Example-driven animation synthesis

Article *in* The Visual Computer · July 2008

DOI: 10.1007/s00371-008-0258-4 · Source: DBLP

2 authors:

Yu-Shuen Wang
National Chiao Tung University
50 PUBLICATIONS **1,828** CITATIONS
SEE PROFILE

Tong-Yee Lee
National Cheng Kung University
190 PUBLICATIONS **4,454** CITATIONS
SEE PROFILE

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

Yu-Shuen Wang · Tong-Yee Lee

Example-driven Animation Synthesis

Abstract We introduce an easy and intuitive approach to create animations by assembling the existing animations. Using our system, the user needs only to simply scribble regions of interest and select the example animations that he/she wants to apply. Our system will then synthesize a transformation for each triangle and solve an optimization problem to compute the new animation for this target mesh. Like playing a jigsaw puzzle game, even a novice can explore his/her creativity by using our system without learning complicated routines, but just using a few simple operations to achieve the goal.

1 Introduction

Creating a visually pleasing and realistic animation is difficult and time-consuming, which requires experienced skill or expensive equipment to accomplish. Therefore, most researchers in computer graphics try to reduce the burden of the animators by providing easy-to-use tools to create animations. In this paper, we introduce a new approach that synthesizes the animation guided by the intuitive user input. The animators or non-professionals can use the system to create animations easily and quickly.

Our system is based on animation reuse [23] which copies the animation of the source mesh and then applies to the target mesh. However, [23] requires the shape of the source and the target meshes to be similar, and reduces the advantage of the application. For example in Figure 1, it is very difficult to find a model looks like this target mesh, and thus we cannot reuse the existing animation on it. In this paper, the above restriction is lifted because we transfer the deformation between partitions rather than the entire mesh. In this example, the animations of the warrior, eagle, and the horse are assembled and transferred to perform the animation of the celestial monster. Through our intuitive user interface, the user can simply scribble the interesting regions of the target mesh, and apply the selected animations to create a new animation.

To transfer the deformation from the source triangle to the correct target triangle, we have to determine the correspondence between these two meshes. Different to the other applications, we found that the correspondence should be more accurate at the flexible regions because the deformations near these regions differ the most. In other words, it is better for the user to specify the marker points on the flexible regions rather than the high curvature regions. By analyzing the source animation, our system is able to notify the user which regions need more markers and which regions can accept more inaccuracy. We then solve for the least squares meshes to transform the two meshes into similar shapes so as to find the pairs of compatible triangles. With our proposed system, the user can specify proper and fewer markers between the meshes and thus saving the operating time on this tedious work.

The main contribution of this paper is providing an easy-to-use animating system for non-professionals, which assembles existing animations to create a new animation. The proposed method also informs the user about putting markers on the proper and right positions. Novice users do not have to learn complicated operations and are able to create animations easily.

2 Related Work

Computer animation is a very active area of research. In the past, a significant number of techniques have been proposed. Therefore, we are not going to give a very thorough overview of such large research topics, but review the related works in the following categories.

Skeleton-based Animation: The most common method used to create animations is skeleton subspace deformation. In this category the mesh is driven by editing the skeleton. The greatest advantage of this method

Computer Graphics Group/Visual System Laboratory, Department of Computer Science and Information Engineering, National Cheng-Kung University, No.1, Ta-Hsueh Road, Tainan 701, Taiwan, R.O.C.

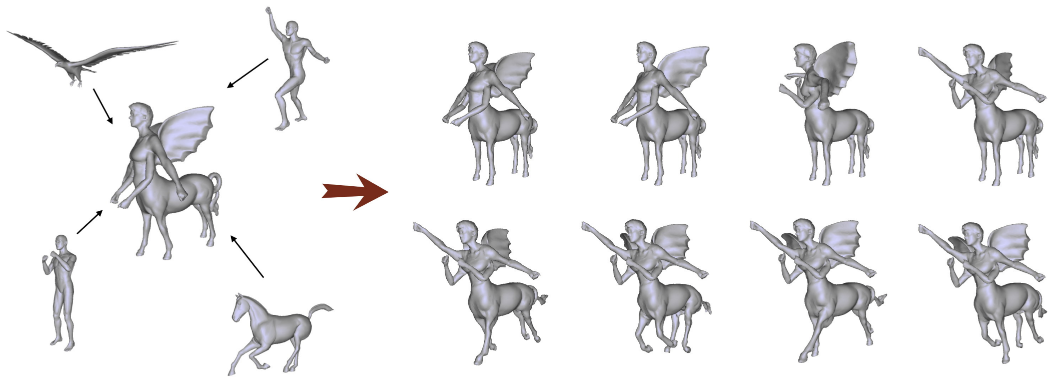

Fig. 1 We transfer the animation of the eagle's wings, horse's body and the warrior's arms to the celestial monster. The different parts of the animations are well-blended and synthesize a new and natural animation.

is its speed, which can be accelerated by hardware. However, it is always time-consuming to rig and skin a model for even professional animators. In addition, the skeleton-based method cannot deal with non-rigid models like a face and potentially suffers from the so-called "collapsing joint" or "candy wrapper" defects, and thus requires better solutions such as [6, 9, 14, 16] to reduce this shortcoming.

Motion Capture Based Animation: The motion capture system can be used to create realistic animations by recording an actor's motion in the real world. There is a number of conventional marker sets varying from markers for capturing the sequence of human poses [1, 2, 7] or facial expressions [5] to hundreds for capturing detailed skin deformations [19]. However, the equipment is very expensive and only a few people can access it to record the motion from real world.

Skeleton-free Mesh Editing: Many skeleton-free mesh editing techniques have been proposed to deform a mesh using a few anchors. The user moves the anchors through the intuitive interface [10] and the system computes new positions for the remaining non-anchor vertices based on their geometric features. The main advantage of these algorithms is that they can preserve the original properties of the mesh such as area [25], volume [4, 27] and local details [8, 17, 22, 26, 27]. In addition, some example-based mesh editing methods [3, 24] interpolate the existing poses in a non-linear way to fit the user's constraints and therefore synthesize meaningful poses.

Animation Reuse: Our approach falls into this category. Without rigging and skinning the models, the user only needs to manually specify compatible markers on the source and target meshes to compute the correspondence. In this category, techniques such as [18, 23] are particularly useful for novice users to create animations. However, some requirements reduce the advantages of the algorithms because 1) the shapes of the two meshes should be similar, and 2) the reference meshes should have similar kinematic poses. Our approach reduces these limitations.

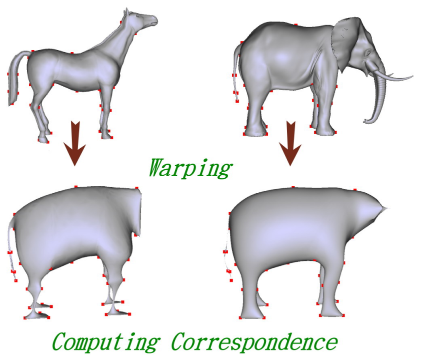

Fig. 2 The source (horse) and the target (elephant) meshes are warped into similar shapes. Note that the elephant's nose and head are shrunk together because there is no marker. Although the shapes in some regions are not very similar, our algorithm can still obtain the proper deformation for each target triangle and then produce reasonable results.

3 Algorithm

In this section, we describe the method of our 3D animation synthesis system. The user first specifies the marker points on the source and the target meshes to establish the correspondence map and then simply scribble the regions of interest on the target mesh to apply source animation. Our method transfers the deformation from each source triangle of the desired animation smoothly and naturally to the target mesh.

3.1 Correspondence

To define how the source animation is transferred to the target mesh, some compatible markers specified on both meshes are required. The system then determines the

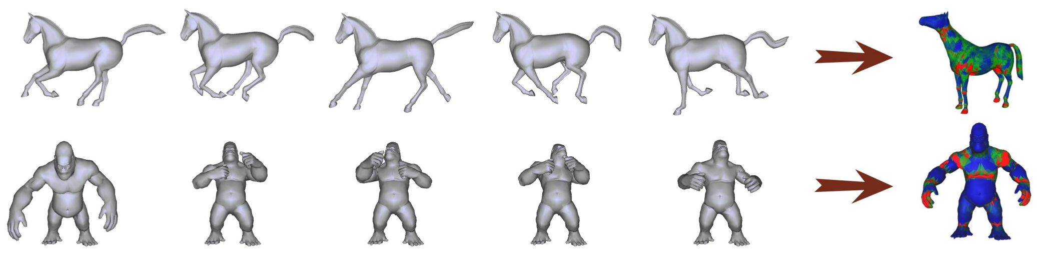

Fig. 3 The indication which helps the user to select the markers is computed according to the source animation. The region in red indicates the surrounding deformations are very different and the user should put the markers there. While the region in blue shows that the deformations near there are similar and thus can accept more inaccuracy.

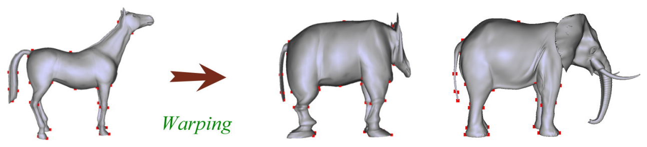

Fig. 4 Implementation of the correspondence algorithm proposed by [23], and some problems found in it. Because there are no markers on the heads of either mesh, the warped head of the source mesh will be very different from the target mesh. Therefore, the triangles on the elephant's trunk are considered compatible to the horse's forelegs, and thus obtain the unreasonable result.

compatible triangles automatically. Before finding the closest source and the target triangle pairs, some algorithms are designed to warp one mesh like the other, yet our algorithm warps the two meshes into similar shapes. Therefore, some redundant markers can be removed in this step and reduce the burden on the user.

We propose an automatic approach to establish the correspondence from a small set of user-specified markers. Before the user tries to select the markers, our algorithm first analyzes the information of the source animation, then uses different colors to visualize the degree of significance on the source mesh (Figure 3). For the deformation transfer application, this indication can significantly help the user make better choices while specifying the markers. For example in Figure 2, since our indication shows that the deformations in the horse's head are nearly identical (Figure 3), no markers are required on the heads while computing the correspondence. Although this kind of simplification will cause some inaccuracies on the elephant's head and nose, the mapped deformation gradients are also suitable for the elephant; therefore, we can still obtain the correct animation.

In [23], although they put more markers on the horse's head to better approximate the elephant's head, we found that it was not necessary and only increased the user's effort. We believe the work in [23] can also work well when the user selects the markers based on Section 3.1.1. However, in some cases like Figure 4, their algorithm computes incorrect correspondence and therefore the deformation was transferred to the wrong triangles. Our correspondence algorithm does not have this problem.

3.1.1 Marker Indication

Similar to [12,13], we computed the maximum difference from the deformation gradients between adjacent faces among all frames of the given animation to represent the flexibility of each triangle. The equation is formulated as:

$$G_{ij} = \max_{f=1\ldots k} \|\mathbf{D}_{fi} - \mathbf{D}_{fj}\|_F^2, \quad \{i,j\} \in \mathbf{U}, \tag{1}$$

where $\mathbf{U}$ is the set of adjacent faces, k is the number of frames in the animation sequence, and $\mathbf{D}$ is the deformation gradient that transforms the triangle from the reference pose to the animation pose. The flexibility of triangle i is thus determined by averaging G_{ij}. As Figure 3 shows, the colors close to red represent the regions are flexible and the deformations around these regions are quite different. The user needs to set markers at these positions to enhance the accuracy of correspondence. Otherwise, the deformation transferred to improper triangles will cause serious mistakes.

3.1.2 Warping

Our algorithm determines the correspondence between the source and the target meshes by warping both of them into similar shapes (Figure 2). According to the user's specification, we constrain the source markers at the positions of their compatible target markers. The least squares meshes [21] are then solved, respectively, using these marker constraints. Since the shapes of the source and the target meshes are similar after the deformation, we can determine the compatible triangles between the meshes based on the distances of their centroids.

We observed that the least squares meshes [21] are fairly smooth except for the regions near constrained vertices. With the aid of this property, our algorithm can avoid bump surface disturbances when we compute the correspondence. The least squares meshes are reconstructed using only the mesh information and a few constrained vertices by minimizing the quadratic energy terms, i.e., we solve

$$\|\mathbf{L}V'\|^2 + \sum \omega|\mathbf{c}_i' - \mathbf{c}_i|^2, \quad \mathbf{c}_i \in \mathbf{C} \tag{2}$$

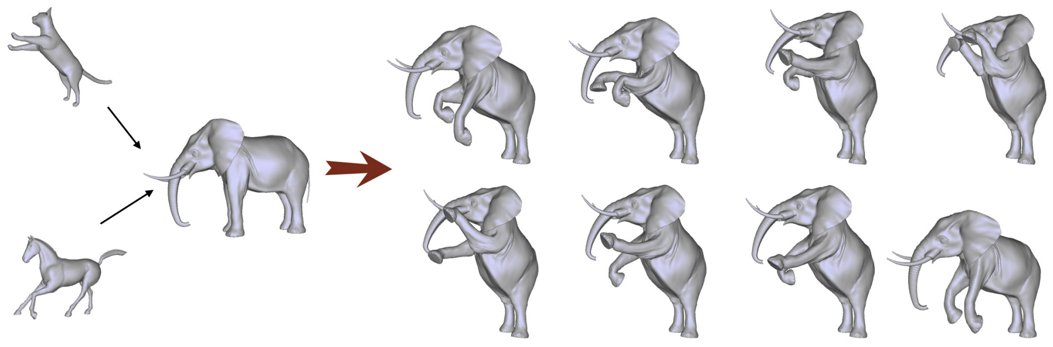

Fig. 5 The stretching and running animations are multiplied together on the elephant for the desired animation.

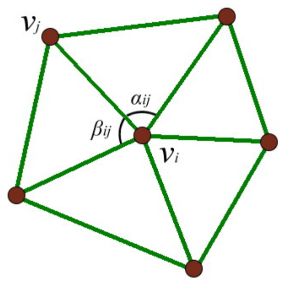

Fig. 6 The angles used in the mean value coordinates for edge {i, j}.

where $\mathbf{L}$ is the Laplacian operator, $\mathbf{V}'$ is the reconstructed vertices, $\mathbf{C} \subset \mathbf{V}$ denotes the set of constrained vertices and ω is a large number to enforce the soft constraint ($\omega = 10000$ in our experiment results). Here, we set coefficients of the Laplacian operator by the mean value coordinate, where

$$\mathbf{L}_{ij} = \begin{cases} \delta_{ij} & if\ \{i,j\} \in \mathbf{E} \\ -q_i & if\ \ i = j \\ 0 & otherwise \end{cases},$$

$$\delta_{ij} = \frac{\tan(\alpha_{ij}/2) + \tan(\beta_{ij}/2)}{\|v_i - v_j\|}, \quad q_i = \textstyle\sum_{\{i,k\}\in\mathbf{E}} \delta_{ik}, \quad (3)$$

$\mathbf{E}$ is the set of edges between two nearby vertices and α_{ij} and β_{ij} are the angles depicted in Figure 6. To solve for the vertex positions of the deformed mesh, we transform the equations into an over- determined linear system

$$\left[\frac{L}{0|\omega}\right]\begin{bmatrix}V'\\C\end{bmatrix} = \begin{bmatrix}0\\C\end{bmatrix}. \quad (4)$$

Since the sparse matrix can be solved with very low cost, our algorithm computes the correspondence efficiently.

3.1.3 Correspondence Map

Once the source and the target meshes are warped into similar shapes, we determine the two triangles are compatible if their centroids are close enough and the angle between their deformed face normals is less than 90°. This orientation test is necessary because triangles that have very short distances but belong to different regions should be prevented from being considered as compatible (e.g., triangles from the two sides of the armpit). The target triangle will correspond to at least one source triangle. That is, if there is no source triangle close enough to the target triangle, we set the closest one compatible to this target triangle.

3.2 Animation Synthesis

After the correspondence is established, the system will know to which target triangle the source triangle deformation should be transferred. Our system provides a user-friendly interface for the user to scribble the regions of interest on the target mesh and select the source animations from the database. The system copies the source partial deformations and applies them to the target mesh. The deformation of each target triangle can be obtained by interpolating or compounding from several source animations. For example in Figure 5, the cat stretching and the horse foreleg running are extracted and are multiplied together to compute the new elephant animation.

However, this naïve approach creates artifacts. For example in Figure 7, the careless scribbling causes kinematic problems. The elephant's forelegs are bent at the wrong positions and look very strange. In addition, the angle at the joint is too sharp because the two deformations are not well-blended. Thus the selection system must be more intelligent and the different deformations between partitions must be smoothly interpolated. In the following, we will describe how to obtain natural and smooth animations.

3.2.1 Intelligent Scribbling

The source and the target meshes are compatible after establishing the correspondence map. Therefore, we can obtain the flexibility of the target triangle from the

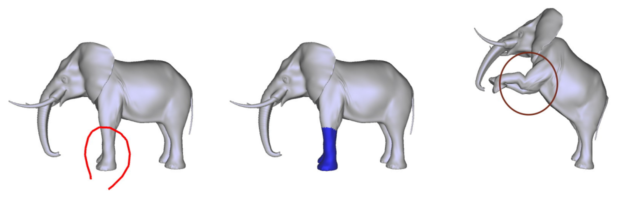

Fig. 7 The regions are sometimes not correctly selected for the deformation and therefore cause artifacts.

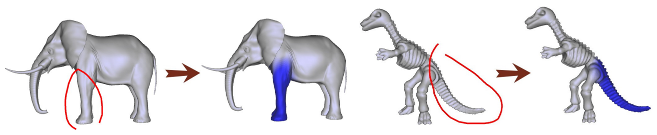

Fig. 8 Our system refines the regions after the user's selection. The regions in blue are the triangles refined by our system. The colors between blue and white mean that the regions are used to blend the different deformations.

source mesh and then separate the target mesh into several partitions according to a threshold γ (usually is 0.1). In the beginning we set each triangle as an individual partition. After that, we merge the nearby partitions if their flexibilities are both less than γ. This process will loop until there is no partition left to merge. We consider the partitions with more than one triangle as **stiff partitions** which copy the deformations of the source meshes, and the rest are **flexible partitions**, which are used to blend the animations between different stiff partitions. Note that the smaller γ means the deformations within the partition are more similar, and thus the extracted stiff partitions are more rigid.

When more than half the number of triangles in a partition is selected, we consider that all triangles in that partition are selected. Otherwise we remove all the triangles from being transferred deformation. We call this approach **intelligent scribbling**. This is because the deformation in the same stiff partition must be similar, especially when kinematic deformations are transferred. From our experience, we observed that this is very convenient and helpful for the user to scribble the regions of interest. In addition, this intelligent design can prevent some unexpected artifacts caused by careless selection. For example in Figure 8, with the aid of our intelligent scribbling, the refined regions are adequate for the kinematics, and the boundaries are well-blended.

3.2.2 Deformation Blending between Partitions

When the source animation is transferred to the target mesh, we should blend this animation with the transferred animation by the triangles in the flexible partitions. For this reason, we first compute the blending weight of each triangle and then determine the deformation gradient based on this weight. The source deformation gradients are interpolated with the identity matrix and then multiplied to those of the target animation if the user wants to compound the two animations. In contrast, the two deformation gradients are simply interpolated if the user requires the two animations to be well-blended. Of course, the weights of the stiff triangles are 1.0 if the triangles are selected, otherwise they are 0.0. The weights of the flexible triangles depend on the distance and the weights of the stiff triangles around them. To obtain smooth blending weights and to keep the weights of the stiff triangles constant, we compute the unknown weights w by minimizing the following quadratic function:

$$\sum_{i\in T}\sum_{j\in Q_i}|w_i-w_j|^2+\kappa\sum_{a\in\Phi}|w_a-1|^2+\kappa\sum_{b\in\Psi}|w_b|^2, \qquad (5)$$

where T denotes the triangles on the target mesh, κ is a large number to enforce the soft constraint ($\kappa = 10000$ in our experiment results), Q_i is the set of triangles adjacent to triangle i, Φ and Ψ denote the selected and unselected triangles, respectively. We solve the equation in a least square sense similar to the warping algorithm mentioned in Section 3.1.2.

The deformation gradients are interpolated using the weights obtained in Equation 5. In Figure 8, the color regions from blue to white are used to blend the animations of different partitions. Because linear combination is not adequate to interpolate transformations, we first decompose the deformation gradient into rotation and stretch components [20] and then interpolate the two transformations respectively. The rotation matrices should be reformulated as quaternion and then interpolated by the SLERP algorithm, while the stretch matrices are linearly interpolated. We compute the synthesized deformation gradients by multiplying the two interpolated transformations, and thus naturally smooth the deformations between partitions.

3.2.3 Deformation Transfer

After the target deformation gradients are determined, we can reconstruct the target animation in the same way as [23] by solving an optimization problem. Therefore, we introduce the objective function:

$$\min_{\mathbf{N}_1+\mathbf{d}_1\ldots\mathbf{N}_{|\mathbf{N}|}+d_{|\mathbf{N}|}}\sum_{j=1}^{|\mathbf{N}|}\|\mathbf{M}_j\times\mathbf{R}_j-\mathbf{N}_j\|_F^2$$

$$\text{subject to}\quad \mathbf{N}_j\mathbf{v}_i+\mathbf{d}_j=\mathbf{N}_k\mathbf{v}_i+\mathbf{d}_k,\quad i,j,k\in p(\mathbf{v}_i),(6)$$

where $p(\mathbf{v}_i)$ are the triangles that share $\mathbf{v}_i$, $\mathbf{M}$ denotes the set of synthesized target deformation gradient, $\mathbf{N}$ is the unknown deformation gradients for the solved animation and $\mathbf{R}$ is the rotation transformation obtained from pose modification.

3.2.4 Pose Modification

The transferred results will have some problems if the source and the target reference poses are dissimilar [13].

Our system solves this problem by applying the method in [13] to modify the orientation of the selected partition. In addition, we also provide a user-friendly interface for the user to edit the target pose. The user selects the regions of interest with our intelligent scribbling system, and then adjusts the orientation of the partition to edit the reference pose. The rotation axis is based on the screen coordinate system so that the operation can be more intuitive to the user. In addition, the target mesh is solved in real-time to achieve interactive editing.

4 Results

We used our system to create several interesting animations by assembling existing animations. The program is written in C++ and run on a Pentium 4 3.4GHz PC, with 2Gb RAM. The timing data and model information we use in this paper are shown in Table 1. The animation sequence is solved efficiently because the linear system can be factored and stored in the pre-computation step. The interface for the user to edit the kinematic pose can be operated in real-time.

For example in Figure 4, we assembled Dino, King Kong, and Lizard animations to create a new interesting animation. The Dino first stomps its feet, beats its breast and waves its tail to show its majestic appearance. The animations are applied on different partitions at different times so that they can work well together and perform surprising results. Another example is shown in Figure 9. We compounded the running animation with the morph sequence [11,15] together so that the elephant mesh can morph into the horse while running. In this way, the two deformation gradients are multiplied together before the objective function is solved to obtain the running morphing sequence.

Our system can animate non-articulated models, as well. For example, we can combine different emotional and verbal expressions on the head model. In Figure 11, we have happy, angry and cry expressions as well as the verbal expressions on different head models. We assembled these expressions together on David's head so that he can talk with different emotions. Because the head animations are not driven by a skeleton, it is meaningless to separate the mesh into stiff and flexible partitions. Therefore we do not need to use the intelligent scribbling system to refine the selected regions, but just blend the two animations based on the geodesic distance. The user chooses a reasonable threshold to determine the region that grows from the boundary. The system interpolates the weights for those regions. In this manner, the mesh is separated into three partitions, selected, unselected, and the intermediate partitions. Similar to what we mentioned in Section 3.2.2, we set all of the faces in the selected partition as 1.0 and all of those in the unselected partition as 0.0. Equation 5 is solved to obtain the blending weights of the triangles in the intermediate partition, therefore blending the two expressions naturally.

Mesh	Triangle	Blending	Factorization	Solve
Monster	39998	2.312s	5.687s	0.156s
Dino	20000	1.063s	1.688s	0.078s
Elephant	85796	6.922s	11.574s	0.391s
David	64021	6.344s	7.922s	0.344s

Table 1 Model information and the timing statistic for blending weights, factorization and solving animation.

5 Conclusions and Limitations

We introduce a new approach to create animations that are harmonically assembled from several existing animations. The proposed system is designed for novice users, requiring only a little manual effort. The main idea behind our algorithm is animation reuse on partial meshes and therefore the shapes of different meshes are not required to be similar. We also propose a new approach for computing the correspondence, which can obtain reasonable results with fewer markers and consequently saving user's time. This simplification does not affect the results in our application because the deformation gradients in the stiff partitions are nearly identical.

Although different animations can be assembled smoothly in our system, there are still some remaining artifacts on the created animation. This is because the animation may not satisfy the momentum conservation requirement. Moreover, the constraint vertex is fixed in the same position through the entire sequence. The mesh only acts on different poses and always stays where it is, making the animation strange. Therefore, in our next research step, we will add kinematic factors to our animation system to solve these problems. In addition, the self-intersection problem still occurred. With improper assembling operations, different parts of the mesh will collide with others and artifacts occur. We take this problem into consideration in our future work.

Acknowledgements We would like to thank the anonymous reviewers for helping us to improve this paper. This work is supported by the Landmark Program of the NCKU Top University Project under Contract B0008 and is supported in part by the National Science Council under Contracts NSC-95-2221-E-006-193-MY2 and NSC-96-2628-E-006-200-MY3.

References

1. Allen, B., Curless, B., Popović, Z.: Articulated body deformation from range scan data. In: ACM Trans. Graph., vol. 21, pp. 612–619 (2002)
2. Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., Davis, J.: Scape: shape completion and animation of people. In: ACM Trans. Graph., vol. 24, pp. 408–416. ACM Press (2005)

3. Der, K.G., Sumner, R.W., Popović, J.: Inverse kinematics for reduced deformable models. In: ACM Trans. Graph., pp. 1174–1179. ACM Press (2006)
4. von Funck, W., Theisel, H., Seidel, H.P.: Vector field based shape deformations. In: ACM Trans. Graph., pp. 1118–1125. ACM Press (2006)
5. Guenter, B., Grimm, C., Wood, D., Malvar, H., Pighin, F.: Making faces. In: In Proceedings of ACM SIGGRAPH 1998, pp. 55–66. ACM Press (1998)
6. Guo, Z., Wong, K.C.: Skinning with deformable chunks. Computer Graphics Forum **24**(3), 373–381 (2005)
7. Herda, L., Fua, P., Plankers, R., Boulic, R., Thalmann, D.: Using Skeleton-Based Tracking to Increase the Reliability of Optical Motion Capture. Human Movement Science Journal **20**(3), 313–341 (2001)
8. Huang, J., Shi, X., Liu, X., Zhou, K., Wei, L.Y., Teng, S.H., Bao, H., Guo, B., Shum, H.Y.: Subspace gradient domain mesh deformation. In: ACM Trans. Graph., vol. 25, pp. 1126–1134. ACM Press (2006)
9. Hyun, D.E., Yoon, S.H., Chang, J.W., Seong, J.K., Kim, M.S., Jüttler, B.: Sweep-based human deformation. The Visual Computer **21**(8-10), 542–550 (2005)
10. Igarashi, T., Moscovich, T., Hughes, J.F.: As-rigid-as-possible shape manipulation. In: ACM Trans. Graph., pp. 1134–1141. ACM Press (2005)
11. Lee, T.Y., Huang, P.: Fast and institutive polyhedra morphing using smcc mesh merging scheme. EEE Transactions on Visualization and Computer Graphics **9**(1), 85–98 (2005)
12. Lee, T.Y., Lin, C.H., Wang, Y.S., Chen, T.G.: Animation key-frame extraction and simplification using deformation analysis. IEEE Transactions on Circuits and Systems for Video Technology **18**(4) (2008)
13. Lee, T.Y., Wang, Y.S., Chen, T.G.: Segmenting a deforming mesh into near-rigid components. Vis. Comput. **22**(9), 729–739 (2006)
14. Lewis, J.P., Cordner, M., Fong, N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In: SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 165–172. ACM Press/Addison-Wesley Publishing Co. (2000)
15. 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)
16. Mohr, A., Gleicher, M.: Building efficient, accurate character skins from examples. In: ACM Trans. Graph., vol. 22, pp. 562–568. ACM Press (2003)
17. Nealen, A., Sorkine, O., Alexa, M., Cohen-Or, D.: A sketch-based interface for detail-preserving mesh editing. In: ACM Trans. Graph., vol. 24, pp. 1142–1147. ACM Press (2005)
18. Noh, J.Y., Neumann, U.: Expression cloning. In: In Proceedings of ACM SIGGRAPH 2001, pp. 277–288. ACM Press (2001)
19. Park, S.I., Hodgins, J.K.: Capturing and animating skin deformation in human motion. In: ACM Trans. Graph., vol. 25, pp. 881–889. ACM Press (2006)
20. Shoemake, K., Duff, T.: Matrix animation and polar decomposition. In: Proceedings of the conference on Graphics interface '92, pp. 258–264. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1992)
21. Sorkine, O., Cohen-Or, D.: Least-squares meshes. In: Proceedings of Shape Modeling International, pp. 191–199. IEEE Computer Society Press (2004)
22. Sorkine, O., Lipman, Y., Cohen-Or, D., Alexa, M., Rössl, C., Seidel, H.P.: Laplacian surface editing. In: Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp. 179–188. Eurographics Association (2004)
23. Sumner, R.W., Popović, J.: Deformation transfer for triangle meshes. In: ACM Trans. Graph., pp. 399–405. ACM Press (2004)
24. Sumner, R.W., Zwicker, M., Gotsman, C., Popović, J.: Mesh-based inverse kinematics. In: ACM Trans. Graph., pp. 488–495. ACM Press (2005)
25. Weng, Y., Xu, W., Wu, Y., Zhou, K., Guo, B.: 2d shape deformation using nonlinear least squares optimization. Vis. Comput. **22**(9), 653–660 (2006)
26. Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., Shum, H.Y.: Mesh editing with poisson-based gradient field manipulation. In: ACM Trans. Graph., pp. 644–651. ACM Press (2004)
27. Zhou, K., Huang, J., Snyder, J., Liu, X., Bao, H., Guo, B., Shum, H.Y.: Large mesh deformation using the volumetric graph laplacian. In: ACM Trans. Graph., pp. 496–503. ACM Press (2005)

Yu-Shuen Wang received his B.S. from National Cheng Kung University, Tainan, Taiwan, Republic of China, 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.

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, National Cheng-Kung University, Tainan, Taiwan. He is an associate editor of the IEEE Transactions on Information Technology in Biomedicine from 2007 to 2010. He is also on the editorial advisory board of the Journal Recent Patents on Engineering, an editor of the Journal of Information Science and Engineering and a region editor of the Journal of Software Engineering. His current research interests include computer graphics, nonphotorealistic rendering, image-based rendering, visualization, virtual reality, surgical simulation, medical visualization and medical system, and distributed and collaborative virtual environments. He leads the Computer Graphics Group, Visual System Laboratory, National Cheng-Kung University http://graphics.csie.ncku.edu.tw/). He is a member of the IEEE and the ACM.

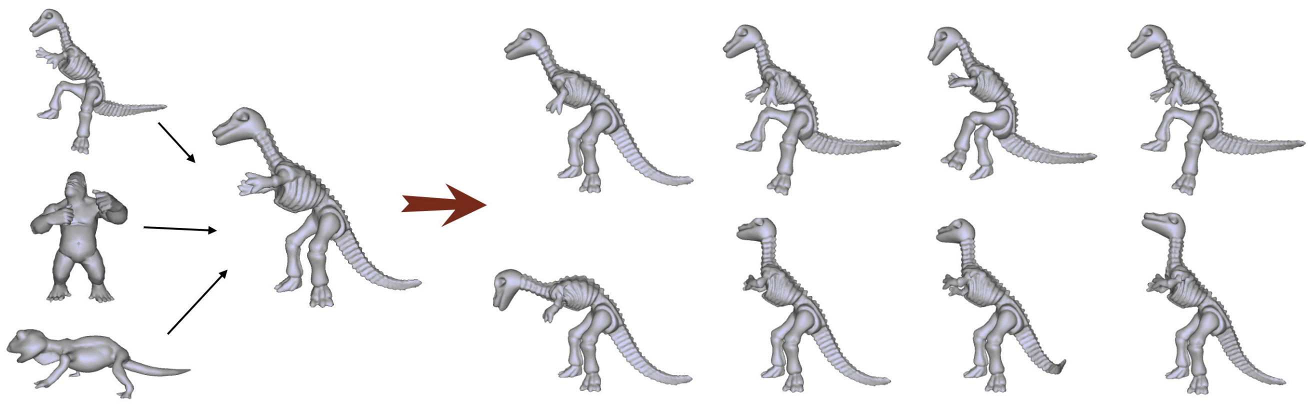

Fig. 9 The Dino animation is assembled from the animations of King Kong's torso, the Lizard's tail and itself.

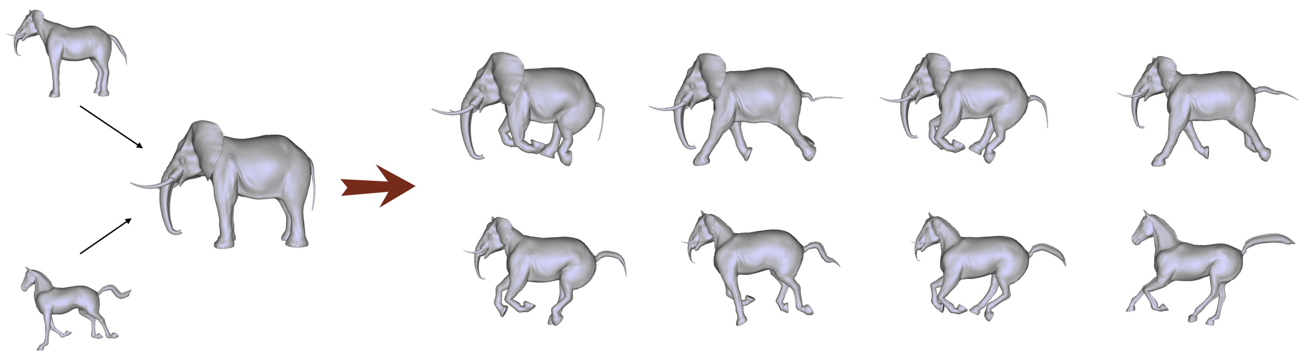

Fig. 10 We compound the deformation gradients of the morphing sequence and the running sequence, so that the elephant can morph to the horse while running.

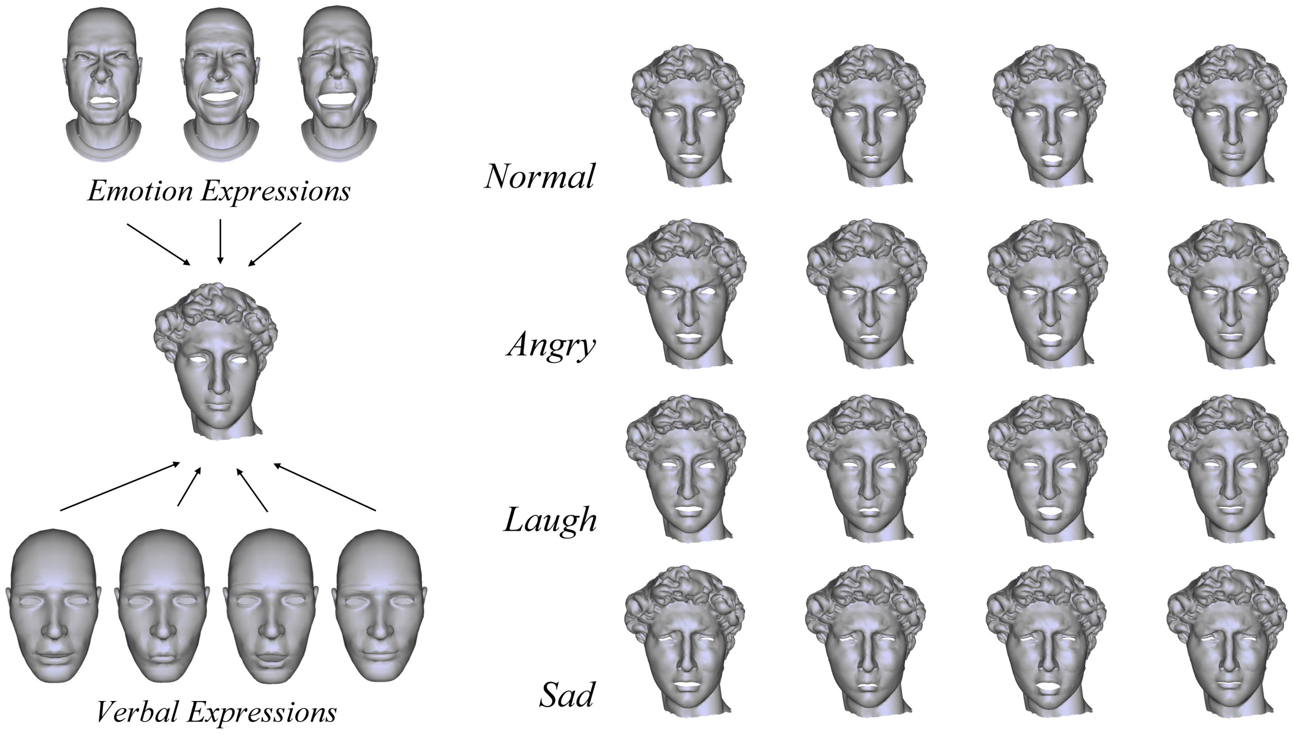

Fig. 11 The emotion and the verbal expressions are assembled on David's head to perform the talking animation with different emotions.