Multi-objective optimization based color constancy

Mohammad Mehdi Faghiih, Mohsen Ebrahimi Moghaddam *

Electrical and Computer Engineering Department, Shahid Beheshti University, G.C., Tehran 1983963113, Iran

A R T I C L E   I N F O

Article history:
Received 10 October 2012
Received in revised form 9 September 2013
Accepted 29 November 2013
Available online 24 December 2013

Keywords:
Color constancy
Illuminant estimation
Scene classification
Multi-objective optimization
Particle swarm optimization
Support vector machine

A B S T R A C T

This paper presents a new combining approach for color constancy, the problem of finding the true color of objects independent of the light illuminating the scene. There are various combining methods in the literature that all of them use weighting approach with either pre-determined static weights for all images or dynamically computed weights for each image. The problem with weighting approach is that due to the inherent characteristics of color constancy methods, finding suitable weights for combination is a difficult and error-prone task. In this paper, a new optimization based combining method is proposed which does not need explicit weight assignment. The proposed method has two phases: first, the best group of color constancy algorithms for the given image is determined and then, some of the algorithms in this group are combined using multi-objective optimization methods. To the best of our knowledge, this is the first time that optimization methods are used in color constancy problem. The proposed method has been evaluated using two benchmark datasets and the experimental results were satisfactory in compare with state of the art algorithms.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

The color of real world objects depends on physical characteristics of objects plus the color of light source illuminating the scene containing the objects. Therefore, a blue object under an illuminant appears differently if the color of illuminant changes. But, thanks to the human visual system, a human can recognize same color of an object even if the scene illuminant largely changes. This ability which is called color constancy is based on removing the influence of the scene illuminant on the color of objects.

There are three groups of color constancy algorithms in the literature; see [1] for a complete review. The first group contains algorithms that have a learning phase and use the information gained in this phase to estimate the scene illuminant. The methods of the second group are based on the low-level image features and algorithms that try to reach better results by combining other algorithms or selecting the best algorithm for a given image are placed in the third group. Our previous works [2–4] and also other neural network based approaches [5–7] are instances of the first group. In general, these methods extract some features from a dataset of images and train a neural network using these features and the real scene illuminant as the inputs and targets, respectively. The network is then used to estimate the scene illuminant of an unseen image. For example, Cardei et al. used color histogram of images in rg-chromaticity space to train a multilayer perceptron (MLP) neural network for color constancy [6]. However, the proposed network architecture is complex; it consists of 3600 input nodes, 400 neurons in the hidden layer, 40 neurons in the second hidden layer and 2 output neurons. Another drawback of this approach is that using rg-chromaticity space discards all intensity information while intensity information can help in estimating the illuminant [6]. Other neural network based approaches for color constancy [5,7] also used the color histogram of images in rg-chromaticity space and therefore, they all suffer from the mentioned drawbacks. Gatun based methods [8,9], genetic based algorithms [10], and also probabilistic methods (color by correlation) [11] are subsets of the first group. Examples of the second group are Gray-World [12,13], White-Patch [14], Shades of Grays [15] and Gray-Edge [16] algorithms. The NIS (Natural Image Statistics) [17] and CAS (Classification-based Algorithm Selection) [18] algorithms are two new algorithms of the third group. NIS [17] is based on the fact that the distribution of edge responses of an image can be modeled using two parameters Weibull distribution as follows [19]:

\[ w(s_c) = \frac{\gamma}{\beta} \left( \frac{s_c}{\beta} \right)^{\gamma - 1} e^{-\left( \frac{s_c}{\beta} \right)^\gamma} \]  

where \( s_c \) is the edge responses in a single color channel, \( \beta > 0 \) is the scale parameter of the distribution and \( \gamma > 0 \) is the shape parameter. These parameters are representative for the statistics of the scene when Weibull distribution is fitted on edge responses of an image. The parameter \( \beta \) represents the contrast of the image, and the parameter \( \gamma \) indicates the grain size. Therefore, a higher value for \( \beta \) shows more contrast, whereas a higher value for \( \gamma \) indicates
a smaller grain size (more fine textures). In NIS paper, it is also showed that the Weibull parameters of images are useful for determining the best algorithm for the images. In other word, for a group of images that one algorithm is the best, the Weibull parameters can be clustered together. Therefore, in NIS algorithm, a classifier is learned using Weibull parameters of a dataset of images to predict the best algorithm for new images. CAS [18] is another method that uses decision trees to detect the best algorithm for a given image. The structure of the decision tree in this approach is such that a color constancy algorithm is placed in each leaf and every intermediate node contains a criterion based on an image feature which is determined in the training phase. For a given image, the traversal of tree begins from the root: then in each node, the feature corresponding to that node will be extracted from image and the next node will be selected by comparing the extracted feature and the node criterion. This process continues till reaching a leaf node which contains an algorithm that is expected to be the best algorithm for the given image. The CAS and NIS algorithms also provide a way for combining multiple algorithms. The classifier used in NIS and the decision tree of CAS assign a weight to each pre-selected algorithm and the combination can be done by weighted averaging the output of algorithms.

The problem with the second group of color constancy methods is that all of them are based on specific assumptions. This limits their use to conditions that satisfy the assumptions. For example, Gray-World algorithm assumes that, on average, the world is gray. This assumption is correct only when there is enough large number of different colors in the image. If this is not the case, then the Gray-World algorithm will not work correctly [20]. Algorithms in the third group solve this problem to some extent. But these algorithms are all based on a weighting approach. They use either pre-determined static weights or dynamically computed weights combining multiple algorithms. The problem with weighting approach is that due to the inherent characteristics of color constancy methods (every algorithm is based on a specific assumption and no algorithm can be thought as universally better than the others), finding suitable weights for combination is a difficult and error prone task. Also, it should be considered that while combining multiple algorithms for a given image, the result is not as good as expected if a good algorithm is combined with a bad one. In other word, it is important to combine the algorithms that have good performance on the input image. Current algorithms of the third group (e.g. CAS and NIS) do not take this into account. They only assign a weight to each pre-defined algorithm and combine all of them. Although the algorithm may assign a near zero weight to bad algorithms, a too bad algorithm with a near zero weight still may decrease the accuracy of the combination output. Also, considering the bad algorithms in the process of determining combination weights increases the complexity of this process and as a result, the algorithm may not be able to determine the weights precisely.

In this paper, a new combining approach for color constancy algorithms is proposed which does not need explicit computation of weights and also always tries to combine those algorithms that have good performance on the given image. For a given image, the method consists of two steps. In the first step, the best group of color constancy algorithms for that image is determined using a classifier based on image features and in the second step, some of the algorithms in this group are combined by a multi-objective optimization method to estimate the scene illuminant. In this way, the proposed method always tries to combine good algorithms for the input image and as a result, the overall performance increases.

The rest of paper organized as follows: Section 2 describes color constancy and reflection model. Proposed approach is discussed in Section 3 and finally, in Section 4, the proposed approach is evaluated using two benchmark color constancy datasets.

2. Color constancy

An image taken by a digital camera under the assumption of Lambertian reflectance can be seen as function $f(x, y)$ depending on three physical factors: the spectral power distribution of scene illumination $e(\lambda)$, the surface spectral reflectance $s(\lambda)$ and the camera sensitivity function $\rho(\lambda)$. The sensor responses at pixel with coordinate $(x, y)$ with this notation can be formulated as the following [17]:

$$f(x, y) = \int_\omega e(\lambda) s(x, y, \lambda) \rho(\lambda) d\lambda.$$  

Here $\omega$ is the visible spectrum. The color constancy is equivalent to estimate $\hat{e}$ (the observed color of the light source) using the following equation [17]:

$$\hat{e} = \frac{e_R}{e_G} = \int_\omega e(\lambda) \rho(\lambda) d\lambda.$$  

After estimating the color of the light source, the image colors can be corrected using this estimate to produce a new image of the scene as if it was taken under a perfect white light (i.e., $\lambda = 1/\sqrt{3}$, $1/\sqrt{3}$, $1/\sqrt{3}$). Under the assumption of diagonal model [8], the correction is simple; first, the image values are divided by $\hat{e}$ and then, the white light is multiplied to the image. This procedure corrects the image colors but its intensity may differ from the original intensity. Therefore, some methods have been proposed in order to preserve the original image intensity such as proposed method in [21].

In general, both values of $e(\lambda)$ and $\rho(\lambda)$ are unknown and therefore, the problem of color constancy is an under-constrained problem and it is not possible to solve this problem without further assumptions. Hence, most color constancy algorithms assume that an hypothesis is met while trying to estimate $\hat{e}$. As an example, a popular color constancy algorithm named Grey-World [12] is based on the assumption that the average reflectance in a scene is achromatic. The following equation shows the formal representation of this assumption:

$$\frac{\int s(\lambda, x) dx}{\int dx} = k$$

Using this assumption, the color of light source can be computed as follows:

$$\int f(x) dx = \frac{1}{\int dx} \int_\omega \int e(\lambda) s(x, \lambda) \rho(\lambda) d\lambda dx$$

$$= \int_\omega \frac{e_R}{e_G} \left( \frac{\int s(\lambda, x) dx}{\int dx} \right) \rho(\lambda) d\lambda.$$  

$$= k \int_\omega e(\lambda) \rho(\lambda) d\lambda = k \hat{e}$$

where $k$ is a constant that is chosen such that the illuminant color $\hat{e}$ has unit length.

White-Patch [14] is another common color constancy algorithm which assumes that the surface in the scene with highest luminance (white patch) reflects maximally and uniformly over the spectrum. With this assumption, the color of light source can be approximated by the color of brightest patch in the image [20].

It is shown in the reference [15] that the Gray-World and White-Patch algorithms are two different instantiations of a more general color constancy algorithm based on Minkowski norm. This
algorithm which is called Shades of Gray can be formulated using the following equation:

$$\left( \frac{\int (f(x))^p \, dx}{\int dx} \right)^{1/p} = k, \bar{e}$$

(8)

In the above equation $p$ is Minkowski norm; by setting $p = 1$, the Eq. (8) is equal to Gray-World assumption and for $p = \infty$, it is the equivalent to White-Patch algorithm.

The Gray Edge algorithm [16] that uses higher-order statistics of image is based on the assumption that the average of reflectance differences in a scene is achromatic:

$$\frac{\int |s^2_\omega (\lambda, x)| \, dx}{\int dx} = k$$

(9)

The $s^2_\omega$ indicates the spatial derivative that is defined as the convolution of the image with the derivative of a Gaussian filter with scale parameter $\sigma$ [22]:

$$\frac{\partial s^{s+t}}{\partial x} \ast \sigma = \frac{\partial s^{s+t}}{\partial x} \ast \sigma$$

(10)

where $*$ denotes the convolution and $s + t$ is equal to the order of the derivative.

The computation of scene light source color with Gray-Edge algorithm is as follows:

$$\frac{\int |f^p (x)| \, dx}{\int dx} = \frac{1}{\int dx} \int e(\lambda) \, |s^2_\omega (\lambda, x)| \, \rho(\lambda) \, d\lambda, dx$$

$$= \int e(\lambda) \left( \frac{\int |s^2_\omega (\lambda, x)| \, dx}{\int dx} \right) \rho(\lambda) \, d\lambda$$

$$= k \int e(\lambda) \rho(\lambda) \, d\lambda$$

(11)

where $|f(x)| = (|R(x)|, |G(x)|, |B(x)|)^T$.

It also exists a more general framework [16] based on Minkowski norm that represents both the well-known methods like Eq. (8) and methods based on higher-order statistics:

$$\left( \frac{\int |f^p (x)| \, dx}{\int dx} \right)^{1/p} = k, \bar{e}$$

(14)

In this equation, $n$ is the order of the derivative, $\sigma$ is the scale parameter of Gaussian filter, the division by $\int dx$ is incorporated into the constant $k$ and also it is assumed that the $p$th Minkowski norm of the $n$th order derivative of the reflectance (PMNDR) in a scene is achromatic:

$$\left( \frac{\int |f^p (x)| \, dx}{\int dx} \right)^{1/p} = k$$

(15)

Eq. (14) is a framework that represents the Gray family of color constancy algorithms. By changing parameters of this equation, it is possible to generate many different color constancy algorithms. For example:

1. $\bar{e}_{0.1, 0}^p$ is equal to Gray-World algorithm.
2. $\bar{e}_{0.1, 0}$ is equivalent to White-Patch algorithm.
3. $\bar{e}_{0.1, 0}$ is the Shades of Gray algorithm.
4. $\bar{e}_{0.1, 0}$ is the family of zero order color constancy algorithms.
5. $\bar{e}_{1.1, 0}$ is the family of first order color constancy algorithms.
6. $\bar{e}_{2.1, 0}$ is the family of second order color constancy algorithms.

### 3. Proposed approach

The proposed method is a combination approach that tries to always combine some Gray algorithms with better performance for the given image in compare to other algorithms. This motivation of this method is based on the observation that in general, when a Gray color constancy algorithm of order $n$ is the algorithm with the best performance on a natural image, the next $k$ best Gray algorithms for that image is also of order $n$. In other word, instead of saying that the best Gray algorithm for a given image is of order $n$, we can say that for the given image the $k$ best Gray algorithms have order $n$. For further clarification, we have done an experiment using the Gray-Ball color constancy dataset [23] (a popular color constancy dataset that contains 11346 natural images). In this experiment, the total number of 75 different Gray algorithms (25 zero order, 25 first order and 25 second order algorithms) have been generated by choosing random values for the parameters of Eq. (14) as follows (the ranges for $p$ and $\sigma$ are chose):

Zero order ($\bar{e}_{0, 0}^{p, \sigma}$):

$$\begin{align*}
\frac{p & = \text{random integer} \in [1, 13]}{
\sigma & = \text{random integer} \in [0, 11]}.
\end{align*}$$

First order ($\bar{e}_{1, 1}^{p, \sigma}$):

$$\begin{align*}
\frac{p & = \text{random integer} \in [1, 13]}{
\sigma & = \text{random integer} \in [0, 11]}.
\end{align*}$$

Second order ($\bar{e}_{2, 2}^{p, \sigma}$):

$$\begin{align*}
\frac{p & = \text{random integer} \in [1, 13]}{
\sigma & = \text{random integer} \in [1, 11]}.
\end{align*}$$

In the parameter adjustment, we tried to select parameters in such a way that all cases of related works in the literature were considered. For example, the maximum value of $\sigma$ that has been used in literature is 7 [16] and therefore, the range for random values of this parameter has been chosen as [1,11] to cover all cases. The same manner has been used for random values of $p$ and its range has been chosen as [1,13].

Then, the performances of all 75 Gray algorithms have been evaluated on the dataset. Then the dataset was split into three subsets: zero order subset contains the images that the order of the best algorithm for all of them is zero, first order subset contains the images that the order of the best algorithm for all of them is one and second order subset contains the images that the order of the best algorithm for them is two. Fig. 1 shows the percentages of dataset images that fall into each subset. Then, for each subset, the percentages of images in the subset that the orders of two, three, four or five best algorithms for them are the same have been calculated and depicted in Fig. 2.

As it can be seen in Fig. 2, when a zero order algorithm is the best algorithm for an image, on average, in 99.5% of times the second best algorithm for that image is also a zero order algorithm. This value for the first order and the second order algorithms is about 90%. Also, the percentages of images in zero order subset that the three, four and five best algorithms for them have the same order.
are 99.1%, 98.3% and 97.9%, respectively. For first and second order algorithms, these values are somewhat lower. This is because that the first and second order algorithms are very similar in nature. They are all based on the higher order statistics of image and therefore, when a non-zero order (first and second order) algorithm is the \( k \)th best algorithm for an image, the \((k+1)\)th best algorithm for that image is with a high probability also a non-zero order algorithm as shown in Fig. 3.

As it is shown in Fig. 3, when a non-zero order (first and second order) algorithm is the best algorithm for a set of images, on average, the second best algorithm for 98.1% of the set is also a non-zero order. Also, the percentages of images that the three, four, and five best algorithms for them are non-zero order algorithms are 99.1%, 98.3% and 97.9%, respectively.

To further approve the motivation, the same test has also been done using another popular color constancy dataset called the Color Checker dataset [24] (this dataset contains 568 indoor and outdoor high quality images) and the results are shown in Figs. 4 and 5. As it can be seen, the results are similar to Gray-Ball dataset results.

From Fig. 2 to Fig. 5 it can be concluded that in general, all five best Gray algorithms for each natural image are either zero order or all of them are non-zero order algorithms. So, in order to choose best Gray algorithms for combination, we must determine the best order and then select some of algorithms with that order.
edge direction histogram, Weibull parameters, number of colors, biologically inspired features and wavelet Local Energy Histogram.

3.1.1.1. rg-chromaticity color histogram. One of the most widely used image descriptors is color histogram [5–7,18]. It represents the color distribution of the image and encodes several useful properties that make it a robust visual feature. It is also invariant and robust with respect to the geometric transformation of the original image like rotation and scale. Using the full color histogram in RGB space as a feature is practically impossible because it requires 256×3 bins. Therefore, the image values must be transformed into the rg-chromaticity space before computing the color histogram. In this way, the number of bins required for color histogram decreases to 2562. Also, the rg-chromaticity color histogram is usually sparse and therefore it is better to use a quantized color histogram as a part of feature vector. In the proposed approach, the color histogram in rg-chromaticity space is quantized by uniformly dividing each color axis into 32 intervals. So, the rg-chromaticity space is subdivided into 1024 smaller squares and each of the original colors is mapped to the square which it falls into. This means that the color histogram part of feature vector is a vector with the size equal to 1024.

3.1.1.2. Edge direction histogram. Edges in images with different types of scenes are different and they have valuable information for image classification. For example, strong edges can be found in buildings, roads, and other man-made structures which usually have definite direction pattern. On the other hand, the objects in the pictures of natural scenes usually have no clear structure and do not show a specific pattern and also, these pictures usually do not contain strong edges. Edge direction histogram is a good tool for determining the edge structures within an image and therefore allows us to distinguish between different image classes. In order to obtain edges, the derivative of a Gaussian filter with \( \sigma = 1 \) is convolved by the luminance image in both the x and y directions (\( G_x, G_y \)). The edge orientation at edge position \((x, y)\) is then computed using the following equation:

\[
\theta(x, y) = \arctan \frac{G_y(x, y)}{G_x(x, y)}
\]

The 36 bins edge direction histogram is then obtained by quantizing the orientations into intervals of 5°. Also, to ensure that only sufficiently strong edges are used in computing the histogram, only the orientations belonging to edges with magnitude above a given threshold are considered. The threshold is also experimentally chosen to be equal to 0.6.

3.1.1.3. Weibull parameters. Natural image statistics (distribution of edge responses) are indicative for type of scene [26] and Geusebroek and Smeulders [19] showed that the natural statistics of an image can be modeled using two parameters Weibull distribution shown in Eq. (1). As mentioned in Section 1, when Weibull distribution is fitted on edge responses of an image, its parameters represent the natural statistics of image.

The Weibull parameters should be computed for derivative of each channel of RGB images separately. But, since RGB channels are highly correlated [17] images should be mapped to a decorrelated color space before computing Weibull parameters. To this end, the decorrelated opponent color space is used as follows:

\[
O_1 = \frac{R - G}{\sqrt{2}}
\]

\[
O_2 = \frac{R + G - 2B}{\sqrt{6}}
\]

\[
O_3 = \frac{R + G + B}{\sqrt{3}}
\]

3.1.1.4. Number of colors. The color range of an image can be represented by the number of distinct colors in the image. This parameter is chosen to be in the feature vector of proposed approach because the zero order algorithms are all base on the Gray-World assumption and this parameter is an indication of whether the Gray-World assumption holds true for the given image or not [20]. If an image contains many different colors, then the average color is likely to be a gray value. While computing this feature, the RGB color channels have been quantized to remove small variations in the color appearance and also decrease the influence of noise. Fig. 6 shows the maximum number of possible colors in an image based on the number of bits in each color channel. If we consider 8 bits in each color channel, the maximum number of possible colors is \( (2^8)^3 = 16777216 \). The possible number of colors for 1, 2, 3, 4, 5 or 7 bits are 1, 8, 64, 512, 4096, 32768, 262144 and 2097152, respectively. As you can see, the number of colors for 7 and 8 bits is very large and among this large number of colors, many of them have small variations and the human visual system cannot discriminate between them and also are sensitive to noise. On the other hand, using small number of bits reduces the influence of noise and also discards many discriminative colors. Considering this trade-off, it seems that choosing 6 bits per pixel is a reasonable choice. Therefore, the quantization has been done by considering only the six most significant bits for each channel. Thus, the maximum number of different colors that can be discriminated is \( (2^6)^3 = 262144 \).

3.1.1.5. Biologically inspired features. Systems based on biologically inspired features try to mimic the process of visual cortex in recognition tasks by simulating the C1 and S1 units of visual cortex [25,28,29]. The C1 units correspond to complex cells in the visual cortex and the S1 units correspond to simple cells in S1 layer of the visual cortex [28]. C1 units use a maximum operator and keep the max response of a local area of S1 units from the same orientation and scale. In order to represent the S1 units, Gabor functions are used because these functions are similar to the receptive field.

![Fig. 6. The maximum number of possible different colors based on the number of bits in each color channel.](image-url)
profiles in the mammalian cortical simple cells [25]. The Gabor mother function is \( f(x, y) = \exp(x_0 + y^2y_0^2)/(2\sigma^2) \times \cos(2\pi x_0/\lambda) \), where \( x_0 = x\cos(\theta) + y\sin(\theta), y_0 = -x\sin(\theta) + y\cos(\theta) \). The ranges of \( x \) and \( y \) determine Gabor filters scale and \( \theta \) controls the orientations. As in [25], a pyramid of Gabor filters with eight scales is used in the proposed approach. The sizes of filters are from \( 7 \times 7 \) pixels to \( 21 \times 21 \) pixels with a step of two pixels. Four values (0°, 45°, 90° and 135°) are considered for \( \theta \) which results in \( 8 \times 4 = 32 \) Gabor filters (the values for the size of filters and also the different values of \( \theta \) are chosen to be the same as the reference [25] because this reference showed the effectiveness of these values in classification of natural images). By applying these Gabor filters on the initial input image 32 feature maps are obtained for \( S_1 \) units. Then, by applying a maximum operation over two adjacent scales of \( S_1 \) units with matching orientations, 16 feature maps for \( C_1 \) units are obtained. Finally, the normalized histogram of each \( C_1 \) unit feature map with total number of 50 bins is used as part of feature vector. Hence, the total length of biologically inspired part of feature vector is \( 16 \times 50 = 800 \).

### 3.1.1.6. Wavelet local energy histogram

The wavelet decomposition extracts the information about the textures and structures within the image [30–32]. The multi resolution wavelet decomposition is a process that applies to the LL (low pass filtered version of the image) sub-band in a recursive manner. It can be repeated until the LL sub-band cannot be further processed or until a specified number of wavelet decomposition is reached. In each level of the wavelet decomposition, three high pass band is created, so with considering the LL band, the total number of bands after a multi resolution wavelet decomposition with \( L \) levels is equal to \( 3^L + 1 \).

To extract wavelet features in the proposed approach, the three level wavelet decomposition has been applied on the luminance image with Daubechies 1 (db1) filter bank, producing a total of 10 bands. Local Energy Histogram (LEH) [32] of each sub-band was used in feature vector. The LEH extracts the local (Norm-1) energy features within \( S \times S \) neighborhoods in each sub-band. Typically, in the \( j \)th high pass sub-band of size \( \Omega_j \) at the \( i \)th scale, the local energy features can be defined by [32]:

\[
E_{Loc}^{ij}(l, k) = \frac{1}{S^2} \sum_{i=1}^{S} \sum_{k=1}^{S} \left| w_{ij}(l + u - 1, k + v - 1) \right|
\]

(20)

where \( l \leq i, k \leq \Omega_j^l - S + 1 \) and \( w_{ij}(m, n) \) is the wavelet coefficient at location \( (m, n) \) in the sub-band. The local energy features in the low pass sub-band, denoted by \( E_{Loc}^L \), for clarity, are also extracted in the same manner according to Eq. (20).

All the above local energy features are non-negative and the average amplitude of the local energy values increases almost exponentially with the scale \( i \). Therefore, the \( E_{Loc}^{ij} \) should be regularized by multiplying the factor \( 1/2^i \) in order to make a uniform measure for those local energy features at different scales [32]. Because the average amplitude of the local energy feature values in low pass sub-band is much higher than high pass sub-bands, the regularization factor for \( E_{Loc}^{ij} \) considered as 1/4^i.

For a particular wavelet sub-band with \( M \) local energy features \( (e_1, e_2, \ldots, e_M) \), the LEH which is capable of modeling the probability density function of the local energy features, can be computed by taking the normalized histogram of \( e_i \) where \( 1 \leq i \leq M \) [32]. Considering 10 sub-bands and the total number of 100 bins in LEH for each sub-band, the length of wavelet part in feature vector is equal to 1000.

#### 3.1.2. Dimension reduction

As shown in Fig. 7, the dimension of obtained feature vector in previous section is very high. It consists of total 2909 numbers: 1024 positive numbers for rg-chromaticity color histogram, 36 numbers for edge direction histogram, 48 numbers for Weibull parameters, one positive number for number of colors, 800 numbers for biologically inspired features and 1000 numbers for LEH of wavelet decomposition.

Such a long length feature vector increases the complexity of the classifier and also may decrease the classification accuracy due to the possible low information dimensions in the feature vectors. Consequently, the length of the feature vector must be decreased by extracting the most discriminative parts before training the classifier. Many dimension reduction algorithms exist in the literature, including locally linear embedding (LLE) [33], ISOMAP [34], and Discriminative and Geometry Preserving Projection (DGPP) [25]. A good dimension reduction algorithm for the task of classification should model both the intra-class geometry and interclass discrimination. It must make the distances between interclass samples as large as possible while keeping the distances between intra-class samples as small as possible. The DGPP algorithm is a new dimensionality reduction method that can precisely do these tasks. DGPP is used in the proposed approach because it outperforms previous methods in classification accuracy while using same classifier and also never meets the under-sampled problem when training samples are insufficient [25].

In summary, for \( n \) samples \( x_i, i = 1 \ldots n \) in the high dimensional space \( R^d \), DGPP maps the input samples to the low dimensional space \( R^t \) using a linear mapping defined by a projection matrix \( U \in R^{d \times t} \) as follows:

\[
X = [x_1, \ldots, x_n] \in R^{d \times n}
\]

\[
Y = UX \in R^{t \times n}
\]

(21)

Each column of \( Y \) is the low dimensional representation of an input sample; i.e. \( y_i = U^T \hat{x} \in R^t \).

In DGPP, there are \( c \) classes and the \( i \)th sample \( x_i \) in the high dimensional space is associated with a class label \( m_i \in \{1, 2, \ldots, c\} \). The projection matrix \( U \) is calculated by minimizing the average weighed pairwise distance between samples in different classes and minimizing the average weighed pairwise distance between samples in an identical class. Also, to implement the local geometry preservation in matrix \( U \), the matrix \( U \) is calculated in DGPP assuming that each sample can be reconstructed by the samples within the same class. Note that the DGPP is not a feature selection algorithm; it projects the feature vectors onto another coordinate system with lower dimensions and hence, it is not possible to determine which parts of feature vectors are more important. See [25] for further details.

#### 3.1.3. The classifier: support vector machine

The support vector machine (SVM) is a popular pattern classification method used in recent years [35]. It is a statistically robust learning method based on the minimization of structural risk. The SVM is for binary classification; it trains a classifier by finding

![Fig. 7. The high dimensional feature vector for each image. The dimensionality of this feature vector must be reduced by DGPP before training the classifier.](image-url)
an optimal separating hyper-plane which maximizes the margin between two classes of data in the kernel induced feature space. In the case of multi-class classification, one approach is to reduce the single multi-class problem into multiple binary classification problems. A common method for such reduction is building total number of \( C(C-1)/2 \) binary classifiers which distinguish between every pair of classes (pairwise approach). The classification is then done by a max-wins voting strategy: every classifier assigns the instance to one of the two classes, then the vote for the assigned class is increased by one vote, and finally the class with the highest votes determines the instance classification.

The SVM has good generalization ability and obtains top-level performance in different applications. Therefore, the proposed approach uses a multi-class SVM classifier to predict the order of the best group of Gray algorithms for a given image. The method only considers the Gray algorithms of orders up to two and therefore, the classifier must classify the images into three classes (the first schema) or two classes (the second schema). Each training image has a feature vector which is the low dimensional vector obtained by applying the DGPP on the high dimensional vector containing the mentioned features in Section 3.1.1. In the first schema, the class \( c_i \) of each training image is set to the order of the Gray algorithm with the maximum performance on the image and the proposed method uses the pairwise approach for SVM multi-class classification. Consequently, \( 3(3-1)/2 = 3 \) binary classifiers must be constructed and each classifier must be trained by data from two different classes. In the second schema, the proposed method uses a single binary SVM classifier with the class \( c_i \) of each training image \( i \) is set to 0 or 1: the 0 class contains images with the best Gray algorithm of zero order and the 1 class contains images with the best Gray algorithm of non-zero order (first order and second order).

3.2. Algorithm combination

Once determining the best group of Gray algorithms for the given image, the next step is to combine the output of some of the algorithms of that group to achieve a new estimate for the color of light source. To this end, the easiest method is to take the average of the illuminant estimates over all algorithms. A straightforward extension is to take the weighted average of the estimated illuminants. Determining weights in this approach is a challenging task. Due to the inherent characteristics of color constancy algorithms, the performance of assigning static weights for each algorithm is poor and also finding optimum dynamic weights is a difficult task. In this paper, a new optimization based approach is proposed that tries to combine the Gray family of color constancy algorithms without weight assignment. The main idea is to combine some Gray algorithms by finding an illuminant vector that satisfies the assumptions of theses algorithms simultaneously.

Each Gray algorithm assumes that a PMNDR of the image is achromatic (Eq. (15)) and estimates the scene illuminant using Eq. (14). The result of this approach is an obvious fact: the corresponding PMNDR of the corrected image with the illuminant estimated by a Gray algorithm is always achromatic. Considering Eq. (14), it is clear that the Gray algorithms estimate the scene illuminant by assuming that the PMNDR of the un-corrected image is equal to the product of the scene illuminant vector by a constant. Consequently, after correcting the image (which can be done by dividing the image values by the scene illuminant in each band and multiplying the results by a white light source), the PMNDR of the corrected image is achromatic (a gray vector) because it is equal to the product of a constant by a white light source. Considering this fact, it is possible to define an equivalent optimization based color constancy method for each Gray algorithm. To this end, the fitness function can be the distance between the corresponding PMNDR of the color corrected version of the given image and the gray vector (a vector with all three elements equal). This way, the optimization method converges to an illuminant vector that when used to color correct the given image, the PMNDR of the corrected image is achromatic (which is the result of the corresponding Gray algorithm).

Using optimization method instead of a Gray algorithm does not have any advantage: its result is the same as the Gray algorithm while it is more complex. But, it is possible to define a new approach for combining color constancy algorithms using optimization methods. The idea is to use multi-objective optimization methods to find an illuminant vector that minimizes the distance between the PMNDR and the gray vector for multiple Gray algorithms simultaneously. The benefit of this new approach is that the method finds the result of combination directly and does not need any weight assignment.

3.2.1. Multi-objective optimization

The multi-objective optimization or Pareto optimization is the process of simultaneously optimizing two or more conflicting objectives subject to certain constraints. In mathematical terms, the problem can be written as:

\[
\min f(\vec{x}) = \left[ f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x}) \right]
\]

Subject to:

\[
g_i(\vec{x}) \leq 0 \quad i = 1, 2, \ldots, m \tag{23}
\]

\[
h_j(\vec{x}) = 0 \quad j = 1, 2, \ldots, p \tag{24}
\]

where \( \vec{x} = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^n \) is the vector of decision variables, \( f_i : \mathbb{R}^n \rightarrow \mathbb{R}, \quad i = 1, \ldots, k \) are the objective functions and \( g_i, h_j : \mathbb{R}^n \rightarrow \mathbb{R}, \quad i = 1, \ldots, m, j = 1, \ldots, p \) are the constraint functions of the problem.

The solution to this problem is to find the Pareto-optimal set: the set of vectors of decision variables that each one is non-dominated with respect to \( x \). A vector of decision variables \( \vec{x} \in \mathbb{R}^n \) is non-dominated with respect to \( x \) if another vector \( \vec{x}' \in \mathbb{R}^n \) does not exist such that \( f(\vec{x}') \) dominates \( f(\vec{x}) \). Given two vector \( \vec{x}, \vec{y} \in \mathbb{R}^n \), it is said that \( \vec{x} \preceq \vec{y} \) if \( x_i \preceq y_i \) for \( i = 1, \ldots, n \) and \( \vec{x} \) dominates \( \vec{y} \) (denoted by \( \vec{x} < \vec{y} \)) if \( \vec{x} \preceq \vec{y} \) and \( \vec{x} \neq \vec{y} \). The Pareto front set is the set of fitness vectors of the Pareto-optimal set members. Fig. 8 shows a particular case of the Pareto front in the case of two objective functions.

The solution set of a problem with multiple objectives does not consist of a single solution as in global optimization. Instead, the aim of the multi-objective optimization is to find a set of different solutions (the so-called Pareto optimal set) while following three main goals [36]:

![Fig. 8. The Pareto front of a set of solutions in a two objective space.](image)
• Maximize the number of elements in Pareto-optimal set.
• Minimize the distance between the found Pareto front and the true (global) Pareto front (assuming its location is known).
• Maximize the spread of solutions found in order to have a distribution of vectors as smooth and uniform as possible.

Various methods exist for finding the Pareto-optimal set that one of them (which is used in this paper) is Particle Swarm Optimization (PSO) [37].

3.2.2. Particle swarm optimization

PSO is a population-based search algorithm based on the simulation of the social behavior of birds within a flock. It is a very popular global optimizer, mainly in problems in which the decision variables are real numbers [38]. The popularity of PSO is because of two key aspects:

• PSO is relatively simple and therefore its implementation is straightforward.
• PSO has been found to be very effective in a wide variety of applications, being able to produce very good results at a very low computational cost [36].

PSO optimizes a problem by having a population (swarm) of candidate solutions (particles) and moving them around in the search-space according to simple mathematical formula over the particle position and velocity. The movement of each particle is influenced by its local best known position and is also guided toward the best known position in the search-space which is updated as better position is found by other particles. This is expected to move the swarm toward the best solutions which are the Pareto-optimal members.

The formal definition of the PSO optimization is as follows: a set of N particles may be considered as population $P_t$ in the generation $t$. Each particle $i$ has a position defined by $x_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,d}\}$ and a velocity defined by $v_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,d}\}$ in the variable space $S$. In generation $t+1$, the velocity and position of each particle $i$ is updated as below:

$$v_{i,j,t+1} = \omega v_{i,j,t} + c_1 R_1(p_{i,j,t} - x_{i,j,t}) + c_2 R_2(p_{j,t}^g - x_{i,j,t}) \tag{25}$$

$$x_{i,j,t+1} = x_{i,j,t} + v_{i,j,t} \tag{26}$$

where $j = 1, \ldots, n, \omega$ is the inertia weight of the particle, $c_1$ and $c_2$ are two positive constants called cognitive and social parameter, respectively; and $R_1$ and $R_2$ are realizations of two independent random variables that assume the uniform distribution in the range $[0, 1]$.

The $p_{j,t}^g$ in Eq. (25) is the position of the global best particle in the population which guides the particle to move toward the optimum. $p_{i,j,t}$ is the best position that particle $i$ could find so far that is like a memory for the particle and is updated in each iteration. The $\omega$ is a positive parameter called inertia weight and is employed to control the impact of the previous history of the velocities on the current velocity, thus to influence the trade-off between the local and global exploration abilities of the particle [38].

3.2.3. Multi-objective particle swarm optimization

The important part of multi-objective PSO is to determine the best global particle $p_{i,t}^g$ for each particle $i$ of the population. The global best particle in single objective PSO is determined easily by selecting the particle that has the best position. But, in multi objective PSO, there is a set of Pareto-optimal solutions (named external archive) as the optimum solutions and each particle in population should select one of the members of external archive as its global best particle. Fig. 9 shows the general structure of a multi objective PSO algorithm.

The problem of selecting members from the external archive has been addressed through the determination of measures that evaluate the quality of each archive member based on density estimators. Using such measures, archive members that promote diversity can be selected. The most commonly used density estimators are the Nearest Neighbor Density Estimator [39] and Kernel Density Estimator [40]. Both measures provide estimations regarding the proximity and number of neighbors for a given point.

After a member of the external archive is selected as the global best particle for each particle, the velocity and position of that particle can be updated using a similar approach to the single objective PSO and also, the evaluation of the particle new position can be done using the optimization problem fitness function.

After updating each particle, the external archive can be updated as follows: a new solution is included in the archive if it is non-dominated with respect to all its members. If some members are dominated by the new solution, then they are usually deleted from the archive.

In order to use PSO to solve a specific problem, the structure of each particle (swarm member) and the fitness function that evaluates the quality of each particle should be determined based on the characteristics of the problem. In our case, each particle is an illuminant vector which is a vector with three elements and the pseudo code of fitness function is also shown in Fig. 10. The inputs to the fitness function are the given image, the parameters of the selected algorithms and the particle to be evaluated. Generally, in a multi-objective optimization, the fitness function should return a vector containing the fitness values for all objectives. The objectives in the proposed approach, as explained in Section 3.2, are the PMNDR of the algorithms and the optimization should seek an illumination vector that minimizes the distance between the PMNDR of each algorithm and the gray vector (which is a vector with identical elements i.e. $[5.5.5]$ or $[x \times x \times]$). Hence, the function starts with the color correction of the given image and then continues with the calculation of the PMNDR for all three color bands of the correct image for each algorithm. Then, the line 6 of the fitness function is for calculating the distance between the PMNDR (vector $M$) of algorithms and the gray vector. It sums up the absolute pairwise difference between the three elements of vector $M$. The result of this operation is minimum (zero) if and only if the vector $M$ be a gray vector and the larger distance between the PMNDR and the gray vector increases this value which is exactly the behavior we need. As you can see, there is no weights (and absolutely there is no need for any weights) in the fitness function. Using this fitness function, the multi-objective PSO optimization finds the Pareto-optimal

![Fig. 9. The structure of multi-objective PSO algorithm.](image-url)
The fitness function for multi-objective PSO optimization:

**Input:** the given image, vectors \( \mathbf{g}, \mathbf{d} \) and \( \mathbf{a} \) containing the \( n, p \) and \( a \) of the selected algorithms for combination, and the particle \( P \) which is an illuminant vector

**Output:** the vector \( F \) containing the fitness for each selected algorithm

1. Color correct the given image using \( P \)
2. for \( i = 1 \) to number of algorithms
3. Calculate the PMNDR of the red, green and blue color channels of the corrected image using the following formulas with \( \mathbf{g}, \mathbf{d}, \mathbf{a} \) as parameters and store the results in vector \( M \):

\[
\begin{align*}
M_r &= \left( \int \frac{\partial^n f_{\text{red}}(x)}{\partial x^n} \, dx \right)^{1/p} \\
M_g &= \left( \int \frac{\partial^n f_{\text{green}}(x)}{\partial x^n} \, dx \right)^{1/p} \\
M_b &= \left( \int \frac{\partial^n f_{\text{blue}}(x)}{\partial x^n} \, dx \right)^{1/p}
\end{align*}
\]

4. Normalize the vector \( M \).
5. Calculate the fitness value of the particle with the following formula and store the result in \( F_i \)
6. \( F_i = |M_r - M_g| + |M_g - M_b| + |M_r - M_b| \)
7. end for
8. return \( F \)

*Fig. 10. Pseudo code of the fitness function for PSO optimization.*

set of illuminant vectors which minimizes the distance between the corresponding PMNDR of the corrected image for all selected algorithms simultaneously.

Having Pareto-optimal set, the next step is to select an illuminant as the estimate for the scene illuminant of the given image. Note that the details of different methods of the PSO optimization are outside the scope of this paper and hence, they are not discussed here. The interested reader can refer to [36] for a valuable review.

### 3.2.4. Illuminant selection

In order to select an estimate of the scene illuminant from the Pareto-optimal set, the CaC [41] methods is used. This method is based on the basic color term categories [42]. These categories were inferred from a large anthropological study based on speakers of 20 different languages and specific documentation from a further 78 languages. In this study, it is concluded that the universal basic color terms defined in most evolved languages are white, black, red, green, yellow, blue, brown, purple, orange, pink, and gray. These categories constitute prior knowledge that is useful for general image understanding and therefore, the CaC method uses these categories as anchors to select an illuminant from a feasible set of illuminants. This method is based on the category hypothesis: Feasible illuminants can be weighted according to their ability to anchor the colors of an image to basic color categories [41]. Based on this hypothesis, the CaC method assigns a probability to each feasible illuminant in such a way that the illuminant which provides a corrected image whose colors fall into basic color categories, gets the maximum probability. See the original paper [41] for the details of the probability assignment.

### 3.3. The proposed method in summary

*Fig. 11* shows the overview of the training phase in the proposed method (the first schema). In summary, the training phase for the first schema of the proposed method consists of the following steps:

1. Choose \( n_{\text{alg}} \) Gray algorithms.
2. For each image \( \text{img}_i, i \in [1, N] \) and Gray algorithm \( \text{alg}_j, j \in [1, n_{\text{alg}}] \), determine the performance \( e_{i,j} \) (angular error) of \( \text{alg}_j \) on \( \text{img}_i \). \( N \) is the total number of training image

3. Form the label vector \( L \in \mathbb{R}^{N \times 1} \) with each element \( L_i \) equal to the order of the algorithm with the highest performance on \( \text{img}_i \) as follows:

\[ L_i = \arg\min_{\text{alg}_j} \{ e_{i,j} \} \]

4. Train the classifier \( C \) using the following steps:

   (a) Extract the high dimensional feature vector \( f^H \in \mathbb{R}^{H \times 1} \) for each image \( \text{img}_i \) using the process defined in Section 3.1.1. Build the matrix \( F^H \in \mathbb{R}^{H \times N} \) such that each column is equal to \( f^H \). \( H \) is the dimension of high dimensional feature vector (2509 in this paper).

   (b) Apply the DGPP on the matrix \( F^H \) to obtain the projection matrix \( U \in \mathbb{R}^{H \times L} \) and the matrix \( F^L \in \mathbb{R}^{L \times N} \) (\( F^L = UF^H \)). Each column of \( F^L \) of matrix \( F^L \) is equal to the low dimensional representation of feature vector for a training image. \( L \) is the dimension of low dimensional (reduced) feature vector.

   (c) Train the classifier \( C \) with the matrix \( F^L \) and the vector \( L \) as the inputs and labels, respectively.

After training phase, the normalized scene illuminant for a given image can be estimated using the following steps (as shown in *Fig. 12*):

1. Extract the high dimensional feature vector \( f^H \in \mathbb{R}^{H \times 1} \) for the given image using the process defined in Section 3.1.1.

2. Use the obtained projection matrix \( U \) in the step 4 (b) of the training phase to obtain the \( f^L \in \mathbb{R}^{L \times 1} \) (the low dimensional representation of \( f^H \)) by the following formula:

\[ f^L = UF^H \]

3. Feed the \( f^L \) to the trained classifier to determine the best order of Gray algorithms for the image and name it as \( o \).

4. Use the PSO optimization with the given image and the parameters of the algorithms of order \( o \) to find the Pareto-optimal set of illuminants as describe in Section 3.2.

5. The normalized estimate of the scene illuminant can be obtained by applying the selection algorithm described in Section 3.2.4 on the Pareto-optimal set of illuminants.

For the second schema, the proposed method steps are identical to the steps in the first schema with the difference that the number of classes in the classifier must be equal to 2.
Fig. 11. Overview of the training phase of the proposed approach (first schema): the training set of images must be labeled with three labels: one label for each considered order of Gray algorithms and the classifier should be trained using the whole training images.

Fig. 12. Overview of the testing phase of the proposed approach (first schema): the normalized estimate of the scene illuminant for a given image can be obtained by first predicting the best order of Gray algorithms for that image using the trained classifier and then, using PSO optimization and applying the selection algorithm on the Pareto-optimal set of illuminants.
4. Results and experiments

In the previous section, a new approach for combining the output of multiple Gray algorithms for a given image has been proposed based on the determination of the best group of Gray algorithms for that image. In this section, the performance of proposed approach has been evaluated and compared with various color constancy algorithms.

4.1. Algorithms to be combined

The first step in the proposed method is to determine the best group of algorithms for the given image among some predefined algorithms. This task can be done among the Gray algorithms with satisfactory precision because of their natural similarity. Also, the Gray algorithms are known as a family of algorithms with satisfactory good performance with regard to their simplicity and computation efficiency [43]. Therefore, the proposed method is designed for only combination of Gray algorithms. The number of Gray algorithms that can be combined with the proposed approach is somewhat arbitrary. In this paper, we chose the number of predefined algorithms to be equal to 9 (three zero order, three first order and three second order algorithms as shown in Table 1) because the experiment explained in Section 3 showed that at least in almost 80% of the times, the three best Gray algorithms for an image are in the same group (have the same order).

4.2. Performance measure

The angular error ε is used to measure the accuracy of the estimated illuminant:

\[ \varepsilon = \cos^{-1}(\hat{\ell}_s, \hat{\ell}_e) \]  

(29)

where \( \hat{\ell}_s \) is the normalized estimated illuminant, \( \hat{\ell}_j \) is a normalized vector representing real scene illuminant and \( \cdot \) denotes the dot product of two vector. The median angular error has been employed to measure the overall performance of algorithm on the dataset, because this measure is known as the most appropriate measure for this aim [17]. Although the median angular error is an appropriate measure for comparing the overall performance of algorithms on a dataset, it does not tell us everything about the distribution of errors. Therefore, it would be better to somehow compare statistical significance of the whole error distribution of algorithms. To this end, the Wilcoxon Sign Test [44] has been used. Given two samples of random variables A and B the Wilcoxon sign test is used to test the null hypothesis: \( H_0: p = P(A > B) = 0.5 \) which means the probability that A has a value larger than B in 50% of the time. In the case of color constancy algorithms, the random variables A and B are the angular error results for two different algorithms and the test can be used to determine whether the performance of the algorithms is the same (the null hypothesis is true) or whether one algorithm performs significantly better than another (the null hypothesis is rejected). The decision to accept or reject the null hypothesis at a given significance level \( \alpha \) is made on the number of times the random variable A are greater than the corresponding values of B.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Zero order algorithms</th>
<th>First order algorithms</th>
<th>Second order algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Median error</td>
<td>5.3</td>
<td>5.3</td>
<td>5.4</td>
</tr>
</tbody>
</table>

4.3. Dataset

The evaluation of the proposed approach has been done using two public color constancy datasets containing real world images. The first dataset is Ciurea and Funt Gray-Ball dataset [23], a large dataset with 11346 images. The images in this dataset have been extracted from 15 video clips taken at different locations. The real scene illuminant for each image in dataset is acquired using the small gray sphere at the bottom right corner of the images. Some sample images of this dataset are shown in Fig. 13. Note that while evaluating the proposed approach, the gray sphere is omitted from images to avoid biasing the algorithms.

The Gray-Ball dataset is a widely used dataset for evaluation of color constancy algorithms; but it has some disadvantages. The main disadvantage of this dataset is that the images have been extracted from video sequences and therefore some of the images have correlation. Another drawback of this dataset is that the images are processed by an unknown post-processing procedure including gamma mapping and lossy compression and also the quality of images is low. Therefore, as second dataset, the proposed approach has been tested using the Color-Checker dataset [45].

The Color Checker dataset consists of 568 indoor and outdoor images. For each image, a MacBeth Color Checker is placed in the scene in order to obtain the real scene illuminant for that image. Although this dataset contains limited number of images comparing to Gray-Ball dataset, the images in this dataset are not correlated and also the quality of images is high. The Color-Checker dataset is available in RAW format as well as tiff images in sRGB format. The problem with tiff images is that they have been processed by camera automatic post processing. Consequently, tiff images contain clipped pixels, they are non-linear, demosaiced, and also include the effect of the camera white balancing. Hence, Shi and Funt [24] reprocessed the RAW data to obtain linear images with a higher

![Fig. 13. Examples of proposed algorithm results for Ciurea and Funt real world image dataset.](image-url)
(12 bit) dynamic range. The proposed approach is tested on these linear images and to avoid biasing the algorithm, the color checker is omitted from images in the evaluation process. Some gamma corrected example images (\( \gamma = 2.2 \)) of this dataset are shown in Fig. 14.

![Fig. 14: Examples of proposed algorithm results for Color-Checker image dataset.](image)

### 4.4. Parameter optimization

The goal of the first phase in the proposed approach is to determine the best group of color constancy algorithms as precisely as possible. To this end, a classifier has been used and therefore, the success of this phase is directly related to the classification accuracy of the classifier. Hence, the more accurate classifier results in more success in the first phase. The inputs to the classifier are the extracted feature vectors from images; so, in order to achieve high classification accuracy, these feature vectors should be discriminative and optimality of parameters (if any parameter exist) in the process of extracting feature vectors is very important. When the input to the classifier is the concatenation of multiple features with multiple possible parameters (which is the case of the proposed approach), the optimization of the input can be done using two different methods:

1. Optimize the parameters of each feature and then use the concatenation of optimized features as input to the classifier.
2. Concatenate the un-optimized features and then optimize the concatenation as a whole entity and use it as input to the classifier.

The main issue in the first method is that the only practical approach for optimization of parameters for each feature is by assigning constant values to parameters of other features and varying the values for the parameters of the feature at hand to find the optimum. This approach is not perfect; because when using the concatenation of different features as a whole entity, the optimum values of their parameters depend together and independently optimizing each one is not correct and there is no guaranty that the concatenation of independently optimized features be an optimized entity. On the other hand, the second method considers the concatenation of features as a whole entity and tries to optimize it. Hence, it is obvious that the second method is preferred over the first one. Therefore, the proposed approach incorporates the second method. A sub-step in the first phase of the proposed approach uses the DGPP algorithm that optimizes the feature vectors and makes them as most discriminative as possible.

As mentioned before, the first phase in the proposed approach has three consecutive steps:

(a) Extracting a lot of various features from images that can help in accurate classification of images.
(b) Applying DGPP to optimize the obtained features in step one and extract the most discriminative parts.
(c) Applying the classifier on the output of step 2.

In this manner, the second step extracts the most discriminative parts of the obtained features in step one, and discards the low-information dimensions. Therefore, if the second step works properly and if we optimize its parameters, there is no need for optimization of the parameters in the first step. An obvious fact is that the second step may not work properly (not be able to extract the discriminative parts) if the output of first step have very non-related information. Therefore, the main requirement for the first step is that the features (and their parameters) should be chosen in such a way that the feature vectors have enough discriminative information related to image classes and presence of some low-information parts is not a vital concern. Hence, the features and the selected values for their parameters in the proposed approach have been chosen with this fact in mind.

The dimensionality reduction algorithm (DGPP) extracts the discriminative part of the feature vector. In this process, one of the most important parameters is the length of the low dimensional vector representing the discriminative part. The original feature vector length in the proposed approach is 2909 and it is important to choose the best length for its low dimensional representation. The classification accuracy highly depends on this parameter and the best value for this parameter must be found experimentally. Therefore, in our experiments we have varied the value of this parameter from 2909 (no reduction) to 3 (99% reduction) and computed the classification accuracy for both schemas of the proposed approach using the k-fold cross validation on the Gray-Ball dataset. In k-fold cross validation, the training data is divided into k parts; next, the classifier is trained on k-1 parts of the data, and tested on the remaining part. This procedure is repeated k times, so every image is in the test set exactly once and all images will be either in the training set or in the test set. In the case of Gray-Ball dataset, the images are from 15 different clips and images of each clip are correlated. Therefore, 15-fold cross validation is used while making sure that the correlated images (images of the same video clip) are grouped in the same part. The results of this experiment for both schemas of the proposed approach are shown in Fig. 15. As it is expected, on overall, the classification accuracy in the second schema is higher than the classification accuracy of the first schema. One reason is that the number of classes in the second schema is less than the number of classes in the first schema and another reason is that the two of three classes in the first schema are rather similar.

As it is shown in Fig. 15, the classification accuracy with no dimensionality reduction is about 50% and 60% in the first and second schema respectively. The reason for the poor classification performance with no dimensionality reduction is that the classification of high dimensional feature vectors needs a complex classifier and also the high dimensional feature vector contains some low information (non-discriminative) parts that prevent convergence of the classifier. Decreasing the length of the low dimensional feature vector from 2909 to about 500 does not improve the classification accuracy so much. The classification accuracy in both schemas increases by decreasing the length of the low dimensional feature...
vector from 500 to about 200. The maximum accuracy for the first schema is 92.6% which obtained by setting the length of low dimensional feature vector to 200. Also, the maximum accuracy for the second schema is 97.7% which obtained by setting the length of low dimensional feature vector to 210. By decreasing the length of low dimensional feature vector from 200 to 210, the classification accuracy decreases with a large slope. This is because that the reduction removes the necessary discriminative information. Since the maximum classification accuracy has been obtained with the length of low dimensional feature vector set to 200 and 210 for the first and second schema respectively, the proposed approach also uses these values.

Different values for the feature extraction parameters affect the length of the feature vector and therefore change the optimums for the dimensionality reduction parameter (the length of the feature vector after dimensionality reduction). Hence, if for example we change the value of quantization step in computation of edge direction histogram, we should repeat the optimization process (explained in previous paragraphs) for the parameter of dimensionality reduction. As it can be seen, if we want to find the optimal value for all of the parameters in feature extraction (which is indeed not essential), there are many combinations and testing evaluating all of these combinations is practically impossible. Therefore, we have chosen the values parameters of the feature extraction based on experiments as follows:

First, a value for each parameter has been chosen either based on the previous references or such that one can reasonably claim that it is a suitable value. Then, for each parameter, we have tried different values (around the initial value) while keeping the values of other parameters constant. Then, we have optimized the dimensionality reduction parameter. We have repeated this process multiple times and have found the mentioned values in Section 3.1.1 which result in high classification performance as shown in Fig. 15. We have achieved the goal of the first phase in the proposed approach (which is high performance in classification and determination of the best group of algorithms for each image); but certainly, there may be other combinations that result in the same (or slightly better) classification performance.

In the case of quantization step in edge direction histogram, the initial value has been chosen as 10 degree which has been suggested in [18] and after mentioned process in previous paragraph, we have found that each value in range [5,10] results in the same performance (and better than other examined values) while slightly changing the optimum for dimensionality reduction parameter. The values for σ in computation of Weibull parameters have also been chosen based on NIS method [17] which its source code is available for download in [27]. The NIS method shows the effectiveness of Weibull parameters in determination of the best algorithm for each image and suggested to use these values in their source code. Note that there are many combinations of different values for this parameter that evaluation of all of them is indeed practically impossible and we could not find better values than the suggested values using the mentioned process in the previous paragraph. The values for other parameters in feature extraction have also been chosen using the mentioned process. Among these parameters, the quantization interval in the rg-chromaticity histogram had more influence on the effectiveness of the dimensionality reduction and consequently the performance of the classifier. Since the color histogram is usually sparse, choosing large values for this parameter adds many non-useful information to the feature vector and significantly increases its length. As a result, the dimensionality reduction algorithm cannot precisely extract the discriminative parts and therefore, the classification performance decreases significantly. Also, using small values for this parameter causes the valuable color information to be largely discarded and therefore, considerably decreases the classification performance. Using the mentioned process, we have found that for this parameter, the values in range [24,32] result in the same high classification accuracy.

4.5. Ciurea and Funt Gray-Ball dataset

To evaluate the performance of the proposed approach on this dataset, 15-fold cross validation is used. Table 2 shows the result of applying several algorithms on the real world dataset of Ciurea and Funt as well as the results of the Wilcoxon Sign Test on this dataset with 599 confidence interval i.e. α = 0.01 (the results of algorithms other than the proposed approach have been obtained from [27]). A plus sign (+) in the ith row and jth column of the table means that algorithm i is statistically better than algorithm j when judged according to the Wilcoxon test. A minus sign (−) implies that it is worse, and if the box is empty the two algorithms are statistically the same. On the basis of the results in Table 2, it can be concluded that although the max error for the proposed method is higher than other methods, on overall, both schemas of the proposed approach are better than other algorithms. The median and mean angular error and also the Wilcoxon Sign Test results support this claim. Note that the performance of the first schema is better than the second schema, while the classification accuracy in the second schema was better than the first schema. The reason

<table>
<thead>
<tr>
<th>Table 2</th>
<th>The performance of various color constancy methods on Gray-Ball image dataset.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Median error</td>
<td>Mean error</td>
</tr>
<tr>
<td>1</td>
<td>Gray-World</td>
</tr>
<tr>
<td>2</td>
<td>Shades of Gray (p=12)</td>
</tr>
<tr>
<td>3</td>
<td>Edge Based Gamut (σ = 3)</td>
</tr>
<tr>
<td>4</td>
<td>1st Order Gray-Edge (p=1, σ=1)</td>
</tr>
<tr>
<td>5</td>
<td>2nd Order Gray-Edge (p=1, σ=2)</td>
</tr>
<tr>
<td>6</td>
<td>NIS</td>
</tr>
<tr>
<td>7</td>
<td>Proposed method (second schema)</td>
</tr>
<tr>
<td>8</td>
<td>Proposed method (first schema)</td>
</tr>
</tbody>
</table>
is that in the second schema, the first and the second order Gray algorithms are placed into one group and therefore, the proposed method cannot determine the best algorithm for images as precisely as in the first schema when the best algorithm for an image is first or second order. Also note that the authors of the CAS algorithm report the performance of CAS on this dataset as 3.21, but they did not use whole dataset for evaluation of their method. In order to decrease the correlation between test images, they selected a subset of images from dataset containing total number of 1135 images. Therefore, this algorithm has been omitted from Table 2. Also note that on a 2.0 GHz Core 2 CPU with 2 GB of RAM, the mean computation time of the proposed approach on this dataset is 4.7 second per image which is much higher than other considered algorithms. This is because (as expected) the proposed approach is an instance of evolutionary optimization methods which are generally slower than the simple mathematic based algorithms.

### 4.6. Shi and Funk color-checker dataset

To evaluate the performance of proposed approach on this dataset, the 3-fold cross validation has been used. The 3-fold is because the author of this dataset divided the images in three categories. The median, mean and maximum angular errors as well as the results of the Wilcoxon Sign Test with 99% confidence interval for various color constancy algorithms on this dataset are shown in Table 3 (the results of algorithms other than the proposed approach have been obtained from [27]). Again, the performance of both schemas of the proposed approach is better than other algorithms, the performance of the first schema is better than the second schema. The mean computation time of the proposed approach on a 2.0 GHz Core 2 CPU with 2 GB of RAM for this dataset is 6.8 s per image which is again much higher than other considered algorithms. Also, since the quality of images in this dataset is much higher than the images in the Gray-Ball dataset, the mean computation time of proposed approach on this dataset is higher than its computation time on the Gray-Ball dataset.

### 4.7. Discussion and further analysis

As mentioned in Section 1, when combining multiple algorithms with the aim of achieving a better algorithm, the result of combination is not as good as expected if a good algorithm is combined with a bad one. A bad algorithm with even a near zero weight still may decrease the accuracy of the combination output. Also, considering the bad algorithms in the process of determining combination weights increases the complexity of this process and as a result, the algorithm may not be able to determine the weights precisely. Therefore, the task of determining the best algorithms for combination (which is in our case the classification phase) is a very important step.

Table 4 shows the misclassification cost for three best algorithms of Table 1 in terms of increase in median angular error on the Gray-Ball dataset. As it can be seen, the misclassification cost when dealing with the zero order algorithms is very high. For example, if we consider the images that $\vec{e}^{0.9.0}$ (which is a zero order algorithm) is the best algorithm for them, using $\vec{e}^{1.2.1}$ (which is a first order algorithm) instead of $\vec{e}^{0.9.0}$ increases the median angular error for these images by 3.73 degree. Therefore, the high classification accuracy in the proposed approach (as discussed in Section 4.4) is very important.

To approve the importance of the classification phase in the proposed approach, we have done another experiment. In this experiment, the performance of the proposed approach is tested on the Gray-Ball dataset without the classification phase. The test has been done using the algorithms in Table 1 and the results are shown in Table 5. As it is expected, the performance of the proposed approach without the classification phase is far worse than the case with the classification phase.

### 5. Conclusion

In this paper, we have proposed a new combining method for color constancy which consists of two steps. First, a classifier is used to determine the best group of color constancy algorithms for the input image and then, some of the algorithms in this group are combined using multi-objective PSO optimization. As a result, the proposed method always combines algorithms that have good performance for the input image and as the experiments showed, the proposed method achieved a better performance compare to the state of the art algorithms.

### References


### Table 3

The performance of various color constancy methods on color-checker dataset.

<table>
<thead>
<tr>
<th></th>
<th>Median error</th>
<th>Mean error</th>
<th>Max error</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Gray-World</td>
<td>6.3</td>
<td>6.4</td>
<td>24.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>Shades of Gray ($\rho=12$)</td>
<td>4.0</td>
<td>4.9</td>
<td>22.4</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>Edge based gamut ($\sigma=3$)</td>
<td>5.0</td>
<td>6.5</td>
<td>29.0</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>1st order Gray-Edge ($\rho=1, \sigma=1$)</td>
<td>4.5</td>
<td>5.3</td>
<td>26.4</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>2nd order Gray-Edge ($\rho=1, \sigma=2$)</td>
<td>4.4</td>
<td>5.1</td>
<td>23.9</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>NIS</td>
<td>3.1</td>
<td>4.2</td>
<td>26.2</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>Proposed method (second schema)</td>
<td>3.0</td>
<td>4.3</td>
<td>37.4</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>Proposed method (first schema)</td>
<td>2.9</td>
<td>4.1</td>
<td>37.3</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### Table 4

Misclassification cost for three different algorithms in terms of increase in median angular error on the Gray-Ball dataset.

<table>
<thead>
<tr>
<th>Best algorithm</th>
<th>Selected algorithm</th>
<th>$\vec{e}^{0.9.0}$</th>
<th>$\vec{e}^{1.2.1}$</th>
<th>$\vec{e}^{2.1.2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without classification</td>
<td>6.1</td>
<td>0.93</td>
<td>0.87</td>
<td>0</td>
</tr>
<tr>
<td>With classification</td>
<td>3.2</td>
<td>5.1</td>
<td>6.5</td>
<td></td>
</tr>
<tr>
<td>Improvement</td>
<td>2.9</td>
<td>1.8</td>
<td>50.3</td>
<td></td>
</tr>
</tbody>
</table>

### Table 5

The performance of the proposed method with and without the classification phase on Gray-Ball dataset (angular error in degree).

<table>
<thead>
<tr>
<th></th>
<th>Median error</th>
<th>Mean error</th>
<th>Max error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without classification</td>
<td>6.1</td>
<td>0.93</td>
<td>0.87</td>
</tr>
<tr>
<td>With classification</td>
<td>3.2</td>
<td>5.1</td>
<td>6.5</td>
</tr>
<tr>
<td>Improvement</td>
<td>2.9</td>
<td>1.8</td>
<td>50.3</td>
</tr>
</tbody>
</table>


[35] L. Keng-Pei, C. Ming-Syan, On the design and analysis of the privacy-preserving SVM classifier, IEEE Transactions on Knowledge and Data Engineering 23 (2011) 1704–1717.


