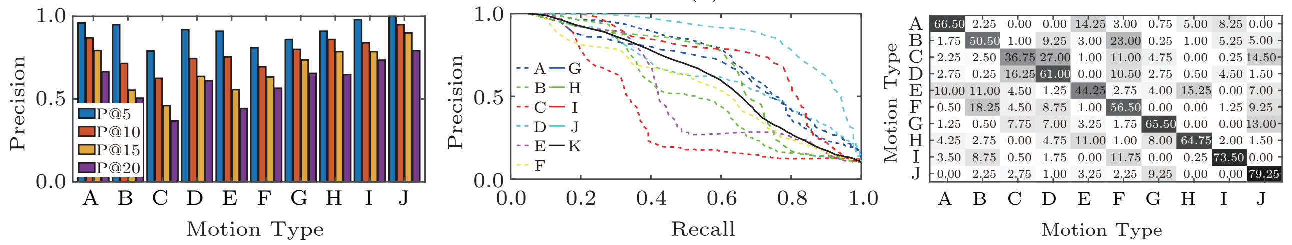
This experiment is conducted on the third dataset. We compare PESTA with SOM that is the most related to our method for sub-sequence retrieval, as explained in Subsection 2.2. For each of the 12 motion types, we manually extract 10 examples from the long MoCap sequences, which together form the query set. Each query is used once for the retrieval. A returned sub-sequence may contain frames from more than one motion type, and is classified to the type



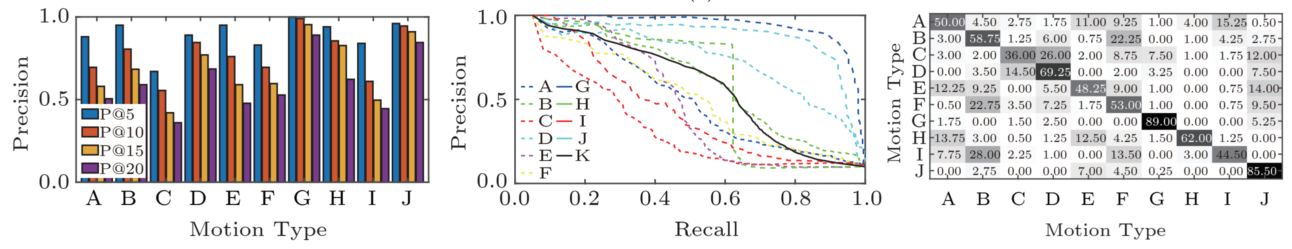
(a)



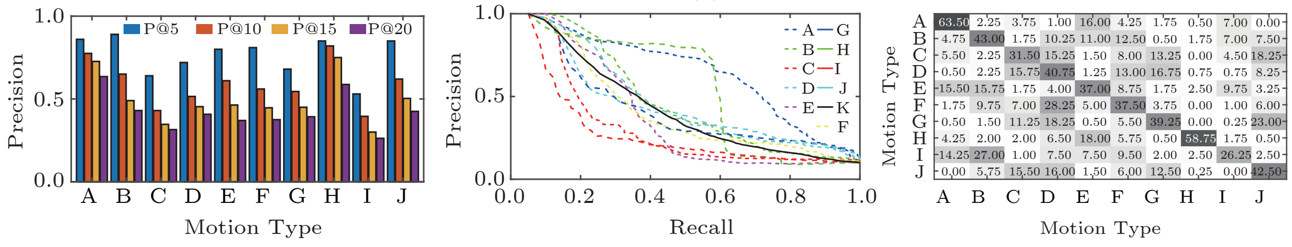
(b)



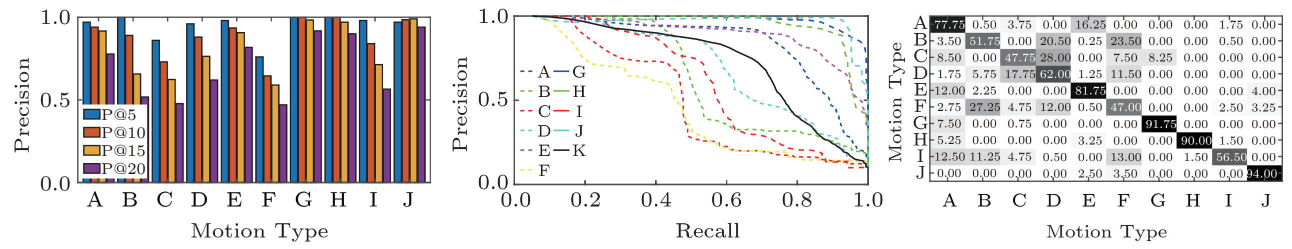
(c)



(d)



(e)



(f)

Fig.6. P@n statistics (the first column), precision-recall curves (the second column) and confusion matrices (the third column) of whole-sequence retrieval of (a) MS[3], (b) WG[8], (c) MW[10], (d) DS[19], (e) SOM[32] and (f) PESTA on the first dataset. The labels A to J represent cartwheel, elbow-to-knee, hop, jog, jumping jack, kick, shuffle, squat, throw-sitting, and walk, respectively. The label K in precision-recall curves represents the mean curve.

Table 3. Per-Type and Overall MAP Statistics of WG[8], SOM[32] and PESTA on the Second Dataset

	Sit down	Get up	Turn	Walk	Punch	Kick	Pick up	Throw Overhead	Overall
#Seq.	30.000	56.000	194.000	739.000	13.000	23.000	76.000	17.000	
WG[8]	0.069	0.052	0.305	0.543	0.096	0.062	0.072	0.107	0.163
SOM[32]	0.123	0.078	0.360	0.537	**0.161**	0.108	0.094	0.141	0.200
PESTA	**0.139**	**0.115**	**0.513**	**0.563**	0.138	**0.111**	**0.112**	**0.163**	**0.232**

Note: *#Seq.* is the number of motion sequences in the dataset containing a motion type. The overall column represents the MAP statistics of all eight motion types. Result in bold is the best in each column.

with the largest number of frames. A hit is made if and only if the returned sub-sequence and the query have the same motion type. The performance statistics are gathered from all the retrievals to draw P@n ($n = 2, 5, 10$) statistics and P-R curves, and the top 10 retrieving results in each retrieval are used to draw the confusion matrices. The P@n statistics, P-R curves and confusion matrices of SOM and PESTA on the query set are shown in Fig.7, which show a clear performance advantage of PESTA over SOM.

8.3.3 Sub-Sequence Retrieval by Sub-Queries

This experiment is conducted on the third dataset. In this experiment, retrieval by a long query is implemented by multiple retrievals by sub-queries. As such, we use SOM and PESTA as benchmark methods that treat each sub-query as a fresh short query and conduct a fresh sub-sequence retrieval by the holistic short query. Our proposed method for retrieval by sub-queries of a long query as described in Section 7 is denoted as PESTA-LQ. We compare PESTA-LQ with PESTA and SOM. We use each sequence in the third dataset as a long query, and use a $(3 \times s)$-frame window sliding at a stride of s $(s = 4)$ through each symbolized long query to sample sub-sequences. If a sample has no less than 60% frames of one motion type, it is classified to this type and kept as a sub-query. Otherwise, it is discarded. A retrieved sub-sequence is classified to the motion type with the largest number of frames in it. A hit is made if and only if the retrieved sub-sequence and the sub-query have the same motion type. The average P@n statistics and the time cost of SOM, PESTA and PESTA-LQ are shown in Table 4, from which we observe that both PESTA and PESTA-LQ achieve significantly higher precision of retrieval than SOM, PESTA-LQ achieves the highest time efficiency and, compared with PESTA, PESTA-LQ leads to slightly lower precision of retrieval but sharply reduced time of retrieval. In this experiment, PESTA-LQ reduces the retrieval time of PESTA by 47.6%, showing its

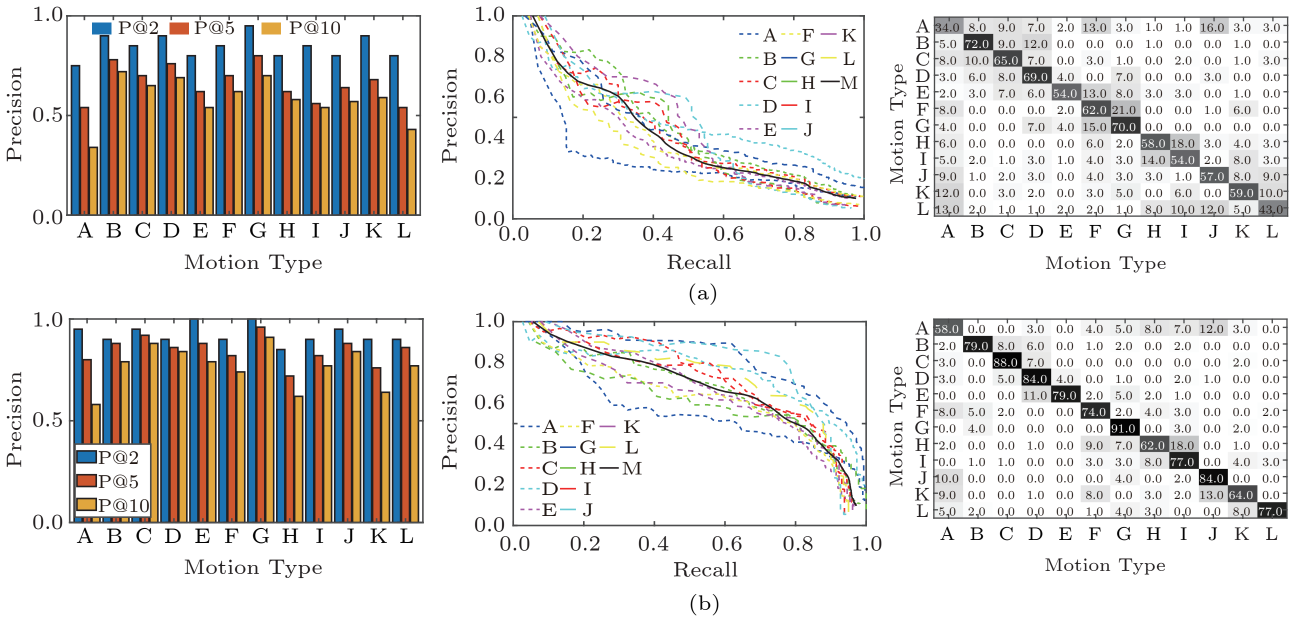


Fig.7. P@n statistics (the first column), precision-recall curves (the second column) and confusion matrices (the third column) of sub-sequence retrieval by holistic queries of (a) SOM[32] and (b) PESTA on the third dataset. The labels A to L represent boxing, tie, leg press, dribble, jump shot, arm raise, arm wave, crab walk, cross step, run circle, rotate, and walk circle, respectively. The label M in precision-recall curves represents the mean curve.

Table 4. P@n Statistics and Time Cost of Sub-Sequence Retrieval by Sub-Queries of SOM[32], PESTA and PESTA-LQ on the Third Dataset

	P@2	P@5	P@10	Time (s)
SOM[32]	0.675	0.596	0.479	10.75
PESTA	**0.913**	**0.824**	**0.716**	15.52
PESTA-LQ	0.886	0.773	0.675	**8.13**

Note: Result in bold is the best in each column.

big advantage in efficiency at slightly sacrificed precision of retrieval.

8.4 Ablation Study

In this subsection, we study the effects of key components in our proposed scheme through the following experiments.

Effect of AAE. In order to study the effect of AAE, we construct two benchmark algorithms by replacing the AAE module in our algorithm by K-means and SOM, which are frequently used for data clustering, and keeping the rest part unchanged. We conduct the whole-sequence retrieval on the first dataset, and collect the MAP statistics in Table 5, where K-means+STA and SOM+STA are the two benchmark algorithms and AAE+STA (i.e., PESTA) is our proposed scheme. On the average, the MAP of AAE+STA is 0.777, which is 31.9% higher than that of K-means+STA and 19.2% higher than that of SOM+STA. The sharp advantage of AAE over K-means and SOM is clearly demonstrated.

Effect of STA. In order to study the effect of STA, we construct two benchmark algorithms by replacing the STA module in our algorithm by DTW and SW and keeping the rest part unchanged. We conduct the whole-sequence retrieval on the first dataset, and collect the MAP statistics in Table 5, where AAE+DTW and AAE+SW are the two benchmark algorithms and AAE+STA (i.e., PESTA) is our proposed scheme. On the average, the MAP of AAE+STA is 0.777, which is 7.2% higher than that of AAE+DTW and 14.9% higher than that of AAE+SW. The advantage of STA over DTW and SW is clearly demonstrated. Further, we modify the original MoCap sequences in the first dataset to introduce singularities. This is done by replacing randomly selected 10-frame clips in each original MoCap sequence by randomly selected 10-frame clips from MoCap sequences of other motion types. We conduct the whole-sequence retrieval on the modified first dataset, and plot the singularity-MAP curves of AAE+DTW, AAE+SW and AAE+STA in Fig.8, where the horizontal axis represents the extent of singularity (the ratio of replaced clips) and the vertical axis represents the MAP. From Fig.8 we observe that MAP decreases when singularity increases for all the three methods while STA leads to higher robustness against singularity than DTW and SW.

Effect of Segment-by-Segment Temporal Alignment. For retrieval by sub-queries which is detailed in Section 7, a segment-by-segment approach is proposed to quickly derive the quasi-optimal temporal alignments between the sub-queries and the repository sequences. The effect of this segment-by-segment approach has already been demonstrated in Subsection 8.3.3 by the performance comparison between PESTA and PESTA-LQ.

8.5 Discussions

8.5.1 *Elasticity*

As verified in Subsection 8.3, the proposed scheme works for both whole-sequence and sub-sequence retrieval.

Further, the proposed scheme supports both holistic query and sub-queries for the retrieval. For the latter, it invents a segment-by-segment approach to quickly form quasi-optimal temporal alignments between the sub-queries and the repository sequences, leading to sharply promoted efficiency at slightly sac-

Table 5. MAP Statistics of Various Combinations of Data Clustering and Temporal Alignment Modules for Whole-Sequence Retrieval on the First Dataset

	1	2	3	4	5	6	7	8	9	10	Overall
K-means+STA	0.602	0.579	0.353	0.688	0.478	0.558	0.855	0.648	0.402	0.728	0.589
SOM+STA	0.590	0.643	0.389	**0.784**	0.501	**0.581**	0.964	0.656	0.500	0.907	0.652
AAE+DTW	0.660	0.511	**0.644**	0.714	0.736	0.478	0.946	0.868	**0.723**	0.971	0.725
AAE+SW	**0.929**	**0.675**	0.591	0.621	0.573	0.360	0.948	0.771	0.338	0.953	0.676
AAE+STA	0.853	0.647	0.562	0.736	**0.898**	0.535	**0.980**	**0.952**	0.628	**0.975**	**0.777**

Note: The following 10 motion types are represented by the labels 1 to 10 respectively: cartwheel, elbow-to-knee, hop, jog, jumping jack, kick, shuffle, squat, throw-sitting, and walk. The overall column represents the MAP statistics of all the 10 motion types. Result in bold is the best in each column.

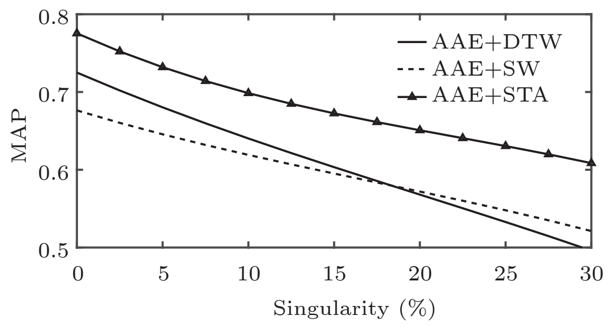


Fig.8. MAP curves of AAE+DTW, AAE+SW and AAE+STA (i.e., PESTA) on the first dataset under different percentages of singularity.

rificed precision of retrieval, as demonstrated in Subsection 8.3.3.

Since the proposed motion representation is based on a learned posture dictionary, the dictionary size may be adjusted to balance between the precision and the time cost of retrieval. To investigate the influences of dictionary size, we experiment with a wide range of dictionary sizes, collect the corresponding P@5, retrieval time (Time), and length of symbolized repository string (Y) statistics on the first dataset, and list the statistics in Table 6. Firstly, we observe from Table 6 that the trends of the quantity change of Time and P@5 roughly consist with that of Y. This is expected since, in general, the time to search from a string is positively related to the string length, and the precision of retrieval is positively related to the representational precision (or, in a rough correspondence, the symbolic string length) of the whole repository. Secondly, we observe from Table 6 that a larger dictionary leads to higher precision but also more time of retrieval in general, though local drops may happen, most notably at the dictionary sizes of 100, 500 and 800 in this example. The local drops may be explained by the fact that both the number and the distribution of words (encoded postures) in the dictionary determine the extent of temporal redundancy reduction. A larger dictionary with slightly fewer words representing postures in certain MoCap sequences of relative dominance in length may still lead to more aggressive reduction of overall temporal redundancy and thus a smaller Y, which may cause drops in precision and time of retrieval in turn.

Table 6. Performance Statistics of PESTA on the First Dataset Under Different Posture Dictionary Sizes

Dictionary Size	P@5	Time (s)	Y
50	0.791	0.069	7 673
100	0.782	0.053	6 941
200	0.816	0.050	6 536
300	0.832	0.064	7 532
400	0.892	0.113	10 035
500	0.881	0.100	9 460
600	0.890	0.136	11 101
700	0.887	0.190	13 089
800	0.887	0.173	12 468

Note: Time is the average retrieval time. Y is the whole length of symbolized repository.

As the proposed scheme is based on unsupervised learning, it is easy to extend the repository with no need for data labelling.

8.5.2 *Limitations*

When symbolizing a MoCap sequence, the proposed PESTA scheme reduces redundancies maximally to preserve the most skeletal motion semantics (see Section 5). As a result, it does not distinguish motions of the same type at different speeds (e.g., walking slowly and walking quickly) or motions with different cycles of the same pattern (e.g., walking two steps and walking four steps). This could be a drawback if speed and/or cycle count is considered as important in motion semantics. Nevertheless, this drawback may be addressed by fully or partially saving the repetitive symbol reduction and/or the repetitive symbol pattern reduction from the PESTA scheme.

Compared with most existing whole-sequence retrieval methods in general, the proposed PESTA scheme may be less time-efficient due to the extra effort it makes in fine-grained temporal alignment.

9 Conclusions

In this work, we proposed an elastic PESTA scheme for content-based MoCap data retrieval. We evaluated PESTA on two public datasets and one self-captured dataset for whole-sequence retrieval, sub-sequence retrieval by holistic queries and sub-sequence retrieval by sub-queries. The overall accuracy and the accuracy of each motion type of our algorithm significantly outperform the compared methods. For multiple sub-queries inside a long-query, the proposed extended scheme improves retrieval efficiency significantly with slight sacrifice of accuracy. In addition, we conducted ablation experiments on the PE module and STA module. Due to the elasticity of PESTA, the PE module is able to strike a balance between accuracy and efficiency, while the STA module can effectively handle singular frames.

In the future, it is worthwhile to extend the proposed scheme for more application scenarios. For instance, we may use a human motion video as the query, extract the posture sequence from it[40–43], and adapt the scheme of this work for the video-based motion retrieval. Further, it is interesting to transfer the concepts and methods of the proposed scheme for the retrieval of other types of time series data such as video and music data.

Conflict of Interest The authors declare that they have no conflict of interest.

References

[1] Liu F, Zhuang Y T, Wu F, Pan Y H. 3D motion retrieval with motion index tree. *Computer Vision and Image Understanding*, 2003, 92(2/3): 265–284. DOI: 10.1016/j.cviu.2003.06.001.

[2] Deng Z G, Gu Q, Li Q. Perceptually consistent example-based human motion retrieval. In *Proc. the 2009 Symposium on Interactive 3D Graphics and Games*, Feb. 2009, pp.191–198. DOI: 10.1145/1507149.1507181.

[3] Lv N, Jiang Z F, Huang Y, Meng X X, Meenakshisundaram G, Peng J L. Generic content-based retrieval of marker-based motion capture data. *IEEE Transactions on Visualization and Computer Graphics*, 2018, 24(6): 1969–1982. DOI: 10.1109/TVCG.2017.2702620.

[4] Jin Y, Prabhakaran B. Knowledge discovery from 3D human motion streams through semantic dimensional reduction. *ACM Trans. Multimedia Computing, Communications, and Applications*, 2011, 7(2): Article No. 9. DOI: 10.1145/1925101.1925104.

[5] Sun C, Junejo I, Foroosh H. Motion retrieval using low-rank subspace decomposition of motion volume. *Computer Graphics Forum*, 2011, 30(7): 1953–1962. DOI: 10.1111/j.1467-8659.2011.02048.x.

[6] Zhu M Y, Sun H J, Deng Z G. Quaternion space sparse decomposition for motion compression and retrieval. In *Proc. the 2012 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Jul. 2012, pp.183–192. DOI: 10.2312/SCA/SCA12/183-192.

[7] Wang P J, Lau R W H, Pan Z G, Wang J, Song H Y. An Eigen-based motion retrieval method for real-time animation. *Computers & Graphics*, 2014, 38: 255–267. DOI: 10.1016/j.cag.2013.11.008.

[8] Xiao Q K, Wang Y, Wang H Y. Motion retrieval using weighted graph matching. *Soft Computing*, 2015, 19(1): 133–144. DOI: 10.1007/s00500-014-1237-5.

[9] Qi T, Feng Y F, Xiao J, Zhuang Y T, Yang X S, Zhang J J. A semantic feature for human motion retrieval. *Computer Animation and Virtual Worlds*, 2013, 24(3/4): 399–407. DOI: 10.1002/cav.1505.

[10] Sedmidubsky J, Budikova P, Dohnal V, Zezula P. Motion words: A text-like representation of 3D skeleton sequences. In *Proc. the 42nd European Conference on Information Retrieval*, Apr. 2020, pp.527–541. DOI: 10.1007/978-3-030-45439-5_35.

[11] Li W, Huang Y, Peng J L. Video-interfaced human motion capture data retrieval based on the normalized motion energy image representation. In *Proc. the 27th International Conference on Neural Information Processing*, Nov. 2020, pp.616–627. DOI: 10.1007/978-3-030-63830-6_52.

[12] Feng L, Shen X, Sun T, Xu X H, Pan X W. Retrieval of human motion data based on energy model. *Journal of Computer-Aided Design & Computer Graphics*, 2007, 19(8): 1015–1021. (in Chinese)

[13] Xiao Q K, Li J F, Xiao Q H. Human motion capture data retrieval based on quaternion and EMD. In *Proc. the 5th International Conference on Intelligent Human-machine Systems and Cybernetics*, Aug. 2013, pp.517–520. DOI: 10.1109/IHMSC.2013.129.

[14] Huang Z W, Wan C D, Probst T, Van Gool L. Deep learning on lie groups for skeleton-based action recognition. In *Proc. the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, Jul. 2017, pp.6099–6108. DOI: 10.1109/CVPR.2017.137.

[15] Li C L, Cui Z, Zheng W M, Xu C Y, Ji R R, Yang J. Action-attending graphic neural network. *IEEE Trans. Image Processing*, 2018, 27(7): 3657–3670. DOI: 10.1109/TIP.2018.2815744.

[16] Zhu W T, Lan C L, Xing J L, Zeng W J, Li Y H, Shen L, Xie X H. Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks. In *Proc. the 30th AAAI Conference on Artificial Intelligence*, Feb. 2016, pp.3697-3703. DOI: 10.1609/aaai.v30i1.10451.

[17] Du Y, Wang W, Wang L. Hierarchical recurrent neural network for skeleton based action recognition. In *Proc. the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2015, pp.1110–118. DOI: 10.1109/CVPR.2015.7298714.

[18] Holden D, Saito J, Komura T, Joyce T. Learning motion manifolds with convolutional autoencoders. In *Proc. the SIGGRAPH Asia 2015 Technical Briefs*, Nov. 2015, Article No. 18. DOI: 10.1145/2820903.2820918.

[19] Wang Y Y, Neff M. Deep signatures for indexing and retrieval in large motion databases. In *Proc. the 8th ACM SIGGRAPH Conference on Motion in Games*, Nov. 2015, pp.37–45. DOI: 10.1145/2822013.2822024.

[20] Sedmidubsky J, Elias P, Zezula P. Effective and efficient similarity searching in motion capture data. *Multimedia Tools and Applications*, 2018, 77(10): 12073–12094. DOI: 10.1007/s11042-017-4859-7.

[21] Lv N, Wang Y, Feng Z Q, Peng J L. Deep hashing for motion capture data retrieval. In *Proc. the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing*, Jun. 2021, pp.2215–2219. DOI: 10.1109/ICASSP39728.2021.9413505.

[22] Sakamoto Y, Kuriyama S, Kaneko T. Motion map: Im-

age-based retrieval and segmentation of motion data. In *Proc. the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Aug. 2004, pp.259–266. DOI: 10.1145/1028523.1028557.

[23] Krüuger B, Tautges J, Weber A, Zinke A. Fast local and global similarity searches in large motion capture databases. In *Proc. the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Jul. 2010.

[24] Choi M G, Yang K, Igarashi T, Mitani J, Lee J. Retrieval and visualization of human motion data via stick figures. *Computer Graphics Forum*, 2012, 31(7): 2057–2065. DOI: 10.1111/j.1467-8659.2012.03198.x.

[25] Kapadia M, Chiang I K, Thomas T, Badler N I, Kider J T. Efficient motion retrieval in large motion databases. In *Proc. the 2013 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, Mar. 2013, pp.19–28. DOI: 10.1145/2448196.2448199.

[26] Xiao J, Tang Z P, Feng Y F, Xiao Z D. Sketch-based human motion retrieval via selected 2D geometric posture descriptor. *Signal Processing*, 2015, 113: 1–8. DOI: 10.1016/j.sigpro.2015.01.004.

[27] Sedmidubsky J, Elias P, Zezula P. Searching for variable-speed motions in long sequences of motion capture data. *Information Systems*, 2019, 80: 148–158. DOI: 10.1016/j.is.2018.04.002.

[28] Kovar L, Gleicher M. Automated extraction and parameterization of motions in large data sets. *ACM Trans. Graphics*, 2004, 23(3): 559–568. DOI: 10.1145/1015706.1015760.

[29] Müller M, Röder T, Clausen M. Efficient content-based retrieval of motion capture data. *ACM Trans. Graphics*, 2005, 24(3): 677–685. DOI: 10.1145/1073204.1073247.

[30] Müller M, Röder T. Motion templates for automatic classification and retrieval of motion capture data. In *Proc. the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Sept. 2006, pp.137–146. DOI: 10.2312/SCA/SCA06/137-146.

[31] Forbes K, Fiume E. An efficient search algorithm for motion data using weighted PCA. In *Proc. the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Jul. 2005, pp.67–76. DOI: 10.1145/1073368.1073377.

[32] Wu S Y, Xia S H, Wang Z Q, Li C P. Efficient motion data indexing and retrieval with local similarity measure of motion strings. *The Visual Computer*, 2009, 25(5/6/7): 499–508. DOI: 10.1007/s00371-009-0345-1.

[33] Gupta A, He J, Martinez J, Little J J, Woodham R J. Efficient video-based retrieval of human motion with flexible alignment. In *Proc. the 2016 IEEE Winter Conference on Applications of Computer Vision*, Mar. 2016. DOI: 10.1109/WACV.2016.7477588.

[34] Makhzani A, Shlens J, Jaitly N, Goodfellow I, Frey B. Adversarial autoencoders. arXiv: 1511.05644, 2015. https://arxiv.org/abs/1511.05644, Sept. 2023.

[35] Jiang Z, Huang Y, Peng J. Recent advances in content-based motion capture data retrieval. *International Journal of Electrical Engineering*, 2018, 25(2): 47–56. DOI: 10.6329/CIEE.201804-25(2).0002.

[36] Li Y, Fermuller C, Aloimonos Y, Ji H. Learning shift-invariant sparse representation of actions. In *Proc. the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2010, pp.2630–2637. DOI: 10.1109/CVPR.2010.5539977.

[37] Goodfellow I J, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A C, Bengio Y. Generative adversarial nets. In *Proc. the 27th International Conference on Neural Information Processing Systems*, Dec. 2014, pp.2672–2680. DOI: 10.1145/3422622.

[38] Manber U, Myers G. Suffix arrays: A new method for on-line string searches. *SIAM Journal on Computing*, 1993, 22(5): 935–948. DOI: 10.1137/0222058.

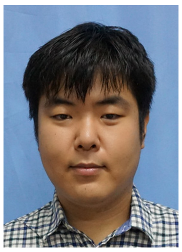
[39] Müller M, Röder T, Clausen M, Eberhardt B, Krüger B, Weber A. Documentation MoCap database HDM05. Computer Graphics Technical Reports CG-2007-2, Universität Bonn, 2007. https://cg.cs.uni-bonn.de/backend/v1/files/publications/cg-2007-2.pdf, July 2023.

[40] Yang W, Ouyang W L, Wang X L, Ren J, Li H S, Wang X G. 3D human pose estimation in the wild by adversarial learning. In *Proc. the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp.5255–5264. DOI: 10.1109/CVPR.2018.00551.

[41] Sun X, Xiao B, Wei F Y, Liang S, Wei Y C. Integral human pose regression. In *Proc. the 15th European Conference on Computer Vision*, Sept. 2018, pp.536–553. DOI: 10.1007/978-3-030-01231-1_33.

[42] Zhao L, Peng X, Tian Y, Kapadia M, Metaxas D N. Semantic graph convolutional networks for 3D human pose regression. In *Proc. the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2019, pp.3425–3435. DOI: 10.1109/CVPR.2019.00354.

[43] Li S C, Ke L, Pratama K, Tai Y W, Tang C K, Cheng K T. Cascaded deep monocular 3D human pose estimation with evolutionary training data. In *Proc. the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2020, pp.6173–6183. DOI: 10.1109/CVPR42600.2020.00621.



Zi-Fei Jiang received his M.S. degree in software engineering from Shandong University, Jinan, in 2014. Now he is a Ph.D. candidate at the School of Software, Shandong University, Jinan. His main research interests are in motion data processing and analysis.

Wei Li is currently doing her postdoctoral research at the School of Software, Shandong University, Jinan. She received her Ph.D. degree in computer science and technology from Shandong University, Jinan, in 2018. Her current research interests include computer vision, and image and video analysis and biometrics.

Yan Huang received her Ph.D. degree in computer science from the University of California, Irvine, in 2009, her M.S. degree in computer science from the University of California, Irvine, in 2003, and her B.S. degree in computer science from Peking University, Beijing, in 1997. Currently, she is an associate professor at the School of Software, Shandong University, Jinan. Her research interest mainly resides in digital geometry processing, and image and video analysis and understanding.

Yi-Long Yin received his Ph.D. degree in mechanical engineering from Jilin University, Changchun, in 2000. Currently, he is a professor at the School of Software, Shandong University, Jinan. His research interest mainly resides in artificial intelligence, machine learning, pattern recognition and data mining.

C.-C. Jay Kuo received his M.S. and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 1985 and 1987, respectively. He is currently the holder of William M. Hogue Professorship, a distinguished professor of electrical and computer engineering and computer science, and the director of the Ming Hsieh Department of Electrical and Computer Engineering at the University of Southern California, Los Angeles. Dr. Kuo is a fellow of the National Academy of Inventors (NAI), the American Association for the Advancement of Science (AAAS) and the International Society for Optical Engineers (SPIE).

Jing-Liang Peng received his Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, in 2006, his B.S. and M.S. degrees in computer science from Peking University, Beijing, in 1997 and 2000, respectively. Currently, he is a professor at the School of Information Science and Engineering, University of Jinan, Jinan. His research interest mainly resides in digital geometry processing, human-centric visual computing and medical imaging.