MANGA COLORIZATION

Bhavye Anand Gupta *Aditya Verma* *Apurv Patani*

Indraprastha Institute of Information Technology Delhi

ABSTRACT

Manga comics contain an intensive amount of strokes, hatching, halftoning and screening. In this paper, we will study a method to colorize the manga over intensity as well as pattern continuous regions. The input is a manga page, with user scribbles that indicate the intended colors. Pattern continuity is measured using Gabor wavelet filters and boundaries are propagated using level set methods. After that, various colorization techniques are used to replace colors.

Index Terms— Image processing, Image segmentation, Image colorization

1. INTRODUCTION

Japenese mangas are seldom colored as coloring is time-consuming and labor-intensive. Digital colorization techniques may provide a convenient way to colorize the mangas. Unfortunately, the intensive usage of strokes, hatching, halftoning and screening in mangas not just enriches the visual content and creates dramatic atmosphere, but also imposes many difficulties to digital colorization [1]. We have devised a step-wise algorithmic approach to colorize mangas in an easy and efficient way.

2. ALGORITHM

The algorithm consists of three major steps:

1. Taking a user scribble on input image.
2. Segmenting the image.
3. Colorizing the segmented region.

Doing this incrementally, we can colorize different regions of image.

2.1. Taking user scribble

The user is presented with a simple, easy-to-navigate GUI (Figure 1) to scribble the image with desired brush color and size. Then the user can tweak the parameters of the algorithm as per the needs, or leave it to the defaults.

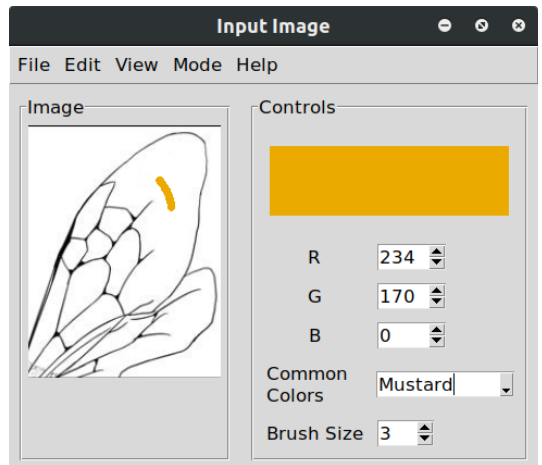


Fig. 1. Interface to take user scribble

2.2. Segmenting the image

We have used level set propagation method for segmenting the image. The level set methods tracks the evolution of a surface Φ that is moving normal to the boundary of the evolving curve with speed F. This speed function governs the actual boundary of the segmented region along with the stopping criteria of the level set evolution [1]-[3]. The level set equation where t is the time of evolution, is given by

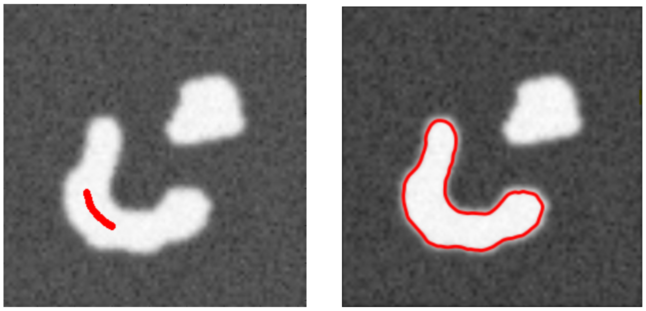
$$\frac{\partial \Phi}{\partial t} = -F|\nabla \Phi| \qquad [1]$$

For segmentation problem, we start by initializing Φ with signed distance from a point on user scribble. We choose $F = F_A + F_G$, where F_A is the ballooning force controlling the speed of the propagation and $F_G = -\epsilon\kappa$, where κ is the curvature of the evolving curve and ϵ is a constant [1]-[3]. F_G helps the curve to converge. Rewriting the level set equation

$$\frac{\partial \Phi}{\partial t} = -h \cdot (F_A + F_G)|\nabla \Phi| \qquad [1]$$

Here h is the halting term that helps the curve evolution to terminate. This term depends upon the type of the curve evolution chosen by the user.

2.2.1. *Intensity-Continuous Region*



(a) Input Image [3] (b) Segmented region

Fig. 2. Image segmentation of intensity-continuous region.

The halting term for intensity continuous region is given by:

$$h_I(x, y) = \frac{1}{1 + |\nabla(G_\sigma \circledast I(x, y))|^2}$$

where $G_\sigma \circledast I(x, y)$ denotes the convolution of image I with low-pass Gaussian filter G_σ. h_I keep track of change in intensity gradient, this means that the expression $|\nabla(G_\sigma \circledast I(x, y))|$ is zero except at the places where the gradient changes abruptly, like at the boundary edges [1].

Leak-Proofing When using the conventional flood-filling technique, there will be leaking (Figure 3(b)) due to the unclosed boundaries commonly found in manga. In order to leak-proof the segmented regions, we introduce one more component, F_I, into the speed function F and rewrite $F = h_I(F_A + F_I + F_G)$, and F_I is defined as [1]

$$F_I(x, y) = -F_A R\left(\frac{|\nabla G_\sigma \circledast I(x, y)| - M_2}{M_1 - M_2 - \delta}\right) \qquad [1]$$

where M_1 and M_2 are the maximum and minimum values of the magnitude of image gradient $|\nabla(G_\sigma \circledast I(x, y))|$; parameter $\delta \in [0, M_1 - M_2]$ is the relaxation factor; function R clamps the input value to $[0, 1]$. [1]

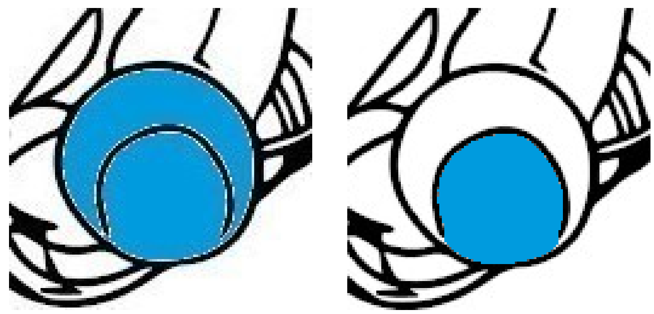
2.2.2. *Pattern-Continuous Region*

A userful choice for obtaing pattern features is Gabor wavelet filter $g_{m,n}$, where m is the orientation index and n is the scale index. We obtain a local texture pattern feature at (u, v) by convolving a window of size $w \times w$ centered at (u, v) with the Gabor wavelet filter $g_{m,n}$. The response is given by

$$W_{m,n}(u, v) = I(u, v) \circledast g_{m,n} \qquad [1]$$

Then, we compute the mean and standard deviation of the magnitude of the response. We create a filter bank containing Gabor wavelet filters in several orientations and scales to

(a) Input Image [7]



(b) Flood-Filling (c) Leak-Proofing

Fig. 3. Comparison of flood fill algorithm with level set method with leak proofing.

make the pattern feature extraction process scale and rotation invariant.

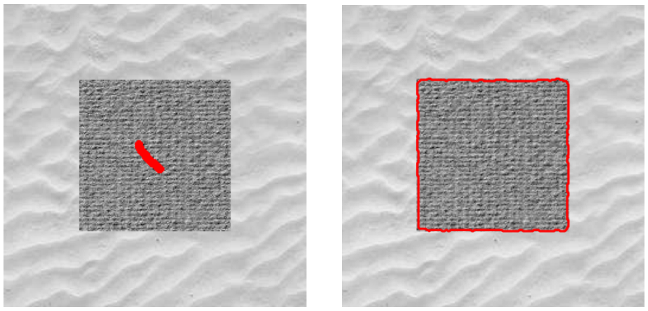
$$\mu_{m,n} = \frac{1}{w^2} \cdot \sum |W_{m,n}(u, v)| \qquad [1]$$

$$\sigma_{m,n} = \sqrt{\frac{1}{w^2} \cdot \sum (W_{m,n}(u, v) - \mu_{m,n})^2} \qquad [1]$$

For m scales and n orientations, a feature vector T is represented as:

$$T = [\mu_{0,0}\ \sigma_{0,0} \ldots \mu_{m,n}\ \sigma_{m,n}] \qquad [1]$$

Then, we perform clustering on the feature vector set of the image in order to segregate out the required region. In our application, we have used the K-Means clustering algorithm which considers N clusters, and the number of clusters N can be set as per requirements using the GUI. Finally we construct the halting term $h_P(x, y)$ for pattern-continuous region, which is equal to one on the cluster that falls upon the user scribble and zero elsewhere.



(a) Input Image [7] (b) Segmented region

Fig. 4. Image segmentation of pattern-continuous region.

2.3. Colorization

After segmenting the image, we present user with the option of performing three types of colorization:

1. Color Replacement
2. Pattern to shading
3. Stroke-Preserving Colorization

2.3.1. Color Replacement

For intensity continuous regions, we simply give and option to fill the segmented regions with the user scribble color.

2.3.2. Pattern to shading

The artists commonly use the change of density in hatching and screening to recreate shading. Such a technique is usually found at the background of the drawing. As color can readily reproduce such shading effect, we can convert the pattern to smooth color shading [1]. We first calculate the local intensity within the pixel neighborhood,

$$s = f \circledast Y_{image} \qquad [1]$$

where Y_{image} is the pixel gray value in the original image in YUV color space and f is a box filter. Note that $s \in [0, 1]$. Then the new color in YUV color space is obtained as [1]

$$Y_{new} = sY_{user} \qquad [1]$$
$$(U, V)_{new} = (U, V)_{user} \qquad [1]$$

2.3.3. Stroke Preserving Colorization

As mentioned before, the artist may use hatching/screening to express material reflectances, textures, or even shapes, it is sometimes necessary to preserve the original pattern during colorization. Instead of naively replacing the whole region with a single color, it is colorized by bleeding colors out of the strokes/patterns [1]. For applying the colorization, we compute the following term:

$$h(x, y) = \frac{1}{1 + |\nabla(G_\sigma \circledast I(x, y))|}$$

Using this term, we compute k:

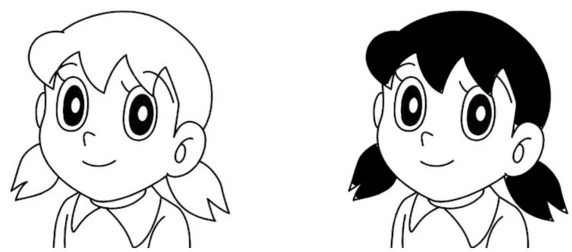
$$k = \mathtt{norm}((1 - h)^2)$$

Where `norm` normalizes the values to the range $[0, 1]$. We use a parameter α which can be changed by the user to perform the colorization. The new color in YUV space is obtained as

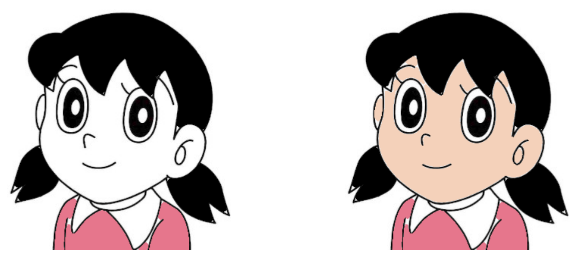
$$Y_{new} = Y_{user} \cdot k, \quad \text{where } k > \alpha$$
$$(U, V)_{new} = (U, V)_{user}$$

3. RESULTS

Figures 5, 6 depict colorization results using Color Replacement method. Figures 7, 8, 9 depict colorization results using Pattern to Shading method. Figure 10 depict colorization result using Stroke-Preserving Colorization method.



(a) Uncolored Image [7] (b) Intermediate Image 1

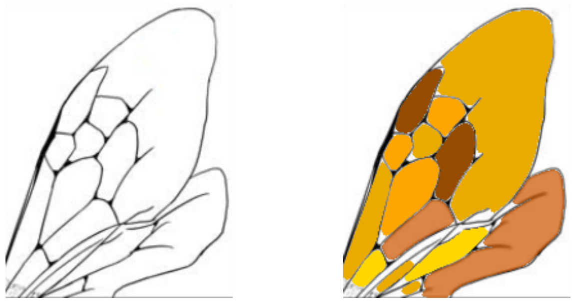


(c) Intermediate Image 2 (c) Final Colored Image

Fig. 5. Incremental colorization using Color Replacement method.

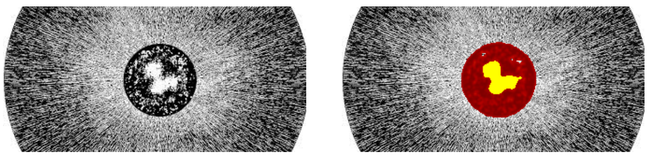
4. CONCLUSION

The algorithm propagates colors over pattern-continuous regions containing hand-drawn hatching and printed screening patterns. The user is free from manually segmenting the patterned regions. Both pattern-continuous and intensity-continuous regions can be segmented under the same mathematical framework.

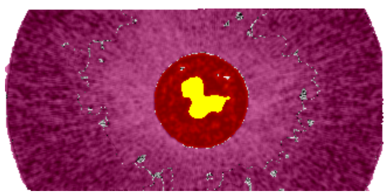


(a) Uncolored Image [1] (c) Final Colored Image

Fig. 6. Color Replacement.



(a) Uncolored Image [1] (c) Intermediate Image 1



(c) Final Colored Image

Fig. 7. Incremental colorization using Pattern to Shading method.

5. REFERENCES

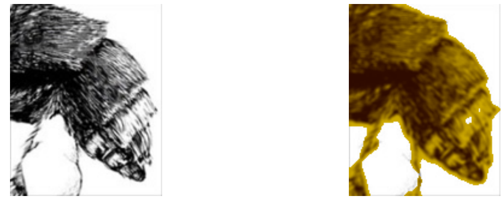
[1] Qu, Y., Wong, T.T. and Heng, P.A., ”Manga colorization”, ACM Trans. on Graphics (TOG), 25(3), pp.1214-1220, 2006.

[2] A. Kristiadi, ”Level Set Method Part I: Introduction”, Novemver 5, 2016. Accessed on: November 15, 2020. [Online]. Available: https://wiseodd.github.io/techblog/2016/11/05/levelset-method/

[3] A. Kristiadi, ”Level Set Method Part II: Image Segmentation”, Novemver 20, 2016. Accessed on: November 15, 2020. [Online]. Available: https://wiseodd.github.io/techblog/2016/11/20/levelset-segmentation/

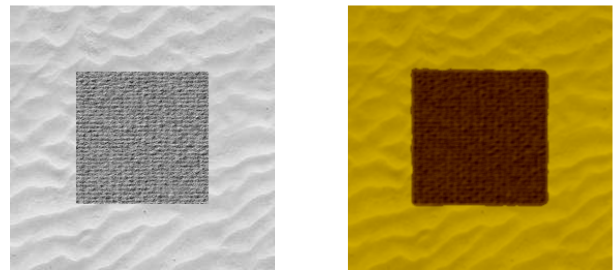
[4] P.O. Persson, ”The Level Set Method”, March 8, 2005. Accessed on: November 20, 2020. [Online]. Available: https://math.mit.edu/classes/18.086/2008/levelsetpres.pdf

[5] P.O. Persson, ”The Level Set Method”, March 7, 2005. Accessed on: November 20, 2020. [Online]. Available: https://math.mit.edu/classes/18.086/2008/levelsetnotes.pdf



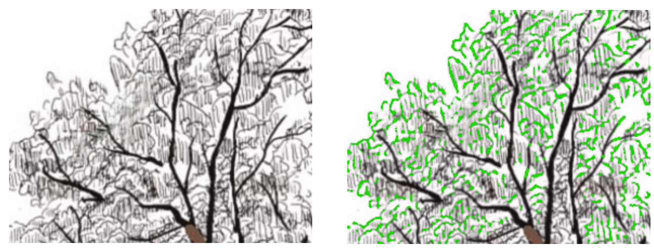
(a) Uncolored Image [1] (c) Colored Image

Fig. 8. Pattern to Shading

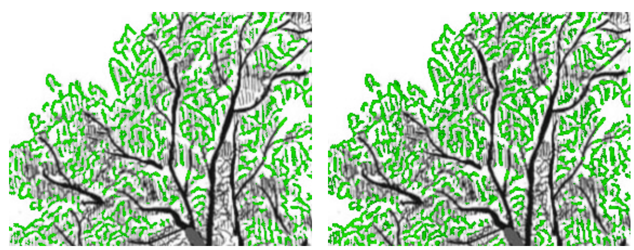


(a) Uncolored Image [7] (c) Final Colored Image

Fig. 9. Pattern to Shading



(a) Uncolored Image [1] (c) $\alpha = 0.8$



(c) $\alpha = 0.5$ (c) $\alpha = 0.2$

Fig. 10. Stroke-Preserving Colorization with different values of α

[6] B. Seibold, ”MIT18086 LEVELSET FRONT ”, March 2008. Accessed on: November 20, 2020. [Online]. Available: http://math.mit.edu/ gs/cse/codes/mit18086_levelset_front.m

[7] Google, Google Images. [Online]. Available: images.google.com