Color-to-Grayscale Image Conversion Algorithms

Yiming Lin
Department of Computer Science
Portland State University
Portland, OR 97201
yl6@pdx.edu

June 6, 2021

Abstract

A study of two color-to-grayscale image conversion algorithms proposed in 2005 (salience-preserving color removal) and 2009 (non-linear global mapping). This paper reports the results of two algorithms' implementation, compares the results between two algorithms.

Figure 1: Left: the original color image; Middle: Photoshop's grayscale mode; Right: output from our algorithm implementation

1 Introduction

Color-to-grayscale image conversion has a very board application in the real world. For example, black-and-white printer, non-photorealistic rendering in black-and-white, image preprocessing for deep learning, and conversion due to artistic reasons all involves the use of color-to-grayscale image conversion. However, the conversion algorithms that have been used in real-world product Photoshop result in a feature-loss image as shown in Figure 1 middle. Thus, new algorithms [1] and [2] were proposed to preserve visually important image features during conversion. We studied two algorithms and implemented them

in Python 3.8.3 64-bit using OpenCV 4.0.1, the resulting conversion from our implementation is shown in Figure 1 right, which successfully preserved missing details in the middle.

2 Algorithms

2.1 Non-linear Global Mapping Algorithm

The algorithm proposed in [2] ensures that (1) same colors have the same grayscale value; (2) features in color image are discriminable in grayscale image; (3) colors have a reasonable ordering in grayscale values; (4) similar lightness stimuli between color and grayscale image.

2.1.1 Mapping Consistency

To map a color image to a grayscale image, the mapping function is in nonlinear functional form, where L, θ, and C are in CIE LCH color space.

$$g(x, y) = L + f(\theta)C \tag{1}$$

Authors of [2] modeled $f(\theta)$ by trigonometric polynomial function, the following real trigonometric polynomial of degree N is used in our implementation :

$$f(x) = a_0 + \sum_{n=1}^{N} a_n cos(nx) + \sum_{n=1}^{N} b_n sin(nx) \tag{2}$$

where a_n and b_n is unknown parameters to find in optimization step. Thus, the problem of finding a good mapping function is equivalent to finding a_n and b_n that minimize objective function which will be discussed later.

2.1.2 Feature Preservation

To preserve feature discriminability, [2] states that the local difference in the color image should be reproduced similarly in the grayscale image. Thus, to achieve this we minimize the difference between color and grayscale image gradients as following:

$$E_s = \sum_{(x,y)\in I} \|\nabla g - \boldsymbol{G}(x, y)\|^2 \tag{3}$$

where ∇g is the gradient of grayscale image, and $\boldsymbol{G}$ is the color difference of pixel color. Equation (3) also defines the objective function to minimize in this algorithm.

2.1.3 Ordering Preservation

When minimizing the objective function equation (3), we need to define the order of two color in 1D measurement and define the sign of measurement to

assign a color ordering. Firstly, the color difference at (x, y) is calculated by

$$\boldsymbol{G}(x,y) = \begin{pmatrix} G^x(x,y) \\ G^y(x,y) \end{pmatrix}^T = \begin{pmatrix} \boldsymbol{c}(x+1,y) \ominus \boldsymbol{c}(x-1,y) \\ \boldsymbol{c}(x,y+1) \ominus \boldsymbol{c}(x,y-1) \end{pmatrix}^T \tag{4}$$

where the color difference operator $\ominus$ is defined by

$$\boldsymbol{c}_i \ominus \boldsymbol{c}_j = sign(\boldsymbol{c}_i, \boldsymbol{c}_j) \sqrt{\Delta L_{ij}^2 + \left(\alpha + \frac{\sqrt{\Delta a_{ij}^2 + \Delta b_{ij}^2}}{R} \right)^2} \tag{5}$$

where ΔL_{ij},Δa_{ij}, Δb_{ij} are difference of $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ in L, a, b space of CIE Lab color system, respectively. Thus, $\sqrt{\Delta L_{ij}^2 + \left(\alpha + \frac{\sqrt{\Delta a_{ij}^2 + \Delta b_{ij}^2}}{R} \right)^2}$ calculates the distance between two Lab colors, next we define the *sign* term so as to assign two colors a order.

In [3] it presents a method to estimate Helmholtz Kohlrausch (HK) effect, which models the influence of chromatic components on the perceptual lightness of colors. Here we use HK effect predictor proposed in [3] to order two colors by calculating ΔL_{ij}^{HK}; if ΔL_{ij}^{HK} is zero, then we consider ΔL; if ΔL is zero, we use the sign of $\Delta L^3 + \Delta a^3 + \Delta b^3$.

There are two models proposed in [3], this algorithm uses VAC model. The estimated HK effect Γ is calculated by

$$\Gamma_{VAC} = \frac{L_{eq}}{L} \tag{6}$$

$$= 1 + \left[-0.8660 q(\theta) + 0.0872 K_{Br} \right] S_{uv}(x,y) \tag{7}$$

where θ is defined by

$$\theta = \tan^{-1} \frac{v' - v'_c}{u' - u'_c} \tag{8}$$

$$v' = \frac{v}{13L} + v'_c \tag{9}$$

$$u' = \frac{u}{13L} + u'_c \tag{10}$$

where L, u, v are in CIE CUV color space, u'_c and v'_c are the u' and v' chromaticity of the reference white light; and q is defined by

$$\begin{aligned} q(\theta) = & -0.01585 \\ & -0.03017 \cos\theta - 0.04556 \cos 2\theta \\ & -0.02667 \cos 3\theta - 0.00295 \cos 4\theta \\ & +0.14592 \sin\theta + 0.05084 \sin 2\theta \\ & -0.01900 \sin 3\theta - 0.00764 \sin 4\theta \end{aligned} \tag{11}$$

and K_{Br} and S_{uv} are defined by

$$K_{Br} = 0.2717 \times \frac{6.469 + 6.362L_a^{0.4495}}{6.469 + L_a^{0.4495}} \tag{12}$$

$$S_{uv}(x, y) = 13[(u' - u'_c)^2 + (v' - v'_c)^2]^{1/2} \tag{13}$$

where $L_a = 63.66cd/m^2$ which is adapting luminance.

Combine equation (6) - (13), we obtain the difference of two colors' lightness estimation

$$\Delta L_{ij}^{HK} = (\Gamma_{VAC})_i L_i - (\Gamma_{VAC})_j L_j \tag{14}$$

2.1.4 Lightness Fidelity

Equation (1) is suitable for ensuring similarity of lightness stimuli between color and grayscale image. When chroma C is zero, a given color is gray then g returns the value of lightness; otherwise, the value of g is dominated by L term since f is very small (the resulting a_n and b_n in optimization step are usually very small), and therefore assign C with a very light weight, so g is dominated by L and so do not have abrupt changes of lightness.

2.1.5 Optimization

Rewrite equation (2) in vector form as

$$\begin{aligned} f(\theta) &= \boldsymbol{t}^T \boldsymbol{x} \\ \boldsymbol{t} &= [\cos\theta, \cos 2\theta, ..., \cos n\theta, \sin\theta, ..., \sin n\theta, 1]^T \\ \boldsymbol{x} &= [A_1, A_2, ..., A_n, B_1, ..., B_n, A_0]^T \end{aligned} \tag{15}$$

Reduce equation (1) to a quadratic equation as

$$\begin{aligned} E_s &= \boldsymbol{x}^T \boldsymbol{M}_s \boldsymbol{x} - 2\boldsymbol{b}_s^T \boldsymbol{x} + C \\ \boldsymbol{M_s} &= \sum_{(x,y)\in I} \left(\boldsymbol{u}\boldsymbol{u}^T + \boldsymbol{v}\boldsymbol{v}^T\right) \\ \boldsymbol{b_s} &= \sum_{(x,y)\in I} (p\boldsymbol{u} + q\boldsymbol{v}) \end{aligned} \tag{16}$$

where $\boldsymbol{u}, \boldsymbol{v}, p, q$ are calculated by

$$\begin{aligned} \boldsymbol{u} &= (C \cdot \boldsymbol{t})_x \\ \boldsymbol{v} &= (C \cdot \boldsymbol{t})_y \\ p &= G^x - L_x \\ q &= G^y - L_y \end{aligned} \tag{17}$$

where subscript x, y denote partial derivatives with respect to x and y directions, respectively, which are obtained by using central difference.

E_s is minimized at $\boldsymbol{x} = \boldsymbol{M}_s^{-1}\boldsymbol{b}_s$, but $\boldsymbol{x}$ can be very large, which results in g to be out of visible range. As stated in section 2.1.4, we want to keep f to be small, so we add another term $E_r = \boldsymbol{x}^T I \boldsymbol{x}$ with a parameter λ, so the energy function becomes

$$E_{image} = E_s + \lambda E_r \tag{18}$$

then the energy function is minimized at

$$\boldsymbol{x} = (\boldsymbol{M}_s + \lambda I)^{-1}\boldsymbol{b}_s \tag{19}$$

In our implementation, λ is a user specified parameter, default is the number of pixels. An appropriate value shall keep resulting $\boldsymbol{x}$ to be small.

2.2 Salience-Persevering Color Removal

In [1] the core idea of the algorithm is to define a scalar color differences δ_{ij} betwen two pixels i, j in the color image, then we find two grayscale pixel values g_i and g_j such that we can minimize difference between $(g_i - g_j)$ and δ_{ij}. Thus, the objective function to be minimized is defined as following:

$$f(g) = \sum_{(i,j)\in K} \left((g_i - g_j) - \delta_{ij}\right)^2 \tag{20}$$

where K is the set of all pairs of pixels to be compared. Thus, we need to firstly define δ_{ij} called target differences.

2.2.1 Target Difference

Target difference δ_{ij} considers both luminance difference ΔL_{ij} and chrominance difference $(\Delta A_{ij}, \Delta B_{ij})$ in CIE Lab color system. Since ΔL_{ij} is a scalar value, $(\Delta A_{ij}, \Delta B_{ij})$ is a vector value, we calculate its L2 norm to map to one dimension and then assign a sign, otherwise it is always positive.

We add user parameter θ such that the sign can be calculated by

$$(\Delta A_{ij}, \Delta B_{ij}) \cdot (\cos\theta, \sin\theta) \tag{21}$$

Target difference is set to either luminance difference or chrominance difference, which is formally defined by

$$\delta(\alpha, \theta)_{ij} = \begin{cases} \Delta L_{ij} & |\Delta L_{ij}| > crunch(\|\Delta\vec{C}_{ij}\|) \\ crunch(\|\Delta\vec{C}_{ij}\|) & \Delta\vec{C}_{ij} \cdot (\cos\theta, \sin\theta) \geq 0 \\ crunch(-\|\Delta\vec{C}_{ij}\|) & \text{otherwise} \end{cases} \tag{22}$$

where *crunch* is defined by

$$crunch(x) = \alpha \times \tanh\frac{x}{\alpha} \tag{23}$$

where α is a user specified parameter. α is larger, then the target difference emphasis more on chrominance difference.

2.2.2 Optimization

In equation (20), the set of pairs of pixels to be compared is determined by user specified parameter μ which set the size of neighborhood pixels of a pixel. Thus, we have

$$K = \{(i, j) : \forall i \in I, \forall j \in N(i)\} \tag{24}$$

where $N(i)$ denotes the set of neighborhood pixels of pixel i, I denotes the set of pixel positions in input color image.

Notice that equation (20) is a convex function, thus the minimum occurs at $\nabla f = 0$. Then we have

$$\begin{aligned} \forall i \in I, \frac{\partial f}{\partial g_i} &= \sum_{(i,j):j \in N(i)} (g_i - g_j - \delta_{ij}) - \sum_{(k,i):k \in N(i)} (g_k - g_i - \delta_{ki}) \\ &= 2Ng_i - 2\sum_{j \in N(i)} g_j - \sum_{j \in N(i)} (\delta_{ji} - \delta_{ij}) \\ &= 0 \end{aligned} \tag{25}$$

Rewrite equation (25) we have

$$2Ng_i - 2\sum_{j \in N(i)} g_j = \sum_{j \in N(i)} (\delta_{ji} - \delta_{ij}) \tag{26}$$

Then transform equation (26) into matrix form of $\boldsymbol{Ax} = \boldsymbol{b}$ such that

$$\begin{aligned} \boldsymbol{A}_{ij} &= 2N, \text{where } i = j \\ \boldsymbol{A}_{ij} &= -2, \text{where } j \in N(i) \\ \boldsymbol{A}_{ij} &= 0, \text{where } j \notin N(i) \\ \boldsymbol{b}_i &= \sum_{j \in N(i)} (\delta_{ji} - \delta_{ij}) \end{aligned} \tag{27}$$

Feed equation (27) into a linear system solver using conjugate gradient method, initial value is assigned by L space of Lab color system.

3 Results and Discussion

3.1 Results of Non-linear Global Mapping

Table 1 shows the results from nonlinear global mapping algorithms, the left column is the original color images, the middle column is the images from Photoshop grayscale mode, and the right column is the output of our implementation.

In the first row, the circles in the color image have isoluminant colors, and Photoshop's grayscale mode makes them disappear, but they are preserved in our output. In the second and third row, the green part in the map and the sun and its reflection in the water disappear in Photoshop, but they are preserved

in our output. In the fourth row, Photoshop's output preserved features in the original image, but our output makes red wall and green wall more discriminable. The fifth row shows that our implementation maps same color to the same grayscale value. And in the last row, the house is more discriminable in our implementation than Photoshop's output.

For all six rows, our outputs successfully preserved features that are failed to preserve in Photoshop, our outputs also exhibit a robust conversion from color image to grayscale image, since it is very obvious that all the visually important image features.

3.2 Results of Salience-Persevering Color Removal

Table 2 shows the results from salience-persevering color removal algorithms, the left column is the original color images, the middle column is the global mapping algorithm of our implementation, and the right column is the output of alience-persevering color removal of our implementation.

Since the original color images in all five rows in table 2 are identical with table 1 which has already shown output from Photoshop, this table compare outputs between two algorithm we implemented. It is obvious that all visually important image features are preserved, but some results are not very good when compared to nonlinear global mapping algorithm: in the first row, the circles are mapped to two different grayscale values in the right, but in the middle they are mapped to almost unique grayscale value for each circle; in the last row, the leftmost part of the house is too dark in the right, but the leftmost part of the house has a gradual transition to a lighter value compared to right part of the house, which is more visually consistent with the original color image; also in the third row, the edge of the sun is more clear in the middle.

3.2.1 Effectiveness of user parameter θ

Figure 2 shows the effectiveness of user parameter θ from the outputs of our implementation. Notice that for any two θ values that is different with 180°, the darker circle becomes lighter circle, the lighter circle becomes darker circle, which is expected since $\cos\theta$ exactly corresponds to its negation of $\cos\theta + 180^\circ$, which exactly reverse the order of color difference.

3.2.2 Effectiveness of user parameter α

Figure 3 shows the effectiveness of user parameter α from the outputs of our implementation. It shows that α is larger, then the resulting grayscale image appears more contrasting.

3.2.3 Effectiveness of user parameter μ

Figure 4 shows the effectiveness of user parameter μ from the outputs of our implementation. It shows that when μ is very small, the resulting image cannot

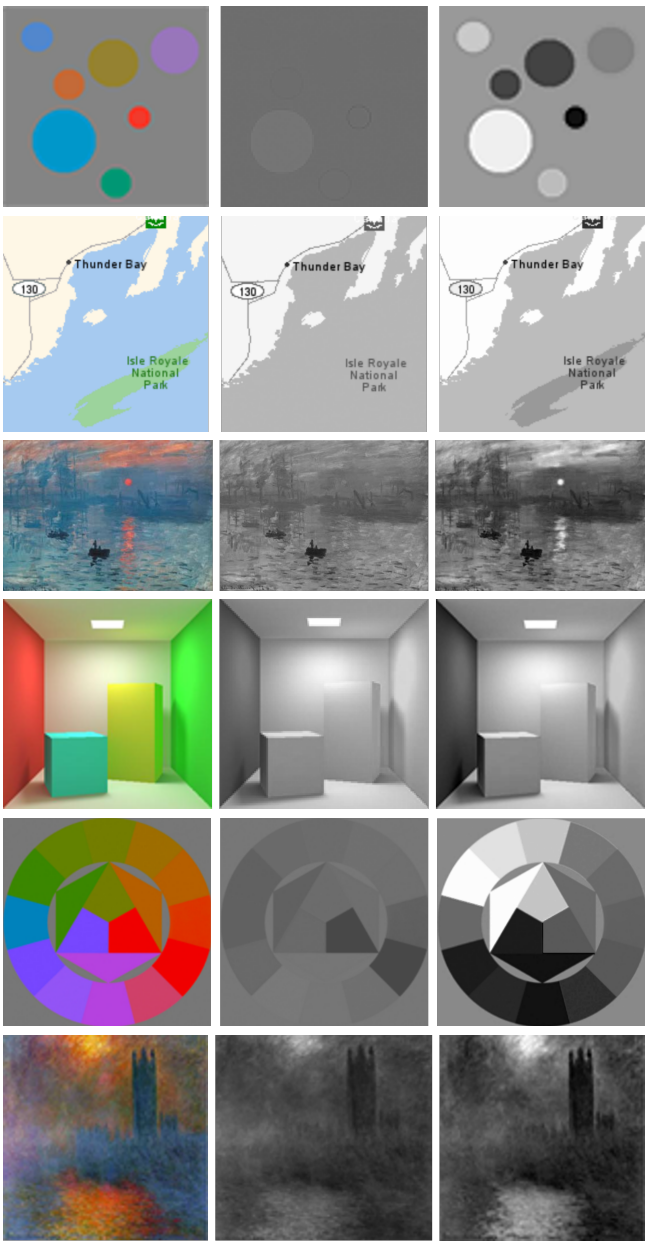

Table 1: Comparison between original color image (left column), Photoshop grayscale mode (middle column), and output from our implementation (right column)

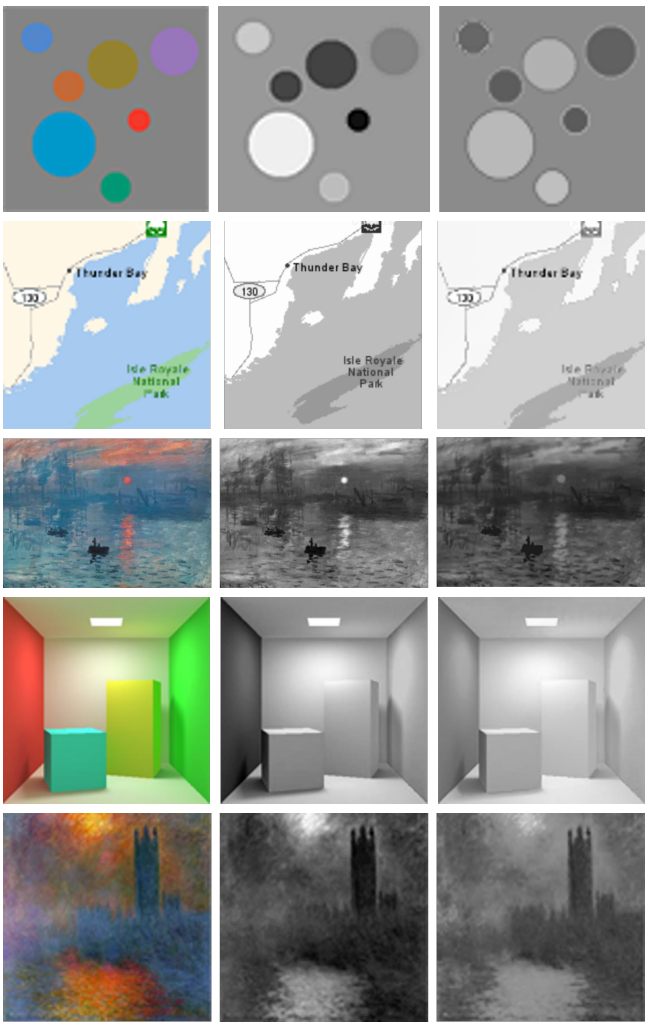

Table 2: Comparison between original color image (left column), global mapping algorithm of our implementation (middle column), and salience-persevering color removal of our implementation (right column)

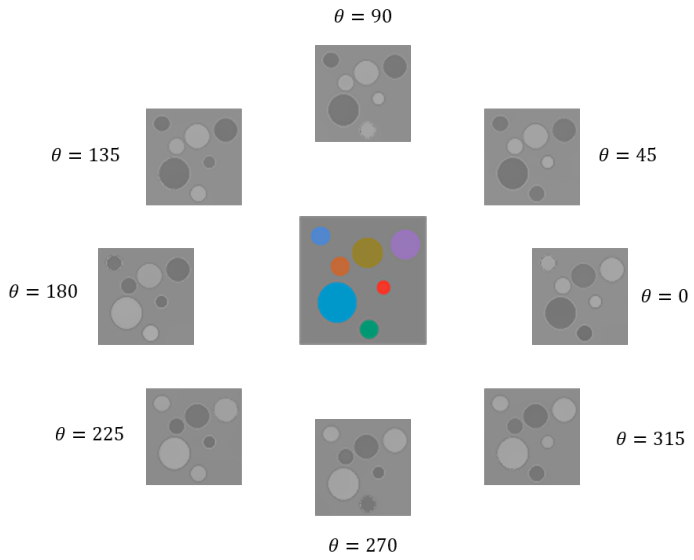

Figure 2: Result with different value of θ

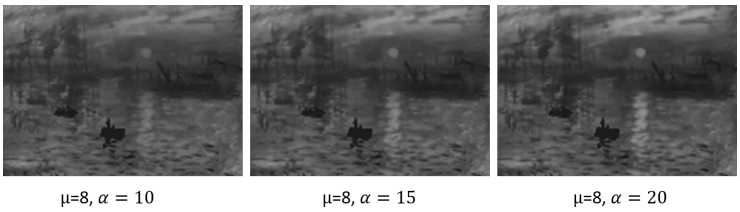

Figure 3: Result with different value of α

preserve features in the color image, better results are obtained when μ is large enough.

4 Conclusion

This paper studied two algorithms of conversion from color image to grayscale image, reported some results from our implementation, and compared results between two different algorithms and Photoshop's grayscale mode output. Both algorithms successfully preserved salient features in original color images in the range of reported images.

In term of algorithm's run-time complexity, salience-preserving color removal algorithm compares all pairs of pixels in the input image (when neighborhood pixels are entire image as recommended in [1]), which results in a $O(N^4)$ algorithm

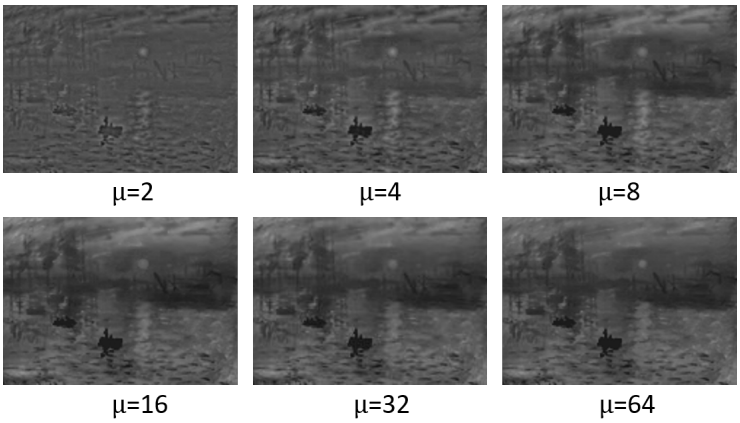

Figure 4: Result with different value of μ

where N is the width or height of the image. For a 100×100 image with $\mu = 64$ it takes around 4 minutes to finished. In contrast, nonlinear global mapping algorithm tries to find a good mapping function, where the number of unknown parameters when $N = 4$ (a good enough size, bigger size does not improves significant output quality) is 9, so the linear system solver can immediately return the results. Also, the algorithm relies on calculating central differences, thus simply traversing each pixel a fix number of times is enough, which results in a $O(N^2)$ algorithm here N is the width or height of the image. A 500×400 input image takes around 20 seconds to run. Thus, it significantly improves the run-time efficiency.

References

[1] Amy A. Gooch, Sven C. Olsen, Jack Tumblin, and Bruce Gooch. Color2gray: Salience-preserving color removal. *ACM Transactions on Graphics*, 24(3):634–639, July 2005. ACM SIGGRAPH 2005 ; Conference date: 31-07-2005 Through 04-08-2005.

[2] Yongjin Kim, Cheolhun Jang, Julien Demouth, and Seungyong Lee. Robust color-to-gray via nonlinear global mapping. *ACM Trans. Graph.*, 28(5):1–4, December 2009.

[3] Yoshinobu Nayatani. Simple estimation methods for the helmholtz—kohlrausch effect. *Color Research & Application*, 22(6):385–401, 1997.