2008－CG－132 (10)
2008／8／22

画像中のキャラクタの３次元化とそのアニメーション

陳 炳宇[†,††] 戴 士強[†]
蕭 淳澤[†] 西田友是[††]

本稿では、我々はユーザがモーション・データ付きの３次元キャラクタ・モデルを利用することによって２次元の絵の中でのキャラクタを３次元空間でアニメーションできるシステムを提案する。このスケルトン付きの３次元キャラクタ・モデルは、２次元キャラクタのシルエットにフィティングされて、テンプレート・モデルとして使用される。２次元キャラクタ・イメージと３次元テンプレート・モデルの間に対応するポイントを指定する後、本システムはテンプレート・モデルをキャラクタ・イメージにフィティングする。さらに、キャラクタ・イメージのカラーとパターンをテンプレート・モデルのテクスチャとして転送する。最終的に、ユーザは、３次元モーション・データを利用することによって、３次元空間でフィティングされた３次元キャラクタ・モデルをアニメーションできる。我々のモデル・フィティング法はキャラクタ・モデルと剛体物とも適応できるので、２次元の絵の中での静的な物体も３次元モデルに変換できる。

Creating and Animating 3D Characters From an Image

Bing-Yu Chen,[†,††] Shih-Chiang Dai,[†] Chun-Tse Hsiao[†]
and Tomoyuki Nishita[††]

In this paper, we present a system that allows the user to animate a character in a picture in 3D space by applying an existed 3D character model with motion data. The 3D character model with skeleton rigged is used as a template model to fit the silhouette of the character in the picture. After assigning some corresponding points between the 2D character image and 3D template character model, the system then fits the model to the image and transfer the colors and patterns of the image to the model as the textures. Finally, the user can apply any motion data to animate the fitted 3D character model in 3D space. Our model fitting algorithm is general to be used for character models and rigid-body objects, so not only the character in the picture but also the static objects can be converted to be 3D models.

1. Introduction

To create a 2D character animation, traditionally, requires the artist to draw each frame manually. In 3D case, it also costs hard labor modeling, skeleton rigging, motion data applying or manually animating, etc. In recent years, many techniques are developed to deform or animate a 2D character image while preserving its rigidity by triangulating it into meshes or representing it as grids. However, to build a 3D character model from a 2D character image is still a very difficult task due to lack of 3D information. Hence, in this paper, we present a system that allows the user to generate a 3D character model from a 2D character image and animate it in 3D space by applying an existed 3D character model with motion data. Therefore, the depth information can be estimated by the existed 3D model.

Our system can use a 3D character model with skeleton rigged by the artist, or the one automatically generated by[1]. What the user needs to do is just to cut out the silhouette of the character in the picture, and assign some corresponding feature points between the 3D character model and 2D character image, our system then does the rest tasks. The occlusion problem can be solved by the same approach, but the character is needed to be separated to some parts as[2]. With an easy-to-use user-interface, our system can help the user to an-

† 国立台湾大学
National Taiwan University
†† 東京大学
The University of Tokyo

imate a character more easily both in 2D and 3D cases.

2. Related Work

There are many interesting ideas proposed to animate a still picture. Chuang *et al.*[3] presented animating pictures by using stochastic motion textures. They animate passive elements, such as water and trees, that are subject to natural forces like wind. Hornung *et al.*[8] took an image and motion data as the input and animated the character in the image. Although their work is similar to ours, the character's moving direction cannot be changed, since they did not reconstruct 3D information for the character. Chen *et al.*[2] generated 3D character models from user-provided 2D sketches, which allow the user to add illumination and perspective texturing effects to a 2D cel animation.

Igarashi *et al.*[10] proposed an as-rigid-as-possible shape manipulation algorithm, which allows the user to move and deform a 2D shape (character) without manually establishing the skeleton or freeform deformation domain beforehand. Since the performance of their method is good with a friendly user-interface and the algorithm itself is very simple to be integrated with other methods, many systems based on it has been presented. Most of them handle only planar motion since they do not have depth information in a single image, which is what we want to solve in this paper. Our goal is to achieve the same quality and performance as such systems while reconstructing reasonable depth information for the characters.

3. Overview

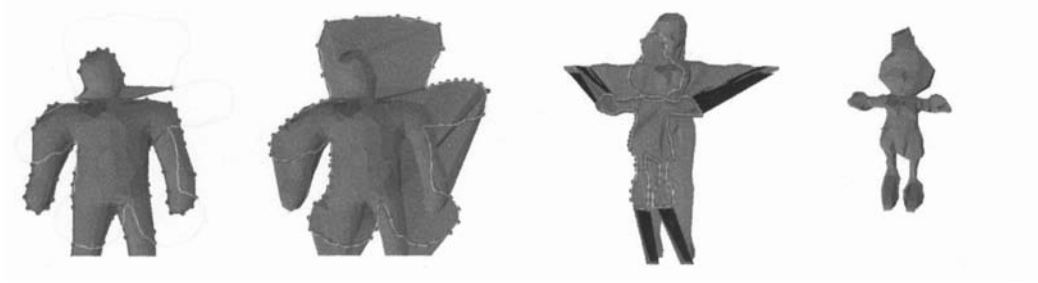
The input of our system is an image and a skeleton rigged 3D character model. The contour of the character in the image is first cut out by the user with some segregation tools[11),14),16)], and the region after cutting out the character can be completed by some inpainting methods[4),15)]. If occlusion occurs between some different parts, the user needs to provide the contour of each part separately. The system first samples some points on the contour as the vertices in 3D space, and the points form a close loop as a "contour loop" C. The pose of the input 3D character model may also need to be adjusted if it differs too much from the pose of the character image. Then, the system roughly generates silhouette vertices[7)], which also form a close loop as a "silhouette loop" S.

Then, some corresponding feature points between the silhouette loop and the contour loop are assigned by the user. The system then fits the silhouette of the character model to the contour points (Sec. 4.1). Let the silhouette vertices be the handles, the method inspired from[10)] is applied to deform the character model while preserving its rigidity (Sec. 4.2). The skeleton of the character model is then also fitted according to the barycentric coordinate (Sec. 4.3). The z value of the vertices of the character model is adjusted according to the average distance between the silhouette vertices and the skeleton (Sec. 4.4). After the character model is fitted to the character image, the colors of the character image are transferred to be the textures of the character model (Sec. 4.5). Finally, we can apply any motion data to the character model or add any visual effect such as shadow or light transport in 3D space.

4. Model Fitting

4.1 Silhouette Fitting

Given a contour loop C extracted from the input character image and a silhouette loop $S = \{s_i | i = 1, \cdots, n\}$, $s_n = s_1$ with n silhouette vertices of the character model, we have to match the silhouette loop with the contour loop. The user first drag $m \leq n$ vertices of the silhouette loop $S' = \{s_{p(j)} | j = 1, \cdots, m\} \subseteq S$, $s_{p(m)} = s_{p(1)} = s_1$ to their corresponding points of the contour loop C manually, where $p(j) = i$ denotes an index mapping from the dragged vertices $s_{p(j)}$ to the silhouette vertices s_i, and the first vertex of S and S' is treated as the same one for easy explanation, i.e., $s_{p(1)} = s_1$. Then, the system fits the remaining $n - m$ silhouette vertices to C automatically while satisfying the constraint: $\overline{s_i s_{i+1}} = \overline{s_{p(j)} s_{p(j+1)}} / (p(j+1) - p(j))$, $p(j) \leq i \leq p(j+1)$, Hence the remaining vetices of the silhouette loop are distributed uniformly to the contour loop C with equal length between every pair of the vertices in subinterval formed by S'.



(a) (b)

図 1 シルエット・フィッティング

Fig. 1 Silhouette fitting. (a) Some feature points (red points) are specified to match the contour loop C by the user. (b) The remaining vertices of the silhouette loop S are then automatically fitted to C by the system.

(a) (b)

図 2 スキン・フィッティング

Fig. 2 Skin fitting. (a) The silhouette loop S has been fitted to the contour loop C. (b) The skin of the template character model T has been fitted by as-rigid-as-possible shape manipulation method[10].

Fig. 1 demonstrates the silhouette matching. The yellow line loop is the contour loop C extracted from the input character image, the blue line loop is the silhouette loop S. The red points are the user specified vertices S' and have been dragged to their corresponding positions on the contour loop.

4.2 Skin Fitting

After fitting the silhouette loop to the contour loop, we have to deform the shape of the template character model T to match that of the original character image I while satisfying the constraint (i.e., S') we have set in Sec. 4.1. In the view space with the same camera parameter in Sec. 4.1, we can first keep the z coordinate fixed and only consider the 3D template character model T as a 2D triangular mesh T_{xy}. Then, as-rigid-as-possible shape manipulation method[10] is used for deforming the 2D triangular mesh T_{xy} with constrained mesh vertices.

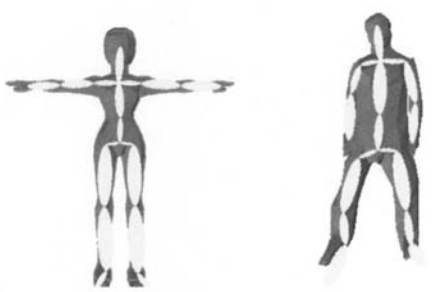
The as-rigid-as-possible shape manipulation algorithm has following steps: $T_{xy} \Rightarrow I \Rightarrow F \Rightarrow D$. In $T_{xy} \Rightarrow I$, an intermediate shape I is determined for the given vertex constrains (i.e., S' in our case) by a Laplace-based deformation. Then, the template mesh faces T_{xy} are fitted to the faces in I rigidly with just translation and rotation, and result in disconnected mesh F. Finally, we can attain the result transformed mesh D by averaging the corresponding vertex positions in F. Note that the last two steps are only for minimizing the scaling error. In our case, we do not need to preserve scaling after skin fitting since we use this method to fit different models, not to deform a model for animation. Based on this criteria, we simply discard the steps $I \Rightarrow F \Rightarrow D$ but use $T_{xy} \Rightarrow I$ only. Through a sparse linear solver, we can fit the remaining vertices of T_{xy} within a second. After fitting the skin (only $x-y$ coordinates) of the template character model T as in Fig. 2, we still have to fit the original skeleton of T corresponding to the fitted skin.

4.3 Skeleton Fitting

In this step, we have to fit the skeleton to the appropriate position related to the fitted skin. Before skeleton fitting, we first project the mesh and skeleton joints of the template character model T onto $x-y$ plane (i.e., T_{xy}), and record each joint position by barycentric coordinate of the triangle which contains the joint. If there exists several triangles contain the same joint, we choose the one nearest to the joint in the original 3D space and belongs to that bone. Fig. 3 shows the skeleton before and after fitting to the skin of the template character model T.

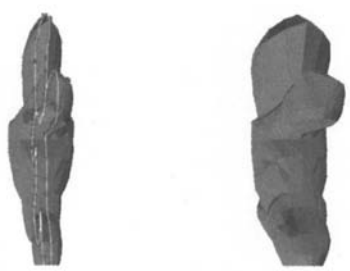
4.4 Thickness Adjustment

After fitting the skin and skeleton, we have transformed the original template character model T to fit the character image I by adjusting its projected $x-y$ coordinates in 2D space (i.e., T_{xy}). However, the quantity of the third (z) coordinate (or thick-



(a) (b)

図 3 スケルトン・フィッティング
Fig. 3 Skeleton fitting. (a) The skeleton before adjustment. (b) The skeleton after adjustment.



(a) (b)

図 4 厚さの調整
Fig. 4 Thickness adjustment. (a) The original thickness of the 3D fitted character model. (b) The thickness after adjustment.

ness) must also be revised to generate a convincing fitted 3D character model.

In our experience, we observed that the distance between the bone and the skin is highly correlated to the average distance between the silhouette vertices $s_i \in S$ and the bone $b_k \in B$ they belong to. Hence, for each bone b_k we record the average distance d_k to its nearest silhouette loop S before the model fitting. If the vertex s_i belongs to several bones, we compute the average with its bone weight ω_{ik} as $d_k = \sum_i^{n-1} \omega_{ik} d_{ik}/(n-1)$, $d_{ik} = ||s_i - b_k||$, where n is the number of $s_i \in S$ and $\omega_{ik} = 0$ if s_i and b_k have no binding relationship.

After skin fitting, we can then compute the new average distance d'_k by using the new position of s_i. Then, the new z value of each vertex is scaled by the ratio: $\sum_k \omega_{vk}(d'_k/d_k)$, where ω_{vk} is the binding weight between the vertex $v \in T$ and the bone b_k. As shown in Fig. 4, the head and leg parts become more reasonable after thickness adjustment.

4.5 Texture Completion

With one single image, it is very difficult to obtain the texture information. Hence, we assume that the texture of the back side can be usually mirrored from the front side except the head part, and directly complete the lost texture information by just mirroring. The backside of the head part is usually the hair, so we complete it by extend the boundary of the front side of head part. If there are still some artifacts, our system allows the user to modify it manually.

5. Result

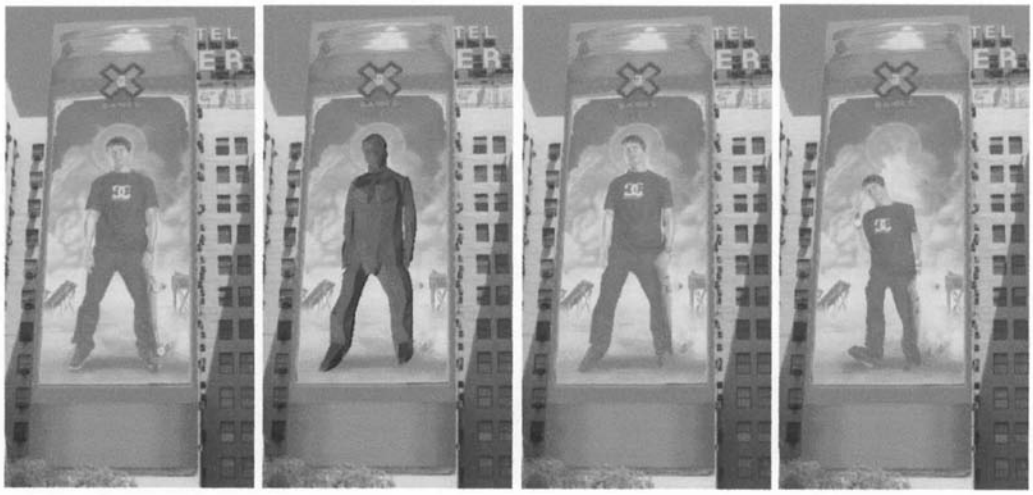
Our system is implemented in C++ with OpenGL, and the character deformation is performed by following the standard linear blend skinning (LBS) method. The motion data used in this paper is downloaded from `http://mocap.cs.cmu.edu/`. Fig. 5 and 6 show the results. The computation time is interactive except inpainting and silhouette cut out. The user interaction time is about 10 min. for a trained user.

6. Conclusion and Futurework

The main advantages of our method are as the following:

- **modeling UI**: Our system is much easier than the previous stroke-based methods such as[9),12)], since the user can take a character image as a reference. Moreover, a nice template model can help us to preserve the features instead of just smoothing the surface.
- **rendering**: Since the fitted character model has 3D information, we can easily apply any visual effect such as shadow or light transport in 3D space.
- **animation**: The fitted character model is rigged with skeleton, so we can easily deform it by applying motion data with good quality and effectiveness.

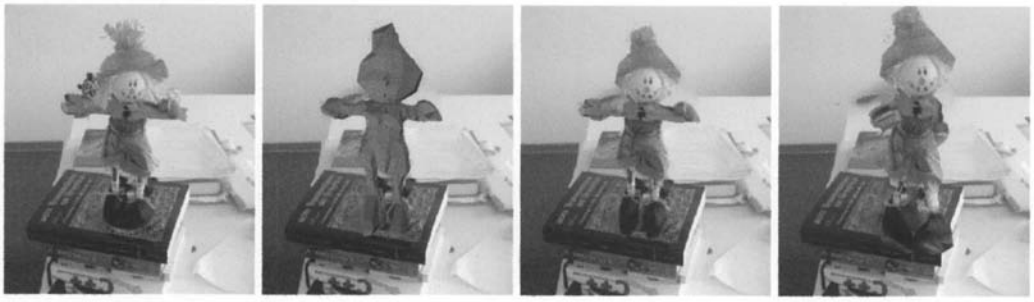
There are two major limitations of our method. Due to lack of depth information, our animation



(a) (b) (c) (d)

図 5 結果

Fig. 5 (a) The original input image. (b) The template character model after fitting the character image. (c) The character model textured by the character image. (d) The motion data is applied to animate the character model.



(a) (b) (c) (d)

図 6 結果

Fig. 6 (a) The original input image. (b) The template character model after fitting the character image. (c) The character model textured by the character image. (d) The motion data is applied to animate the character model. Besides the character in the input image, a rectangular parallelepiped is also used to fit the books in the image to let the character model "stand" on the books.

looks weird from side view and the texture has some critical distortions around the silhouette edge. It is very difficult to estimate the depth information with only a picture. We may justify it by more information, such as providing another picture from side view. Currently, the texture distortion is refined by the user. Some texture synthesis algorithms[5),6)] may be applied to refine it automatically. Furthermore, one can apply some automatic pose estimation algorithms, such as[13)], to reduce the complexity of user-interface.

謝辞 This work was partially supported by the Interchange Association, Japan and National Science Council of Taiwan under NSC95-2221-E-002-273 and NSC97-2918-I-002-028.

参 考 文 献

1) Baran, I. and Popović, J.: Automatic rigging and animation of 3D characters, *ACM Transactions on Graphics*, Vol.26, No.3, p.72 (2007). (SIGGRAPH 2007 Conference Proceedings).
2) Chen, B.-Y., Ono, Y. and Nishita, T.: Character Animation Creation using Hand-drawn Sketches, *The Visual Computer*, Vol.21, No.8-10, pp.551–558 (2005). (Pacific Graphics 2005 Conference Proceedings).
3) Chuang, Y.-Y., Goldman, D.B., Zheng, K.C., Curless, B., Salesin, D.H. and Szeliski, R.: Animating pictures with stochastic motion textures, *ACM Transactions on Graphics*, Vol.24, No.3, pp.853–860 (2005). (SIGGRAPH 2005 Conference Proceedings).
4) Drori, I., Cohen-Or, D. and Yeshurun, H.: Fragment-based image completion, *ACM Transactions on Graphics*, Vol. 22, No. 3, pp. 303–312 (2003). (SIGGRAPH 2003 Conference Proceedings).
5) Efros, A.A. and Leung, T.K.: Texture Synthesis by Non-Parametric Sampling, *Proceedings of 1999 IEEE International Conference on Computer Vision*, Vol.2, pp.1033–1038 (1999).
6) Fang, H. and Hart, J. C.: Detail preserving shape deformation in image editing, *ACM Transactions on Graphics*, Vol.26, No.3, p.12 (2007). (SIGGRAPH 2007 Conference Proceedings).
7) Hertzmann, A.: Introduction to 3D Non-Photorealistic Rendering: Silhouettes and Outlines, *ACM SIGGRAPH 1999 Conference Courses*, pp.7–11–7–14 (1999).
8) Hornung, A., Dekkers, E. and Kobbelt, L.: Character animation from 2D pictures and 3D motion data, *ACM Transactions on Graphics*, Vol.26, No.1, p.1 (2007).
9) Igarashi, T., Matsuoka, S. and Tanaka, H.: Teddy: a sketching interface for 3D freeform design, *ACM SIGGRAPH 1999 Conference Proceedings*, pp.409–416 (1999).
10) Igarashi, T., Moscovich, T. and Hughes, J.F.: As-rigid-as-possible shape manipulation, *ACM Transactions on Graphics*, Vol. 24, No. 3, pp. 1134–1141 (2005). (SIGGRAPH 2005 Conference Proceedings).
11) Li, Y., Sun, J., Tang, C.-K. and Shum, H.-Y.: Lazy snapping, *ACM Transactions on Graphics*, Vol. 23, No. 3, pp. 303–308 (2004). (SIGGRAPH 2004 Conference Proceedings).
12) Nealen, A., Igarashi, T., Sorkine, O. and Alexa, M.: FiberMesh: designing freeform surfaces with 3D curves, *ACM Transactions on Graphics*, Vol. 26, No. 3, p. 41 (2007). (SIGGRAPH 2007 Conference Proceedings).
13) Parameswaran, V. and Chellappa, R.: View Independent Human Body Pose Estimation from a Single Perspective Image, *Proceedings of 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol.2, pp.16–22 (2004).
14) Rother, C., Kolmogorov, V. and Blake, A.: "GrabCut": interactive foreground extraction using iterated graph cuts, *ACM Transactions on Graphics*, Vol.23, No.3, pp.309–314 (2004). (SIGGRAPH 2004 Conference Proceedings).
15) Sun, J., Yuan, L., Yuan, L., Jia, J. and Shum, H.-Y.: Image completion with structure propagation, *ACM Transactions on Graphics*, Vol.24, No.3, pp.861–868 (2005). (SIGGRAPH 2005 Conference Proceedings).
16) Wang, J., Agrawala, M. and Cohen, M.F.: Soft scissors: an interactive tool for realtime high quality matting, *ACM Transactions on Graphics*, Vol. 26, No. 3, p. 9 (2007). (SIGGRAPH 2007 Conference Proceedings).