Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Cartoon capture by key-frame based contour tracking[*]

Chun-luan ZHOU, Jun XIAO[‡]
(*Institute of Artificial Intelligence, School of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China*)
E-mail: chunluanzhou@163.com; junx@cs.zju.edu.cn

Received Apr. 30, 2010; Revision accepted July 7, 2010; Crosschecked July 30, 2010; Published online Aug. 18, 2010

Abstract: Traditional cartoons have been widely used in entertainment, education, and advertisement. Thus, a large amount of cartoon data is available. In this paper, we propose a new technique for capturing the motion of a character in an existing cartoon sequence. This technique tracks the contours of the cartoon character in the sequence, and key frames are used to guide the tracking. We model contour tracking as a space-time optimization problem in which an energy function including both temporal and spatial constraints is defined. First, the user labels the contours of the character on the key frames. Then, the contours on the intermediate frames are tracked by minimizing the energy function. The user may need to interactively adjust the tracking result and restart the optimization process to refine the result. Finally, an edge snapping algorithm is applied to make the tracking result more precise. Experiments show that our technique works effectively.

Key words: Cartoon reuse, Cartoon capture, Key frame, Contour tracking
doi:10.1631/jzus.C1000123 **Document code:** A **CLC number:** TP391.72

1 Introduction

Traditional cartoons have been widely used in entertainment, education, and advertisement. Therefore, a large amount of cartoon data exists. However, there has not been much study on reusing existing cartoon data. To the best of our knowledge, there are two such techniques: one is cartoon re-sequencing, and the other is cartoon capture and retargeting.

Cartoon re-sequencing synthesizes new sequences by reordering existing cartoon data. Inspired by video textures (Schodl *et al.*, 2000), de Juan and Bodenheimer (2004) proposed a technique, cartoon textures, to combine similar-looking frames of a cartoon character into a user-directed sequence. The user specified the start frame and the end frame of the cartoon character, and then a certain number of frames of this character were selected automatically from a library of cartoon data as intermediate frames to generate a smooth sequence. van Haevre *et al*. (2005) combined cartoon textures with computer-assisted animation. In their work, re-sequencing was first applied to select a sequence of key frames and the intermediate frames between two adjacent key frames were generated by interpolation. Yu *et al*. (2007) and Zhuang *et al*. (2008) also used re-sequencing to reuse cartoon data. Motion direction and edge distance were used to evaluate the similarity of different frames of a cartoon character for synthesizing visually smooth sequences. However, cartoon re-sequencing can generate only sequences of existing cartoon characters.

In some instances, we need to generate a motion of a new character which has a similar style to the style of an existing motion of another character. Bregler *et al.* (2002) presented a technique, cartoon capture and retargeting, to track the motion of an original character in a cartoon sequence and then to retarget this motion onto a new character. The key frames of the original character were first selected manually. Then, the original motion was parameterized by affine deformations and key-shape deformations. While retargeting, the user provided corre-

[‡] Corresponding author
[*] Project supported by the National Natural Science Foundation of China (No. 60903134), the National Key Technology R & D Program of China (No. 2007BAH11B00), and the Fundamental Research Funds for the Central Universities of China (No. 2009QNA5021)

sponding key frames of the new character, and the parameters of the original motion were applied to these provided key frames to generate a new motion, with the original motion style preserved. For retargeting, Wang and Li (2002) used thin-plate splines (Chui and Rangarajan, 2003) to model cartoon motion as global affine transformations and local non-affine deformations. In the work of Rastegari and Gheissari (2008), cartoon motion was represented by the combination of rigid motions and non-rigid deformations.

The key challenge for cartoon capture and retargeting is to track the motion of a character in traditional cartoon. Usually, a traditional cartoon is recorded at a low frame rate (typically 24 frames/s) so that the change between two adjacent frames is large (Fig. 1a). Furthermore, a character may undergo large degrees of deformation throughout a cartoon sequence (Fig. 1b). These make motion tracking in traditional cartoon difficult. In the work of Bregler *et al.* (2002), two techniques were used to track motion in traditional cartoon: contour capture and video capture. Contour capture was applied to contours of the cartoon character. However, the correspondences between contours in different frames should be manually specified by the user. Video capture used a vision-based region tracking technique (Lucas and Kanade, 1981) to track each part of a character individually. Since the global structure of the character was ignored and some parts of a cartoon character have similar texture, video capture was not robust. Wang and Li (2002) represented a cartoon character by its contours and the motion was tracked by shape matching. Rastegari and Gheissari (2008) captured the rigid motion by computing similarity transformations and applied Fourier transform to obtain non-rigid motions. However, the latter two methods are not suitable when occlusion happens.

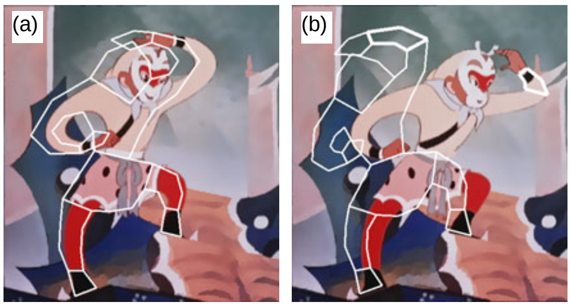

Fig. 1 Challenges for tracking in traditional cartoon
(a) The change between two adjacent frames; (b) Part deformation in a cartoon sequence

In this paper, we propose a new technique to capture the motion of a cartoon character by tracking its parts, using a key-frame based contour tracking method. Agarwala *et al.* (2004) presented a key-frame based algorithm to track curves in a video sequence. Their algorithm is based on the assumption that the change between two adjacent frames is small. It is usually not the case in a cartoon sequence and therefore their algorithm is not suitable for tracking curves in traditional cartoon. We also use key frames to guide the tracking and employ user interaction to refine the tracking result. We model contour tracking as a space-time optimization problem. An energy function that includes both temporal and spatial constraints is defined to deal with the problem that large changes exist between adjacent cartoon frames. Different from Bregler *et al.* (2002), we track the parts simultaneously and use the global structure of the character. First, the user sets curves along the contours of each part of the cartoon character on the key frames. Then, the positions of these curves on the intermediate frames are tracked by minimizing the energy function. When some aspects of the result are unsatisfactory, the user can make some adjustments and restart the optimization process to refine the result. Finally, an edge snapping algorithm is used to make the tracked curves on each intermediate frame adhere to the contours.

Our cartoon capture technique can be used by many existing cartoon-reusing systems. It can be applied in cartoon animation systems (Bregler *et al.*, 2002; Sumi and Nakajima, 2003; Agarwala *et al.*, 2004) to capture cartoon motion for generating new cartoon sequences. The part contours attained using the proposed technique can be used by some cartoon re-sequencing methods (Yu *et al.*, 2007; Zhuang *et al.*, 2008; Yang *et al.*, 2009) to evaluate the similarity of different frames of a character instead of exterior boundaries.

2 The proposed technique

We capture cartoon motion by tracking the parts of a character in a cartoon sequence. This technique includes three steps: first, the user labels the contours of the character on the key frames by drawing curves; then, a key-frame based contour tracking method is used to track the contours on the intermediate frames;

finally, an edge snapping algorithm is used to make the tracking result more precise.

2.1 Contour labeling

Given a sequence of a cartoon character, the user first selects key frames from it. The sequence is divided into several subsequences so that two adjacent subsequences are separated by a key frame, and the first frame and the last frame of each subsequence are key frames.

For a subsequence, the user sets control points along the contours of the cartoon character on the first frame and links each pair of adjacent control points by a segment. In this manner, the contours of the character on the first frame are represented by curves, and each part of the character is a polygon whose vertices are those control points on its contours. Then the curves on the first frame are copied to other frames in the subsequence and the user adjusts the curves on the last frame to the desired positions by moving the control points. Fig. 2 illustrates how a subsequence is labeled before tracking.

2.2 Key-frame based contour tracking

After contour labeling, the original cartoon sequence is divided into one or more subsequences. For each subsequence, the goal is to determine the positions of the control points on each intermediate frame and then adjacent control points are linked to form complete contours. The control points on the intermediate frames are tracked by minimizing an energy function defined below.

2.2.1 Energy function

The energy function consists of two types of terms, temporal term and spatial term. The temporal term makes the curves change smoothly in time sequence, while the spatial term forces the curves on each intermediate frame to get close to image edges and their desired image region. The energy function is defined as a linear combination of six terms:

$$E = w_{\mathrm{L}}E_{\mathrm{L}} + w_{\mathrm{C}}E_{\mathrm{C}} + w_{\mathrm{A}}E_{\mathrm{A}} + w_{\mathrm{V}}E_{\mathrm{V}} + w_{\mathrm{E}}E_{\mathrm{E}} + w_{\mathrm{R}}E_{\mathrm{R}}, \quad (1)$$

where E_{L}, E_{C}, E_{A}, and E_{V} are temporal terms, and E_{E} and E_{R} are spatial terms. w_{L}, w_{C}, w_{A}, w_{V}, w_{E}, and w_{R} are relative weights.

1. Temporal terms

The first three terms prevent the contours from changing abruptly in shape over time and the fourth term makes each control point change its velocity smoothly.

The length term E_{L} measures the length change of the curves over time:

$$E_{\mathrm{L}} = \sum_{i=1}^{N_{\mathrm{f}}-1}\sum_{j=1}^{N_{\mathrm{s}}}(\| s_i(j)\| - \| s_{i+1}(j)\|)^2, \quad (2)$$

where N_{f} and N_{s} are the numbers of frames in the subsequence and segments on the curves in each frame respectively, and $s_i(j)$ is the jth segment on the ith frame.

The curvature term E_{C} penalizes the change of curvature of the curves in time sequence:

$$E_{\mathrm{C}} = \sum_{i=1}^{N_{\mathrm{f}}-1}\sum_{j=1}^{N_{\mathrm{c}}}\| f_{\mathrm{C}}(c_i(j)) - f_{\mathrm{C}}(c_{i+1}(j))\|^2, \quad (3)$$

where N_{c} is the number of control points on each frame, and $c_i(j)$ is the jth control point on the ith frame. The function f_{C} is used to compute the curvature of a control point on the curves approximately:

$$f_{\mathrm{C}}(c_i(j)) = (c_i(a_j(1)) - c_i(j)) - (c_i(j) - c_i(a_j(2))), \quad (4)$$

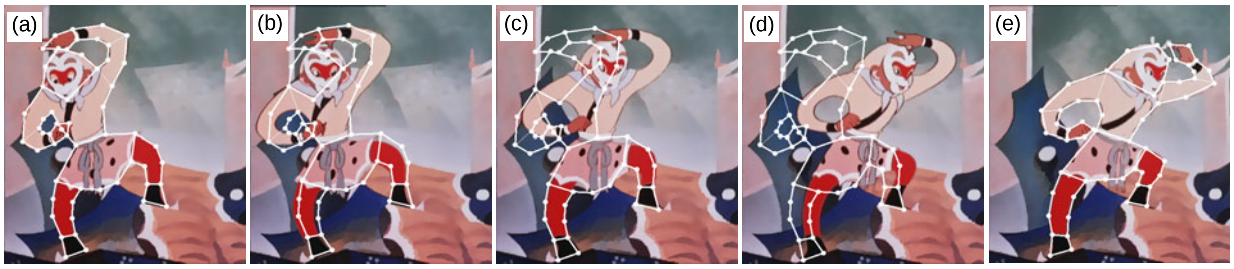

Fig. 2 Contour labeling on a subsequence

(a) and (e) are key frames; (b), (c), and (d) are three of the intermediate frames. On each frame, thick white segments form the contours of the cartoon character and thin white segments are added to show its parts explicitly where $a_j(1)$ and $a_j(2)$ are the indexes of a pair of neighboring control points of the jth control point.

The area term E_A restricts the area changes of every part of the character in the cartoon sequence:

$$E_A = \sum_{i=1}^{N_f-1}\sum_{j=1}^{N_p}(f_S(p_i(j)) - f_S(p_{i+1}(j)))^2, \quad (5)$$

where N_p is the number of the parts composing the cartoon character and $p_i(j)$ is the polygon representing the jth part on the ith frame. The function f_S computes the area of a polygon.

The velocity term E_V is defined as

$$E_V = \sum_{i=1}^{N_f-2}\sum_{j=1}^{N_c}\|(c_{i+1}(j)-c_i(j))-(c_{i+2}(j)-c_{i+1}(j))\|^2. \quad (6)$$

2. Spatial terms

Besides making the curves change smoothly in time sequence, we should also make sure that the curves get close to the image edges and that each part of the cartoon character lies on its correct image. Two spatial terms, edge term and region term, are defined to achieve these goals.

The edge term E_E forces the curves on each frame to get close to image edges. First, the canny edge detector (Canny, 1986) is used to detect edges in each frame. Due to noises in the frames, many false edges may be detected. We use mean shift (Comaniciu and Meer, 2002) to filter the noises in each frame before edge detection. Then we compute distance transform $\boldsymbol{D}_i$ for the ith edge image, where $\boldsymbol{D}_i$ is an $H\times W$ matrix with H and W being the height and the width of the edge image, respectively. $D_i(x, y)$ is the Manhattan distance from pixel (x, y) to its nearest edge pixel in edge image i. For a segment $S_i(j)$, we sample t pixels uniformly on it and denote their image coordinates as $\{(x_k, y_k)|1\le k\le t\}$. The distance from the segment $S_i(j)$ to image edges is defined as

$$f_{\text{Segdist}}(s_i(j)) = \sum_{k=1}^{t} D_i(x_k, y_k). \quad (7)$$

We have tried k from 3 to 10 in our experiments and varying k in this range has only a small effect on the result. Fig. 3 illustrates how f_{Segdist} is computed for a segment. Finally, the edge term is computed by

$$E_E = \sum_{i=2}^{N_f-1}\sum_{j=1}^{N_s} f^2_{\text{Segdist}}(s_i(j)). \quad (8)$$

With the edge term, the curves may get close to wrong edges in some frames (e.g., due to clutters in these frames). To solve this problem, the region term is defined to attract the curves to the desired positions.

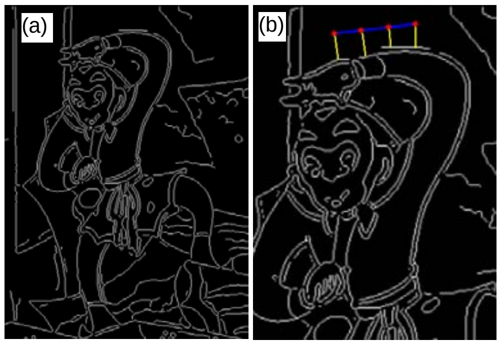

Fig. 3 Computation of the edge term
(a) An edge image; (b) The distance from a segment to image edges. Four pixels are sampled on the segment

To compute the region term, we first roughly track the positions of each part of the character in the frame sequence as follows.

1. A polygon $p_i'(j)$ is generated by linear interpolation using $p_1(j)$ and $p_K(j)$, where $K=N_f$. The textures of $p_1(j)$ and $p_K(j)$ are used respectively to compute the texture of $p_i'(j)$ by the affine coloring algorithm (Algorithm 1) and we obtain two colored polygons denoted as $q_i(j)$ and $r_i(j)$.

Algorithm 1 Affine coloring algorithm

Assume a polygon image region $\boldsymbol{A}=[\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_K]$, with $\boldsymbol{a}_i=[x_i\ y_i]^T$ being its ith vertex, is used to color another polygon image region $\boldsymbol{B}=[\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_K]$, where K is the number of vertices. The procedure of the affine coloring algorithm is listed below.

Step 1: The affine transformation $\boldsymbol{T}$ that deforms $\boldsymbol{B}$ into $\boldsymbol{A}$ by

$$\boldsymbol{A} = \boldsymbol{T}\cdot\begin{bmatrix}\boldsymbol{B}\\1\end{bmatrix}$$

is estimated via minimizing the following error term using the least-squares optimization:

$$\text{error} = \left\|\boldsymbol{A} - \boldsymbol{T}\cdot\begin{bmatrix}\boldsymbol{B}\\1\end{bmatrix}\right\|^2,$$

where $\boldsymbol{T}$ is a 2×3 matrix.

Step 2: For each pixel $\boldsymbol{p}=[x_p\, y_p]^{\mathrm{T}}$ in $\boldsymbol{B}$, $\boldsymbol{T}$ is used to obtain a point $\boldsymbol{q}=[x_q\, y_q]^{\mathrm{T}}$ by

$$[x_q\ y_q]^{\mathrm{T}} = \boldsymbol{T}\cdot\begin{bmatrix}\boldsymbol{p}\\1\end{bmatrix}.$$

If $\boldsymbol{q}$ is within $\boldsymbol{A}$, then $\boldsymbol{p}$ is colored by the color of $\boldsymbol{q}$ in image region $\boldsymbol{A}$.

Step 3: Each uncolored pixel in $\boldsymbol{B}$ is colored by the color of its nearest colored pixel.

2. Let $q_{ci}(j)$ and $r_{ci}(j)$ be the centers of $q_i(j)$ and $r_i(j)$, respectively. The center of a polygon is computed by the mean of the coordinates of its vertices in our implementation. We search in frame i for the optimal position $q_{ci}^*(j)$ of the center of $q_i(j)$ in a search window (e.g., a 100×100 rectangle) centering at $q_{ci}(j)$. By optimal position $q_{ci}^*(j)$, we mean that when $q_{ci}(j)$ is aligned to $q_{ci}^*(j)$, the color difference between $q_i(j)$ and the polygon region in the frame i where $q_i(j)$ lies is minimum. The optimal position $r_{ci}^*(j)$ of the center of $r_i(j)$ is determined in the same way.

3. The one of $q_{ci}^*(j)$ and $r_{ci}^*(j)$ with smaller color difference is selected as the optimal center, $p_{ci}^*(j)$, of the polygon $p_i(j)$.

Fig. 4 shows a tracking result of the right upper arm of a cartoon character. Occasionally, e.g., due to occlusion, the optimal center of a part in some frame obtained by the above tracking procedure may deviate from the desired position, and user interaction is allowed to adjust it to the desired position.

Denote the center of $p_i(j)$ as $p_{ci}(j)$. We define the region term as

$$E_{\mathrm{R}} = \sum_{i=2}^{N_{\mathrm{f}}-1}\sum_{j=1}^{N_{\mathrm{p}}} \| p_{ci}(j) - p_{ci}^*(j) \|^2 . \tag{9}$$

To make the region term small, the centers of every part in each frame should get close to the optimal positions. Therefore, the curves of each part are attracted to the desired region. Fig. 5 illustrates how the region term works.

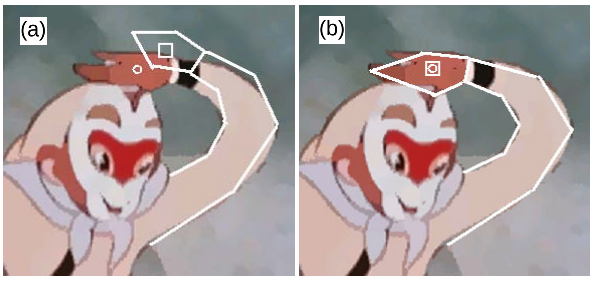

Fig. 5 Illustration of the region term

(a) The white circle shows the optimal center for the hand and the white rectangle is its current center; (b) The current center is attracted to the optimal center and the entire part gets close to its desired region

2.2.2 Optimization

Now we give the method used for minimizing the energy function E (Eq. (1)). The optimization variables are the positions of the control points on the intermediate frames. Since the control points on the key frames are set by the user, they have already located at the desired positions and are not included in the optimization variable set. Note that the energy function can be rewritten in the form

$$E = \sum_{i=1}^{N} f_i^2(u). \tag{10}$$

It is in the form of the nonlinear least-squares problem, which can be solved by a Levenberg-Marquart method (Nocedal and Wright, 2006).

To apply the Levenberg-Marquart method, we must make sure that partial derivatives of each f_i are well defined. Computing derivatives for most terms in the energy function is direct except terms in the edge term (Eq. (8)), since the function f_{Segdist} is defined by

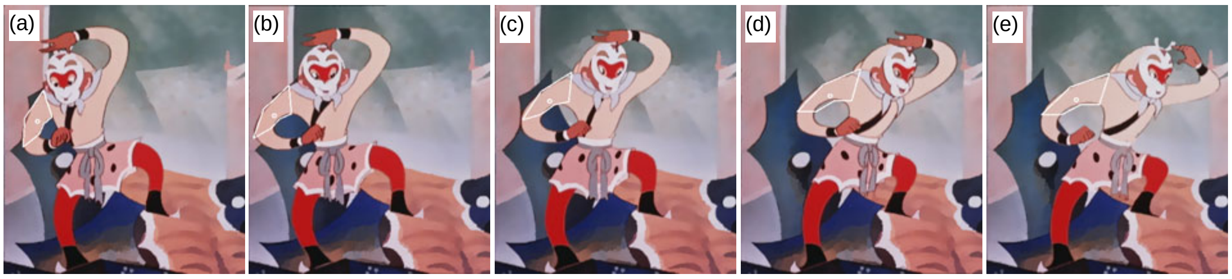

Fig. 4 Part tracking on a subsequence

(a) and (e) are key frames; (b), (c), and (d) are three of the intermediate frames. The white circles show the optimal centers of the right upper arm in the sequence

$\boldsymbol{D}_i$, whose parameters are integers. To compute the partial derivatives of f_{Segdist}, we extend the definition of $\boldsymbol{D}_i$ to real parameters.

Assume that we want to compute $D_i(x, y)$, where x and y are real numbers. Let x^{I} and y^{I} be the integer parts of x and y, respectively. Denote the fraction part of x as x^{F} and that of y as y^{F} such that

$$\begin{cases} x = x^{\text{I}} + x^{\text{F}}, \\ y = y^{\text{I}} + y^{\text{F}}. \end{cases} \tag{11}$$

We define $D_i(x, y)$ as

$$\begin{aligned} D_i(x,y) = \min\{ & D_i(x^{\text{I}}, y^{\text{I}}) + x^{\text{F}} + y^{\text{F}}, \\ & D_i(x^{\text{I}}+1, y^{\text{I}}) + 1 - x^{\text{F}} + y^{\text{F}}, \\ & D_i(x^{\text{I}}, y^{\text{I}}+1) + x^{\text{F}} + 1 - y^{\text{F}}, \\ & D_i(x^{\text{I}}+1, y^{\text{I}}+1) + 1 - x^{\text{F}} + 1 - y^{\text{F}}\}. \end{aligned} \tag{12}$$

By Eqs. (11) and (12), the partial derivatives of f_{Segdist} can be computed normally.

2.2.3 Interactive refinement

Due to the difficulty of the tracking problem, the result produced in the optimization phase may have some unsatisfactory aspects. The user can adjust some control points on the intermediate frames by moving them to the desired positions, and then restarts the optimization to refine the result.

Any control point adjusted by the user becomes a hard constraint for the optimization problem; that is, the position of this control point is fixed during the optimization process. This constraint would force control points in the current frame and adjacent frames to adjust their positions and finally to propagate to the entire frame sequence. Since the energy function (Eq. (1)) is defined in an unconstrained form, the positions of the adjusted control points are removed from the optimization variable set before the energy function is minimized.

2.3 Edge snapping

Since each part obtained by the contour tracking method is represented by a polygon and is not precise enough, we use an edge snapping algorithm to refine it; that is, each segment of a part is changed into a curve sticking to the contours (image edges). We adopt the method presented by Mortensen and Barrett (1995) to achieve this goal.

Given a segment, its two endpoints are fixed and the optimal path is found to link them. This problem is modeled as a graph searching problem. Each pixel in a frame represents a node, and there is an edge connecting it and each of its eight neighbor pixels. An edge, linking pixel $\boldsymbol{p}$ and one neighbor pixel $\boldsymbol{q}$, has a local edge cost defined by

$$l(\boldsymbol{p},\boldsymbol{q}) = w_{\text{Z}} f_{\text{Z}}(\boldsymbol{q}) + w_{\text{G}} f_{\text{G}}(\boldsymbol{q}) + w_{\text{D}} f_{\text{D}}(\boldsymbol{p},\boldsymbol{q}). \tag{13}$$

Term f_{Z} is small when $\boldsymbol{q}$ is an edge pixel and term f_{G} makes the optimal path pass through pixels of high gradient magnitudes. These two terms together force the optimal path to stick to image edges, and term f_{D} adds smooth constraints to the optimal path. Weights are set as mentioned in Mortensen and Barrett (1995): w_{Z}=0.43, w_{G}=0.43, w_{D}=0.14. The optimal path is determined by finding the minimum cumulative cost path between the two specific pixels in the graph.

3 Experimental results

For contour tracking, the weights in the energy function (Eq. (1)) have been determined experimentally: w_{L}=8, w_{C}=9, w_{A}=3, w_{V}=4, w_{E}=4, and w_{R}=32. These weights are fixed in the system and work well for all the sequences we have tried.

Fig. 6 shows a cartoon capture result on a 12-frame sequence. The first row is the result obtained automatically using the contour tracking method. Then, we adjust some control points and restart the optimization process. A better result is shown in the second row. The third row shows the more precise result after edge snapping.

Figs. 7 and 8 show another two cartoon capture results. To demonstrate the effectiveness of our cartoon capture technique, we count the number of adjusted part centers and the number of control points adjusted by the user in the cartoon capture process, compared to the total number of control points in a cartoon sequence. User interaction needed to produce results in Figs. 6–8 is given in Table 1. The control points set on key frames are not counted either in the number of adjusted control points or in the total number of control points. Only a little effort is required for the user to track the contours of a character on the intermediate frames. Before curving tracking,

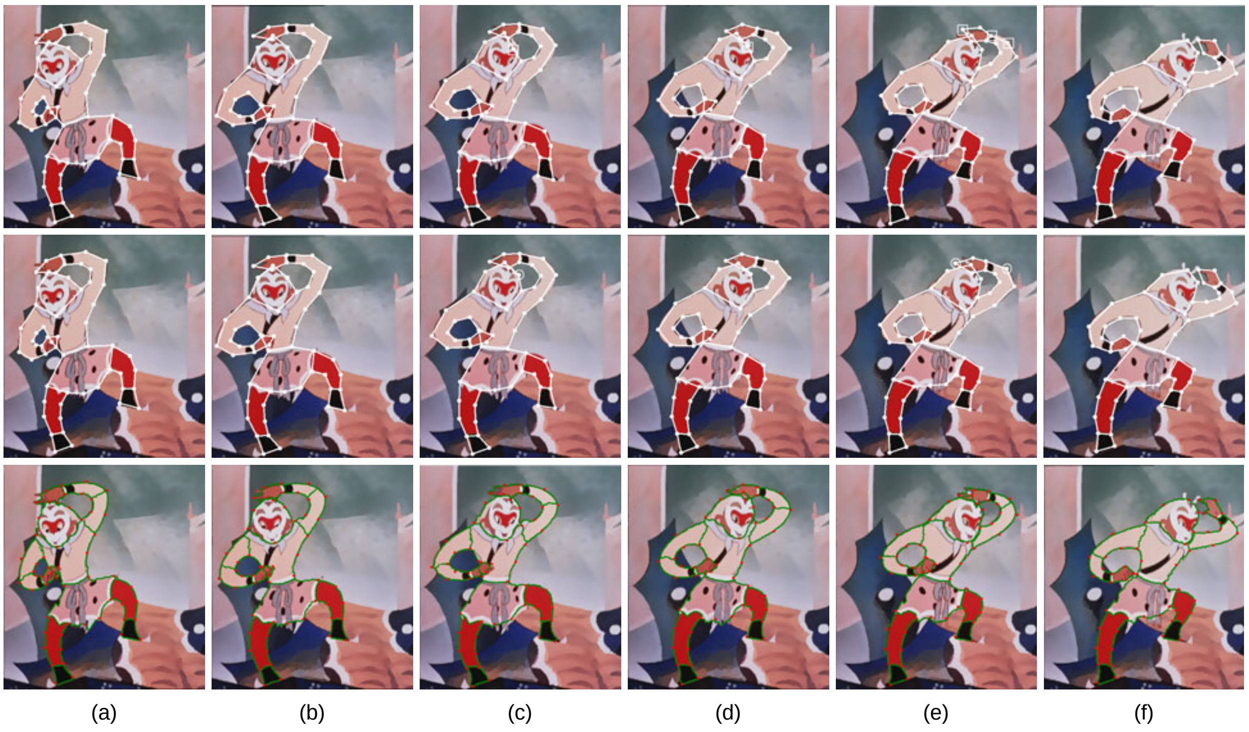

Fig. 6 Cartoon capture result of a 12-frame subsequence

(a) and (f) are key frames; (b), (c), (d), and (e) are four of the intermediate frames. Row 1: the curve tracking result without user interaction. Most of the control points on the intermediate frames are near their desired positions except for a few points marked by white squares. Row 2: a better result after refinement. The control points adjusted are highlighted by white circles. Note that we adjust the control point in (c) and the corresponding point in (d) gets closer to its correct position after the second optimization. Row 3: the more precise result after edge snapping

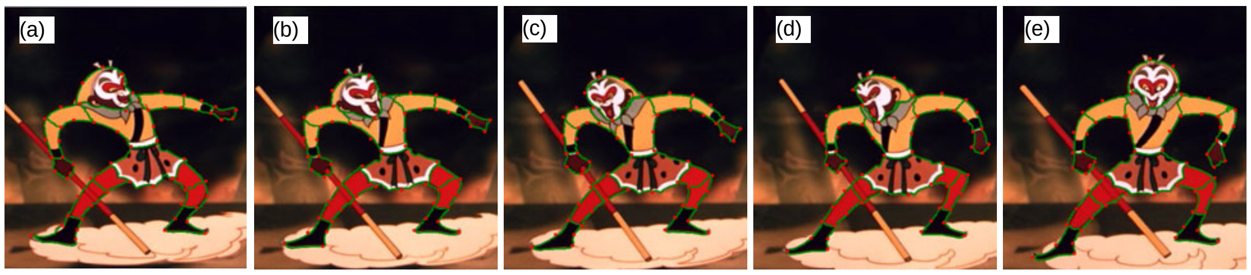

Fig. 7 Cartoon capture result of a 15-frame subsequence

(a) and (e) are key frames; (b), (c), and (d) are three of the intermediate frames

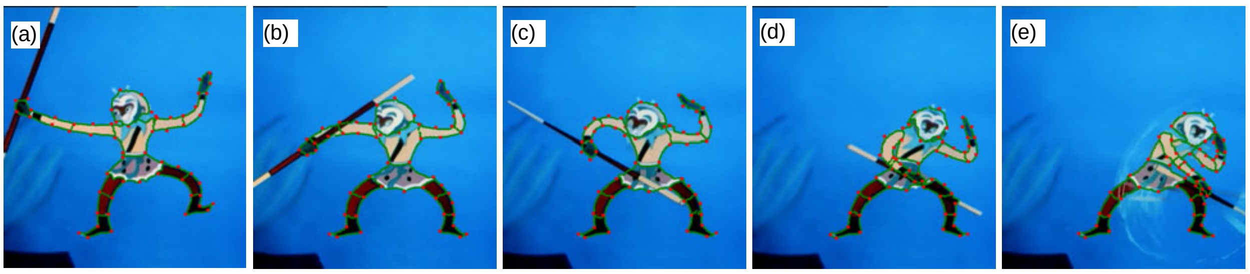

Fig. 8 Cartoon capture result of a 16-frame subsequence

(a) and (e) are key frames; (b), (c), and (d) are three of the intermediate frames the user looks over the cartoon sequence and roughly adjusts those part centers deviating from their desired positions. When refining the tracking result, the user needs only to adjust some unsatisfactory control points and to restart the optimization process. These operations can be easily done using the mouse. Compared to manually setting all the control points in every intermediate frame, much less effort is required.

Table 1 User interaction for contour tracking

Sequence	Number of part centers	Number of control points	Total number of control points	Ratio (%)[*]
Fig. 6	13	7	560	3.6
Fig. 7	21	23	754	5.8
Fig. 8	33	41	798	9.3

[*] $\text{ratio}=\frac{\text{number of part centers}+\text{number of control points}}{\text{total number of control points}}\times 100\%$

4 Conclusions

In this paper, we have proposed a new cartoon capture technique to track the motion of a character in a cartoon sequence. This technique tracks the contours of the cartoon character using a key-frame based contour tracking method. Since the difference between two adjacent frames is large and a character usually undergoes a large degree of deformation in a cartoon sequence, contour tracking in a traditional cartoon is difficult. We use key frames to guide the tracking and model contour tracking as a space-time optimization problem. The energy function defined uses both temporal and spatial constraints for tracking. The user labels contours of the cartoon character on key frames, and then the contours on intermediate frames are tracked by minimizing the energy function. User interaction is also incorporated into the contour tracking process to refine the result. Combining tracking and user interaction allows our technique to effectively capture the motion of a character in a traditional cartoon.

References

Agarwala, A., Hertzmann, A., Salesin, D.H., Seitz, S.M., 2004. Keyframe-based tracking for rotoscoping and animation. *ACM Trans. Graph.*, **23**(3):584-591. [doi:10.1145/1015706.1015764]

Bregler, C., Loeb, L., Chuang, E., Deshpande, H., 2002. Turning to the masters: motion capturing cartoons. *ACM Trans. Graph.*, **21**(3):399-407. [doi:10.1145/566654.566595]

Canny, J., 1986. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, **8**(6):679-698. [doi:10.1109/TPAMI.1986.4767851]

Chui, H., Rangarajan, A., 2003. A new point matching algorithm for non-rigid registration. *Comput. Vis. Image Understand.*, **89**(2-3):114-141. [doi:10.1016/S1077-3142(03)00009-2]

Comaniciu, D., Meer, P., 2002. Mean shift: a robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, **24**(5):603-619. [doi:10.1109/34.1000236]

de Juan, C., Bodenheimer, B., 2004. Cartoon Textures. Proc. ACM SIGGRAPH/Eurographics Symp. on Computer Animation, p.267-276. [doi:10.1145/1028523.1028559]

Lucas, B.D., Kanade, T., 1981. An Iterative Image Registration Technique with an Application to Stereo Vision. Proc. 7th Int. Joint Conf. on Artificial Intelligence, p.674-679.

Mortensen, E.N., Barrett, W.A., 1995. Intelligent Scissors for Image Composition. Proc. 22nd Annual Conf. on Computer Graphics and Interactive Techniques, p.191-198. [doi:10.1145/218380.218442]

Nocedal, J., Wright, S.J., 2006. Numerical Optimization (2nd Ed.). Springer, New York, p.258-265.

Rastegari, M., Gheissari, N., 2008. Multi-Scale Cartoon Motion Capture and Retargeting without Shape Matching. Proc. Digital Image Computing Techniques and Applications, p.320-326. [doi:10.1109/DICTA.2008.51]

Schodl, A., Szeliski, R., Salesin, D.H., Essa, I., 2000. Video Textures. Proc. 27th Annual Conf. on Computer Graphics and Interactive Techniques, p.489-498. [doi:10.1145/344779.345012]

Sumi, F., Nakajima, M., 2003. A Production Method of Reusing Existing 2D Animation Sequences. Proc. Computer Graphics Int., p.282-287. [doi:10.1109/CGI.2003.1214483]

van Haevre, W., di Fiore, F., van Reeth, F., 2005. Uniting Cartoon Textures with Computer Assisted Animation. Proc. 3rd Int. Conf. on Computer Graphics and Interactive Techniques, p.245-253. [doi:10.1145/1101389.1101437]

Wang, H., Li, H., 2002. Cartoon Motion Capture by Shape Matching. Proc. 10th Pacific Conf. on Computer Graphics and Applications, p.454-456. [doi:10.1109/PCCGA.2002.1167899]

Yang, Y., Zhuang, Y., Xu, D., Pan, Y., Tao, D., Maybank, S., 2009. Retrieval Based Interactive Cartoon Synthesis via Unsupervised Bi-distance Metric Learning. Proc. 17th ACM Int. Conf. on Multimedia, p.311-320. [doi:10.1145/1631272.1631316]

Yu, J., Zhuang, Y., Xiao, J., Chen, C., 2007. Adaptive control in cartoon data reusing. *Comput. Animat. Virt. Worlds*, **18**(4-5):571-582. [doi:10.1002/cav.189]

Zhuang, Y., Yu, J., Xiao, J., Chen, C., 2008. Perspective-aware cartoon clips synthesis. *Comput. Animat. Virt. Worlds*, **19**(3-4):355-364. [doi:10.1002/cav.258]