Vis Comput (2017) 33:1279–1289
DOI 10.1007/s00371-016-1313-1

ORIGINAL ARTICLE

Joint entropy-based motion segmentation for 3D animations

Guoliang Luo[1] · **Gang Lei**[1] · **Yuanlong Cao**[1] · **Qinghua Liu**[1] · **Hyewon Seo**[2]

Published online: 9 September 2016

Abstract With the recent advancement of data acquisition techniques, 3D animation is becoming a new challenging subject for data processing. In this paper, we present a joint entropy-based key-frame extraction method, which further derives a motion segmentation method for 3D animations. We start by applying an existing deformation-based feature descriptor to measure the degree of deformation of each triangle within each frame, from which we compute the statistical joint probability distribution of triangles' deformation between two consecutive subsequences of frames with a fixed length. Then, we further compute joint entropy between the two subsequences. This allows us to extract key-frames by taking the local maximal from the joint entropy curve (or *energy curve*) of a given 3D animation. Furthermore, we classify the extracted key-frames by grouping the key-frames with similar motions into the same cluster. Finally, we compute a boundary frame between each of the two neighboring frames with different motions, which is achieved by minimizing the variance of energy between the two motions. The experimental results show that our method successfully extracts representative key-frames of different motions, and the comparisons with existing methods show the effectiveness and the efficiency of our method.

Keywords Joint entropy · Key-frame extraction · Motion segmentation · 3D animation

1 Introduction

With the increasing advancement of 3D animation techniques, 3D animation data are becoming a new subject with the challenge for data processing. Similar to the processing of video and motion capture data, temporal segmentation is to cut a long sequential data by detecting boundary frames, or key-frames, that undergo distinctive changes. Such temporal segmentation results can be further used for variant applications, such as action recognition and shape retrieval.

In this work, we propose a joint entropy-based method for key-frame extraction, based on which we further propose a motion segmentation method for 3D animations. Using an existing deformation-based feature descriptor, we first measure the deformation of each triangle within each frame. To observe the changes of deformation in the animation, we define a modified joint probability distribution that statistically measures the deformation changes of each triangle in successive frames. Then, we use the modified joint probability to compute modified joint entropy, from which we extract the frames with local maximum values as key-frames. Note that we have a sequence of extracted key-frames, we further classify them based on motions, i.e., the key-frames with similar motions are grouped into the same cluster. Furthermore, between each of the two different neighboring key-frames, we compute a boundary frame that cuts the sequence into subsequences with different motions.

Denote that this work is an extended version of our previous short paper [15], which focus on the motion-based key-frame extraction for 3d animations. In this paper, we have not only rephrased the abstract, introduction and related

Electronic supplementary material The online version of this article (doi:10.1007/s00371-016-1313-1) contains supplementary material, which is available to authorized users.

✉ Guoliang Luo
28guoliang.luo@gmail.com

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

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

works sections, but also included a large portion of the following extended works:

- Key-frame clustering. The key-frames within the same cluster are recognized as the same motion. See Sect. 4.1.
- Motion segmentation for 3D animations. To cut an animation into subsequences with different motions, we compute a boundary frame between two neighboring key-frames with different motions. See Sect. 4.2.
- Comparative evaluation. To evaluate our motion segmentation method, we have generated a set of 3D animations using CMU motion captured datasets [2]. By applying our method over the 3D animations, and compare the performance of the temporal segmentation methods [20,25,26] on the corresponding motion captured data. Denote that this work is particularly important because the temporal segmentation method for 3D animations remains a relatively unexplored topic. See Sect. 5.
- Quantitative error measurement. We have formulated an error measurement strategy for the quantitative evaluation of the proposed segmentation method, which shows the effectiveness of our method. See Sect. 5.3.
- Discussion on parameters. We have provided a further discussion on the parameter settings and the limitations of our method in Sect. 5.4.

The rest of the paper is organized as follows. In Sect. 2, we first review the existing temporal segmentation methods used for different data types. Then, we present our joint entropy-based key-frame extraction method in Sect. 3. Based on the extracted key-frames, in Sect. 4, we further present a motion segmentation method to cut a 3D animation into the subsequences of different motions. After that, we evaluate the presented key-frame extraction method and the motion segmentation method, and show the experimental results in Sect. 5. Then, we finally conclude the works in Sect. 6.

2 Related works

Among the temporal segmentation techniques in both Computer Vision and Computer Graphics fields, the most popular method is achieved by detecting abrupt changes between neighboring frames, i.e., a frame with large amount of measurable differences with referring to previous frame. Such frames with abrupt changes, also known as key-frames, provide important hints for temporal segmentation.

For the temporal segmentation of videos, Liu et al. detect the turning points of the motion acceleration of the observed object, and take the frames containing turning points as the segmentation boundaries [13]. For the same object, many other criterions have been used in the other works, such as local minima (or maxima) of velocity [23], angular velocity of the observed points or objects [6]. In [11], Krishna et al. recently propose a learning-based approach that first trains a One Class Classifier based on Support Vector Machine [17] with frames in the first 3 s. Then, they apply the learned model over the successive image which returns a novelty score. A boundary frame is chosen if the novelty score exceeds a predefined threshold. González et al. [8] estimate camera motion types for video temporal segmentation using hierarchical hidden Markov model (HHMM) [5], which consists of two hidden Markov model (HMM) [4,18]: one is trained to model the exact hidden states within a motion type, the other is to model the transition between the current motion and the next motion to cut a video sequence into different motions. These temporal segmentation methods for video/image data normally are devised based on various local properties, such as color, brightness, contrast, and saturation. However, 3D models mainly have geometrical property, which limits the possibilities to apply the above segmentation methods for 3D animations.

For the motion capture data, Barbič et al. use a sliding window and conduct spectral analysis within the window, which returns a reconstruction error [3]. Then, they detect the boundary frame over the error curve if the error exceeds a predefined threshold. In [10], Janus and Nakamura follow the same scheme but modeling the scanned frames with hidden Markov model (HMM) [4], and observes the changes of the probability density of the HMM. Although the authors have demonstrated the methods for mocap data, they are hardly applicable for animations due to data size, i.e., 3D models may have large quantity of vertices while motion capture data have a small number of joints with sequential angular data.

The number of the existing works on the temporal segmentation of 3D animations is rather small [12,16]. Lee et al. propose a Genetic Algorithm-based optimization method to extract key-frames that can be used to reconstruct other frames [12]. Recently, Luo et al. present a temporal segmentation method for deforming meshes, i.e., an animated mesh with fixed topology [16]. They first compute pairwise frame distance based on deformation, then define a within-segment dissimilarity which is the average of all the pairwise frame distance within a subsequence. The segmentation is finally done by minimizing the sum of the within-segment dissimilarity. Experimented on different shapes with similar motions, the results show consistency of the segmentation method. However, the computation of the algorithm increases exponentially along with the length of the animation.

Our work has been inspired by Zhou et al.'s temporal segmentation method for motion captured data based on (hierarchical) aligned cluster analysis (H)ACA [25,26]. They first conduct a temporal clustering which is similar to the idea of over-segmentation and results in small sequences of motion primitives. The classification of these motion primitives leads to motion segmentation. Instead, for a given

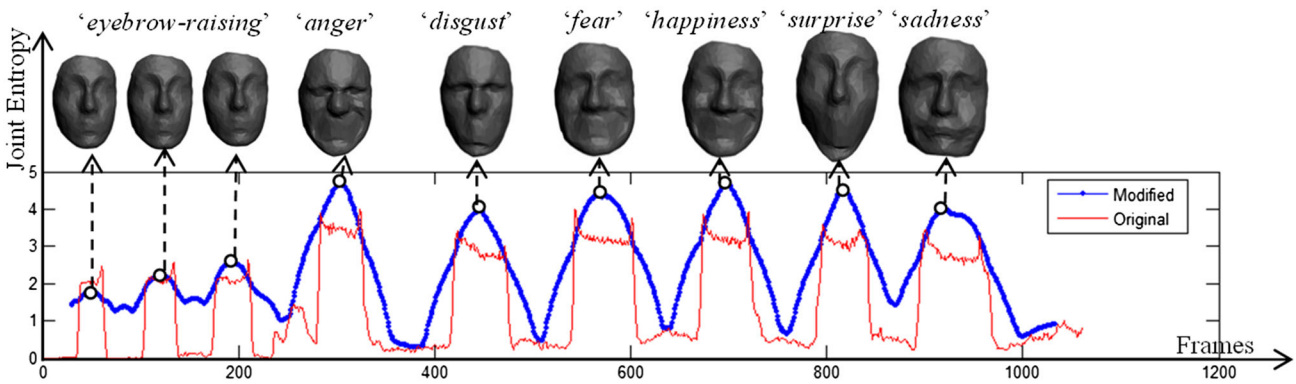

Fig. 1 The original joint entropy between each of the two neighboring frames, and the modified joint entropy between two subsequences of frames of 'Face1'. The *black circles* correspond to the extracted key-frames using our method

deforming mesh, we extract key-frames as the motion primitives, and the further classification of the key-frames enables the longer segments which correspond to motion segments.

3 Key-frame extraction

3D animation can be divided into two types: time-varying mesh and deforming mesh, depending on the topology of a mesh changes along time or not. However, computing the correspondence between consecutive frames may transform a time-varying mesh into deforming mesh, where the correspondence is yet another challenging topic [19,22]. For the sake of simplicity, we assume that the input 3D animations in this work are deforming meshes.

Definition: Strain [14] Strain is a normalized scalar value, which reflects the degree of deformation, i.e., larger values denote higher degree of deformation, and vice-versa. As a local feature descriptor, strain is invariant to rigid deformations, including translation, rotation and scaling.

For a deforming mesh, we start by computing a deformation based feature descriptor, strain [14], to represent the dynamic behaviors of the triangles within each frame. Denote that for our method, one can choose from many optional feature descriptors, such as average geodesic distance [7], heat kernel signature [21], curvature [24] and strain [14]. In this work, we use strain because we intend to perceive our motion segmentation results as temporal segmentation by taking each key-pose as a boundary, and compare with an existing temporal segmentation method by Luo et al., who used strains as feature descriptor [14].

Then, we devise a modified joint probability distribution between neighboring frames, which can be further used for computing modified joint entropy. A modified joint entropy measures the deformation changes between neighboring frames, which we can apply for key-frame extraction by detecting frames with local maximum change of deformation.

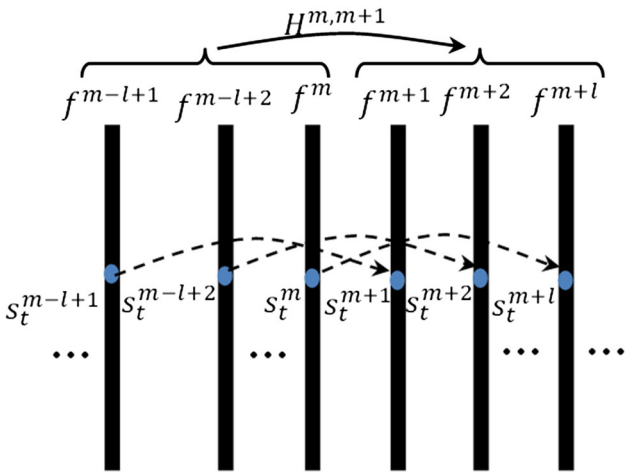

Fig. 2 Modified joint entropy between two successive subsequences of frames. Each *column* represents a frame, and each *dashed arrow* denotes a pair of strain values of the same triangle in f^m and f^{m+l}, where $m = l, \ldots, M - l$

3.1 Joint probability distribution and joint entropy

Given an animated mesh with M frames and T triangles within each frame $f^m, m = 1, \ldots, M$, we compute the strain of each triangle within each frame $s_t^m, t = 1, \ldots, T$, with regard to the corresponding triangle in the rest frame.

In probability theory, joint probability distribution can be seen as a joint cumulative distribution function. For example, with two random variables X and Y, the joint probability distribution is the probability that X falls in range ϕ_x and Y falls in range ϕ_y, respectively. By following the above definition, we define the joint probability distribution between two frames as follows:

$$P_{ij}(f^m \rightarrow f^{m+1}) = \sum_{t=1}^{T} P(s_t^m \in \phi_i, s_t^{m+1} \in \phi_j)/T, \quad (1)$$

where $m = 1, \ldots, M-1$, and $i, j = 1, \ldots, B$, with B denoting that the strain values classified into B scales. Thus, ϕ_i and ϕ_j denote two ranges. We fix $B = 10$ in our experiments, and, therefore, $\phi_1 = [00.1]$, $\phi_2 = (0.10.2]$, $\ldots$, $\phi_{10} = (0.91.0]$. Moreover, $P(s_t^m \in \phi_i, s_t^{m+1} \in \phi_j)$ denotes the amount of the same triangles with strain values $\in \phi_i$ in the frame f^m and becomes $\in \phi_j$ in the next frame f^{m+1}. For the sake of simplicity, we denote $P_{ij}(f^m \rightarrow f^{m+1})$ as P_{ij}^m in the rest context.

Note that we have the joint probability distribution between f^m and f^{m+1}, we proceed to compute the joint entropy as follows:

$$H^m = -\sum_{i,j=1}^{B} P_{ij}^m \cdot log\, P_{ij}^m \tag{2}$$

After computing the joint entropy between each pair of neighboring frames, we obtain a joint entropy curve of a given 3D animation. However, the deformation between neighboring frames is normally small and, thus, can be easily affected by noise. For this reason, the obtained joint entropy curve can hardly be smooth over inter-frame changes. See Fig. 1 with an example of the joint entropy curve of a facial expression data. Although the height of the curve reflects the amount of deformation in the corresponding frames, the frame-to-frame changes are highly sensitive to noises.

3.2 Modified joint entropy

To avoid the noises, we present a modification of the proposed computation of joint probability distribution. Specifically, we compute the joint probability distribution between two consecutive subsequences of frames, see Fig. 2. This also can be expressed with a modification of Eq. 1, as follows:

$$P'_{ij}(f^{\tau_1} \rightarrow f^{\tau_2}) = \Sigma_{t=1}^{T} P(s_t^{\tau_1} \in \phi_i, s_t^{\tau_2} \in \phi_j)/(T \times l) \tag{3}$$

where l is the length of each subsequence, $\tau_1 = (m-l+1) : m$, $\tau_2 = (m+1) : (m+l)$, and $s_t^{\tau_1}$ and $s_t^{\tau_2}$ are the strain values of the two neighboring subsequences. Specifically, $s_t^{\tau_1}$ is a matrix with size of $T \times l$, which contains the strain values of all triangles from the frame f^{m-l} to f^m. Similarly, $s_t^{\tau_2}$ contains the strain values of all triangles from the frame f^{m+1} to f^{m+l}. For the sake of simplicity, we denote $P'_{ij}(f^{\tau_1} \rightarrow f^{\tau_2})$ as $P_{ij}^{m'}$ in the remaining context.

Having the modified joint probability distribution, we compute the modified joint entropy H^m also using Eq. 2, with $P_{ij}^{m'}$ replacing P_{ij}^m. Denote that after the modification, $m = l, .., M-l$.

As can be seen in Fig. 1, the modified joint entropy curve becomes smoother. This is because (1) we compare the strain value of a triangle in f^m with the strain value in f^{m+l} instead of f^{m+1}, see Fig. 2, and (2) we take into account of $2l$ frames, i.e., l frames before and l frames afterwards, for computing the frame entropy of the frame f^m.

3.3 Key-frame extraction

Note that we have obtained the joint entropy curve (*energy curve* in the remaining context) of a given 3D animation, we compute the local maximal of the curve as key-frames, which correspond to the poses that undergo significant deformation within a temporal period. However, due to noises, there may be several local maximal in short temporal range. To avoid such noises, each time we extract the local maximal that is also the maximal within ω frames. The complete key-frame extraction algorithm is described in Algorithm 1. In this algorithm, **max**(V) is a function that returns the index and the corresponding value in V, and **keyFs**(end) is the last item of **keyFs**. Finally, using a low threshold, we drop the extracted local maximal with small values, such as the last frame in Fig. 1. This is because the subject stops performing motions after the last motion, which means the remaining low energies are computed because of noises. Finally, our key-frame extraction algorithm runs in $O(\omega \cdot M)$ time and the computation of modified joint entropy runs in $O(l \cdot M)$ time. In total, the complexity is about $O(M)$ as ω and l are relatively small compared to M.

Algorithm 1 Key-frame extraction algorithm

Inputs: $\mathbf{H}(i), i = 1, \ldots, F$; **keyFs** = [ ]
{/* $\mathbf{H}(i)$ is the joint entropy value of frame f^i, and **keyFs** saves the extracted key-frames. */}
for $i = 1$ **to** $F - \omega$ **do**
  $[loc, val] =$ **max**($\mathbf{H}(i, \ldots, i + \omega)$)
  if **keyFs**(end) $\notin [i, i + \omega]$ **then**
    keyFs = **keyFs** $\cup\, loc$
  else
    if **keyFs**(end) $\in [i, i + \omega]$&**keyFs**(end) $\neq loc$ **then**
      keyFs(end) = loc
    end if
  end if
end for
return $keyFs$

4 Motion segmentation

In this section, we extend the extracted key-frame results towards motion segmentation for 3D animations. We first classify the extracted key-frames as each represents the corresponding motion. Based on the clustering, we represent each key-frame as the average of the key-frames within the same cluster. Then, to determine the motion segmentation boundaries between neighboring key-frames with different motions, we choose the frame that minimizes the variances

of the energy curve for both left-side frames (from previous key-frame to the boundary) and right side (from boundary to the next key-frame).

Denote that we have also experimented the above scheme by replacing energy curve with frame distances, computed as the Euclidean distance between frame vectors $f^m, m = 1, \ldots, M$. However, the results tend to be much less robust due to the noises, since our joint entropy-based energy curve is a smoothed value.

4.1 Key-frame clustering

We first denote the obtained key-frames of a deforming mesh as f^{m_p}, $p = 1, \ldots, P$, where m_p is the index of a key-frame and P is the total number of key-frames.

Note that we have obtained a set of key-frames, we apply K-means clustering method [9] to group the key-frames with similar motions, where the number of clusters K can be predefined as the number of motion types observed from the deforming mesh. Based on the clustering results, we can denote the key-frames within the same cluster with the same $f_k, k = 1, \ldots, K$, which indicates the same motion type. Thus, we can use $f^{m_p} \subseteq f_k$ to denote that the key-frame f^{m_p} exhibits the motion type f_k. Furthermore, to equally represent the key-frames within each cluster, we compute the average of the key-frames within each cluster as $\bar{f}_k$, which contains the averaged strains of the same triangles among the key-frames.

$$\bar{f}_k = (\Sigma_p f^{m_p})/|k| = (\Sigma_{p,t} s_t^{m_p})/|k|, \quad \forall f^{m_p} \subseteq f_k, \tag{4}$$

where $|k|$ is the number of key-frames within the k-th cluster.

Since the key-frames are representative of motions, we also name key-frame clustering as motion clustering in the remaining contexts. See Figs. 3, 4, 5 and 6 for the motion clustering results of the experimental animations.

4.2 Boundaries for motion segmentation

In this section, we propose a method to determine boundaries for motion segmentation by minimizing the variances of the energy between each of the two neighboring different motions.

We compute the boundary between two different motions independently. We first denote the key-frames of the two neighboring motions as $f^{m_{p_1}}$ and $f^{m_{p_2}}$, $p_1, p_2 = 1, \ldots, P$. For each frame in between, i.e., $m_{p_1} < m < m_{p_2}$, we compute the variance of the energy within the two segment separately, and sum up the two variances,

$$\xi^m = var(H^{m_{p_1}}, H^m) + var(H^m, H^{m_{p_2}}) \tag{5}$$

where H^m, $H^{m_{p_1}}$ and $H^{m_{p_2}}$ are the modified joint entropy.

By following the above computation, we obtain a variance curve between two motions $f^{m_{p_1}}$ and $f^{m_{p_2}}$. Then, we choose the frame with minimum variance as the boundary frame, or transition frame. This is because a key-frame is normally the most representative frame of a motion, which also means that the frames within a motion are constantly deforming and, therefore, the variance of energy within a motion is large. On the contrary, the variance of energy within the transition period is smaller. For this reason, we choose the frame between two key-frames with minimum variance of energy as the transition frame between two motions. See the variance of energy computed for several animations in Figs. 3 and 5.

Denote that we have also examined the above scheme by replacing energy curve with frame distances, the Euclidean distance between frame vectors $f^m, m = 1, \ldots, M$. However, the results tend to be much less robust due to the noises. This fact again reflects the advantage of our joint entropy-based smoothing method for energy curve.

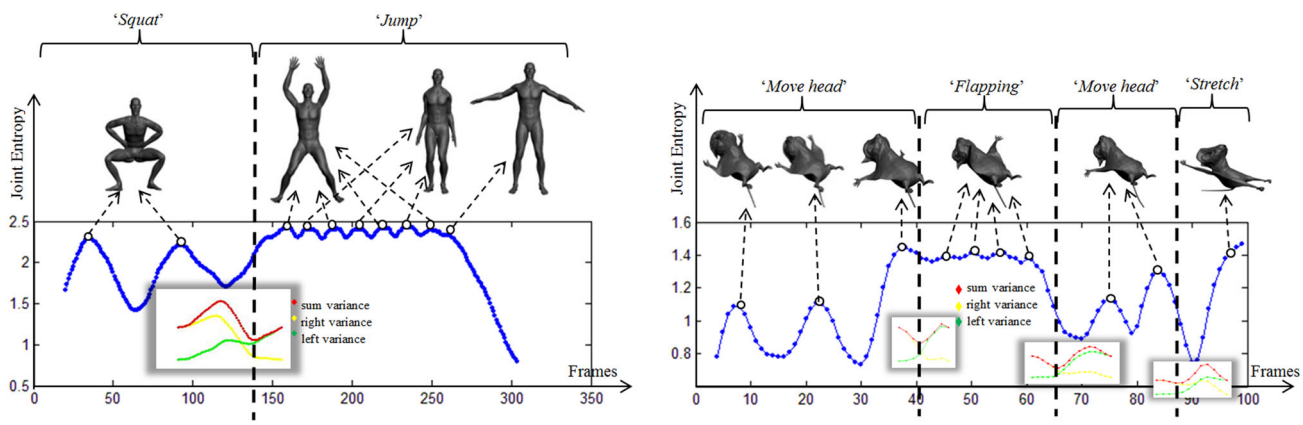

Fig. 3 Extracted key-frames, and the motion segmentation for (*left*) 'Exercise' and (*right*) 'Squirrel' using our method. The floating embedded sub-images are the variance of energy between two different neighboring motions, which is used for computing boundaries

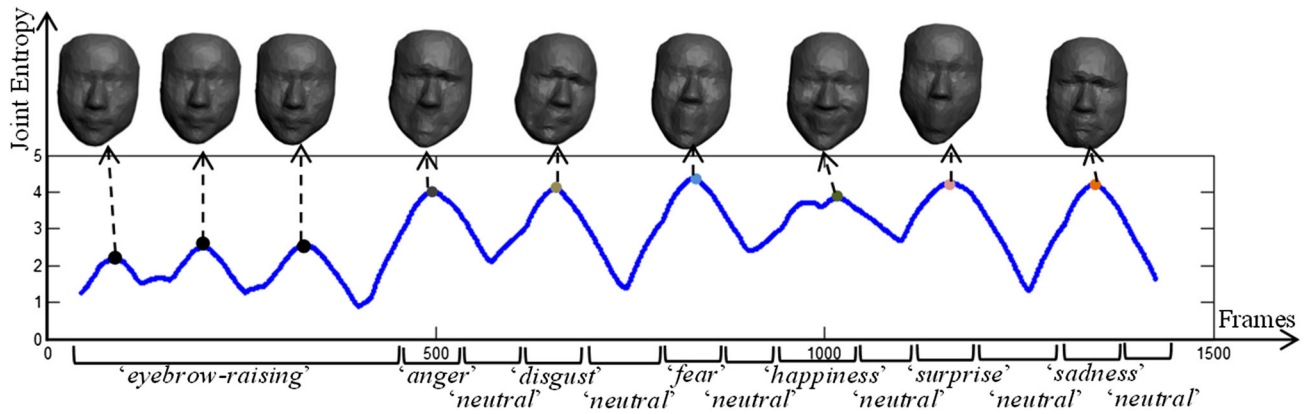

Fig. 4 Comparison with a temporal segmentation method by Luo et al. [16]. With 'Face2' (*top*), our method extracts key-frames corresponding to each facial expression, while (*bottom*) Luo et al.'s method distinguishes different facial expressions from 'neutral' poses but does not differentiate 3-time '*eyebrow-raising*'. Additionally, the extracted key-frames are colored based on motion clustering, see Sect. 4.1

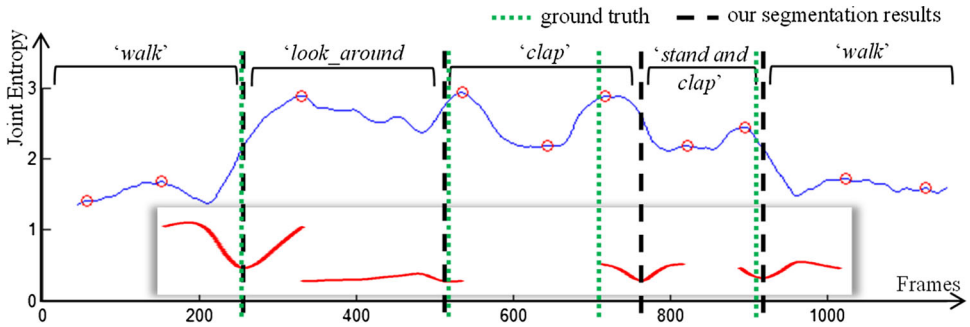

Fig. 5 Motion segmentation of the animation generated using the '86_09' from CMU motion captured data [2]

5 Results and discussion

In this section, we first experiment our key-frame extraction method with a variant of 3D animations. Then, we evaluate the performance of the motion segmentation method by comparing with the existing temporal segmentation methods for both 3D animations (Sect. 5.2) and motion capture data (Sect. 5.3). Our methods are implemented in Matlab environment on an Intel(R) 3.2 GHz computer with 4G memory. One may also find more results in the supplementary video.

5.1 Experimental data

Table 1 shows the data used for experiments, including the number of triangles, the number of frames and the corresponding frame rate γ (*per second*) of each animation. Table 1 also shows the length of subsequence l for computing modified joint entropy, ω for extracting key-frames, and the timings (in *seconds*) for processing each of the data. Detailed discussion on the parameters can be seen in Sect. 5.4.

'Face1' and 'Face2' are obtained by copying two captured facial motions to the corresponding scanned high-resolute face models, with the frame rate $\gamma = 30$ frames/s. Both data contain nine expressions in order: three-time '*eyebrow-raising*', '*anger*', '*disgust*', '*fear*', '*happiness*', '*surprise*' and '*sadness*', and always returns to '*neutral*' pose in between. As shown with snapshots in Figs. 1 and 4, our method correctly discovered the 9 key-frames which correspond to the 9 expressions.

'Exercise' is generated in 3dsmax [1] with the frame rate $\gamma = 5$, by attaching a skeleton motion to a static 3D model, which performs a series of exercise motions: 2-time '*Squat*' and 4-time '*Jump*'. As can be seen in Fig. 3, our method returns the following key-frames: the lowest poses for each of the 2-time '*Squat*', the highest and lowest of arm poses for each '*Jump*' motion (2 key-frames of each '*Jump*'). Denote that the last key-frame is different from the other lowest arm poses of jumping because the character directly returns to rest pose without completing the full jumping motion. On the other hand, by applying motion clustering, the first two '*Squat*' key-frames are grouped into the same cluster, and the other key-frames are recognized as the other motion, i.e., '*Jump*'. Based on the clustering results, our motion segmentation divides the animation into '*Squat*' and '*Jump*' subsequences.

'Squirrel' is a synthesized animation data with the frame rate $\gamma = 3$, for which our method successfully extracts ten key-frames for 'Move head', 'Flapping', 'Move head' and 'Stretch' motions. The motion segmentation results are also shown in Fig. 3. As can be seen, the third key-frame is a

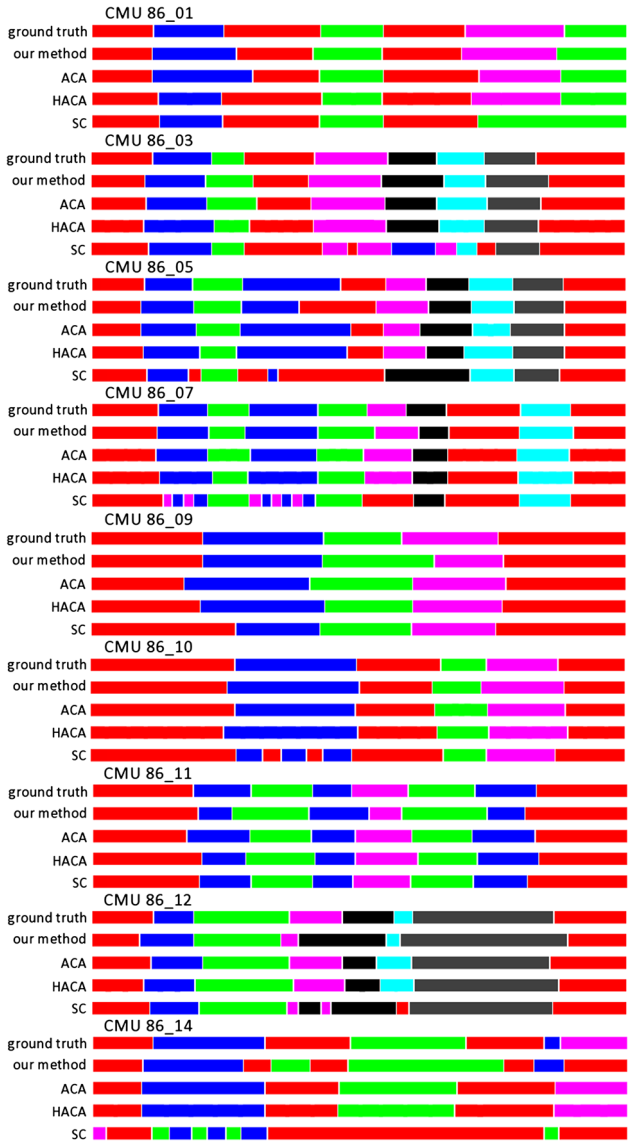

Fig. 6 Motion segmentation result and the temporal segmentation results of a set of motions from CMU motion captured data without 'standstill' motion. The *colors code* the motion clustering results for each of the data using each method

combination of two motions, 'Move head' and 'Flapping', while the 'Move head' dominates and thus recognized as 'Move head' motion by our clustering method. Moreover, our segmentation method successfully classified the key-frames for both 'Move head' motions before and after the 'Flapping' motion.

5.2 Comparison with existing temporal segmentation for 3D animations

In [16], Luo et al. propose a temporal segmentation method for deforming meshes that groups similar neighboring frames into the same segment. For the sake of simplicity, to show the difference between our method and the method in [16], we show the most representative result by comparing with 'Face2'.

Table 1 The deforming meshes used in our experiments

Animations	Triangles	Frames	γ	l/ω	Timings
Face1[a]	1272	1063	30	1.0/2.0	1.8
Face2[a]	1171	1472	30	1.5/2.0	3.0
Exercise	29,999	54	5	4.0/2.0	14.3
Squirrel	7646	103	3	1.3/2.7	0.6

Denote that timings are in the unit of *second*, and l/ω are reported in the unit of the corresponding frame rate γ. For example, l/ω of 'Face1' are $1.0 \cdot \gamma/2.0 \cdot \gamma$, i.e., 30/60, respectively

[a] Courtesy of the IGG lab at the University of Strasbourg

As can be seen in Fig. 4, on comparison, our method is different from Luo et al.' s method which distinguishes facial expressions from '*neutral*' poses. This is because our method applies the motion-based feature descriptor, i.e., strains and, therefore, the detected key-frames correspond to dynamic motions. On the other hand, our method can discover 3-time 'eyebow-raising' without users' further interruption. Additionally, our method $O(M)$ runs more efficiently than their method $O(M^2)$ who requires to compare every possible pair of frames.

5.3 Comparison with the temporal segmentation for motion captured data

To compare our motion segmentation method with the temporal segmentation for motion captured data [25,26], similar to the 'Exercise' data used in Sect. 5.1, we generate a set of 3D animations using 'subject 86' from CMU motion captured data [2]. Without losing the temporal continuity of motions, we sample the original sequence for each of the 4 consecutive frames. Thus, the frame rate γ becomes 30 as the frame rate of the original motion capture data is 120. For all these experimental data, our segmentation method runs in maximally 2 min, including 90 s for extracting key-frames and 30 s for computing the segmentation boundaries. The length of each animation has been shown in Fig. 7. Moreover, since our motion segmentation can only address the animation with dynamic motions, see Sect. 5.2 and Fig. 4, we only experiment with the animations without '*standstill*' motions.

To apply our motion segmentation method, with $\gamma = 30$, we set $l \in [1.3\ 1.7]$ and $\omega \in [2.0\ 2.3]$ for the key-frame extraction, and the number of clusters for the motion clustering can be obtained in the user annotations provided by Zhou et al. [25]. Figure 5 shows the segmentation results of '86_09' from CMU dataset using our method. As can be seen, our result correctly discovers 5 motions and classifies the first and last motion as the same, i.e., '*walk*'. Moreover, 3 out of the computed 4 boundaries almost overlap with the

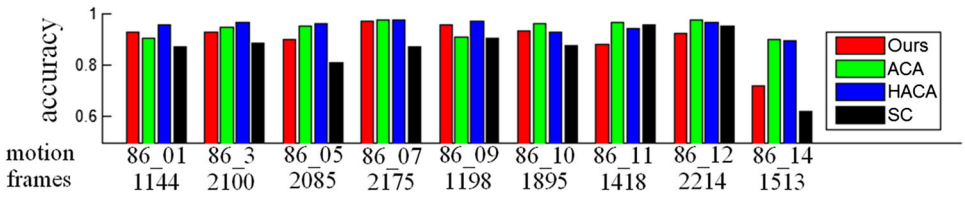

Fig. 7 Accuracy values of different segmentation methods

ground-truth, except the boundary between '*clap*' and '*stand and clap*'. This is because the '*stand and clap*' involves the '*stand*' status, which is a limitation of our motion segmentation method.

Figure 6 shows the segmentation results for each of the experimental data. In specific, each set contains the ground-truth segmentation, the temporal segmentation result for the original motion captured data using ACA [26], HACA [25] and spectral clustering (SC) [20], and the motion segmentation result for the generated animations using the proposed method. The colors indicate whether the motions are the same or not, within the segmentation result for each of the data using each segmentation method, i.e., the color code is valid only within each bar. As can be seen, for most of the experimental data, the segmentation results using our method show: (1) after motion clustering, the order of the motions are the same as the ground-truth. (2) The boundaries between different motions are close to the ground-truth.

Denote that in the last data '86_14', it contains motions '*walk_lead_ball*', '*through_ball*', '*walk_lead_ball*', '*lead_ball_both_hand*', '*walk_lead_ball*', '*through_ball*' and '*walk*' in order. Our method and SC both recognize '*walk*' as the same as '*walk_lead_ball*', which can be explained as they both belong to '*walk*' motion. On comparison, ACA/HACA both fail to recognize the last second motion, '*through_ball*', while our method and SC succeed.

Error measurement To further evaluate the segmentation results of each method shown in Fig. 6, we propose the following error measurement method:

$$\varepsilon = \frac{\Sigma_{i,j=1}^{M} f(c_i, c_j)}{M \cdot (M-1)}, \tag{6}$$

where c_i and c_j are the cluster labels of frames f^i and $f^j (i, j = 1, \ldots, M)$ in the input deforming mesh, and $f(c_i, c_j)$ is an indicator function:

$$f(c_i, c_j) = \begin{cases} 1, & c_i = c_j \\ 0, & c_i \neq c_j \end{cases}$$

Figure 7 shows the accuracy for the segmentation result using each method with each of the CMU data. As can be seen, none of the method's accuracy is prior to the other methods for all data. The average accuracy of each method is 90.3 % (Our method), 94.2 % (ACA), 95.0 % (HACA) and 86.1 % (SC). Although the average accuracy of our method is slightly inferior to Zhou et al.'s [25,26], (1) it is worth to remind that our method directly works on the 3D animations, while the others are proposed for the corresponding motion captured data. That is, our method works efficiently and effectively for 3D animations. (2) In addition, normally, a 3D mesh is smoothly deformed in the neighboring frames. Thus, a small difference of the boundary location will not significantly affect the segmentation results as long as each segment indicates a motion. In fact, for each segmentation boundary's location, the ground truth suggests either a potential range or a specific frame [25]. We choose the latter criteria for the purpose of quantitative evaluation.

5.4 Discussion and limitation

We further discuss about the limitations and the user-specified parameters of our method as follows.

- No static segments. Our motion segmentation method is based on key-frame extraction, which exports the representative poses corresponding to dynamic motions. Then, we compute the segmentation boundaries in between dynamic motions. Due to the difference in the definition of our segmentation, our results do not contain static segments, such as the '*standstill*' segments in [25,26] or the '*neutral*' segments in [16].
- Joint entropy VS. distortion. Although the modified joint entropy proposed in Sect. 3.2 is based on a deformation-based feature descriptor, the entropy (or energy) curves are not equal to the distortion level of animations. Figure 8(1) shows a synthesize example of the strain distribution within a frame. Assuming the strain values within (0.3 0.4] and (0.5 0.6] are changing towards each other with the same rate. That is, the distortion level of a region on mesh increases while another region decreases, but the total distortion of the mesh remains unchange. In fact, as being depicted in Fig. 8 (2,3), the actual process involves two stages, (1) starting with $P'(s_t^{\tau_1} \in (0.3\ 0.4], s_t^{\tau_2} \in (0.4\ 0.5])$ and $P'(s_t^{\tau_1} \in (0.5\ 0.6], s_t^{\tau_2} \in (0.4\ 0.5])$, (2) and then mixing with $P'(s_t^{\tau_1} \in (0.4\ 0.5], s_t^{\tau_2} \in (0.5\ 0.6])$ and $P'(s_t^{\tau_1} \in (0.4\ 0.5], s_t^{\tau_2} \in (0.3\ 0.4])$. This process leads to a varying energy curve by computing the modified joint probability distribution in Eq. 3, which reflects the deforming process of the animation. On the other

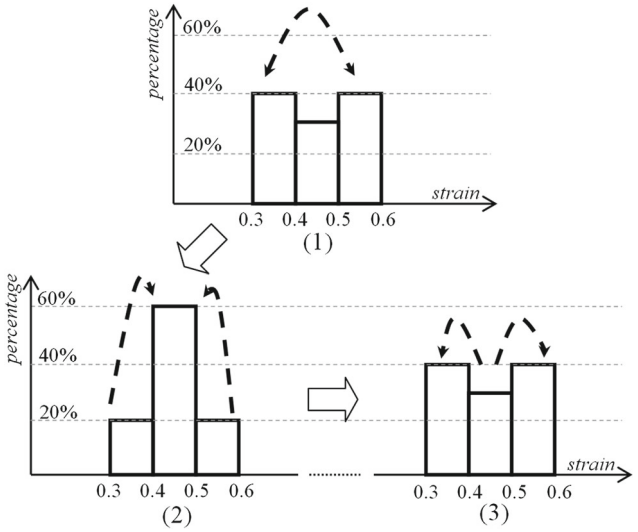

Fig. 8 Joint entropy VS. distortion. (*1*) The distribution of strains within a frame, in which the strain values within (0.3 0.4] and (0.5 0.6] are equally changing/deforming towards each other, (*2*) a middle stage of the deformation and (*3*) the final strain distribution, which is the same in (*1*)

hand, the total amount of the deformation on the mesh remains unchange. Therefore, the entropy curve is more robust than the summation of the deformation for describing the dynamic behaviors in a 3D animation.

- Parameters. We have three key user-specified parameters in the method, i.e., K, ω and l. Among them, (a) K denotes the number of motion types within an animation. Similar to Zhou et al.'s methods [25,26], a user needs to preview the animation and provides K in advance. (b) ω is the width of the temporal window, in which we extract local extrema. In general, large value of ω will lead to the loss of motion segments, while small ω will lead to noises, i.e., trivial movements. In our experiments, we set ω as approximately $2 \cdot \gamma$. (c) l is a smoothing factor for the modified joint entropy, see Sect. 3.2. Normally, we set $l \geq \gamma$ to remove noises. In our experiments, for the animation data with the same frame rate, we set approximately the same l and ω for the data using our method, see Sect. 5.3.

6 Conclusions

We have presented a joint entropy-based method for key-frame extraction of 3D animations. Our method takes the advantage of joint entropy to investigate the deformation changes between neighboring frames. We further improve the robustness over noise by computing joint entropy between two subsequences of frames. Then, we group the similar key-frames and compute a boundary between each of the two neighboring key-frames from different groups, which is also the motion segmentation result of 3D animations. The comparative experiments show that our method (1) not only performs efficiently in $O(M)$ time while an existing method runs for $O(M^2)$ time [16] (M is the length of an animation), (2) but also outputs competitive segmentation results compared to the temporal segmentation algorithms for motion capture data.

In conclusion, our method properly extracts key-frames representing different motions in 3D animations, and the proposed efficient motion segmentation method can produce segmentation results that are highly similar to user annotated results. The results can be potentially used as a candidate preprocessing tool for variant of applications such as shape retrieval and action recognition.

Acknowledgements This work has been jointly supported by the National Nature Science Foundation of China (Nos. 61602222, 61562044), the Science and Technology Research Program funded by the Education Department of Jiangxi Province (No. GJJ150359), the Doctoral Research Project of JXNU (6754), the Science and Technology Research Project of Jiangxi Provincial Department of Education (No. GJJ14246) and the French national project SHARED (Shape Analysis and Registration of People Using Dynamic Data, No. 10-CHEX-014-01). We are also thankful to the assistance from our colleagues Lei Haopeng, Yi Yugen and Li Zhangyu.

References

1. Autodesk inc.: 3d studio max. http://www.autodesk.com/products/3ds-max. Accessed 23 Nov 2015
2. Cmu graphics lab: Carnegie mellon university motion capture database. http://mocap.cs.cmu.edu. Accessed 01 Dec 2015
3. Barbič, J., Safonova, A., Pan, J.-Y., Faloutsos, C., Hodgins, J. K., Pollard, N. S.: Segmenting motion capture data into distinct behaviors. In: Proceedings of Graphics Interface 2004, Canadian Human–Computer Communications Society, pp. 185–194 (2004)
4. Baum, L.E.: An equality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. Inequalities **3**, 1–8 (1972)
5. Fine, S., Singer, Y., Tishby, N.: The hierarchical hidden markov model: analysis and applications. Mach. Learn. **32**(1), 41–62 (1998)
6. Fod, A., Matarić, M.J., Jenkins, O.C.: Automated derivation of primitives for movement classification. Auton. Robots **12**(1), 39–54 (2002)
7. Gal, R., Shamir, A., Cohen-Or, D.: Pose-oblivious shape signature. IEEE Trans. Vis. Comput. Gr. **13**(2), 261–271 (2007)
8. González-Díaz, I., Martínez-Cortés, T., Gallardo-Antolín, A.: Díaz-de María, F.: Temporal segmentation and keyframe selection methods for user-generated video search-based annotation. Expert Syst. Appl. **42**(1), 488–502 (2015)
9. Hartigan, J.A., Wong, M.A.: Algorithm as 136: a k-means clustering algorithm. Appl. Stat. **28**, 100–108 (1979)
10. Janus, B., Nakamura, Y.: Unsupervised probabilistic segmentation of motion data for mimesis modeling. In: 12th International Conference on Advanced Robotics, 2005. ICAR'05. Proceedings. IEEE, pp. 411–417 (2005)

11. Krishna, M.V., Bodesheim, P., Körner, M., Denzler, J.: Temporal video segmentation by event detection: a novelty detection approach. Pattern Recognit. Image Anal. **24**(2), 243–255 (2014)
12. Lee, T.Y., Lin, C.H., Wang, Y.S., Chen, T.G.: Animation key-frame extraction and simplification using deformation analysis. IEEE Trans. Circuits Syst. Video Technol. **18**(4), 478–486 (2008)
13. Liu, T., Zhang, H.-J., Qi, F.: A novel video key-frame-extraction algorithm based on perceived motion energy model. IEEE Trans. Circuits Syst. Video Technol. **13**(10), 1006–1013 (2003)
14. Luo, G., Cordier, F., Seo, H.: Similarity of deforming meshes based on spatio-temporal segmentation. 3D Object Retrieval Workshop, 3DOR (2014)
15. Luo, G., Lei, G., Seo, H.: Joint entropy based key-frame extraction for 3d animations. In: Computer Graphics International (CGI) 2015, the 32nd annual conference (Strasbourg, France, June 2015), Université de Strasbourg (2015)
16. Luo, G., Seo, H., Cordier, F.: Temporal segmentation of deforming mesh. In: Computer Graphics International (CGI) 2014, the 31st annual conference (Sydney, Australia, June 2014), University of Sydney (2014)
17. Maji, S., Berg, A., Malik, J.: Classification using intersection kernel support vector machines is efficient. CVPR 2008. IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp 1–8 (2008)
18. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Proc. IEEE **77**(2), 257–286 (1989)
19. Shamir, A.: A survey on mesh segmentation techniques. In: Computer Graphics Forum, vol. 27, pp. 1539–1556. Wiley Online Library (2008)
20. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 888–905 (2000)
21. Sun, J., Ovsjanikov, M., Guibas, L.: A concise and provably informative multi-scale signature based on heat diffusion. In: Computer Graphics Forum, vol. 28, pp. 1383–1392. Wiley Online Library (2009)
22. Van Kaick, O., Zhang, H., Hamarneh, G., Cohen-Or, D.: A survey on shape correspondence. In: Computer Graphics Forum, vol. 30, pp. 1681–1707. Wiley Online Library (2011)
23. Wang, T.-S., Shum, H.-Y., Xu, Y.-Q., Zheng, N.-N.: Unsupervised analysis of human gestures. In: Advances in Multimedia Information Processing–PCM 2001, pp. 174–181. Springer, Berlin (2001)
24. Yamauchi, H., Gumhold, S., Zayer, R., Seidel, H.-P.: Mesh segmentation driven by gaussian curvature. Vis. Comput. **21**(8–10), 659–668 (2005)
25. Zhou, F., De la Torre, F., Hodgins, J.K.: Hierarchical aligned cluster analysis for temporal clustering of human motion. IEEE Trans. Pattern Anal. Mach. Intell. **35**(3), 582–596 (2013)
26. Zhou, F., Torre, F., Hodgins, J. K.: Aligned cluster analysis for temporal segmentation of human motion. In: 8th IEEE International Conference on Automatic Face and Gesture Recognition, 2008. FG'08, IEEE, pp. 1–7 (2008)

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.

Gang Lei is an associate professor at the Jiangxi Normal University. He is also currently involved in a Ph.D. program in Jiangxi University of Finance and Economics. His research interests include data mining, educational informatics and objective orient techniques.

Yuanlong Cao received his Ph.D. degree from Beijing University of Posts and Telecommunications (BUPT) in Sep 2014, received his MS degree in software engineering from BUPT in July 2008, and received his BS degree in computer science and technology from Nanchang University, China, in July 2006. He worked as an intern/engineer in BEA TTC, IBM CDL, and DT Research (Beijing) from 2007 to 2011, respectively. He is a lecturer in the School of Software at Jiangxi Normal University, China. His research interests include multimedia communications and next generation Internet technology.

Qinghua Liu is now working as an experimentalist in the Software of School, Jiang Xi Normal University (JXNU). He received the BS degree in software engineering from JXNU, China, in 2007, and his Master degree in management science and engineering in 2011, from JXNU. His research interests include multimedia networking and next generation Internet technology.

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.