新しいオートホワイトバランス（ＡＷＢ）のための日向・日陰領域分割

Shade Segmentation for New Automatic White Balance (AWB)

袁　勛*　李　滔*　師　忠超*　王　剛*　関　海克**
Xun YUAN　Tao LI　Zhongchao SHI　Gang WANG　Haike GUAN

要　旨

日向と日陰が混在するシーンを撮影するとき，画面全体で均一なホワイトバランスの設定が行われる従来法では，最適な色再現は困難である．本論文で提案する新しいＡＷＢ技術では日向と日陰領域ごとにホワイトバランスパラメータを設定することで，最適な色再現ができる．そのため，日向・日陰領域を高精度かつ自動的に認識する技術が必要となる．

本論文は新しいＡＷＢ処理ための日向・日陰領域の自動認識方法を提案する．本技術は晴天・曇天認識，日向・日陰認識，空認識，認識結果の後処理の4つのステップで日向・日陰の自動認識を行う．

多様なシーンについて実験を行い，本技術の日向・日陰の認識率が90%以上であることを確認した．アルゴリズムの高速化検討を行い，リアルタイム処理を実現した．

ABSTRACT

Traditional automatic white balance (AWB) functions could not adequately handle images with both sunlight and shadow. In this situation, new AWB is powerful by setting different AWB parameters for different image regions. Shade segmentation is the key problem for new AWB, which aims at segmenting bright and shadow regions in an image with very high accuracy.

This article presents our contributions on shade segmentation for new AWB, which consists of the following four steps: sunny-vs.-rainy image classification, shade detection, sky detection, and post-processing.

Extensive experiments on a wide variety of images demonstrate that our shade segmentation algorithm is able to discriminate the bright and shade area with more than 90% accuracy, and achieve real-time processing speed.

＊ リコーソフトウェア研究所（北京）有限公司
Ricoh Software Research Center (Beijing) Co., Ltd.
＊＊ グループ技術開発本部　デバイスモジュール技術開発センター
Device and Module Technology Development Center, Corporate Technology Development Group

1. Introduction

Automatic White Balance (AWB) is the automatic adjustment of the intensities of the colors[1] (typically red, green, and blue primary colors). An important goal of this adjustment is to render specific colors (particularly white colors) correctly.

Traditional AWB algorithms make global adjustment of the color in the whole image using uniform transformation parameters. It could not adequately handle images with both sunlight and shadow. In this situation, new AWB is powerful by setting different AWB parameters for different image regions. Shade segmentation is the key problem for new AWB, which aims at segmenting bright and shadow regions in an image with very high accuracy.

The framework of shade segmentation for new AWB is illustrated in Fig. 1 below. In the first step, images are classified as sunny image or rainy image. As there is no sunlight and shadow area in rainy image, we only perform new AWB for sunny images. In the second step, shade detection is used to classify each image block as bright or shadow image block at a coarse level. Because shade detection usually classifies blue sky area as shadow area falsely, we use the third step, i.e. sky detection to correct such misclassified blocks. In the final step, a post-processing step is performed to merge neighborhood image blocks with similar color appearance as one area, in order to remove some misclassified small areas and improve the segmentation accuracy.

As the first three steps are all classification problems, we use machine learning techniques to build the final classification model, based on a series of features. The framework for the three tasks is illustrated in Fig. 2. In the training process, features are extracted from training images (for sunny-vs.-rainy image classification) or image blocks (for sky detection and shade detection).

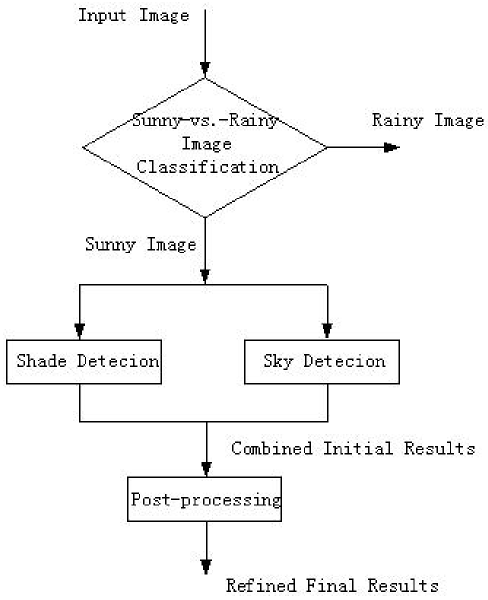

Fig. 1 Framework of Shade Segmentation for New AWB.

Then a model is trained based on the features of training images (or image blocks). In the test process, we also extract features from test images (or image blocks) firstly as in the training process. After that, the model output by the training process is used to predict the labels of the test images (or image blocks), based on the features of test images (or image blocks).

The three classification tasks are highly related to an academic research field named scene classification. Extensive research has been dedicated to this topic in recent years. The major difference between these works is different scene types, different features, and different classification models. Due to limited page, we only select some most relevant, or the most famous works[2][3][4][5][6][7], and list them in the reference.

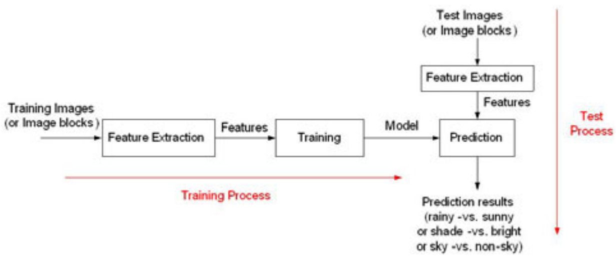

Fig. 2 Framework for the Three Classification Tasks in New AWB.

In the final step, a region grow procedure is performed to merge neighborhood image regions with similar color appearance, and get refined shade segmentation results.

The rest of this paper is organized as follows. In Section 2, we introduce the learning-based method for the first three steps of shade segmentation, including both feature extraction and classifier building. Then, we give the description of post processing procedure in Section 3. The experimental results are presented in Section 4, followed by the conclusions in Section 5.

2. Learning-Based Method for the First Three Steps

In this section, we introduce the learning-based method for the first three steps. As illustrated in Fig. 2, there are two major components for learning-based method: feature extraction, and classifier building. We introduce the different features for the three steps in subsection 2-1. After that, two alternative classifiers (SVM and AdaBoost) are compared in subsection 2-2.

2-1 Feature Extraction

We test different effective features for sunny-vs.-rainy image classification, sky detection, and shade detection. The features include color features, image capture device parameters, and some transformations of them. To achieve real time processing speed, we just use simple features here. Experimental results show that such features are discriminative enough for the three classification tasks.

2-1-1 Features for Sunny-vs.-Rainy Image Classification

The following 8 features for sunny-vs.-rainy image classification have been tested now:

- 2 calibrated AE-related features: mean of calibrated AE, standard deviation of calibrated AE
- 3 image capture device parameters: exposure time, ISO number, F number
- 2 B/R-related features: mean of B/R, standard deviation of B/R

Here AE are all calibrated for different parameters of image capture device as follow:

$$\mathrm{AE} = \mathrm{AE} * \mathrm{rate} \tag{1}$$

$$\mathrm{rate} = \left(\frac{0.01}{\mathrm{exposure\_time}}\right) \times \left(\frac{200}{\mathrm{ISO}}\right) \times \left(\frac{\mathrm{F\_num}}{4}\right)^2 \tag{2}$$

Where exposure_time, ISO, F_num refer to the exposure time, ISO number, and F number of the image capture device.

The mean and standard deviation of AE are calculated as follows:

$$mean_{AE} = \frac{\sum_{i=1}^{N} AE_i}{N} \quad std_{AE} = \sqrt{\frac{\sum_{i=1}^{N} (AE_i - mean_{AE})^2}{N-1}} \tag{3}$$

Where N is the total number of image blocks.

2-1-2 Features for Sky Detection

Sky detection has been investigated by Luo et. al[4)5)6)]. However, their methods have two limitations in our application: first, the high computation complexity makes it impossible for real time processing; second, it makes an assumption that sky area has a de-saturation rate from the top to the bottom of one image, which is not necessarily the case in our experimental images.

To overcome the above limitations, we design another simple yet effective method to perform real time sky detection. It should be pointed out that our method cannot discriminate sky from other blue colored objects as in [4)5)6)]. However, our aim of sky detection is for the shade segmentation of new AWB. In this application, such false alarms (misclassified blue colored objects) have little negative impact to the final AWB results.

In sunny-vs.-rainy image classification, the features are extracted from the whole image. While in sky detection and shade detection, the features are extracted from each image block. Currently, we have tested the following features for sky detection:

- R, Gr, Gb, B
- hue, value, saturation
- B/R, B/G

Hue, value, and saturation are the block's color components in HSV color space.

2-1-3 Features for Shade Detection

The following features for shade detection have been tested now:

- R, Gr, Gb, B
- AE value
- B/R
- Saturation

2-2 Classifier Building

For the classifier building, we compare the following two alternatives: Support Vector Machine[9] (SVM), and AdaBoost[10)11)]. We firstly introduce linear-kernel SVM in subsection 2-2-1, and then AdaBoost in subsection 2-2-2.

2-2-1 Linear-Kernel SVM

Due to limited page, we give a brief description of SVM theory here, please refer to [9] for more details. SVM is proposed from optimal hyper-plane in linear separable situation. Then it is extended in linear inseparable situation. As shown in Fig. 3, SVM is trying to maximize the margin between positive and negative samples, while minimize the training error.

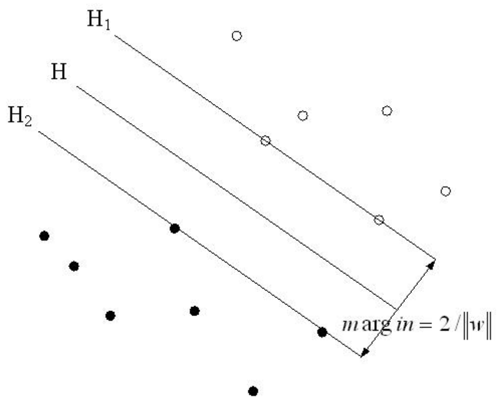

Fig. 3 Illustration of Optimal Hyper-plane.

In the training process, SVM tries to solve the following optimization problem:

$$\phi(w,\varepsilon) = \frac{1}{2}(w \bullet w) + C(\sum_{i=1}^{n} \varepsilon_i)$$
$$y_i[(w \bullet x_i) + b] - 1 + \varepsilon_i \geq 0 \qquad (4)$$

In Eqn. 4, $w \cdot w$ stands for the margin of the classifier (as shown in Fig. 3), $\sum_{i=1}^{n} \varepsilon_i$ is the classification error, and C is a pre-defined constant. Therefore, we can see that C controls the tradeoff between classification error and margin of the classifier.

The prediction function for SVM is:

$$f(x) = \mathbf{sgn}(\sum_{i=1}^{sv\_num} alphaY_i * K(x_i \bullet x) - rho) \qquad (5)$$

where sv_num is the total number of support vectors x_i, $K(x_i \cdot x)$ is the kernel function of the two vectors x_i and x.

To speed up the prediction time, we adopt the linear kernel. In this case, the kernel function $K(x_i \cdot x)$ becomes the inner product $(x_i \cdot x)$, and the prediction function can be re-written as:

$$\begin{aligned} f(x) &= \mathbf{sgn}(\sum_{i=1}^{sv\_num} alphaY_i * (x_i \bullet x) - rho) \\ &= \mathbf{sgn}(\sum_{i=1}^{sv\_num} ((alphaY_i * x_i) \bullet x) - rho) \\ &= \mathbf{sgn}((\sum_{i=1}^{sv\_num} alphaY_i * x_i) \bullet x - rho) \\ &= \mathbf{sgn}(w \bullet x - rho) \end{aligned} \qquad (6)$$

Where w can be pre-computed before the prediction, the prediction time can be reduced greatly.

2-2-2 Real AdaBoost

Boosting[10] is a method of finding a high accurate hypothesis (classification rule) by combining many "weak" hypotheses, each of which is only moderately accurate. We use a generalized version of AdaBoost, which is called Real AdaBoost[11], in which the weak learner returns a real-value prediction.

Next we give a brief introduction of the weak learner and Real AdaBoost procedure we used here.

The weak learner is a histogram-type classifier constructed on each single dimension of the feature vector, as illustrated in Fig. 4.

The output of the weak classifier is a piece-wise function, as shown in Fig. 5 below.

Input: (x_1, y_1), ..., (x_m, y_m) where y_i = 0, 1 for negative and positive examples respectively, training sample weight $w(x_i)$

1) Divide the feature space into n sub space $X_1, X_2, \ldots, X_n$
2) Given the training sample weight $w(x_i)$, calculate W_l^j, l=0, 1, j=1, 2, ..., n

$$W_l^j = P(x_i \in X_j, y_i = l) = \sum_{i:x_i \in X_j, y_i = l} w(x_i)$$

3) The output of the weak learner on the training sample weight $w(x_i)$ is

$$h(x) = \frac{1}{2}\log(\frac{W_1^j + \varepsilon}{W_0^j + \varepsilon}), \quad j = 1, \ldots, n, \quad \forall x \in X_j$$

where ε is a small positive constant

4) The corresponding training error is calculated as follows:

$$err = \sum_{h(x_i) \geq 0, y_i = 0} w(x_i) + \sum_{h(x_i) < 0, y_i = 1} w(x_i)$$

Fig. 4 The Weak Classifier.

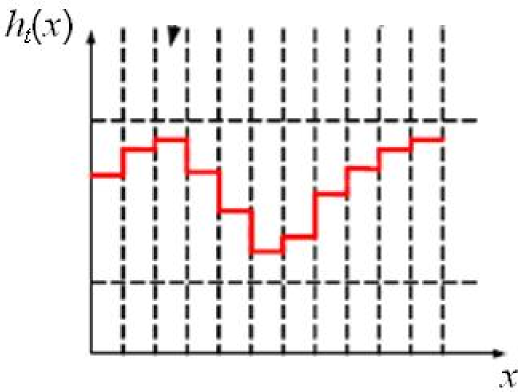

Fig. 5 The Output Piece-Wise Function of the Weak Classifier.

Based on this real-value output weak learner, our AdaBoost procedure is shown in Fig. 6.

In our AdaBoost procedure, the sums of sample weights for positive and negative samples are always kept equal to be 1/2. This is to tackle the unbalance problem in training data when the number of negative samples is much larger than the number of positive samples.

Input: (x_1, y_1), ⋯, (x_m, y_m) where y_i = -1, 1 for negative and positive examples respectively

1) Initialize weight $w(x_i)$ =1/2n, 1/2p for y_i=-1, 1 respectively, where n and p are the number of negative and positive samples respectively.

2) For t = 1, ..., T

a) According to the weak learner shown in Fig. 4, train one weak learner $h_j \in R$ for each feature j using $w(x_i)$, and get the corresponding training error err_j.

b) Choose h_t=h_j with the lowest error, set err_t=err_j.

c) Update:

$$w_{t+1}(x_i) = \frac{w_t(x_i)}{Z_{tp}} \times e^{-y_i h_t}$$

for positive samples, and

$$w_{t+1}(x_i) = \frac{w_t(x_i)}{Z_{tn}} \times e^{-y_i h_t}$$

for negative samples, where Z_{tp} and Z_{tn} are normalization factors to ensure that $w_{t+1}(x_i)$ will be a distribution, and the weight of positive and negative samples all sum up to 1/2.

Output:

The final hypothesis is

$$H(x) = \text{sgn}(\sum_{t=1}^{T} h_t(x))$$

Fig. 6 The Real AdaBoost Algorithm Uses Real-Valued Weak Classifier Output h_t.

3. Post Processing

After the first three steps, post-processing is performed to merge neighborhood image blocks with similar color appearance. The procedure of post-processing is shown in Fig. 7:

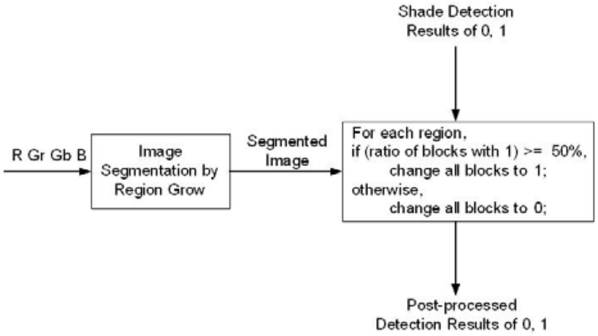

Fig. 7 The Procedure of Post-Processing.

In region grow [8], neighborhood image blocks with similar color appearances will be merged as one region. The "similar" is judged by calculating the color difference between image blocks:

$$Diff_{ij} = abs(Gr_i - Gr_j) + abs(Gb_i - Gb_j) + abs(R_i - R_j) + abs(B_i - B_j) \quad (7)$$

Where the suffix *i* and j indicate the color components from image blocks *i* and *j*. If $Diff_{ij}$ is smaller than a predetermined threshold *T*, the two neighborhood blocks are considered to have "similar" color difference, and will be merged into one region. The threshold *T* indicates the difference between neighborhood blocks. Currently, we set it empirically to be 50.

4. Experimental Results

In this section, we give the experimental results for the four steps in four subsections respectively.

4-1 Sunny-vs.-Rainy Image Classification

The image dataset for scene recognition contains 166 images (75 for sunny, and 91 for rainy). The following four feature combinations are compared, and the accuracy is listed in Table 1.

- 2-D: mean of AE, standard deviation of AE
- 3-D: exposure time, F number, ISO number
- 4-D: mean of AE, standard deviation of AE, mean of B/R, standard deviation of B/R
- 7-D: mean of AE, standard deviation of AE, mean of B/R, standard deviation of B/R, exposure time, F number, ISO number

Table 1 Accuracy for Sunny-vs.-Rainy Image Classification.

	2-D	3-D	4-D	7-D
Linear-kernel SVM	60.93%	62.50%	81.25%	76.56%
AdaBoost	59.38%	70.31%	78.13%	81.25%

4-2 Sky Detection

The experiments for sky detection are performed in 25 images (16 images for training, and the rest 9 images for test). The following feature combinations are tested and the results are summarized in Table 2.

- 3-D: R, (Gr+Gb)/2, B
- 4-D: R, (Gr+Gb)/2, B, saturation
- 5-D: R, (Gr+Gb)/2, B, B/R, B/G
- 6-D: R, (Gr+Gb)/2, B, B/R, B/G, saturation
- 8-D: R, (Gr+Gb)/2, B, B/R, B/G, saturation, hue, value

Table 2 Accuracy for Sky Detection.

	3-D	4-D	5-D	6-D	8-D
Linear-kernel SVM	93.97%	93.53%	93.16%	93.18%	93.15%
AdaBoost	92.88%	90.37%	91.67%	91.35%	91.35%

Fig. 8 shows two sample images for sky detection. We can see Most of the sky area is successfully located in the images.

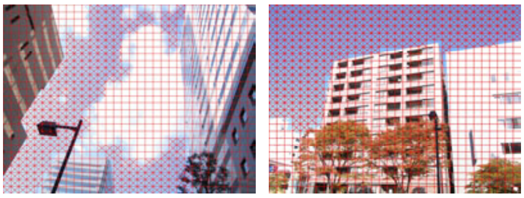

Fig. 8　Sample Result Imagesof Sky Detection.

4-3　Shade Detection

The experiments for shade detection are performed in 90 images (64 images for training, and the rest 26 images for test). The following feature combinations are tested and the results are summarized in Table 3.

- 1-D: AE value
- 3-D: R, G, B
- 4-D: R, G, B, AE value
- 5-D: R, G, B, AE value, B/R
- 6-D: R, G, B, AE value, B/R, saturation

Table 3　Accuracy for Shade Detection.

	1-D	3-D	4-D	5-D	6-D
Linear-kernel SVM	89.69%	89.88%	89.86%	90.02%	91.37%
AdaBoost	89.59%	90.82%	90.29%	91.39%	91.57%

Fig. 9 shows two sample result images for shade detection.　We can see that some sky area is misclassified as shade area in shade detection, which should be corrected in the sky detection step.

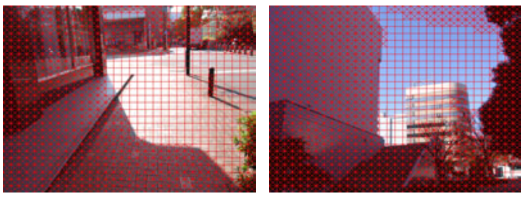

Fig. 9　Sample Result Images of Shade Detection.

4-4　Post-Processing

The effect of post processing is shown in Fig. 10.　We can see that there is some misclassified area within the blue circle in Fig. 10 (b) before post processing, which is successfully corrected in Fig. 10 (a) after post processing.

(a) After Post Processing　　(b) Before Post Processing

Fig. 10 Sample Images for the Post-processing Procedure.

5. Conclusion

In this paper, we present a framework of shade segmentation for new AWB.　The framework consists of four steps: sunny-vs.-rainy image classification, sky detection, shade detection, and post processing.　As the first three steps are all classification problems, we use machine learning techniques to build the final classification model.　Different features (color, image capture device parameters, etc.) and classifiers (SVM, AdaBoost) are compared for the three classification problems.　A post-processing procedure is performed to merge neighborhood image regions with similar color appearance, and get refined shade segmentation results.　Extensive experiments on a wide variety of images demonstrate the effectiveness and efficiency of our shade segmentation framework.

References

1) Hsien-Che Lee.　"Introduction to color imaging science".　Cambridge University Press, 2005.

2) M. R. Boutell, C. B. Brown, and J. Luo. "Review of the state of the art in semantic scene classification". Technical Report, 2002.
3) A. Vailaya and A. Jain. "Detecting sky and vegetation in outdoor images". Proc. IS&T/SPIE Symp. Electronic Imaging Science and Technology, 2000.
4) A. C. Gallagher, J. Luo, and W. Hao. "Improved blue sky detection using polynomial model fit". IEEE International Conference on Image Processing, 2004.
5) Bahman Zafarifar, and Peter H. N. de With. "Blue sky detection for picture quality enhancement". Advanced concepts for intelligent vision systems, 2006.
6) Jiebo Luo, and Stephen P. Etz. "A physical model-based approach to detecting sky in photographic images". IEEE Transaction on Image Processing, 2002.
7) Anna Bosche, Andrew Zisserman, and Xavier Munoz. "Scene Classification using a Hybrid Generative / Discriminative Approach". IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008.
8) Rafael C. Gonzalez, and Richard e. Woods. "Digital Image Processing" 3rd Ed. Prentice Hall. 2008
9) V. Vapnik. "The nature of statistical learning theory". Springer-Verlag, New York, 1995.
10) J. Friedman, T. hastie, and R. Tibshirani. "Additive logistic regression: a statistical view of boosting". Annals of Statistics, 2000.
11) Robert E. Schapire, and Yoram Singer. "Improved boosting algorithms using confidence-rated predictions". Proceedings of the eleventh annual conference on computational learning theory, 1998.