Visual Comput (2005) 21: 551–558
DOI 10.1007/s00371-005-0333-z

ORIGINAL ARTICLE

Bing-Yu Chen
Yutaka Ono
Tomoyuki Nishita

Character animation creation using hand-drawn sketches

Published online: 31 August 2005

B.-Y. Chen (✉)
National Taiwan University
robin@ntu.edu.tw

Y. Ono · T. Nishita
The University of Tokyo
ono@nis-lab.is.s.u-tokyo.ac.jp,
nis@is.s.u-tokyo.ac.jp

Abstract To create a character animation, a 3D character model is often needed. However, since humanlike characters are not rigid bodies, to deform the character model to fit each animation frame is tedious work. Therefore, we propose an easy-to-use method for creating a set of consistent 3D character models from some hand-drawn sketches while keeping the projected silhouettes and features of the created models consistent with the input sketches. Since the character models possess vertexwise correspondences, they can be used for frame-consistent texture mapping or for making character animations. In our system, the user only needs to annotate the correspondence of the features among the input-vector-based sketches; the remaining processes are all performed automatically.

Keywords Cel animation · Nonphotorealistic rendering · 3D morphing · Consistent mesh parameterization · Sketches

1 Introduction

The techniques of computer graphics are widely used for supporting the animation creation process. In the traditional approach, animators had to draw each frame of an animation by hand on paper or cel. This has now been superseded by a method in which the frames are drawn using computer-assisted tools or by rendering the scenes using 3D models. Moreover, if the animators use vector-based drawings rather than raster-based paintings, they can scale the drawings to any size and the images thus require less storage space. Meanwhile, using 3D models the animators can add many attractive effects to their animations, e.g., shading, shadowing, or texture mapping, which are difficult or time-consuming to draw by hand.

However, it is still difficult to construct humanlike character models to support the animation creation process, since their shapes are often drawn with considerable distortions due to the characters' motion, changing viewpoints or animators' exaggerations. Although it might be possible to make several 3D models whose shapes change with such distortions, deforming the models manually for each frame is a very time-consuming task. Therefore, we propose a method for creating a set of 3D polygon models that correspond to the input frames of some hand-drawn sketches. Animators can use the set of models for easily adding 3D effects, especially for adding shading effects or shadows to the animation and for mapping textures to a character while preserving the user-specified features with frame-to-frame coherence.

Our method takes a single cut of an animation from a sequence of vector-based images drawn by animators as the input, where each of the images contains the shape and features of a character in a certain (key) frame of the animation. The correspondences of the features between the frames are specified by the user. Our system then creates a consistent 2D base domain according to the features and silhouettes of the character on each frame. By subdividing the base domain recursively the system generates a set of consistent 2D triangle meshes that approximate the features and silhouettes of the character in each frame. After an inflation process, a set of consistent 3D polygon models is created, so that the projected silhouettes and features of the models are consistent with the input frames. Feature specification is the only process carried out by the user; the remaining processes are automatically done by our system.

The created 3D models have the following properties.

(1) **Silhouette preservation**: The projected silhouette of each created model coincides with that of the character on the corresponding original frame.
(2) **Frame-to-frame correspondences**: The created models exhibit vertexwise correspondences.
(3) **Feature preservation**: All features of the input image are embedded in the corresponding model, and the projection of these features coincides on the original frame.

The *silhouette preservation* property allows the animators to use the set of consistent models that have been created for adding shading effects or shadows, and the *frame-to-frame correspondence* property allows them to use frame-consistent texture mapping. Texture mapping with user-specified constraints along the input vectors between the input image and the model, as well as of models, is possible due to the *feature preservation* property. Moreover, since the created models possess vertexwise correspondences, the method can assist the animators in generating in-between shapes among the input frames by applying morphing techniques.

2 Related work

To add attractive and impressive 3D effects onto cel animations, animators usually require 3D geometric information. The method of obtaining 3D information for rigid objects is quite straightforward since modelers can simply be called on to construct 3D models. Those 3D models can be directly rendered by using so-called toon or comic shaders [22], together with several stylized rendering methods [7, 8, 15]. However, for humanlike characters, there seems to be no simple solution to obtaining 3D information due to their artistic distortions.

Rademacher [18] presented a typical method of creating an animation using a 3D character model, generated by a professional animator. In this method, the animator-generated 3D character model is deformed to match several reference images, and the deformed models are then interpolated to create a 3D geometry whose shape deforms with the changes of viewpoint. Martín et al. [13] also presented a related method. Since the animators manually define the 3D models at each key viewpoint, these methods were able to satisfy the three properties that are highlighted in our method, i.e., silhouette preservation, feature preservation, and frame-to-frame correspondences, and they could be used for many applications. However, manually editing the 3D models is still a time-consuming task. Li et al. [12] also provide a sketch-based method to generate character animation.

A texture mapping method for cel animation presented by Corrêa et al. [3] also uses 3D models created by animators. Although a reference model must be created manually, a simple interface for deforming the model to meet the silhouette preservation and frame-to-frame correspondence criteria are provided. Therefore, this technique may also be used for adding shading effects or shadows. However, since the feature preservation requirement is only an approximation, it cannot be used for complex textures that must critically satisfy the user-specified constraints.

In order to create a 3D model, Igarashi et al. [6] and Karpenko et al. [10] proposed easy-to-use sketching systems with which the user draws only the silhouette. The systems can then create a 3D model that satisfies the silhouette preservation requirement for a single frame. However, it is not simple to extend this to animation since the frame-to-frame correspondence criterion is obviously not considered. A method proposed by Petrović et al. [17] for adding shadows cast by the characters on the scenes requires only a small effort on the part of animators because it creates 3D models semiautomatically using the above methods. However, these models do not possess feature preservation or frame-to-frame correspondence properties, so the range of applications where these models can be used is very limited.

Some computer vision techniques could be used to construct 3D models from 2D tracking points. However, most of these techniques assume that the object is rigid, and hence they are not applicable to the characters in character animations. Several methods, e.g., those presented by Bregler et al. [2] and by Torresani et al. [21], have been proposed for generating 3D nonrigid models, but they all require a large number of images and tracking points. Therefore, these methods cannot be applied to keyframe character animation in general. Our previous work [16] also used some computer vision methods to construct a set of consistent 3D character models from an existing cel animation for adding some effects to the original animation. Hence, the methods presented in this paper are different from the previous one, since the information on some hand-drawn sketches is not as much as that on a cel animation that has more frames than the input sketches.

In this paper we aim to identify a method that can be used to create consistent 3D models featuring silhouette preservation, feature preservation, and frame-to-frame correspondence properties. Applications for these models include adding shading effects and shadows and mapping textures within the animator's constraints. The burden for the animator with this technique is not so different from the method of Petrović et al. [17].

3 System overview

In this section, the system overview is described from an animator's viewpoint. Initially, he or she loads a sequence of images, which are hand-drawn sketches, representing some keyframes of a character animation shown in Fig. 1a–c. The animator then overwrites some stroke vectors on the features of the images as shown in Fig. 1d–f. This can be done by using some commercial tools that can convert raster-based or scanned hand-drawn sketches into vector-based images. Of course, if the original input images are already vector-based like the example shown in Fig. 5a, this step can be omitted. After specifying the corresponding points and paths between the frames by adding some ID numbers, the preprocessing setup step has been completed. The animator can then obtain a set of consistent 3D models automatically, as shown in Fig. 1g–i. These models can then be used for further applications, such as texture mapping and shadowing, as shown in Fig. 1j–l.

With some complex characters, some areas of the character are hidden by others, for example, the forearms and the upper arms of the dancing bear shown in Fig. 1. In this case, the animator has to draw some stroke vectors and specify the correspondence of the missing parts by separating the input images into multiple layers. This decomposition is natural for the process of making cel animations [4].

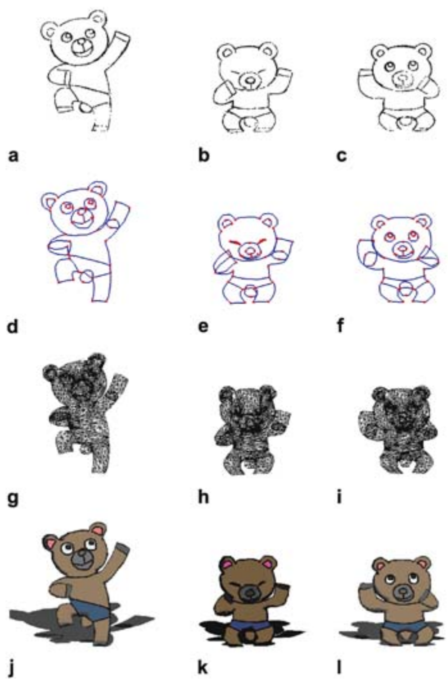

Fig. 1a–l. User-input hand-drawn sketches of a dancing bear and corresponding output models. **a–c** Three of the six input sketches. **d–f** Converted stroke vectors from input sketches with user-specified correspondences. **g–i** Output 3D models shown in wireframe. **j–l** Texture-mapped models with shadows using toon rendering

4 Generation of consistent 2D meshes

After the preprocessing setup described in Sect. 3, we now have F vector-based images as in Fig. 1d–f, where F is the number of input frames. These images are treated as F sets of 2D (x–y) planar graphs, and each graph is denoted as $G_f = (W_f, P_f)$, $f = [1, F]$, where W_f is a set of points in Re^2 and P_f is a set of finite simple paths in Re^2 connecting two different points in W_f, and each path is sampled to a polyline. Moreover, we assume that the graphs are *consistent*, which can be guaranteed by guiding the user's input, where saying two graphs are *consistent* means that there are one-to-one correspondences among their points and paths as in the two graphs shown in Fig. 2a.

To generate 2D meshes from the graphs, it is necessary to convert the graphs so that they contain no isolated points and paths as shown in Fig. 2b. Therefore, we separate our consistent 2D triangle mesh generation algorithm from a sequence of input graphs into two steps. In the first step (Sect. 4.1), we create a set of consistent base domains, which are consistent *triangulated graphs* $G'_f(G_f) = (W_f, P'_f)$ of $G_f = (W_f, P_f)$, where $P'_f \subseteq P_f$, as shown in Fig. 2c for all of the frames, which means some paths are inserted into the graph G_f to make each *patch* have only three points and paths. A *patch* is defined as a closed region bounded by the paths of the graph.

In the second step (Sect. 4.2), we create a set of consistent 2D triangle meshes M_f in which the triangulated graphs G'_f are embedded by subdividing each patch of G'_f, as shown in Fig. 2c.

4.1 Consistent graph triangulation

The algorithm described in this section creates a set of consistent base domains, which are triangulated graphs $G'_f(G_f) = (W_f, P'_f)$, from a set of consistent input graphs $G_f = (W_f, P_f)$, by adding the paths, one by one, to G_f. This method, which sequentially adds paths to the graph, is modified from the method described by Kraevoy et al. [11]. In order to describe the algorithm clearly, we use $G^*_f = (W_f, P^*_f)$ as an intermediate graph between the given set of consistent graphs $G_f = (W_f, P_f)$ and the output set of consistent base domains $G'_f(G_f) = (W_f, P'_f)$ for all of the frames. We first compute a path q_1 connecting $\{p_{1,i}, p_{1,j}\}$, where $i \neq j$ and $p_{1,i}, p_{1,j} \in W_1$, that does not cross the unbounded region and that minimizes the path length. If path q_1 is found, then

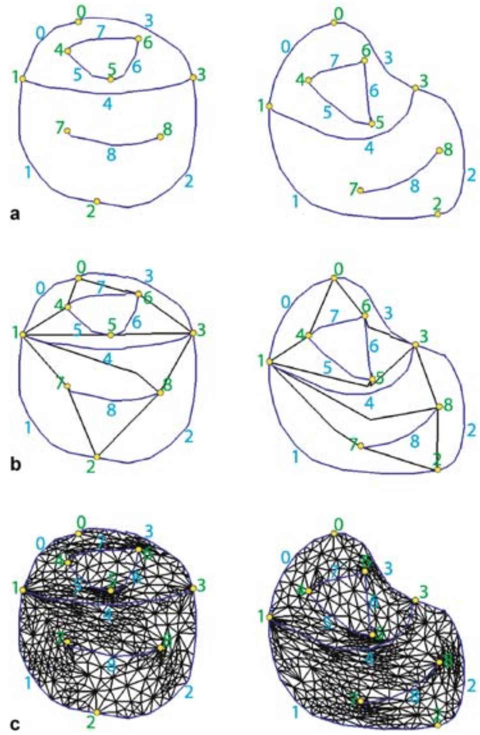

Fig. 2. a Input graphs of two frames. *Green numbers*: corresponding ID numbers for points; *blue numbers*: paths. **b** Triangulated graphs of **a**. **c** Output-consistent 2D triangle meshes

we sequentially compute paths $q_2, \ldots, q_F$ for connecting $\{p_{2,i}, p_{2,j}\}, \ldots, \{p_{F,i}, p_{F,j}\}$, respectively, where $q_2, \ldots, q_F$ are corresponding paths with q_1, so that the graphs $G_1^* = (W_1, P_1^* \cup \{q_1\}), \ldots, G_F^* = (W_F, P_F^* \cup \{q_F\})$ are still consistent.

In order to find paths $q_2, \ldots, q_F$, we need to search all possible paths in the graphs from the standpoint of the topology. To achieve this topological search and to compute the paths, we use *trapezoidal maps* of the graphs as shown in Fig. 3. The paths are generated by connecting the centroids of the trapezoids and the centers of the vertical extensions, as in Fig. 3b,c. The paths are then optimized by removal of the redundant points. The conditions for removing the redundant points are that (1) removing the point must not change the topology of the path and (2) removing the point must not involve moving a remaining path too close to other points or paths. The second condition is necessary to avoid degeneracy in the following algorithms. For example, removing the squared point in Fig. 3d is topologically possible, but it would make the path too close to other points or paths. After all possible paths are found, we will choose the path with the same topology as the path in the first frame. Once the paths $q_1, \ldots, q_F$ are decided and added to $G_1^*, \ldots, G_F^*$, we will check if $G_1^*, \ldots, G_F^*$ have become *triangulated graphs* to stop or continue the triangulation process.

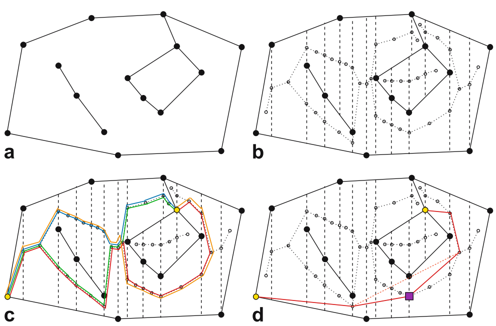

Fig. 3a–d. Path search/computation using a trapezoidal map. **a** Input graph. **b** Trapezoidal map of **a** with its "road map." **c** Four topologically different paths for connecting *yellow points*. **d** Optimized path derived from *red path* in **c**

Note that this triangulation algorithm is not symmetric, because graph G_1^* is dealt with first, and the others follow. Although it is possible to make the algorithm symmetric by further computing path sets for $G_2^*, \ldots, G_F^*$ as the first steps and then deciding the minimum average length path sets, we have found that just dealing with G_1^* in the first instance is sufficient in our experiments.

4.2 Consistent face subdivision

We now have consistent base domains $G'_f(G_f) = (W_f, P'_f)$, and each patch in a base domain has a simple boundary defined by three paths, as shown in Fig. 2b. To create consistent 2D triangle meshes M_f in which the input graphs G_f are embedded as shown in Fig. 2c, we first define 2D meshes, $M_f^* = (W_f, K_f^*)$, by identifying G'_f as meshes, where K_f^* is a simplicial complex derived from the paths of G'_f. Since the paths of G'_f are represented by polylines, M_f^* may not consist of triangles or be consistent among frames in general. To establish consistency over M_f^*, we apply an edge-split operator to M_f^* to make the boundary of each face have the same number of vertices among frames as shown in Fig. 4b. A consistent triangulation method for simple boundary polygons may then be applied to all the faces of M_f^* independently, the results of which are shown in Fig. 4c, and we use M'_f to denote the generated consistent 2D triangle meshes. The triangulation method we used here is an adaptive semiregular refinement method (a combination of 4-to-1 and 2-to-1 subdivisions).

Although M'_f are consistent 2D triangle meshes, they may have some undesirable creases due to the paths that were added for graph triangulation and thus may not be valid. Therefore, we apply a smoothing method to M'_f based on the algorithm of Freitag et al. [5] that moves each vertex locally to minimize an energy function while constraining the positions of the vertices corresponding to G_f and get M_f.

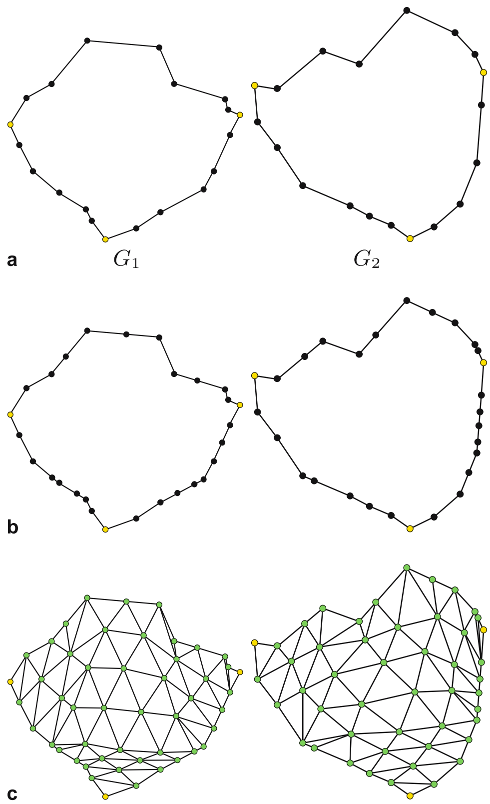

Fig. 4a–c. A process of consistent face subdivision. **a** Input patches of two base domains $\boldsymbol{G}'_1(\boldsymbol{G}_1) = (\boldsymbol{W}_1, \boldsymbol{P}'_1)$ and $\boldsymbol{G}'_2(\boldsymbol{G}_2) = (\boldsymbol{W}_2, \boldsymbol{P}'_2)$. *Yellow points*: corresponding points in $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$. **b** Results after applying edge-split operators to **a**. **c** Consistent 2D triangle meshes are generated by subdividing *b*

4.3 Consistent 3D mesh generation

After creating consistent 2D (x–y) triangle meshes $M_1, \ldots, M_F$, we inflate each vertex of the meshes to determine its z(depth)-coordinate. We apply an inflation method to the boundary vertices of M_f, based on the method proposed by Igarashi et al. [6], and create a height field for the positions inside the boundary. Each z-coordinate of the vertices of M_f is determined from the height field. Since the values of the height field for frame f are only dependent on the boundary of M_f, frame-to-frame coherence with respect to the z-coordinates is not considered. To maintain the frame-to-frame coherence, we apply an optimization method based on Taubin's paper [20], to smooth the depth of the vertices, both inter-frame and within-frame, by an iterative process.

If the characters in the input frames are composed of several layers, we create the 3D meshes separately for each layer and then bring them together as a composition. If the user has defined the same closed input paths for two different layers, the paths are regarded as *stitches*. Two meshes are combined by uniting the vertices of the stitches. After combining the layers, they are smoothly shifted and sheared so that they do not intersect with each other.

Since the animator draws the characters aesthetically, it is difficult to generate their 3D shapes *fully* automatically. In addition to the above automatic inflation method, we also provide several tools to let the animator manually adjust the z-coordinate of the vertices if necessary, such as the methods described in [17–19]. We only allow the changes in the positions of vertices along the direction of the projections in order to maintain the silhouette preservation and the feature preservation properties.

5 Results and applications

The dancing bear models shown in Fig. 1j–l were created from six input images. Three of these are shown in Fig. 1a–c. In the preprocessing step, 60 curves are drawn on each frame. By using the current system, it takes about 20 s to produce six consistent 3D models using a desktop PC with an Intel Pentium 4 1.7-GHz CPU. Each model contains 6,018 faces (triangles) and 3,280 vertices. Overwriting stroke vectors on the input hand-drawn sketches and specify the correspondences of 59 features among the six frames takes 30 min with our user interface, which is comparable to the time taken by other tools for object space morphing.

The running dinosaur models shown in Fig. 5b were also created from six input images. Three of these are shown in Fig. 5a. It takes about 40 s to produce six consistent 3D models with 86 features, each of which contains 4,862 faces and 2,550 vertices. Since the constructed dinosaur models are three-dimensional, we can change the viewpoint to render the dinosaur model as the images shown in Fig. 5c. However, since we have the input images from only one viewpoint, changing the viewpoint so

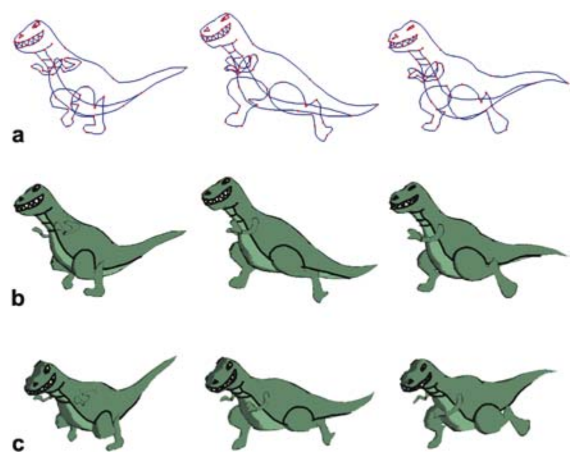

Fig. 5a–c. Example of a running dinosaur. **a** Three of six input-vector-based images. **b** Texture-mapped models using toon rendering. **c** Viewed from another viewpoint

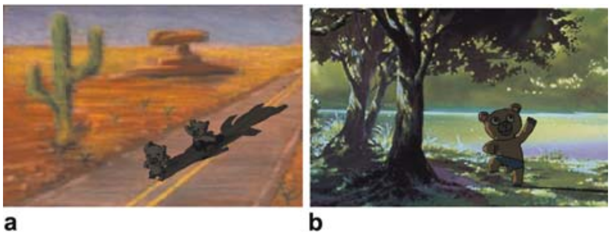

Fig. 6. a Long shadows cast by two bears. A shadow caused by one bear is cast onto another bear. The *background image* is taken from Corrêa et al.'s paper [3]. **b** Simulating effect of light through trees

Fig. 7. Simple textured models. Note that the texture on the groins of both dinosaur models is continuous

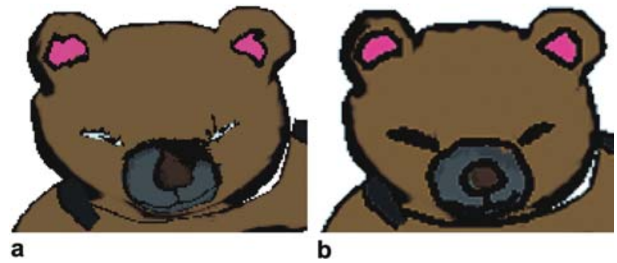

Fig. 8. a Texture-mapped model with some errors. Eyes and mouth are drawn with serpentine *lines*, which are quite different from **b**. **b** Correct texture-mapped result, which is a closeup view of Fig. 1k

that it is too large will cause some errors due to the lack of information from other viewpoints.

The created set of 3D character models can be used for the following applications.

Shading effects and shadows. Figure 6 shows two synthesized scenes that use different illumination conditions. In Fig. 6a, we use low-contrast diffusion colors to simulate an evening scene. On the other hand, in Fig. 6b, we use high-contrast diffusion colors and apply specular effects to mimic the light through trees. The lighting conditions and the position of the ground can be changed by the user. To add shading effects to the characters, we implemented our rendering algorithm on programmable graphics hardware (NVIDIA GeForce FX) and were able to render the scenes in real time. Scenes that include shading effects or shadows can be easily generated once the 3D models have been created. It typically takes 2 or 3 min to change the illumination conditions and the directions of the light sources. Since our method can be seen as an extension of the work by Petrović et al. [17], adding shadows or shading produces almost the same results as their method.

Texture mapping. Since the 3D models that were created exhibit feature-preservation and frame-to-frame correspondence properties, mapping a texture image onto each model or obtaining intermediate models is straightforward. The stroke vectors drawn on the images by the animator work as guides or constraints for texture mapping or morphing. Figure 7 shows a simple example of mapping one texture image onto two models. Note that the pattern of the texture corresponds on each model. We transfer the same texture from the first model to others by giving the same texture coordinates to the corresponding vertices, i.e., *texture transfer*. However, simple texture mapping is not sufficient in some cases. Figure 8a shows a bad example, since it produced some serpentine lines around the animator-specified stroke vectors. The model shown in Fig. 8a is the same as that shown in Fig. 8b, which is a closeup view of Fig. 1k, but the eyes and the mouth of the bear model in Fig. 8a have some errors. This unpleasant effect is caused by the differences in the vertex densities of the models in the regions of the stroke vectors, as in the models shown in Fig. 1g,h. Although we can put hard constraints onto the stroke vectors, we cannot ensure the uniformity of the vertex densities around them. This problem is closely related to the manner of drawing silhouettes in an artistic style [7, 8, 15]. Since in a character animation the features are the most important parts, the problem must be solved.

In our approach, we record the texture around each animator-specified stroke vector separately from the ordinal textures by parameterizing its width onto a vertex sequence that corresponds to the stroke vector. To render the vertex sequence corresponding to the stroke vectors, we first check whether (a part of) the vertex sequence is visible. If it is, we disable the z-buffer, generate triangle strips in the image space according to the parameterized width, and draw them directly onto the image plane. Figure 8b (also Fig. 1k) shows the modified texture-mapped model of Fig. 8a.

The texture-mapped dinosaur model shown in Fig. 7 is composed of several layers, e.g., the body and the left leg are two different, but connected, layers. By applying the same constraints to the stitches of the connected layers, the texture on connected layers can be mapped smoothly. Note that the texture on the groin of the dinosaur model shown in Fig. 7 is continuous.

Morphing and shape blending. To morph models by using their vertexwise correspondences, many 2D or 3D morphing methods [1, 9, 14] might be applied. However, most existing morphing methods are designed for morphing between two models and cannot be easily extended to more

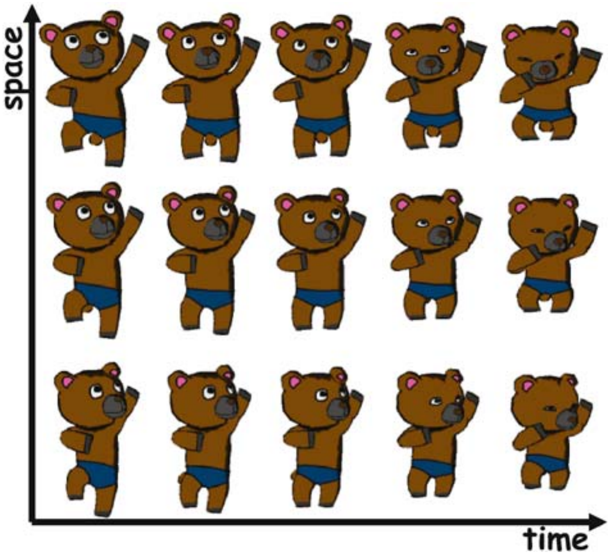

Fig. 9. *Upper row*: intermediate models constructed between models shown in Fig. 1j,k. *Middle row*: pseudoview-dependent models generated from models in *upper* and *lower rows*. *Lower row*: morphing sequence generated from two other input frames from another viewpoint

than two models. Hence, we currently choose to use a simple spline interpolation method to construct the intermediate models, as shown in the upper row of Fig. 9, for generating a smooth animation sequence. Moreover, by specifying the corresponding strokes and features on the images of two different characters, two 3D models with a consistent mesh parameterization can be constructed. Then, the morphing sequence between the two character models can also be generated. Two or more different character models can also be blended.

Pseudoview-dependent models. Given sets of input frames from different viewpoints, by specifying the corresponding strokes and features onto the images, the 3D models that interpolate the viewpoints can be constructed as shown in the middle row of Fig. 9. We thus can help the animators generate both temporal and spatial intermediate shapes. Since the models are constructed from a set of consistent 2D graphs, if there are no corresponding features on the input frames, we cannot construct the models. Therefore, even if the user gives an input of a set of images to show the back of the bear, we can still not generate a set of pseudoview-dependent models from the front of the bear to the back.

6 Conclusions and future work

In this paper, we proposed a method for creating a set of consistent 3D models from a sequence of hand-drawn strokes. The models have the following properties.

(1) Their projected silhouettes coincide with the input strokes, and therefore they can add plausible shading effects or shadows to a scene.

(2) They possess vertexwise correspondence, so it is possible to obtain a continuous animation sequence by interpolating their coordinates or to add the same textures to all of the models by propagating the texture coordinates to the corresponding vertices.

(3) The correspondence of the vertices can be controlled by the animator's drawing strokes, and thus the technique can be applied to morphing or texture mapping with constraints set by the animator.

The additional effort required of the animator, beyond the traditional process of making cel animations, is only to specify several correspondences on input strokes, where the animator would like to place constraints. Thus, our method can yield a range of applications with only a minimum of effort. Since our method can construct several 3D models with a consistent mesh parameterization for different characters, several related applications might also be achieved.

Since the 3D position of the vertices of the created models are estimated, the surfaces of the created models may be a little bumpy. This limitation is not so important when we apply toon shading to the models. When it comes to background characters as in an animation of crowds, the details of the character models are also not so necessary. Therefore, it may be acceptable to use the created models in those cases. Moreover, our method is also suitable for adding some effects to cel animation, such as what has been done in [17]. Helping traditional 2D animators to produce a prototype of 3D character animation is also one of our major contributions. The generated 3D animated character models can then be further modified using commercial modeling tools such as Maya.

The following list of topics indicates aspects of the work that we hope to cover in the future:

(1) Reduce the labor of animators by finding the correspondences between the input images automatically or semi-automatically. This might become possible by utilizing computer vision techniques.

(2) Enable the creation of models from nonconsistent images.

(3) Given a character animation sequence, transfer the motion of the character to another character that is only drawn on a single image.

Acknowledgement We thank Dr. Ken-ichi Anjyo (Oriental Light & Magic Inc.) for providing the input images (Fig. 1). This work was partially supported by the National Science Council of Taiwan under numbers 93-2213-E-002-084.

References

1. Alexa, M., Cohen-Or, D., Levin, D.: As-rigid-as-possible shape interpolation. In: Proceedings of SIGGRAPH, pp. 157–164 (2000)
2. Bregler, C., Hertzmann, A., Biermann, H.: Recovering non-rigid 3d shape from image streams. In: Proceedings of CVPR, pp. 2690–2696 (2000)
3. Corrêa, W.T., Jensen, R.J., Thayer, C.E., Finkelstein, A.: Texture mapping for cel animation. In: Proceedings of SIGGRAPH, pp. 435–446 (1998)
4. Fekete, J.D., Bizouarn, É., Cournarie, É., Galas, T., Taillefer, F.: Tictactoon: A paperless system for professional 2-d animation. In: Proceedings of SIGGRAPH, pp. 79–90 (1995)
5. Freitag, L.A., Jones, M.T., Plassmann, P.E.: An efficient parallel algorithm for mesh smoothing. In: Proceedings of IMR, pp. 47–58 (1995)
6. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: A sketching interface for 3d freeform design. In: Proceedings of SIGGRAPH, pp. 409–416 (1999)
7. Kalnins, R.D., Davidson, P.L., Markosian, L., Finkelstein, A.: Coherent stylized silhouettes. ACM Trans. Graph. **22**(3), 856–861 (2003). (Proceedings of SIGGRAPH 2003)
8. Kalnins, R.D., Markosian, L., Meier, B.J., Kowalski, M.A., Lee, J.C., Davidson, P.L., Webb, M., Hughes, J.F., Finkelstein, A.: Wysiwyg npr: Drawing strokes directly on 3d models. ACM Trans. Graph. **21**(3), 755–762 (2002). (Proceedings of SIGGRAPH 2002)
9. Kanai, T., Suzuki, H., Kimura, F.: Metamorphosis of arbitrary triangular meshes. IEEE Comput. Graph. Appl. **20**(2), 62–75 (2000)
10. Karpenko, O., Hughes, J.F., Raskar, R.: Free-form sketching with variational implicit surfaces. Comput. Graph. Forum **21**(3), 585–594 (2002). (Proceedings of Eurographics 2002)
11. Kraevoy, V., Sheffer, A., Gotsman, C.: Matchmaker: Constructing constrained texture maps. ACM Trans. Graph. **22**(3), 326–333 (2003). (Proceedings of SIGGRAPH 2003)
12. Li, Y., Gleicher, M., Xu, Y.Q., Shum, H.Y.: Stylizing motion with drawings. In: Proceedings of SCA, pp. 309–319 (2003)
13. Martín, D., García, S., Torres, J.C.: Observer dependent deformations in illustration. In: Proceedings of NPAR, pp. 75–82 (2000)
14. Michikawa, T., Kanai, T., Fujita, M., Chiyokura, H.: Multiresolution interpolation meshes. In: Proceedings of PG, pp. 60–69 (2001)
15. Northrup, J.D., Markosian, L.: Artistic silhouettes: A hybrid approach. In: Proceedings of NPAR, pp. 31–38 (2000)
16. Ono, Y., Chen, B.Y., Nishita, T.: 3d character model creation from cel animation. In: Proceedings of CyberWorlds, pp. 210–215 (2004)
17. Petrović, L., Fujito, B., Williams, L., Finkelstein, A.: Shadows for cel animation. In: Proceedings of SIGGRAPH, pp. 511–516 (2000)
18. Rademacher, P.: View-dependent geometry. In: Proceedings of SIGGRAPH, pp. 439–446 (1999)
19. Singh, K., Fiume, E.L.: Wires: A geometric deformation technique. In: Proceedings of SIGGRAPH, pp. 405–414 (1998)
20. Taubin, G.: Curve and surface smoothing without shrinkage. In: Proceedings of ICCV, pp. 852–857 (1995)
21. Torresani, L., Hertzmann, A., Bregler, C.: Learning non-rigid 3d shape from 2d motion. In: Proceedings of NIPS, pp. 577–580 (2003)
22. Winnemöller, H., Bangay, S.: Geometric approximations towards free specular comic shading. Comput. Graph. Forum **21**(3), 309–316 (2002). (Proceedings of Eurographics 2002)

BING-YU CHEN received his B.S. and M.S. in computer science and information engineering from the National Taiwan University, Taipei, in 1995 and 1997, respectively, and his Ph.D. in information science from the University of Tokyo, Japan, in 2003. He has been an assistant professor in the Department of Information Management and the Graduate Institute of Networking and Multimedia of the National Taiwan University since 2003. His research interests are mainly in computer graphics, geometric modeling, and Web and mobile graphics. He is a member of IICM, ACM, and IEEE.

YUTAKA ONO received his B.S. and M.S. in information science and computer science from the University of Tokyo, Japan, in 2002 and 2004, respectively. He has been working for SEGA Corp. since 2004. His research interests are mainly in computer graphics, geometric modeling, and computer animation.

TOMOYUKI NISHITA received his B.S., M.S., and Ph.D. in electrical engineering from Hiroshima University, Japan, in 1971, 1973, and 1985, respectively. He worked for Mazda Motor Corp. from 1973 to 1979. He became a lecturer at Fukuyama University in 1979, associate professor in 1984, and professor in 1990. He moved to the Department of Information Science of the University of Tokyo as a professor in 1998 and has been a professor in the Department of Complexity Science and Engineering of the University of Tokyo since 1999. He received Research Award on Computer Graphics from IPSJ in 1987, and also received Steven A. Coons awards from ACM SIGGRAPH in 2005. His research interest is mainly in computer graphics. He is a member of IEICE, IPSJ, ACM, and IEEE.