Cloning Skeleton-driven Animation to Other Models

Wan-Chi Luo[†] Jian-Bin Huang[†] Bing-Yu Chen[‡] Pin-Chou Liu[†]
National Taiwan University
[†]*{maggie, azar, toby}@cmlab.csie.ntu.edu.tw* [‡]*robin@ntu.edu.tw*

Abstract*-3D animation has been manipulated widely in movies and video games nowadays. To make a 3D model move, traditionally, requires animators' efforts to edit the key poses of the model. It is a time-consuming task, especially when dealing with an imposing scene such as those full of different animals or soldiers. In this paper, we propose an efficient technique to clone skeleton-driven animation data from one to another model, including skeleton, binding weights and key-frame poses. With the proposed technique, users will only need to specify few common features between the source model and the target ones, and our system can transfer the animation automatically. The cloned animation can also be refined by adjusting either the cloned skeleton, binding weights, or key poses. In these settings, we can speed up the process of making crowd motion sequences and enable the reuse of animation.*

Keywords: Computer Animation, Skeleton Transfer, Animation of Crowds, Consistent Parameterization, Content Creation

1. Introduction

Models in movies or games usually have skeletons for easier motion editing, and are referred to "animatable models" since they carry animation data, whereas those without animation data will be called "static models" in this paper. Animators bring 3D static models to life by making plausible and life-like motions, and one of the common solutions is to build a skeleton, to set its binding weights with neighboring vertices, and then editing the key-poses.

Imagine there are more than one hundred dancers dancing uniformly. To generate such animation with various 3D body shapes, it is necessary to repeat the editing process over and over again, even for identical motion sequences. There are many researches focus on "motion retargeting" or "transferring motion captured data", but most of them need to define the skeleton and binding weights of the target model before transferring the source model's motion. Constructing the same skeleton structures for each dancer may be easy, but setting the binding weights is really a tedious task and may cost a lot of time even for experienced animators. Therefore, we propose an efficient technique which can clone a skeleton-driven animation from source model to target ones easily such that users will be able to generate similar animation data quickly, and refine it later if necessary.

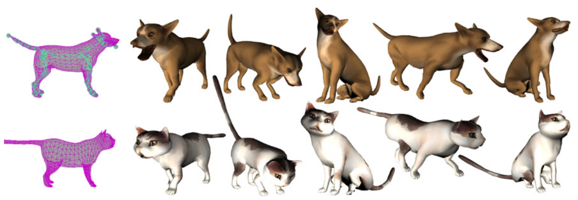
Figure 1. The motion of the dog model is transferred to the cat model.

In other words, if we have a running dog model and a static standing cat model, the result of our system will be a running cat model, as shown in Figure 1. Furthermore, we can not only clone the animation between two models with similar topologies, but also the models with different topologies, for example a monkey and a cola can.

The basic idea is that we have to construct a coherent skeleton structure for the target model first. Once we can derive the same skeleton structure and clone the animation data, the key-frames can be transferred directly. Because the target model only comprises mesh information, we have to disperse the skeleton information to nearby vertices on the mesh. As the descriptions in [3], we then utilize the correspondence of all vertices between the two models to reconstruct the skeleton. Moreover, the binding weights and texture information are also generally saved as the attributes of the vertices, so we can clone this information through the correspondence too.

One of the common ways to find the correspondence for all vertices between two models is surface parameterization. With this technique, users need to specify common features between two models, and to dissect both of the two models into homologous patches manually. We provide a user-friendly interface like other 3D-morphing programs to accomplish this control process easily. In order to transfer the skeleton data, we represent each joint of the skeleton as two vertices, or "markers", which form a numerical relations to the joint. Due to the consistent correspondence, we can find those markers on the target model and reconstruct the skeleton of the same structure. As soon as the animation data is transferred, our system can clone the animation from the source model to the target one.

Our major contribution is that we facilitate users by speeding up the process of making similar motion sequence for crowds, and by enable the reuse of animation data. Target models do not need to pre-define their skeletons and binding weights before cloning animation. We can transfer these two important attributes to target models automatically after specifying common features of source and target models.

2. Related Work

In this paper, we make users to dissect models into patches manually, and they also need to specify some common features in order to find the consistent correspondence. Each relative pair of patches is planar parameterized and aligned according to those common features. After overlaying the aligned embeddings, the correspondence of all vertices can be found. In our experiments, we can observe when the patch is more like a disk, when the planar parameterization has less fold-over problem, and when the correspondence is more correct.

Since an arbitrary 3D model may be closed or nonclosed, works have been published, discussing how to decompose a model into patches. Eck et al. [5] used the Voronoi diagrams and Delaunay triangulations to partition a model into several parts, but partitioning two models consistently is not easy because the sites are selected randomly. As for Normal Meshes, Guskov et al. [12] used a mesh simplification method presented by Garland and Heckbert [9] to create a base domain for only one model. Lee et al. [14] provided a 3D morphing method, with which users first manually assign the corresponding features of the source and target models, and then using MAPS (Multi-resolution Adaptive Parameterization of Surfaces) [15], the independent coarse base domain can be found through a simplification hierarchy. The parameterizations can thus be established on the merged base domain. The more the two models are dissimilar, however, the more user control will be needed to solve the correspondence problem. Katz and Tal [13] proposed an algorithm to dissect a model into meaningful patches by utilizing the combination of geodesic distance and angular distance. By applying maximum flow algorithm, they can also smooth the boundaries between patches. However, the decomposing algorithm was not proposed to find decompositions for compatible model, and the base domain founded by this method may not be consistent.

Praun and Hoppe [19] provided an algorithm performing consistent mesh parameterizations for several models. To get the consistent mesh parameterizations, the users have to specify a common base domain and manually map it to all of the models first. Then, the parameterizations can be accomplished by subdividing the base domain for separate models. Since cloning animation needs more precise correspondence for all vertices, especially the features, adopting the common base domain is hard to find for all cases because the numbers of features are diverse. This problem can be solved by constructing a common base domain for each pair of the source and target models, but may requires more efforts of users.

In the topic of planar parameterization, barycentric mapping, described by Tutte [23], defined every internal node as the baryceter of its neighbors. The shape of the mesh does not influence the position of the internal node, since the mesh connectivity is the only concern in this method. Floater [6] suggested a shape-preserving method to preserve chord length and barycentricity by using the combination of barycentric mapping, where each internal node is a convex combination of its neighbors. Eck et al. [5] proposed a discrete harmonic map method, which preserved aspect ratio of triangles. Moreover, Shlafman et al. [21] compared parameterization methods of ”barycentric”, ”shape-preserving”, and ”harmonic” according to various distortion measures, and ”harmonic mapping” emerged minimum distortion in their experiments. But when the patch is dissimilar to a disk, harmonic mapping resulted in fold-over severely. Desbrun et al. [4] and Lévy et al. [16] proposed different methods to compute discrete conformal mapping. With their technique, fold-over-free embeddings can be generated if the patch is not similar enough to a disk. To resolve the fold-over problem, we incorporate a spring method, and will discuss it later.

There are also many other studies focusing on spherical parameterization, which is an approach that can deal only with models that are genus-zero. One of the advantages of this method is that models do not need to be decomposed. Alexa [1] suggested a spring method to map a model onto a unit sphere, but the method only concerns about the connectivity of the mesh and causes high distortions. Then, Gotsman et al. [11] mapped a simple 3D model onto a unit sphere by solving a quadratic system. Praun and Hoppe [19] also proposed a coarse-to-fine algorithm to embed a model onto a unit sphere robustly, but the base domain of the model will limit the possibility of feature alignment.

In the topic of feature alignment, Alexa [1] proposed a method to align features on a unit sphere. It cannot guarantee, however, that those features can align to designated position. And finally, for feature alignment on planar embedding, many 2D image warping algorithms are proposed. Most of the studies used [8] to prevent the fold-over problem, and we also adopt this method to align features on embeddings. Alexa [2] and Floater and Hormann [7] presented exhaustive surveys, respectively.

Topic of transferring animation becomes important nowadays. Summer and Popović [22] proposed a good method to transfer deformation between triangle meshes. After getting the mapping between the triangles of the source and target models, they computed the affine transformation that encoded the ideal change of orientation, scale, and skew of each triangle. Then deformation transfer solved an optimization problem to maintain consistency.

3. Animation Data

In order to clone the animation sequence of a source animatable model to a target static model, we have to transfer all of the animation data, including binding weights, skeleton, and key-frame poses. There are more details can be founded in [17] and [18].

These animation data of source model can be motion captured data or constructed by animators. Generally, many studies can apply motion captured data to other characters. However, these studies need to pre-design the skeleton and binding weights of the target model. Since we know that skinning method may cause artifacts when twisting and bending, animators may design the skeleton and binding weights purposely, such as adding more extra joints, to prevent these artifacts. Both of them depend much on animators' experiences.

Moreover, although there are some powerful tools such as "Maya" provide the function to paint the binding weights for each joint on the mesh, it is really a tedious work. Since our method can transfer both the skeleton and binding weights to the target models. Users do not need to preprocess the target models except marking some common features.

4. Consistent Surface Parameterization

Because the binding weights are saved as a vertex's attributes, we need to get the correspondence of all the vertices between the source and target models. We use a planar parameterization method to find the correspondence of the two models.

4.1. Discrete Conformal Mapping

First, the user needs to assign the correspondence by partitioning each model into the same number of patches. In our system, the boundary of a patch can be represented by several vertices (i.e., called anchors) in sequence, and then it can be obtained by calculating the shortest path between two anchors on the mesh. After dissecting, since each pair of patches is disk-like, we mapped them to a plane by discrete conformal mapping [4]. The conformal mapping results are shown in Figure 2. Their algorithm is to solve a sparse linear system, which minimizes the combination of Chi Energy and Dirichlet Energy on triangulations.

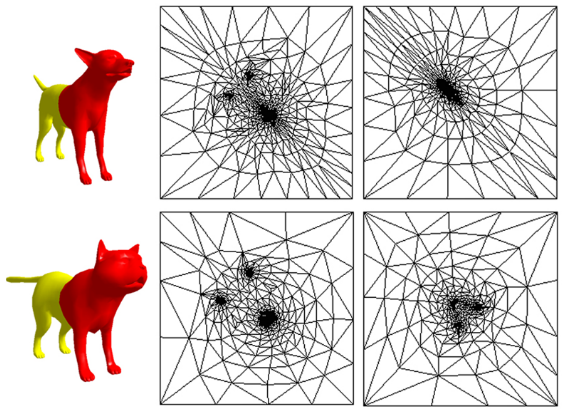

Figure 2. We dissect the dog and cat models into two correspondent patches, respectively. The right side rectangles show their conformal maps.

4.2. Relax Conformal Map

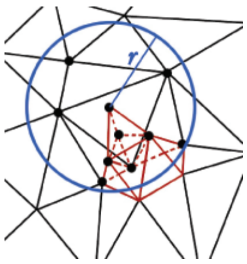

Figure 3. If a vertex is detected as a fold-oververtex, those vertices inside an effect radius *r* are needed to be relaxed.

However, if the patch of the model is not very similar to a disk, a conformal map without fold-over may not be found. Therefore, we combine the spring method described in [2] to relax those fold-over vertices after conformal mapping process. As shown in Figure 3, after making the conformal map, we detect which vertex of the conformal map is fold-over, and then each vertex v_i insides the effect radius r of the fold-over vertex needs to be performed spring relaxation. Moreover, each vertex v_i will move to

$$p_i = c \cdot \frac{\sum (v_i - v_j)\|v_i - v_j\|}{\sum \|v_i - v_j\|},$$

where v_j is the neighbor of v_i, and c is a constant (experimentally $c = 2$). The longer edge will be shortened so that the vertex v_i will be put in the center of its 1-ring neighborhood. This relaxation will not be terminated until no fold-over vertex is found. Owing to this relaxation algorithm does not concern the shape of the original mesh, the radius should not be too large and we only use it to solve the fold-over problem. Figure 4 shows the difference of using spring relaxation or not.

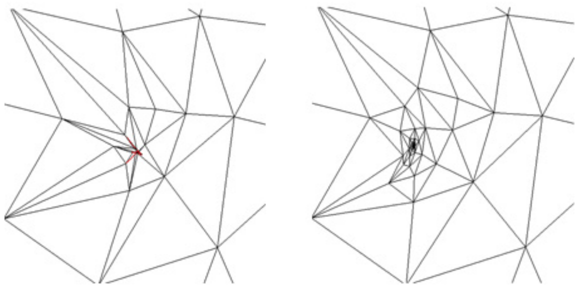

Figure 4. (a) An original conformal map. (b) A conformal map with relaxation.

4.3. Feature Alignment

We overlay each pair of patches' embeddings to find the correspondence of all vertices. Cloning animation to other models needs more precise mapping between all vertices than 3D morphing. If the correspondence is not precise enough, the binding weights will be transferred incorrectly and the deformation of the target model will be strange. On the purpose of computing better correspondence between each pair of homologous patches, we align the important features on the conformal map.

After creating the fold-over free conformal map for each pair of patches, the user needs to specify the common features of the source and target models manually or by other automatic or semi-automatic algorithms. In this paper, we use a fold-over free warping scheme [8] to align those features. Many researches in mesh morphing also use this method to align features on planar embeddings. It deserves to be mentioned that we do not need to change the edge connection when fold-over occurs, because we only want to retrieve the correspondent relation of the vertices.

4.4. Correspondence Representation

Given two aligned embeddings, we overlay them to find the correspondence of all vertices between the two models. There is no need to merge the faces, edges, and vertices as done in some traditional approaches of 3D metamorphosis, while we only need to know each vertex of the source model are laid at which face of the target model and its numerical relation. Assume that each vertex p of the target model's embedding lies in a triangle of the source model's embedding with 2D coordinates q_1, q_2, and q_3. The relation of p, denoted by $R(p)$, can be expressed using the barycentric coordinate:

$$R(p) = \frac{\Delta p q_2 q_3}{\Delta q_1 q_2 q_3} \times q_1 + \frac{\Delta q_1 p q_3}{\Delta q_1 q_2 q_3} \times q_2 + \frac{\Delta q_1 q_2 p}{\Delta q_1 q_2 q_3} \times q_3$$

where $\Delta q1q_2q_3$ denotes the area of the triangle consists of vertices q_1, q_2, and q_3.

5. Animation Cloning

In order to clone the animation of the source model to the target one or ones, we have to construct the same skeleton structure. Since the target model only has the mesh information, we have to disperse each joint position to vertices close to it. Allen et al. [3] proposed a method to transfer skeleton data. They chose two or three points on the mesh as markers for each joint, and then calculated the local positions of these markers in the associated joint's coordinate. As a result of the consistent parameterization, the corresponding positions of these markers on the other mesh can be derived, and the skeleton poses and bone lengths can be constructed using inverse kinematics. We utilize this method to re-construct the same structure of skeleton for the target static model.

Because we deal with 3D model with meshes instead of articulated figure, we solve, mainly, the problem of transferring animation data. The animation data including skeleton and binding weights generated by our system can also be imported into "Maya" for further refinements.

5.1. Skeleton Transfer

After surface parameterization, for each joint in the skeleton, we choose the nearest vertex to the joint as a marker A on the source model mesh. A 3D vector forming marker A to associated joint can intersect the source model mesh, and we call this intersection marker B. Markers A and B have a numerical relation to the associated joint. As shown in Figure 5, assume v_1 is marker A and v_2 is marker B, joint J_0 can be represented as $J_0 = \alpha \cdot v_1 + \beta \cdot v_2$, where

$$\alpha = \|v_2 - J_0\| / \|v_1 - v_2\|, \beta = \|v_1 - J_0\| / \|v_1 - v_2\| .$$

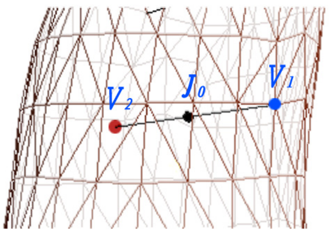

Figure 5. v_1 and v_2 are used to record J_0, where v_1 is the nearest vertex on the mesh, and v_2 is the intersect point of the mesh by $\overline{v_1 J_0}$.

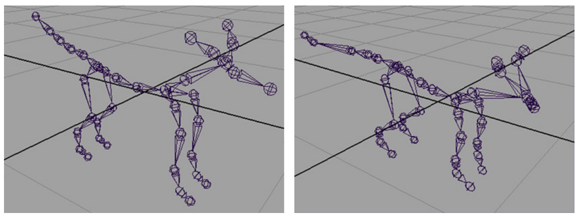

Figure 6. The skeleton transformation result shown in "Maya". (a) The skeleton of the dog is constructed by an animator, and (b) the skeleton of the cat is constructed by our system automatically.

We record marker *A*, marker *B*, α, and β to represent the associated joint information. According to the consistent parameterization, we can find relative marker *A'* and marker *B'* on the target model mesh. Afterward, the relative joint can be computed by calculating $\alpha \cdot A' + \beta \cdot B'$. Following calculating all joint positions in the target model, we can clone the joint connectivity as the source model's skeleton structure. Figure 6 shows the result of the transferred skeleton from a dog to a cat.

There are, however, some limitations, resulted from that we only choose two markers to record the joint's position, while marker *A* is defined as the vertex closest to the associated joint.

5.2. Binding Weights Transfer

Rely on the correspondence mentioned in Section 4.4, vertex *p*'s binding weight can also be retrieved by calculating the combination of binding weights of q_1, q_2, and q_3.

6. Result

As shown in Figure 1, the source animatable model is a dog with a skeleton and motion data, and the target static model is a cat that only has mesh information, where the two models contain, respectively, 8,136 and 5,400 triangles. To clone 20 key-frame poses and skeleton data from the dog model to the cat takes about 2 minutes on a desktop PC with an Intel Pentium 4 3GHz CPU, and to specify the correspondence of 40 features between the two models costs 10-15 minutes through our user interface. This is much faster than traditional methods to clone animation, and is more perceptual for people who are new to creating animation.

Utilizing our system, an animation sequence can be reapplied to different models which only have geometry information. To perform the surface parameterization including relaxation and alignment for the dog model and the cat model takes 1.546 and 1.548 seconds, respectively. Creating the correspondence for all of the vertices needs 11.328 seconds and reconstructing skeleton for the target model costs 0.141 seconds. Figures 7 and 8 show other results created by our method.

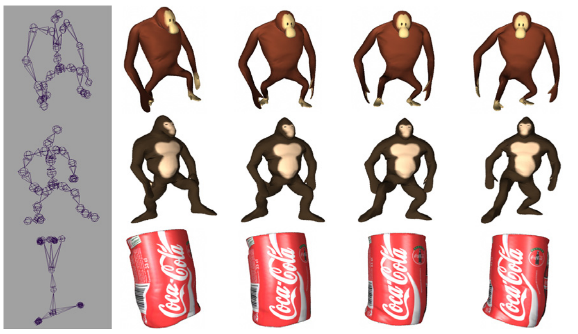

Figure 7. The monkey is the source model, and the gorilla and the cola can are the target models. The monkey and the gorilla have 1,884 and 5,454 triangles, respectively, and have 53 common specified features. The cola can has been specified 20 common features with the source model.

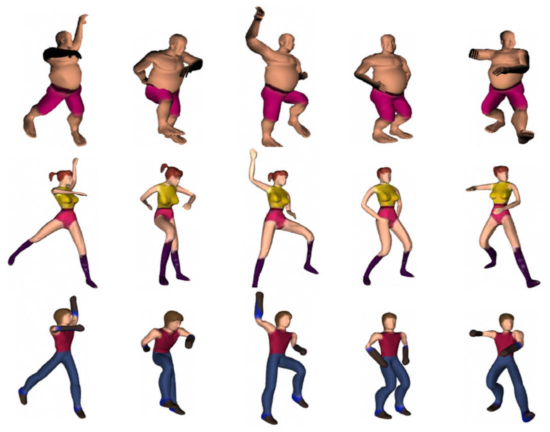

Figure 8. The dancing motion sequence of a fat man model is cloned to the girl and the boy models, respectively. They have 6,848, 928, and 4,356 triangles and 41 common features are specified among them.

7. Conclusions and Future Work

An efficient method for cloning animation from an animatable model to a static one is presented in this paper. The vertex-wise correspondence between the two models is derived from their planar surface parameterizations with feature alignment. Hence, a model's motion, color, texture, skeleton, and binding weights can be transferred to other ones, and an animation sequence can be reused to different models, even the target models only have geometry information. Therefore, through our method, the time-costly routines that produce the skeleton, binding weights, and the same animation sequence for target models can be reduced. Moreover, the target models with the transferred animation data can be imported into "Maya" for later refinement if necessary. In our experiments, the result generated by our system without any refinement is still adoptable, especially is useful in video games, background crowds, and animations that do not require very high-quality deformations.

Currently, the user must choose a ”proper” pose of the source model from an animation sequence before performing our algorithm, where the ”proper” means the pose of the source model is similar to that of the target static model. If the source and target models are in different poses, for example, the source model is standing but the target one is sitting, the cloning results may be a little strange. One of our future works is to adjust the initial pose automatically to let the source and target models be the same pose before cloning the animation sequence.

To transfer the joints of the source model, we just recorded a joint by only two vertices. Although the result shows this simple method works well in almost all cases, using more vertices to record a joint can also be considered to enhance the precision.

We can discovery that when the source and target models have different proportion of limbs and in some animation sequences a part of the target model will intersect itself. This problem also occurs in motion capturing. Gleicher [10] presented a space-time constraint method for 3D models. This method focused on adapting the motion of one articulated figure to another with identical structure but different segment lengths. They assume that the configuration of an articulated figure is specified by a hierarchical joint tree. Besides, when the two models' shapes differ very much, self-intersection also will occur. We should pay more efforts to solve the mesh intersection problem in the future.

7. Acknowledgements

The work was supported in part by the National Science Council of Taiwan under Contracts NSC93-2213-E-002-084-.

References

[1] M. Alexa. Merging polyhedral shapes with scattered features. *The Visual Computer*, 16(1):26–37, 2000.

[2] M. Alexa. Recent advances in mesh morphing. *Computer Graphics Forum*, 21(2):173–196, 2002.

[3] B. Allen, B. Curless, and Z. Popović. The space of human body shapes: Reconstruction and parameterization from range scans. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003)*, 22(3):587–594, 2003.

[4] M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum (Proceedings of Eurographics 2002)*, 21(3):209–218, 2002.

[5] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proceedings of ACM SIGGRAPH 1995*, pages 173– 182, 1995.

[6] M. Floater. Parameterization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997.

[7] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In *Proceedings of Multiresolution in Geometric Modelling 2003*, 2003.

[8] K. Fujimura and M. Makarov. Foldover-free image warping. *Graphical Models and Image Processing*, 60(2):100– 111, 1998.

[9] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of ACM SIGGRAPH 1997*, pages 209–216, 1997.

[10] M. Gleicher. Retargeting motion to new characters. In *Proceedings of ACM SIGGRAPH 1998*, pages 33– 42, 1998.

[11] C. Gotsman, X. Gu, and A. Sheffer. Fundamentals of spherical parameterization for 3d meshes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003)*, 22(3):358– 363, 2003.

[12] I. Guskov, K. Vidimce, W. Sweldens, and P. Schr¨oder. Normal meshes. In *Proceedings of ACM SIGGRAPH 2000*, pages 95– 102, 2000.

[13] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003)*, 22(3):954–961, 2003.

[14] A. Lee, D. Dobkin, W. Sweldens, and P. Schröder. Multiresolution mesh morphing. In *Proceedings of ACM SIGGRAPH 1999*, pages 343–350, 1999.

[15] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. Maps: Multiresolution adaptive parameterization of surfaces. In *Proceedings of ACM SIGGRAPH 1998*, pages 95–104, 1998.

[16] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002)*, 21(3):362–371, 2002.

[17] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformations: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of ACM SIGGRAPH 2000*, pages 165–172, 2000.

[18] A. Mohr and M. Gleicher. Building efficient, accurate character skins from examples. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003)*, 22(3):562–568, 2003.

[19] E. Praun and H. Hoppe. Spherical parameterization and remeshing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003)*, 22(3):340–349, 2003.

[20] E. Praun, E., W. Sweldens and P. Schröder, Consistent Mesh Parameterizations. In *Proceedings of ACM SIGGRAPH 2001*, 179-184, 2001

[21] S. Shlafman, A. Tal, and S. Katz. Metamorphosis of polyhedral surfaces using decomposition. *Computer Graphics Forum (Proceedings of Eurographics 2002)*, 21(3):219–228, 2002.

[22] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2004)*, 23(3):399–405, 2004.

[23] W. T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 13(3):743–768, 1963