FIREFLY: A HARDWARE-FRIENDLY REAL-TIME LOCAL BRIGHTNESS ADJUSTMENT METHOD

Nikola Banić and Sven Lončarić

Image Processing Group, Department of Electronic Systems and Information Processing
University of Zagreb, Faculty of Electrical Engineering and Computing, Croatia
E-mail: {nikola.banic, sven.loncaric}@fer.hr

ABSTRACT

Brightness adjustment is an important part of image enhancement, but some of the best brightness adjustment methods in terms of result quality are too complex to run in real-time and many are not suitable for hardware implementation. In this paper a fast learning-based and hardware-friendly method for local brightness adjustment is proposed that in real-time obtains results of higher quality than some of the best methods. The results are presented and discussed. The source code is at `http://www.fer.unizg.hr/ipg/resources/color_constancy/`.

***Index Terms*—** Brightness adjustment, fitting, hardware-friendly, image enhancement, interpolation, look-up table.

1. INTRODUCTION

Brightness adjustment is an important task in image enhancement and it is part of many image enhancement methods. These include simple modifications of histogram equalization [1]–[3], adaptive histogram equalization [4] [5], Retinex-based methods [6]–[12], naturalness preserving methods [13] [14], unsharp masking [15] [16], automatic color equalization [17] [18], etc. Many specific image processing or computer vision tasks perform well only if brightness adjustment and image enhancement in general produce results of sufficient quality [19].

For real-time hardware implementations the applied method should be as fast as possible, simple, and hardware-friendly [20]. Unfortunately, many methods that produce high-quality results are either too slow to run in real-time or too complex for a simple and cheap implementation.

In this paper, a novel hardware-friendly local brightness adjustment method is proposed. It uses only pixel intensities and the mean of their local neighborhoods. A pre-calculated look-up table avoids complex calculations and results in high speed. The proposed method is named Firefly and it is shown to outperform several other brightness adjustment methods in terms of computation speed and resulting image quality.

The paper is structured as follows: in Section II the motivation for a new model is explained, in Section III the proposed method is described, in Section IV experimental results are presented and discussed. Section V concludes the paper.

2. MOTIVATION

Image brightness adjustment is performed by processing its luminance channel. In most cases this is the Y channel of the image in the YUV colorspace [21]. Other luminance channels may be used as well, e.g. the ones of the HSV, HSL, and Lab colorspaces or a parametrized luminance channel [22]. The luminance value Y of a pixel represented in the RGB colorspace as a vector $p = (R, G, B)^T$ is

$$Y = 0.299R + 0.587G + 0.114B. \tag{1}$$

If Y is mapped to Y' after brightness adjustment, then in the RGB colorspace the pixel p is mapped to

$$p' = \frac{Y'}{Y}p = \left(\frac{Y'}{Y}R, \frac{Y'}{Y}G, \frac{Y'}{Y}B\right)^T. \tag{2}$$

Local brightness adjustment generally produces results of higher quality than global brightness adjustment. Adjusting the value of Y to Y' can be described by a mapping

$$Y' = f\left(Y, \mathbf{m}\right) \tag{3}$$

where f is a local adjustment function and the local neighborhood around p is described by a parameter vector $\mathbf{m} = (m_1,\ m_2, ..., m_n)^T$. Examples of $\mathbf{m}$ include the histogram of local neighborhood of p as in contrast limited adaptive histogram equalization (CLAHE) [5] or the whole image as in Automatic Color Equalization (ACE) [17]. If the components of $\mathbf{m}$ have to be calculated first before applying f to obtain Y', then the dimension and calculation of $\mathbf{m}$ influence both the computational cost of f and the suitability of the whole method for hardware implementation.

If only a single scalar is used to describe the local neighborhood of p, then Eq. (3) is simplified to

$$Y' = f\left(Y, m\right). \tag{4}$$

Further, in many cases images are represented using 8 bits per channel, which means that the values of both Y and Y' are from the set $U = \{0,\ 1, ..., 255\}$. If m is also represented by using only values from U, then the mapping of f is

$$f : U \times U \mapsto U. \tag{5}$$

If values of such f are pre-calculated and stored into a 256×256 look-up table (LUT), the per pixel complexity of brightness adjustment depends only on calculation of m since reading from LUT can be performed in $O(1)$. Three problems arise: what should the parameter m represent, how to calculate m fast, and which values for LUT give best results?

3. THE PROPOSED METHOD

3.1. Local neighborhood description

One of the simplest choices for m in Eq. (4) is the mean brightness of the local neighborhood. By using integral images it can be calculated in $O(1)$ per pixel complexity [23]. If the memory available for calculating and storing integral images is limited, an alternative is to take only $r \times r$ pixels evenly spread across the image and to perform the averaging on them by applying a $k \times k$ kernel. This approximates the mean brightness around the taken $r \times r$ image pixels. To give the closer pixels more weight, the averaging can be repeated s times. For any other image pixel p the mean can be approximated by first taking the already calculated means for four of the processed $r \times r$ pixels closest to p and merging them with bilinear interpolation in $O(1)$ per pixel complexity.

3.2. LUT learning algorithm

A learning-based approach is used to determine LUT values. The learning data consists of a set of images and a set of their versions enhanced by some previously chosen image enhancement methods that give high quality enhancement results. More detail on how to choose these methods will be given in Section 4.2. Given an image and its enhanced version, for each pixel p in the original image and its corresponding pixel p' in the enhanced image a triplet (Y, m, Y') can be obtained where Y and Y' are the luminance values of p and p', respectively, and m describes the local neighborhood of p. The idea is to use many such triplets (Y_i, m_i, Y'_i) as ground-truth to fill LUT's values. However, instead of one, for some combinations of Y and m there may be more triplets or even none, which opens the question which Y' to store in LUT.

As for multiple triplets with the same values of Y and m, they can all be replaced with a single triplet (Y, m, Y'') where Y'' is the mean of multiple triplets' values of Y'_i:

$$Y'' = \frac{\sum\limits_{Y=Y_i, m=m_i} Y'_i}{\sum\limits_{Y=Y_i, m=m_i} 1}. \tag{6}$$

To solve the problem of missing triplets for some values of Y and m, first a model for f in Eq. (4) is chosen, e.g. a polynomial of an arbitrary order n:

$$f(Y, m) = \sum_{\substack{i=0 \\ i+j \le n}}^{n} \sum_{j=0}^{n} a_{ij} Y^i m^j = \\ = a_{00} + a_{10} Y + a_{01} m + ... + a_{1n-1} Y m^{n-1} + a_{0n} m^n. \tag{7}$$

The chosen model is then fitted to points representing the available triplets. Since f has two parameters, this is essentially a surface fitting. After the fitting, f is used to fill the whole LUT. If f is smooth, then so is the surface stored in LUT, which is good for the quality of the final result.

The proposed method is named Firefly for its simplicity and the pseudocodes for its learning and application phases are given in Algorithm 1 and Algorithm 2, respectively.

Algorithm 1 Firefly - Learning

$T = \{\}$
for $i = 1..imgsN$ **do**
  $I = LoadImage(i)$
  $E = EnhanceImage(I)$
  $T = T \cup ExtractTriplets(I, E)$
end for
$T' = MergeTriplets(T)$ using Eq. (6)
$f = FitToData(T')$
for $Y = 0..255$ **do**
  for $m = 0..255$ **do**
    $LUT(Y, m) = \min(\max(\lfloor f(Y, m) \rfloor, 0), 255)$
  end for
end for

Algorithm 2 Firefly - Application

$I = GetImage()$
for $i = 1$ **to** $I.rowsCount$ **do**
  for $j = 1$ **to** $I.columnsCount$ **do**
    $Y = I.GetLuminance(i, j)$
    $m = I.CalculateInterpolatedMean(i, j)$
    $Y' = LUT(Y, m)$
    $I.SetLuminance(i, j, Y')$
  end for
end for

3.3. Sharp images

For some images the application of Firefly may result in loss of sharpness. This can be simply fixed by calculating the ratio $f(Y, m)/Y$ for all pixels, averaging these ratios with a 3×3 kernel, and multiplying the initial pixel brightness with the new averaged ratios. However, since this is rarely needed be-

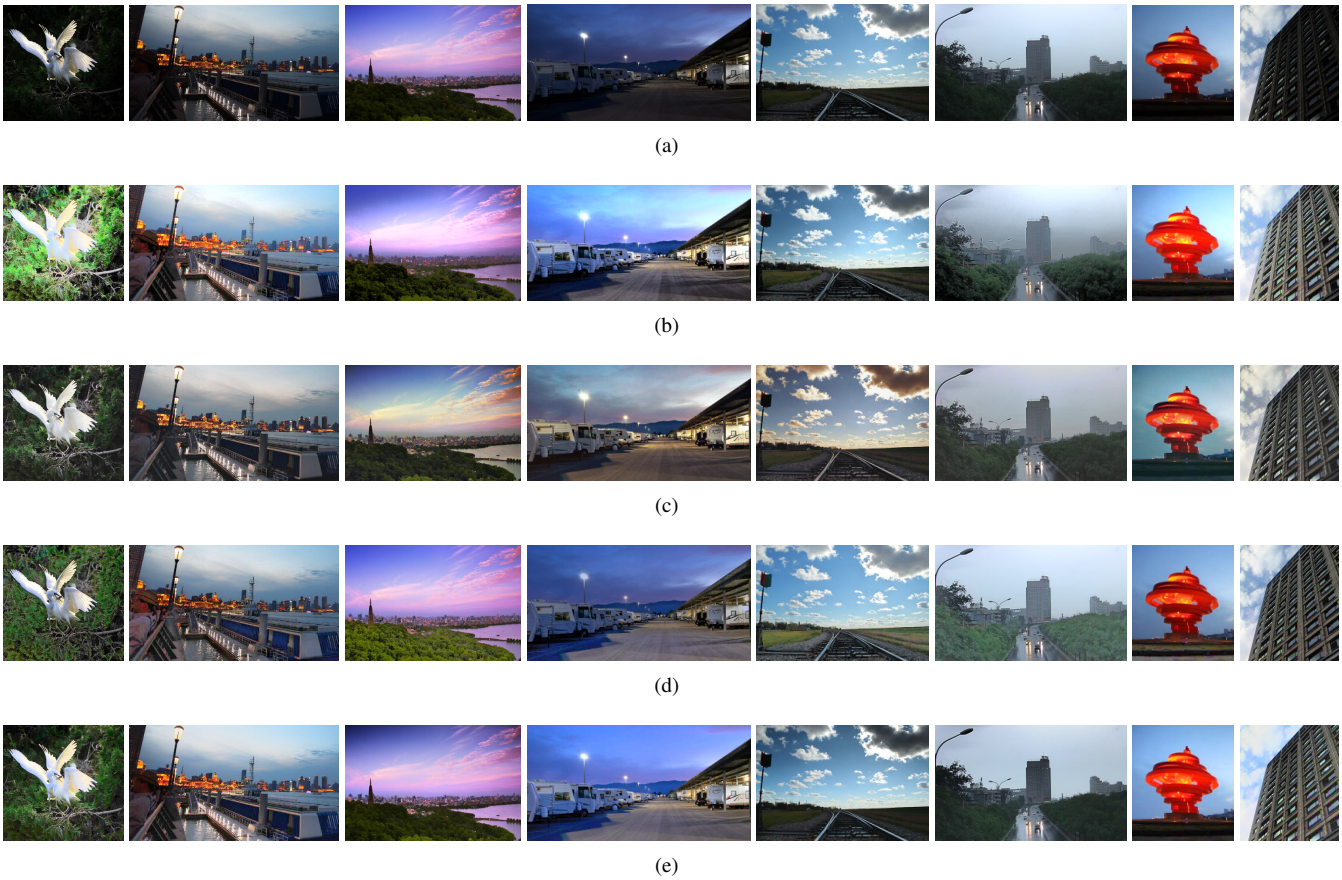

Fig. 1. Processing of NPEA test images: (a) original, (b) HE, (c) ACE, (d) NPEA, and (e) Firefly.

cause it represents a special case, the experimental results in the next section were obtained without applying this step.

4. EXPERIMENTAL RESULTS

4.1. Image datasets

To obtain the triplets, all 568 8-bit images from the ColorChecker dataset [24] were used. Most of them are of size 874×583 and many have unadjusted brightness. The images were enhanced by performing histogram equalization (HE), Automatic Color Equalization (ACE) [17], and Naturalness Preserved Enhancement Algorithm (NPEA) [14]. The obtained triplets were used to fit the polynomial in Eq. 7 for $n = 5$. Optimal values for Firefly's parameters r, k, and s were determined as described later by using an online available challenging dataset described in [14]. All methods including Firefly were finally tested on the same eight images that were used to test NPEA in its original paper [14].

4.2. Image quality assessment

Several objective image quality assessment measures were used to test the performance of the Firefly method: statistical naturalness (SN) [25], information content (IC) [26], and local information content (LIC) [27]. SN is a number in interval $[0, 1]$ based on intensity statistics of natural images with higher being better, IC measures the Shannon entropy of the image luminance values, and LIC is the mean IC for 8×8 equal image patches as proposed in [27]. Both IC and LIC can be seen as contrast measures on global and local scale, respectively, and they are calculated on the grayscale versions of the images provided by Eq. (1). To measure their combined value during the learning of Firefly's parameter values, the product SN·IC·LIC was used.

Four different cases of obtaining Firefly LUTs have been tested. For each case different image pairs were used to obtain the triplets. To obtain the triplets for the first three cases, the images from the ColorChecker dataset were enhanced by applying ACE, HE, and NPEA, respectively. For the fourth case the image pairs were generated so that for each of the original ColorChecker image its enhanced version was the one from the previous three cases that had the highest product SN·IC·LIC. These four cases are denoted $\text{Firefly}_{\text{ACE}}$, $\text{Firefly}_{\text{HE}}$, $\text{Firefly}_{\text{NPEA}}$, and $\text{Firefly}_{\text{all}}$, respectively. For each of them the rest of the testing process was the same.

First, Algorithm 1 was performed for every combination

of parameters $r \in \{5, 10, ...100\}, k \in \{1, 2, ..., 5\}$, and $s \in \{0, 1, ...5\}$ on ColorChecker images. Each combination resulted in one LUT, which was then applied to the dataset described in [14]. The LUT that resulted in the highest mean product SN·IC·LIC for the images of this dataset was used to perform the final testing on the eight images that were also used for the final validation of NPEA [14]. In this way each part of the testing process was performed on a different image set. The results and images of the final tests are shown in Table 1 and Fig. 1, respectively. It is evident that the image pairs used to create the triplets play a significant role in the Firefly's resulting image quality. When the combined results of other methods are used to create triplets, Firefly outperforms all other tested methods in terms of objective quality measures. The LUT with triplets and the code to reproduce and use it is available at `http://www.fer.unizg.hr/ipg/resources/color_constancy/`. It is important to mention that in the process of learning the values for LUT other image quality measures can also be used successfully.

Table 1. Performance measures for different methods applied to images used to originally test NPEA (higher is better).

Method	mean SN	mean IC	mean LIC	mean SN·IC·LIC
do nothing	0.34	7.17	5.38	14.00
ACE	0.71	7.52	5.85	31.79
HE	0.64	**7.69**	5.71	28.18
NPEA	0.58	7.47	5.86	26.18
$\text{Firefly}_{\text{ACE}}$	0.74	7.36	5.63	30.71
$\text{Firefly}_{\text{HE}}$	0.73	7.50	5.76	31.33
$\text{Firefly}_{\text{NPEA}}$	0.57	7.34	5.50	23.81
$\text{Firefly}_{\text{all}}$	**0.78**	7.44	**6.02**	**34.33**

4.3. Interpolation

For hardware implementation the 256×256 LUT can be downsampled to $d \times d$. Bilinear interpolation can approximate any original value. Fig. 2 shows that for results obtained with and without interpolation, the average per pixel perceptual CIELab difference ΔE^*_{ab} is below the just-noticeable difference (JND) threshold of 2.3 [28] for most values of d.

4.4. Computation speed

The computation speed test was performed on a computer with Intel(R) Core(TM) i5-2500K CPU with only one core used. For ACE, HE, and NPEA their online available source codes were used and the parameters were set to default values. In the case of ACE, its fast implementation [18] was used, for HE its OpenCV implementation was used, and for NPEA the version provided by the authors [14] was used. The results in Table 2 show that in terms of computation speed Firefly outperforms all other tested methods.

Table 2. Combined computation time for first 100 images of the ColorChecker dataset.

Method	Time (s)
ACE	235.20
HE	4.37
NPEA	2024.48
Firefly	**3.95**

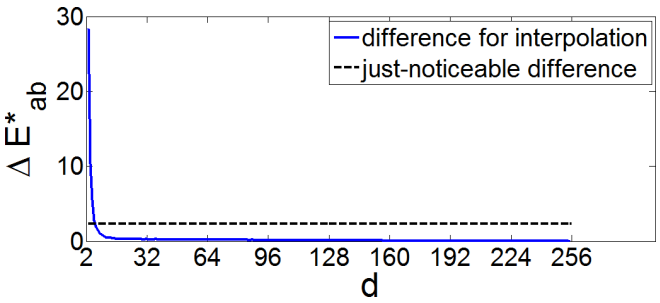

Fig. 2. Average per pixel ΔE^*_{ab} for NPEA test images between results obtained with a 256×256 and a $d \times d$ LUT.

5. CONCLUSIONS

A novel fast learning-based and hardware-friendly method for local brightness adjustment has been proposed. It is shown to achieve results of higher quality than some successful and more complex models. The method also outperforms all other tested methods in terms of computation speed.

6. ACKNOWLEDGEMENT

The authors thank Nenad Markuš for his useful advice. This research has been partially supported by the European Union from the European Regional Development Fund by the project IPA2007/HR/16IPO/001-040514 "VISTA - Computer Vision Innovations for Safe Traffic."

7. REFERENCES

[1] Yu Wang, Qian Chen, and Baomin Zhang, "Image enhancement based on equal area dualistic sub-image histogram equalization method," *Consumer Electronics, IEEE Transactions on*, vol. 45, no. 1, pp. 68–75, 1999.

[2] Chao Wang and Zhongfu Ye, "Brightness preserving histogram equalization with maximum entropy: a variational perspective," *Consumer Electronics, IEEE Transactions on*, vol. 51, no. 4, pp. 1326–1334, 2005.

[3] Haidi Ibrahim and Nicholas Sia Pik Kong, "Brightness preserving dynamic histogram equalization for image contrast enhancement," *Consumer Electronics, IEEE Transactions on*, vol. 53, no. 4, pp. 1752–1758, 2007.

[4] Stephen M Pizer, E Philip Amburn, John D Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart ter Haar Romeny, John B Zimmerman, and Karel Zuiderveld, "Adaptive histogram equalization and its variations," *Computer vision, graphics, and image processing*, vol. 39, no. 3, pp. 355–368, 1987.

[5] Karel Zuiderveld, "Contrast limited adaptive histogram equalization," in *Graphics gems IV*. Academic Press Professional, Inc., 1994, pp. 474–485.

[6] Edwin H Land and John McCann, "Lightness and retinex theory," *JOSA*, vol. 61, no. 1, pp. 1–11, 1971.

[7] Daniel J Jobson, Z-U Rahman, and Glenn A Woodell, "A multiscale retinex for bridging the gap between color images and the human observation of scenes," *Image Processing, IEEE Transactions on*, vol. 6, no. 7, pp. 965–976, 1997.

[8] Zia-ur Rahman, Daniel J Jobson, and Glenn A Woodell, "Retinex processing for automatic image enhancement," *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 100–110, 2004.

[9] Brian Funt, John McCann, and Florian Ciurea, "Retinex in MATLAB," *Journal of electronic imaging*, vol. 13, no. 1, pp. 48–57, 2004.

[10] Edoardo Provenzi, Massimo Fierro, Alessandro Rizzi, Luca De Carli, Davide Gadia, and Daniele Marini, "Random spray retinex: a new retinex implementation to investigate the local properties of the model," *Image Processing, IEEE Transactions on*, vol. 16, no. 1, pp. 162–171, 2007.

[11] Nikola Banić and Sven Lončarić, "Light Random Sprays Retinex: Exploiting the Noisy Illumination Estimation," *Signal Processing Letters, IEEE*, vol. 20, no. 12, pp. 1240–1243, 2013.

[12] Nikola Banić and Sven Lončarić, "Color Badger: A Novel Retinex-Based Local Tone Mapping Operator," in *Image and Signal Processing*, pp. 400–408. Springer, 2014.

[13] Kai-Qi Huang, Qiao Wang, and Zhen-Yang Wu, "Natural color image enhancement and evaluation algorithm based on human visual system," *Computer Vision and Image Understanding*, vol. 103, no. 1, pp. 52–63, 2006.

[14] Shuhang Wang, Jin Zheng, Hai-Miao Hu, and Bo Li, "Naturalness preserved enhancement algorithm for non-uniform illumination images," *Image Processing, IEEE Transactions on*, vol. 22, no. 9, pp. 3538–3548, 2013.

[15] Andrea Polesel, Giovanni Ramponi, V John Mathews, et al., "Image enhancement via adaptive unsharp masking," *Image Processing, IEEE Transactions on*, vol. 9, no. 3, pp. 505–510, 2000.

[16] Guang Deng, "A generalized unsharp masking algorithm," *Image Processing, IEEE Transactions on*, vol. 20, no. 5, pp. 1249–1261, 2011.

[17] Alessandro Rizzi, Daniele Marini, and Carlo Gatta, "From retinex to automatic color equalization: issues in developing a new algorithm for unsupervised color equalization," *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 75–84, 2004.

[18] Pascal Getreuer, "Automatic color enhancement (ACE) and its fast implementation," *Image Processing On Line*, , no. 2012, 2012.

[19] HK Sawant and Mahentra Deore, "A comprehensive review of image enhancement techniques," *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, vol. 1, no. 2, pp. 39–44, 2010.

[20] Iva Harbaš, Nikola Banić, Darko Jurić, Sven Lončarić, and Marko Subašić, "Computer vision-based advanced driver assistance systems," in *34th Conference on Transportation Systems with International Participation AUTOMATION IN TRANSPORTATION 2014*, 2014.

[21] A. Koschan and M. Abidi, *Digital Color Image Processing*, Wiley, 2008.

[22] Nikola Banić and Sven Lončarić, "Improving the Tone Mapping Operators by Using a Redefined Version of the Luminance Channel," in *Image and Signal Processing*, pp. 392–399. Springer, 2014.

[23] Franklin C Crow, "Summed-area tables for texture mapping," *ACM SIGGRAPH computer graphics*, vol. 18, no. 3, pp. 207–212, 1984.

[24] Peter V Gehler, Carsten Rother, Andrew Blake, Tom Minka, and Toby Sharp, "Bayesian color constancy revisited," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[25] Hojatollah Yeganeh and Zhou Wang, "Objective quality assessment of tone-mapped images," *Image Processing, IEEE Transactions on*, vol. 22, no. 2, pp. 657–667, 2013.

[26] Monica Borda, *Fundamentals in information theory and coding*, vol. 6, Springer, 2011.

[27] Li-Yu Chang and Chi-Farn Chen, "The Enhancement of Remote Sensing Images by Local Brightness Balancing," in *Asian Association on Remote Sensing*, 2013.

[28] Marc Mahy, Luc Van Eycken, and André Oosterlinck, "Evaluation of uniform color spaces developed after the adoption of CIELAB and CIELUV," *Color research and application*, vol. 19, no. 2, pp. 105–121, 1994.