The Visual Computer manuscript No.
(will be inserted by the editor)

Tong-Yee Lee · Yu-Shuen Wang · Tai-Guang Chen

Segmenting a Deforming Mesh into Near-Rigid Components

Abstract Given a deforming mesh in an animation, we propose a new method to segment this mesh into several near-rigid sub-meshes. From this deforming mesh over all frames of an animation, we can analyze the degree of deformation between two nearby faces on the mesh. Then, our algorithm partitions the given deforming mesh into near-rigid components where the segmentation boundaries always pass at regions of large deformation. As a result, the mesh segmentation is invariant to all frames of the given animation and the motion of faces in each near-rigid-component can be represented by the same approximate affine transformation. To demonstrate the usefulness of the algorithm, we solve the restriction of deformation transfer for triangle meshes [31] which requires similar reference poses between source mesh and target mesh.

Keywords near-rigid sub-meshes · mesh segmentation · deformation transfer

1 Introduction

1.1 Background

Mesh segmentation plays an important role in computer graphics for applications such as metamorphosis [10,17,18,22,30,33], texture mapping [12,20], compression [14,19], simplification [8], collision detection [21], and skeleton extraction [16]. Different applications may require different mesh segmentations due to their different requirements. In general, most previous studies can be classified into two categories: patch-based and part-based segmentation methods [29]. The patch-based methods usually segment a mesh into disk-like patches for applications such as texture mapping [12,20] and parameterization [10]. On the other hand, the part-based methods partition a mesh into several so-called meaningful components [1,2,15,16,25] for applications such as modeling by example [7] and 3D model retrieval [34].

Tong-Yee Lee
No. 1, Ta-Hsueh Road, Tainan,Taiwan, ROC.
Tel.: +886-6-2757575 Ext.62531
Fax: +886-6-2747076
E-mail: tonylee@mail.ncku.edu.tw

Yu-Shuen Wang
E-mail: braveheart@csie.ncku.edu.tw

Tai-Guang Chen
E-mail: hf@csie.ncku.edu.tw

1.2 Related Work

Using different partitioning metrics, many automatic approaches have been proposed such as region growing [13], hierarchical clustering [9], iterative clustering [23,27], spectral clustering [5], fuzzy clustering [16] and approximate convex decomposition [21]. Among them, many distance metrics such as geodesic distance and angular distance between triangles are used to partition static meshes. Large distance usually means the barrier exists between partitions, thereby cutting the given mesh into sub-meshes at regions with large distance.

Most previous methods concentrate on segmenting static meshes. Recently, there are several methods working on segmenting non-static (e.g. deforming) meshes [11,15,19,28]. In [15], the approach transforms the original mesh into a multi-dimensional space so that the sub-meshes are similar in different poses. In [11], the mean shift clustering of rotation sequence is used to identify the near-rigid structure of the deforming mesh and the near-rigid structure is considered as a skeleton of the original animation to obtain the bone skin weights. However, in [11], there is no attempt to separate the mesh apart clearly; faces in deformable regions are not clustered into any partition. Furthermore, even some faces in the same partition are not connected. Therefore, [11] can not benefit too much for some applications with the requirement of the faces to be connected in the same partition. Both [19,28] analyze the motion for each vertex and

cluster the vertices with similar motion, thereby partitioning meshes into sub-parts. Both methods compress each partition independently in their animation compression applications. However, from their results, the cutting boundary may deviate from the deformable regions. Therefore, the near-rigid partitions are not always obtained.

1.3 Contributions

In this paper, we propose a new method to segment a deforming mesh into several near-rigid sub-meshes. We describe a new distance metric based on the degree of the deformation between two nearby triangles over all frames of an animation. In this paper, a region with larger distance behaves with more deformation than areas with smaller distance. The proposed algorithm sets segmentation boundaries at regions with more deformation in order to obtain near-rigid sub-meshes. These near-rigid sub-meshes are potentially helpful to many applications such as LOD of an animation mesh [11] and deformation transfer [31].

In [31], there is a very useful approach proposed to transfer the existing deformation from the source mesh to the target mesh. However, this approach requires the reference poses of both source and target meshes to be similar. Otherwise, the transferred results can be very unpleasant. Since our segmentation method can segment a deforming mesh into near-rigid sub-meshes, we can determine the rigid transformation of each near-rigid sub-mesh; thereby adjusting the reference poses of both source mesh and target mesh to be similar. The deformation transferred from source mesh to target mesh with variant reference poses is used to demonstrate the usefulness of the proposed algorithm.

2 The Distance Metric for Deforming Mesh

We propose a distance metric based on the deformation gradient [31] of two adjacent faces in an animation sequence. In this paper, this metric is called deformation distance. A very recent work [26] in mesh editing application also uses deformation gradient to extract material properties including rigid and non-rigid information. Deformation gradient is the non-translation portion of an affine transformation which can represent the change between the reference pose and the aimed pose, including orientation, scale and skew. Let v_i and $\tilde{v}_i, i \in 1\ldots3$, be the vertices of two triangles, and Sumner et al. [31] add the fourth vertex v_4 in the direction perpendicular to the triangle, where

$$v_4 = v_1 + \frac{(v_2 - v_1) \times (v_3 - v_1)}{\sqrt{(v_2 - v_1) \times (v_3 - v_1)}} \quad (1)$$

Therefore, the non-translation transformation (i.e., deformation gradient) between these two triangles can be obtained by $M = \tilde{V}V^{-1}$, where

$$V = [v_2 - v_1 \; v_3 - v_1 \; v_4 - v_1] \quad (2)$$

$$\tilde{V} = [\tilde{v}_2 - \tilde{v}_1 \; \tilde{v}_3 - \tilde{v}_1 \; \tilde{v}_4 - \tilde{v}_1] \quad (3)$$

Thus, for each aimed pose t in the animation sequence, we can determine the deformation gradient of each face f_i, because there exists a deformation transformation M_i^t between f_i^r and f_i^t, where f_i^r and f_i^t denote f_i in reference and aimed poses, respectively. The difference of the deformation gradient between any two adjacent faces, f_i and f_j can be obtained by a matrix subtraction $M_i^t - M_j^t$. We set the Frobenius norm of this difference to be the deformation distance between these two faces. In addition, our algorithm normalizes this distance by dividing it by the geodesic distance $Geod(f_i, f_j)$, where $Geod(f_i, f_j)$ is the distance between the centroid of adjacent faces, f_i and f_j. This is due to the following observation in Figure 1. In this figure, when two similar discrete lines are bent into curves, the one with fewer vertices has larger rotation angles for adjacent edges marked by circles. Therefore, we should take this factor into account and make deformation of the smaller triangles as obvious as possible by geodesic length normalization. This is especially true in deformable regions which usually have smaller faces than other regions. Later in Section 3.2, this observation is very useful to determine if two adjacent faces are deformable or not. The normalization of deformation distance can deal with both cases well in Figure 1.

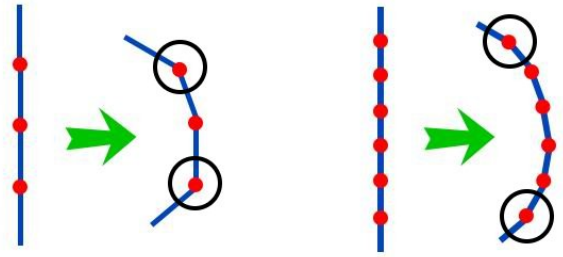

Fig. 1 The right curve with more vertices has smaller rotation angle between adjacent edges.

Our algorithm computes the deformation distance between adjacent faces i and j in each frame, and chooses the largest one denoted as D_{ij}. D_{ij} is used to evaluate the degree of deformation between these two faces among all frames of the given animation. Therefore, the proposed distance metric for the deformation of two adjacent faces is formulated as:

$$D_{ij} = max(\frac{\| M_i^t - M_j^t \|_F^2}{Geod(f_i, f_j)}) \; \forall t \in T \quad (4)$$

Where T is a sequence of animation poses. The distance between two adjacent faces will be comparatively small if both faces have similar deformation gradients. Otherwise, the distance becomes large. Barriers are formed at the place which has a large deformation, and these barriers separate the faces apart on the two sides of them. This is the reason why the

proposed algorithm can partition the mesh at regions with large deformation.

3 Segmentation Algorithm

The proposed segmentation method is composed of the following steps.

1. Generate a dual graph for the given deforming mesh in Section 3.1.
2. Select feature faces and determine to which partition it belongs in Section 3.2
3. Execute face clustering process to partition the mesh in Section 3.3
4. Smooth segmentation boundaries in Section 3.4

More details about each algorithm will be described in the following Subsections.

3.1 Dual Graph Generation

In [16], a dual graph of the mesh is generated to help mesh segmentation. Our algorithm builds a dual graph for the given deforming mesh, too. In this algorithm, the distance metric for the dual graph is computed by the combination of geodesic distance and deformation distance. Since the distance between two nearby faces is obtained, we apply all-pairs shortest path algorithm to find out the distance between any pair of faces on the mesh. The formal expression is shown below:

$$Dis(f_i, f_j) = \delta \frac{Geod(f_i, f_j)}{avg(Geod)} + (1-\delta)\frac{Deform(f_i, f_j)}{avg(Deform)} \quad (5)$$

Where f_i and f_j are any pair of faces on the mesh, $Geod(f_i, f_j)$ and $Deform(f_i, f_j)$ are the geodesic distance and deformation distance between face i and face j, respectively, and δ is the user-specified weight which usually ranges from 0.1 to 0.2 in our experimental results.

3.2 Automatic Feature Faces Extraction

In this step, the algorithm automatically extracts several feature faces on the mesh to represent the initial partitions. Then, the algorithm assigns the remaining faces to a certain partition in the face clustering step. The purpose of the algorithm is to partition a deforming mesh into near-rigid sub-meshes. Therefore, on each sub-mesh, the part that deforms the most tends to lie near its boundary. In other words, the feature faces to be extracted are the triangles far from these boundary (i.e., deformable) regions.

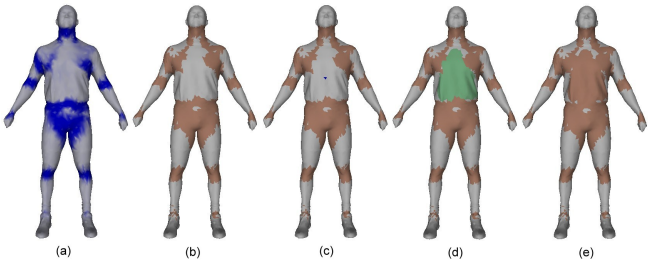

Fig. 2 (a) We show the deformable regions on the mesh. The color indicates that the region is deformable or not. It is close to blue if the region has more deformation; otherwise close to white. (b) Showing the faces in set A with maroon and set B with white. (c) A feature face p in blue is extracted from set B. (d) The influence region R in light green is obtained by the level set algorithm. (e) Assign all the faces in the influence region to set A.

After a feature face p has been extracted, our algorithm will create an influence region R for p. An influence region R is a near-rigid cluster and it includes a set of faces near p. For each pose, we use [31] to compute the deformation gradients of all faces in R and then average these deformation gradients [24] to represent the approximate deformation of the influence region R. The deformation of R is near-rigid throughout the animation sequence. Next, our algorithm will checks if this newly founded feature face p will form a new partition or will join with an existing partition. We determine this by evaluating the similarity of two average deformation gradient between any existing partition and this influence region R. Therefore, our algorithm may extract several feature faces to represent a partition and potentially get the better results, especially for the long, narrow sub-meshes. Finally, our algorithm iteratively extracts other feature faces until the area of the influence region R is smaller than a certain threshold.

Our feature extraction algorithm is described in the following steps. In this algorithm, the face sets A and B are used to help us to find the feature faces in the deforming mesh.

1. Look over all faces on the mesh and determine the face f_i is in the deformable region or not. If the distance from f_i to any of its adjacent face f_j is larger than a certain threshold ε, where this threshold ranges from 0.3 to 0.8 in our experimental results, we assign f_i to the set A; otherwise, assign f_i to the set B (Figure 2 (a) to (b)). The set B contains faces that deform slightly and are potential candidates for features faces.
2. Extract the feature face p from the set B which is the farthest face (i.e., with the maximum distance by Eq. (5)) to faces in the set A (Figure 2 (b) to (c)). This is a good choice to pick a feature face that is far away from the faces in deformable regions and all existing partitions.
3. Create the influence region R for the new feature face p and find the average deformation gradient N for R [24]. Then, we compare R with all the existing clusters $\Psi_i, i = 1...k$. If there exists some cluster j such that its average

deformation gradient $N^{'}$ is similar enough to N (i.e., $\| N - N^{'} \|_F^2 \leq \gamma$, where this threshold ranges from 0.4 to 1.6 in our experimental results), we merge R to Ψ_j; otherwise we set R as a new partition Ψ_{k+1} (Figure 2 (c) to (d)).

4. Move all faces including p in the influence region R from the set B to the set A. Therefore, in the next iteration (in step 2), we can find a new feature face that is not only far away from the deformable region, but is also far away from the existing partitions. Note that we let P_i be a set of feature faces in partition i, where $P_i \in \Psi_i$ and is needed in the face clustering algorithm in Section 3.3. Therefore, the extracted feature face p is also assigned to P_i. Repeat steps 2 to 4 until the area of the influence region R is smaller enough (Figure 2 (d) to (e)).

Note that in the step 4, after several iterations, the influence region R is becoming smaller and smaller, since more and more faces were grouped into the set A. This algorithm uses the idea of the level set to create the influence region for a feature face. We set the newly obtained feature face (in step 2) as level 1 and all remaining faces in the set A as level 0. By applying [32], the harmonic distance can be viewed as a fade out value between 1.0 and 0.0. Then our algorithm picks up the faces whose value is between 1.0 and 0.01 to form the influence region. Since faces in the set A were excluded for being selected as feature faces, we set these faces as level 0. Therefore, the algorithm does not select any face in the set A to appear in the influence region for a new feature face. Finally, after the above algorithm, the faces in P_i are extracted feature faces for partition i.

3.3 Face Clustering

After the feature faces are selected in Section 3.2, the face clustering algorithm determines to which partition each face belongs based on the distance between the face and the feature faces. Since the distance is much larger when there are deformable edges between two faces. In this case, they are very likely to be assigned to different partitions. Therefore, the partitions are separated by deformable edges. As shown in Figure 3 (a), red circles are two feature faces l and r, and the green triangle is an arbitrary face on the mesh. The path from the green triangle to the left feature face l needs to take more effort to cross a barrier (deformable edge), however, to the right feature face r needs not. Therefore, the green triangle is much closer to the right feature r, and face clustering algorithm will assign the green triangle to the right partition. Here we let P_i and F_i be the set of feature faces and the set of all faces in partition i, where $i = 1 \ldots k, \bar{F} = \bigcup_{i=1}^{k} F_i$, and $\bar{P} = \bigcup_{i=1}^{k} P_i$, such that $P_i \subseteq F_i$, $F_i \cap F_j = \emptyset$ and $P_i \cap P_j = \emptyset$ if $i \neq j$. Therefore, in order to achieve the above face clustering, our algorithm minimizes the following objective function:

$$\min \sum_{P_i \subseteq \bar{P}} \sum_{f \in F_i} Dis(f,p) \ \exists p \in P_i \tag{6}$$

In the above equation, $Dis(f,p)$ is the distance from face f to feature face p.

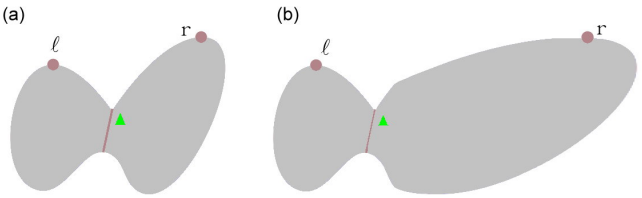

Fig. 3 (a) For the green face, the distance to l is larger than the distance to r because of the deformable edges. (b) The geodesic distance from the green face to r is a lot larger and thus the combined distance is enlarged.

However, due to unobvious deformable edges (i.e., deforms slightly), or the geodesic distance is very far from the feature face, some faces near boundary may be assigned to the wrong partitions by minimizing Eq. (6). For example in Figure 3 (b), if the geodesic distance from the green face to the right feature face r is much larger, then the combined distance to r may be larger than the distance to l. Therefore, the green face will be assigned to the wrong partition in the left side. To reduce this problem, we consider the influence of other non-feature faces in a partition. In Figure 3 (b), since there is a deformable region, the distance between the green face to all the faces in the left partition is larger. Therefore, the average distance from the green face to the left partition is probably larger than the right one, and then the green face is assigned to the right partition. In addition, as mentioned in Section 3.2, those feature faces are always selected to be far away from deformable regions. It is a good heuristic to make these feature faces to remain as close as possible to the center of the partition. In other words, both the distance from the face to feature faces and the average distance to all faces in the partition are considered, and therefore we reformulates the objective function Eq. (6) by the following:

$$\min \sum_{P_i \subseteq \bar{P}} \sum_{f \in F_i} \min(Dis(f,p), avg(Dis(f,F_i))) \ \exists p \in P_i \tag{7}$$

Where $avg(Dis(f,F_i))$ is the average distance from f to the faces in F_i, and the distance between the face f and the partition i is $\min(Dis(f,p), avg(Dis(f,F_i))) \ \exists p \in P_i$. The algorithm executes face clustering by minimizing Eq.(7) in an iterative manner which is described as follows:

1. Initially, consider the feature faces in each partition as its initial partition. That is, we set $F_i = P_i$.

2. Next, for each face $f \in \bar{F}$, assign f to the partition i if the distance (i.e., using $\min(Dis(f,p), avg(Dis(f,F_i)))$ $\exists p \in P_i$) from f to the partition i is the shortest one.

3. Finally, if there is a face f that belongs to partition i in the previous iteration, but f is assigned to partition j in this iteration, where $i \neq j$, we repeat Step 2 to Step 3 again.

Since $avg(Dis(f,F_i))$ will be changed as iterations are executed, the partition of the face f where it belongs potentially changes. The repeated loop will end until there is no face transferred from one partition to another one. In our experiment, the number of iteration is usually less than 10.

Using our face clustering algorithm, two disjoint regions with similar movement may be clustered into the same partition as shown in Figure 4. By applying disjoint set algorithm [6] to each partition, we can find out several clusters which are disconnected to each other. The partition is valid if and only if there is only one cluster in this partition. Otherwise, our algorithm will assign the feature faces in each cluster to a new partition and go back to the face clustering again.

Fig. 4 Disjoint regions (i.e., with same color) may be clustered into the same partition.

3.4 Boundary Smoothing

Up to now, the algorithm has already segmented the deforming mesh into several partitions. But, the rough cutting boundary may look visually unpleasant. So the goal of this step is to make them as smooth as possible. There are several boundary smoothing algorithms available to achieve this goal [4,15,16]. For simplicity in implementation and without changing the connectivity of the mesh, we refer to [15] and apply maximum flow minimum cut algorithm [6] to solve it. The idea of smoothing algorithm is to find a minimum cut that can pass through the deformable region with the smallest capacities. In order to preserve the smooth boundary and cutting positions still lie on the deformable regions, the capacity is the combination of the deformation distance with the length of the edge which is shared by these two faces. Therefore, we apply the definition of [15] and set the capacity of each two adjacent faces in the region as follows:

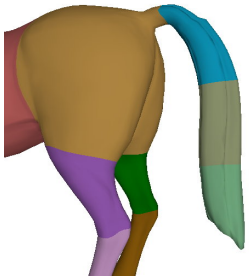

Fig. 5 The tail of horse is segmented into three partitions according to deformation information in its animation sequence.

$$Capacity_{ij} = \alpha \frac{D_{ij}}{avg(D)} + (1-\alpha)\frac{edge_{ij}}{avg(edge)} \quad (8)$$

where $edge_{ij}$ is the length of an edge shared by face i and face j, $avg(D)$ is the average of D_{ij}, and $avg(edge)$ is the average of $edge_{ij}$. In our experimental results, we set α less than 0.1.

4 Segmentation Results

Figure 7 shows several experimental results by using our segmentation algorithm. Different partitions are rendered in different colors and some key frames of their mesh sequences are also shown in this figure. Using the proposed algorithm, the segmented boundaries can always be set on the deformable regions of the meshes. The meshes can be successfully partitioned into several near-rigid components. In the next section, we will demonstrate the usefulness of these segmented results. Let us have a further look at the racing horse example in Figure 5. In this example, the tail of horse is segmented into three partitions due to the deformation of tail in the animation sequence as shown in Figure 7. In contrast to our algorithm, both [15,16] consider the shape property such as concavity rather than deformation information. Both [15,16] do not segment the horse tail into three components, since there is no obvious concavity on the tail in Figure 5. Therefore, it is not suitable to apply [15] and [16] for segmenting deforming meshes into near-rigid components. As mentioned in Section 1, different applications require different segmentation schemes.

Figure 6 shows an example of the experimental comparison between [19] and our algorithm. Both algorithms segment the same mesh of a running horse sequence used in [19]. Using the proposed algorithm, thighs, legs, and hoofs are grouped into different partitions. These parts are separated from the head and body part. In this animation sequence, the movement of head and body is very similar and is different from those of other parts. Therefore, our result is more reasonable than that of [19] in this example. Although [19] also use motion similarity to cluster faces, their results can

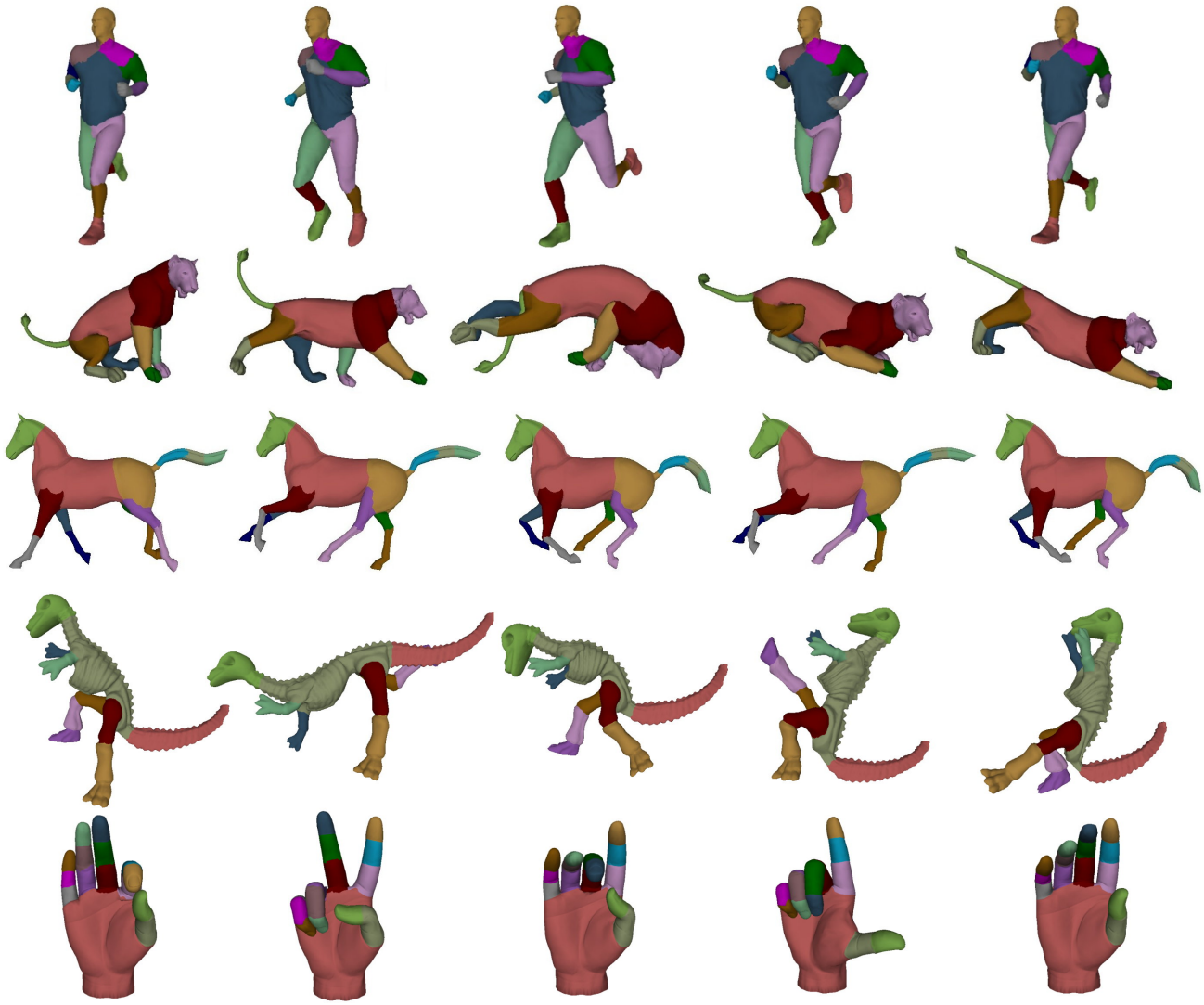

Fig. 7 More Segmentation examples.

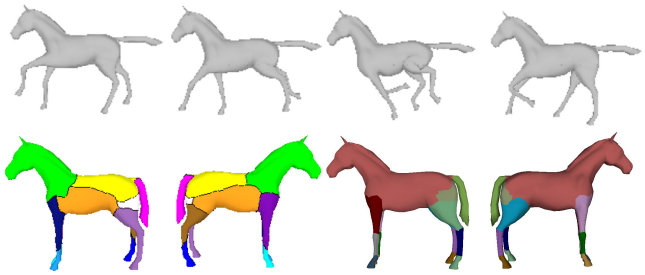

Fig. 6 An experimental comparison between the proposed algorithm (right) and [17] (left) for segmenting the same mesh of a running sequence.

not segment meshes into near-rigid components well. More examples can be seen in their paper [19].

Fig. 8 An experimental comparison between the proposed algorithm (right) and [11] (left) for segmenting the same mesh of a running sequence.

Another comparison with [11] is shown in Figure 8. From this example, it is very obvious to notice that both segmentation results are quite different. In the left side [11] of Figure 8, those faces in the same partition (with the same color) may not be connected to each other. In addition, some faces in

black color do not belong to any partition. This kind of segmentation result is suitable to some applications like skinning mesh [11]. However, this method may not benefit some applications such as deformation transfer [31], since we can not use these partitions to adjust the reference poses of source and target meshes.

	Dual Graph Generation	Feature Extraction	Clustering Final Cut
Horse (16843 faces) (48 key-frame)	405(s)	181(s) (28 features)	220(s)
Human (15000 faces) (25 key-frame)	296(s)	79(s) (17 features)	333(s)
Dinosaur (20000 faces) (9 key-frame)	539(s)	163(s) (23 features)	339(s)
Hand (15855 faces) (9 key-frame)	328(s)	292(s) (56 features)	153(s)
Lion (9996 faces) (9 key-frame)	115(s)	57(s) (29 features)	106(s)

Table 1 Timing statistics for segmentation result on animation sequences.

Our segmentation algorithm was implemented on Intel Pentium 4 3.4GHz PC with 2.0 G Ram. Several timing statistics for experimental results are shown in Table 1. The major bottlenecks of the proposed algorithms are: 1) dual graph generation, and 2) face clustering and boundary smoothing parts. The computational complexity of these two parts is: $O(F^2 logF + IF^2)$, where F is the number of faces in the mesh and I is the number of iterations in the face clustering algorithm. These two parts are involved with all-pairs shortest path computation, therefore the computational cost increases as the number of faces increases. To reduce the cost of all-pairs shortest paths, [15,16] suggests simplifying the meshes first, then segmenting meshes and finally propagating segmented results to the original meshes. In the future, we plan to include this suggestion to reduce the computation cost.

5 Deformation Transfer with Variant Poses

To demonstrate the usefulness of the algorithm, we apply our near-rigid segmented results to the application of deformation transfer with variant poses. For this application, our algorithm first partitions the source mesh according to its animation. Next, the target mesh is compatibly segmented into the same number of partitions based on their face correspondence between source and target meshes using [31]. Given a corresponding partition pair in Figure 9, and the face correspondence is known between these two corresponding partitions. Therefore, the corresponding normal vectors of each pair faces are known, too. Then, we can apply [3] to compute a rigid transformation that can be used to transform these normal vectors in the target partition to the corresponding normal vectors in the source partition as close as possible.

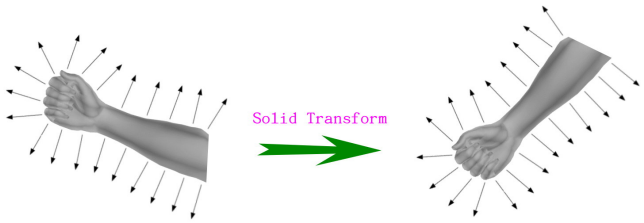

Fig. 9 Rigid transformation can adjust the orientation of the target partition to be similar to the source partition.

For each pair partitions, we compute a rigid transformation for them. Each rigid transformation is used to adjust the orientation of each target partition to that of the corresponding source partition in the reference poses. Therefore, when our algorithm transfers the deformation of a source face to the target face, we do not only consider the deformation gradient of the face, but also consider the rigid transformation of its belonged partition. In Eq. (9), we reformulate Eq. (7) in [31] to solve the restriction of deformation transfer for triangle meshes in [31] which requires similar reference poses between source mesh and target mesh.

$$\min_{T_1+d_1 \ldots T_T+d_T} \sum_{j=1}^{|M|} \| P_{tj} \times S_{sj} - T_{tj} \|_F^2$$

$$\text{subject to } T_j v_j + d_j = T_k v_i + d_k, \forall i, \forall j, k \in p(v_i) \qquad (9)$$

where P_{tj} is the rigid transformation of the partition which face j belongs.

Since our algorithm can segment meshes into near-rigid components well, we can find good rigid transformations for them. As a result, we can obviously see that the results with rigid transformations are better than those simply obtained by [31] as shown in Figure 10 and 12. From these two figures, our results look quite good, although there are some minor artifacts. For example, the hoofs of the camel are unnaturally deformed in the bottom row of Figure 10. This is because the segmentation does not select the hoofs as separate components.

6 Conclusions and Future Work

This paper presents an automatic approach to segment a deforming mesh into near-rigid components. In addition, the application of deformation transfer with variant poses is used to demonstrate the usefulness of the algorithm. There are several future works to be explored soon. First, we would like to reduce our computation cost by the method suggested in [15,16]. Second, the memory requirement of the all-pairs shortest path is a significant limitation and drawback of the proposed algorithm. We plan to investigate a new method without the requirement of the all-pairs shortest path. Third,

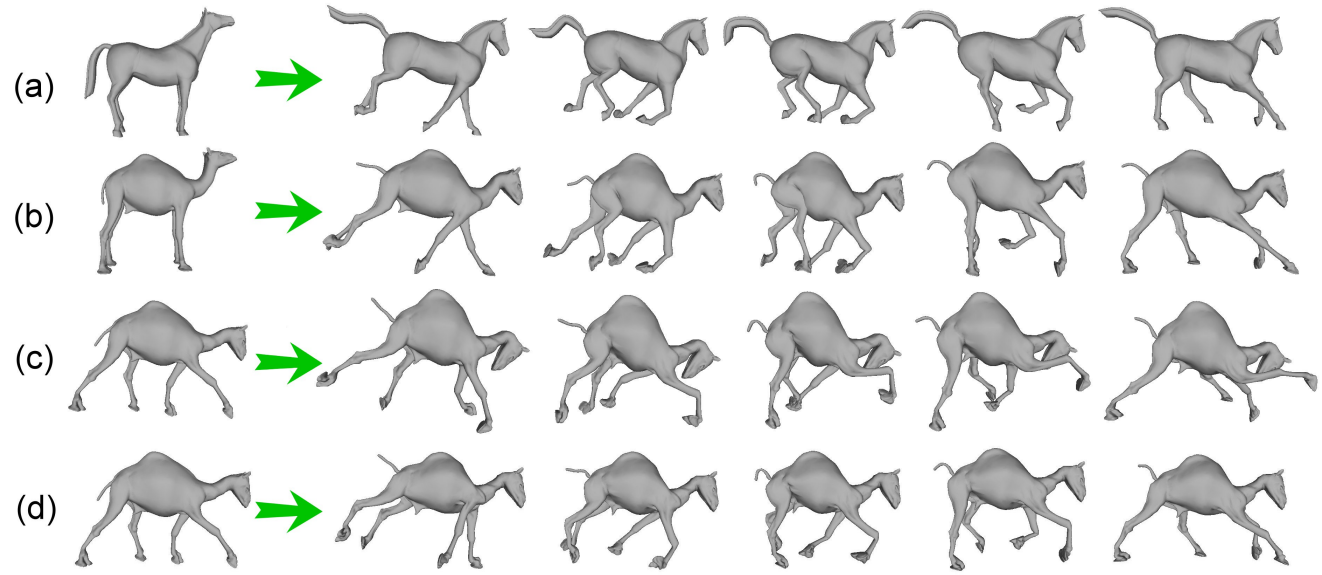

Fig. 10 (a) and (b) show that deformation transfer from horse to camel. However, the retargeted animation is crashed in (c) due to different reference poses. More reasonable results are obtained by our algorithm in (d).

the purpose of our algorithm is to partition a deforming mesh into near-rigid components. However, to segment a mesh which has large deformation everywhere such as an example from [31] in Figure 11, the partitioned results using our algorithm can be over-segmented as shown in the right side of Figure 11. Although these partitions look over-segmented, there is still no big problem to apply these segmented results to the application in this paper. In the future, we are seeking a better approach to handle such the highly deformed cases. Finally, we will apply our approach to other important applications such as LOD of animation and collision detection for deforming meshes [11].

Fig. 11 The over-segmented result because that the mesh has large deformation everywhere.

Acknowledgements The authors give their sincere thanks to anonymous reviewerś helpful comments to improve their paper. They thank the Computer Science and Artificial Intelligence Laboratory for the horse and lion animation meshes, and Microsoft Research Asia for dinosaur animation meshes. This project is supported by the National Science Council, Taiwan, under contract No. NSC-94-2213-E-006-005 and NSC-94-2213-E-006-063.

References

1. Attene, M., Falcidieno, B., Spagnuolo, M.: Hierarchical mesh segmentation based on fitting primitives. Vis. Comput. **22**(3), 181–193 (2006). DOI http://dx.doi.org/10.1007/s00371-006-0375-x
2. Attene, M., Katz, S., Mortara, M., Patane, G., M.Spagnuolo, Tal, A.: Mesh segmentation - a comparative study. In: SMI '06: Proceedings of the Shape Modeling International 2006. IEEE Computer Society (2006)
3. Besl, P.J., McKay, N.D.: A method for registration of 3-d shapes. IEEE Trans. Pattern Anal. Mach. Intell. **14**(2), 239–256 (1992). DOI http://dx.doi.org/10.1109/34.121791
4. Bischoff, S., Weyand, T., Kobbelt, L.: Snakes on triangle meshes. Bildverarbeitung fur die Medizin pp. 208–212 (2005)
5. Chung, F.R.K.: Spectral graph theory. CBMS Regional Conference Series in Mathematics (1997)
6. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. McGraw-Hill (2001)
7. Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., Dobkin, D.: Modeling by example. ACM Trans. Graph. **23**(3), 652–663 (2004). DOI http://doi.acm.org/10.1145/1015706.1015775
8. Garland, M., Willmott, A., Heckbert, P.S.: Hierarchical face clustering on polygonal surfaces. In: SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics, pp. 49–58. ACM Press, New York, NY, USA (2001). DOI http://doi.acm.org/10.1145/364338.364345
9. Garland, M., Willmott, A., Heckbert, P.S.: Hierarchical face clustering on polygonal surfaces. In: SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics, pp. 49–58. ACM Press, New York, NY, USA (2001). DOI http://doi.acm.org/10.1145/364338.364345
10. Gregory, A.D., State, A., Lin, M.C., Manocha, D., Livingston, M.A.: Interactive surface decomposition for polyhedral morphing. The Visual Computer **15**(9), 453–470 (1999)
11. James, D.L., Twigg, C.D.: Skinning mesh animations. ACM Trans. Graph. **24**(3), 399–407 (2005). DOI http://doi.acm.org/10.1145/1073204.1073206
12. Julius, D., Kraevoy, V., Sheffer, A.: D-charts: Quasi-developable mesh segmentation. In: Computer Graphics Forum, Proceedings of Eurographics 2005, vol. 24, pp. 581–590. Eurographics, Blackwell, Dublin, Ireland (2005)
13. Kalvin, A.D., Taylor, R.H.: Superfaces: Polygonal mesh simplification with bounded error. IEEE Comput. Graph. Appl. **16**(3), 64–77 (1996). DOI http://dx.doi.org/10.1109/38.491187
14. Karni, Z., Gotsman, C.: Spectral compression of mesh geometry. In: SIGGRAPH '00: Proceedings of the 27th annual conference

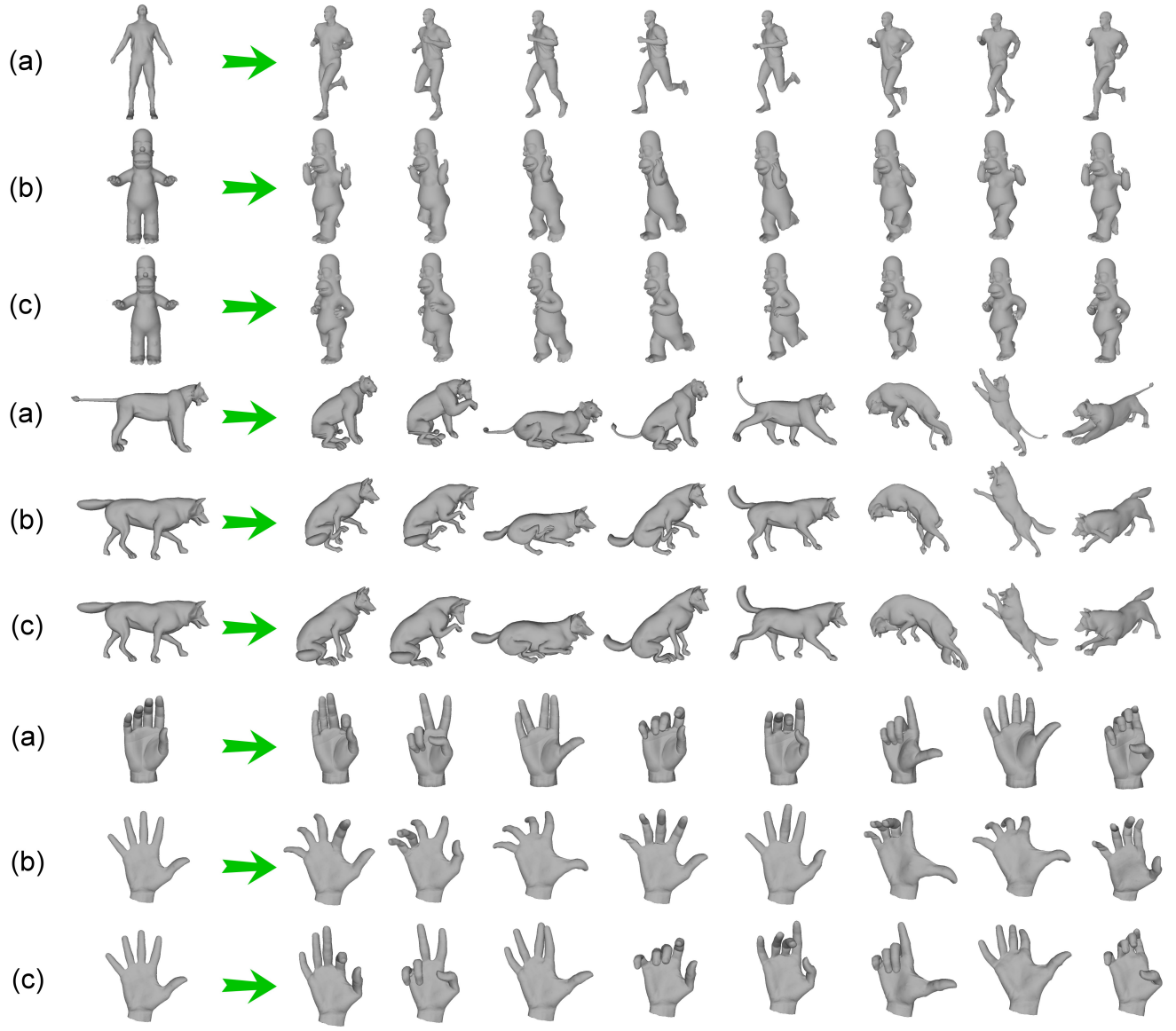

Fig. 12 More Examples of deformation transfer with variant poses. (a)The animation of source meshes. (b)The unpleasant results because that the kinematic pose between source and target meshes are different. (c)The results corrected by our algorithm.

on Computer graphics and interactive techniques, pp. 279–286. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (2000). DOI http://doi.acm.org/10.1145/344779.344924
15. Katz, S., Leifman, G., Tal, A.: Mesh segmenetation using feature point and core extraction. Visual Computer **21**(8-10), 865–875 (2005)
16. Katz, S., Tal, A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Trans. Graph. **22**(3), 954–961 (2003). DOI http://doi.acm.org/10.1145/882262.882369
17. Lee, T.Y., Huang, C.C.: Dynamic and adaptive morphing of three-dimensional mesh using control maps. IEICE Transactions **88-D**(3), 646–651 (2005)
18. Lee, T.Y., Huang, P.H.: Fast and intuitive metamorphosis of 3d polyhedral models using smcc mesh merging scheme. IEEE Transactions on Visualization and Computer Graphics **9**(1), 85–98 (2003). DOI http://dx.doi.org/10.1109/TVCG.2003.1175099
19. Lee, T.Y., Lin, P.H., Yan, S.U., Lin, C.H.: Mesh decomposition using motion information from animation sequence. COMPUTER ANIMATION AND VIRTUAL WORLDS **16**, 519–529 (2005)
20. Lévy, B., Petitjean, S., Ray, N., Maillot, J.: Least squares conformal maps for automatic texture atlas generation. In: SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp. 362–371. ACM Press, New York, NY, USA (2002). DOI http://doi.acm.org/10.1145/566570.566590
21. Lien, J.M., Amato, N.M.: Approximate convex decomposition. In: SCG '04: Proceedings of the twentieth annual symposium on Computational geometry, pp. 457–458. ACM Press, New York, NY, USA (2004). DOI http://doi.acm.org/10.1145/997817.997889
22. Lin, C.H., Lee, T.Y.: Metamorphosis of 3d polyhedral models using progressive connectivity transformations. IEEE Transactions on Visualization and Computer Graphics **11**(1), 2–12 (2005). DOI http://dx.doi.org/10.1109/TVCG.2005.12
23. Lloyd, S.: Least square quantization in pcm. IEEE Transactions on Information Theory **28**, 129–137 (1982)
24. Moakher, M.: Means and averaging in the group of rotations. SIAM J. Matrix Anal. Appl. **24**(1), 1–16 (2002). DOI

http://dx.doi.org/10.1137/S0895479801383877
25. Mortara, M., Patane, G., Spagnuolo, M., Falcidieno, B., Rossignac, J.: Plumber: a method for a multi-scale decomposition of 3d shapes into tubular primitives and bodies. In: Ninth ACM Symposium on Solid Modeling and Applications SM'04, Palazzo Ducale, Genova, June 9-11. 2004, pp. 339–344. ACM (2004)
26. Popa, T., Julius, D., Sheffer, A.: Material aware mesh deformations. SMI (2006)
27. Sander, P.V., Wood, Z.J., Gortler, S.J., Snyder, J., Hoppe, H.: Multi-chart geometry images. In: SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 146–155. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2003)
28. Sattler, M., Sarlette, R., Klein, R.: Simple and efficient compression of animation sequences. In: SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 209–217. ACM Press, New York, NY, USA (2005). DOI http://doi.acm.org/10.1145/1073368.1073398
29. Shamir, A.: A formulation of boundary mesh segmentation. In: 3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium on (3DPVT'04), pp. 82–89. IEEE Computer Society, Washington, DC, USA (2004). DOI http://dx.doi.org/10.1109/3DPVT.2004.13
30. Shlafman, S., Tal, A., Katz, S.: Metamorphosis of polyhedral surfaces using decomposition. Computer Graphics forum **21**(3) (2002)
31. Sumner, R.W., Popović, J.: Deformation transfer for triangle meshes. ACM Trans. Graph. **23**(3), 399–405 (2004). DOI http://doi.acm.org/10.1145/1015706.1015736
32. Zayer, R., Rössl, C., Karni, Z., Seidel, H.P.: Harmonic guidance for surface deformation. In: M. Alexa, J. Marks (eds.) Computer Graphics Forum, Proceedings of Eurographics 2005, vol. 24, pp. 601–609. Eurographics, Blackwell, Dublin, Ireland (2005)
33. Zöckler, M., Stalling, D., Hege, H.C.: Fast and intuitive generation of geometric shape transitions. The Visual Computer **16**(5), 241–253 (2000)
34. Zuckerberger, E., Tal, A., Shlafman, S.: Polyhedral surface decomposition with applications. Computers & Graphics **26**(5), 733–743 (2002)

Tong-Yee Lee received the PhD degree 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 has served as a guest associate editor for IEEE Transactions on Information Technology in Biomedicine from 2000 to 2006. His current research interests include computer graphics, non-photorealistic rendering, image-based rendering, visualization, virtual reality, surgical simulation, and distributed and collaborative virtual environment. He leads the Computer Graphics Group/Visual System Lab at National Cheng-Kung University (http://couger.csie.ncku.edu.tw/∼vr).He is a member of the IEEE.

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, 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.