3D Character Model Creation from Cel Animation

Yutaka Ono
The University of Tokyo
ono@nis-lab.is.s.u-tokyo.ac.jp

Bing-Yu Chen
National Taiwan University
robin@ntu.edu.tw

Tomoyuki Nishita
The University of Tokyo
nis@is.s.u-tokyo.ac.jp

Abstract

When creating a cel animation, the animators often use 3D character models to add some effects on the character or to generate intermediate images between the key frames. However, it is a troublesome and time-consuming task to create a 3D model. In this paper, we present an easy-to-use approach for creating a set of consistent 3D character models from the user-specified strokes on a 2D image sequence. The created consistent 3D models can be used in cel animation editing systems for adding shadowing effects, textures, etc. Moreover, since the vertices of the consistent 3D models have one-to-one correspondence among the frames, by using 3D morphing techniques, this approach can also be used to generate intermediate images between the key frames.

1. Introduction

The techniques of computer graphics are widely used for supporting the creation process of cel animations. To create a cel animation, the animator often uses 3D models to add effects. For example, toon rendering makes it possible to smoothly embed 3D models into a 2D cel animation. With these kinds of techniques, it is also possible to utilize the respective advantages of 3D models and hand-drawn 2D sketches to provide a cel animation, although creating a 3D model is a time-consuming task. For some rigid objects with less deformation such as cars or machines, since these kinds of 3D models can be used in many scenes and only need to be created just once, to create such 3D models is useful for supporting the creation process of cel animations. However, for the human-like characters in a cel animation, since their shapes are often drawn with considerable distortions due to the characters' motions, changing viewpoints, or animators' exaggerations, it is difficult to create their 3D models and only one model can not represent them in all frames. Although it might be possible to make several 3D character models whose shapes change with such distortions, deforming the models manually for each frame is a very time-consuming task. Hence, a rough 3D model is often used for adding shading effects, such as shadows, on only a few frames of a cel animation.

To create such a rough model for adding shading effects, it is easy and effective to use a sketching system with which the user can draw just the silhouette and some features of the character, such like the systems provided by Igarashi et al. [4] and Karpenko et al. [5]. Although using these small systems can create a 3D model quickly and easily, since these approaches only take the silhouette of each individual animation frame into account, it is difficult to apply these methods to add some effects on a cel animation, such as texture mapping, if the coherence cannot be ignored. Moreover, it is also a tedious and time-consuming work to adjust the 3D model created using these sketch systems, since the character in a cel animation is usually hand-drawn and has many distortions for aesthetic effect due to the characters' motion, changing viewpoints, or animators' exaggerations.

In this paper, we propose a method for creating a set of consistent 3D character models from an image sequence captured from a cel animation. The projected silhouette of each created 3D character model coincides with the silhouette of the character shown on the corresponding original frame of the input cel animation. Moreover, all of the user-specified features on the input image are embedded in the corresponding produced model and the projection of the features on the model coincides with the features shown on the original frame. Therefore, the created consistent 3D character models can be used for adding shading effects, texture mappings, etc. Furthermore, these models can also be used for generating the intermediate images in-between two original key frames.

To create such a set of consistent 3D character models, the correspondence among all of the frames should be established first through several user-specified "feature strokes". Then, by embedding the feature strokes into 3D space, the consistent 3D character models can be created. Moreover, creating a cel animation using vector-based drawings instead of using bitmap images is recently becoming more popular. Since our method can also be used by just defining the correspondence of the vectors among the frames, it

is possible to use our method smoothly to assist in the creation process of cel animations.

2. Related work

Rademacher [8] presented a typical method to create an animation using a 3D character model which is generated by a professional animator. In this method, the animator-generated 3D character model is deformed to match some reference images, and then the deformed models are interpolated to create an animation with distortion tolerance. Although the animator can use this method to create an animation by carefully editing the 3D character model, deforming the model manually is a tedious task due to the number of the key frames. Although Corrêa et al. [3] presented a method to deform a 3D model to fit an image with some user efforts, the 3D model needs to be created by an animator before using their method.

Bregler et al. [2] proposed a method that allows a 3D character model to act like an existing character in a cel animation by tracking its actions. This method also needs a well-generated 3D character model created by a professional animator. Moreover, to make this 3D character model act like the captured motion of the character in the animation, it is necessary to manually edit the 3D model.

In order to create a 3D model, Igarashi et al. [4] and Karpenko et al. [5] proposed easy-to-use sketching systems with which the user draws only the silhouette. The systems can then create a 3D model which preserves the silhouette under some conditions. Petrović et al. [6] utilized these methods to create rough 3D models from the 2D frames of a cel animation and used the generated models to add shadows to the original animation. Since their approach generates a rough and individual 3D model for each frame without correspondence among them, the generated 3D models cannot be used for other applications which need to keep the coherence.

To create a 3D model from several 2D animation frames, many computer vision techniques have been presented. However, most of them assume that the 3D model is rigid, so these methods are not applicable to non-rigid 3D models. Several methods, such as the one presented by Bregler et al. [1], were proposed for generating a 3D model and its motion, but they are also not applicable to a character model in a cel animation which has many distortions.

In this paper, we present a method to create a set of consistent 3D character models from an input 2D cel animation. Our method is different from the method used in [6], where the authors only create an individual 3D model for each frame. Since we can create a set of consistent 3D character models which preserve the user-defined correspondence among the frames, it is possible to use the set of models to add shading effects, texture mapping, or to generate intermediate character models in between the original key frames.

3. System Overview and Definition

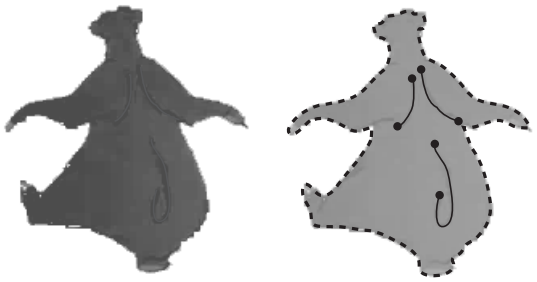
In this section, the system overview is first described briefly. To start, the user first loads a sequence of images representing a cel animation of a character. He or she then specifies the silhouettes and some curves to denote the features (feature strokes) of the character among the all frames of the input cel animation. The correspondence among all of the frames is also established at the same time. On this occasion, if the shape of the character is complex and consists of several components, this specification process should be done for each component. Then, each feature stroke is embedded into 3D space to be a 3D feature stroke. Finally, we create consistent 3D models from the silhouettes shown on the 2D images for all of the frames. The 3D feature strokes are embedded into the created 3D models to deform them. Since the created 3D models have vertex-wise correspondence, they can be used for frame-consistent texture mapping or for making 3D animation through morphing techniques.

To clarify the descriptions of our method, we begin by establishing some terminologies. The ”silhouette” is defined as the boundary of a character’s component which is our target object and is formed as a closed curve on the 2D image space. The ”feature points” are the points used to specify the correspondence among the frames. The ”feature stroke” is defined as a non-closed curve which links the feature points and its length is larger than 0. We describe the feature points and feature strokes on the image space as the 2D feature points and 2D feature strokes. Moreover, the feature points and feature strokes in the 3D space are identified as the 3D feature points and 3D feature strokes. The coordinates of the silhouette and feature strokes are defined by subtracting the mean of all of the feature points, and the origin of the local coordinate is normalized to the center of the character. Figure 1 shows an example of the silhouette, 2D feature points, and 2D feature strokes. In this paper, we assume the feature strokes are not crossed to each other on the image plane ($x - y$ space). The details of each process will be described in the following sections.

4. Feature Stroke

4.1. Feature Stroke Input

Assume the input cel animation has m frames and the silhouette, 2D feature points, and 2D feature strokes are specified by the user on frames $j = 1, 2, \cdots, m$. Furthermore, the correspondence among the frames is also specified by



(a) (b)

Figure 1. An example of silhouette, 2D feature points, and feature strokes. (a) An input image. (b) Its silhouette, feature strokes, and feature points, which are marked by a dotted line, solid lines, and the endpoints of solid lines, respectively. ©Disney

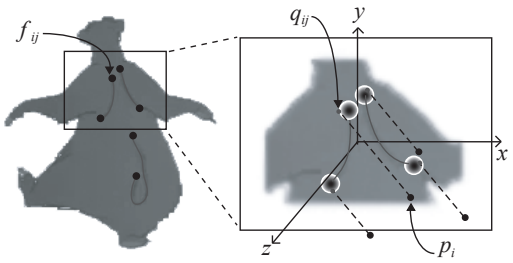


Figure 2. The diagram of the estimating function. The value of the estimating function is decided by using the distance between the 2D feature point f_{ij} and the projected 2D point q_{ij} of its corresponding 3D feature point p_i by applying R_j and s_j. ©Disney

the user as index $i = 1, 2, \cdots, n$ at the same time, where n is the maximum index number. The feature points do not need to be specified on all of the frames. The user can only specify the feature points that are visible on some certain frames. That means if a feature only appears on some frames, it can also be specified as a feature point. Similarly, the user can also specify the feature strokes that are only visible on some certain frames.

4.2. Frame Registration

The character which will be constructed as a 3D model has various motions in the input cel animation. We assume that the orthogonal projection is used for each frame. According to the user-specified 2D feature points on each frame, we approximate the relative locations of the input frames in the 3D space. That means we wish to have a rotation matrix R_j of the character and a scaling parameter s_j of the image for frame j that can approximate the location relative to the first frame.

We first use Tomasi and Kanade's method [9] to roughly approximate the 3D positions of the feature points, which have been specified on each frame by applying some scaling and rotation parameters. By using their method, although some of the feature points do not appear on all of the frames, the approximation process also works well. However, since their method works only for the rigid objects, if the object is a character in a cel animation which has dynamic motions and deformations, the approximating result may not be the proper position, or even far from it. To properly generate the 3D feature stroke (Section 4.3), we let the user be able to designate the scaling parameters s_j and rotation matrix R_j directly and clearly for each frame. To reduce the work required from the user, we modify Tomasi and Kanade's method [9] as the follows.

Assume the input cel animation has m frames and the user specifies n 2D feature points on each frame of the animation. We first use Tomasi and Kanade's method [9] to obtain the initial positions p_i of n 3D feature points, where $i = 1, 2, \cdots, n$. Then, we maximize the estimating function $\sum_{i,j} F_{ij}(q_{ij})$ for frame $j = 1, 2 \cdots, m$, where q_{ij} lies on the image plane and is the projected 2D point of p_i by using R_j and s_j. The rotation matrix $R_j(\alpha_j, \beta_j, \gamma_j)$, which is represented by Euler angles, and the scaling parameter s_j are the variables of the estimating function. $F_{ij}(x)$ is a density function and is defined as follows.

$$F_{ij}(x) = \begin{cases} (\|x - f_{ij}\| - r)^2, & \text{if } \|x - f_{ij}\| \leq r_j \\ 0, & \text{otherwise} \end{cases}, \quad (1)$$

where f_{ij} is the i-th 2D feature point on the frame j and r_j is a user-specified threshold that can be the same value for all of the frames. The estimating function means that the closer the projection of the 3D feature point and its corresponding 2D feature point, the larger the value of the estimating function. Moreover, if necessary, the user can arbitrarily fix α_j, β_j, or γ_j to increase the precision of the frame registration. Through this method, by setting a proper r_j, even if some of the feature points are moved rapidly, we can still obtain a relative closer frame registration.

4.3. 3D Feature Stroke Generation

From the scaling parameter s_j and rotation matrix R_j obtained in the previous section, we can embed the 2D feature strokes into 3D space to generate 3D feature strokes for each frame. Since the 3D feature stroke generation process can be operated independently for the feature strokes with

different index numbers, in the following description of this algorithm, we will focus on only one set of corresponding 2D feature strokes, with a specific index, that appears on all of the frames.

First, the 2D feature stroke which has the maximum length on all of the frames is selected as the "base feature stroke". Then, this base feature stoke is approximated as a polyline. The number of vertices of the polyline depends on a user-specified base segment length. The other 2D feature strokes on other frames are segmented as polylines by the same number of vertices.

If we define the 2D coordinates of the vertex to be $x-y$ values when embedding to the 3D space, the only problem in this process is defining, in the camera coordinates, the z value of each vertex which forms the polylines on each frame, and this problem is converted to minimize an estimating function. The initial z value can be set by interpolating the z values of the feature points obtained by Tomasi and Kanade's method [9] described in the pervious section. The estimating function G for a set of 3D feature strokes $S = \{S_j | 1 \leq j \leq m\}$, where $S_j = (v_1^j, v_2^j, \ldots, v_{l+1}^j)$, with number of segments l on frame j is defined as the following formulas:

$$G(S) = \sum_{1 \leq j \leq m} (V(S_j) + \varepsilon K(S_j)), \quad (2)$$

$$V(S_j) = \|L(S_j) - \frac{1}{m} \sum_{1 \leq j' \leq m} L(S_{j'})\|^2, \quad (3)$$

$$L(S_j) = \sum_{1 \leq k \leq l} \|s_j^{-1} R_j^{-1} (v_{k+1}^j - v_k^j)\|, \quad (4)$$

$$K(S_j) = \sum_{1 \leq k \leq l+1} \|M(v_k^j) - \frac{1}{m} \sum_{1 \leq j' \leq m} M(v_k^{j'})\|^2, \quad (5)$$

$$M(v_k^j) = s_j^{-1} R_j^{-1} v_k^j, \quad (6)$$

where $K(S_j)$ is used to prevent the 3D feature strokes from occurring on the same plane, and ε is a small positive coefficient. That means if we assume the length in the 3D space of the corresponding feature stroke among the frames does not change, this estimating function will minimize the sum of the differences between the average length and the length of each feature stroke on each frame. Although there is no guarantee that this algorithm converges to an optimized result as many other minimization problems of non-linear functions, since the value of this estimating function is usually positive, it still can converge to a minimum value.

Furthermore, if part of one 2D feature stroke can not be seen, we first segment the visible part of it by the base segment length, and the invisible part is then estimated by interpolation. Finally, the whole 2D feature stroke, including both of the original visible and estimated invisible parts, is segmented as a polyline by the same number of vertices as other 2D feature strokes. Hence, the same algorithm can be used to generate 3D feature strokes of such 2D feature strokes. Figure 3 shows the examples of 3D feature strokes. Although the details of the shapes among the frames are different, through the item of $K(S_j)$, the common shape can still be generated.

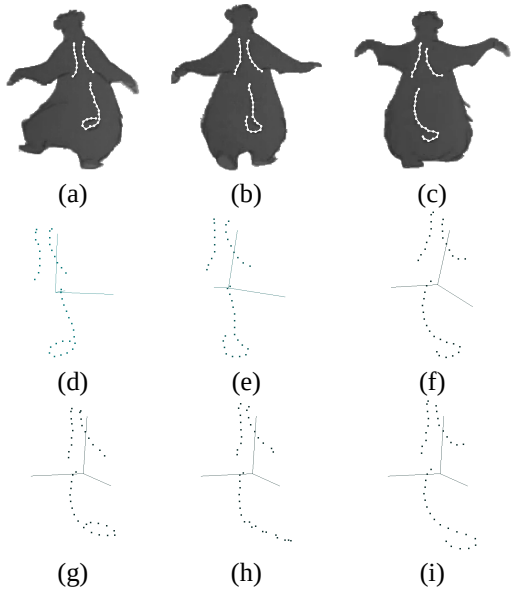


Figure 3. The process of generating 3D feature strokes. (a)∼(c) 2D feature strokes. (d)∼(f) The 3D feature strokes generated from (a)∼(c). (g)∼(i) The same 3D features strokes as (d)∼(f) but viewed from different viewpoint. ©Disney

5. Consistent 3D Models Creation

In this section, the 3D model creation method for each frame is described. Although the main idea of the inflation method is similar to the algorithm proposed by Igarashi et al. [4], instead of creating a 3D model by just inflating the vertices of triangulated silhouette along the $\pm z$ axis which is perpendicular to the image plane ($x-y$ space), we use the z value obtained from using the 3D feature strokes which is generated in the previous section. In the rest of this section, we use a simple example to describe the algorithm. In this simple example, all of the feature points and feature strokes are visible on all of the frames.

To create consistent 3D models which take into account the correspondence among the frames, we first define a set of triangles as a "common domain" [7] among the frames (Figure 4 (c)). This common domain can be obtained by constraint Delaunay triangulation algorithm which takes the connectivity of 2D feature strokes among the feature points

as the constraint. At the same time, we define the patch as the paths which connect the feature strokes and have the same connectivity as the common domain (Figure 4 (d)). If some paths of the patch intersect each other, some new vertices will be inserted to the paths to re-adjust the route to make the paths non-intersecting. We then parameterize the patch to the common domain, and this parameterization is obtained by performing a parameterization method to each corresponding triangle. Using the obtained parameterization, a recursive 4-to-1 subdivision is performed to each triangle in the common domain to embed triangles inside the silhouette (Figure 4 (e)).

Finally, by moving the position of each vertex along the $\pm z$ axis, the 3D model can be created (Figure 4 (f)). In Igarashi et al.'s method [4], they use the distance from the silhouette as the z value of an inside vertex. In our method, we set the z value by using the coordinates of 3D feature points and 3D feature strokes. That means for the vertices on the corresponding feature strokes, the z values are set as those of the vertices on the 3D feature strokes. For the other vertices, we use the z values of the vertices on the nearest 3D feature strokes and Igarashi et al.'s method [4] to decide their z values, i.e., the z value is decided as a weighted sum of the z value of the vertex on the nearest 3D feature stroke and the distance between the target vertex and the nearest 3D feature stroke.

If not all of the feature points and feature strokes are visible on all of the frames, the visible feature points and feature strokes are set as the front domain and the invisible feature points and feature strokes are set as the back domain. Then, the two domains are independently arranged in the 2D domain and the 3D model is created by inflating them independently along the $+z$ and $-z$ axes. The connectivity of the domain is decided by constraint Delaunay triangulation algorithm which sequentially adds the constraints from the feature strokes on the first frame.

6. Results

We use a cel animation with 15 frames as our example. Figure 4 shows the 3D model creation process of one frame. It takes 2$\sim$4 minutes to draw the silhouette and feature strokes on each frame by the user. We implement the algorithm of 3D feature stroke generation with MATLAB. Using a desktop PC with an Intel Pentium 4 1.7GHz CPU, it takes about 2 minutes for 3D feature stroke generation and 1 minute for consistent 3D models creation which is implemented with C++.

The created consistent 3D models are used for the following applications. Figure 5 shows the example of generating an in-between model and applying texture mapping to the created consistent 3D models. Since the correspondence among the models of each frame is defined clearly, it is easy

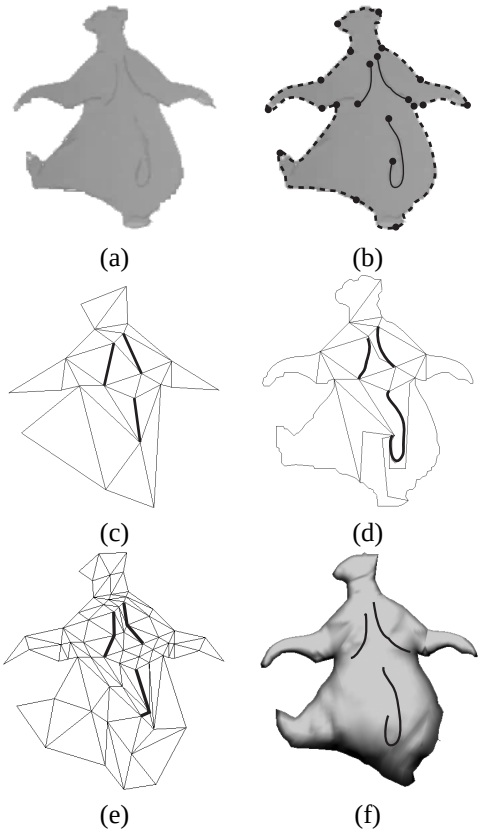


Figure 4. Creating a 3D model for each frame. (a) An input frame. (b) The silhouette, feature points, and feature strokes specified on (a). (c) The base domain defined by the silhouette and feature strokes. (d) The patch obtained from the parameterization of (c). (e) The triangle mesh generated from subdividing the base common domain once. (f) The 3D model created from inflation after subdividing the base common domain four times. The thick solid lines shown in (b)$\sim$(f) are the corresponding feature strokes. ©Disney

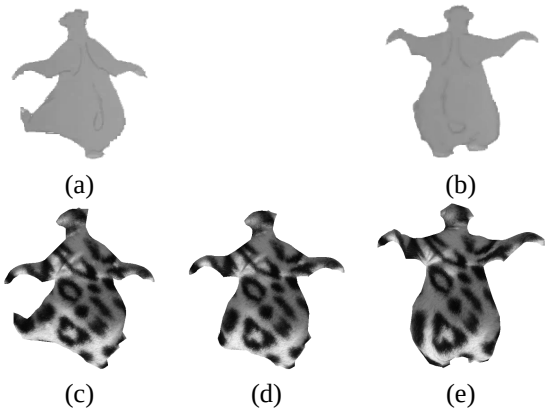


Figure 5. 3D morphing between two texture mapped models. (a), (b) Part of input frames. (c), (e) The created 3D models with texture mapping corresponded with (a) and (b). (d) The 3D model obtained from interpolating (c) and (e). The texture coordinates used in (d) and (e) are the same as (c). ©Disney

Figure 6. Adding shadows to the input animation. (a) An input frame. (b) The result of adding shadows to (a). ©Disney

to apply texture mapping on them.

Figure 6 uses two 3D models created by our method to add shadows to the original image. Figure 6 (b) shows the image obtained by performing a product operation to the original image and the shadow image which is calculated by setting the ground in the 3D space. The lighting source is set to be parallel from the right side of the image to the left side. Hence, the shadow of the right character can be mapped to the left character. Therefore, we can generate the results as those have been shown in [6]. Furthermore, by drawing more feature strokes, we can even generate the animation with complex shadows due to the bump of the character model.

7. Conclusions and Future Work

In this paper, we proposed a consistent 3D model creation method from a cel animation with user-specified corresponding silhouette and feature strokes among the frames. The 3D feature strokes are generated by considering the length and position of the corresponding feature strokes among the frames. Hence, the created consistent 3D models can used to interpolate the key frames. Through the created 3D models, we can achieve the goal of supporting cel animation, such as Figure 5 and Figure 6.

Regarding future work, we hope to extend the range of 3D model creation and automatically extract and trace the features. The length of the feature strokes is also needed to be considered when interpolating the models.

8. Acknowledgment

The input cel animation used in this paper is downloaded from the web site[1] of Bregler et al.'s paper [2]. We would like to thank them and Disney which provides the cel animation for their paper.

References

[1] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. In *Proceedings of IEEE Computer Vision and Pattern Recognition 2000*, pages 2690–2696, 2000.

[2] C. Bregler, L. Loeb, E. Chuang, and H. Deshpande. Turning to the masters: Motion capturing cartoons. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002)*, 21(3):399–407, 2002.

[3] W. T. Corrêa, R. J. Jensen, C. E. Thayer, and A. Finkelstein. Texture mapping for cel animation. In *Proceedings of ACM SIGGRAPH 1998*, pages 435–446, 1998.

[4] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3d freeform design. In *Proceedings of ACM SIGGRAPH 1999*, pages 409–416, 1999.

[5] O. Karpenko, J. F. Hughes, and R. Raskar. Free-form sketching with variational implicit surfaces. *Computer Graphics Forum (Proceedings of Eurographics 2002)*, 21(3):585–594, 2002.

[6] L. Petrović, B. Fujito, L. Williams, and A. Finkelstein. Shadows for cel animation. In *Proceedings of ACM SIGGRAPH 2000*, pages 511–516, 2000.

[7] E. Praun, W. Sweldens, and P. Schröder. Consistent mesh parameterizations. In *Proceedings of ACM SIGGRAPH 2001*, pages 179–184, 2001.

[8] P. Rademacher. View-dependent geometry. In *Proceedings of SIGGRAPH 1999*, pages 439–446, 1999.

[9] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[1] http://mrl.nyu.edu/~ bregler/tooncap/masters02.mov