CGI2014 manuscript No.
(will be inserted by the editor)

Temporal Segmentation of Deforming Meshes

Guoliang Luo · Hyewon Seo · Frederic Cordier

Abstract In this study, we investigate a new method for temporal segmentation of a deforming mesh, i.e., a sequence of meshes with fixed topology. The main idea is to use the deformation behavior of the mesh triangles to measure the distances between each frame pair, from which optimization based segmentation is formulated. More specifically, we perform temporal segmentation by finding frame boundaries that minimize within-segment distances. Our experimentation on numerial examples confirms the effectiveness of the presented approach. It further shows that we can obtain consistent temporal segmentation on different deforming meshes exhibiting identical or similar motions, despite their shape differences.

Keywords Temporal Segmentation · Deforming Mesh

1 Introduction

With an abundance of animation techniques available today, animation data has become a subject of various data processing techniques in Computer Graphics community, such as compression and mesh segmentation. Created from animation software or from motion capture data, a large portion of the animation data is 'deforming mesh', an ordered sequence of static mesh frames whose topology is fixed (fixed number of vertices and fixed connectivity).

Most existing works on deforming mesh segmentation compute spatial clustering according to geodesic and kinematic affinities of vertices or triangles [6, 11]. In such cases, it is clear that the segmentation results may significantly be different depending on the deformation exhibited on the mesh. We believe that temporal segmentation should be preceded prior to spatial segmentation, in order to obtain sensible results that comply well with the given deformation.

In this paper, we propose a new method for temporal segmentation of a deforming mesh. By taking the first frame as a reference frame, we first compute the deformation of each triangle at each frame by applying the per-triangle feature descriptor proposed by Luo et al. [8]. Based on this descriptor, we then define distance metric for each frame pair, from which optimization based segmentation is formulated. More specifically, we perform temporal segmentation by finding frame boundaries by minimizing within-segment dissimilarities. To the best of our knowledge, our work is the first that proposes temporal segmentation algorithm for deforming meshes.

The remainder of the paper is organized as follows. In Section 2, we locate our work in the fields of temporal segmentation of character animation data. Section 3 describes our temporal segmentation method based on frame dissimilarities. Next, in Section 4, we demonstrate our results over several deforming meshes, and compare with Barbič et al.'s method [1]. Finally, we conclude the paper in Section 5.

2 Related works

Temporal segmentation of deforming mesh is a relatively unexplored problem in computer graphics. In the field of character animation, however, a number of high-level temporal segmentation techniques have been de-

G. Luo, H. Seo
Université de Strasbourg (ICube, UMR 7357, CNRS), Strasbourg, France. E-mail:{gluo, seo}@unistra.fr.

F. Cordier
Université de Haute Alsace (LMIA, EA 3993), Mulhouse, France. E-mail:frederic.cordier@uha.fr.

veloped. For example, a motion captured animation sequence is segmented into multiple meaningful motion clips (such as walking or running), which are collectively used to organize a structure called 'motion graphs' [5]. Generation of new motion of a character then comes down to traversing the constructed motion graph. Several supervised [4] and unsupervised learning methods [1,3] have been applied here. Barbič et al. [1] propose methods for placing cuts in the sequence by observing the change in the intrinsic dimensionality or the statistical distribution of frame data. Janus and Nakamura adopts Viterbi algorithm for segmenting the motion data, which finds the optimal state sequence in a Hidden Markov model (HMM) that best represents an observed sequence [3].

Low-level segmentation techniques, mostly developed in the area of video processing, segment motion data into atomic segments prior to some other tasks, such as motion classification or recognition. Commonly used are local minima/maxima of velocity [10], and/or angular velocity [2] of some trajectories of material points. Here we propose an distance-based temporal segmentation of mesh sequences, which has not been attempted before.

3 Temporal segmentation

Based on a feature descriptor proposed by Luo et al. [8], we proceed with temporal segmentation as follows: we first compute dissimilarity for every frame pair, then measure the maximal frame dissimilarity within all possible sub-sequences, and take the sub-sequences with small maximal dissimilarity as temporal segments.

3.1 Frame affinities

In [8], Luo et al. have presented a per-triangle based feature descriptor named as strain. The per-triangle strain values are independent from translation and rotation, and higher values indicate larger deformation. 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 triangle strain values $\mathbf{s^p} = \{s_i^p\}, i = 1, \ldots, N$.

In order to filter the exceptional strain values in case of abnormal behaviors of triangles, we normalize the strains by using a Gaussian Kernel function:

$$\bar{s_i^p} = exp(-0.5 \cdot \delta_t^{-2} \cdot (s_i^p)^2), \tag{1}$$

where δ_t is the standard deviation of strain values, as customary in statistics. With normalization, the large strain values by degenerated triangles converge to 0. We then compute pairwise frame dissimilarities as:

$$D_f(p,q) = \|\bar{\mathbf{s^p}} - \bar{\mathbf{s^q}}\|_{L_2}. \tag{2}$$

Based on the distances between frame pairs, we consider every possible sub-sequence [p, q] (from f^p to f^q) as a temporal segment, with its within-segment dissimilarity $\mathbf{D_s}$ defined as follows:

$$D_s(p,q) = \begin{cases} \max\limits_{p \le m,n \le q} D_f(m,n),\ p < q \\ D_s(q,p),\ p > q \\ 0,\ p = q \end{cases} \tag{3}$$

where p, q=1,..., M.

Intuitively, $D_s(p,q)$ is the maximum dissimilarity among frame pairs within [p, q]. Figure 2 shows the color representation of the dissimilarity matrix computed for several animation data.

3.2 Temporal segmentation

Our goal of temporal segmentation is to segment a given mesh sequence $\mathcal{M}$ into distinct $\mathcal{M}_1$,..., $\mathcal{M}_K$ segments. K as well as the boundary frames I_{B_k}, k=1,..., K, (indices of the first frame in each segment) are to be determined. In Figure 1, a matrix representation of the frame dissimilarity is illustrated. Subsequences with low distance values (colored in blue) are considered as candidates for segment boundaries. Then, we use a threshold ρ to determine whether a within-segment dissimilarity is small enough to form a cluster, i.e., if $D_s(p,q) < \rho$, [p, q] is considered as a temporal segment. In our experiment, ρ is driven from $\theta_1 \cdot \max(\mathbf{D_s})$, where $\theta_1 \in$ [0 1] is a user specified parameter. To avoid over-segmentation, we scan the affinity matrix in descending order of subsequence length in favor of longer sub-sequences. In Figure 1, sub-sequence [m,n] is firstly found as a temporal segment, i.e., $D_s(m,n) < \rho$. Afterwards, we recursively repeat the segmentation over the remaining subsequences [1, m-1] and [n+1 M]. Finally, for a temporal segment whose number of frames is less than $\theta_2 \cdot M$ (θ_2 = 0.02), we merge it to its closer neighboring segment.

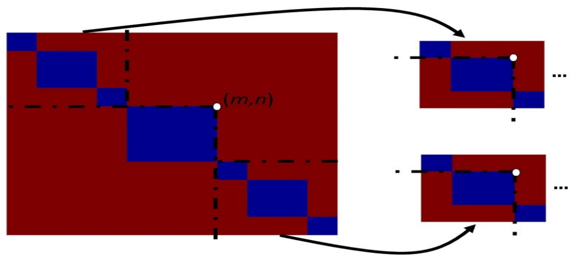

Fig. 1: An example of temporal segmentation.

The complete algorithm is described in Algorithm 1. The function min() returns the minimum value and the corresponding index in a vector. In the worst case, the algorithm runs in $O(M^2)$ time.

Algorithm 1 Temporal segmentation algorithm

```
 1: Init: $I_B = [\,]$, $I_h = 1$, $I_t = M$, $D_s$
 2: function TEMPSEG($I_B, D_s, I_h, I_t$)
 3:     $L = I_t - I_h + 1$
 4:     for $l = L \to 1$ do
 5:         for $p = 1 \to L - l + 1$ do
 6:             $D_{s-sub}(p) = D_s(p, p + l - 1)$
 7:         end for
 8:         $[D_{s-min}, p]$=min($D_{s-sub}$)
 9:         if $D_{s-min} < \rho$ then
10:             $I_B = [I_B\ p]$
11:             $q = p + l - 1$
12:             TEMPSEG($I_B, D_s, I_h, p - 1$)
13:             TEMPSEG($I_B, D_s, q + 1, I_t$)
14:             break
15:         end if
16:     end for
17: end function
18: return $I_B$
```

4 Results and discussions

We have tested our method with both synthesized and motion captured animation data. For the complete temporal segmentation results of deforming meshes, readers may refer to our supplementary video submissions.

Table 1 shows the dimensions of the used dataset, the thresholds and the timings of the temporal segmentation of each data. The method has been implemented in Matlab on an Intel Core 3.40GHz PC with 16GB of RAM. The computation for the temporal segmentation starts to be heavy as the mesh sequence becomes long, or the data dimension becomes large. The segmentation results of the deforming meshes are shown in Figure 2.

Name	Number of faces	Number of frames	θ_1	Timings (second)
Michael	29999	55	0.8	2.1
Gorilla	29999	55	0.8	1.9
Camel	43778	51	0.8	3.3
Horse	16843	51	0.8	0.5
Face1	286	1097	0.6	565.8
Face2	269	1472	0.7	1128.9

Table 1: Timings of temporal segmentation.

Both ‘Michael’ and ‘Gorilla’ data have been generated by rigging TOSCA high-resolution meshes to the same walking skeleton provided by 3ds max studio. These two models have similar temporal segmentation which corresponds to ‘right stop’, ‘left forward&stop’, ‘right forward&stop’, and ‘left forward’.

‘Camel’ and ‘Horse’ data are obtained by interpolating a number of key poses that are available from Sumner et al.’s work on Deformation Transfer [9]. The temporal segmentation we have obtained for ‘Camel’ corresponds to ‘run’, ‘head-right’, ‘head-left’, ‘run’, ‘head-down’ and ‘head-up’. Because the tail of ‘Horse’ occupies around 8% of mesh, comparing to 1% for the tail of ‘Camel’, and the tail undergoes high deformation from f^{12} to f^{18}, we obtain one more segment for ‘Horse’ between $[f^{12}, f^{18}]$.

We acquired the ‘Face’ data by motion capturing two person’s facial expressions by using Vicon system. Each of them performs facial expressions of predefined order: three times of ‘eyebrow raise’, ‘anger’, ‘disgust’, ‘fear’, ‘happiness’, ‘surprise’, ‘sadness’, with a ‘neutral’ face intervals in between. In Figure 2, we obtain 13 segments for both of the data, where the three ‘eyebrow raise’ are recognized as one segment. By using a smaller θ_1, we can obtain 19 segments that the three ‘eyebrow raise’ are separated into three temporal segments. The segmentation results of ‘Face 1’ by using different thresholds are shown in the supplementary video.

We have compared our method with a Principal Component Analysis (PCA) based motion segmentation method for motion capture data, proposed by Barbič et al. [1]. In order to adapt the method for deforming meshes, one can either (a) replace the joints with triangles as the primitives within each frame, or (b) extract the skeleton of the deforming mesh and directly apply the method on the skeleton sequence. For the sake of simplicity, we have chosen the method (a).

In [1], Barbič et al.’s method starts with a short initial motion segment to estimate the number of Principal Components (PCs). Then, for the successive frames that can be precisely reconstructed by using the estimated number of PCs, we merge them with the initial segment, otherwise, a boundary is made. The remaining motion sequence is segmented by repeating this procedure. In Figure 2(b), for applying Barbič et al.’s method on ‘Camel’, we set the length of the initial segment as 7, so that it is sufficient for PCA method to capture the features of ‘run’ motion. However, because this length is longer than the durations of ‘head-right’ and ‘head-left’ (5 frames), we obtain boundary frames (f^{18} and f^{25}) shifted from the truth (f^{15} and f^{20}). Moreover, since the ‘head-down’ motion is too short (f^{41} to f^{44}), the initial segment (f^{41} to f^{47}) contains both ‘head-down’ and ‘head-up’ motions. For this reason, PCA method over-fitted the initial segment and therefore cannot separate the two motions. In order to

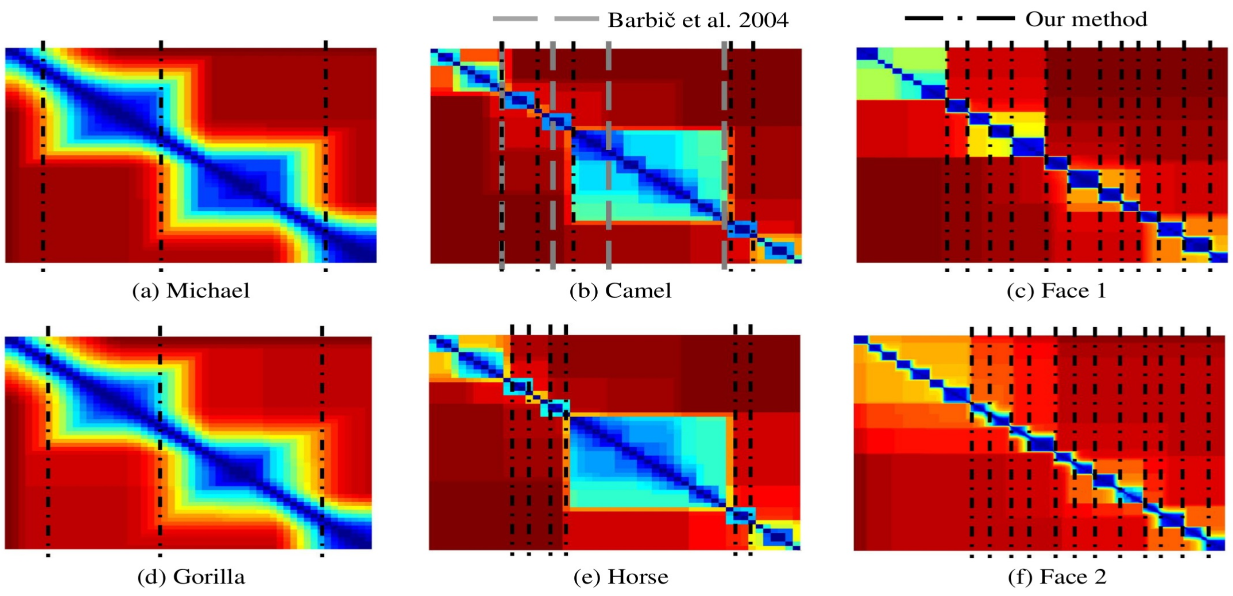

Fig. 2: Temporal segmentation results and comparisons.

avoid the over-fitting affects, Lin et al. apply Barbič et al.'s method in both forward and backward directions, and unify the two boundary sets [7]. However, Lin et al.'s method tends to produce over segmentation. On comparison, our method by minimizing within-segment frame distance neither requires a initial segment nor has over-fitting affects.

5 Conclusions

We have proposed a new method for the temporal segmentation of deforming meshes, a work that has not been done before. Based on a deformation-based feature descriptor, we first compute affinities for every frame pair, then identify sub-sequences with highest inter-frame affinities as temporal segments. The method successfully handles sequences with over thousand frames, as well as meshes with thousands of triangles.

In our experiments, the values of the threshold θ_1 are provided by user depending on which level of motion details are desired. An interesting improvement would be to learn θ_1 from user-supplied segmentation. Additionally, it is also our intention to further extend the method so as to obtain consistent segmentation across a number of objects undergoing similar deformation. With the consistent temporal segmentation results, applications such as animation editing and shape retrieval according to motions can be developed in the future.

References

1. Barbič, J., Safonova, A., Pan, J.-Y., Faloutsos, C., Hodgins, J. K., and Pollard, N. S. Segmenting motion capture data into distinct behaviors. In *Proceedings of Graphics Interface 2004* (2004), Canadian Human-Computer Communications Society, pp. 185–194.
2. Fod, A., Matarić, M. J., and Jenkins, O. C. Automated derivation of primitives for movement classification. *Autonomous robots 12*, 1 (2002), 39–54.
3. Janus, B., and Nakamura, Y. Unsupervised probabilistic segmentation of motion data for mimesis modeling. In *Advanced Robotics, 2005. ICAR'05. Proceedings., 12th International Conference on* (2005), IEEE, pp. 411–417.
4. Kahol, K., Tripathi, P., and Panchanathan, S. Automated gesture segmentation from dance sequences. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on* (2004), IEEE, pp. 883–888.
5. Kovar, L., Gleicher, M., and Pighin, F. Motion graphs. In *ACM SIGGRAPH 2008 classes* (2008), ACM, p. 51.
6. Lee, T.-Y., Wang, Y.-S., and Chen, T.-G. Segmenting a deforming mesh into near-rigid components. *The Visual Computer 22*, 9-11 (2006), 729–739.
7. Lin, I.-C., Peng, J.-Y., Lin, C.-C., and Tsai, M.-H. Adaptive motion data representation with repeated motion analysis. *Visualization and Computer Graphics, IEEE Transactions on 17*, 4 (2011), 527–538.
8. Luo, G., Cordier, F., and Seo, H. Similarity of deforming meshes based on spatio-temporal segmentation. In *Eurographics Workshop on 3D Object Retrieval* (2014), Eurographics Association, pp. 77–84.
9. Sumner, R. W., and Popović, J. Deformation transfer for triangle meshes. In *ACM Transactions on Graphics (TOG)* (2004), vol. 23, ACM, pp. 399–405.
10. Wang, T.-S., Shum, H.-Y., Xu, Y.-Q., and Zheng, N.-N. Unsupervised analysis of human gestures. In *Advances in Multimedia Information ProcessingPCM 2001*. Springer, 2001, pp. 174–181.
11. Wuhrer, S., and Brunton, A. Segmenting animated objects into near-rigid components. *The Visual Computer 26*, 2 (2010), 147–155.