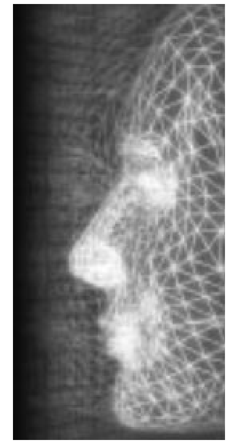
COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2007; **18**: 235–245
Published online 29 June 2007 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/cav.178

Mesh pose-editing using examples



By Tong-Yee Lee, Chao-Hung Lin, Hung-Kuo Chu, Yu-Shuen Wang, Shao-Wei Yen and Chang-Rung Tsai*

An easy-to-use mesh pose-editing system is presented. We take advantage of both skeleton-based and example-based approaches in order to provide an intuitive way for artists to edit mesh poses. Our system automatically extracts the skeletons of the remaining example models once the skeleton of a reference mesh is constructed. In our editing system the desired skeleton can be easily and naturally posed using an inverse kinematics (IK) algorithm incorporated with searching the optimal weights in the defined skeleton space of examples meshes. Eventually, the desired shape with detailed deformation can be constructed by blending the example meshes. Experimental results show that the proposed system provides an easy and intuitive control on mesh pose-editing. Copyright © 2007 John Wiley & Sons, Ltd.

Received: 14 May 2007; Accepted: 14 May 2007

KEY WORDS: pose-editing; example mesh; animation; deformation; skeleton

Introduction

Editing fascinating 3D models quickly and conveniently has been a very important research topic in computer animation. Editing a model might be very tedious work. In the worst-case scenario, artists must be forced to edit a model vertex by vertex, if a natural configuration and detailed deformation are required. Inverse kinematics (IK)[1–4] is the most common approach to pose an articulated object by determining joint configurations. However, to accurately edit a natural pose an IK system must work with some kinematic constraints. It is also not easy to set correct constraints for expressing a desired figure. An efficient solution to realistic skeleton poses creation is presented in References.[5,6] Their generated poses are derived from existing natural poses. However, if the detailed and lifelike skin of a model is demanded, an additional skinning approach[7,8] must be applied here to produce a natural skin.

Another solution is to directly pose a mesh (i.e., skeleton-free editing) by a few selected anchors.[9–13] The user moves these anchors through an intuitive interface and then computes new positions for the remaining non-anchor vertices based on their geometric features. In addition, some researches have been concentrated on preserving the original geometric characteristics such as area[14] and object volume.[15,16] Later, example-based mesh editing approaches are presented.[17,18] From input examples, they present a novel mesh-based IK approach to find meaningful mesh deformations that meet the specified vertex constraints. Each example mesh is represented as a feature vector formed by deformation gradients of faces.

In this paper, we develop a new and intuitive way for editing a mesh based on the existing example models. In the proposed approach, the skeletons of example meshes are automatically computed. First, based on a cyclic-coordinate descent (CCD) algorithm,[4] the users can intuitively specify their desired shape (skeleton) by simply controlling the end-effectors. Then, we take the edited/desired skeleton as a constraint to find the optimal weights in the example skeleton space. Finally, the existing example meshes are blended with the obtained weights to synthesize a mesh with a meaningful skin deformation inferred from example meshes. The remaining of this paper is organized as follows. Next section overviews our editing system. The detailed methods in this system are introduced in Section

*Correspondence to: T.-Y. Lee, Computer Graphics Group, Visual System Lab, Department of Computer Science and Information Engineering, National Cheng-Kung University, No.1 Ta-Hsueh Road, Tainan 701, Taiwan.
E-mail: tonylee@mail.ncku.edu.tw

Methodology. Section *Results and Discussions* shows the results created by our system with discussions. Finally, conclusions are described in Section *Conclusion*.

System Overview

We schematically illustrate the proposed mesh pose-editing system in Figure 1. This system consists of two main procedures: initialization and editing. We briefly describe these two procedures below:

- Initialization: First, one of the example models is selected as the reference model. This reference model is decomposed into several connected near-rigid components (as shown in Figure 1(b)). Second, the skeleton of a reference model is constructed according to the partitioned result. Finally, those skeletons of the remaining examples are automatically computed (as shown in Figure 1(c)).
- Editing and Animation: To provide an easy and intuitive control over pose editing for artists, the CCD algorithm is applied. The artists interactively select an active kinematic chain of joints and then simply specify the position of an end-effector for posing their desired skeleton. Our algorithm then takes this edited/desired skeleton as a constraint to search the optimal weights in the skeleton space spanned by example skeletons. The obtained weights are next applied to blend the example meshes to generate a desired mesh. The artists can continue editing the mesh to create another key-pose. Animation can be achieved by linearly interpolating these weights between two key-poses.

Methodology

Skeleton Construction of Example Meshes

We need to build a skeleton for each example mesh. First, given the input examples, we use[19,20] to partition the reference model into several near-rigid components according to the deformation gradients[21] computed from these input models. A skeleton can be built by a collection of these near-rigid components connected by joints. In this simple manner, the connected skeleton of a reference model is constructed according to partitioned results. To automatically obtain a desired and good skeleton is not an easy task. Our system also allows users to manually edit the connected skeleton if necessary.

Second, once the skeleton of a reference model is generated, we automatically compute the skeletons of the remaining examples as follows. We adopt the concept of the mean value coordinate (MVC)[22] to find

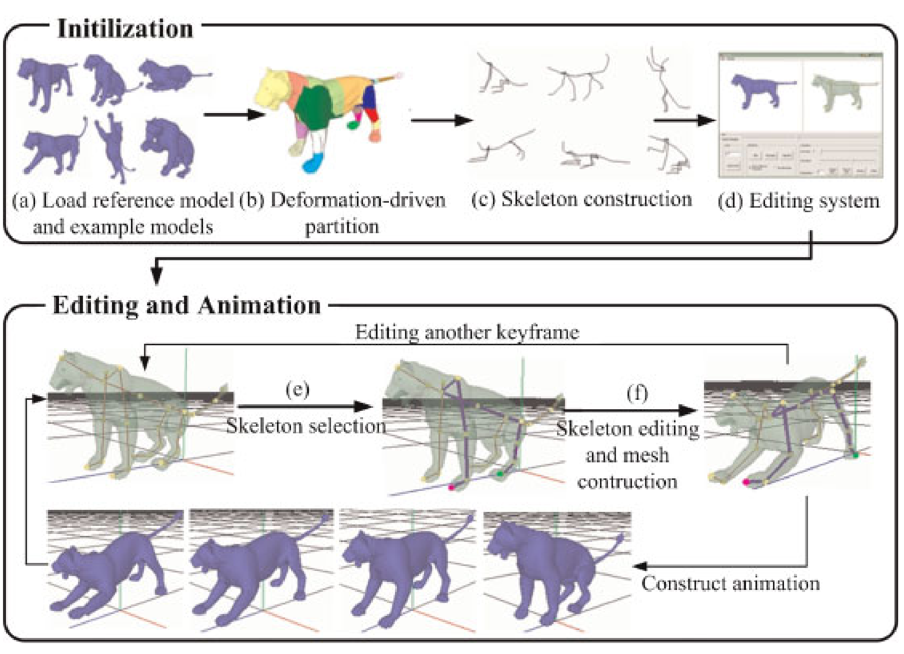


Figure 1. System overview.

DOI: 10.1002/cav

the positions of corresponding joints in the example models. The basic idea of MVC is to present the coordinate of an interior in a closed mesh as a weighted linear combination of the mesh vertices. We compute MVCs of all joints in the reference skeleton, that is, the weights of vertices, using equation (1). Then, using equation (1) again, the position p_{joint} of each joint in an example skeleton is calculated from the obtained weights from the reference skeleton joints and vertex coordinates from this example model.

$$p_{\text{joint}} = \frac{\sum_i \sum_{k=1}^{3} w_{f_i}(k) v_{f_i}(k)}{\sum_i \sum_{k=1}^{3} w_{f_i}(k)} \tag{1}$$

In the above equation, $v_{f_i}(k)$ is the vertex v_k on face f_i, and $w_{f_i}(k)$ is its corresponding weight for linear interpolation. Figure 2 shows an example of the skeleton construction for example meshes.

Example-based Inverse Kinematics Using Skeletons

The proposed system provides artists an intuitive way to edit the skeleton for their desired shapes. Our algorithm searches the optimal skeleton configuration (i.e., a set of weights) among the skeleton space spanned by the skeletons of example models. In this way, we can ensure that the generated skeletons are natural looking/meaningful due to reproduction/blending from the example skeletons, while meeting the artists' constraints. Finally, the obtained weights are applied to blend the example meshes for the desired shape.

Example Skeleton Space

Given a bone b_s in the reference skeleton and its corresponding bone b_d in the other example skeleton (i.e., with different posture), we would like to compute the *bone deformation gradient*, defined by the Jacobian matrix of affine transformation, from the local frame, defined by a reference bone b_s, to the local frame, defined by an example bone b_d. A bone local frame can be easily defined by bone itself and two additional basic vectors $(\overline{b_s} \times \overline{b_d})$ and $(\overline{b_s} \times \overline{b_d}) \times \overline{b_s}$ (or$(\overline{b_s} \times \overline{b_d}) \times \overline{b_d}$). Next, normalize them to become three orthonormal axes. Then, a pure rotation between bones b_s and b_d can be easily computed. Let $B_{i,j}$ be the bone deformation gradient of a bone j between a reference skeleton S and an example skeleton S_i. A new deformation gradient $B_{\text{target},j}$ for a target/edited skeleton S_{target} in the example skeleton space can be defined as a weighted linear combination of all bone deformation gradients $B_{i,j}$ from the n_{model} input examples (as shown in Figure 3). That is:

$$\sum_{i=1}^{n_{\text{model}}} w_i B_{i,j} = B_{\text{target},j}\ , \text{for } j = 1 \ldots n_{\text{bone}} \tag{2}$$

where w_i is the weight of B_i for representing the new deformation gradient B_{target}, and n_{bone} represents the number of bones. In this equation, we represent a target skeleton pose by a linear combination of example skeletons.

It is well known that the linear blending of rotation matrices will result in unnatural effects. We adopt exponential mapping[18,23,24] to solve this problem. First, we map the rotation matrix from SO(3) to the Lie algebra so(3) using the logarithm function. Next, we linearly interpolate the rotation matrix in so(3) and then map back to SO(3) using the exponential function. Therefore, equation (2) is rewritten as:

$$\exp\left(\sum_{i=1}^{n_{\text{model}}} w_i \log B_{i,j}\right) = B_{\text{target},j}\ , \text{for } j = 1 \ldots n_{\text{bone}} \tag{3}$$

To obtain the blending weights w_i while meeting the artists' specified constraints, that is, the new edited

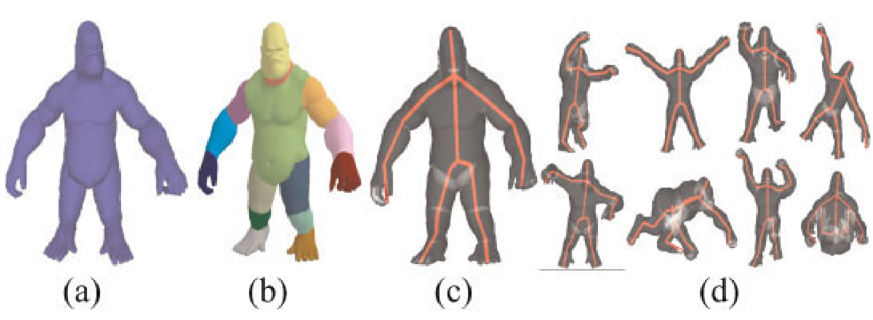


Figure 2. Skeleton construction. (a) A selected reference model, (b) The partitioned result of (a), (c) The constructed reference skeleton, (d) The corresponding skeletons for the remaining examples.

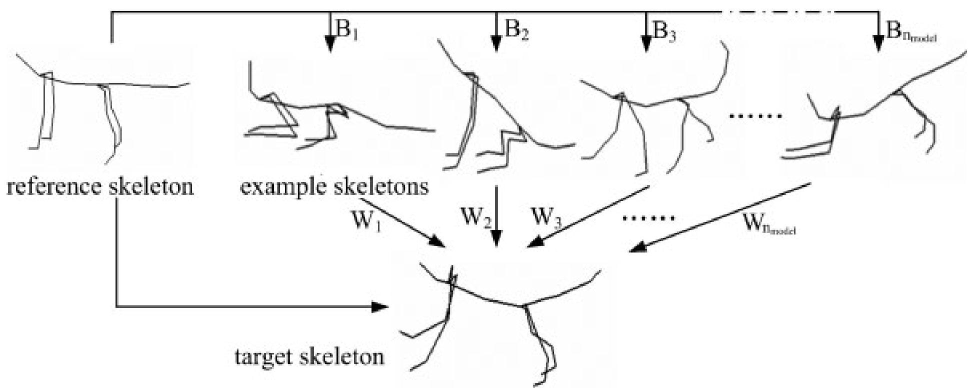


Figure 3. An illustration of the skeleton space spanned by the example skeletons.

skeleton, we minimize the following equation:

$$\arg\min_{w_i}\left\|\exp\left(\sum_{i=1}^{n_{\text{model}}} w_i \log B_{i,j}\right) - B_{\text{target},j}\right\|, \text{for} \quad j = 1 \ldots n_{\text{bone}} \tag{4}$$

Equation (4) can be treated as a linear system form Ax = b:

$$[\log B_{i,j}][w_i] = [\log B_{\text{target},j}] \tag{5}$$

We solve this linear system in the least square sense using the general approach $x = (A^tA)^{-1}A^tb$. Since this matrix size is not very large, it can be solved with very low cost. Therefore, our algorithm can compute the desired skeleton/weights very fast.

Mesh Reconstruction Using Deformation Gradients

Once the weights are obtained in Section *Example Skeleton Space*, the example meshes are directly blended with these weights to generate the desired mesh. First, we calculate the triangle deformation gradient $T_{i,j}$ between the triangle j in the example mesh i and its corresponding triangle in the reference mesh. To avoid creating unnatural effects, polar decomposition[25] is adopted to decompose the deformation gradient $T_{i,j}$ into rotation and scale/shear components:

$$T_{i,j} = R_{i,j}S_{i,j} \tag{6}$$

Again, we use exponential mapping to individually blend the rotation components in so(3) with the obtained weights from equation (5):

$$T_j(w) = \exp\left(\sum_{i=1}^{n_{\text{model}}} w_i \log R_{i,j}\right)\sum_{i=1}^{n_{\text{model}}} w_i S_{i,j}, \text{for} \quad j = 1 \ldots n_{\text{bone}} \tag{7}$$

Second, the feature vector f_{target} of the desired mesh can be formed by collecting the blended deformation gradients $\{T_j(w)\}_{j=1}^{m}$ calculated using equation (7),[18] where m represents the number of triangles on a mesh model. The feature vector can be rewritten as the following equation:

$$f_{\text{target}} = Gx \tag{8}$$

where vector $x = (x_1, \cdots, x_n, y_1, \cdots, y_n, z_1, \cdots, z_n) \in \Re^{3n}$ is the collection of desired mesh vertex coordinates and n represents the number of vertices. G is a sparse matrix in which the coefficients are only related to the reference mesh vertices. We can recover the Cartesian coordinates of the desired mesh vertices x by solving the equation:

$$x = \arg\min_x \|Gx - (f_{\text{target}} + c)\| \tag{9}$$

In the above equation, c is a constant vector created from vertex constraints. Because the feature vector is invariant to mesh translation, we need to specify the Cartesian coordinate of one vertex to resolve the translational degree of freedom. Then, this linear system becomes full-ranked and thus has a unique solution in the least-squares sense. The analytical least-square solution is expressed in the form: $x = (G^T G)^{-1} G f_{\text{target}}$. Eventually, the desired shape with detailed deformation is obtained.

Mesh Construction With Multi-Weights

In the previous subsections, we blend the example meshes with a set of weights to generate the required

DOI: 10.1002/cav

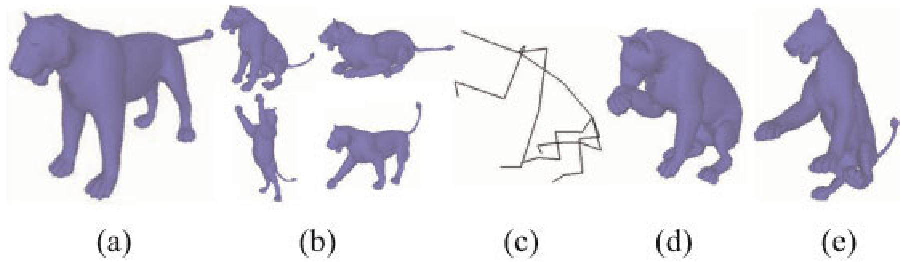


Figure 4. (a) A reference model, (b) Example models, (c) A required skeleton, (d) The required shape corresponding to (c), (e) The result generated by the proposed approach.

shape. Given a target skeleton, our method determines an optimal weight for each example skeleton. Similarly, in[18] they compute an optimal weight for each example mesh based on deformation gradients. For both methods, the generated shape might potentially not meet the artist's constraints well when the example shapes or skeletons and target shape or skeleton are very distinct. Figure 4, for example, shows a target skeleton (see (c)), that is, the skeleton constraints specified by artists. However, the resulted shape (see (e)) by the proposed method, as shown in Figure 4(e), is different from the user's desired shape (see (d)).

A straightforward solution is to increase the number of example meshes. However, it requires extra cost or effort to create more examples. To solve this problem, like,[18] more constraints are needed to fix some local regions and then perform local mesh editing. However, this approach may require many manual efforts, that is, manually select the fixed regions, to accurately set several constraints. In this paper, we extend the basic idea in Section *Example Skeleton Space* and redefine the example skeleton space as a set of subspaces. Each subspace is spanned by an individual bone in the example models. It means that each bone will generate a set of weights for its target bone. In this manner the skeleton space is extended and more meaningful shapes are available for editing. The linear system for calculating weights for a bone k is expressed as below:

$$\begin{bmatrix} \log T_{1,k} & \log T_{2,k} & \dots & \log T_{n_{\text{model}},k} \\ \log T_{1,1} & \log T_{2,1} & \dots & \log T_{n_{\text{model}},1} \\ \log T_{1,2} & \log T_{2,2} & \dots & \log T_{n_{\text{model}},2} \\ \dots & \dots & \dots & \dots \\ \log T_{1,P} & \log T_{2,P} & \dots & \log T_{n_{\text{model}},P} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \dots \\ w_{n_{\text{model}}} \end{bmatrix} = \begin{bmatrix} \log T_{\text{target},k} \\ \log T_{\text{target},1} \\ \log T_{\text{target},2} \\ \dots \\ \log T_{\text{target},P} \end{bmatrix} \tag{10}$$

where w_i represents the weight of a corresponding bone k from each example, P represents the number of immediately adjacent bones to this bone K. $T_{i,j}, 1 \le i \le n_{\text{model}}, 1 \le j \le P$ represents an affine transformation from its jth neighboring bone between the example model i and a reference model. In this equation, the position of bone is related to its neighboring bones. Generally, the number P of immediately adjacent bones is usually 2 or 3. Therefore, it is quite fast to solve equation (10) for each bone k. However, it might generate an unnatural shape since the example meshes might be blended with an extra large weight as well as it might generate a shape with inaccurate size because the sum of weights is not equal to 1. Therefore, we further add two additional terms to penalize possible solutions that are far from the example skeletons. The first term is the weight summation. It implies that a solution close to the mean of example models is favored to select. A large weight will be penalized. The second term favors that the sum of weights is equal to 1. It means that the scale of solution must be equal to example models. Adding these two terms, the equation is reformulated as below:

$$\arg\min_{w_i} \left(\left\| \sum_{i=1}^{n_{\text{model}}} \log(w_i T_{i,j}) - \log(T_{\text{target},j}) \right\| + K_1 \sum_{i=1}^{n_{\text{model}}} w_i + K_2 \left[\left(\sum_{i=1}^{n_{\text{model}}} w_i \right) - 1 \right] \right), \text{for } j = k, 1 \dots P \tag{11}$$

where the parameters K_1 and K_2 are the coefficients of the penalty terms. The parameter K_1 determines the degree of extrapolation. In Figure 5, for example, the smaller value of K_1 represents a larger degree of extrapolation. The proper values of parameters K_1 and K_2 are needed for creating a natural shape. In our experiments, the parameter K_1 is set to 1.2 and parameter K_2 is set to 2.8.

Comp. Anim. Virtual Worlds 2007; **18**: 235–245
DOI: 10.1002/cav

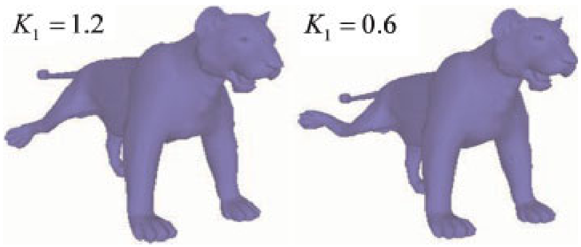


Figure 5. An effect of setting the parameter K_1.

As a result, the linear system in equation (10) is reformulated as:

$$\begin{bmatrix} \log T_{1,k} & \log T_{2,k} & \dots & \log T_{n_{\text{model}},k} \\ \log T_{1,1} & \log T_{2,1} & \dots & \log T_{n_{\text{model}},1} \\ \log T_{1,2} & \log T_{2,2} & \dots & \log T_{n_{\text{model}},2} \\ \dots & \dots & \dots & \dots \\ \log T_{1,P} & \log T_{2,P} & \dots & \log T_{n_{\text{model}},P} \\ K_1 & 0 & \dots & 0 \\ 0 & K_1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & K_1 \\ K_2 & K_2 & \dots & K_2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \dots \\ w_{n_{\text{model}}} \end{bmatrix} = \begin{bmatrix} \log T_{\text{target},k} \\ \log T_{\text{target},1} \\ \log T_{\text{target},2} \\ \dots \\ \log T_{\text{target},P} \\ 0 \\ 0 \\ \dots \\ 0 \\ K_2 \end{bmatrix} \tag{12}$$

Editing System

An editing system based on the CCD algorithm[4] is presented. The CCD algorithm is an iterative heuristic search technique that attempts to minimize the position and orientation errors by modifying one joint variable at each step. In our system, the CCD algorithm updates an active kinematic chain of joints rather than all of the skeleton joints. This active chain is selected by users (the purple line segments shown in Figure 6(b)) before the CCD algorithm is executed. Each joint angle is modified in order to minimize the distance between the desired position and the end-effector (the end joint on the selected chain is illustrated as the red point in Figure 6(b) and (c)). The user edits bones by simply manipulating the end-effector.

In the proposed system, artists do not need to edit all bones. The unedited bones can be automatically inferred from the edited bones. The system will gather information from example skeletons to modify the unedited bones with regard to the most possible configuration in the skeleton space. In this manner, the artists can edit skeletons more conveniently to obtain a natural looking/meaningful target skeleton. The calculation of all bone weights is computed in two steps. First, we calculate the weights of the bones that are on the selected active chain using equation (12). Second, the weights of bones on the selected chain are considered as constraints for finding weights of the remaining bones using equation (12) again. At this time, some weights are known from the first step. In other words, the positions and weights of the unedited bones are inferred from the bones on an active chain (Figure 7).

For this editing system, to pose a mesh, artists simply select a chain and then move an end-effector of this chain to its new position. The results can be obtained in real

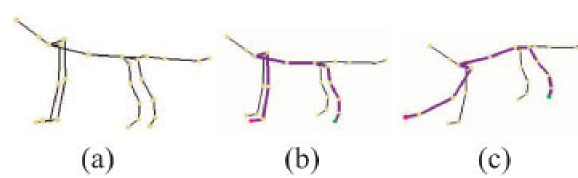


Figure 6. An illustration of posing a skeleton using the CCD algorithm. (a) The original skeleton, (b) A selected active chain (the purple line segments), (c) Specify some fixed points (e.g., the green point) and freely move the end-effector (the red point) to edit the skeleton.

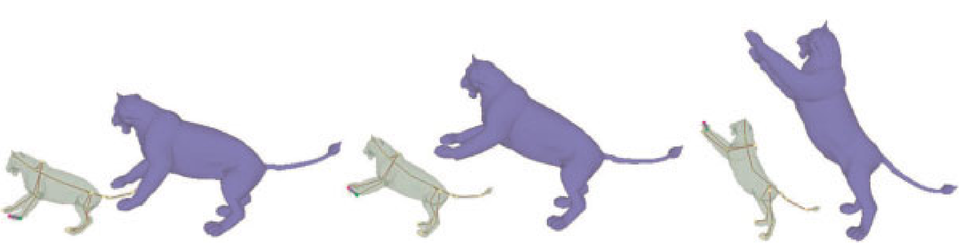


Figure 7. Edited results using the proposed system.

DOI: 10.1002/cav

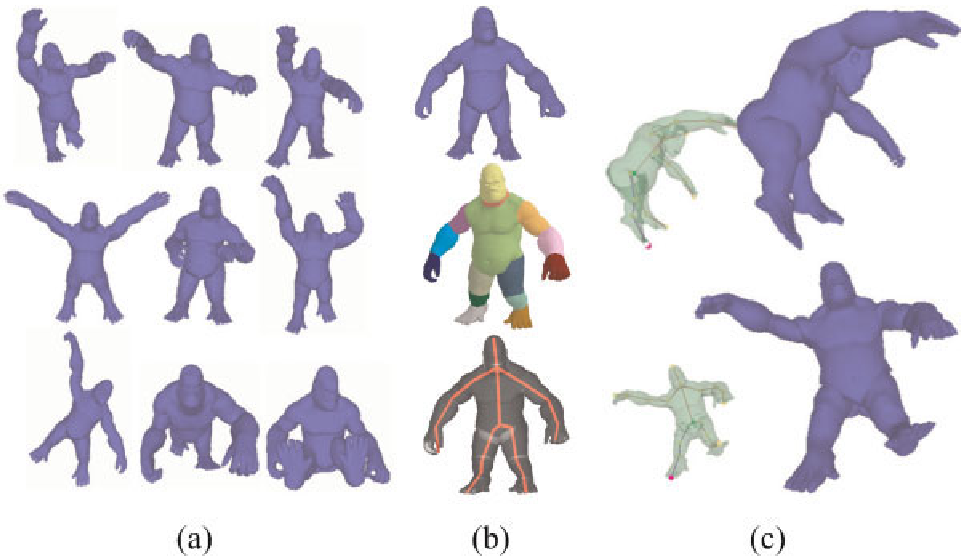


Figure 8. A mesh editing example. (a) Input example models, (b) The partitioned result of a reference model and its corresponding skeleton, (c) The edited results.

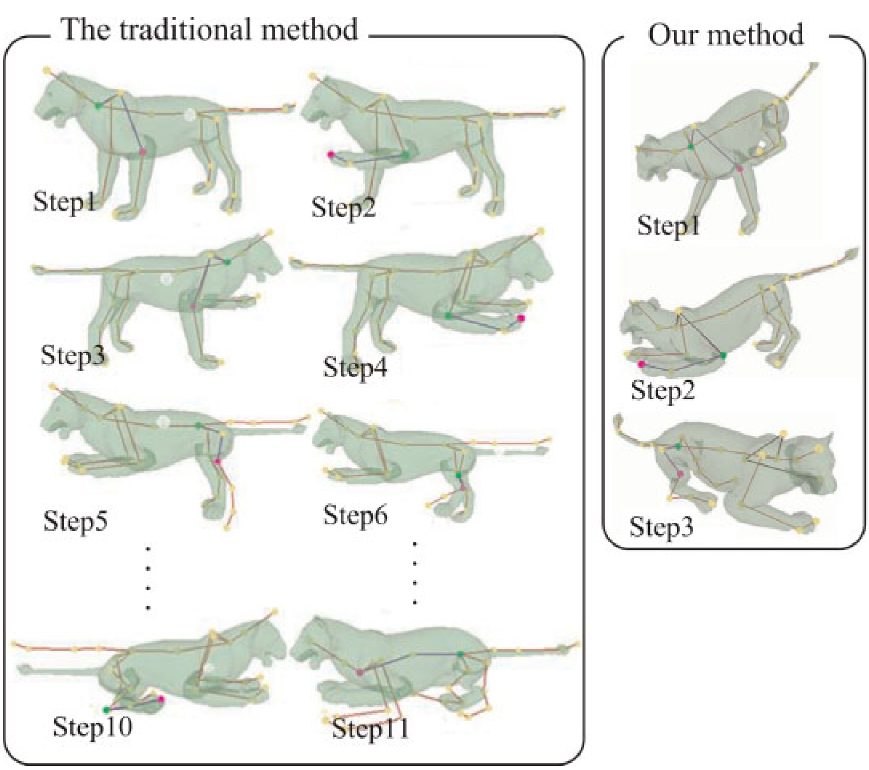


Figure 9. A comparison between the IK approach using a CCD algorithm and our editing system (i.e., CCD plus examples).

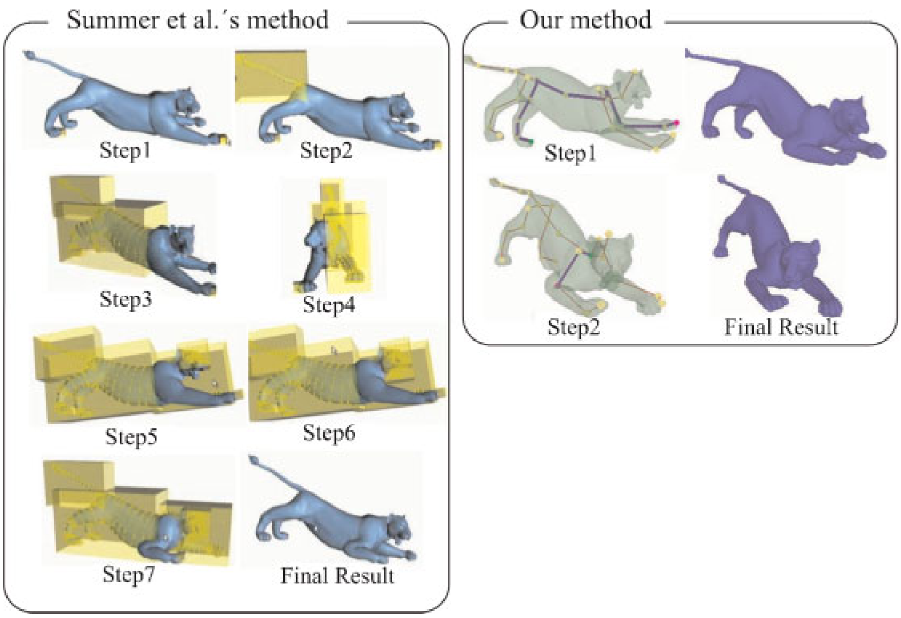


Figure 10. A comparison between[18] and our approach. The left figure is extracted from Reference.[18]

time. An active chain can include many bones (which means a nearly-global editing) or only a few bones (which means a local editing). Therefore, our system is very flexible to edit poses freely for different purposes.

Results and Discussions

We use the proposed pose-editing system to quickly create several interesting poses and animations. This system is run on a 3GHz Pentium 4 PC with 1G Ram and NVIDIA GeForce 6600 display card. First, we demonstrate an example of pose-editing in Figure 8. There are nine example models (see (a)) used in this example. The partitioned result of a reference model and its corresponding skeleton are shown in (b). The partitioned components are visualized in different colors. With our system, the artist can interactively edit a character pose by simply and intuitively moving the selected end-effector (the red point in (c)) in real-time. The supplementary video demonstrates an easy manipulation of our pose-editing system. In addition, we can also generate some interesting animation sequence as shown in Figures 11 and 12. After editing several key poses an interesting motion animation, such as walking, slipping, and jumping, can be created by linearly interpolating these bone weights and then reconstructed in-between mesh.

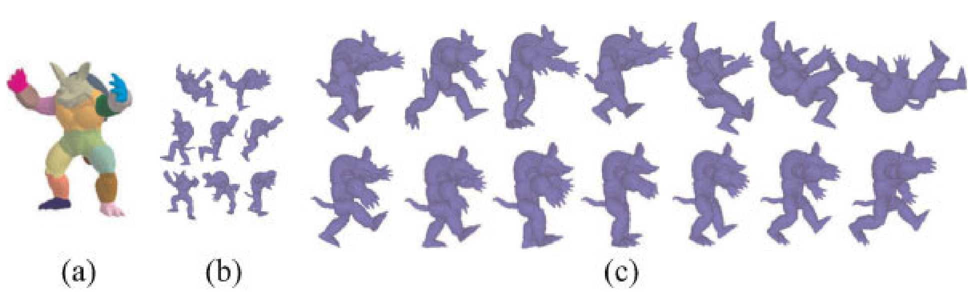


Figure 11. Monster slipping and walking animation. (a) The partitioned result of a reference model, (b) Input example models, (c) The key-frames of slipping (top row) and walking (bottom row) animation.

Comp. Anim. Virtual Worlds 2007; **18**: 235–245
DOI: 10.1002/cav

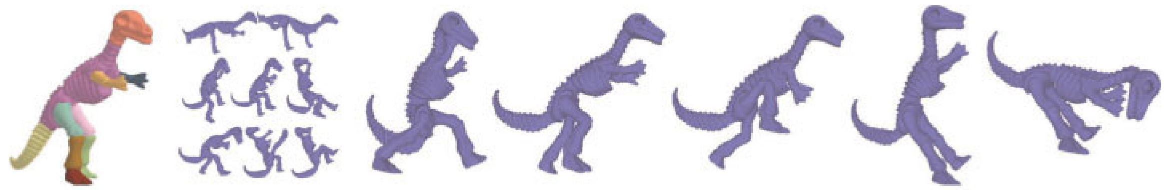


Figure 12. Dino jumping animation. (a) The partitioned result of a reference model, (b) Input example models, (c) The key-frames of jumping animation.

Compared to the traditional skeleton-based editing approaches, that is, forward kinematics and IK, our editing system only needs about 1∼5 steps in our test examples to specify the positions of end-effectors to meet artists' constraints. It also always generates natural looking results inferred from examples. However, the forward kinematics approach requires the artists to accurately set the rotation at each selected joint, and the quality of the results totally depends on artists' skill, effort spent, and time. A traditional IK provides more high-level control than forward kinematics. However, it also needs many steps to pose a desired shape. For example, Figure 9 shows a comparison between the proposed system and IK using a CCD algorithm. In addition, both techniques need to carefully skin the mesh for obtaining the desired deformation. In contrast, our approach avoids this tedious skin task by blending examples.

In contrast to other example-based editing approaches, such as,[17,18] the proposed system generally has similar ability in terms of editing control, such as editing the example shown in Figure 9. In terms of computing efficiency, the proposed system is faster than[18] and is comparable to.[17] In Reference,[18] the approach computes blending weights by taking deformation gradients of all mesh triangles into account. Our approach just needs to consider bone configurations. The number of bones is much fewer than the number of triangles.

However, considering some editing requirements such as the local editing, more constraints are required for[17,18] to edit new poses due to the insufficient number of examples. For example, in Figure 10 local editing is required in the front right leg of the Lion model; by References [17,18] several regions are selected as fixed regions to ensure that these regions are constrained during local editing. Therefore, both[17,18] need more steps to carefully select these regions. Using the proposed system, we only need two steps to achieve similar results. At each step we only need to select an active chain for manipulating the end-effector as shown in Figure 10. In addition, most animators are used to skeleton-based editing. In this sense, our system seems more intuitive to them than References.[17,18] When the animators edit a chain of bones, they can expect their editing results. Using References,[17,18] although they are both very good methods and allow the animators to freely pose models by handles or constraint vertices, from our experience the results are sometimes beyond the imagination of the animators if they do not know what example models are used.

Conclusion

A novel mesh pose-editing approach is presented. In our editing system, artists simply specify the new position of an end-effector to fast pose their desired skeletons. Then, we search the optimal solution in the defined skeleton space to compute blending weights. The detailed deformation of the desired shape can be computed by blending the examples meshes. We do not need to tediously edit skin weights in contrast to traditional skeleton-based editing systems. Experimental results and supplementary video clips demonstrate that the proposed system is an easy-to-use and intuitive pose-editing system. In near future, we like to explore possible application of our techniques to morphing problem.[26,27]

ACKNOWLEDGEMENTS

This paper is supported by the National Science Council, Taiwan, Republic of China, under contract No. NSC-95-2221-

E-006-029, NSC-95-2221-E-006-368, and NSC-95-2221-E-006-193-MY2.

References

1. Boulic N, Thalmann M, Thalmann D. A global human walking model with realtime kinematic personification. *The Visual computer* 1990; **6**: 344–358.
2. Chadwick J, Haumann D, Parent R. Layered construction for deformable animated characters. *Computer Graphics* 1989; **23**(3): 243–252.
3. Koga Y, Kondo K, Kuffner J, Latombe JC. Planning motions with intentions. *Computer Graphics* 1994; **28**(4): 395–408.
4. Wang LCT, Chen CC. A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *IEEE Transactions on Robotics and Automation* 1991; **7**(4): 489–499.
5. Grochow K, Martin SL, Hertzmann A, Popovic Z. Style-based inverse kinematics. *ACM Transactions on Graphics* 2004; **23**(3): 522–531.
6. Rose CF III, Sloan P-PJ, Cohen MF. Artist-directed inverse-kinematics using radial basis function interpolation. *Computer Graphics Forum* 2001; **20**: 239–250.
7. Guo Z, Wong K-G. Skinning with deformable chunks. *Computer Graphics Forum* 2005; **24**(3): 373–381.
8. Mohr A, Gleicher M. Building efficient, accurate character skins from examples. *ACM trans. Graph* 2003; **22**(3): 562–568.
9. Huang J, Shi X, Zhou K, *et al*. Subspace gradient domain mesh deformation. *ACM Transactions on Graphics* **2006**: **25**(3): 1126–1134.
10. Lipman Y, Sorkine O, Cohen-or D, Levin D, Rossl C, Seidel H-P. *Differential coordinates for interactive mesh editing*. In *SMI 2004: Proceedings of the international conference on Shape Modeling and Applications*, 2004; pp. 181–190.
11. Sorkine O, Lipman Y, Cohen-or D, Alexa M, Rossl C, Sheidel HP. *Laplacian surface editing*. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 2004; pp. 179–188.
12. Sorkine O, Cohen-Or D. *Least-squares meshes*. In *Proceedings of Shape Modeling International*, IEEE Computer Society Press, 2004; pp. 191–199.
13. Yu Y, Zhou K, Xu D, *et al*. Mesh editing with Poisson-based gradient field manipulation. *ACM Transactions on Graphics* 2004; **23**(3): 644–651.
14. Igrashi T, Moscovich T, Hughes J. *Spatial keyframing for performance-driven animation*. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2005; pp. 107–116.
15. Von Funck W, Theisel H, Seidel H-P. *Vector field based shape deformations*. In *SIGGRAPH'06*, 2006; pp. 1118–1125.
16. Zhou K, Huang J, Snyder J, *et al*. *Large mesh deformation using the volumetric graph laplacian*. In *SIGGRAPH'05*, 2005; pp. 496–503.
17. Der KG, Sumner RW, Popovic J. Inverse kinematics for reduced deformable models. *ACM Transactions on Graphic* 2006; **25**(3).
18. Sumner RW, Zwicker M, Gotsman C, Popovic J. Mesh-based inverse kinematics. *ACM Transactions on Graphics* 2005; **24**(3): 488–495.
19. Lee TY, Wang YS, Chen TG. Segmenting a deforming mesh into near-rigid components. In *Pacific Graphics 2006*, 2006; **22(**9): 729–739.
20. Tong-Yee Lee, Ping-Hsien Lin, Shaur-Uei Yan, Chun-Hao Lin. *Mesh decomposition using motion information from animation sequence. Proceedings of International Conference on Computer Animation and Social Agents* (CASA). 2005; Hong-Kong.
21. Sumner RW, Popovic J. Deformation transfer for triangle meshes. *ACM Transactions on Graphics* 2004; **23**(3): 399–405.
22. Ju T, Schaefer S, Warren J. *Mean value coordinates for closed triangular meshes*. *ACM SIGGRAPH 2005*, 2005; pp. 561–566.
23. Alexa M. Linear combination of transformations. *ACM Transactions on Graphics* 2002; **21**(3): 380–387.
24. Murry RM, Li Z, Sastry SS. *A Mathematical Introduction to Robotic Manipulation*. CRC Press: Broken Sound Parkway, NW, Boca Raton, FL, USA, 1994.
25. Shoemake K, Duff T. *Matrix animation and polar decomposition*. In *Proceedings of Graphics Interface 92*, 1992; pp. 259–264.
26. Tong-Yee Lee, Huang PH. Fast and institutive polyhedra morphing using smcc mesh merging scheme. *IEEE Transactions on Visualization and Computer Graphics* 2003; **9**(1): 85–98.
27. Chao-Hung Lin, Tong-Yee Lee. Metamorphosis of 3D Polyhedral Models Using Progressive Connectivity Transformations. *IEEE Transactions on Visualization and Computer Graphics* 2005; **11**(1): 2–12.

Authors' biographies:

Tong-Yee Lee received the PhD degree in computer engineering from Washington State University, Pullman, in May 1995. Now, he is a Professor in the Department of Computer Science and Information Engineering at National Cheng-Kung University in Tainan, Taiwan, Republic of China. He has served as an Associate Editor for IEEE Transactions on Information Technology in Biomedicine from 2000 to 2007. His current research interests include computer graphics, non-photorealistic rendering, image-based rendering, visualization, virtual reality, surgical simulation, and distributed and collaborative virtual environment. He leads the Computer

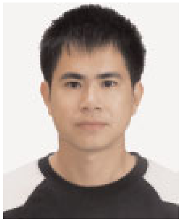
DOI: 10.1002/cav

Graphics Group/Visual System Lab at National Cheng-Kung University (http://graphics.csie.ncku.edu.tw/). He is a member of the IEEE.

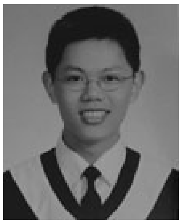
Chao-Hung Lin was born in Koushung, Taiwan, Republic of China, in 1973. He received his BS in computer science/engineering from Fu-Jen University, MS and PhD in computer engineering from National Cheng-Kung University, Taiwan in 1997, 1998 and 2004, respectively. Now, he is Assistant Professor in the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

Hung-Kuo Chu received the BS degree in computer science/engineering from National Cheng-Kung University, Tainan, Taiwan, in 2003. Now, he is pursuing his PhD degree at Department of Computer Science and Information Engineering, National Cheng-Kung University. His research interests include computer graphics.

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



Shao-Wei Yen received the BS degree from Department of Civil Engineering, National Taiwan University, Taipei, Taiwan, in 2000. Now, he is pursuing his PhD degree at Department of Computer Science and Information Engineering, National Cheng-Kung University. His research interests include computer graphics.



Chang-Rung Tsai received the MS degree in computer science/engineering from National Cheng-Kung University, Tainan, Taiwan, in 2006. His research interests include computer graphics.

DOI: 10.1002/cav