Animation Key-frame Extraction and Simplification Using Deformation Analysis

Tong-Yee Lee, Chao-Hung Lin, Yu-Shuen Wang, and Tai-Guang Chen

Abstract—Three-dimensional animating meshes have been widely used in the computer graphics and video game industries. Reducing the animating mesh complexity is a common way of overcoming the rendering limitation or network bandwidth. Thus, we present a compact representation for animating meshes based on novel key-frames extraction and animating mesh simplification approaches. In contrast to the general simplification and key-frames extraction approaches which are driven by geometry metrics, the proposed methods are based on a deformation analysis of animating mesh to preserve both the geometric features and motion characteristics. These two approaches can produce a very compact animation representation in spatial and temporal domains, and therefore, they can be beneficial in many applications such as progressive animation transmission and animation segmentation and transferring.

Index Terms—animating meshes, simplification, key-frame extraction, deformation analysis

1 Introduction

Recently, many researchers preferred representing animation using a sequence of deforming meshes (i.e., time-varying meshes) to satisfy the high-quality skin detail requirement [1, 2]. High-resolution deforming meshes with a great number of frames are required to represent pleasing and realistic animations. In this paper, we present novel methods for animation simplification and key-frame extraction. With the proposed approach, many redundant frames (i.e., those that can be interpolated by other frames) and perceptually meaningless animation data can be omitted or reduced, therefore greatly reducing the animation data requirement. Many important applications can also benefit from this novel approach, which will be demonstrated in this paper.

The aim of key-frame extraction is to search for good key-frames for reconstructing all other frames in an animation. The number of key-frames should be as small as possible and the quality of the reconstructed frames should be as high as possible. Generally, the animation key-frame extraction can be solved as a key-frame search optimization problem. However, the search space is large (i.e., large number of frames) and this search space is not perfectly smooth (i.e., the number of key-frames is not determined *a priori*). To solve these problems, we propose a deformation-driven genetic algorithm (GA) to efficiently search optimal key-frames from a sequence of animating meshes. In the literature, many mesh simplification approaches have been proposed to simplify static meshes such as [3, 4]. However, naïvely simplifying animating meshes using these methods cannot guarantee preserving both the geometric and the motion features of the animating meshes. These static techniques can guarantee simplifying a single frame with good geometric feature fidelity. Simplifying any key-frame in animating meshes may eliminate the important details required to represent other frames. To better simplify animating meshes, the proposed method will analyze the deforming information presented in the entire animating sequence.

The major contributions of the proposed methods for animation key-frame extraction and simplification are listed below:

- A deformation-driven **GA** is proposed to fast search good representative animation key-frames. These extracted key-frames can reconstruct other frames with good quality.
- A novel mesh simplification method based on animating mesh deformation information is proposed. With this new approach, the characteristics of animating meshes in both spatial and temporal domains can be successfully preserved. This method generates a sequence of simplified animating meshes with a static connectivity. Therefore, the annoying popping effect caused by changing connectivity/geometry is automatically avoided.

Many important applications such as animation segmentation and transferring can benefit from the proposed methods. Experimental evaluations of these applications are demonstrated and compared with other previous schemes.

2 Related Work

There have been many methods proposed for animation key-frame extraction and simplification. We describe only the most related schemes for animating mesh in this section.

2.1 Key-frame Extraction

Similar to video summarization [5, 6, 7], animation key-frame or key-posture extraction is the process of selecting

- *T.-Y. Lee, Y.-S. Wang, T.-G. Chen are with the Computer Science and Information Engineering Department, National Cheng-Kung University, Tainan, Taiwan (e-mail:tonylee@mail.ncku.edu.tw, braveheart@csie.ncku.edu.tw, taiguang@csie.ncku.edu.tw)*
- *C.-H. Lin is with the Geomatics Department, National Cheng-Kung University, Tainan, Taiwan. (e-mail:linhung@mail.ncku.edu.tw)*

meaningful frames from time-varying data. The related works can be categorized into *curve simplification*, *clustering*, and *matrix factorization* [8] based on what domain the key-frame extraction problem will be transferred. In [9], the animation data is considered as a trajectory curve in a high dimensional space. The key-frame extraction is converted into a curve simplification problem. In [10], the key-frame extraction becomes a clustering problem that attempts to group frames with similar posture. Once the key-frames are extracted, the remaining animation frames can be reconstructed by a linear combination of the extracted key-frames. In the category of matrix factorization [8], an animation is represented as a matrix formed by placing all the vertices of a frame in a row. This matrix is then approximately factorized into a weight matrix and a key-frame matrix. For example, in [8, 9, 10], the optimal solution to the key-frame search is obtained under similar constraints or objective functions. However, these approaches are very time-consuming and not efficient for practical applications. To the best of our knowledge, little research work has addressed this issue. There are two factors that make this problem more challenging. First, the high resolution animation sequence usually consists of a large number of frames (above 100 frames). Second, the optimal number of extracted key-frames cannot be determined *a priori*. Therefore, an efficient search approach is required. We introduce the deformation-driven GA to efficiently search good representative animation key-frames. Once the key-frames are extracted, similar to [8, 10], the animation is reconstructed by a linear combination of the extracted key-frames for better approximation.

2.2 Animation Mesh Simplification

The previous methods can be grouped into two categories: *static connectivity* and *dynamic connectivity* [11, 12]. Many researchers [11, 12, 13, 14] proposed adaptively changing the connectivity and dynamically improving the quality of the resulting simplified meshes. To create an appealing simplified mesh at each frame, a great amount of connectivity updating is required, thereby usually causing the unpleasant popping effect. These previous methods attempted to reduce this problem by maintaining both temporal and spatial coherence in updating the connectivity. However, the popping still could not be eliminated. In practice, a small number of popping effects will make an animation unpleasant. In the *static connectivity* category, several methods [15, 16] extend the standard static mesh simplification (i.e., QSlim algorithm [3]) to take all the frames of an animation into account when choosing a collapsing-edge candidate. All simplified animating meshes are represented by a single connectivity and therefore the popping effect caused by changing connectivity/geometry is automatically avoided. The saliency algorithm presented in [25] is used in mesh simplification [26]. In [26], the authors modify the QSlim algorithm by weighing the quadric error with a salient value. The purpose is to preserve the salient regions in the simplification process. This approach can be easily extended to simplify time-varying meshes using the approach presented in [15]. However, this approach may lose some geometric details due to the over–preservation of geometric features. The major challenges of simplifying time-varying meshes are maintaining both the geometric features and the motion characteristics of an animation. To solve these two difficulties, we incorporate the temporal information for the simplification of time-varying meshes. We select the collapsing-edge candidates using the metric of weighing the quadric geometrical error with the deformation degree. In this manner, the geometric features can be preserved by the quadric error metric, and the motion characteristics can be preserved by the deformation degree metric.

3 Deformation Analysis

Deformation analysis measures a triangle deformation degree which describes how significant motion behaves in this triangle in an animation. For this analysis, we measure the difference of the deformation gradient [1], called the deformation distance, between any two adjacent triangles. Recently, the deformation gradient has been applied to various computer graphic applications including deformation transferring [1], animating mesh segmentation [17, 18], and mesh deformation [19]. The deformation gradient is a non-translation affine transformation between two triangle orientation matrices. Basically, a deforming mesh with n frames is represented as $A = \{M^t\}, 1 \le t \le n$, where M^t represents the deforming mesh in the frame t and each M^t consists of many triangles called f_i^ts. Each face f_i^t consists of three vertices v_1^t, v_2^t, and v_3^t. The orientation matrix O_i^t of f_i^t can be computed by $\left[v_2^t - v_1^t \;\; v_3^t - v_1^t \;\; v_4^t - v_1^t\right]$, where $v_4^t = v_1^t + \left(v_2^t - v_1^t\right) \times \left(v_3^t - v_1^t\right) \Big/ \sqrt{\left|\left(v_2^t - v_1^t\right) \times \left(v_3^t - v_1^t\right)\right|}$. Similarly, the orientation matrix of the same face in the reference frame r denoted as f_i^r is $O_i^r = \left[v_2^r - v_1^r \;\; v_3^r - v_1^r \;\; v_4^r - v_1^r\right]$. Usually, the first frame is selected as the reference frame. The non-translation affine transformation, called deformation gradient D_i^t, between f_i^t and f_i^r can then be computed using:

$$D_i^t = O_i^t \left(O_i^r\right)^{-1} \tag{1}$$

The differential deformation gradient between two adjacent faces f_i^t and f_j^t at time frame t can be measured using the matrix subtraction $D_i^t - D_j^t$. We set the Frobenius norm of this differential deformation gradient as the deformation distance between these two faces, that is:

$$DD_{i,j}^t = \left\| D_i^t - D_j^t \right\|_F \tag{2}$$

We compute the deformation distances between the adjacent faces at each frame. The largest one denoted as $DD_{i,j}$ is chosen to represent the deformation degree between two adjacent faces among all the frames in an animation. To define the deformation degree of a face p, we simply average the deformation degrees between one face, f_i, and its neighboring faces, f_k, that is:

$$DDegree_{f_i} = \underset{f_k \in \text{Adjacency}(f_i)}{\text{avg}} \left(DD_{i,k}\right) \tag{3}$$

The deformation degree of f_i will be comparatively large if the region around this face has a very different motion. Otherwise, the deformation degree will be small if this region has a similar motion. Therefore, the larger face deformation degree of f_i implies a more important mo-

tion feature in the animation behaving on this face. We use the face deformation degree as a metric to simplify animating meshes and extract animation key-frames.

4 SIMPLIFYING DEFORMING MESH

The goal of our simplification approach is to simplify the deforming mesh under the minimizing animation distortion constraint. The proposed method is based on the QSlim algorithm [3] which introduces the quadric error metric (QEM) for collapsed-edge selection. The QSlim algorithm formulates the vertex distortion cost as the quadric distances from the vertex to its adjacent planes. This approach is originally designed to simplify a static mesh. To simplify a deforming mesh, we extend this distortion cost as the quadric distances from the vertex to its adjacent faces weighted by the face deformation degree described in Equation (3). Therefore, the distortion cost of vertex v^f at the time frame f is represented as follows:

$$d(v^f) = v^{f\,T}\left(\sum_{p\in planes(v^f)}\left(pp^T * DDegree_p\right)\right)v^f \tag{4}$$

where p represents the plane (face) adjacent to the vertex. Like [3], the vertex quadric Q_v^f is defined as the sum of the outer product of the face adjacent to vertex v^f multiplied by this face deformation degree.

$$Q_v^f = \sum_{p\in planes(v^f)}\left(pp^T * DDegree_p\right) \tag{5}$$

Initially, the distortion cost $d(v)$ will be zero because the adjacent faces pass through this vertex. When the two vertices v_1^f and v_2^f are contracted to a new vertex v_k^f, the quadric error of this new vertex conforms to the additive rule $Q_{v_k}^f = Q_{v1}^f + Q_{v2}^f$. Therefore, the edge-collapse cost of an edge (v_1^f, v_2^f) during the iterative contraction is formulated as follows:

$$\mathrm{cost}^f(v_1^f, v_2^f) = v_k^{f\,T}\left(Q_{v1}^f + Q_{v2}^f\right)v_k^f \tag{6}$$

Note that the new vertex v_k can be directly solved by optimizing the vertex position frame by frame under the constraint $\min \mathrm{cost}^f(v_1^f, v_2^f)$ as discussed in [3].

For a deforming mesh, the edge-collapse cost of an edge (v_1, v_2) can be simply formulated as the sum of edge-collapse cost at all time frames, that is:

$$Cost(v_1, v_2) = \sum_{f\in Frames} Cost^f(v_1^f, v_2^f) \tag{7}$$

As mentioned in Section 2, to avoid the popping effects caused by the connectivity dynamic update, our simplification method prefers a single connectivity to represent all animating meshes in the animation. As mentioned in [11; 12], when the mesh deformation is highly non-rigid, the static connectivity usually leads to bad simplification to certain frames. The measurement of the proposed edge-collapse cost considers both the geometric and the motion features. Therefore, the proposed method will not suffer from this problem. The experimental results will show that our method performs well for both rigid and non-rigid mesh animation.

5 ANIMATION KEY-FRAME EXTRACTION

In this section, we propose a deformation-driven GA method to fast-search animation key-frames.

5.1 Extracted Key-frames Encoding

For any search and learning method, the candidate solution encoding method is very vital to the success of a genetic algorithm. Binary encoding is the most common and simplest method. We use fixed-length and fixed-order binary coding, namely, a bit string, to represent a candidate key-frame sequence as a chromosome. Each frame is encoded as a single bit. The "1"-bit represents a key-frame and the "0"-bit represents a non-key-frame. In this manner, each key-frame sequence is represented as a bit string. The length of a bit string is equal to the number of animation frames and the encoding order is from left-to-right (i.e., from the first frame to the last frame). Figure 1 shows an example of a key-frame sequence encoding.

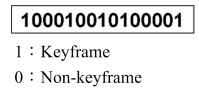


Fig. 1. An example of a key-frame sequence encoding.

5.2 Population Initialization

Theoretically, the initial population (i.e., bit strings) is chosen randomly in order to maintain a wide chromosome spectrum. However, from our experimental practice, this simple manner is not good enough in terms of the efficiency since an animation is usually a smooth sequence. The adjacent frames are similar. Therefore, the adjacent frames are usually not selected to be the key-frames at the same time. A more appropriate setting for an initial chromosome is a bit string with "1" bit regular interleaving, as shown in Figure 2(a). To keep a wide spectrum of the chromosomes while considering the algorithm efficiency, we set 80% of the chromosomes in the initial population to be chosen randomly and the remaining 20% of chromosomes set as bit strings with the "1" bit regularly interleaving. Figure 2(b) shows an example of our initial population. The population size is set at 1000 in our experiments.

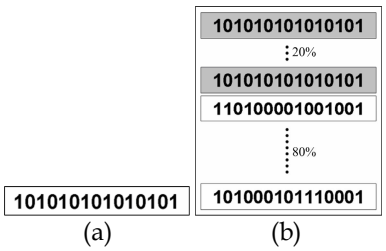


Fig. 2. (a) A chromosome with "1" bit regular interleaving (b) The initial population.

5.3 Fitness Function

The fitness function, or objective function, is used to evaluate the possibility of a chromosome for being the best key-frame extraction result. Each chromosome in the current population will be assigned a score (fitness) which represents how well that chromosome represents the key-frames of a given animation. From the key-frame extraction point of view, the number of extracted key-

frames must be as small as possible while the reconstructed animation error using these extracted key-frames must be minimized. Therefore, two factors are taken into account for key-frame extraction optimization: 1) the errors in the reconstructed animation and 2) the number of extracted key-frames. Therefore, the fitness function is defined by:

$$Fitness(T) = \frac{1}{\alpha \times Error(T) + (1-\alpha) \times Ratio(T)} \quad (8)$$

where the instance T represents a candidate solution and α is a user-defined weight coefficient. $Ratio(T)$ is the ratio of the number of selected key-frames over the number of animation frames.

In Equation (8), the error term $Error(T)$ is defined as the difference between the original animation and its corresponding reconstructed animation using extracted key-frames. To efficiently calculate this term, we simply adopt a liner interpolation to interpolate the non-key-frames from the extracted key-frames. We then use Equation (9) to calculate the error between two corresponding frames.

$$FrameError\left(frame^A, frame^B\right) = \sum_{p \in \text{Faces}} \left(\beta \times \left\| Deformation_{p^A, p^B} \right\|_F + (1-\beta) \times Distance_{p^A, p^B}\right) \quad (9)$$

where β is a user-defined weight coefficient, and $frame^A$ and $frame^B$ represent a frame in the original animation and its corresponding frame in the reconstructed animation, respectively. The error function *FrameError*() is factorized into two terms: *Deformation* and *Distance*. The *Deformation* term is to measure the Frobenius norm of the deformation gradient (i.e., calculated by Equation (1)), between a face p^A in $frame^A$ and its corresponding face p^B in $frame^B$. The *Distance* term calculates the displacement of these two corresponding faces. It simply averages the displacement of three vertices in the face. Because the fitness function is factored into two distinctive terms, the error term $Error(T)$ must be normalized. We assume that the worst key-frame extraction case, that is, the case with the maximum error, is only the first and last frames selected as key-frames. The intermediated frames are interpolated using only these two frames. In this situation, the reconstructed animation will have the maximum error. Under this assumption, the normalization process is calculated using:

$$Error = \frac{NonKeyframeError}{MaximumError} \quad (10)$$

where the *MaximumError* represents the worst case error and the *NonKeyframeError* represents the candidate solution error (i.e., extracted key-frames).

In Equation (8), the fitness function formed by two terms is a trade-off between the reconstruction error and the number of key-frames. The fewer the number of key-frames selected, the greater the generated reconstructed error. Therefore, the fitness function is designed as a weighted combination of these two terms. The significance of these two terms can be controlled by a user-defined parameter α.

5.4 Selection

After the encoding and the initial population are decided, the next step is to choose a chromosome in the population that will create offspring for the next generation. The purpose of selection is to emphasize the fitter chromosome in the population in hopes that their offspring will in turn have higher fitness. In this paper, we adopt the simplest and most popular one, that is, fitness-proportionate selection. The expected value of a chromosome (i.e., the expected number of times a chromosome will be selected to reproduce), is that chromosome's fitness calculated by the fitness function. In other words, a chromosome with a higher fitness value has a higher probability of being selected.

5.5 Genetic Operators

Once the parent chromosomes are selected, we use two evolution strategies to generate offspring for the next generation: *Crossover* and *Mutation*. These two operators are described in detail as follows.

***Crossover*:** A two-point crossover approach is adopted here. Two loci are randomly chosen and the segments between them in the parent chromosomes are then exchanged to form two offspring. Figure 3(a) shows an example of this operation.

***Mutation*:** We propose two mutation approaches. The first approach randomly chooses the mutation position and then exchanges the bit value (i.e., "1" changes into "0" or "0" changes into "1"), as shown in Figure 3(b). The second approach is that we first randomly select a "1" bit as a mutation locus and then shift this locus right or left one bit to generate the offspring, as shown in Figure 3(c). The shift direction is also randomly decided. The purpose of this design is to perform locus position fine-turning. A fitter key-frame may be located around a candidate key-frame. This mutation approach can speed up algorithm convergence.

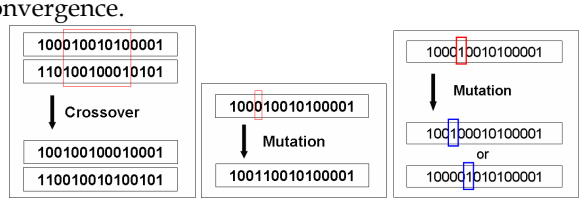


Fig. 3. Two GA operators: Crossover and Mutation.

How the mutation and crossover rates are set is very important. In our experiments, the crossover probability is set to 95% and the mutation probability is set to 5%. In terms of mutation probability, the probability of the random selection is set to 97% (the first approach) and the probability of locus shifting is set to 3% (the second approach).

5.6 Termination Condition

A genetic algorithm is usually executed in an iterative manner. In general, in the last generation, there are many highly fit chromosomes in the population. Therefore, the termination condition is when 50% of the chromosomes in the population have the same fitness, the fittest one is the same during the last 100 iterations.

5.7 Animation Reconstruction

Once key-frames are extracted from an animation, we can

construct an approximate animation using these key-frames. In the paper, we adopt Huang et al.'s approach [8] to perform reconstruction. The basic idea is that each frame of an animation is regarded as a linear combination of extracted key-frames. Therefore, each frame is approximated by linearly blending the key-frames as defined in Eq. (11):

$$F_{orig}^{i} \approx \sum_{j=1}^{m} w_{ij} F_{key}^{j}, \ for\ i = 1 \ldots n \tag{11}$$

where w_{ij} is the weight of the j-th keyframe (F_{key}^{j}) for reconstructing i-th frame (F_{ori}^{i}), n represents the number of frames, and m represents the number of key-frames.

The quality of animation reconstructed by all key-frames is better than that reconstructed by local key-frames. The animation reconstruction space spanned by local key-frames is only a subspace of that spanned by all key-frames. Therefore, the frames interpolated by local key-frames also can be blended by all key-frames, but not vice versa. However, the global support may result in temporal aliasing. Fortunately, this problem can be greatly reduced by optimizing the blending weights in the purpose of minimizing the reconstruction distortion. To obtain optimum weights w_{ij}, we minimize the following equation:

$$\arg\min_{w_{ij}} \left\| F_{orig}^{i} - \sum_{j=1}^{m} w_{ij} F_{key}^{j} \right\|^{2}, \ for\ i = 1 \ldots n \tag{12}$$

Therefore, an animation sequence with n frames can be represented by m key-frames and its corresponding weights for all frames, that is:

$$\left\{ \left(F_{key}^{1}, w_{11}, \cdots, w_{n1}\right), \left(F_{key}^{2}, w_{12}, \cdots, w_{1n}\right), \cdots, \left(F_{key}^{m}, w_{1m}, \cdots, w_{nm}\right) \right\} \tag{13}$$

In the key-frame extraction optimization process, we simply use linear interpolation to construct an animation. The reconstructed result using linear interpolation is good enough to evaluate the fitness value. Usually, the reconstruction quality using [8] is not significantly better (i.e., varying from 3% to 12%) than that using simple linear interpolation. However, the computational cost is much higher than that in linear interpolation. The reconstruction is performed frequently in the optimization process. It is too time-consuming to frequently reconstruct a candidate solution using [8]. To take efficiency into account, we simply use linear interpolation in the optimization step. Then, once the key-frames are extracted, we use [8] to reconstruct the final animation for better quality. The number of key-frames is determined automatically in the proposed approach. We extract fewer key-frames in a near-rigid animation and extract more key-frames in an extremely non-rigid animation. If the input animation has extreme deformation between frames (it implies that each frame is a key-frame), we need another morphing/interpolation technique to interpolate more in-between frames for a more smooth animation. This problem is not part of the scope of this paper.

6 Experimental Results and Discussions

The various animation sequences used in the experiment are shown in Table 1. To demonstrate the robustness and feasibility of the proposed animation key-frame extraction and simplification methods, both rigid-body and soft-body animation sequences were experimentally evaluated. The proposed methods are based on the deformation analysis. Therefore, we start with the visualization of the deformation analysis of various animation sequences in Figure 4. Obviously, the faces lying around the joints have a larger deformation degree (green color). This means that these faces are significant motion features and must be preserved in the simplification process.

TABLE 1
Various experimental animation sequences.

Animation sequence	Rigid/Soft Body	#Vertex	#Face	#Frame
Runner	rigid-body	7502	15000	25
Dancer	rigid-body	7061	14118	201
Chicken	rigid-body	3030	5664	400
Homer	rigid-body	5103	10202	25
Horse-Gallop	rigid-body	8431	16843	48
Horse-Collapse	soft-body	8431	16843	53
Cow	soft-body	2904	5804	204

Fig. 4. The deformation analysis of various deforming mesh including Runner, Dancer, Cow, Horse-Collapse, Horse-Gallop, and Chicken. The deformation degree is coded by color ranging from red (the largest degree) to blue (the smallest degree).

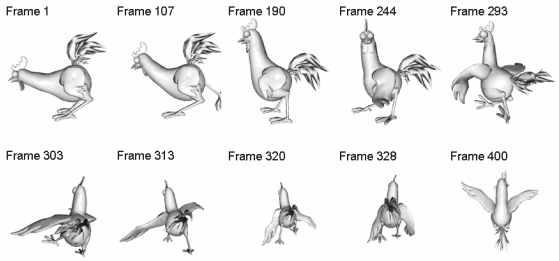


Fig. 5. The selected key-frames of Chicken animation.

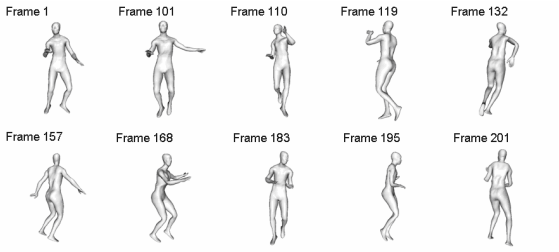


Fig. 6. The selected key-frames of Dancer animation.

The experimental results were evaluated on an Intel Pentium 4 2.4G with 1G memory PC. The configuration of all parameters in the genetic algorithm was identical in all experiments (population size: 1000; crossover probability: 95%; mutation probability: 5%; terminate condition: 50% same chromosomes in the population and the fittest

chromosome is unchanged during 100 iterations). These parameters were obtained from experiments (attempt to find the most suitable parameter setting). Some key-frame extraction results are shown in Figures 5 and 6.

To evaluate the performance of the proposed key-frame extraction approach we compare it using only a state-of-the-art method: KeyProbe [8] as follows. As for the other well-known approaches such as Curve Simplification [9], K-mean [20], ICA [21], and PCA [22; 23], a detailed discussion and comparison with [8] are provided in [8]. We adopted the commonly used peak signal-to-noise ratio (PSNR) to measure the reconstruction distortion. Similar to [8], we restrict the number of extracted key-frames to 10, 20, 30, 40, and 50 in the extraction process; that is, parameter α is set to 1.0 in Equation (8). For these selected cases, we calculate the minimum, maximum, and average PSNR over all the frames for each case (shown in Table 2). We can see that our approach is very computationally efficient in comparison with [8] while the quality is comparable to the KeyProbe (i.e., slightly better than the KeyProbe for the maximum and minimum PSN, but slightly worse than the KeyProbe for the average PSNR). The execution performance of our approach depends very much on the criterion for genetic algorithm success (i.e., the number of generations). The execution performance of KeyProbe mainly depends on the number of frames. In the case of chicken animation with 400 frames, the KeyProbe takes more than 12 hours to extract 50 key-frames. Our approach only takes about half an hour. Therefore, our technique is a better approach than the time-consuming KeyProbe to extract an animation with a large number of frames.

TABLE 2

The performance comparison between the proposed method and KeyProbe [8]. 1st column: the input animation; 2nd column: the number of extracted key-frames (#KF); 3rd~7th columns: the performance of our approach; 8th~11th columns: the performance of KeyProbe; Min. Max and Avg. PSNR represents the minimum, maximum and average PSNR over all frames, respectively; #Gen. represents the number of generations in GA.

Model	# KF	Our approach					KeyProbe [8]			
		Min. PSNR	Max. PSNR	Avg. PSNR	# Gen.	Time (min.)	Min. PSNR	Max. PSNR	Avg. PSNR	Time (min.)
Chicken	50	43.60	131.45	64.75	12238	24.78	40.82	94.79	66.24	822.77
	40	39.75	127.04	59.35	9450	20.83	39.48	86.50	60.14	594.79
	30	32.98	126.46	54.62	4578	11.85	34.85	84.71	55.25	404.08
	20	28.48	127.72	46.67	8180	24.46	29.58	77.14	47.63	150.33
	10	21.58	135.80	40.06	2302	18.72	22.54	66.97	39.64	35.15
Dancer	50	49.89	69.30	62.14	5808	14.76	49.65	77.28	62.83	626.77
	40	46.27	65.45	56.99	9135	23.16	44.71	71.96	57.85	453.65
	30	41.99	62.14	50.78	4414	17.97	40.17	61.74	51.69	251.06
	20	36.65	54.71	43.41	4210	30.93	36.55	57.28	44.36	112.74
	10	24.50	48.05	33.23	1702	9.12	27.27	48.71	34.87	28.08

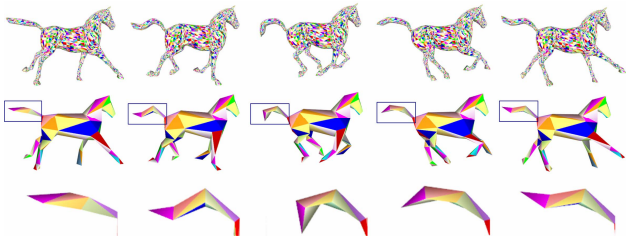


Fig. 7. The deforming mesh simplification. The deforming mesh is simplified from 8431 vertices (original) to 84 (1%) vertices.

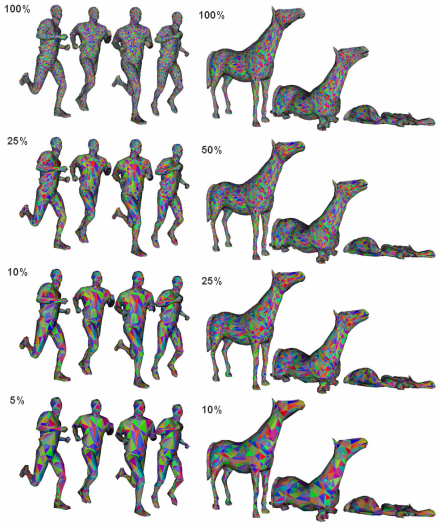


Fig. 8. The deforming mesh simplification.

For animation simplification, we first demonstrate the ability of motion feature preservation in Figure 7. The proposed method can successfully preserve the tail motion details, as shown in Figure 7 (the blue quadrilateral), while preserving the geometric appearance. This is because the faces around the joints of the tail have a larger deformation degree. They are recognized as significant motion features and are assigned a lower priority for the collapsed-edge selection. Therefore, the proposed method can preserve both the geometry and motion features in the simplification process. Figure 8 shows the other experimental results. In Figure 8-left, the runner animation is simplified from 7502 vertices (original) to 1875 (25%), to 750 (10%), and finally to 375 (5%) vertices. From this figure, the proposed approach can preserve an animation appearance of both geometry and motion, even in the most simplified case (5%). In Figure 8-right, the collapsing horse animation (soft-body) simplification is demonstrated. In this extreme, highly non-rigid deformation example, the proposed method can still preserve both the geometric and motion details well in the simplification process. The problem incurred by single connectivity will lead to bad approximations in certain frames, as mentioned in [11, 12], frames which did not occur using the proposed method. For the dynamic connectivity approaches such as [11, 12], they attempt to change the connectivity to better fit the deforming surface and produce high quality approximation on each frame. However, the deforming meshes with non-consistent connectivity will cause unpleasant popping effects. Even though the temporal coherence is managed, the popping effect cannot be avoided. Therefore, the simplified animation results generated by the proposed method are more smooth and pleasing than that generated by [11, 12]. It implies that the

least geometrical error does not mean it is the best solution. If the overall geometrical error is higher for an obtained simplification, it can be a good solution when the motion information is taken into account. For visual comparison, please see our accompanying video (http://graphics.csie.ncku.edu.tw/Paper_Video/TCSVT/TCSVT_anim_mesh_2007.mp4). We also show the quality and visual comparisons between our approach and several related approaches [15, 26] in Figures 9 and 10. We extend and apply the mesh saliency approach presented in [26] to the animating mesh simplification by taking the sum of edge-collapsing cost in all frames as the simplification metric (Figure 10(f)). We also slightly modify the simplification metric presented in [15] (Figure 10(e)) for the maximum edge-collapsing cost over all frames (Figure 10(d)). We can see that the proposed method has better quality than these works, in terms of shape preservation and PSNR, on parts of the animating mesh that deform

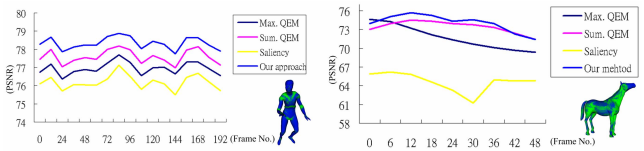
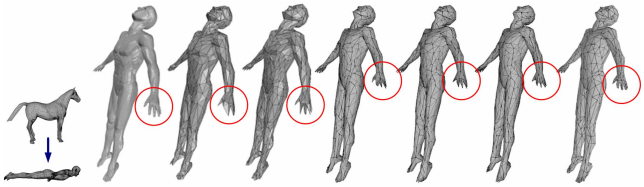


Fig. 9. Quality comparison between the Max. QEM, Sum. QEM [15], Saliency [26] and the proposed method.



(a) (b) (c) (d) (e) (f) (g)

Fig. 10. The visual comparisons for a horse-to-man animation. (a) The original animation (1st frame); (b) [11]; (c) [12] (d) Max. QEM; (e) Sum. QEM [15]; (f) saliency [26]; (g) our method. Note that (a) ~ (c) are cited from [12].

relatively more as compared to non-feature parts of the mesh. Please see our accompanying video for visual comparison.

7 Applications

Many applications can benefit from the proposed methods. In this paper, we demonstrate two applications and their evaluated results: 1) deforming mesh decomposition and 2) animation compression and compact progressive representation for transmission.

Efficient Deforming Mesh Decomposition

Deforming mesh decomposition plays a critical role in computer graphics. Recently, many schemes [18, 24] concentrated on automatically partitioning the deforming mesh. However, these approaches are very time consuming due to the all-pairs shortest path finding and iterative face clustering requirements. Based on the proposed simplification scheme, the decomposition computation cost can be greatly reduced. We briefly introduce the workflow of our partitioning scheme as follows. First, we generate the coarser deforming mesh using the simplification approach mentioned in Section 4. Any deforming mesh decomposition approach can then be adopted here to decompose this simplified mesh. In this work we use our previous deforming mesh partitioning method [18]. The next step is to map the decomposition boundary back to the original deforming mesh using projection. A common boundary smooth process is then applied to smooth out the decomposition boundaries. Figure 11 shows the experimental results. The success of [18] is significantly related to deformation behaved by animating meshes. After simplification using the proposed method, the simplified animating meshes can still preserve motion features in the original meshes. Therefore, the proposed method can create good partitioning results based on simplified animating meshes. Table 3 shows the performance comparison between [18] and the proposed scheme. Obviously, the computation cost is reduced using the proposed scheme.

As mentioned in [18], the deforming mesh decomposition result can be applied to the deformation transfer application [1]. Using the proposed decomposition scheme, we can also effectively handle the deformation transfer with difference reference poses. Figure 12 shows the experimental result. The reference poses are different in the

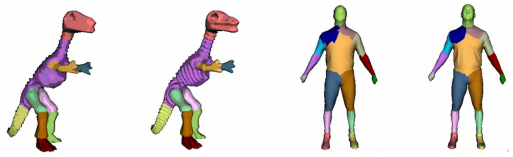


Fig. 11. The partitioned results of Dinosaur and Runner deforming meshes.

TABLE 3

The performance comparison between the proposed approach and Lee et al.'s approach [18].

		Simplification	Deformation Analysis	Segmentation	Mapping	Total Time
Dinosaur	[18]	--	163 sec.	878 sec.	--	1041 sec.
	The proposed method	301 sec.	12 sec.	12 sec.	1 sec.	326 sec.
Runner	[18]	--	79 sec.	629 sec.	--	708 sec.
	The proposed method	165 sec.	12 sec.	8 sec.	1 sec.	186 sec.

hands (the red ellipses in Figure 12-Left).

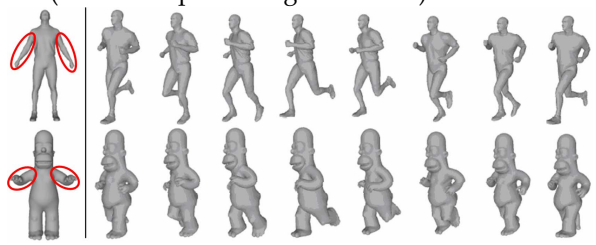


Fig. 12. Left: the reference pose; Top right: the source animation; Bottom right: the deformation transfer result generated using the proposed method.

Animation Compression and Compact Progressive Representation

In computer graphics, reducing the data complexity is a common way to overcome the rendering limitation or

network bandwidth. Both the proposed key-frame extraction and simplification methods can reduce the data complexity of a given animation sequence. The extracted key-frames can be regarded as a compact representation in the temporal domain, as well as a simplified animation which can be regarded as a compact representation in the spatial domain. With the combination of these two approaches, the animation data size can be greatly reduced, as shown in Figure 13. Furthermore, we also can construct a compact progressive animation representation for network transmission. Using the simplification approach, an animation can be represented as a progressive animation representation [4], that is, a base animation $\left(M_0^0,\cdots M_n^0\right)$and a sequence of vertex split records $\left(vspl_0^0,\cdots vspl_n^0\right),\cdots,\left(vspl_0^k,\cdots vspl_n^k\right)$. With the extracted key-frames, we can replace the base animation using extracted key-frames and a set of weights for non-key-frames reconstruction. We can only store the vertex split records for key-frames. Therefore, the progressive animation representation is very compact as follows:

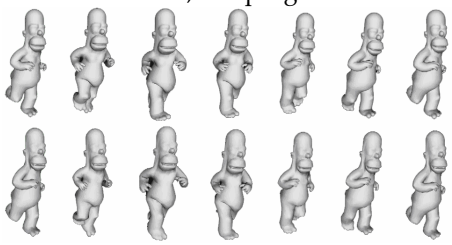


Fig. 13. Top row: the original animation (no. of face: 10202 no. of frames: 25). Bottom row: the animation generated by the proposed simplification and key-frame extraction approaches (no. of face: 3062; no. of key-frames: 6).

$$\left\{\left(M_{key0}^0,\cdots,M_{keym}^0,\left(w_{ij}\right)\right),\left(vspl_{key0}^0,\cdots vspl_{keym}^0\right),\cdots,\left(vspl_{key0}^k,\cdots vspl_{keym}^k\right)\right\} \quad (14)$$

8 Conclusions

We have introduced novel methods for animation mesh simplification and key-frame extraction to produce a very compact representation for animating meshes. The proposed approaches are driven not only by the common geometric error analysis, but also by the deformation analysis of animating meshes. Therefore, the geometric features and the motion characteristics can be well-preserved in the simplified meshes, even for an extremely non-rigid animation. In addition, the simplified results will not have any annoying popping caused by changing connectivity/geometry, because single connectivity is used to represent all frames. In key-frame extraction, the deformation-driven GA method is proposed to solve the optimization search problem. The experimental results show that this new method is very efficient in searching for good key-frames while the quality is comparable to that of a state-of-the-art technique [8]. We also experimentally demonstrated the usefulness of the proposed methods to the relative applications: deforming mesh decomposition and compact progressive representation for animation transmission. The experimental results show that the proposed method can be successfully applied to these two applications.

Acknowledgment

This paper is supported by the National Science Council, Taiwan, Republic of China, under contract No. NSC-95-2221-E-006-368, NSC-95-2221-E-006-193-MY2, NSC-96-2628-E-006-200-MY3 and NSC-96-2221-E-006-312-MY2. Many thanks for anonymous reviwers' helpful comments to improve our paper.

References

[1] Sumner, R.W. and Popovic, J.: Deformation transfer for triangle meshes. ACM Trans. Graph. 23(3), pp. 399–405, 2004.

[2] Sumner, R. W., Zwicker, M., Gotsman, C. and Popovic, J.: Mesh-Based Inverse Kinematics, ACM Transactions on Graphics (SIGGRAPH 2005), 24(3), pp. 488-495, August 2005.

[3] Garland, M. and Heckbert, P. S.: Surface simplification using quadric error metrics. In Proc. SIGGRAPH 97, pp. 209-216, 1997.

[4] Hoppe, H.: Progressive meshes. In Proceedings of SIGGRAPH 96, pp. 99-108, 1996.

[5] Dimitrova, N., Zhang, H.-J., Shahraray B., Sezan I., Huang, T. and Zakhor, A.: Applications of video-content analysis and retrieval, IEEE Multimedia, Vol.9, Issue 3, July-Sept. pp. 42 – 55, 2002.

[6] Gong, Y. and Liu, X.: Video summarization and Retrieval using Singular Value Decomposition. *ACM Multimedia Systems Journal*, 9(2), pp 157-168, August 2003.

[7] Uchihashi, S., Foote, J., Girgensohn, A., Boreczky, J.: Video manga: generating semantically meaningful video summaries. In: Proceedings of the 7th ACM International Conference on Multimedia (Part 1), pp. 383–392. ACM Press, New York 1999.

[8] Huang, K.S., Chang, C.F., Hsu, Y.Y. and Yang, S.N.: Key Probe: A Technique for Animation Keyframe Extraction. The Visual Computer, Vol. 21, No. 8-10, pp. 532-541, 2005.

[9] Lim, I.S. and Thalmann, D.: A key-posture extraction out of human motion data. In: Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC 2001), pp. 1167-1169, 2001.

[10] Park, M. J. and Shin, S.Y.: Example-based motion cloning. Comput. Animat. Virtual Worlds 15(3-4), pp. 245-257, 2004.

[11] Kircher, S. and Garland, M.: Progressive Multiresolution Meshes for Deforming Surfaces. ACM/Eurographics Symposium on Computer Animation, pp. 191–200, 2005.

[12] Huang, F.-C., Chen, B.-Y. and Chuang, Y.-Y.: Progressive Deforming Meshes based on Deformation Oriented Decimation and Dynamic Connectivity Updating. Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2006 (SCA06), pp. 53 - 62, Vienna, Austria, 2006.

[13] Shamir, A. and Pascucci A.: Temporal and spatial level of details for dynamic meshes. In proceedings of the ACM Symposium on Virtual Reality Software and Technology, pp. 77-84, 2001.

[14] Shamir, A., Passucci, V. and Bajaj C.: Multiresolution dynamic meshes with arbitrary deformations. In Proc. Visualization '00, IEEE Computer Society Press, pp. 423–430, 2000.

[15] Mohr, A. and Gleicher, M.: Deformation sensitive decimation. technical report. Rep. 4/7/2003, University of Wisconsin, Madison.

[16] Decoro, C. and Rusinkiewicz, S.: Pose-independent simplification of articulated meshes. Proceedings of the 2005 symposium on Interactive 3D graphics and games, Washington, District of

Columbia, pp. 17-24, April 2005.

[17] James, D. L. and Twigg, C. D.: Skinning mesh animations. ACM Transactions on Graphics (SIGGRAPH 2005), 24(3), August 2005.

[18] Lee, T.Y., Wang, Y.S. and Chen, T.G.: Segmenting a Deforming Mesh into Near-Rigid Components. In Pacific Graphics 2006, Vol. 22, Issue 9, pp. 729 – 739, September 2006.

[19] Popa, T., Julius, D. and Sheffer, A.: Material aware mesh deformations. IEEE International Conference on Shape Modeling and Application, June 2006.

[20] Liu, F., Zhuang, Y., Wu, F. and Pan, Y.: 3D motion retrieval with motion index tree. Comput. Vis. Image Understand. 92(2). pp. 265-284, 2003.

[21] Hyvarinen, A., Karhunen, J. and Oja, E.: Independent Component Analysis. Wiley, New York, 2001.

[22] Jolliffe, I. T.: Principal Component Analysis. Springer, Berlin Heidelberg New York, 2002.

[23] Alexa, M. and Muller, W.: Representing animations by principal components. Computer Graphics Forum, 19(3):411--418, August 2000.

[24] Lee, T.Y., Lin P.H., Yan S.U. and Lin C.H.: Mesh decomposition using motion information from animation sequences. In Computer Animation and Social Agents 2005, Vol. 16, issue 3-4, pp. 519 – 529, July 2005.

[25] Itti, L., Koch, C. and Niebur E.: A model of saliency based visual attention for rapid scene analysis. IEEE Trans. Pattern Analysis and machine Intelligence, Vol. 20, No. 11, pp. 1254-1259, 1998.

[26] Lee, C. H., Varshney, A. and Jacobs, D. W.: Mesh saliency. ACM Trans.on Graphics,24(3):659-666, 2005.

Prof. Tong-Yee Lee (Tony) received his PhD in computer engineering from Washington State University, Pullman, in May 1995. Now, he is a Professor in the department of Computer Science and Information Engineering at National Cheng-Kung University in Tainan, Taiwan, Republic of China. He serves as an associate editor for IEEE Transactions on Information Technology in Biomedicine from 2000 to 2007. He is also on the Editorial Advisory Board of Journal Recent Patents on Engineering, editor for Journal of Information Science and Engineering and region editor for Journal of Software Engineering. His current research interests include computer graphics, non-photorealistic rendering, image-based rendering, visualization, virtual reality, surgical simulation, medical visualization and medical system, distributed & collaborative virtual environment. He leads a Computer Graphics Group/Visual System Lab at National Cheng-Kung University (http://graphics.csie.ncku.edu.tw/). He is a member of the IEEE and ACM.

Chao-Hung Lin was born in Koushung, Taiwan, Republic of China, in 1973. He received his B.S in computer science/engineering from Fu-Jen University, M.S. and Ph.D. in computer engineering from National Cheng-Kung University, Taiwan in 1997, 1998 and 2004, respectively. Now, he is an Assistant Professor in the department of geomatics at National Cheng-Kung University in Tainan, Taiwan. His current research interests include computer graphics, image processing, visualization and modeling.

Yu-Shuen Wang received his B.S. from National Cheng Kung University, Tainan, Taiwan, in 2004. Currently, he is a Ph.D. candidate in the Department of Computer Science and Information Engineering at National Cheng Kung University. His research interests include computer graphics, mesh segmentation, skeletonization and computer animation.

Tai-Guang Chen received his B.S. from National Sun Yat-sen University, Kaohsiung, Taiwan, in 2004. Currently, he is a graduate student in the Department of Computer Science and Information Engineering at National Cheng Kung University. His research interests include computer graphics, mesh segmentation, and computer animation.