Eurographics Workshop on 3D Object Retrieval (2014)
B. Bustos, H. Tabia, J.-P. Vandeborre, and R. Veltkamp (Editors)

# Similarity of Deforming Meshes Based on Spatio-temporal Segmentation

G. Luo[1], F. Cordier[2] and H. Seo[1]

[1]Université de Strasbourg (ICube, UMR 7357, CNRS), Strasbourg, France
[2]LMIA, Université de Haute Alsace (LMIA, EA 3993), Mulhouse, France

**Abstract**

*In this study, we investigate a similarity metric for comparing two deforming meshes. While there have been a large body of works on computing the similarity of static shapes, similarity judgments on deforming meshes are not studied well. Our algorithm uses the degree of deformation to binarily label each triangle in deforming mesh in the spatio-temporal domain, which serves as basis for the spatio-temporal 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 computation 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 cluster label sequences, and to compute the similarity metric by normalizing the alignment score. We show that similarities of animation data can be captured correctly by our approach. This may be significant, as it solves a problem that cannot be handled by current approaches.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

## 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, with an aim of developing a new representation that encodes well the given motion. After labeling each triangle by using its strain and forming clusters in the spatio-temporal domain, the segmentation results are encoded in an evolving graph. 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 similarities of animation data can be captured correctly by our approach.

## 2. Related works

Segmentation of a mesh into meaningful parts has been a widely studied problem [AKM*06, GF08, LZ04, Sha08]. Existing methods aim to form meaningful segments or segment boundaries by optimizing pre-defined low level criteria, such as convexity of segments, boundaries lying along concavities, etc. 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 [CGF09].



DOI: 10.2312/3dor.20141053

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 [KHS10] or a boundary edge function [BLVD11], or to transfer labels [WGW*13]. The unsupervised approaches presented by Golovinskiy et al. [GF09], Sidi et al. [SvKK*11] and Huang et al. [HKG11] 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. Mamou et al. [MZP06] 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. [LWC06] segment a mesh into near-rigid parts by 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. [KY10], who apply spectral clustering technique for segmenting an input mesh into multiple spatial parts according to the vertex trajectories. In their mesh compression scheme, Sattler et al. [SSK05] 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. (Section 4.1)
2. Decompose the deforming mesh into spatio-temporal segments. (Section 4.2)
3. Generate the graph representation of the spatio- temporal segmentation results. (Section 4.3)
4. Cluster the collection of graphs from the two dataset, and assign to each graph a cluster label. (Section 5.1)
5. Apply a sequence alignment algorithm to compute the locally optimal alignment between the two cluster label sequences. (Section 5.2)
6. Compute the similarity metric by normalizing the alignment score. (Section 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. A 3 by 3 affine matrix $\mathbf{F}$ and displacement vector $\mathbf{d}$ transforms $\mathbf{v_i}$ into $\tilde{\mathbf{v}}_\mathbf{i}$ as follows:

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

Similarly to Sumner et al. [SZGP05], 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

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

Non-translational component of $\mathbf{F}$ encodes the change in orientation, scale, and skew induced by the deformation. 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.

In continuum mechanics, the matrix $\mathbf{F}$ is called deformation gradient tensor [CLA*78] 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 stretched first and then rotated. Then we have $\mathbf{F}=\mathbf{RU}$, where $\mathbf{R}$ denotes the rotation tensor and $\mathbf{U}$ the stretch tensor. Since we want to differentiate triangles according to their degree of stretch, 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 stretch tensor. We obtain principal stretches by the eigenanalysis of $\mathbf{C}$, and compute the deformation of a triangle as $(\lambda_1+1/\lambda_3)$, where $\lambda_1$, $\lambda_2$ and $\lambda_3$ are the principal components of $\mathbf{C}$.

As mentioned earlier, 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,..., $M$, as a vector of strain values $\mathbf{s^p}=(s_1^p,\ldots,s_N^p)^T$, which we obtain by the method described in Section 4.1. The strains are then normalized into [0 1] by using a Gaussian Kernel Function (GKF):

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

where $\sigma$ 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 $\tau_s$:

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

The threshold $\tau_s$ has been fixed as 0.5 in our experiments.

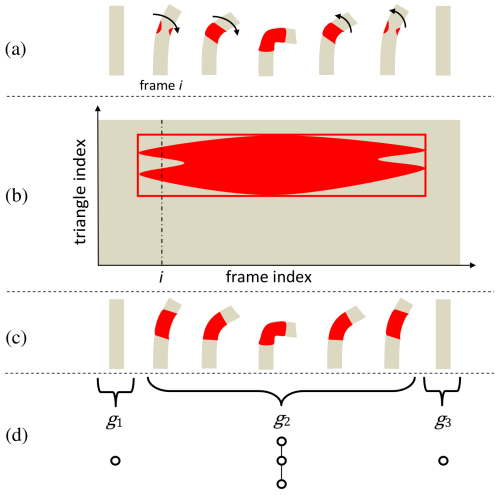


**Figure 1:** *An example of spatio-temporal segmentation of 'bending-cylinder'. (a) Binary labeling. (b) Spatio-temporal segmentation. (c) Spatio-temporal segment. (d) Evolving graph representation.*

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 1(a) illustrates this idea with a 'bending-cylinder' example. The red regions represent the 'deformed' regions with a set of 'deformed' triangles that are connected. Figure 1(b) 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 are resulted 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 Figure 1(b). 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 Figure 1(b), also shown in red region in Figure 1(c)). 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 along either space or time of each triangle in **S**.

**Algorithm 1** Spatio-temporal segmentation

**Require:** $L_i^p$, $ST_i^p = 0$, $(i = 1,\ldots,N,\ p = 1,\ldots,M)$,
  **S=T=I=P=Ø**
  **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** $t$], 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**

### 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. 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 Figure 1(d).

## 5. Pairwise sequence alignment

In this section, we present a metric for computing 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.

### 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 Section 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 clustering; this is because the graphs have different number of vertices and edges, and these clustering methods work only for vectors of same dimension. To overcome this problem, we adopted the graph embedding method proposed by Riesen et al. [RB09a, RB09b]. 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 $V_i = (d(g_i, g_1), d(g_i, g_2), \ldots, d(g_i, g_n))^T$, 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. [NRB06] have proposed an efficient algorithm to compute the graph edit distance.

The overview of graph embedding algorithm is shown in Figure 2. After the graph embedding, each graph is represented with a vector $V_i = (d(g_i, g_1), \ldots, d(g_i, g_n))^T$, $i \in [1, \ldots, n]$.

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 $n$). In order to reduce the dimension of the data, we apply the Principal Component Analysis (PCA) [Jol05]. 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 $r$.

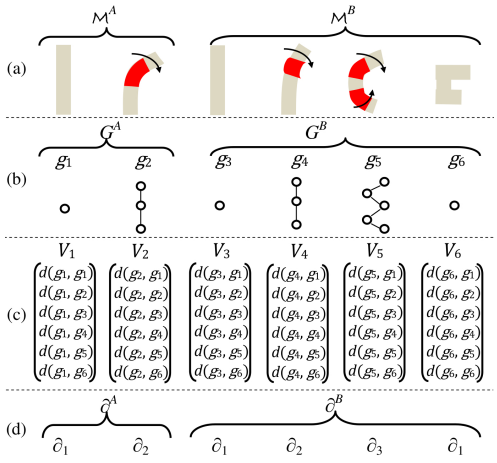


**Figure 2:** *Graph embedding and clustering. (a) The input deforming meshes $\mathcal{M}^A$ and $\mathcal{M}^B$. (b) The sequences of evolving graphs $G^A$ and $G^B$. (c) The graph embedding. Each graph gi is represented with a vector $V_i$, where $d(g_i, g_j)$ denotes the graph edit distance between graphs $g_i$ and $g_j$. (d) The sequences of graph cluster labels $\partial^A$ and $\partial^B$.*

Finally, we apply K-means clustering method on the vectors $V_i'$ to cluster all the graphs $g_i$ into $K$ cluster; $K$ 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 $k^{th}$ cluster ($k \in [1, \ldots, K]$) are given the same cluster label $\partial_k$. 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 $\partial^A$ and $\partial^B$. 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 $\partial_k$, 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 (Section 5.2).

Figure 2(d) 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 $\partial^A$ and $\partial^B$, respectively.

### 5.2. Local sequence alignment

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

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 [SW81], 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.

In order to compute the alignment between the cluster label sequences $\partial^A$ and $\partial^B$, we first compute the affinity matrix of the clusters. As explained in Section 5.1, each of these cluster labels $\partial_k$ corresponds to a cluster whose ceter 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, $D_{k_1K_2} = \sqrt{c_{k_1} - c_{k_2}}$. The affinity matrix $\vartheta$ is a matrix whose dimension is $K$ by $K$; 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:

$$\vartheta_{k_1k_2} = \bar{D} - D_{k_1K_2}, \text{ with } k_1, k_2 \in [1, \ldots, K], \qquad (1)$$

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

Once the similarity matrix has been computed, we use the improved Smith-Waterman algorithm proposed by Barton et al. [Bar93]. An Matlab implementation is available. This algorithm takes as input the two cluster label sequences $\partial^A$ and $\partial^B$ with their corresponding similarity matrix $\vartheta$; it generates a set of pairs of matching cluster labels $Q : \{\partial^A_i \leftrightarrow \partial^B_{Q(i)}\}$, where $Q(i)$ indicates the label in $\partial^B$ that is aligned to the $i^{th}$ label in $\partial^A$, 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:

$$\delta_{AB} = \sum_{i=1}^{n_Q} \vartheta_{\partial^A_i \partial^B_{Q(i)}} - T \cdot \varepsilon, \qquad (2)$$

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

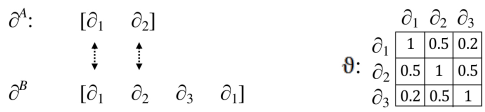


**Figure 3:** *The sequence alignment between $\partial^A$ and $\partial^B$. Matching cluster labels are shown with dashed lines.*

The matching score $\delta_{AB}$ is simply the summation of the similarity values $\vartheta_{\partial^A_i \partial^B_{Q(i)}}$ of each of the matching pairs of cluster labels subtracted by $T \cdot \varepsilon$ which is the penalty score of the gaps. Figure 3 shows an alignment score between $\partial^A$ and $\partial^B$ without a gap. Here the alignment score is $\delta_{AB} = \vartheta_{\partial^A_1 \partial^B_1} + \vartheta_{\partial^A_2 \partial^B_2} - 0 = 2$.

Although $\delta_{AB}$ in Equation 2 can be negative, the algorithm that computes the matching score must return a non-negative result. This is because the empty set $Q$=Ø is always taken into account when computing the most optimal alignment. In case of mismatching between $\partial^A$ and $\partial^B$ such that $\delta_{AB}$ in Equation 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 $O(n^3)$ [Jol05], followed by the K-means clustering whose time complexity is $O(n^{rK+1} \log n)$ [IKI94] with $r$ being the number of principal components used for the PCA and $K$ the number of clusters (see Section 5.1). Our algorithm also requires computing the sequence alignment whose time complexity is $O(n^A \cdot n^B)$ [SW81] 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 [NRB06].

## 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 [tos14] with a walking skeleton. The two other models, “Head” and “Face_1” are obtained by linear interpolation of 8 key poses (anger, fury, grin, laugh, rage, sad, smile and surprise) from Sumner et al.’s work on Deformation Transfer [mes14]. “Camel” and “Horse_1” are also from [mes14]. “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 are shown in Figure 5.

| Name | Nb. of triangles | Nb. of frames | Name | Nb. of triangles | Nb. of frames |
|---|---|---|---|---|---|
| Camel | 43778 | 48 | Boy | 10146 | 54 |
| Horse_1 | 16858 | 48 | Head | 31620 | 80 |
| Horse_2 | 29984 | 80 | Face_1 | 57836 | 80 |
| Michael | 29999 | 54 | Face_2 | 1171 | 1473 |
| Gorilla | 29999 | 54 | Face_3 | 1272 | 1064 |

**Table 1:** *The deforming meshes used in our experiments.*

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

**Similarity measurement**. 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 minutes for each data in a matlab implementation. Figure 5 shows several segmentation results we have obtained by 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 material.

Since the alignment score in Equation 2 depends on the number of aligned labels and the affinity matrix, we compute a similarity value by normalizing the alignment score as follows:

$$\rho_{AB} = \frac{\delta_{AB}}{\sqrt{\delta_{AA} \cdot \delta_{BB}}}. \quad (3)$$

Figure 4 shows the similarity scores we obtained for the example models. As expected, deforming meshes with similar motion shows 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.

Our similarity metric can distinguish motion speed of input deforming meshes. Although “Horse_1” and “Horse_2” have the same frame rate and motion, the similarity between

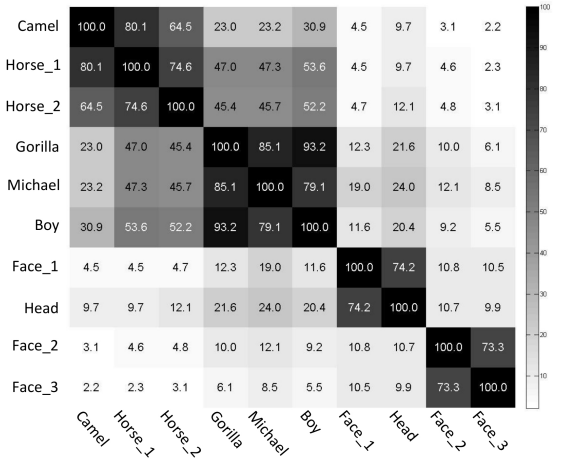


**Figure 4:** *Matrix of similarities among deforming meshes. The values are shown in percentage (%).*

them is not 100%, because their motion speeds are different (“Horse_2” is slower than “Horse_1”).

**Limitations**. One limitation of the proposed similarity metric is its expensive computational cost, mainly due to the computation of GED. With all evolving graphs (composed of 1135 graphs) of the ten dataset we have used in our results, it takes about two hours 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, a work that has not been done before. In particular, we develop the idea of using both spatial and temporal deformation behaviors of the mesh, which we encode as an evolving graph represented. Based on this representation, we further developed a similarity metric by adopting sequence comparison. The results show that our similarity metric successfully compares deforming meshes according to their motions.

One obvious potential of our segmentation-based similarity metric 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.

**Acknowledgement.** We are grateful to MIRALab at University of Geneva, who has provided us the scanned face data. We would like to thank Frédéric Larue and Olivier Génevaux for providing us with the facial motion capture data, and thank Arash Habibi for providing us the "Horse_2" data. We also would like to thank Vasyl Mykhalchuk, who developed a tool for visualizing the evolving graphs. This work has been supported by the French national project SHARED (Shape Analysis and Registration of People Using Dynamic Data, No.10-CHEX-014-01).

## References

[AKM*06] ATTENE M., KATZ S., MORTARA M., PATANÉ G., SPAGNUOLO M., TAL A.: Mesh segmentation-a comparative study. In *Shape Modeling and Applications, 2006. SMI 2006. IEEE International Conference on* (2006), IEEE, pp. 7–7. 1

[Bar93] BARTON G. J.: An efficient algorithm to locate all locally optimal alignments between two sequences allowing for gaps. *Computer applications in the biosciences: CABIOS 9*, 6 (1993), 729–734. 5

[BLVD11] BENHABILES H., LAVOUÉ G., VANDEBORRE J.-P., DAOUDI M.: Learning boundary edges for 3d-mesh segmentation. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 2170–2182. 2

[CGF09] CHEN X., GOLOVINSKIY A., FUNKHOUSER T.: A benchmark for 3d mesh segmentation. In *ACM Transactions on Graphics (TOG)* (2009), vol. 28, ACM, p. 73. 1

[CLA*78] CRANDALL S. H., LARDNER T. J., ARCHER R. R., COOK N. H., DAHL N. C.: An introduction to the mechanics of solids. 2

[GF08] GOLOVINSKIY A., FUNKHOUSER T.: Randomized cuts for 3d mesh analysis. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 145. 1

[GF09] GOLOVINSKIY A., FUNKHOUSER T.: Consistent segmentation of 3d models. *Computers & Graphics 33*, 3 (2009), 262–269. 2

[HKG11] HUANG Q., KOLTUN V., GUIBAS L.: Joint shape segmentation with linear programming. In *ACM Transactions on Graphics (TOG)* (2011), vol. 30, ACM, p. 125. 2

[IKI94] 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* (1994), ACM, pp. 332–339. 5

[Jol05] JOLLIFFE I.: *Principal component analysis*. Wiley Online Library, 2005. 4, 5

[KHS10] KALOGERAKIS E., HERTZMANN A., SINGH K.: Learning 3d mesh segmentation and labeling. *ACM Transactions on Graphics (TOG) 29*, 4 (2010), 102. 2

[KY10] KALAFATLAR E., YEMEZ Y.: 3d articulated shape segmentation using motion information. In *Pattern Recognition (ICPR), 2010 20th International Conference on* (2010), IEEE, pp. 3595–3598. 2

[LWC06] LEE T.-Y., WANG Y.-S., CHEN T.-G.: Segmenting a deforming mesh into near-rigid components. *The Visual Computer 22*, 9-11 (2006), 729–739. 2

[LZ04] LIU R., ZHANG H.: Segmentation of 3d meshes through spectral clustering. In *Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on* (2004), IEEE, pp. 298–305. 1

[mes14] Mesh data from deformation transfer for triangle meshes, 2014. URL: http://people.csail.mit.edu/sumner/research/deftransfer/data.html. 5

[MZP06] MAMOU K., ZAHARIA T., PRÊTEUX F.: A skinning approach for dynamic 3d mesh compression. *Computer Animation and Virtual Worlds 17*, 3-4 (2006), 337–346. 2

[NRB06] NEUHAUS M., RIESEN K., BUNKE H.: Fast suboptimal algorithms for the computation of graph edit distance. In *Structural, Syntactic, and Statistical Pattern Recognition*. Springer, 2006, pp. 163–172. 4, 5

[RB09a] RIESEN K., BUNKE H.: Graph classification based on vector space embedding. *International Journal of Pattern Recognition and Artificial Intelligence 23*, 06 (2009), 1053–1081. 4

[RB09b] RIESEN K., BUNKE H.: Reducing the dimensionality of dissimilarity space embedding graph kernels. *Engineering Applications of Artificial Intelligence 22*, 1 (2009), 48–56. 4

[Sha08] SHAMIR A.: A survey on mesh segmentation techniques. In *Computer graphics forum* (2008), vol. 27, Wiley Online Library, pp. 1539–1556. 1

[SSK05] 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* (2005), ACM, pp. 209–217. 2

[SvKK*11] 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)* (2011), vol. 30, ACM, p. 126. 2

[SW81] SMITH T. F., WATERMAN M. S.: Identification of common molecular subsequences. *Journal of molecular biology 147*, 1 (1981), 195–197. 5

[SZGP05] SUMNER R. W., ZWICKER M., GOTSMAN C., POPOVIĆ J.: Mesh-based inverse kinematics. In *ACM Transactions on Graphics (TOG)* (2005), vol. 24, ACM, pp. 488–495. 2

[tos14] Tosca, 2014. URL: http://tosca.cs.technion.ac.il/book/resources_data.html. 5

[WGW*13] WANG Y., GONG M., WANG T., COHEN-OR D., ZHANG H., CHEN B.: Projective analysis for 3d shape segmentation. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 192. 2

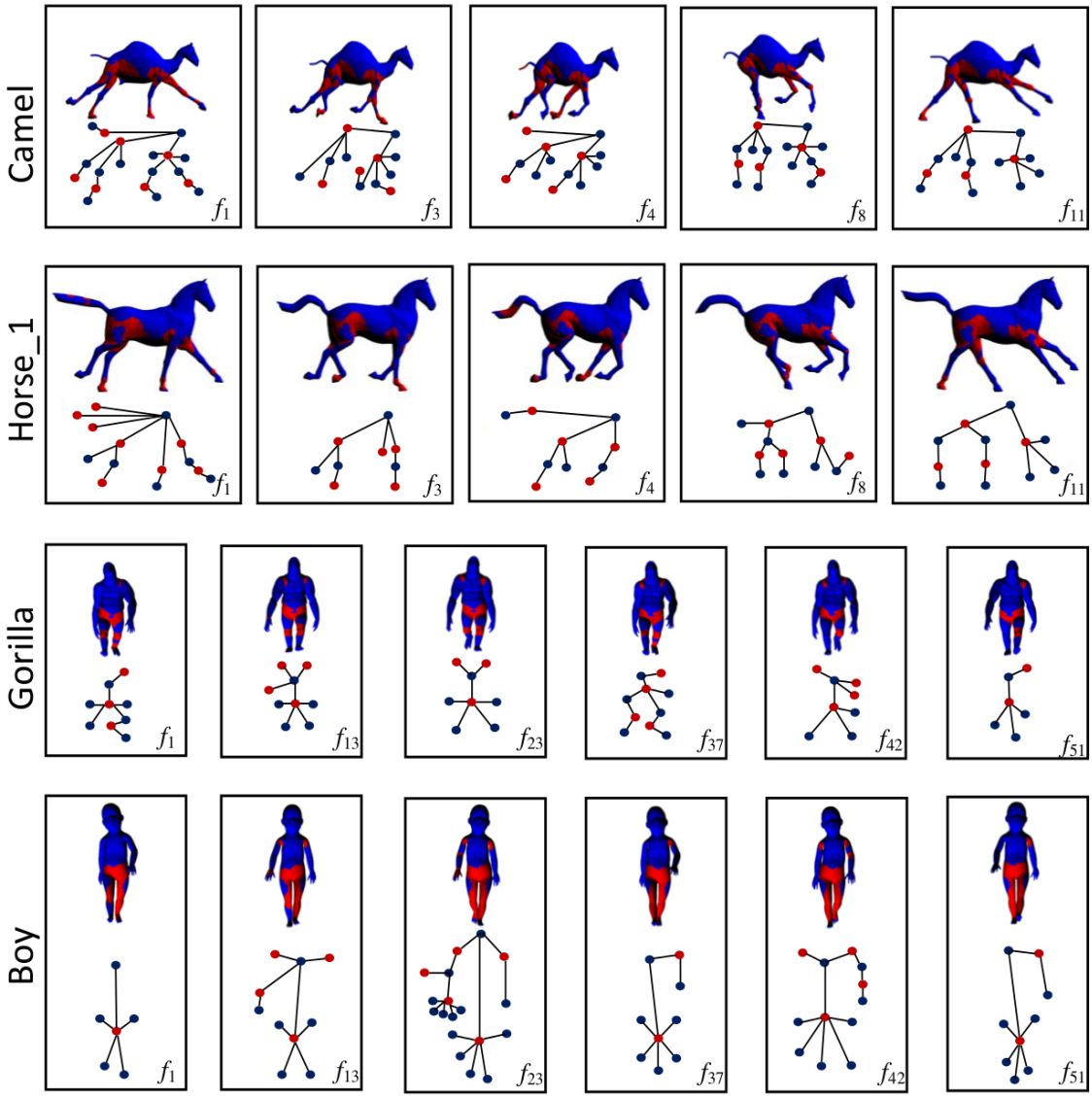


**Figure 5:** *The spatio-temporal segmentation and the graph representation of "Camel", "Horse_1", "Gorilla" and "Boy".*