21st International Conference on Pattern Recognition (ICPR 2012)
November 11-15, 2012. Tsukuba, Japan

Color-Line Vector Field and Local Color Component Decomposition for Smoothing and Denoising of Color Images

Keiichiro Shirai* *Shinshu Univ.*
Masahiro Okuda* *Univ. of Kitakyushu*
Masaaki Ikehara *Keio Univ.*

Abstract

Most conventional smoothing and denoising methods for color images deal with each color channel independently, which results in discolorations due to unbalancing the relation between the color components. In this paper, we propose a smoothing algorithm to reduce discolorations based on "color-lines". Our iterative algorithm consists of a local color decomposition step by color-line vectors and an iterative filtering step. Our numerical simulation shows that the method improves image quality with less discolorations while keeping the smoothing capability.

1. Introduction

When filtering the color image for the sake of smoothing or denoising, generally the RGB values are converted to a space such as YCbCr and L*a*b* so as to separate the luma and chroma components, then the luma component, to which the human perception is more sensitive, is processed mainly. However, since these color conversions are done pixel by pixel, they do not take into account the correlation among neighboring pixel colors. Thereby, when filtering color images, the color correlations often become unbalanced, which results in color heterogeneity over a whole image region.

Recently as an efficient feature that represents the correlation in the color range domain, the "color-line" (*cl*) feature [5], which is defined as a line that approximates the shape of color distribution in a local region (see Fig. 1), is often used in image processing. This phenomenon appears because natural images contain intermediate colors on the boundary of two color regions and contain the luminance variations by lights and shadows.

Based on the *cl* feature, [9] and our previous work [7] remedy outliers located away from the line in order to reduce discoloration artifact. The guided filter [4] has a similar framework and has a capability to reduce discolorations.

*First authors with equal contribution.

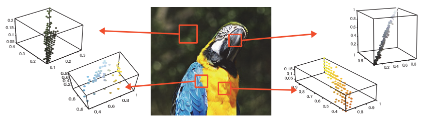


Figure 1. Color-line feature.

In this paper, we extend our previous work [7] and introduce a novel smoothing algorithm. Our method improves image quality with less discolorations while keeping the smoothing capability.

2. Algorithm Overview

The flowchart of our algorithm is shown in Fig. 2. The algorithm mainly consists of two steps: a color decomposition step and a filtering step, and these steps are processed iteratively. The more detailed description of the procedure is given as follows:

1. At each pixel, we estimate local color distribution and set its principal axis to the *cl* vector by using principal component analysis (PCA) (described in Sec. 3).
2. Each pixel color is decomposed into (i) the *cl* component along the *cl* vector and (ii) residual component perpendicular to it (Sec. 4).
3. The decomposed color components are smoothed or denoised by an iterative filtering method (Sec. 5).
4. Finally restore the color by the inverse color transform, then turned back to the step 1.

In this way, the filtered colors vary along each *cl* axis and do not corrupt the shapes of color distributions. However, *cl* vectors obtained in the early stage are noisy, and noise reduction in the spatial domain is still needed to the *cl* vector itself. Therefore, we try to iteratively refine *cl* vectors by the above iterative process in the color range domain and the spatial domain.

3. Color-line vector field

The *cl* vector that is the axis of local color distribution is obtained by PCA, in which it is derived as the

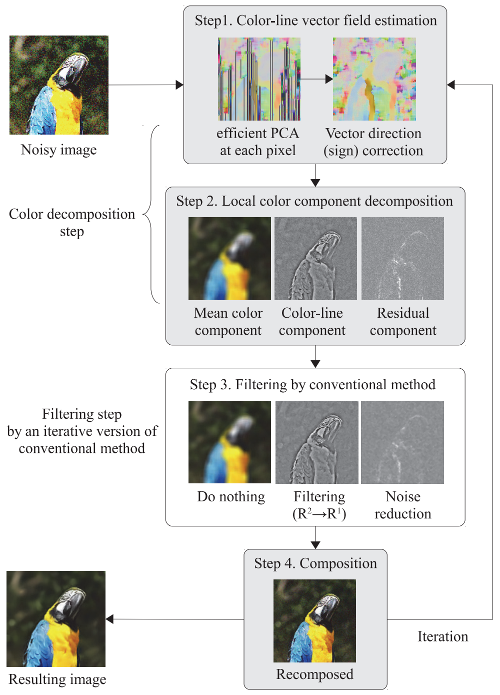

Step1. Color-line vector field estimation
efficient PCA at each pixel
Vector direction (sign) correction
Noisy image
Color decomposition step
Step 2. Local color component decomposition
Mean color component
Color-line component
Residual component
Step 3. Filtering by conventional method
Filtering step by an iterative version of conventional method
Do nothing
Filtering (R²→R¹)
Noise reduction
Step 4. Composition
Recomposed
Resulting image
Iteration

Figure 2. Algorithm flowchart.

eigenvector corresponding to the maximum eigenvalue. In this section, we describe the efficient calculation for PCA at each pixel (Sec. 3.1) and how to make the *cl* vector directions lined up to remove ambiguity of the vector direction (Sec. 3.2). The method prevents the vectors from irregular sign flips, which causes gaps among their neighboring vectors, and become the interference of the smoothing.

3.1. Calculation of color-line vectors by PCA

To find the largest eigenvalue d and the corresponding eigenvector $\mathbf{v}$, we use a traditional power iteration algorithm, instead of recent analytical methods because of the computational efficiency for the pixel-wise calculation. Moreover there is little perceptible difference between resulting images.

First a covariance matrix $\mathbf{C}_i$ of each pixel i needs to be computed by using pixel colors $\mathbf{I}_j = [r_j, g_j, b_j]^T$ of the neighboring pixels $j \in \mathcal{N}(i)$ and the mean color $\boldsymbol{\mu}_i = \frac{1}{w}\sum_{j\in\mathcal{N}(i)} \mathbf{I}_j$ as: $\mathbf{C}_i = (\frac{1}{w}\sum_{j\in\mathcal{N}(i)} \mathbf{I}_j^T \mathbf{I}_j) - \boldsymbol{\mu}_i^T \boldsymbol{\mu}_i$, where $w(=49)$ is the number of pixels in a 7×7 filter window. Then the power iteration is given as follows:

1. update eigenvector: $\tilde{\mathbf{v}}_i = \mathbf{C}_i \mathbf{v}_i^{(t)}$
2. update eigenvalue: $d_i^{(t+1)} = \|\tilde{\mathbf{v}}_i\|$
3. normalization: $\mathbf{v}_i^{(t+1)} = \tilde{\mathbf{v}}_i / d_i^{(t+1)}$,
4. convergence check: break if $|\mathbf{v}_i^{(t)T}\mathbf{v}_i^{(t+1)} - 1| < \tau$

where t is the number of iterations, the initial eigenvector is set by random values normalized to be $\|\mathbf{v}^{(0)}\| = 1$. As the angle tolerance for the convergence condition, we set $\tau = 0.0001$ in the experiment.

Since color pixels have correlation among its neighbors, their eigen-pairs also tend to be similar in its neighborhood. Thereby the number of iterations at a current pixel can be reduced by setting a well converged neighboring eigenvector to the initial eigenvector of a current pixel. We propagate a resulting eigenvector at a current pixel i to the next adjacent pixel $i+1$ as the initial eigenvector:

$$\mathbf{v}_{i+1}^{(0)} = \mathbf{v}_i^{(\text{converged})} \tag{1}$$

3.2. Direction alignment of color-line vectors

There is inherently an ambiguity in the sign of the eigenvectors $s_i v_i$, where $s_i(=+1 \text{ or } -1)$. In order to determine the sign of each vector $\mathbf{v}_i$ that minimizes the energy function among neighboring pixel pairs $\{i,j\}$: $\sum_{\{i,j\}} \|s_i\mathbf{v}_i - s_j\mathbf{v}_j\|^2$, we adopt the following Jacobian relaxation method [1]:

$$s_p^{(t+1)} = \text{sign}\Big(\sum_{q\in\mathcal{N}(p), q\neq p} (s_p^{(t)}\mathbf{v}_p)^T (s_q^{(t)}\mathbf{v}_q)\Big), \tag{2}$$

where $q \in \mathcal{N}(p)$ is 3×3 neighboring pixels of a pixel p. The sign s_p of the pixel p is determined so as to fit the dominant direction of neighboring vectors by using the inner products as the criterion. In practice, we calculate the summation in Eq. 2 by $\{(s_p^{(t)}\mathbf{v}_p)^T \sum_q (s_q^{(t)}\mathbf{v}_q)\} - 1$ with the use of box filtering for the acceleration.

The above relaxation method is only effective for pixel-wise flip. In order to flip signs over a large region, we use a multiresolution approach like the multigrid's V-cycle [1] as shown in Fig. 3, where the sign-aligned vectors $\{s_i\mathbf{v}_i\}$ are propagated to coarser resolutions, and then the resulting signs $\{s_i\}$ are propagated to finer resolutions.

For generating the multiresolution pyramid for vector and sign images in Fig. 3, we use the gaussian pyramid decomposition [2]. Additionally, in the decimation process, for the purpose of giving the priority to pixels around edges which have large eigenvalues, we multiply the eigenvalue d_i as the weight for the pixel: $d_i s_i \mathbf{v}_i$, then apply the decimation filter and re-normalize the half-sized vector field. The multiresolution eigenvalue images are generated by the same approach of the gaussian pyramid.

4. Local color component decomposition

The color of each pixel $\mathbf{I}_i$ is locally decomposed into the *cl* component D_i along the *cl* axis, and the residual

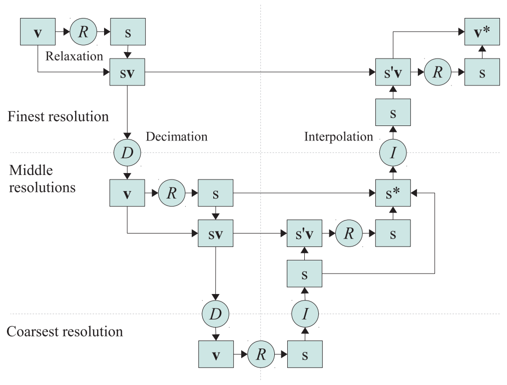
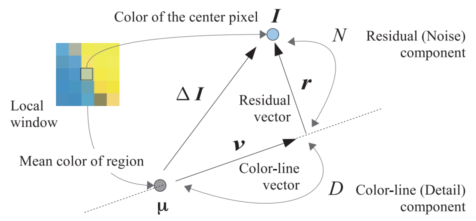


Figure 3. Vector direction alignment by multiresolutional relaxation method.



Color of the center pixel
I
N
Residual (Noise) component
Local window
ΔI
Residual vector
r
v
Mean color of region
Color-line vector
D
Color-line (Detail) component
μ

Figure 4. Color decomposition.

component N_i perpendicular to it, which corresponds to detail and noise components, respectively, as shown in Fig. 4.

The *cl* component is given as the inner product of the normalized *cl* vector $\mathbf{v}_i$ and the difference vector $\Delta\mathbf{I}_i = \mathbf{I}_i - \boldsymbol{\mu}_i$:

$$D_i = \mathbf{v}_i^T \Delta\mathbf{I}_i. \tag{3}$$

The residual component is obtained as the L_2 norm of the residual vector $\mathbf{r}_i = (\Delta\mathbf{I}_i - D_i\mathbf{v}_i)$:

$$N_i = \|\mathbf{r}_i\|. \tag{4}$$

The composition is done by the following equation after filtering is applied to two color components $\bar{D}_i$ and $\bar{N}_i$:

$$\mathbf{I}_i = \boldsymbol{\mu}_i + \bar{D}_i\mathbf{v}_i + \bar{N}_i\frac{\mathbf{r}_i}{\|\mathbf{r}_i\|}. \tag{5}$$

5. Filtering and noise reduction

Filtering for *cl* component. For filtering the *cl* component given as a gray scale image, we adopt a conventional iterative filer. Since the smoothing capability of non-iterative filter such as conventional linear low-pass filters or median filters is too strong, the iterative version such as anisotropic diffusion is more appropriate in our framework. The ”iterative version” iteratively solves an original filter equation, such as the diffusion equation for the anisotropic diffusion, which is an iterative relaxation method corresponding to the gaussian filtering based on direct convolution method. In brief, we smooth or denoise an image iteratively until one can obtain desired filtering effect. In the experiment of this paper, we use an anisotropic diffusion [8] as the iterative version of the bilateral smoothing filter, and use powerless BM3D [3] as the iterative version of the Color BM3D denoising filter [3].

Noise reduction for residual component. The residual component is also given as a gray scale and it is filtered with our noise reduction filter. To reduce noises with small intensities , we first apply Geman McClure robust function $w(x) = x^2/(\kappa + x^2)$, and add the weight as $\hat{N}_i = w(N_i)N_i$ (we set $\kappa = 0.02^2$). On the other hand, to remain low frequency components with large intensities, we use the 3×3 median filter for smoothing them.

6. Experimental results

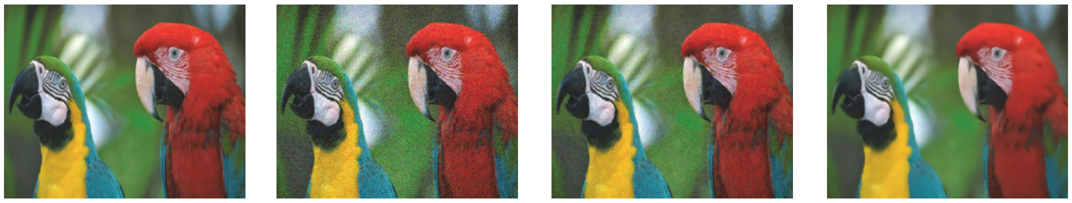
The results of our method are shown in Fig. 5. For comparison, we use Tschumperlé’s anisotropic diffusion [8] and Color BM3D [3] as applications of the smoothing and the denoising respectively. These images illustrate, (a) original image, (b) noisy image with additional gaussian noise, (c) result of conventional method, and (d) our iterative method with 4 iterations. In addition, PSNRs are shown below the images for the numerical evaluation , also ΔEs [6] are shown as a perceptual criteria (the smaller, the better). The parameters of the conventional methods are adjusted so as to give the best evaluation values, while in our method, we set parameters of the conventional methods so as to give the similar evaluations.

From the results of the anisotropic diffusion, one can see our method is able to reduce the discoloration artifacts more and can confirm the validity of the method in the both appearance and numerical evaluations. While from the results of BM3D, the differences at a glance are quit little, and our numerical evaluations are slightly inferior. However, the original BM3D tends to produce discolorations especially in smoothly varying gradation as one can see in the 4-th row of Fig. 5. Our method significantly reduces discolorations and improves perceptual appearance. Although the difference of evaluation values comes from the remaining noises around edges in our method, human perception is insensitive to these noises.

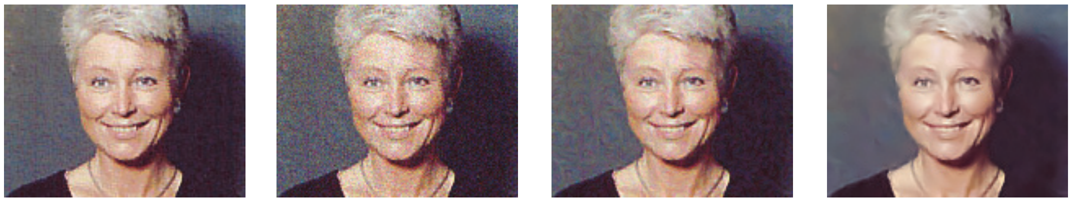
7. Conclusion

In this paper, we present a discoloration reduction algorithm with the color-line consideration. Using our

Smoothing by an anisotropic diffusion [8] and with our algorithm

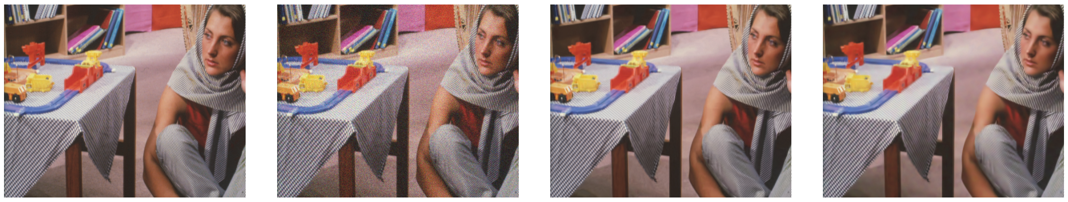


(a) Original (PSNR / ΔE) (b) Noisy (24.45 / 8.30) (c) Diffusion (30.18 / 4.22) (d) Ours (**32.86 / 2.20**)

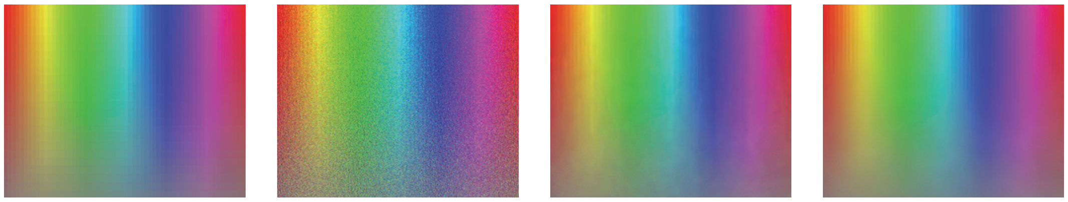


(a) Original (PSNR / ΔE) (b) Noisy (24.40 / 10.69) (c) Diffusion (28.97 / 6.32) (d) Ours (**29.53 / 4.65**)

Denoising by the Color BM3D [3] and the BM3D with our algorithm



(a) Original (PSNR / ΔE) (b) Noisy (24.43 / 10.16) (c) CBM3D (**34.07 / 2.26**) (d) Ours (32.38 / 2.82)



(a) Original (PSNR / ΔE) (b) Noisy (24.42 / 6.41) (c) CBM3D (39.92 / 1.04) (d) Ours (**40.41 / 0.91**)

Figure 5. Comparison of smoothing and denoising performance of conventional methods [8, 3].

algorithm in combination with conventional methods, the perceptual looking is improved. As for the algorithm, the efficient PCA and the vector direction alignment method are mainly described, as the improvement from our previous work.

References

[1] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A multigrid tutorial second edition*. SIAM, 2000.

[2] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. *IEEE Trans. on Comm.*, 31(4):532–540, 1983.

[3] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Color image denoising via sparse 3d collaborative filtering with grouping constraint in luminance-chrominance space. *in Proc. of IEEE ICIP*, 1:313–316, 2007.

[4] K. He, J. Sun, and X. Tang. Guided image filtering. *in Proc of IEEE ECCV*, 2010.

[5] L. Over and M. Werman. Color lines: Image specific color representation. *in Proc. of IEEE Conf. on CVPR*, 2:946–953, 2004.

[6] G. Sharma, W. Wu, and E. N. Dalal. The ciede2000 color-difference formula: implementation notes, supplementary test data, and mathematical observations. *Color Reserach and Application*, 30(1):21–30, 2005.

[7] K. Shirai, H. Shoji, and M. Ikehara. Color-line considered anisotoropic diffusion for smoothing of color images. *IEICE Trans. (Japanese)*, J93-A(3):181–189, 2010.

[8] D. Tschumperlé and R. Deriche. Vector-valued image regularization with pde's: A common framework for different applications. *IEEE Trans. on PAMI*, 27(4):506–517, 2005.

[9] L. Yang, P. V. Snder, J. Lawrence, and H. Hoppe. Antialiasing recovery. *ACM TOG (Proc. of SIGGRAPH)*, 30, 2011.