Skeleton Constrained Dual-Resolution Modeling for Sketch-Based Deformation

Bing-Yu Chen* Fu-Che Wu† Tsung-Yi Lin† Meng-Chang Su†
National Taiwan University
* E-mail: robin@ntu.edu.tw
† E-mail: {joyce,wildsun,kyatapi}@cmlab.csie.ntu.edu.tw

***Abstract*—In this paper, a dual-resolution modeling framework for a sketch-based animation editing system is presented. For the easy-to-use purpose, a sketch-based user interface is often used for editing a skeleton-based animation model. However, due to the loss of relationship information about the details of the model, a shape deformed from the skeleton in some cases will produce unexpected artifacts. To reduce these artifacts, in our system, we first generate a coarse mesh as an abstract layer from the input model based on quadratic error metrics while considering the skeleton constraints. Then, the original model is projected to the abstract layer to construct a displacement map to preserve the difference between the original model and the simplified one. Hence, when we apply a Laplacian-based smoothing operator to reduce the deforming artifacts, the smoothing operator can only be applied to the coarse mesh and provide the corresponding changes to the original model to obtain a smooth but feature preserved editing result.**

I. Introduction

To produce a 3D character animation, it is needed to provide the key-poses of a 3D character model, and the animation can then be generated by interpolating the key-poses. However, to edit the 3D character model is not very easy, since there may be many degrees of freedom (DOF) that need to be controlled. Although traditional or commercial model editing systems can help talented animators to provide high quality 3D character animation, these systems are usually hard to use. Even for the talented animators, they still need to read a lot of documents and manuals before using the systems and always need to spend a lot of time for editing. Therefore, we propose an easy-to-use animation editing framework with a sketch-based user interface in this paper. An easy-to-use framework usually implies low quality editing result. To prevent this disadvantage, we also provide a surface smoothing operator with a dual-resolution meshes editing structure in this framework. Hence, even for a novel user, he or she can also easily generate a character animation in a short time with acceptable quality.

Our system basically consists of two modules. One is an automatic skeleton extracting module which is used to automatically construct the skeleton and skin binding of a 3D character model for further editing. The other is a sketch-based key-frame editing module which is used to edit the key-poses of the character model with the extracted skeleton and skin binding. This editing module contains a smoothing operator and a dual-resolution meshes structure which uses the original mesh and its simplified version for editing. The smoothing operator is only applied to the simplified mesh and provides the corresponding changes to the original one to obtain a smooth but feature preserved editing result.

The key-pose editing in this framework is based on the skeleton-driven deformation. First, an automatic skeleton extracting module (Sec. III-A) is introduced to automatically construct the skeleton and skin binding of a 3D character model for further editing. Then, a sketch-based key-frame editing module (Sec. III-B) is provided to let a user adjust the auto-generated skeleton to create the key-poses of the character model. Since a skeleton-based deformation system may produce notorious artifacts, a Laplacian-based smoothing operator is introduced in the editing module to smooth these undesired artifacts (Sec. IV-B). Based on the smoothed shape, we recalculate the skin binding in the animation to produce a better result (Sec. IV-C). To avoid the details of the editing model to be over-smoothed caused by the smoothing operator and to increase the interactivity of the editing, the editing model is simplified first to generate a coarse mesh as an abstract layer. Then, the original model is projected to the abstract layer to construct a displacement map to preserve the difference between the original model and the simplified one. Hence, the smoothing operator can only be applied to the coarse mesh and provide the corresponding changes to the original one to obtain a smooth but feature preserved editing result. Since we are using two kinds of meshes for editing a 3D model, this structure is so-called dual-resolution meshes structure (Sec. IV-D) in this paper.

II. Related work

To edit the key-poses or key-shapes of a 3D model, some deformation methods are proposed. Free-form deformation (FFD) introduced by Sederberg and Parry [1] is one of the model editing methods. Since the user needs to handle many control points, the editing process is not easy to handle well. Some other methods deform an object by its skeleton intuitively because the skeleton describes the pose of a shape well. For example, Magnenat-Thalmann *et al.* used such technique to provide a linear blend skinning method [2], [3]. Lazarus *et al.* used an axis instead of using a lattice to try to provide an efficient and intuitive deformation method called axial deformations (AxDf) [4]. Chang and Rockwood used a Bézier curve to define the desirable skeleton of the deformed object [5], and Singh and Fiume used wires for deformation [6].

To provide a much simple and intuitive model editing environment, some sketch-based editing systems are proposed recently. Teddy [7] is a sketch-based shape modeling tool. Using Teddy, the user can draw some 2D free-form strokes interactively on screen and the Teddy system then can automatically construct the plausible 3D polygonal surface. Chaudhuri *et al.* provided a sketch-based 2D to 3D interface [8]. Using this interface, a base mesh of the 3D character model can be modified to closely match to an input sketch with minimal user interaction. Motion Doodles [9] is a novel system for sketching the motion of a character. Using this system, a user can only draw a continuous sequence of lines, arcs, and loops, and an animated motion can then be created for the character. Hua and Qin developed a scalar-field based FFD (SFFD) [10]. With the SFFD method, users can directly sketch a scalar field of an implicit function to control the deformation of any embedded model.

Yoshizawa *et al.* developed a new scheme for free-form skeleton-driven global mesh deformations [11]. In their method, a Voronoi-based skeletal mesh is first extracted from a given model, and the skeletal mesh can then be modified by FFD. Duncan and Swain introduced a new tool that allows animators to change the position of some sets of controls with mouse strokes [12]. Their system works well with pen-based input devices and provides an interface for posing complex skeleton intuitively. Kho and Garland developed a sketch-based mesh deformation system [13]. In their system, a user can just use a 2D sketch to easily deform a mesh in 3D space. In their system, they also provide many operators such as twisting, indirect control, smoothing, and adaptive refinement.

Capell *et al.* [14] proposed a framework for the skeleton-driven animation of elastically deformable characters. A character is embedded in a coarse volumetric control lattice, which provides the structure needed to be deformed by the finite element method. Nealen *et al.* [15] presented a method based on this idea for the intuitive editing of surface meshes by means of view-dependent sketching. This system lets the user easily determine the handle, either by silhouette selection and cropping, or by sketching directly onto the surface. Zhou *et al.* [16] presented a novel technique for large deformations on 3D meshes using the volumetric graph Laplacian. They first construct a graph to represent the volume inside the input mesh. This graph's Laplacian encodes the volumetric details as the difference between each point in the graph and the average of its neighbors. Preserving these volumetric details during deformation imposes a volumetric constraint that prevents unnatural changes in volume. Huang *et al.* proposed a subspace technique that builds a coarse control mesh around the original mesh and projects the deformation energy and constraints onto vertices of the control mesh by using the mean value interpolation [17].

For easy implementation and suitable for an arbitrary mesh, a bind weight approach is adopted. However, there are artifacts when deforming the shape of a model. To reduce these artifacts, a dual-resolution approach is often used. Zorin *et al.* [18] proposed a multi-resolution representation based subdivision for interactive editing. To avoid the requirement of the subdivision connectivity, Kobbelt *et al.* [19] used a discrete fairing technique to define the decomposition and reconstruction operations by separating the high-frequency details from the low-frequency shape and eventually recombine the two information to recover the original mesh.

III. Motion Creation

In our system, the motion of a 3D character model is created based on the skeleton of it. A skeleton tree plays a major role in the definition of motion. It is also useful to apply a motion data for editing by sketches. To achieve this purpose, an automatic skeleton extracting process is proposed. Following, each skeleton can be assigned a motion data or edited by stoke curves drown by sketches.

A. Skeleton Extraction

Our goal for extracting the skeleton from a 3D model is to find a concise skeleton that is suitable for animation control and editing. To avoid the problem that a generated skeleton is too complex to control, our skeleton is represented as the composition of a set of some important points with linkages. There are three main stages for generating such a control skeleton.

- **Feature detection:** This initial procedure detects the features of a 3D character model to find some suitable positions to represent skeletal points inside the model.
- **Connection construction:** To preserve the topology relation of the model, the neighborhood relationship of each skeletal point is inherited from the edge connectivity of the surface mesh.
- **Refinement for shape variation:** Two skeletal points connected by a straight line cannot suitably describe the variation of the shape. Hence, we introduce a force field to force the skeleton path to follow the shape's varied form.

To select some important feature points as the essential skeletal points, we develop a procedure to evaluate which parts are suitable as the end points, connection points, and joint points, where are three kinds of essential points of our skeletal model. Fig. 1 shows different stages in the algorithm. The input model is shown in Fig. 1 (a). Initially, a Voronoi diagram is constructed to locate the initialized candidates (Fig. 1 (b)), and some essential domain balls are extracted (Fig. 1 (c)). An end point is defined as locally far from the main part: a geodesic distance is measured on the surface, and then a watershed algorithm is applied to detect where an end point is (Fig. 1 (d)). Then, a visibility repulsive force field is introduced to adjust all other skeletal points except the end points to the local minima and form the connection points (Fig. 1 (e)). Following this, a neighborhood relationship from the mesh connectivity is determined (Fig. 1 (f)), and then the point that has more than two connection linkages is assigned as a joint point. Finally, a snake algorithm based on the repulsive force field is applied, and all linkages are adjusted to fit the shape variation (Fig. 1 (g)). For more details, please refer to [20].

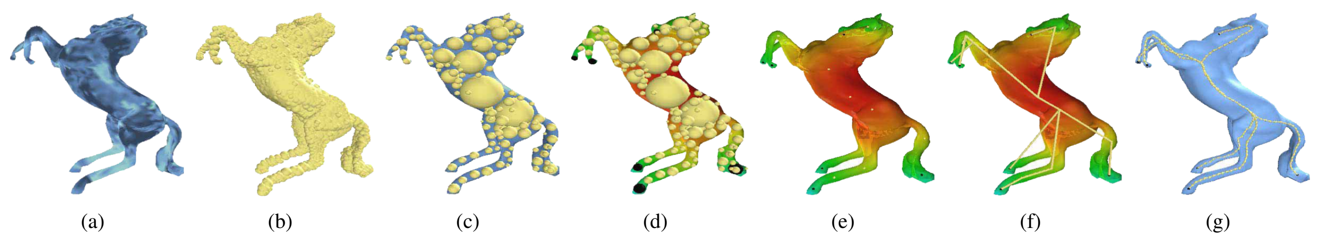

Fig. 1. Different stages in skeleton extracting algorithm.

B. Key-pose Editing

To edit the key-poses of the 3D model for each key-frame is to determine the variation of the local frame among the selected skeleton. Based on the above results, all of the skeletons consist of many axis line segments, which are fit by NURBS curves and defined as:

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i},$$

where P_i is the control point, w_i is the weight, $N_{i,p}(u)$ is the i-th B-spline basis function of degree p and is defined recursively as following:

$$N_{i,p}(u) = \frac{u-u_i}{u_{i+p}-u_i} N_{i,p-1}(u) + \frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+p}} N_{i+1,p-1}(u),$$

$$N_{i,0}(u) = \begin{cases} 1 & \text{if} u \in [u_i, u_{i+1}) \\ 0 & \text{otherwise} \end{cases}.$$

To fit a NURBS curve into these sample points, the number of control points is determined by the skeleton length. A knot vector is calculated by the cord length and a least square approximation is adopted.

To deform a skeleton for the key-poses, each skeletal point is embedded in a local frame. There are six DOF that need to be controlled including the position and the rotation angle. The position can be determined by drawing a stroke on the screen. To edit a 3D curve by a sketch, we provide two projection planes. One is the view plane, and the other is the ground plane as shown in Fig. 2.

The user can rotate the model to the best view point to edit the shape. After a skeleton curve is selected as a target, the mouse curser is snapped on this curve to select a (connection) point to begin the deformation. After the left button is pressed, the user can draw a new skeleton curve. All of the changes are constrained on the planes that are parallel to the projection plane. Since there are two continuous curves, at each (connection) point we can thus determine its direction of the tangent line. Therefore, the rotation angle is determined by the change of direction on their tangent lines. When we draw a new curve, the system predicts the final curve to let the user be able to understand the final result and the new position of the end point. After the left button is released, the cursor will move to the shadow curve of the target skeleton to edit in the ground plane. Fig. 2 shows the editing sequence of the sketch-based editing module.

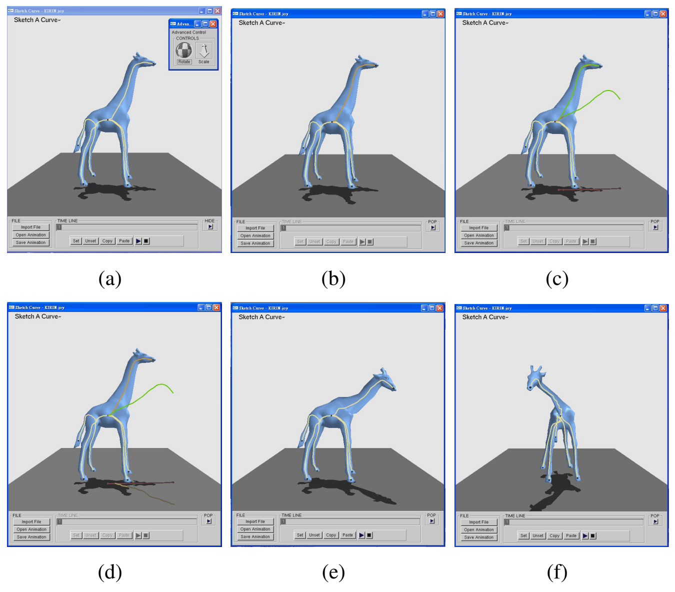

Fig. 2. (a) The user can easily rotate or scale the 3D model by a pop-up window. (b) Initially, a target skeleton is selected. (c) The user draws a new skeleton curve on the view plane. (d) The user can draw another curve for z-axis through its projected skeleton in shadow on the ground plane. (e) and (f) Two different views of the editing result.

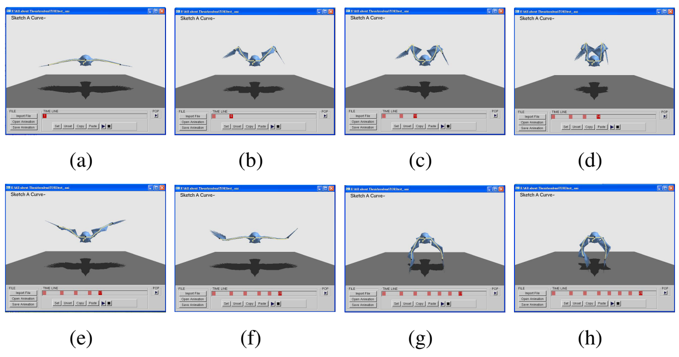

Fig. 3. The sequence of edited key-poses.

C. Motion Interpolation

If the key-poses have been edited as Fig. 3, the animation may be discontinuous between the key-frames. In the previous methods, the space-time constraints problem has been discussed widely [21], [22], [23]. We do not focus on this problem too much and adapt an interpolation approach based on interpolating the skeleton curve to generate the in-between poses for smoothing the motion. We assume that the begin and the end of the key-frames are represented as the static poses. Thus, the motion consists of acceleration and deceleration processes. If the configuration of the begin pose is θ_b and that of the end pose is θ_e, the in-between poses are defined as $\theta(t) = w(t)\theta_b + (1 - w(t))\theta_e$, where $w(t)$ is a normalized slow-in/slow-out weighting function. If the user has specified more than one key-poses, an interpolation sequence is generated only based on these key-poses. The final motion is the interpolated motion sequence composed of the original key-poses.

IV. Shape Deformation

Since to create the motion of a 3D character model by editing its skeleton may produce notorious artifacts, in this section, the shape deformation due to the motion of the skeleton is described. First, the initial character pose is referred to the "dress pose". Each vertex on the surface of the model is assigned a set of influences. Each influence has different skin bindings which define the binding relationship between its skin (surface of the model) and skeleton. After the skeleton has deformed, the new position of each vertex is calculated by the following equation [24]:

$$v'_d = \sum\nolimits_{i=1}^{n} w_i M_i D_i^{-1} v_d,$$

where v'_d is the new position, v_d is the original position at the dress pose, w_i are the influence weighting, M_i is the transform matrix, and D_i^{-1} is the inverse of the dress pose matrix associated with the i-th influence.

To deform the shape according to the deformed skeleton. First, we determine the skin binding relationship between the shape and the skeleton. Then, a smoothing operator is introduced to solve the artifacts on the shape due to the skeleton deformation. To avoid the details of the editing model to be over-smoothed caused by the smoothing operator, a dual-resolution mashes structure is also proposed.

A. Initial Skin Binding

In general, the skin binding relationship is defined as the mapping among the nearest features between the shape and the skeleton. However, sometimes it is hard to find a proper mapping. For example, around the joint of the skeleton, it may produce wrong skin binding since the geometric distance is not so reliable for finding the skin binding. To avoid this case, we need to preserve the correct topological relationship.

First, we build many domain balls on the skeleton as the representable feature points. The surfaces of the domain balls constitute a control surface. Then, we define a geometry cost function based on the distance between the control surface and the mesh surface. For the two surfaces Ω and Ψ, we calculate the normalized geodesic distance. Let $\{p_1, p_2, \cdots, p_n\}$ and $\{q_1, q_2, \cdots, q_n\}$ be the feature sequence in Ω and Ψ, respectively. Then, $0 \leq d(p_1) \leq d(p_2) \leq \cdots \leq d(p_n) \leq 1$, where $d(p_i)$ is the normalized geodesic distance. If $p \in \Omega$ and $q \in \Psi$, a cost function for mapping them is defined as:

$$E(p,q) = C_t(p,q) + \lambda C_g(p,q),$$

where $C_t(p,q)$ and $C_g(p,q)$ are the topology and geometry costs, respectively, and λ is the weighting. The topology cost is defined as the difference of the covering range between the two binding nodes, which is represented as $\max(d(p_i)) - \min(d(p_j))$, $\forall p_i, p_j \in M(q)$, where $M(q)$ represents the feature set that is matched to q. The covering range in mesh space is defined as the neighborhood area at the binding position.

To reduce the computational cost, if a face is very close to a control surface, we can bind them first. Then, we recover the rest parts until all of the faces have defined their binding relationship based on the cost function. Each vertex on the mesh needs a definition corresponding to the skeleton. Thus, after one patch on the mesh surface is bound to the skeleton, we can determine its skin binding on each vertex based on its neighboring faces. Fig. 4 shows the result.

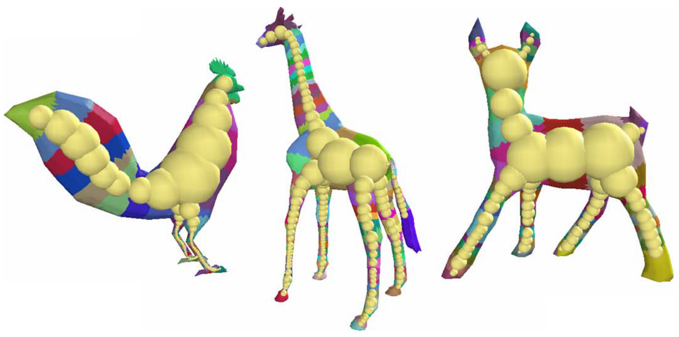
Fig. 4. The faces on the mesh are bound to the domain balls on the skeleton.

B. Smoothing Operation

To deform a model by editing its skeleton is simple, but there may exist many notorious artifacts, such as the following examples:

- In the joint part of two different skeletons (linkages), it is hard to deform it well.
- Large deformation at elbow-like position causes artifacts.
- Twist operation makes the volume decreased.

To avoid these artifacts on the surface, we provide a smoothing operator to solve this problem. Because the Gaussian filter is shrinkage, the Fourier descriptor, $x' = f(k)x$, which can be applied N times iteratively, i.e., $x^N = f(k)^N x$, can be used as a low-pass filter. However, the exact projection of the subspace of low frequency is just not longer feasible. Taubin [25] provided a transfer function to approximate a Fourier descriptor as:

$$f(k) = (1 - \lambda k)(1 - \mu k),$$

where μ and λ determine the pass-band frequency and $k_{pb} = \frac{1}{\mu} + \frac{1}{\lambda} > 0$. In our implementation, we set $\lambda = -0.5$ and $\mu = 0.526$, respectively, and $k = \Delta x$ is the discrete Laplacian. The discrete Laplacian of a discrete surface can be defined by the following weighted average over the neighborhoods:

$$L(x_i) = \sum\nolimits_{j \in i^*} w_{ij}(x_j - x_i),$$

where w_{ij} is defined as:

$$\frac{\|v_j - v_i\|^{-1}}{\sum_{k \in i^*} \|v_k - v_i\|^{-1}}.$$

In general, the result of the Taubin smoothing operation is good enough. However, it sometimes produces over-smoothing of the surface. To solve this problem, we provide a constraint-based smoothing operator. First, we need to draw a reference curve. Based on the reference curve, a region of faces is selected as a smoothing target. Then, a domain surface that is based on this reference curve is constructed as a constraint.

To construct this domain surface, the distance between the reference curve and the skeleton curve is determined by the nearest distance. The distance is represented as the radius when a domain surface is constructed on the skeleton curve. Let $p(v_i)$ be a projection position on the skeleton curve of vertex v_i, $d(v_i, p(v_i))$ be the distance between the vertex v_i and its projection $p(v_i)$, and $r_{p(v_i)}$ be the radius of the projection point, then a smoothing operator can be defined by minimizing the following energy function:

$$E = \sum_{v_i \in R} (L(v_i) + w_c \left\| d(v_i, p(v_i)) - r_{p(v_i)} \right\|),$$

where R is the selected region, $L(v_i)$ is the Laplacian, and w_c is a weight of the constraint. This idea is similar to Kho and Garland's approach [13]. In contrast, their approach prefers to preserve its original position, but we prefer to preserve the reference curve. Fig. 5 shows the smoothing result.

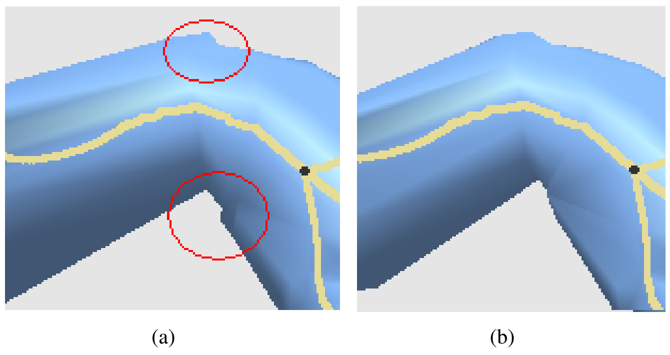

Fig. 5. After the smoothing operator, the artifacts in (a) are removed as (b).

C. Skin Binding Adjustment

As described in the previous section, each smoothed pose is represented as the adjusted pose. To perform a better animation, we need to recalculate their skin binding to fit the adjusted pose. The same idea has been presented in PSD (pose space deformations) [26], SBE (shape by example) [27], MWE (multi-weight enveloping) [28], EigenSkin [29], and Morhr's work [24].

As described before, the new position of each vertex of the deformed model is defined as:

$$v'_d = \sum_{i=1}^{n} w_i M_i D_i^{-1} v_d.$$

However, if we use the original skin binding, it can not produce the adjusted pose. Thus, we need to recalculate the skin binding to fit the pose. Let v_s be an adjusted pose, we adjust the skin binding to make v'_d fit into v_s. If the interpolation result is not good enough, we can apply one more smoothing operator on the selected key-frame. Assuming that we have already adjusted m key-poses, then the skin binding can be adjusted as minimizing the following equation:

$$\sum_{i=1}^{m} \left\| {v'}_d^i - v_s^i \right\|^2,$$

where i is a adjusted key-pose. After the skin binding is recalculated, the animation sequence is also smoothed.

D. Dual-Resolution Meshes Structure

To avoid over-smoothing the detail features on the surface and to speed up the interaction, we do not apply the smoothing operator on the original model directly but a simplified one instead as Fig. 6. Initially, we simplify the original model into a coarse one by using the quadratic error metric (QEM) approach [30]. It is an fast and intuitive method to simplify a 3D model, but it only takes the shape of the model into consideration. DeCoro and Rusinkiewicz [31] proposed a view- and pose-independent method for the automatic simplification of skeletally articulated meshes. Based on their idea, we also add the skeleton constraints to the QEM.

To take the skeleton into consideration, the new quadratic error matric Q'_i is modified as:

$$Q'_i = w_1 * Q_i + w_2 * S_i + w_3 * E_i,$$

where Q_i is the original QEM cost of v_i, S_i is the cost computed by the distance from v_i to the skeleton, E_i is the cost computed by the distance from v_i to the end of the skeleton, and w_1, w_2, w_3 are the weighting of these three functions.

Then, we determine the relationship between the coarse model and the original one as a displacement map for the coarse model. If a vertex of the original model can be projected onto a face of the coarse mesh, it will bind to this face. Otherwise, it will bind to the closest vertex. Thus, we need to find a hyperplane on this vertex with its neighbor. A moving-least-squares (MLS) model on the shape is adopted [32]. This approach estimates a projection hyperplane:

$$H_r = \{x | n \cdot x - D = 0, x \in R^3\}, \|n\| = 1,$$

where $\cdot$ is the standard inner product, n is the normal vector, and D is the distance to the plane, so that the following quantity is minimized:

$$\sum_{i \in I} (n \cdot r_i - D)^2 \theta(\|r_i - r\|),$$

where r is the nearest point from the center of a cell, r_i are some neighbor points near r, I is the set of the index i, and θ is a normal distribution function. Since the weighting function $\theta(\|r_i - r\|)$ decreases as the distance $\|r_i - r\|$ increases, the resulting hyperplane H_r approximates a tangent hyperplane to the shape near the point r.

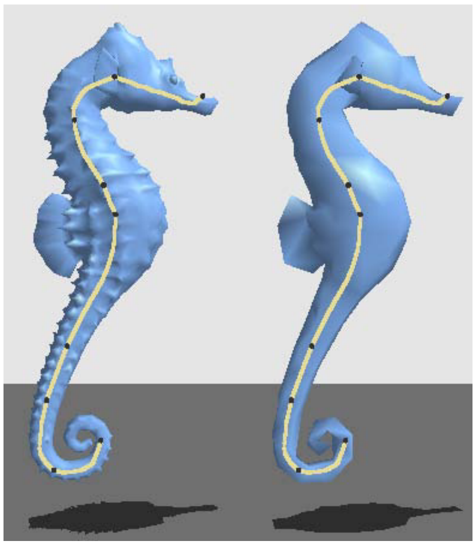

Fig. 6. The dual-resolution meshes of a seahorse model.

Thus, each vertex on the original model will be projected onto the closest face or the hyperplane of the coarse model. The resulting form is $v_f = H_i(u, v) + d \cdot n$, where v_f is the position of the vertex on the original model, $H_i(u, v)$ is a projection position on the hyperplane i, d is the distance, and n is the normal vector.

V. Results

The major goal of this paper is trying to make the key-pose editing process easier and suitable for a novel user. To evaluate our system, we invited a professional animator and an eight-years-old child (Fig. 8) to use our system. The animator has many years of experience in animation production, but the child only has a little experience on computer operating.

Fig. 7 is generated by the animator in a few minutes. After a little training time, he was already familiar with this system. In his experience, the system is easy to learn and use. Although our system has some limitations, such as the pose can be specified only in one projection plane, the testing animator said this system is good enough for a prototype editing to present the idea. Fig. 9 is generated by the child also in a few minutes.

VI. Conclusions and Future Work

Editing an animation sequence is not an easy work; we hence provide a friendly user interface to let this work become intuitive and fun in this paper. In our system, a user could select a character to generate its control skeleton automatically. For key-pose editing, each skeleton segment can be adjusted by drawing a new curve. The major drawback of our system is the limitation of motion deformation along a projected plane. To avoid the artifacts due to the sketch-based editing system, a dual-resolution meshes structure is also provided in this paper. Therefore, a smoothing operator can be applied only to the deformed coarse mesh to obtain a smoothed editing result. Then, the corresponding changes can be provided to the original model to achieve a smooth but feature preserved final result.

Fig. 8. A child is playing our system.

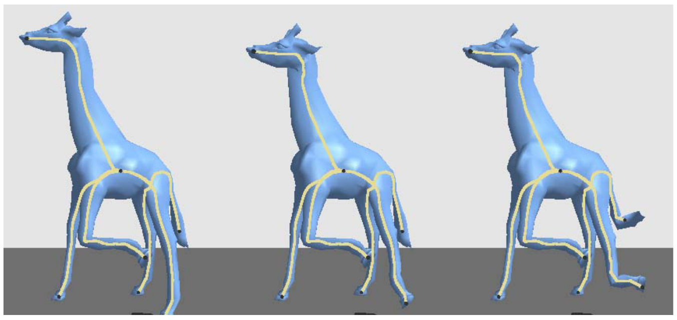

Fig. 9. An animation sequence generated by an eight-years-old child, a novice.

Furthermore, there are some work that could be improved in the future, such as the trajectory editing, inverse kinematics based on physical constraints and more motion templates. Besides using the mouse to draw a new sketch curve, we can also use a background image as a guide. Based on the shape in the image, we can construct a skeleton of the shape. Thus, by specifying the mapping relationship between the 3D skeleton and the 2D skeleton, an animation sequence can be driven from a video.

Acknowledgment

This work was partially supported by the National Science Council of Taiwan under NSC97-2221-E-002-224, and also by the Excellent Research Projects of the National Taiwan University under NTU97R0062-04.

References

[1] T. W. Sederberg and S. R. Parry, “Free-form deformation of solid geometric models,” *ACM Computer Graphics (SIGGRAPH 1986 Conference Proceedings)*, vol. 20, no. 4, pp. 151–160, 1986.

[2] N. Magnenat-Thalmann, R. Laperrire, and D. Thalman, “Jointdependent local deformations for hand animation and object grasping,” in *Graphics Interface 1998 Conference Proceedings*, pp. 26–33, 1988.

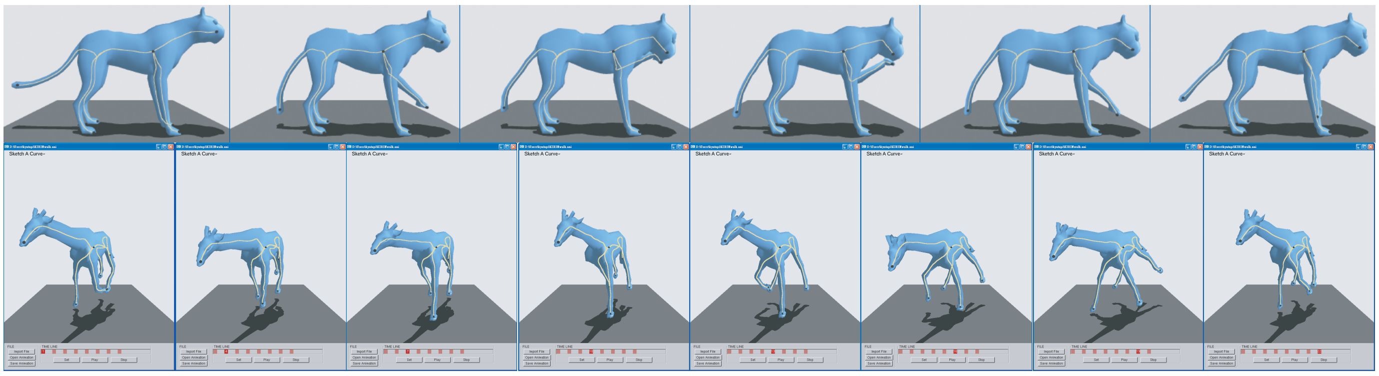
Fig. 7. Two animation sequences generated by an expert.

[3] N. Magnenat-Thalmann and D. Thalmann, “Human body deformations using joint-dependent local operators and finite element theory,” in *Making Them Move: Mechanics, Control, and Animation of Articulated Figures*, pp. 243–262, Morgan Kaufmann, 2006.

[4] F. Lazarus, S. Coquillart, and P. Jancène, “Axial deformations: an intuitive deformation technique,” *Computer-Aided Design*, vol. 26, no. 8, pp. 607–613, 1994.

[5] Y.-K. Chang and A. P. Rockwood, “A generalized de casteljau approach to 3d free-form deformation,” in *ACM SIGGRAPH 1994 Conference Proceedings*, pp. 257–260, 1994.

[6] K. Singh and E. L. Fiume, “Wires: a geometric deformation technique,” in *ACM SIGGRAPH 1998 Conference Proceedings*, pp. 405–414, 1998.

[7] T. Igarashi, S. Matsuoka, and H. Tanaka, “Teddy: a sketching interface for 3d freeform design,” in *ACM SIGGRAPH 1999 Conference Proceedings*, pp. 409–416, 1999.

[8] P. Chaudhuri, P. Kalra., and S. Banerjee, “A system for view-dependent animation,” *Computer Graphics Forum (Eurographics 2004 Conference Proceedings)*, vol. 23, no. 3, pp. 411–420, 2004.

[9] M. Thorne, D. Burke, and M. van de Panne, “Motion doodles: an interface for sketching character motion,” *ACM Transactions on Graphics (SIGGRAPH 2004 Conference Proceedings)*, vol. 23, no. 3, pp. 424–431, 2004.

[10] J. Hua and H. Qin, “Free-form deformations via sketching and manipulating scalar fields,” in *Proceedings of the 2003 ACM Symposium on Solid Modeling and Applications*, pp. 328–333, 2003.

[11] S. Yoshizawa, A. G. Belyaev, and H.-P. Seidel, “Free-form skeleton-driven mesh deformations,” in *Proceedings of the 2003 ACM Symposium on Solid Modeling and Applications*, pp. 247–253, 2003.

[12] B. Duncan and M. Swain, “Sketchpose: Artist-friendly posing tool,” in *ACM SIGGRAPH 2004 Conference Abstracts and Applications*, p. 7, 2004.

[13] Y. Kho and M. Garland, “Sketching mesh deformations,” in *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, pp. 147–154, 2005.

[14] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popović, “Interactive skeleton-driven dynamic deformations,” in *ACM SIGGRAPH 2002 Conference Proceedings*, pp. 586–593, 2002.

[15] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or, “A sketch-based interface for detail-preserving mesh editing,” *ACM Transactions on Graphics (SIGGRAPH 2005 Conference Proceedings)*, vol. 24, no. 3, pp. 1142–1147, 2005.

[16] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H.-Y. Shum, “Large mesh deformation using the volumetric graph laplacian,” *ACM Transactions on Graphics (SIGGRAPH 2005 Conference Proceedings)*, vol. 24, no. 3, pp. 496–503, 2005.

[17] J. Huang, X. Shi, X. Liu, K. Zhou, L.-Y. Wei, S.-H. Teng, H. Bao, B. Guo, and H.-Y. Shum, “Subspace gradient domain mesh deformation,” *ACM Transactions on Graphics (SIGGRAPH 2006 Conference Proceedings)*, vol. 25, no. 3, pp. 1126–1134, 2006.

[18] D. Zorin, P. Schröder, and W. Sweldens, “Interactive multiresolution mesh editing,” in *ACM SIGGRAPH 1997 Conference Proceedings*, pp. 259–268, 1997.

[19] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel, “Interactive multiresolution modeling on arbitrary meshes,” in *ACM SIGGRAPH 1998 Conference Proceedings*, pp. 105–114, 1998.

[20] F.-C. Wu, W.-C. Ma, R.-H. Liang, B.-Y. Chen, and M. Ouhyoung, “Domain connected graph: the skeleton of a closed 3d shape for animation,” *The Visual Computer*, vol. 22, no. 2, pp. 117–135, 2006.

[21] L. Kovar and M. Gleicher, “Automated extraction and parameterization of motions in large data sets,” *ACM Transactions on Graphics (SIGGRAPH 2004 Conference Proceedings)*, vol. 23, no. 3, pp. 559–568, 2004.

[22] A. Witkin and M. Kass, “Spacetime constraints,” in *ACM SIGGRAPH 1998 Conference Proceedings*, pp. 159–168, 1988.

[23] A. Witkin and Z. Popović, “Motion warping,” in *ACM SIGGRAPH 1995 Conference Proceedings*, pp. 105–108, 1995.

[24] A. Mohr, L. Tokheim, and M. Gleicher, “Direct manipulation of interactive character skins,” in *Proceedings of the 2003 Symposium on Interactive 3D Graphics*, pp. 27–30, 2003.

[25] G. Taubin, “A signal processing approach to fair surface design,” in *ACM SIGGRAPH 1995 Conference Proceedings*, pp. 351–358, 1995.

[26] J. P. Lewis, M. Cordner, and N. Fong, “Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation,” in *ACM SIGGRAPH 2000 Conference Proceedings*, pp. 165–172, 2000.

[27] P.-P. J. Sloan, C. F. Rose, and M. F. Cohen, “Shape by example,” in *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pp. 135–143, 2001.

[28] X. C. Wang and C. Phillips, “Multi-weight enveloping: least-squares approximation techniques for skin animation,” in *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 129–138, 2002.

[29] P. G. Kry, D. L. James, and D. K. Pai, “Eigenskin: real time large deformation character skinning in hardware,” in *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 153–159, 2002.

[30] M. Garland and P. Heckbert, “Surface simplification using quadric error metrics,” in *ACM SIGGRAPH 1997 Conference Proceedings*, pp. 209–216, 1997.

[31] C. DeCoro and S. Rusinkiewicz, “Pose-independent simplification of articulated meshes,” in *Proceedings of the 2005 Symposium on Interactive 3D graphics and Games*, pp. 17–24, 2005.

[32] D. Levin, “Mesh-independent surface interpolation,” in *Geometric Modeling for Scientific Visualization*, pp. 37–49, Springer-Verlag, 2004.