Region Matching with Proxy Ellipses for Coloring Hand-Drawn Animations

Yoshihiro Kanamori*
University of Tsukuba

Abstract

Hand-drawn animations are fascinating due to their expressiveness. In the production pipeline, manual coloring of line drawings is tedious and labor-intensive. We propose a method for accelerating the coloring step by matching corresponding regions in successive frames and propagating the same color to the regions. We derive a matching cost between two regions based on ellipses that approximate the orientations, scales and positions of regions. Such ellipses are insensitive to frequent changes of the region-contour shape unlike contour-based features used in previous methods. Our method finds an optimal matching that minimizes the total matching cost quickly. We demonstrate the speed and the matching accuracy of our method with comparisons.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Bitmap and framebuffer operations

Keywords: cartoon animation, colorization, region tracking, bipartite graph matching

1 Introduction

Hand-drawn 2D cartoon animations have fascinated and inspired people over the decades. Unfortunately, making a hand-drawn animation film is still quite time-consuming and labor-intensive even after assisting software tools appeared. Such making process includes the following steps; a line drawing is manually drawn on a sheet of paper and scanned for each frame. The scanned frames are cleaned up, and then colorized by manually painting each closed region in the line drawing with an appropriate color selected from a predefined color palette. Because the coloring step requires little creativity, automating the step is highly demanded.

A common strategy for automating the coloring step is to find the correspondence of regions between successive frames, and then propagate the same color to the corresponding regions. We follow this strategy as well. Many existing techniques attempt to find region-wise correspondence using the contour of each region [Madeira et al. 1996; Garcia Trigo et al. 2009]. However, contour-based features become inaccurate frequently due to changes on the contour shapes, which decreases the efficiency of contour-based methods. Instead we propose an alternative region descriptor insensitive to contours' changes; we use an ellipse that approximate the orientation, scale and position of each region. The ellipse is calculated via the centroid and the covariance matrix of pixel positions in each region. We calculate a matching cost for each pair of regions in successive frames, and find an optimal matching that minimizes the total matching cost by solving a bipartite graph matching.

*e-mail:kanamori@cs.tsukuba.ac.jp

The advantages of our method are as follows: 1) **better matching accuracy** to reduce the number of regions that must be colored manually, 2) **no additional user input** to be integrated in the existing production pipeline seamlessly, and 3) **quick response** so that the user can correct matching misses interactively. We demonstrate these advantages through several experiments and comparisons.

2 Related Work

A simple approach to color regions in multiple frames simultaneously is the classical *onion fill*, where semi-transparent frames are stacked and the user specifies a seed point contained in all the regions that the user wants to color. This requires that all regions to be colored to have a common overlapping area. The fact that this is error-prone and not easy in general makes animation studios avoid this method.

Several methods such as [Madeira et al. 1996; Garcia Trigo et al. 2009] attempt to find region-wise correspondences between two successive frames. Contour descriptors such as the string encoding of gradients and *dominant points* are often used in these methods, but they are not effective when the region contours change largely from one frame to the next one. Skeletons extracted from regions or specified manually [Qiu et al. 2006] can also be used for matching regions. However, skeletons limit the matching to articulated characters.

Sýkora et al. [2011] proposed a method for coloring and texturing a cartoon animation sequence by registering a pair of frames followed by painting with *LazyBrush* [Sýkora et al. 2009b]. However, their as-rigid-as-possible shape matching method [Sýkora et al. 2009a] needs several seconds for each pair of frames and also suffers from animations where regions undergo non-rigid motions.

Our method employs region-wise matching using proxy ellipses as shape descriptors that are insensitive to changes of contour shapes. The calculation of ellipses and matching is very fast for the whole frames, providing instant feedback to the user. Our method does not require any other user input than those in the traditional pipeline, i.e., click-and-paint operations, and thus can be seamlessly integrated into the production process.

In computer vision, elliptical descriptors are also used for matching to describe Affine transforms together with high-dimensional feature vectors including SIFT. An example is [Sivic and Zisserman 2003]. Such ellipses and feature vectors are calculated from the rich texture information, which is not available in 2D cartoon animations. We thus propose new methods for calculating ellipses and matching costs.

3 Our Region Matching Method

Given frames of line drawings (bitmap images), we first extract regions by applying a seed fill algortihm. In case that a region is not closed, it is fixed manually or filled using a sophisticated method [Sýkora et al. 2009b]. Our method then finds the correspondence between a pair of regions in two successive frames. Specifically, we solve the matching problem as a bipartite graph matching to minimize the total cost for matching. The key idea of our algorithm is to derive a cost function using an ellipse that approximates the orientation, scale and position of each region (Figure 1).

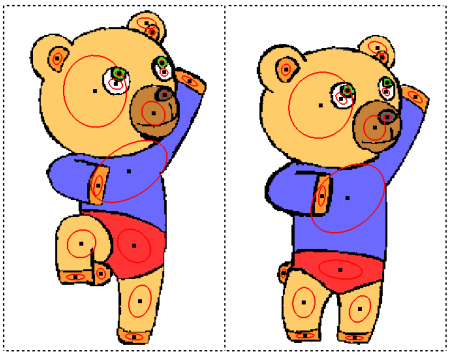

Figure 1: *Region-wise matching between two successive frames is calculated based on a* proxy ellipse *(red) for each region.*

Ellipse i of region i is calculated via the centroid $\mathbf{t}_i$ and the covariance matrix C_i of pixel positions in region i. Let the larger and smaller eigenvalues of C_i be λ_i^{max} and λ_i^{min}, and the corresponding eigenvectors be $\mathbf{e}_i^{max}$ and $\mathbf{e}_i^{min}$. The longer and shorter axes of ellipse i are then $\sqrt{\lambda_i^{max}}\mathbf{e}_i^{max}$ and $\sqrt{\lambda_i^{min}}\mathbf{e}_i^{min}$. Note that eigenvalues and eigenvectors can be calculated easily in 2D case.

Suppose two successive frames f and $f+1$ have N_f and N_{f+1} regions, respectively. To find the correspondence, we calculate a cost matrix $A = \{a_{ij}\}$ where each element a_{ij} corresponds to a matching cost between region i in frame f and region j in frame $f+1$ (where $i \in \{1, 2, \cdots, N_f\}$ and $j \in \{1, 2, \cdots, N_{f+1}\}$). We derive a matching cost a_{ij} from the differences between regions i and j in terms of the orientations, scales and positions as follows.

$$a_{ij} = w_{angle}\, a_{ij}^{angle} + w_{scale}\, a_{ij}^{scale} + w_{pos}\, a_{ij}^{pos}, \quad (1)$$

$$a_{ij}^{angle} = \cos^{-1}(\mathbf{e}_i^{max} \cdot \mathbf{e}_j^{max}), \quad (2)$$

$$a_{ij}^{scale} = \sqrt{\frac{\max\{\lambda_i^{max}, \lambda_j^{max}\}}{\min\{\lambda_i^{max}, \lambda_j^{max}\}}} + \sqrt{\frac{\max\{\lambda_i^{min}, \lambda_j^{min}\}}{\min\{\lambda_i^{min}, \lambda_j^{min}\}}} - 2, \quad (3)$$

$$a_{ij}^{pos} = \|\mathbf{t}_i - \mathbf{t}_j\|^2, \quad (4)$$

where w_{angle}, w_{scale}, and w_{pos} are weights. The positions $\mathbf{t}_i$ and $\mathbf{t}_j$ are normalized with the image size. We design the terms a_{ij}^{angle}, a_{ij}^{scale} and a_{ij}^{pos} so that they become zero if regions i and j share ellipses with the same orientation, scale and position.

To improve the matching accuracy, we also integrate the neighborhood information of regions i and j into the matching cost a_{ij}. We simply consider a centroid of neighboring regions and measure the distance between centroids in two successive frames. Neighboring centroid $\mathbf{n}_i$ for region i is calculated as follows.

$$\mathbf{n}_i = \frac{\sum_{k\in\{O_i,i\}} \alpha_k \beta_k \exp\{-\frac{\|\mathbf{t}_k - \mathbf{t}_i\|^2}{\sigma_i^2}\}\, \mathbf{t}_k}{\sum_{k\in\{O_i,i\}} \alpha_k \beta_k \exp\{-\frac{\|\mathbf{t}_k - \mathbf{t}_i\|^2}{\sigma_i^2}\}}, \quad (5)$$

$$\sigma_i = \sigma_0 \max_{k\in O_i} \|\mathbf{t}_k - \mathbf{t}_i\|, \quad (6)$$

where O_i denotes the adjacent (one-ring) neighbors of region i. α_k is the area of region k. To reduce the influence of neighboring region k that is large but contacting with region i only slightly, we multiply contact ratio $\beta_k \in [0, 1]$ which represents how much the boundary of region i contacts with region k. If region k is the background, we set $\alpha_k = \alpha_i$ and $\mathbf{t}_k = \mathbf{t}_i$. We also consider the distance between regions k and i as a weight. σ_0 is a coefficient. Finally, the following term $a_{ij}^{neighbor}$ multiplied by a weight $w_{neighbor}$ is added to the matching cost a_{ij}.

$$a_{ij}^{neighbor} = \|\mathbf{n}_i - \mathbf{n}_j\|^2. \quad (7)$$

With the cost matrix A, we solve a bipartite graph matching using the Hungarian algorithm. If the numbers of regions are not the same, i.e., $N_f \neq N_{f+1}$, some regions are left unmatched, which means that the regions do not have correspondence because they appear or disappear in either frame.

After matching among all frames is done, each region is connected in a chain of correspondence between frames. When the user assigns a color to a region, the color is propagated to the regions in subsequent frames, using chains as in [Garcia Trigo et al. 2009]. Although the user has an option to edit the chains of wrong correspondence easily as done in [Garcia Trigo et al. 2009], we just color each region independently as usual in the current production pipeline to avoid additional manipulation of chains.

Table 1: *Computational times (seconds) with different image sizes (pixel) and numbers of frames. Segmentation includes analysis of neighboring regions.*

Images	# Frames	Segmentation	Matching	Total
Fig. 2 (320×280)	5	0.017	0.010	0.027
Fig. 3 (410×240)	4	0.045	0.017	0.062
Fig. 4 (1024×768)	8	0.86	0.038	0.89

4 Results

We implemented our prototype using C++ and OpenGL, and conducted experiments on a laptop with an Intel Core i7-2820QM CPU 2.30GHz and 3.2GB RAM. The computational times are summarized in Table 1. For quantitative evaluations, we count the minimum number of regions that were assigned manually because of first appearance or incorrect matching; with better matching, more regions would be assigned with correct colors automatically. In the experimental results, each region was initially assumed as background and chains were not modified during manual coloring.

Figure 2 shows our result with the girl example of [Garcia Trigo et al. 2009]. After manual coloring of the first frame (14 regions except for background), colors were propagated along chains. Whereas about seven color assignments per frame were required with [Garcia Trigo et al. 2009]'s matching result, ours required for just three regions per frame. Considering that assignments for newly appeared regions such as the mouth in the second frame and the eyes in the third frame were mandatory, our matching quality is high.

Figure 3 shows our result with the tiger example of [Qiu et al. 2006]. Whereas [Qiu et al. 2006] require manual specification of a skeleton, our method does not require operations other than color assignments. The numbers of regions are 37, 36, 40 and 34 except for background. More than 80% of regions in each frame were assigned with correct colors.

Figure 4 shows our result with a more practical example. From the second frame, more than 82% of regions in each frame were assigned with correct colors.

5 Conclusion

In this work, we have proposed a method for semi-automating the coloring step in the current production pipeline of 2D cartoon animations. Our method constructs region-wise correspondence between two successive frames based on a cost function derived from

Figure 2: *Coloring of the girl example of Garcia Trigo et al. Top row: our matching result. Regions in the same pseudo color are in the same chain. Middle row: our coloring result. From the second to the fifth frame, the numbers of regions assigned with colors manually were 3, 4, 3 and 3 (highlighted in red). Bottom row: Garcia Trigo et al.'s matching result. 7, 9, 9 and 3 regions were manually colored, respectively.*

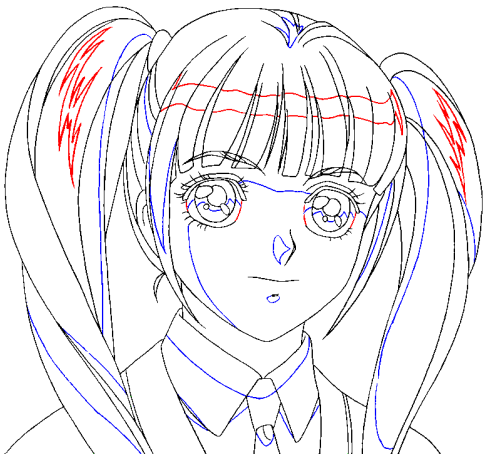

Figure 5: *Blue and red lines to specify highlighted and shadowed regions (image by courtesy of CELSYS Inc.)*

a proxy ellipse that approximates the orientation, size and position of each region. Our method yields good matching results quickly without requiring additional operations other than click-and-paint operations in the current production pipeline.

In future work, we would like to enhance the matching accuracy. One option is to manage topological changes of regions, i.e., split and fusion, mainly caused by occlusions. We also would like to handle blue and red lines that are used in production to specify highlighted and shadowed regions in line drawings (Figure 5).

References

GARCIA TRIGO, P., JOHAN, H., IMAGIRE, T., AND NISHITA, T. 2009. Interactive region matching for 2D animation coloring based on feature's variation. *IEICE Transactions (E92-D)*, 6, 1289–1295.

MADEIRA, J. S., STORK, A., AND GROSS, M. H. 1996. An approach to computer-supported cartooning. *The Visual Computer 12*, 1, 1–17.

QIU, J., SEAH, H. S., AND TIAN, F. 2006. Auto coloring with character registration. In *Proceedings of the 2006 international conference on Game research and development*, CyberGames '06, 25–32.

SIVIC, J., AND ZISSERMAN, A. 2003. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, vol. 2, 1470–1477.

SÝKORA, D., DINGLIANA, J., AND COLLINS, S. 2009. As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '09, 25–33.

SÝKORA, D., DINGLIANA, J., AND COLLINS, S. 2009. LazyBrush: Flexible painting tool for hand-drawn cartoons. *Computer Graphics Forum 28*, 2, 599–608.

SÝKORA, D., BEN-CHEN, M., ČADÍK, M., WHITED, B., AND SIMMONS, M. 2011. Textoons: practical texture mapping for hand-drawn cartoon animations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering*, NPAR '11, 75–84.

Figure 3: *Coloring of the tiger example of Qiu et al. Top row: our matching result. Whereas Qiu et al. require manual specification of a skeleton, our method does not. Bottom row: our coloring result with manual color assignments (highlighted in red). From the second to the fourth frame, the numbers of manual color assignments were 5, 8 and 7. Each frame is trimmed for display.*

Figure 4: *Coloring of a boy sequence (images by courtesy of CELSYS Inc.) First and second rows: our matching result. Third and fourth rows: our coloring results. The numbers of regions as well as manual color assignments are listed in each frame.*