Animating Characters in Pictures

Shih-Chiang Dai
National Taiwan University
jeffrey@cmlab.csie.ntu.edu.tw

Chun-Tse Hsiao
National Taiwan University
hsiaochm@cmlab.csie.ntu.edu.tw

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

ABSTRACT

Animating pictures is an interesting and useful visual effect in entertainment industry. In this paper, we present a system that allow user to animate a character in pictures in 3D space by applying 3D motion data. We use a 3D character mesh with skeleton rigged as template model. The user need to cut out silhouette of the character in pictures, and assign correspondence points between 2D character image and 3D template model. System then fits the template model to the image. Finally, the user can apply any 3D motion data to create animations in 3D space.

1. INTRODUCTION

To create a 2D character animation, traditionally, required artist to draw each frame by hand.In 3D case, it also cost hard labor modeling, skeleton rigging, applying motion data or animating by hand, and so on. In recent years, many techniques are developed to help artist reduce those work. By triangulating characters into mesh or represent by grid, one can easily reuse texture in image, and preserve the rigidity of character while deformation. Thus creating character motion becomes easy and fast. However, we don't have enough information to create a 3D animation, since what we have is only a picture. Therefore we need to make some assumptions: the object we want to animate is a human-like character, and most texture information can be get from the image. With these assumptions, the depth information can be estimated by existing models, and the lost texture information can be completed by some intuition.

Our system can use a 3D character mesh with skeleton rigged by artist, or automatically generate by [2]. What users need to do is just to cut out the silhouette of character, and to assign some feature point correspondence, system will then do model fitting for the rest. Occlusion problem can be solved by this approach separately to each part. With an easy-to-use UI, our method can help artist create animation more easily, both in 2D and 3D case. Or for fun to anyone interested in visual effects.

2. RELATED WORK

There has been many interesting idea proposed to animate a still picture. [5] animating pictures using stochastic motion textures. They animate passive elements, such as water and trees, that are subject to natural forces like wind. [10] takes an image and 3d motion data as input, similar to our work, but this method does not work for motions where the character changes its moving direction. [3] generate 3D character models from user-specified strokes, which allow them to add illumination and perspective texturing effects to 2D cel animation.

The most famous one is ARAP(as-rigid-as-possible shape manipulation) algorithm [12], which lets a user move and deform a two-dimensional shape without manually establishing a skeleton or freeform deformation (FFD) domain beforehand. Performance of their work is excellent, and UI is very user-friendly. Algorithm itself is also very simple to integrate with others. Thus, many research based on it has been developed. Most of them handle only planar motion since we don't have depth information in single image. This is the main problem we want to solve in this paper. Our goal is to achieve the same quality and performance as below, reconstruct reasonable depth information in addition. Of course, a user-friendly UI is needed.

3. OVERVIEW

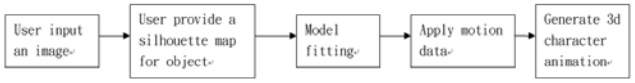

Figure 1: System overview.

Figure 1 shows our system overview. Our method takes an image as input. Users need to cut out the contour of the character. If occlusion occurs between different parts, users need to provided contours of each part separately. System will sample points on the contour as vertex in 3D space. They will be formed into a close loop we call "contour loop".

Several tools are provided for cutting out the contour [13, 4, 18], or users can modify manually to get better quality. Then the cut out region is completed by inpainting algorithms [6, 17]. We implement a grabcut algorithm [16] to solve this problem.

The next step is to modify the template model's pose if it differs too much from the character pose in the image.

System then roughly generates silhouette vertices [9], which also form a close loop we call "silhouette loop". If the initial result fails to form a close loop, or is not suitable(i.e. the loop's numbers of two types are not the same), users can modify it manually.

Now users need to assign some feature vertices of a silhouette loop corresponded to a contour loop. System then fit the silhouette to contour point. Let the silhouette vertices be handles, we then apply [12] to deform the whole template model while preserving rigidity. The skeleton is fitted then according to the barycentric coordinate we precompute before the fitting step start.Then we can adjust the z value according to average distance between silhouette vertex and bone. The lost texture information is automatically completed with some prior knowledge. Users can modify texture for better appearance. Finally we can apply any motion data to our mesh, or add any effect such as shadow, light transport in 3D space, since we have a 3D mesh with skeleton binding.

4. ALGORITHM

In this section, we describe each step of model fitting and texture completion in detail.

4.1 Silhouette Fitting

Given a contour loop C extracted from input image and a silhouette loop

$$S = \{s_1, s_2, \cdots, s_n\} \quad \text{with } s_1 = s_n$$

which is the sequence of n silhouette loop vertices correspond to template model, we have to match silhouette loop with contour loop. The final match of silhouette loop will be used as constrain vertices for Skin Fitting steps which will be discussed later. In this step user should drag m vertices

$$S' = \{s_{i(1)}, s_{i(2)}, \cdots, s_{i(m)}\} \quad \text{with } s_{i(1)} = s_{i(m)} = s_1$$

in silhouette loop to their correspondent positions manually (Note that S' is subsequence of S). Our system will fit the remaining silhouette vertices to C automatically satisfy the constrain

$$\text{length}(s_j, s_{j+1}) = \frac{\text{length}(s_{i(p)}, s_{i(p+1)})}{i(p+1) - i(p)}$$

$$\text{where} \quad i(p) \le j \le i(p+1), 1 \le p \le m-1$$

length(a, b) denote contour length along a to b. That is, we uniform distribute every vertices of silhouette loop to coutour loop with equal length between every pair of vertices in subinterval formed by S'. If there exists several S, we do the same operation separately.

Figure 2 demonstrates the principle of silhouette matching .Yellow line is the contour loop extracted from input image, blue line is the silhouette loop with vertices as blue dots and red points are user specified vertices with corresponding positions on contour.

4.2 Skin Fitting

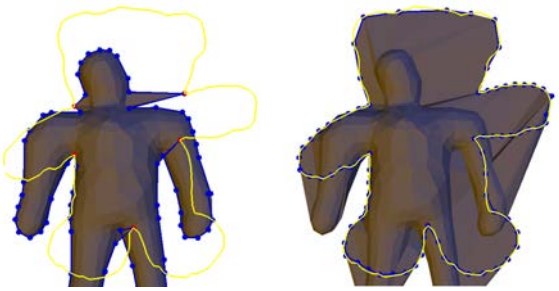

Figure 2: Silhouette fitting. Left: feature points matched by user. Right:remain S automatically fitted by system

After fitting silhouette loop with contour loop, we have to deform the shape of template model $T$to match the shape of main object in original image satisfing the constrain we have done in silhouette fitting step. In the view space with the same camera parameter in silhouette fitting, we can keep z position fixed and consider the original 3D template model as a 2D triangular mesh. For 2D triangle mesh deformation with constrained mesh vertices, the as-rigid-as-possible(ASAP) shape manipulation technique [12] in a suitable choice.

The ARAP algorithm has following steps:

$$T \Rightarrow I \Rightarrow F \Rightarrow D$$

In $T \Rightarrow I$ an intermediate shape I is determined for the given vertex constrains(In our case, S') by a Laplace-based deformation. Then original template mesh faces are fitted to faces in I rigidly with just translation and rotation, result in disconnected mesh F. Finally, we can attain the result transformed mesh D by averaging corresponding vertex positions in F. Note that the last two steps are for scaling error minimization, but in our case we do not need to preserve scaling after skin fitting step since we use ARAP to fit different models, not to deform same models for animation. Futhermore, for different target object in input image, the area different between projected template mesh and target object could be severe and we definitely do not prefer enforcing scale invariant. Based on the criteria we mentioned before, we simply discard steps $I \Rightarrow F \Rightarrow D$ but use $T \Rightarrow I$ only. By a sparse linear solver, we can fit the remain part within a second.

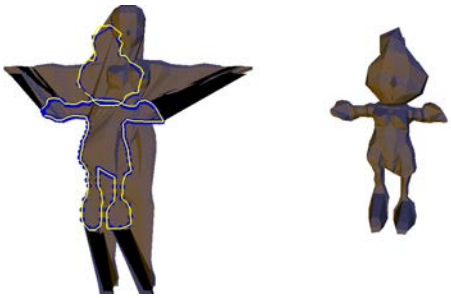

Figure 3: Skin fitting. Left: S which have been fitted to C. Right: skin fitted by ARAP algorithm

After fitting skin of template mesh as in Figure 3, we still

have to fit the original skeleton corresponding to fitted skin.

4.3 Skeleton Fitting

In this step, we have to fit the skeleton to the appropriate position related to skin. Before fitting step starts, we project the mesh and skeleton joints into xy plane, as before and record each joint position by barycentric coordinate of the triangle which contain the joint. If there exists several triangles contain the same joint, we choose the one nearest to the joint in original 3D space, and is belongs to that bone.

Let (v_1, v_2, v_3) the vertices of representative triangle, joint position with barycentric coordinate (a, b, c) before transformation can be represented as $\mathbf{v} = av_1 + bv_2 + cv_3$. After skin is fitted, new position of representative triangle become (v_1', v_2', v_3') then the join's new position could be computed as $\mathbf{v}' = av_1' + bv_2' + cv_3'$.

Figure 4 shows that the skeleton has been fitted to the right position.

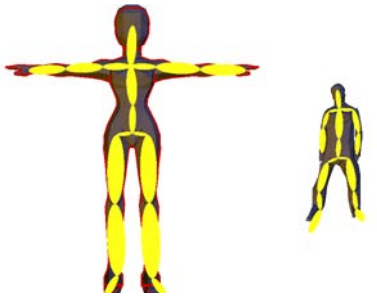

Figure 4: Skeleton fitting. Left:skeleton before adjustment. Right:skeleton after adjustment

4.4 Thickness Adjustment

After skin and skeleton fitting, we have transformed the original template mesh by adjusting xy coordinate in 2D space. However, the quantity of third coordinate(or thickness) must also be revised to generate convincing 3D target mesh. As our experiment, we observed that the distance between the bone and the skin, is highly correlated to the average distance between silhouette vertices and the bone they belong to as seen in Figure 5.

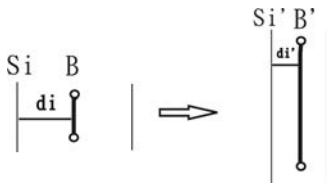

Figure 5: Left image represent thickness d_i between skin S_i and bone B. d_i' on the right is thickness after adjustment which correlated to distance between silhouette vertices and bone.

So the same, we record the average distance D for each bone before fitting step starts. If the vertex belong to several bones, we compute the average with its bone weight.

$$D = \frac{\sum \omega_i \, d_i}{N} \tag{1}$$

$$d_i = \|s_i - B\|, \; s_i \in S \tag{2}$$

where N is the numbers of vertices in S, ω_i is bone weight, and B is the bone. We have D for each bone computed, denoted by D_i.

After skin is fitted, and new joint position is got in previous step, we can now compute the new average distance D' similarly by (1). The new z value of vertices is scaled by the ratio

$$R = \sum \omega_i \, (D_i'/D_i) \tag{3}$$

the joint new z value is simply scale by the ratio D_i'/D_i.

As shown in Figure 6, the head part and the leg part seems more reasonable after adjustment.

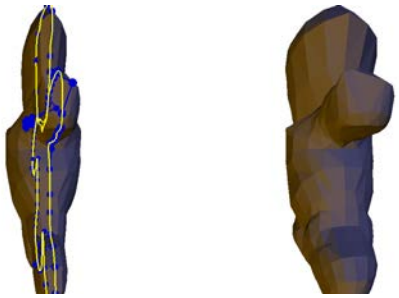

Figure 6: Thickness adjustment. Left:original thickness. Right: thickness adjusted by R

4.5 Lost Texture Completion

With a single image, the only way we can get the lost texture information is by guessing. Applying some prior knowledge, we assume the texture of back side usually mirrors front side except its head part. So we directly complete the lost texture information by mirroring. The backside of head part is usually hair, so we complete it by extend the boundary of the front side of head part. If there are still some artifacts, system allow users to modify it manually.

5. RESULTS

Our method is implemented in c++, rendering by openGL. We follow the standard linear blend skinning (LBS) to perform Character deformation. Animation can be displayed real-time. Motion data we use can be downloaded at CMU website : `http://mocap.cs.cmu.edu/`.

In Figure 7, a) is the original picture, b) is the fitted template model, c) is rendering with texture mapping, d) is animation applying motion data. In Figure 8, a) is the original character image, b) is the fitted template model, c) is rendering with texture mapping and embedding into a picture, d) is animation applying motion data. Computation time is all

a) b) c) d)

Figure 7: Results model fitting and animation.

interactive except inpainting and silhouette cut out. User interaction time is about 10 min. for a trained user.

6. CONCLUSION

The main advantage of our method is following:

- **modeling UI**: our system UI's complexity is consider easier than stroke-based method such as [11, 14], since users can take image as reference in stead of imaginary ability. And a nice template model can help us preserve feature, instead of smooth surface.
- **rendering**: The model we get after fitting step is with texture information, so we can easily render it. And since we have 3D information, we can easily apply 3D feature such as shadow, light transport effect.
- **animation**: The result mesh is rigged with skeleton, so we can easily deform it by applying motion data with good quality and effectiveness.

There are two limitation of our method. Due to lack of depth information, our animation looks weird from side view. Texture gets distortion critically around silhouette edge. It's difficult to estimate the depth information with only a picture and a template model. We may justify it by more information, such as another picture from side view, or more models for machine learning [1]. We currently let texture distortion be refined by user. More texture synthesis algorithms can be applied to refine it automatically [8, 7].

Our system can be an infrastructure for those who interested in animating characters in 3D space. High quality rendering effect can be applied to the 3D model. With more information user provided, even high quality modeling can be achieved. On the other hand, one can apply some automatic pose estimation algorithms,such as [15],to reduce the complexity of UI if he cares about simpler work flow than better results. Finally, the same framework can be extended to more kinds of object, such as rigid body, animal. It could be a useful and interesting method for many applications.

7. ACKNOWLEDGMENTS

a) b) c) d)

Figure 8: Results model fitting and animation.

This work was partially supported by the National Science Council of Taiwan under NSC95-2221-E-002-273.

8. REFERENCES

[1] B. Allen, B. Curless, and Z. Popović. The space of human body shapes: reconstruction and parameterization from range scans. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 587–594, New York, NY, USA, 2003. ACM.

[2] I. Baran and J. Popović. Automatic rigging and animation of 3d characters. *ACM Trans. Graph.*, 26(3):72, 2007.

[3] B.-Y. Chen, Y. Ono, and T. Nishita. Character animation creation using hand-drawn sketches. *The Visual Computer*, 21(8-10):551–558, 2005. Pacific Graphics 2005 Conference Proceedings.

[4] Y.-Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski. A bayesian approach to digital matting. In *Proceedings of IEEE CVPR 2001*, volume 2, pages 264–271. IEEE Computer Society, December 2001.

[5] Y.-Y. Chuang, D. B. Goldman, K. C. Zheng, B. Curless, D. H. Salesin, and R. Szeliski. Animating pictures with stochastic motion textures. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 853–860, New York, NY, USA, 2005. ACM.

[6] I. Drori, D. Cohen-Or, and H. Yeshurun. Fragment-based image completion. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 303–312, New York, NY, USA, 2003. ACM.

[7] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 1033, Washington, DC, USA, 1999. IEEE Computer Society.

[8] H. Fang and J. C. Hart. Detail preserving shape deformation in image editing. *ACM Trans. Graph.*, 26(3):12, 2007.

[9] A. Hertzmann. Introduction to 3d non-photorealistic rendering: Silhouettes and outlines. In *SIGGRAPH 99*, chapter Course Notes. ACM Press, 1999.

[10] A. Hornung, E. Dekkers, and L. Kobbelt. Character animation from 2d pictures and 3d motion data. *ACM Trans. Graph.*, 26(1):1, 2007.

[11] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: a sketching interface for 3d freeform design. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 409–416, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[12] T. Igarashi, T. Moscovich, and J. F. Hughes. As-rigid-as-possible shape manipulation. *ACM Trans. Graph.*, 24(3):1134–1141, 2005.

[13] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *ACM Trans. Graph.*, 23(3):303–308, 2004.

[14] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Fibermesh: designing freeform surfaces with 3d curves. *ACM Trans. Graph.*, 26(3):41, 2007.

[15] V. Parameswaran and R. Chellappa. View independent human body pose estimation from a single perspective image. *cvpr*, 02:16–22, 2004.

[16] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004.

[17] J. Sun, L. Yuan, L. Yuan, J. Jia, and H.-Y. Shum. Image completion with structure propagation. *ACM Trans. Graph.*, 24(3):861–868, 2005.

[18] J. Wang, M. Agrawala, and M. F. Cohen. Soft scissors: an interactive tool for realtime high quality matting. *ACM Trans. Graph.*, 26(3):9, 2007.