Skeleton-driven Animation Transfer based on Consistent Volume Parameterization

Yen-Tuo Chang[1], Bing-Yu Chen[2], Wan-Chi Luo[1], and Jian-Bin Huang[1]

National Taiwan University
[1]{draphix,maggie,azar}@cmlab.csie.ntu.edu.tw, [2]robin@ntu.edu.tw

Abstract. To edit or create the animation of a 3D character model has always been an important but time-consuming task, since the animator usually needs to set up the character's skeleton, paint its binding weights, and adjust its key-poses. Hence, we propose an animation transfer system in this paper to take a well-edited character animation as the input. Then, the system can transfer the skeleton, binding weights, and other attributes of the given character model to another static model with only few corresponding feature points specified. The transferring process is based on a mapping between the space around two character meshes. In this paper, the mapping is called consistent volume parameterization, which inherits consistent surface parameterization. Hence, the animator can start to create a skeleton-driven animation for the new character model without any prior setting. Moreover, our system is also capable of cloning a skeleton-driven animation to several other character models which can be used in a crowd animation.

1 Introduction

3D virtual characters are getting widely used in movies with special visual effects, computer generated animations, computer games, etc. Animator brings a 3D character model to life by making plausible and lifelike motions, and one of the common solutions is to set up the character's skeleton, paint its binding weights on the surface, and adjust its key-poses. The skeleton of a character model consists of a set of bones, which connect each other with a set of joints. Its binding weights define the binding relationship between its skin (surface of the model) and the skeleton. This relationship specifies the degree of dependencies of each vertex on the surface to a set of bones of the skeleton. After the skeleton and binding weights are well set up, the animator can edit the key-poses by adjusting the skeleton, then the mesh surface of the character model is deformed with the adjusted skeleton according to the binding weights. By interpolating the key-poses, a skeleton-driven animation is generated.

Besides creating an animation of one 3D character model, a crowd animation, which includes vast amount of 3D character models in a scene, is also widely used, such as hundreds of guests dancing simultaneously in a royal hall, or hundreds or thousands of soldiers fighting in a battle. As we mentioned above, the process of creating or editing a character animation is a time-consuming task. To

create a crowd animation including huge amount of different character models is thus extremely tedious. Although we can create only few character models and repeatedly put the same set of models everywhere in the scene to form the crowd animation, this method leads to low quality of visual feelings.

Therefore, in this paper we propose a system to transfer the skeleton of a given 3D character model to any other models which originally have only mesh data. Furthermore, the motion data well-edited previously for the given character model could also be transferred to the target models, such as binding weights, key-poses, and texture coordinates, as shown in Fig. 2. Hence, our system makes it much easier to create a huge amount of character models performing the same actions in a scene. Besides, the animators can also import the transferred skeleton, binding weights, and other data to an animating software for further editing. The whole system is fully automatic except few feature points are specified by the user to establish the correspondence between the source model and the target one at the beginning.

In order to transfer the skeleton from the source character model to the target one, a mapping between two mesh surfaces is first required. We use cross parameterization proposed by Kraevoy and Sheffer in [1] to construct this mapping relationship. Generally, parameterization refers to mapping a geometry to a domain of lower dimension, and makes it easier to be processed. We consistently parameterize two mesh surfaces to the same base domain, which is basically a simplicial complex consists of all the user-specified feature points. The system then automatically partitions and cross-parameterizes the surfaces of the two models. Right after the correspondence between the two surfaces is established, the consistent volume parameterization, which is a mapping from the space of the source model body to that of the target one, is generated by using the 3D mean value coordinates adapted from [2]. The space of a 3D model body is defined as the space around and inside the model body in this paper. For each joint of the source model skeleton, which usually lies inside the mesh surface, a corresponding 3D position is found for the target model by applying a smooth and continuous function. The skeleton of the source model is thus appropriately transferred to the target model. Then, the key-poses can be also transferred through the skeleton transferring. The binding weights and other attributes of the source model, such as texture coordinates, can be transferred through consistent surface parameterization which has already been established before generating consistent volume parameterization.

2 Related Work

In this paper, we take the advantages from several parameterization methods to develop a best way to construct the correspondence between two or more models. In order to parameterize a 3D mesh with or without holes to a 2D domain, most of the methods cuts the mesh surfaces into some patches which are homeomorphic to disks. Eck *et al.* [3] proposed a classic method to parameterize a model of any topology. The triangular mesh is partitioned using Voronoi diagram and

Delaunay triangulation. However, this method is hard to adapt into consistent parameterization because the Voronoi sites are selected randomly. Zhou *et al.* [4] partitioned the mesh without any user specified features. Their system analyzes the spectrum information along the surface and automatically cut the mesh into some charts according to a stretch-minimizing criterion. Although this approach gained both the advantage on efficiency and the stretch-minimization of parameterization, it is still rather hard to be applied to make correspondence between two or more models.

Cross parameterization or inter-surface mapping refers to the mapping between two models. Lee *et al.* [5] used MAPS [6] to get the base domains of two models after some corresponding feature points and lines are specified by the user. They then created an inter-surface parameterization between the two models by mapping the two models to their base domains first, and constructing a correspondence between the two base domains with user assistance. Praun *et al.* [7] provided consistent mesh parameterization to get the inter-surface mapping. In their method, the user first specifies a common base domain and maps it to all models manually. Then, the consistent parameterization can be done by subdividing the base domain for each model. Schreiner *et al.* [8] established a common base domain of two models based on the corresponding feature points specified by the user. They created the common base domain by linking all common feature pairs together, then used progressive meshes [9] with constraints to create multiresolution meshes of the path networks and used a coarse-to-fine mapping optimization method to find a continuous mapping between the multiresolution meshes. Kraevoy and Sheffer [1] created the common base domain in a similar way as in [8]. They then used mean value parameterization to map the vertices of a model to their belonging triangle on the common base domain. The inter-surface mapping was finally constructed through mapping the base domains.

In the topic of representing a position in 3D space by other primitives, convex combination is usually used for parameterization in several different approaches. It means that every vertex can be represented as a convex combination of its neighboring vertices. Ju *et al.* [2] applied the mean value coordinate [10] into triangular mesh parameterization, and made it possible to represent a vertex in a closed 3D mean-value domain.

For transferring the animation to other static models, Sumner and Popović [11] proposed a method to transfer the deformation by first matching the features of the inputs and then transforming each triangle with this mapping. Bregler *et al.* [12] transferred the motion of a 2D character animation to other 3D models by analyzing its affine transformations and key-poses on the plane. Zhou *et al.* [13] turned a 3D static model into a solid wire-frame, then the deformation can be edited according to graphical Laplacian. Their system also accomplishes the transfer of planar deformation from 2D cartoons. However, all these animation transfer methods are not able to generate the skeleton or other animation data for the target static model, to use the transferred animation for further editing is not easy. Allen *et al.* [14] also presented a method which is

able to transfer the skeleton of a 3D human model to other ones. Their method records the position of each joint with only two or three vertices around, thus it may introduce great distortion in most of our cases.

3 System Overview

As shown in Fig. 4, our system takes the animated source model and a static target one as the input (upper-left two squares in the figure), and requires the user to specify some corresponding feature points as the first step of the whole processes (upper-middle). The rest processes of our system are fully automatic. The common base domain derives from the corresponding feature points on the surface of each model (middle blue-shaded figure in upper-right rectangle). Then, the two input meshes are partitioned and mapped to the common base domain to construct the consistent surface parameterization (upper-right). We then apply the 3D mean value coordinates to achieve the consistent volume parameterization using the constructed consistent surface parameterization (middle-left). Due to the consistent volume parameterization, the skeleton of the source model can be transferred to the target one (middle-right), and the binding weights and other attributes are also copied according to the consistent surface parameterization to complete the animation transfer process (lower two figures).

4 Consistent Surface Parameterization

To construct the consistent surface parameterization of two meshes is to find a correspondence map $\mathbf{M}_{ts}$, so that for each vertex $v_i^t \in \mathbf{V}_t$, $i = 1, ..., N_t$ of a *target* mesh M_t with N_t vertices can have a meaningful position $\mathbf{M}_{ts}(v_i^t)$ on the surface of a *source* mesh M_s with N_s vertices, where $\mathbf{V}_t$ is a set of vertices of M_t. The correspondence position $\mathbf{M}_{ts}(v_i^t)$ may be a vertex $v_i^s \in \mathbf{V}_s$, $i = 1, ..., N_s$ of M_s or inside a triangle of M_s which consists of three vertices in $\mathbf{V}_s$. The quality of the consistent surface parameterization $\mathbf{M}_{ts}$ depends on how much the new position $\mathbf{M}_{ts}(v_i^t)$ of each vertex v_i^t signifies its original position.

The first step of our system is to specify the corresponding feature points of two models manually by the user through a typical user interface as shown in Fig. 3. The feature points should be the vertices of the two models. Hence, we can define a correspondence map of U pairs of the common feature points as $\mathbf{M}_{ts}(v_i^t) = v_i^s$ for $i = 1, ..., U$, where $U < N_t, N_s$, and vise versa, i.e., $\mathbf{M}_{st}(v_i^s) = v_i^t = \mathbf{M}_{ts}^{-1}(v_i^s)$. After specifying U pairs of the common feature points on the surfaces of the two models, we adapt Kraevoy and Sheffer's method [1] to create the cross parameterization $\mathbf{M}_{ts}$ of the two models for the rest vertices v_i^t, $i = U + 1, ..., N_t$ of M_t.

Before constructing the cross parameterization of the two models, we first connect the feature points to establish the base domains B_s and B_t of the two models M_s and M_t, respectively. The base domains are required to be consistent, which means that there are one-to-one correspondences among the vertices and

paths of the two base domains. To connect two feature points, we use the Dijkstra's algorithm to trace the shortest paths along the edges between two feature points on the surface of the mesh and add some Steiner points to subdivide the edges while tracing if necessary.

When connecting the feature points, we also keep the consistence of the corresponding paths on both meshes in each tracing step. Hence, when we connect two feature points v_i^t and v_j^t of M_t, we must connect their corresponding feature points v_i^s and v_j^s of M_s to check if the two paths are both valid to add to the base domains B_s and B_t, where $i, j = 1, ..., U$. To identify a path is valid or not, we test if it intersects any existing path. The cyclical order of the paths emitted from one feature point is also compared to those of its corresponding feature point on the other model to check if they are consistent. The same-sidedness of a vertex relative to a path is also examined on both models to ensure the correctness of the topology. If there is no valid path when connecting two feature points, we will try to add some Steiner points to the faces between the two feature points, so that a valid path can be found for the two feature points.

Finally, the base domains B_s and B_t of the two models M_s and M_t are constructed and guaranteed to be consistent. Then, the two models are partitioned to be some patches and the number of patches of the two models are the same as shown in Fig. 5. Moreover, every patch consists of three feature points and three paths connected the three feature points, so that when replacing the paths with straight lines, the base domains consist of some planar triangles as shown in Fig. 6.

The partitioned model is then parameterized to its base domain. Initially, a mean value parameterization approach in [10] is applied to every patch and put all triangles of the patch onto the corresponding planar triangle of the base domain. The corner vertices are fixed on the feature location of the base domain, while the interior vertices are placed with barycentric coordinates. The algorithm then applies a smoothing step to improve the distribution of the parameterization by refining the shapes of the patches. To avoid too sparse or too dense distribution on a parameterization which introduces large distortion, the vertices are shifted to its neighboring patches and thus relax the unbalancing on the surface.

Finally, we establish the mapping $\mathbf{M}_{ts}^B$ between the two consistent base domains B_t and B_s of the two models M_t and M_s by just mapping their patches one-by-one. Hence, the mapping $\mathbf{M}_{ts}$ between the target model M_t and the source one M_s can be constructed through the following equation:

$$\mathbf{M}_{ts} = \Pi_s^{-1} \cdot \mathbf{M}_{ts}^B \cdot \Pi_t,$$

where Π_t refers to the parameterization between model M_t and its base domain B_t, and so does Π_s.

5 Consistent Volume Parameterization

After the consistent surface parameterization between two models is established, in this section, a mapping called consistent volume parameterization is then

constructed for the 3D space of two models. For representing the volumetric space around a surface model, we adapt the 3D mean value coordinates proposed by Ju *et al.* [2]. The original 2D mean value coordinates was proposed by Floater [10], which is used to construct a continuous distribution of an attribute value $f[x]$ of a weighting function f for all points x inside or outside a planar polygon. The value can be any kind of data such as color, heat, or texture coordinate. Ju *et al.* extended this approach to compute the 3D mean value coordinates inside a closed triangular mesh.

The mean value coordinates has the following general form:

$$f[x] = \frac{\sum_{i=1}^{n} w_i f_i}{\sum_{i=1}^{n} w_i},$$

where $f[x]$ is the interpolated value, w_i and f_i are the weight and attribute value of each vertex v_i of the closed triangular mesh M. By applying this algorithm, a set of weights w_i are determined to represent the relationship between a specific position x inside the mesh M and all the triangles that have a projection area on the unit sphere of x. The value f_i on the vertices v_i of all these triangles are interpolated according to the weights w_i, and then the value $f[x]$ for x can be obtained.

To construct the consistent volume parameterization is to define a correspondence map $\mathbf{M}_{ts}^{V}$, so that for each point x^t inside or closely outside the target closed triangular mesh M_t can have a meaningful position $x^s = \mathbf{M}_{ts}^{V}(x^t)$ inside or closely outside the source closed triangular mesh M_s. Hence, we define the weighting function f_i^s for each vertex v_i^s of the source model M_s as the mapping position of x^t, then we can obtain the mapping from x^t to x^s through f_i^s. According to the consistent surface parameterization,

$$f_i^s = \sum_{j=1}^{m} \alpha_j \mathbf{M}_{ts}(f_j^t),$$

where α_j is the weights for barycentric coordinate and $\sum_{j=1}^{m} \alpha_j = 1$, $m = 1$ or 3. If v_j^t is mapped to a vertex v_i^s, m is set to be 1, i.e., $v_i^s = \mathbf{M}_{ts}(v_j^t)$, otherwise m is set to be 3, since v_i^s is mapped by a triangle of M_t. Using the 3D mean value coordinates:

$$f[x^s] = \frac{\sum_{i=1}^{n} w_i f_i^s}{\sum_{i=1}^{n} w_i},$$

we define a mapping function $x^s = \Omega_s(f_i^s)$ from the vertices v_i^s on the surface of M_s to one of their interior points x^s through the 3D mean value coordinates.

Thus, the consistent volume parameterization is established by interpolating the mapping value generated by the consistent surface parameterization, and finding a corresponding position x^t around M_t for any specified position x^s around M_s by

$$x^s = \Omega_s(\sum_{j=1}^{m} \alpha_j \mathbf{M}_{ts}(f_j^t)),$$

and $f_j^t = \Omega_t^{-1}(x^t)$. Hence, the mapping $\mathbf{M}_{ts}^{V}$ between the space of the target model M_t and that of the source one M_s can be constructed through the following

equation:

$$\mathbf{M}_{ts}^{V} = \Omega_s \cdot \mathbf{M}_{ts} \cdot \Omega_t^{-1}.$$

As shown in Fig. 7, the consistent volume parameterization is constructed between the two models. For each point of the source (fat man on left side) model, we can find its corresponding point around the target (Mario on right side) model. The points inside the models are corresponding joints. Other places inside the models are also mapped smoothly.

Although we call this method as consistent volume parameterization, it does not always find a mapping position inside the target mesh for a joint inside the source. The exception may happen when the source mesh is convex but the target is not. The mapped position may be closely outside the target surface, but it still belongs to the volumetric space of the character. Fortunately, the joints of a skeleton are not necessary to be all inside the mesh surface. The transferred animation still looks well under this circumstances.

6 Animation Transfer

To transfer the animation data from a skeleton-driven source animation model to a target static one, we first transfer the skeleton of the source animation model through the consistent volume parameterization of the two models. Then, the binding weights and other attributes of the source animation model can be transferred to the target static one through the consistent surface parameterization of the two models. Finally, the key-poses are transferred by applying the same animation data to the generated skeleton of the target model as that of the source animation model. If an animator wishes to modify the transferred animation, he or she can use model editing tools to edit the transferred skeleton, binding wights, key-poses, or other attributes of the target model.

6.1 Skeleton Transfer

The skeleton of a 3D character model is defined to be a set of "joint and bone" pairs. The definition basically imitates the skeleton structure of a vertebrate, in which bones are rigid sticks and connect each other by a set of joints. Each joint has a rotation vector representing the current direction which the bone connected to it points toward. Therefore, transferring a skeleton is identical to transferring the positions of all the joints of a given skeleton to their new positions, and connects them with bones according to the structure of the original one.

Our system automatically finds the corresponding position of each joint inside the target model through the consistent volume parameterization. After the skeleton is transferred from the source animation model to the target static one, the skeletons of the two models are consistent which means there are one-to-one correspondences among their joints and bones. Hence, the key-poses of the source animation model can easily be transferred by assigning the same motion data, such as the rotation angle of a joint of the source model, to its corresponding joint of the target model as shown in Fig. 8.

Source	#vertex/#face of Source	Target	#vertex/#face of Target	#feature	time for partitioning (sec.)	time for parameterization (sec.)
dog	4070/8136	cat	2702/5400	17	50	67
fat man	3426/6848	Mario	2934/5864	25	107	17
dog	4070/8136	pig	5570/11136	22	276	23

Table 1. The performance testing of four pairs of animation transfer, including the time for partitioning and parameterization, which are the two main steps of constructing the consistent surface parameterization. The numbers of vertices and faces of the source and target models and the number of corresponding feature points specified on the two models are also listed. The dog and cat models are shown in Fig. 10 (upper and middle), the fat man and Mario models are shown in Fig. 1 (left and right), and the dog and pig models are shown in Fig. 11 (upper and middle).

6.2 Binding Weights Transfer

The binding weights, like other surface attributes such as texture coordinate, are assigned to the vertices on the surface of the model. Hence, we can transfer the binding weights from the source model to the target one through the consistent surface parameterization as shown in Fig. 9.

7 Results

Fig. 10 and Fig. 11 show the animation transfer results from a black dog model to a cat model, a brown dog model, a pig model, and a can model, which originally have only mesh data. Hence, our system works well when the target model has a shape similar to the source one, as was expected.

According to the experiments, our system processes with a desirable efficiency in all cases. For a novel user who can use a 3D model viewing system with a mouse, he or she can use our system with nearly no training in advance. As shown in Table 1, $15 \sim 30$ features are required to produce a good parameterization. This takes about $5 \sim 10$ minutes in specifying the corresponding feature points on the surface of the two models. After the feature specifying stage, our system is fully automatic. In all cases, the bottleneck of performance is the construction of the consistent surface parameterization. Besides this stage, the rest processes of our system is typically performed in real-time. As shown in Table 1, the construction of the consistent surface parameterization consists of partitioning two models consistently according to the corresponding feature points, and parameterizing the patches to the base domain. Although the processing time of the construction of the consistent surface parameterization requires from seconds to few minutes, this is still extremely tolerable to the user, since it may be an off-line process. The testing platform is a desktop PC with an Intel Pentium 4 3.4GHz CPU and 1.5GB memory.

8 Conclusions and Future Work

In this paper, we propose a system to transfer the skeleton and animation data from a source animation model to a target static one which has only mesh data. The system requires the user to mark few feature points, and then generates the output skeleton and animation data for the target model automatically. The skeleton has a nice shape and can be edited to produce other animation by the animator by importing the generated skeleton and animation data to a model editing tool. Our system runs in a range from 30 seconds to 5 minutes, thus enormously saves the time and work for the animators.

For our future work, there are some possible ways to improve the efficiency and quality of our system. Surazhsky *et al.* [15] proposed a robust algorithm to trace the geodesic directly on the mesh surface, regardless of the topology below. It is possible to modify our system to partition the meshes according to this method. From our experiments, the initial poses of the input models affect the quality of the transferred animation greatly. Inconsistency in initial poses may distort the distribution and default angles of each joint, and thus produce animation that is not concurrent to the source. We can add one more step to automatically adjust the pose of the target model to fit that of the source after transferring the skeleton. The adjusted model could be treated as a new input with the new initial pose, so the system may redo the consistent surface parameterization process again and find some more accurate position for all the joints, as long as the output animation. Although the feature specification is regarded necessary, it is still desirable to derive some methods to automatically detect and specify the common feature points.

Finally, the consistent volume parameterization may benefit a lot of areas in computer graphics. It is possible to transfer the inside structure of a model to another, thus may reduce many work in 3D modeling. It can also be used to transfer the internal texture, layered information, or the structure needed to shade with translucency.

Acknowledgements

This work was supported in part by the National Science Council of Taiwan under Grant No. NSC93-2213-E-002-084 and NSC94-2213-E-002-097.

References

1. Kraevoy, V., Sheffer, A.: Cross-parameterization and compatible remeshing of 3d models. ACM Transactions on Graphics (SIGGRAPH 2004 Conference Proceedings) **23**(3) (2004) 861–869
2. Ju, T., Schaefer, S., Warren, J.: Mean value coordinates for closed triangular meshes. ACM Transactions on Graphics (SIGGRAPH 2005 Conference Proceedings) **24**(3) (2005) 561–566

3. Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., Stuetzle, W.: Multiresolution analysis of arbitrary meshes. In: ACM SIGGRAPH 1995 Conference Proceedings. (1995) 173–182
4. Zhou, K., Synder, J., Guo, B., Shum, H.Y.: Iso-charts: stretch-driven mesh parameterization using spectral analysis. In: Proceedings of the 2004 Eurographics Symposium on Geometry Processing. (2004) 45–54
5. Lee, A.W.F., Dobkin, D., Sweldens, W., Schröder, P.: Multiresolution mesh morphing. In: ACM SIGGRAPH 1999 Conference Proceedings. (1999) 343–350
6. Lee, A.W.F., Sweldens, W., Schröder, P., Cowsar, L., Dobkin, D.: Maps: multiresolution adaptive parameterization of surfaces. In: ACM SIGGRAPH 1998 Conference Proceedings. (1998) 95–104
7. Praun, E., Sweldens, W., Schröder, P.: Consistent mesh parameterizations. In: ACM SIGGRAPH 2001 Conference Proceedings. (2001) 179–184
8. Schreiner, J., Asirvatham, A., Praun, E., Hoppe, H.: Inter-surface mapping. ACM Transactions on Graphics (SIGGRAPH 2004 Conference Proceedings) **23**(3) (2004) 870–877
9. Hoppe, H.: Progressive meshes. In: ACM SIGGRAPH 1996 Conference Proceedings. (1996) 99–108
10. Floater, M.S.: Mean value coordinates. Computer Aided Geometric Design **20**(1) (2003) 19–27
11. Sumner, R.W., Popović, J.: Deformation transfer for triangle meshes. ACM Transactions on Graphics (SIGGRAPH 2004 Conference Proceedings) **23**(3) (2004) 399–405
12. Bregler, C., Loeb, L., Chuang, E., Deshpande, H.: Turning to the masters: motion capturing cartoons. In: ACM SIGGRAPH 2002 Conference Proceedings. (2002) 399–407
13. Zhou, K., Huang, J., Snyder, J., Liu, X., Bao, H., Guo, B., Shum, H.Y.: Large mesh deformation using the volumetric graph laplacian. ACM Transactions on Graphics (SIGGRAPH 2005 Conference Proceedings) **24**(3) (2005) 496–503
14. Allen, B., Curless, B., Popović, Z.: The space of human body shapes: reconstruction and parameterization from range scans. ACM Transactions on Graphics (SIGGRAPH 2003 Conference Proceedings) **22**(3) (2003) 587–594
15. Surazhsky, V., Surazhsky, T., Kirsanov, D., Gortler, S.J., Hoppe, H.: Fast exact and approximate geodesics on meshes. ACM Transactions on Graphics (SIGGRAPH 2005 Conference Proceedings) **24**(3) (2005) 553–560

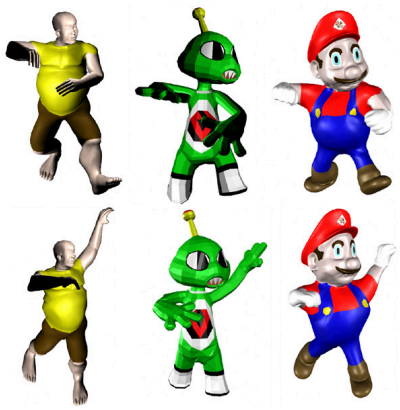

Fig. 1. The motion of the fat-man model (left) is transferred to the alien (middle) and Mario (right) models.

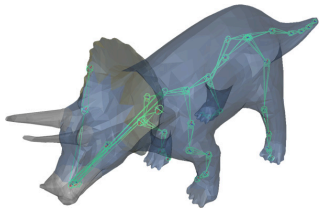

Fig. 2. The transferred skeleton and binding weights.

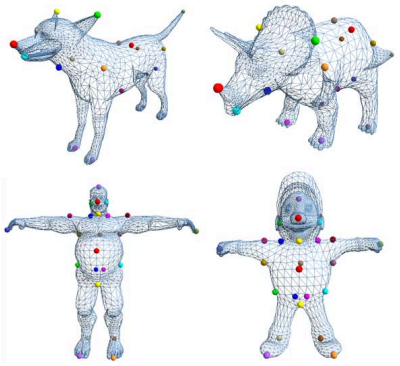

Fig. 3. Few pairs of feature points are specified on the surfaces of the dog (upper-left) and triceratops (upper-right) models, and also for the fat-man (lower-left) and Mario (lower-right) models.

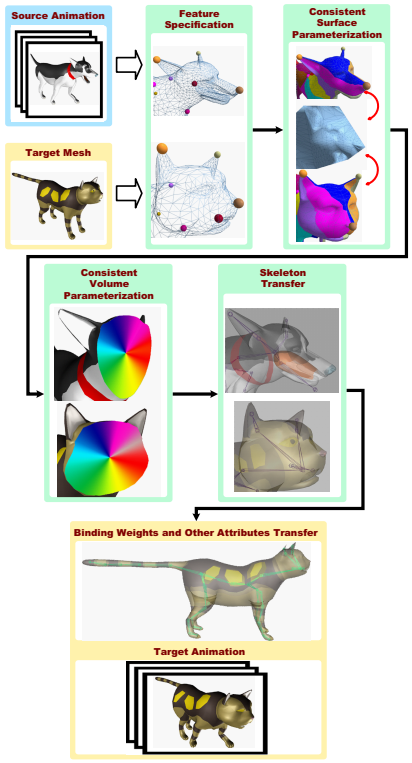

Fig. 4. System flowchart of our method.

Fig. 5. The meshes are partitioned into some patches according to the paths of the base domain. This leads to a meaningful mapping between the segments of the dog (upper-left) and triceratops (upper-right) models, and also for the fat-man (lower-left) and Mario (lower-right) models.

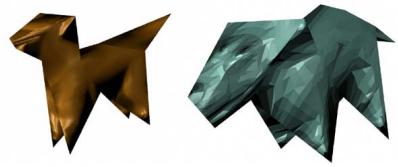

Fig. 6. The triangular patches are parameterized onto the base domain and form a pair of coarse meshes with the same topology of shape.

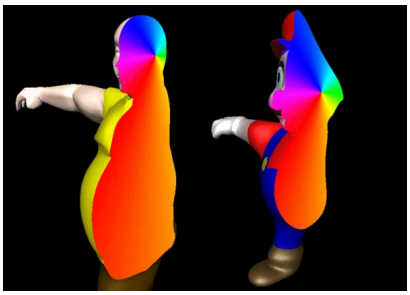

Fig. 7. The graphical representation of mapping a joint in the center of the head of the fat man model (left) to a corresponding position inside the Mario model (right) by applying the 3D mean value coordinates.

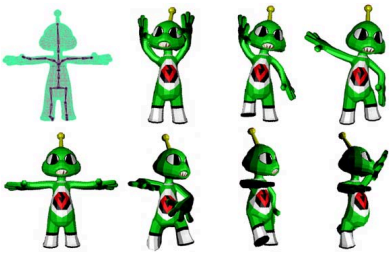

Fig. 8. A transferred skeleton-driven animation generated by our system. The skeleton is nicely shaped according to the character model (upper-left) and the motion can also be transferred from the fat man model shown in Fig. 1 (left).

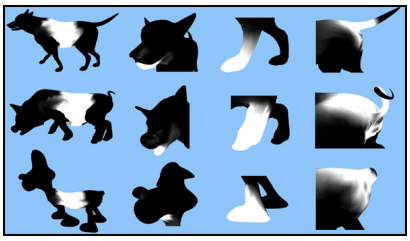

Fig. 9. The binding weights of the black dog model (upper) is transferred to the pig (middle) and brown dog (lower) models. The black dog model is shown in Fig. 10 (upper) and Fig. 11 (upper), the pig model is shown in Fig. 11 (middle), and the brown dog model is shown in Fig. 10 (lower).

Fig. 10. The motion of the black dog model (upper) is transferred to the cat (middle) and brown dog (lower) models.

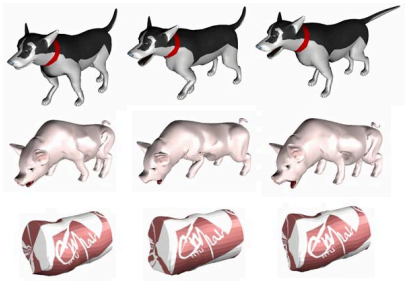

Fig. 11. The motion of the black dog model (upper) is transferred to the pig (middle) and can (lower) models.