Vis Comput (2016) 32:243–256
DOI 10.1007/s00371-015-1178-8

ORIGINAL ARTICLE

Spatio-temporal segmentation for the similarity measurement of deforming meshes

Guoliang Luo[1] · **Frederic Cordier**[2] · **Hyewon Seo**[3]

Published online: 1 December 2015

Abstract Although there have been a large body of works on computing the similarity of static shapes, similarity judgments on deforming meshes are not studied well. In this study, we investigate a similarity measurement method for comparing two deforming meshes. Based on the degree of deformation, we first binarily label each triangle within each frame as either 'deformed' or 'rigid', then merge the 'deformed' triangles in both spatial and temporal domains for the segmentation. The segmentation results are encoded in a form of evolving graph, with an aim of obtaining a compact representation of the motion of the mesh. Finally, we formulate the similarity measurement as a sequence matching problem: after clustering similar graphs and assigning each of the graphs with the cluster labels, each deforming mesh is represented with a sequence of labels. Then, we apply a sequence alignment algorithm to compute the locally optimal alignment between the two label sequences, and to compute the similarity by normalizing the alignment score. The experimental results over several datasets show that the similarities of animation data can be captured correctly using our approach. This may be significant, as it solves a problem that cannot be handled by current approaches.

Keywords Spatio-temporal segmentation · Deforming mesh · Similarity · Sequence alignment

1 Introduction

With the proliferation of animation and motion capture techniques available today, animated meshes are becoming ubiquitous. Yet, the analysis and retrieval of such animation data remain as new research challenge. Such tasks require efficient representations of animation data, such as segmentation. Most existing works on deforming mesh segmentation compute spatial clustering according to geodesic and kinematic affinities of vertices or triangles. In such cases, it is clear that the spatial segmentation results may significantly be different depending on the deformation exhibited on the mesh. Ideally, they should represent well the motion exhibited on the mesh. However, when it comes to a long, complex motion composed of several basic motions, one may obtain overly segmented patches, which do not represent well each basic motion.

In this work, we propose a new spatio-temporal segmentation technique for deforming meshes towards the object of similarity measurement. First, after labeling each triangle using its strain and forming clusters in the spatio-temporal domain, the segmentation results are encoded in an evolving graph. Then, we further demonstrate the applicability of the evolving graph representation to the similarity measure of deforming meshes. After clustering similar graphs and assigning each of the graphs with the cluster labels, each deforming mesh is represented with a sequence of labels. Then, we apply a sequence alignment algorithm to compute the locally optimal alignment between the two cluster label sequences, from which we compute the similarity score. Our results show that simi-

Electronic supplementary material The online version of this article (doi:10.1007/s00371-015-1178-8) contains supplementary material, which is available to authorized users.

✉ Guoliang Luo
28guoliang.luo@gmail.com; guoliang.luo@jxnu.edu.cn

[1] Jiangxi Normal University (MIMLab), Nanchang, China

[2] LMIA, Université de Haute Alsace (LMIA, EA 3993), Mulhouse, France

[3] Université de Strasbourg (ICube, UMR 7357, CNRS), Strasbourg, France

larities of animation data can be captured correctly by our approach.

Note that this work is based on our previous workshop paper [26], with a large portion of the following extensions:

- To validate the choice of Smith–Waterman algorithm for the sequence alignment, we compare with Dynamic Time Wrapping [24] and Longest Common Substring algorithm [15], see Sect. 6.1. The results show that the performance of Smith–Waterman algorithm is more robust over noises. Moreover, the computation of the alignment is not only a critical stage for the similarity measurement scheme, but also allows to obtain pairwise temporal alignment. See Sect. 6.1.
- We further discuss the properties of the proposed similarity measurement method. See Sect. 5.3.
- We further validate our similarity measurement method with different datasets, including various representative animations (Sect. 6.2.1), animations driven by mocap data from 3dsMax motion library [1] (Sect. 6.2.3), and CVSSP-3D Data Sets [9,13,37] (Sect. 6.2.4).
- Furthermore, we compare the computed similarity results against human-based scores, which also shows the effectiveness of our method. See Sect. 6.2.2.

2 Related works

During the last two decades, a large amount of research has taken place on 3D static shape similarity measurement. One such research trend compares models using 3D shape descriptors, including shape histogram [4], spin image [18], and spherical harmonics [23]. The other main type is to represent models using graphs based on either segmentation [12,22], or skeleton extraction [40]. Researchers have also intensively studied the similarity measurement of motion capture data for indexing and retrieval [28,42]. However, the similarity measurement, as well as the segmentation, of deforming meshes remain as new challenges.

Segmentation of a mesh into meaningful parts has been a widely studied problem [5,6,33,34]. Existing methods aim to form meaningful segments or segment boundaries by optimizing pre-defined low-level criteria, such as convexity of segments and boundaries lying along concavities. However, methods segmenting a single shape in isolation do not perform well over all cases, because geometric criteria may not provide sufficient cues to identify all the semantically meaningful parts [10].

One branch of improvements has focused on data-driven approaches, which use knowledge learned from labeled dataset (supervised learning) or a set of shapes (unsupervised learning). The supervised approaches utilize manually pre-segmented training sets to learn a labeling function [21] or a boundary edge function [8], or to transfer labels [41]. The unsupervised approaches presented by Golovinskiy et al. [14], Sidi et al. [35] and Huang et al. [16] analyze a set of shapes belonging to a same class together, and co-segment them into consistent parts. Based on utilizing information from multiple meshes, these methods significantly outperform the single mesh segmentation methods. However, they require either a sufficiently large number of ground-truth data (that are manually segmented and labeled) or huge computation steps of optimization. With the increased dimensionality of the deforming mesh, obtaining the ground-truth dataset becomes very hard. Optimization framework for co-segmenting several deforming meshes may be intractable due to the huge computation steps. In addition, a co-segmentation scheme might impose a hard constraint on the input deforming meshes: all their motions must be similar.

Recently, several works have been developed for segmenting deforming mesh [20,25,27,32,43]. Mamou et al. [27] group vertices by applying K-means clustering on the multidimensional vector data composed of the transformation matrices of a vertex at each frame. Lee et al. [25] segment a mesh into near-rigid parts using a distance metric for each pair of triangles based on geodesic distance and deformation distance. Similar distance metric has been adopted by Kalafatlar et al. [20], who apply spectral clustering technique for segmenting an input mesh into multiple spatial parts according to the vertex trajectories. Wuhrer and Brunton [43] achieve the same object by determining the segmentation boundaries along the regions with highest deformation. With an aim of mesh compression, Sattler et al. [32] use clustered PCA to cluster the mesh vertices according to their similarities of trajectories such that those belonging to a same cluster have similar motion. All these methods compute the spatial segmentation of a given deforming mesh according to geodesic and kinematic affinities of vertices or triangles.

3 Overview

As input we have a deforming mesh (a sequence of meshes with fixed connectivity), which we resample along time so that all input data have the same frame rate. The main steps of our technique are as follows:

1. Compute the strain value for each triangle in each frame (Sect. 4.1).
2. Decompose the deforming mesh into spatio-temporal segments (Sect. 4.2).
3. Generate the graph representation of the spatio-temporal segmentation results (Sect. 4.3).
4. Cluster the collection of graphs from the two datasets, and assign to each graph a cluster label (Sect. 5.1).
5. Apply a sequence alignment algorithm to compute the locally optimal alignment between the two cluster label sequences (Sect. 5.2).

6. Measure the similarity by normalizing the alignment score (Sect. 6).

Finally, our experiment results and the evaluation on both synthetic and motion capture data are shown in Sect. 6.

4 Spatio-temporal segmentation

We now describe our spatio-temporal segmentation algorithm that makes use of the feature descriptor based on the deformation behavior of a triangle at each frame of the deforming mesh. Our goal is to obtain a compact representation of the segmentation results, which is done by adopting evolving graphs.

4.1 Strain computation

We begin by computing the degree of deformation of each triangle at each frame. The affine transformation of each triangle is computed, between every two consecutive frames. Let $\mathbf{v_i}$ and $\tilde{\mathbf{v}}_\mathbf{i}$ be the vertices of a triangle before and after the deformation, respectively. As shown in Fig. 1, a 3 by 3 affine matrix $\mathbf{F}$ and displacement vector $\mathbf{d}$ transforms $\mathbf{v_i}$ into $\tilde{\mathbf{v}}_\mathbf{i}$ in the following manner:

$$\mathbf{F} \cdot \mathbf{v_i} + \mathbf{d} = \tilde{\mathbf{v}}_\mathbf{i}, \quad \mathbf{i} = 1, 2, 3.$$

Similar to Sumner et al. [39], we add a fourth vertex in the direction of the normal vector of the triangle and subtract the first equation from the others to eliminate $\mathbf{d}$. Then, we obtain $\mathbf{F} = \tilde{\mathbf{V}} \cdot \mathbf{V}^{-1}$ where

$$\mathbf{V} = [\mathbf{v_2} - \mathbf{v_1}\ \mathbf{v_3} - \mathbf{v_1}\ \mathbf{v_4} - \mathbf{v_1}],$$

and

$$\tilde{\mathbf{V}} = [\tilde{\mathbf{v}}_\mathbf{2} - \tilde{\mathbf{v}}_\mathbf{1}\ \tilde{\mathbf{v}}_\mathbf{3} - \tilde{\mathbf{v}}_1\ \tilde{\mathbf{v}}_\mathbf{4} - \tilde{\mathbf{v}}_\mathbf{1}].$$

Based on the above definition, $\mathbf{F}$ has no translational component, e.g., rotation, scale or shear. Note that this representation specifies the deformation in per-triangle basis, so that it will be independent of the specific position and orientation of the mesh in world coordinates.

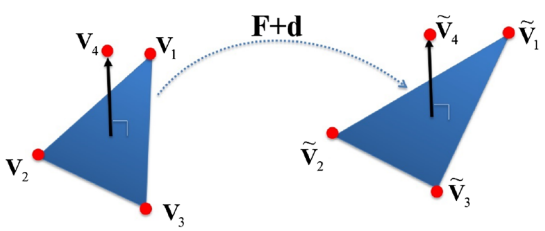


Fig. 1 Deformation between two triangles

In continuum mechanics, the matrix $\mathbf{F}$ is called deformation gradient tensor [11] as it explains the relationship between a material vector in the reference object (before deformation) and the deformed one, i.e., $\mathbf{F_{ij}} = \partial \mathbf{v_i} / \partial \tilde{\mathbf{v}}_\mathbf{j}$. Without loss of generality, we assume that the triangle is deformed first and then rotated. Then, we have $\mathbf{F} = \mathbf{RU}$, where $\mathbf{R}$ denotes the rotation matrix and $\mathbf{U}$ the deformation matrix. Since we want to differentiate triangles according to their degree of deformation, we eliminate the rotation component of $\mathbf{F}$ by computing the right Cauchy deformation tensor $\mathbf{C}$ as defined by:

$$\mathbf{C} = \mathbf{F^T F} = (\mathbf{RU})^\mathbf{T}(\mathbf{RU}) = \mathbf{U^T U}.$$

It shows that $\mathbf{C}$ is equal to the square of the right deformation matrix. We obtain principal deformation by the eigenanalysis of $\mathbf{C}$, and compute the deformation of a triangle as $(\lambda_1 + 1/\lambda_3)$, where λ_1, λ_2 and λ_3 are the principal components of $\mathbf{C}$. Since a rest-frame-based strain can be significantly different depending on the choices of the rest frame, in this work, we compute deformation gradients as well as triangle strains in each frame by referring to its previous frame.

4.2 Spatio-temporal segmentation

We start by labeling each triangle of each frame as either 'deformed' or 'rigid'. We chose binary labeling for the sake of simplicity although multi-way labeling could also work at higher computational cost. Given a mesh sequence $\mathcal{M}$ with M frames and N triangles, we represent each frame f^p, $p = 1, \ldots, M$, as a vector of strain values $\mathbf{s^p} = (s_1^p, \ldots, s_N^p)^\mathrm{T}$, which we obtain by the method described in Sect. 4.1. The strains are then normalized into [0 1] using a Gaussian Kernel Function (GKF):

$$\bar{s_i^p} = \exp(-0.5 \cdot s_i^p \cdot \sigma^{-2}), \quad i = 1, \ldots, N,$$

where σ is a width parameter that is derived from the average of strain values: $\sigma = 2(\sum_{i,p} s_i^p)/(N \times M)$. Finally, all triangles $t_i^p (i = 1, \ldots, N)$ in each frame f^p are binary labeled as 1 ('deformed') or 0 ('rigid'), by comparing their strain values to a threshold τ_s:

$$L_i^p = \begin{cases} 0, & \text{if } \bar{s_i^p} < \tau_s. \\ 1, & \text{otherwise.} \end{cases}$$

The threshold τ_s has been fixed as 0.5 in our experiments.

Once we have the per-frame and per-triangle labeling, we carry out our spatio-temporal segmentation by finding triangles with identical labels that are adjacent along space or time. Figure 2a illustrates this idea with a 'bending-cylinder' example. The red regions represent the 'deformed' regions with a set of 'deformed' triangles that are connected.

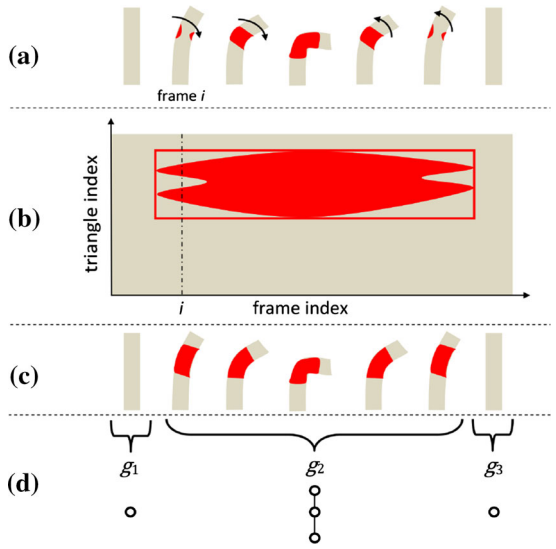


Fig. 2 An example of spatio-temporal segmentation of 'bending-cylinder'. **a** Binary labeling. **b** Spatio-temporal segmentation. **c** Spatio-temporal segment. **d** Evolving graph representation

Algorithm 1 Spatio-temporal segmentation

Inputs: L_i^p, $ST_i^p = 0$, $(i = 1, \ldots, N,\ p = 1, \ldots, M)$, **S=T=I=P=Ø**
{/*
L_i^p: binary label of the i-th triangle in the p-th frame.
S: 'deformed' triangles in a segment.
T: neighboring 'deformed' triangles of **S**.
I, **P**: the spatial and temporal indices of the triangles in **S**.
ST_i^p: spatio-temporal segmentation result.
*/}
while $\exists i, p, L_i^p = 1$ **do**
  S=L_i^p, **T**=**Neighbors**(**S**)-**S**
  while $\exists t \in \mathbf{T}, L(t) = 1$ **do**
    S=**S** $\cup \{t\}$, **T**=**Neighbors**(**S**)-**S**
  end while
  while $\exists i, p, t_i^p \in \mathbf{S}$ **do**
    $L_i^p = 0$,**S**=**S**-t_i^p
    I=[**I** t],**P**=[**P** p]
  end while
  $\forall i \in$ I,$p \in \mathbf{P}$, $ST_i^p = 1$
  S=T=I=P=Ø
end while
return **S**,**T**

Figure 2b shows the 2D representation of the labeling, with the horizontal axis denoting the time (frame index) and the vertical axis denoting the space (triangle index). The two small disconnected 'deformed' regions at frame i (the dashed vertical line) are merged into one region in the later frames. We consider these two regions as belonging to the same spatio-temporal segment because they result from the same deforming action. The procedure for our spatio-temporal segmentation is summarized as follows: we start with a 'deformed' triangle, and apply a region growing algorithm to merge with other 'deformed' triangles that are adjacent either along space or time. See the red area in Fig. 2b. Then, we compute the space interval and the time interval of the merged area, and take all triangles in that interval as a spatio-temporal segment (rectangle in Fig. 2b, also shown in red region in Fig. 2c). We continue the above procedure until all the 'deformed' triangles have been merged into a spatio-temporal segment. The complete spatio-temporal segmentation algorithm is shown in Algorithm 1, which runs in $O(M \cdot N)$time. Note that the function **Neighbors**(**S**) returns the 'deformed' triangles that are adjacent in either space or time of every triangle in **S**.

4.3 Evolving graph representation

We now describe our graph representation of the spatio-temporal segments. In the time interval of each spatio-temporal segment, we compute a sequence of key frames, where each key frame contains either the occurrence of a new spatio-temporal segment or the disappearance of a segment. Note that the first frame is always considered as a key frame. For each key frame, we represent its spatial segmentation with a graph, where a node represents a spatial segment and an edge connects two spatially adjacent segments, see Figs. 2d and 3b. A series of graphs we obtain for a deforming mesh is an evolving graph, i.e., a graph that evolves over time. The evolving graph of the 'bending-cylinder' is shown in Fig. 2d.

5 Pairwise sequence alignment

In this section, we present a method for measuring the similarity between two deforming meshes, which are represented with evolving graphs. We first cluster similar graphs of the two deforming meshes; graphs belonging to the same cluster are assigned with the same label. As a result, each deforming mesh is represented with a sequence of cluster labels. Then, we apply a sequence alignment algorithm to compute the locally optimal alignment between the two cluster label sequences. Finally, we measure the similarity between the two deforming meshes by normalizing the sequence alignment score.

5.1 Graph clustering

Let $\mathcal{M}^A$ and $\mathcal{M}^B$ be two deforming meshes. Let $G^A = \{g_1^A, g_2^A, \ldots, g_{n^A}^A\}$ and $G^B = \{g_1^B, g_2^B, \ldots, g_{n^B}^B\}$ be their corresponding evolving graphs which have been generated using the algorithm described in Sect. 4, where n^A and n^B are the number of graphs for $\mathcal{M}^A$ and $\mathcal{M}^B$, respectively.

The first step is to cluster similar graphs so that they can be labeled. Unfortunately, this step cannot be done using existing clustering methods, such as K-means clus-

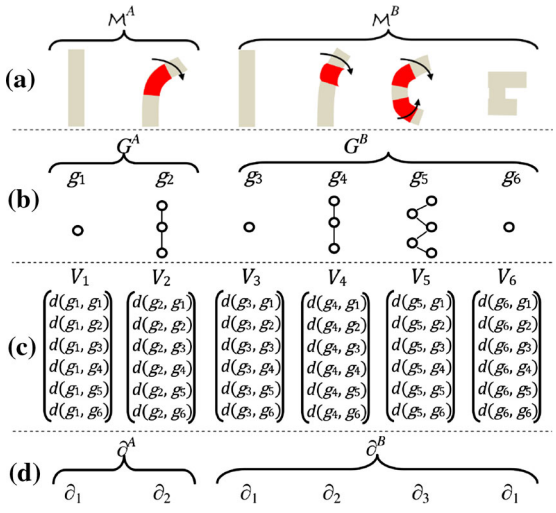


Fig. 3 Graph embedding and clustering. **a** The input deforming meshes $\mathcal{M}^A$ and $\mathcal{M}^B$. **b** The sequences of evolving graphs G^A and G^B. **c** The graph embedding. Each graph g_i is represented with a vector V_i, where $d(g_i, g_j)$ denotes the graph edit distance between graphs g_i and g_j. **d** The sequences of graph cluster labels ∂^A and ∂^B

tering; this is because the graphs have different numbers of vertices and edges, and these clustering methods work only for vectors of the same dimension. To overcome this problem, we adopted the graph embedding method proposed by Riesen et al. [30,31]. The purpose of this method is to compute a mapping between the graphs and a vector space. The graph embedding method works as follows: given a set of graphs $G = \{g_1, g_2, \ldots, g_n\}$ whose cardinality is n, the vector associated to a graph g_i is defined as $V_i = (d(g_i, g_1), d(g_i, g_2), \ldots, d(g_i, g_n))^{\mathrm{T}}$, where $d(g_i, g_j)$ is a graph dissimilarity metric between the graphs g_i and g_j.

In our case, the set G is the union of the sets of evolving graphs of $\mathcal{M}^A$ and $\mathcal{M}^B$, i.e., $G^A \cup G^B$, with cardinality $n = n^A + n^B$. The graph dissimilarity metric $d(g_i, g_j)$ is calculated using the graph edit distance. The graph edit distance between g_i and g_j is defined as the minimum number of graph edit operations to transform g_i to g_j; these operations include additions and deletions of nodes and edges. Neuhaus et al. [29] have proposed an efficient algorithm to compute the graph edit distance.

The overview of graph embedding algorithm is shown in Fig. 3. After the graph embedding, each graph is represented with a vector $V_i = (d(g_i, g_1), \ldots, d(g_i, g_n))^{\mathrm{T}}, i \in [1, \ldots, n]$.

Note that the size of the data produced by the graph embedding may be very large depending on the size of G. If G is composed of n graphs, the dimension of the output data is n^2 (n vectors V_i whose dimension is n). To reduce the dimension of the data, we apply the Principal Component Analysis (PCA) [19]. The PCA method uses orthogonal transformation to convert the set of vectors into a set of values of linearly uncorrelated variables called principal components. Redundant information is removed by representing the vectors V_i with the top r principal components. Hence, each graph g_i is represented with a vector V_i' whose dimension is r.

Finally, we apply K-means clustering method on the vectors V_i' to cluster all the graphs g_i into K clusters; K is a user-specified parameter which is chosen depending on the range of the deformation in $\mathcal{M}^A$ and $\mathcal{M}^B$. This value is set from 5 to 8 in our experiments. All the graphs g_i belonging to the kth cluster ($k \in [1, \ldots, K]$) are given the same cluster label ∂_k. Therefore, the deforming mesh $\mathcal{M}^A$, which is represented with a sequence of evolving graphs $G^A = \{g_1^A, g_2^A, \ldots, g_{n^A}^A\}$, is now represented with a sequence of cluster labels $\partial^A = \{\partial_1^A, \partial_2^A, \ldots, \partial_{n^A}^A\}$. A cluster label sequence $\partial^B = \{\partial_1^B, \partial_2^B, \ldots, \partial_{n^B}^B\}$ is also computed for G^B. Although G^A and G^B contain different graphs, the same cluster label may appear in ∂^A and ∂^B. This is because K-means clustering has been computed on the union set $G^A \cup G^B$. Therefore, two graphs of G^A and G^B may belong to the same cluster, and thus be assigned the same label.

In addition to the cluster labels ∂_k, we also compute the center of each cluster c_k, which is the mean vector of all the vectors V_i' whose corresponding graph g_i belongs to the cluster k. These cluster centers are required later to compute the sequence alignment (Sect. 5.2).

Figure 3d shows an example of graph clustering for two deforming cylinder meshes. The deformation of $\mathcal{M}^A$ is composed of the bending of the center part of the cylinder. The deformation of $\mathcal{M}^B$ includes the bending of the upper and lower parts of the cylinder with the bending of upper part starting first. After graph clustering, $\mathcal{M}^A$ and $\mathcal{M}^B$ are represented with the cluster label sequences ∂^A and ∂^B, respectively.

Note that the computation of sequence alignment increases exponentially along with the number of types of cluster labels. Using the graph clustering, the number of labeling types is much less than graph types, and therefore the alignment between graph labeling sequences saves computation significantly than between original graph sequences. Additionally, similar graphs are grouped into the same cluster and recognized as the same, which allows to remove noise due to segmentation, e.g., noises may generate extra segments.

5.2 Local sequence alignment

Now that we have computed the cluster label sequences ∂^A and ∂^B of the deforming meshes $\mathcal{M}^A$ and $\mathcal{M}^B$, the next step is to compute the alignment between the two sequences ∂^A and ∂^B by finding similar subsequences between them.

Local sequence alignment algorithm is commonly used in bioinformatics to identify similar regions among DNA sequences. The purpose of the alignment method is to locate

and align the most similar subsequences between two DNA sequences, which allow gaps within the alignment. One of the most known methods is the Smith–Waterman algorithm [36], which is adopted here. It finds the optimal local alignment based on dynamic programming approach. It requires inputs of an affinity matrix between sequence items and a gap penalty value.

To compute the alignment between the cluster label sequences ∂^A and ∂^B, we first compute the affinity matrix of the clusters. As explained in Sect. 5.1, each of these cluster labels ∂_k corresponds to a cluster whose center is c_k. The cluster distance matrix D is a matrix whose size is K by K; each of its elements $D_{k_1k_2}$ is the distance between the cluster k_1 and k_2; it is calculated as the Euclidean distance between the cluster centers c_{k_1} and c_{k_2}, that is, $D_{k_1k_2} = \sqrt{c_{k_1} - c_{k_2}}$. We then compute an affinity matrix ϑ whose dimension is K by K, and each of its elements $\vartheta_{k_1k_2}$ is the affinity value between the clusters k_1 and k_2 and is computed as follows:

$$\vartheta_{k_1k_2} = \bar{D} - D_{k_1k_2}, \quad \text{with} \quad k_1, k_2 \in [1, \ldots, K], \tag{1}$$

where $\bar{D}$ is the average value of all the elements of the distance D. Unlike the distance matrix, the affinity matrix may have negative and positive values; positive values indicate a high level of affinity between the clusters, and negative values indicate a low level of affinity.

Once the affinity matrix has been computed, we use the improved Smith–Waterman algorithm proposed by Barton et al. [7]. A Matlab implementation is available. This algorithm takes as input the two cluster label sequences ∂^A and ∂^B with their corresponding affinity matrix ϑ; it generates a set of pairs of matching cluster labels $Q : \{\partial_i^A \leftrightarrow \partial_{Q(i)}^B\}$, where $Q(i)$ indicates the label in ∂^B that is aligned to the i^{th} label in ∂^A, and T is the total number of non-matching cluster labels that are located among the matching ones. The set of matching pairs Q is computed such that the following matching score is maximized:

$$\delta_{AB} = \sum_{i=1}^{n_Q} \vartheta_{\partial_i^A \partial_{Q(i)}^B} - T \cdot \varepsilon, \tag{2}$$

where $\varepsilon = \beta \cdot \bar{D}$ is the penalty coefficient for the gaps occurring in the alignment; the coefficient β, which has been set to 1/6 in our experiments, can be adjusted depending on how large gaps we want to allow (smaller β value will allow larger gaps and vice versa) in the alignment.

The matching score δ_{AB} is simply the summation of the affinity values $\vartheta_{\partial_i^A \partial_{Q(i)}^B}$ of each of the matching pairs of cluster labels subtracted by $T \cdot \varepsilon$ which is the penalty score of the gaps. Figure 4 shows an alignment score between ∂^A and ∂^B

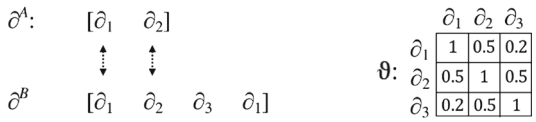


Fig. 4 The sequence alignment between ∂^A and ∂^B. Matching cluster labels are shown with *dashed lines*

without a gap. Here, the alignment score is $\delta_{AB} = \vartheta_{\partial_1^A \partial_1^B} + \vartheta_{\partial_2^A \partial_2^B} - 0 = 2$.

Although δ_{AB} in Eq. 2 can be negative, the algorithm that computes the matching score must return a non-negative result. This is because the empty set $Q = \emptyset$ is always taken into account when computing the most optimal alignment. In case of mismatching between ∂^A and ∂^B such that δ_{AB} in Eq. 2 is negative, the algorithm returns the empty set Q whose matching score is 0.

Time complexity Let $\mathcal{M}^A$ and $\mathcal{M}^B$ be two deforming meshes whose evolving graph sequences are G^A and G^B. Let n^A, n^B and n be the numbers of graphs of G^A, G^B and the total number of graphs (i.e., $n = n^A + n^B$), respectively. Our method involves computing the PCA whose time complexity is $O(n^3)$ [19], followed by the K-means clustering whose time complexity is $O(n^{rK+1} \log n)$ [17] with r being the number of principal components used for the PCA and K the number of clusters (see Sect. 5.1). Our algorithm also requires computing the sequence alignment whose time complexity is $O(n^A \cdot n^B)$ [36] and the graph embedding whose time complexity is $O(n^2 \cdot T_{GED})$, with T_{GED} being the polynomial time for computing the graph edit distance [29].

5.3 Similarity measurement

The main drawback of the alignment score defined in Eq. 2 is that its value is a function of the frame rate of the two sequences to be compared. Let A_1, B_1, A_2 and B_2 be four sequences, A_2 and B_2 carrying the same animated mesh as A_1 and B_1, respectively, but with higher frame rate. The alignment score $\delta_{A_1B_1}$ is higher than $\delta_{A_2B_2}$ although the sequences only differ in the frame rate. To alleviate this problem, we further normalize the alignment score as follows:

$$\rho_{AB} = \frac{\delta_{AB}}{\sqrt{\delta_{AA} \cdot \delta_{BB}}}. \tag{3}$$

This normalized alignment score ρ_{AB} holds the following properties:

P.1 *Non-negativity* $\rho_{AB} \geq 0$. As explained in Sect. 5.2, the matching score δ_{AB} is non-negative, and so is the value of ρ_{AB}. A value of ρ_{AB} equals to 0 implies that no alignment has been found between the two sequences.

P.2 *Symmetry* $\rho_{AB} = \rho_{BA}$. The alignment algorithm score in Eq. 2 does not depend on the order in which the sequences are aligned. That is, the same pairs of matching cluster labels are found whether ∂^A is aligned to ∂^B, or ∂^B to ∂^A. It follows that δ_{AB} is equal to δ_{BA}, and therefore ρ_{AB} is equal to ρ_{BA}.

P.3 *Boundness* $\rho_{AB} \leq \sqrt{\frac{\delta_{AA}}{\delta_{BB}}} \leq 1$, assuming $\delta_{AA} \leq \delta_{BB}$. According to Eq. 2, the matching score increases as the number of matching pairs gets larger. It follows that $\delta_{AB} \leq \delta_{AA}$. This is because the number of matching pairs for ∂^A being matched to itself is always larger than or equal to those matched to ∂^B. Similarly, we have $\delta_{AB} \leq \delta_{BB}$. It follows that:

$$\rho_{AB} = \frac{\delta_{AB}}{\sqrt{\delta_{AA} \cdot \delta_{BB}}} = \sqrt{\frac{\delta_{AB} \cdot \delta_{AB}}{\delta_{AA} \cdot \delta_{BB}}} \leq \sqrt{\frac{\delta_{AA} \cdot \delta_{BB}}{\delta_{AA} \cdot \delta_{BB}}} = 1.$$

If the two input sequences ∂^A and ∂^B are identical, (i.e., $\partial^A = \partial^B$), we have $\rho_{AB} = 1$.

More strictly, the upper bound of ρ_{AB} is $\sqrt{\frac{\delta_{AA}}{\delta_{BB}}}$, assuming $\delta_{AA} \leq \delta_{BB}$. As explained above, $\delta_{AB} \leq \delta_{AA}$, and it follows that:

$$\rho_{AB} = \frac{\delta_{AB}}{\sqrt{\delta_{AA} \cdot \delta_{BB}}} \leq \frac{\delta_{AA}}{\sqrt{\delta_{AA} \cdot \delta_{BB}}} = \sqrt{\frac{\delta_{AA}}{\delta_{BB}}}.$$

Based on the definition of alignment score in Eq. 2, if ∂^A is a subsequence of ∂^B, we have $\delta_{AA} = \delta_{AB}$. Hence, $\rho_{AB} = \sqrt{\frac{\delta_{AA}}{\delta_{BB}}}$.

P.4 *Subsequence* Assuming $\delta_{AA} \leq \delta_{BB}$, if $\rho_{AB} = \sqrt{\frac{\delta_{AA}}{\delta_{BB}}}$, we have $\partial^A \subseteq \partial^B$, i.e., ∂^A is a subsequence of ∂^B.

Given $\rho_{AB} = \sqrt{\frac{\delta_{AA}}{\delta_{BB}}}$, it follows that:

$$\rho_{AB} = \sqrt{\frac{\delta_{AA}}{\delta_{BB}}} = \sqrt{\frac{\delta_{AA} \cdot \delta_{AA}}{\delta_{AA} \cdot \delta_{BB}}} = \frac{\delta_{AA}}{\sqrt{\delta_{AA} \cdot \delta_{BB}}}.$$

By comparing to the definition of ρ_{AB} in Eq. 3, we have $\delta_{AA} = \delta_{AB}$. As can be seen in Eq. 2, the maximal alignment score to any other sequence is the alignment to a sequence itself. Therefore, the alignment score between ∂^A and ∂^B being equal to the alignment score between ∂^A and ∂^A indicates that ∂^A is aligned to a subsequence of ∂^B, i.e., $\partial^A \subseteq \partial^B$.

To sum up, the properties P.1 and P.3 show that $\rho_{AB} \in [0\ \sqrt{\frac{\delta_{AA}}{\delta_{BB}}}]$. A value close to 1 indicates that the two sequences ∂^A and ∂^B are similar, i.e., the two deforming meshes $\mathcal{M}^A$ and $\mathcal{M}^B$ perform similar motion. In addition, the properties P.3 and P.4 show that $\rho_{AB} = \sqrt{\frac{\delta_{AA}}{\delta_{BB}}}$ is the necessary and sufficient condition for $\partial^A \subseteq \partial^B$, i.e., ∂^A is a sebsequence of ∂^B.

6 Results

The deforming meshes used in our experiments include both synthetic animations and motion capture sequences, which are summarized in Table 1. "Michael", "Gorilla" and "Boy" are generated by rigging TOSCA high-resolution meshes [3] with a walking skeleton. The two other models, "Head" and "Face_1" are obtained by linear interpolation of eight key poses (anger, fury, grin, laugh, rage, sad, smile and surprise) from Sumner et al.'s work on Deformation Transfer [2]. "Camel" and "Horse_1" are also from them [2]. "Horse_2" is the same model as "Horse_1" except that the speed of motion and the starting pose have been modified. "Face_2" and "Face_3" have been obtained by applying the motion capture of two person's facial expressions to their scanned faces. They contain various expressions such as 'eyebrow-raise', 'anger', 'disgust', 'fear', 'happy', 'surprise', and 'sad'. Selected frames of several deforming meshes, as well as the segmentation results and the corresponding graph representations are shown in Fig. 5.

All our algorithms have been implemented using Matlab, and the results were computed on a Windows PC with 3.4 GHz Intel Core i7-2600 processor, 4 GB of RAM.

We first process each deforming mesh with our spatio-temporal segmentation method to generate the sequence of evolving graphs for each of them. The computation time devoted to this process is approximately 2 min for each sequence in a matlab implementation. Figure 5 shows several segmentation results we have obtained using our algorithm. In each figure, 'deformed' segments are shown in red and 'rigid' segments in blue. For the complete spatio-temporal segmentation, please refer to our supplemental video material.

6.1 Choice of sequence alignment technique

Sequence alignment techniques can be categorized into *local* and *global* types. Smith–Waterman algorithm used in our method is a local sequence alignment technique that aligns the *most similar* subsequences between two sequences, which allows gaps within the alignment (see Sect. 5.2). Similarly, Longest Common Substring (LCS) algorithm also aligns two sequences locally that the aligned subsequences are not required to occupy consecutive positions [15]. But differently, LCS computes the *exact* longest common subsequences (only identical items in two sequences can be aligned). On the other hand, global sequence alignment method, such as a well-known technique Dynamic Time Wraping (DTW) algorithm [24], aligns an entire sequence to

Table 1 The deforming meshes used in our experiments

Name	Nb. of triangles	Nb. of frames	Motion	Type
Camel	43, 778	48	Gallop	Synthetic
Horse_1	16, 858	48	Gallop	Synthetic
Horse_2	29, 984	80	Gallop	Synthetic
Michael	29, 999	54	Walk	Synthetic
Gorilla	29, 999	54	Walk	Synthetic
Boy	10, 146	54	Walk	Synthetic
Head	31, 620	80	Facial expressions	Synthetic
Face_1	57, 836	80	Facial expressions	Synthetic
Face_2	1171	1473	Facial expressions	Motion capture
Face_3	1272	1064	Facial expressions	Motion capture

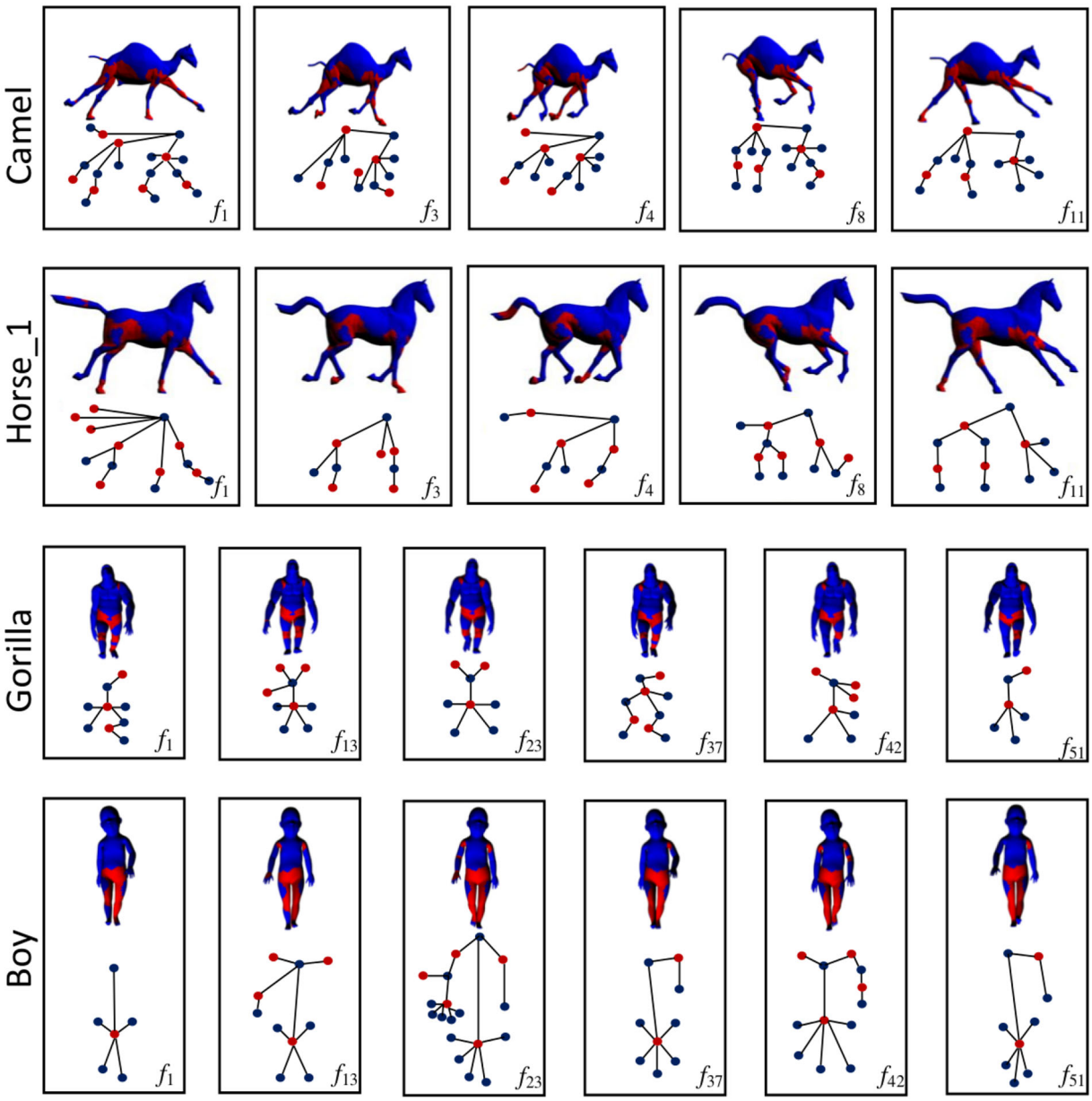


Fig. 5 The spatio-temporal segmentation and the graph representation of "Camel", "Horse_1", "Gorilla" and "Boy"

the other by minimizing total aligned element distance without a gap. One variant to DTW is a constraint on the alignment rule such that the first (and the last) elements of the two sequences are forced to be aligned with each other. We name this restricted DTW as mDTW. In this section, we validate our choice of Smith–Waterman algorithm rather than LCS, DTW or mDTW by comparing their performance between sequence pairs "Gorilla-Michael" and "Horse_1-Horse_2". The features of the four sequence alignment methods are summarized in Table 2.

To apply DTW and mDTW for optimally aligning two deforming meshes, one needs to compute a frame distance matrix. To do this, we first represent each deforming mesh into a key frame sequence, where each key frame represents a subsequence of frames with the same spatial segmentation (see Sect. 4.3). That is, if a key frame is representing a subsequence with n' frames, the key frame will be repeated for n' times. We then compute the frame distance as a graph edit distance since each key frame is associated with a graph representation.

Figure 6 shows the distance matrices between several deforming meshes, with color varying from blue to red indicating distance values from low to high. Figure 6 also shows the comparisons of sequence alignment using methods of Smith–Waterman, DTW and mDTW. An alignment result can be seen as an alignment path in the similarity matrix. We summarize the comparisons as follows:

Smith–Waterman vs. DTW (1) In Fig. 6a, the alignment path computed with the DTW method only aligns about a half of "Gorilla" sequence to "Michael" sequence due to the globally minimized alignment distance. In comparison, the alignment path computed with the Smith–Waterman method lies along the diagonal of the matrix, which is more preferable because "Gorilla" and "Michael" contain the same "Gallop" motion. (2) Moreover, in Fig. 6a, different from the alignment path of DTW, the frames f^{17} and f^{54} of "Michael" do not have aligned frames in "Gorilla" in the path computed with the Smith–Waterman algorithm. The reason is that Smith–Waterman is a local sequence alignment method, unlike the DTW method which is a global alignment method. That is, DTW returns continuous path but Swith–Waterman allows gap in the path. This feature is particularly interesting because a dissimilar frame occurred by noise would be skipped in the alignment path. (3) Lastly, in Fig. 6b, DTW method aligns the "Horse_1" to nearly the entire sequence of "Horse_2". In comparison, Smith–Waterman method aligns "Horse_2" to 7 frames of "Horse_1", which correctly reflects the fact that "Horse_2" contains 1 cycle of "Gallop" motion while "Horse_1" contains 4 cycles.

Table 2 Summary of the features of four sequence alignment methods

Method	Type	Features
Smith–Waterman	Local	Similar items can be aligned
LCS	Local	Only identical items can be aligned
DTW	Global	One sequence is entirely aligned to the other
mDTW	Global	Two sequences are entirely to each other

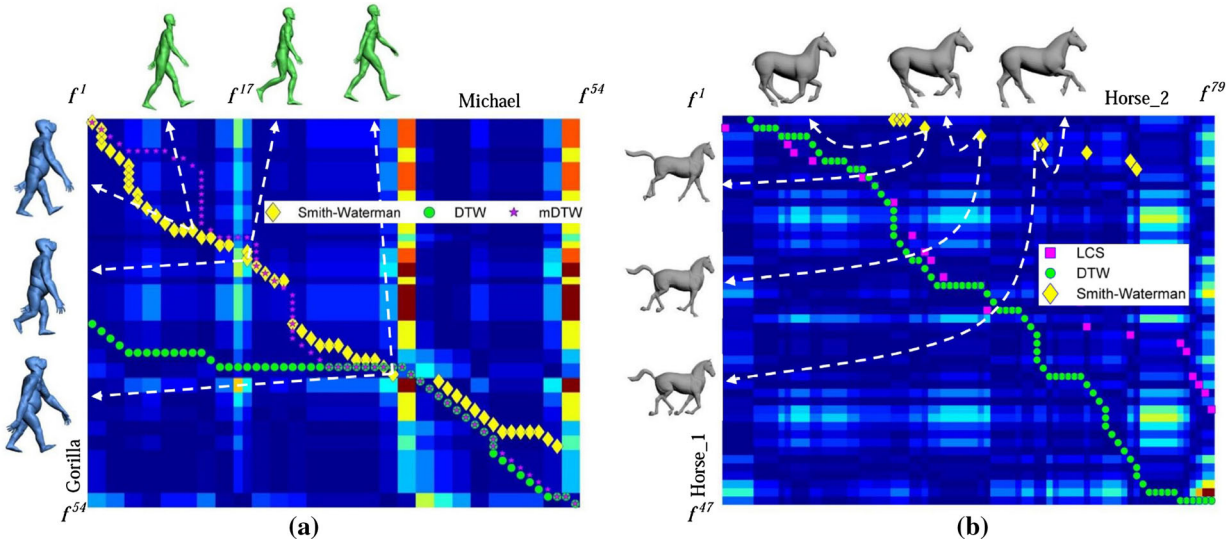


Fig. 6 Comparisons of the sequence alignment methods among Smith–Waterman, DTW, mDTW and LCS, using **a** 'Gorilla' and 'Michael', and **b** 'Horse1' and 'Horse2'

Smith–Waterman vs. mDTW Although the alignment path obtained with the mDTW method is similar to the path based on Smith–Waterman method, mDTW is not applicable to "Horse_1-Horse_2" because these two deforming meshes have different starting poses, which makes it not reasonable to force the first/last frames being aligned. Moreover, similar to DTW, mDTW does not allow gaps in the alignment, while Smith–Waterman method does. *Smith–Waterman vs. LCS*. As can be seen in Fig. 6b, similar to Smith–Waterman, LCS also computes the alignment path with gaps. However, because LCS only accepts *exact* matching of item pairs, those similar pairs recognized by Smith–Waterman are not detected by LCS. Thus, given the fact that a graph can always be slightly varied due to noise occured in segmentation stage, the frames with similar movements may not be detected by LCS.

Based on the above comparisons, we observe that Smith–Waterman algorithm has the advantage of allowing gaps and is capable of revealing the repetition of motions. For these reasons, we have chosen Smith–Waterman algorithm to align sequences, based on which we have developed our similarity measurement method.

6.2 Similarity measurement

6.2.1 Similarity of deforming meshes

Figure 7 shows the similarity scores we obtained for the example models. As expected, deforming meshes with similar motion show high similarity scores. Note that "Horse_2" has different motion speed and starting pose compared to "Camel" or "Horse_1", but the similarities among these three are higher than the others because they all show galloping motions. On the other hand, although the shape of "Face_1" is similar to those of "Face_2" and "Face_3", similarities of "Face_1" to these two are low because they exhibit different facial expressions. Additionally, the average similarity between "Gallop"-"Walk" motions is higher than either "Gallop"-"Facial expression" or "Walk"-"Facial expression", which complies with human judgment. Lastly, as can be seen in Fig. 5, although the number of nodes for "Gallop" animations is similar to those of "Walk" animations, our similarity metric can still distinguish the two motions.

6.2.2 Evaluation of motion similarities

Before we evaluate our similarity measurement method, we first study how humans perceive the motion similarity between deforming meshes. We invite 11 participants who are not aware of our segmentation method and show them with the 10 animated meshes used in our experiments. Based on subjective observations, each participant gives a score on motion similarity (with a larger number between [0 100] indicating higher similarity) between each pair of the deforming meshes. Therefore, we obtain $11 \times C_{10,2} = 495$ pairwise motion similarities of deforming meshes based on human perception.

Having created the human-based ground-truth similarity between deforming meshes, we evaluate our similarity results by applying Pearson's correlation [38]. A Pearson's correlation ranges from -1 to $+1$, with $+/-$ indicating positive/negative relationship between two variables, and the values reflecting the degree of linear relationships. To compute Pearson's correlation between ground truth and our results, we save the 495 human-rated similarities into a vector $\mathbf{V_{gt}}$, and create another vector $\mathbf{V_{ours}}$ where each value $\mathbf{V_{ours}}(i), i = 1, \ldots, 495$ is the corresponding similarity value of $\mathbf{V_{gt}}(i)$ but computed using our method. That is, $\mathbf{V_{ours}}$ actually contains 11 times repetition of the similarity results shown in Fig. 7.

Figure 8 shows the scatterplot between $\mathbf{V_{gt}}$ and $\mathbf{V_{ours}}$, and the linear regression between the two vectors. Among the human scores, there are 4 participants out of 11 who give scores of the similarities between "Horse_2" and "Walk" deforming meshes not more than 10 %, shown in the dashed circle. On the other hand, there are 4 participants who give scores of the similarities between "Camel" and "Walk" deforming meshes not less than 50 %, shown in the dotted circle. For "Face_2" and "Face_3", (1) all the participants give the scores to "Gallop" and "Walk" animations less than 10 %, (2) with "Face_1" and "Head", although they perform different facial expressions, 3 participants give scores up to 40 % but the other 8 participants give scores less than 20 %. Although we have instructed the participants to rate scores of motions, several of them still tend to give slightly higher scores to similar shapes even with different motions. Finally,

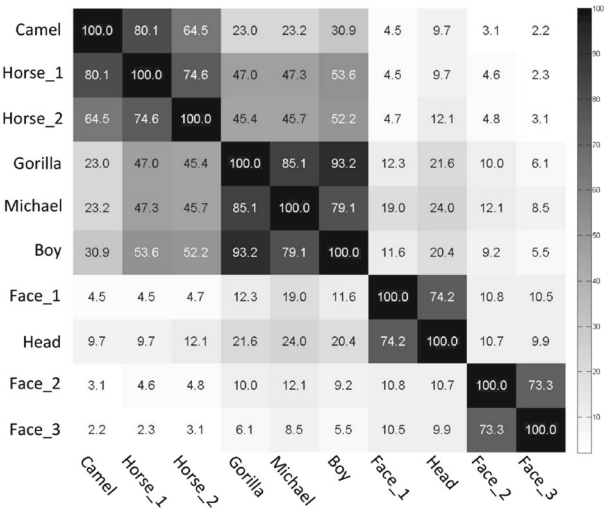


Fig. 7 Matrix of similarities among deforming meshes. The values are shown in percentage (%)

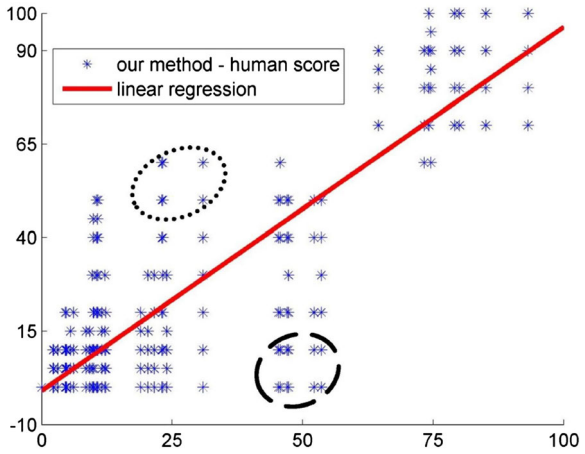


Fig. 8 Scatterplot between the similarities of deforming meshes computed using our method (*horizontal*) and the human scores of similarities (*vertical*). The *red line* is the linear regression of the 2D point distribution

we obtain 0.9008 as the Pearson's correlation between $\mathbf{V_{gt}}$ and $\mathbf{V_{ours}}$. The correlation value indicates that our similarity measurement method has a high degree of correlation with human perception on motion similarity.

6.2.3 Granularity of the motion similarities

We further proceed to evaluate the performance of our similarity measurement method for similar motions. To this end, we use 6 'biped' animations (including 'Jog1' and 'Jog2', 'Jump1' and 'Jump2', 'Walk1' and 'Walk2', with only the difference of movement directions, i.e., turn left/right, between each pair) from 3dsMax motion library [1], and attach them to 3 meshes, i.e., "Michael", "Gorilla" and "Boy", which results in 18 deforming meshes. Please refer to our supplemental video for the animations.

By applying our similarity measurement method, we obtain a motion similarity matrix among these deforming meshes, see Fig. 9a. We describe this result and its evaluation as follows:

- In Fig. 9a, we can represent each deforming mesh as each row of the similarity matrix. Then, by applying K-means clustering, we successfully classify the 18 deforming meshes into 3 clusters of different motion types, i.e., 'Jog', 'Jump' and 'Walk'.
- Based on the above motion classification, we convert the motion similarity matrix of deforming meshes in Fig. 9a to motion similarity ranking matrix in Fig. 9b, where each row shows the rankings of all the motions to a motion based on the average motion similarities in Fig. 9a. In this motion similarity ranking matrix, we use '1/2/3' to indicate the rankings of the similarity to all the motions, where '1'/'3' is the highest/lowest ranking (note that the motion similarity ranking matrix is not a symmetric matrix, because an animation A's most similar animation is B does not mean that B's most similar animation is A).
- To validate our motion similarity rankings, we invite 11 participants to give the rankings for the 3 motions by observing the 18 animations. In Fig. 9c, the number within each parenthesis are the number of participants who give the ranking number before the corresponding parenthesis. Note that 1 participant out of the 11 considered 'Jog' and 'Jump' being equally different to 'Jump' and gave ranking '3' for both, see the second row in Fig. 9c. Apart from this, our computed ranking results are met with most of the human rankings of the three motions.

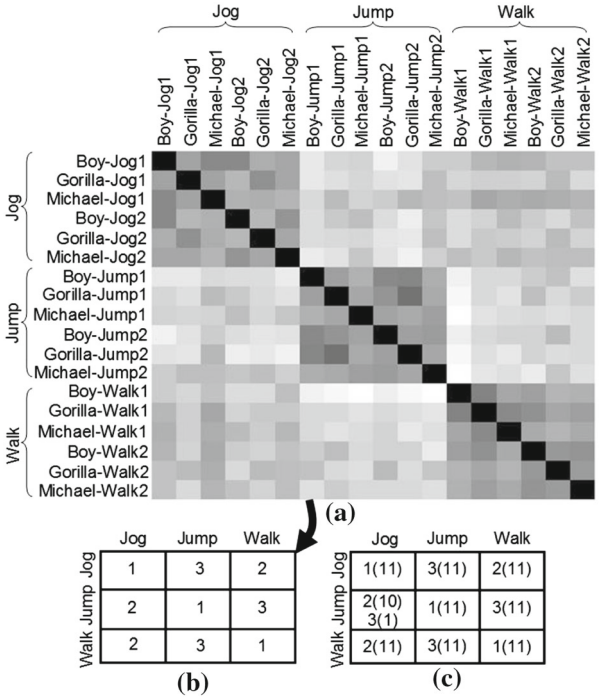


Fig. 9 Similarities among three similar motions, 'Jog', 'Jump' and 'Walk'. **a** Similarity matrix among 18 deforming meshes. **b** *Each row* shows the rankings of all the motions to a motion based on the average motion similarities in **a**. **c** Human-rated motion similarity rankings for each motion, where the numbers within each parenthesis is the number of participants who give the ranking before the corresponding *parenthesis*

Therefore, based on the above experiments on 18 deforming meshes with similar motions, i.e., 'Jog', 'Jump' and 'Walk', our similarity measurement method can successfully distinguish these three similar motion types. Moreover, the similarities obtained using our method reflect human perceptions on motion similarity because our motion ranking results comply well with human rankings among the three similar motions.

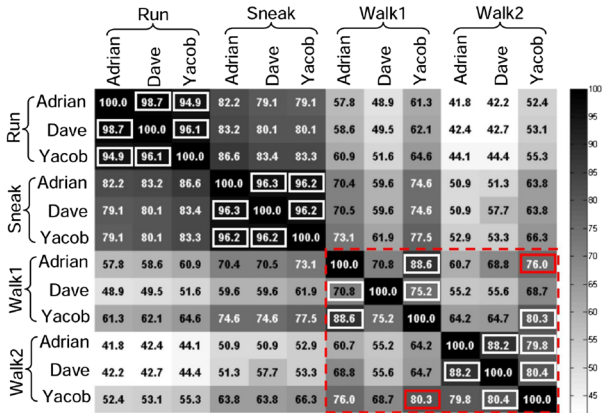


Fig. 10 Similarities among four similar motions, 'Run', 'Sneak', 'Walk2'('Walkcowboy') and 'Walk2'('Walkelderly'), by 3 different people (Adrian, Dave and Yacob). In *each row*, the *two box* are the two animations with highest similarities, except 100 % with itself. Note that *white boxes* indicate the same actions performed by different people, and *red boxes* indicate different actions. The *red dashed box* contains the similarities among the animations performing two walking actions, 'Walk1' and 'Walk2'

6.2.4 Experiments with CVSSP-3D Data Sets

In this section, we further evaluate the proposed similarity measurement method using CVSSP-3D Data Sets [9,13,37]. This dataset contains a set of synthetic deforming meshes of 3D human actions by different people. In specific, the creators copy the actions by different people to the same 3D human mesh with 1290 vertices and 2108 triangles. In our experiment, we choose 4 representative actions, including 'Run', 'Sneak', ('Walk1') 'Walkcowboy" and ('Walk2') 'Walkelderly', by three different people (Adrian, Dave and Yacob).

The similarities among the chosen animations measured using our method are shown in Fig. 10. For each animation in each row, we use two boxes to mark the two animations with highest similarities. For each row, we expect the animations performing the same actions having the highest similarities, and mark them with white boxes, otherwise, red boxes. As can be seen, only 2 out of 24 ($2/24 = 0.083$) do not meet the expectations. On the other hand, after all, 'Walk1' and 'Walk2' are very similar walking actions. This is correctly reflected in the dashed red box, that the average similarities between 'Walk1' and 'Walk2' are higher than with 'Run' or 'Sneak'.

6.3 Discussions

Graph edit distance (GED) In our method, we compute GED by measuring the graph structure difference between two graphs. One natural extension of the graph distance computation is to take into account the node attributes such as segment surface area, and edge attributes such as distance between segment centers. However, such node and edge attributes could vary due to shape differences, which lead to inconsistent motion similarity of deforming meshes using our method. On the contrary, our goal is to develop a similarity measurement method independent of shape differences. To this end, segment surface area and segment distance are not considered for computing graph distance.

Limitations One limitation of the proposed similarity measurement method is its expensive computational cost, mainly due to the computation of GED. With all evolving graphs (1135 graphs) of the ten animations that we have used in our results, it takes about 2 h to compute the complete clusters. However, once the clusters have been computed for a dataset with sufficient variety, computing the labels for a new deforming mesh will be a matter of computing the graph embedding of each graph in its evolving graph, and clustering each of the graphs to the closest cluster center c_k. It should be reminded that our scheme allows to obtain not only the similarity scores, but also pairwise temporal alignments with gaps. Another limitation is that we assume a deforming mesh can be segmented into either 'deformed' and 'rigid' parts, at the graph representation stage. For this reason, our segmentation algorithm is not applicable to highly dynamic animations such as the surface simulation of flowing water, which will return one single segment.

7 Conclusion

We have presented a new method for spatio-temporal segmentation of deforming mesh. Our method can treat not only skeleton-driven animations, but also 3D skin models, such as the facial data in our experiments. Moreover, based on the evolving graph representation of the segmentation results, which encodes both spatial and temporal deformation behaviors of the mesh, we further developed a similarity measurement method by adopting sequence comparison. The results show that our similarity measurement method successfully compares deforming meshes according to their motions.

One obvious potential of our segmentation-based similarity measurement method is its extension towards a shape query application, which will enable querying a database of deforming meshes. Additional efforts on efficient indexing and speedup computations will be required. Since our method has shown the effectiveness for both very different motions and similar motions in our experiments, we are confident that our method can be used for retrieval operations over large datasets in the future.

Acknowledgments We first would like to thank Professor Adrian Hilton and Dr. Peng Huang from the University of Surrey, for providing us a set of synthetic deforming meshes, which have been used for eval-

uation. We are grateful to MIRALab at the University of Geneva, who has provided us with the scanned face data. This has allowed us to make the highly resoluted facial animations "Face_2" and "Face_3" by transferring our mocap facial expressions to the scanned meshes. We would like to thank Frédéric Larue and Olivier Génevaux for providing us with the facial motion capture data, and Arash Habibi for providing us the "Horse_2" data. We also would like to thank Vasyl Mykhalchuk, who has developed a tool for visualizing the evolving graphs. This work has been jointly supported by the French national project SHARED (Shape Analysis and Registration of People Using Dynamic Data, No. 10-CHEX-014-01), and the Science and Technology Research Project of Jiangxi Provincial Department of Education (GJJ14246).

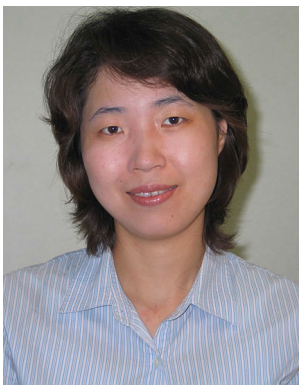
References

1. Autodesk Inc.: 3D Studio Max. http://www.autodesk.com/products/3ds-max/overview. Accessed 23 Nov 2015
2. Sumner, R., Popovic, J.: Mesh data from deformation transfer for triangle meshes. Computer Graphics Group at MIT. http://people.csail.mit.edu/sumner/research/deftransfer/data.html. Accessed 23 Nov 2015
3. Bronstein, A.M., Bronstein, M.M., Castellani, U., Falcidieno, B., Fusiello, A., Godil, A., Guibas, L.J., Kokkinos, I., Lian, Z., Ovsjanikov, M., Patane, G., Spagnuolo, M., Toldo, R.: Project: tools for non-rigid shape comparison and analysis. http://tosca.cs.technion.ac.il/. Accessed 23 Nov 2015
4. Ankerst, M., Kastenmüller, G., Kriegel, H.-P., Seidl, T.: 3d shape histograms for similarity search and classification in spatial databases. In: Advances in Spatial Databases, pp. 207–226. Springer (1999)
5. Attene, M., Falcidieno, B., Spagnuolo, M.: Hierarchical mesh segmentation based on fitting primitives. Vis. Comput. **22**(3), 181–193 (2006)
6. Attene, M., Katz, S., Mortara, M., Patané, G., Spagnuolo, M., Tal, A.: Mesh segmentation—a comparative study. In: IEEE International Conference on Shape Modeling and Applications, 2006. SMI (2006)
7. Barton, G.J.: An efficient algorithm to locate all locally optimal alignments between two sequences allowing for gaps. Comput. Appl. Biosci.: CABIOS **9**(6), 729–734 (1993)
8. Benhabiles, H., Lavoué, G., Vandeborre, J.-P., Daoudi, M.: Learning boundary edges for 3d-mesh segmentation. In: Computer Graphics Forum, vol. 30, pp. 2170–2182. Wiley Online Library (2011)
9. Budd, C., Huang, P., Klaudiny, M., Hilton, A.: Global non-rigid alignment of surface sequences. Int. J. Comput. Vis. **102**(1–3), 256–270 (2013)
10. Chen, X., Golovinskiy, A., Funkhouser, T.: A benchmark for 3d mesh segmentation. In: ACM Transactions on Graphics (TOG), vol. 28, p. 73. ACM (2009)
11. Crandall, S.H., Lardner, T.J., Archer, R.R., Cook, N.H., Dahl, N.C.: An introduction to the mechanics of solids
12. Cyr, C.M., Kimia, B.B.: 3d object recognition using shape similarity-based aspect graph. In: Eighth IEEE International Conference on Computer Vision, 2001. ICCV 2001. Proceedings, vol. 1, pp. 254–261. IEEE (2001)
13. Gkalelis, N., Kim, H., Hilton, A., Nikolaidis, N., Pitas, I.: The i3dpost multi-view and 3d human action/interaction database. In: Conference for Visual Media Production, 2009. CVMP'09, pp. 159–168. IEEE (2009)
14. Golovinskiy, A., Funkhouser, T.: Consistent segmentation of 3d models. Comput. Gr. **33**(3), 262–269 (2009)
15. Gusfield, D.: Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology. Cambridge University Press, Cambridge (1997)
16. Huang, Q., Koltun, V., Guibas, L.: Joint shape segmentation with linear programming. In: ACM Transactions on Graphics (TOG), vol. 30, p. 125. ACM (2011)
17. Inaba, M., Katoh, N., Imai, H.: Applications of weighted voronoi diagrams and randomization to variance-based k-clustering. In: Proceedings of the Tenth Annual Symposium on Computational Geometry, pp. 332–339. ACM (1994)
18. Johnson, A.E.: Spin-images: a representation for 3-D surface matching. Ph.D. thesis, Citeseer (1997)
19. Jolliffe, I.: Principal Component Analysis. Wiley Online Library, New York (2005)
20. Kalafatlar, E., Yemez, Y.: 3d articulated shape segmentation using motion information. In: 2010 20th International Conference on Pattern Recognition (ICPR), pp. 3595–3598. IEEE (2010)
21. Kalogerakis, E., Hertzmann, A., Singh, K.: Learning 3d mesh segmentation and labeling. ACM Trans. Gr. (TOG) **29**(4), 102 (2010)
22. Katz, S., Leifman, G., Tal, A.: Mesh segmentation using feature point and core extraction. Vis. Comput. **21**(8–10), 649–658 (2005)
23. Kazhdan, M., Funkhouser, T., Rusinkiewicz, S.: Rotation invariant spherical harmonic representation of 3d shape descriptors. In: Symposium on Geometry Processing, vol. 6 (2003)
24. Kruskall, J.B., Liberman, M.: The symmetric time warping algorithm: from continuous to discrete. In: The Theory and Practice of Sequence Comparison, in time Warps, String Edits and Macromolecules (1983)
25. Lee, T.-Y., Wang, Y.-S., Chen, T.-G.: Segmenting a deforming mesh into near-rigid components. Vis. Comput. **22**(9–11), 729–739 (2006)
26. Luo, G., Cordier, F., Seo, H.: Similarity of deforming meshes based on spatio-temporal segmentation. In: 3D Object Retrieval Workshop, 3DOR (2014)
27. Mamou, K., Zaharia, T., Prêteux, F.: A skinning approach for dynamic 3d mesh compression. Comput. Anim. Virtual Worlds **17**(3–4), 337–346 (2006)
28. Müller, M., Röder, T., Clausen, M.: Efficient content-based retrieval of motion capture data. In: ACM Transactions on Graphics (TOG), vol. 24, pp. 677–685. ACM (2005)
29. Neuhaus, M., Riesen, K., Bunke, H.: Fast suboptimal algorithms for the computation of graph edit distance. In: Structural, Syntactic, and Statistical Pattern Recognition, pp. 163–172. Springer (2006)
30. Riesen, K., Bunke, H.: Graph classification based on vector space embedding. Int. J. Pattern Recognit. Artif. Intell. **23**(06), 1053–1081 (2009)
31. Riesen, K., Bunke, H.: Reducing the dimensionality of dissimilarity space embedding graph kernels. Eng. Appl. Artif. Intell. **22**(1), 48–56 (2009)
32. Sattler, M., Sarlette, R., Klein, R.: Simple and efficient compression of animation sequences. In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 209–217. ACM (2005)
33. Shamir, A.: A survey on mesh segmentation techniques. In: Computer Graphics Forum, vol. 27, pp. 1539–1556. Wiley Online Library (2008)
34. Shapira, L., Shamir, A., Cohen-Or, D.: Consistent mesh partitioning and skeletonisation using the shape diameter function. Vis. Comput. **24**(4), 249–259 (2008)
35. Sidi, O., van Kaick, O., Kleiman, Y., Zhang, H., Cohen-Or, D.: Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. In: ACM Transactions on Graphics (TOG), vol. 30, p. 126. ACM (2011)
36. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J. Mol. Biol. **147**(1), 195–197 (1981)
37. Starck, J., Hilton, A.: Surface capture for performance-based animation. Comput. Gr. Appl. IEEE **27**(3), 21–31 (2007)
38. Stigler, S.M.: Francis Galton's account of the invention of correlation. Stat. Sci. **4**(2):73–79 (1989)

39. Sumner, R.W., Zwicker, M., Gotsman, C., Popović, J.: Mesh-based inverse kinematics. In: ACM Transactions on Graphics (TOG), vol. 24, pp. 488–495. ACM (2005)
40. Sundar, H., Silver, D., Gagvani, N., Dickinson, S.: Skeleton based shape matching and retrieval. In: Shape Modeling International, 2003, pp. 130–139. IEEE (2003)
41. Wang, Y., Gong, M., Wang, T., Cohen-Or, D., Zhang, H., Chen, B.: Projective analysis for 3d shape segmentation. ACM Trans. Gr. (TOG) **32**(6), 192 (2013)
42. Wu, S., Xia, S., Wang, Z., Li, C.: Efficient motion data indexing and retrieval with local similarity measure of motion strings. Vis. Comput. **25**(5–7), 499–508 (2009)
43. Wuhrer, S., Brunton, A.: Segmenting animated objects into near-rigid components. Vis. Comput. **26**(2), 147–155 (2010)

Guoliang Luo is currently an assistant professor at the Jiangxi Normal University in China. He earned his Ph.D. degree in computer science at the University of Strasbourg, France. He obtained his master's degree in computer science from the Uppsala University, Sweden. His current research interests include computer graphics, segmentation of mesh sequences, and compression of 3D animations.

Frederic Cordier is an associate professor at the Universit? de Haute Alsace, France. His research interests include 3D modeling and texturing, human–computer interaction, and physics-based simulation. He holds a Ph.D. in computer science from the University of Geneva, Switzerland.



Hyewon Seo is a CNRS researcher at the University of Strasbourg, France. Her research interests include imaging, visual simulation, human–computer interaction, and virtual reality. She has graduate degrees in computer science from the University of Geneva in Switzerland and the KAIST in South Korea.