Programming and Computer Software, Vol. 30, No. 6, 2004, pp. 347–358. Translated from Programmirovanie, Vol. 30, No. 6, 2004.
Original Russian Text Copyright © 2004 by Lukin, Kubasov.

High-Quality Algorithm for Bayer Pattern Interpolation

A. Lukin and D. Kubasov

Department of Computational Mathematics and Cybernetics, Moscow State University, Vorob'evy gory, Moscow, 119992 Russia
e-mail: lukin@graphics.cs.msu.su, kubasov@graphics.cs.msu.su

Received May 28, 2004

Abstract—In this paper, a brief survey of the algorithms for interpolating images represented in the form of the Bayer patterns is given, and a new interpolation algorithm considerably improving the image quality is suggested. The suggested algorithm is based on the Kimmel algorithm improved by introducing several modifications, such as a more accurate interpolation of the green color, adaptive control of the number of the iterations, and projection onto the input data.

1. INTRODUCTION

Digital cameras and scanners are no longer exotic things or equipment of expensive photo labs: they have become a permanent part of the market of digital devices for home use. The desire to occupy a wider sector of the market is pushing the manufacturers to simplify and, hence, cheapen production while preserving the consumer features and functionality of the devices.

One of such simplifications is to use one array of photo sensors for forming images instead of the three arrays corresponding to the basic colors. Such an array contains photo sensors of several kinds, each of which is sensitive to a certain basic color. These elements are arranged in the array in the mosaic form, which is referred to as the Bayer pattern.

Thus, each element of the array contains information about only one color component, whereas the output digital image must contain all three components R, G, and B for each pixel.

The problem of the interpolation of Bayer patterns (also referred to as demosaicing or demosaicking) consists in obtaining a full-color image by its Bayer pattern. The purpose of the algorithm is to interpolate each color plane at the points where the value of the corresponding color component is unknown.

2. SURVEY OF THE EXISTING METHODS

The majority of the existing methods for solving this problem can informally be divided into linear methods, adaptive algorithms, and mathematical recovery methods.

2.1. Linear Methods

2.1.1. Independent interpolation of color planes. The simplest demosaicing method is to separately apply linear interpolation (for example, bilinear or bicubic) to each color plane. Although this method is fast and easy-to-implement, its application results in the appearance of noticeable artifacts.

One of such artifacts, color moire, which is typical of almost all methods, results from the fact that the positions of sensors of different colors do not coincide. Therefore, many methods use the redundancy of the green pixels in the mosaic for qualitative recovery of high frequencies of the image, and then interpolate the red and blue colors with the use of the recovered green.

2.1.2. Interpolation of color ratios. The recovery of the red and blue colors based on the recovered green relies on certain assumptions about the relationship between the color planes.

One of the assumptions of this kind is the assumption of a fixed ratio of colors (for example, red-to-green or blue-to-green ratios) within one object in the image [2]. Then, having a fully recovered green component, one can interpolate the red-to-green ratios at the adjacent points rather than the values of the red color at these points. This yields a better result compared to the independent interpolation of the color planes, since the green component has a higher sampling rate and can be recovered more accurately (even by means of a linear method).

A weak point of this method is image areas containing a small amount of green. In this case, the ratios of the other colors to the green are high and not accurate because of the noise. This difficulty can be overcome by interpolating the logarithms of the ratios (i.e., the residuals of the colors) rather than the ratios themselves.

2.2. Adaptive Methods

The essence of the adaptive methods is to separately select particular linear filters for each interpolated pixel depending on heuristic rules calculated in the neighborhood of a given pixel.

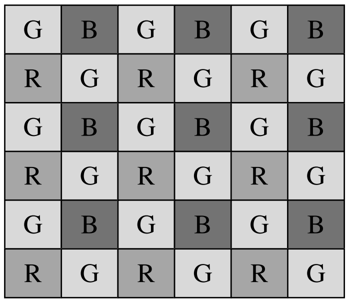


Fig. 1. Bayer pattern.

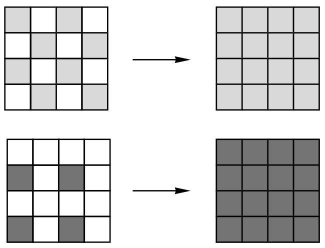
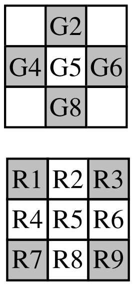


Fig. 2. Interpolation of color planes.



1) Green

$G5 = (G2 + G4 + G6 + G8)/4$

2) Red

$R2 = (R1 + R3)/2$

$R4 = (R1 + R7)/2$

$R5 = (R1 + R3 + R7 + R9)/4$

3) Blue (similar to the red)

Fig. 3. Linear interpolation of a pixel in position 5.

2.2.1. The Edge-adaptive method. When interpolating color ratios (or their residuals), the quality of the initial interpolation of the green color has a great effect on the final result. Therefore, this interpolation should be as accurate as possible; for example, one could use the edge-directed interpolation instead of the linear interpolation.

The simplest variant of the edge-directed interpolation works as follows. For the pixel to be interpolated, the vertical and horizontal gradients are determined by its known neighboring pixels [1], and the direction in which the gradient is smaller is assumed to be the direction along the edge in the neighborhood of this pixel. Then, the desired value is determined as the half-sum of two closest neighbors along the edge.

This method can be improved by considering a larger domain and taking into account the gradients of the other colors when selecting the interpolation direction [2].

After the green color has been recovered by this method, the red and blue component are found by the method of the interpolation of the color ratios.

2.2.2. The Kimmel algorithm. In the Kimmel algorithm, the recovery of the image proceeds in the following three stages:

(1) interpolation of the green color,

(2) interpolation of red and blue by using green, and

(3) correction.

Let us describe these stages in more detail.

2.2.2.1. Interpolation of the green color in the Kimmel algorithm. The unknown value of the green component is determined in the Kimmel algorithm as a linear combination of the known values of the four closest neighbors. The weights E_i in this linear combinations are the probabilities that G_i belong to the same object as G_5.

The weights E_i are computed as follows.

First, the notion of a "derivative" along the four directions (vertical, horizontal, and two diagonal) at a given point is introduced.

Suppose that it is required to calculate the derivatives at the point $P5$ (Fig. 8). The use of the letter P means that the derivatives are computed in the same way independent of what component is given in the pattern at point $P5$. Then, the derivatives are computed by the formulas

$$D_x(P_5) = \frac{P_4 - P_6}{2}, \quad D_y(P_5) = \frac{P_2 - P_8}{2},$$

$$D_{xd}(P_5) = \frac{P_3 - P_7}{2\sqrt{2}}, \quad D_{yd}(P_5) = \frac{P_1 - P_9}{2\sqrt{2}},$$

where P_i are values of the intensity at this point specified in the pattern. It should be emphasized that, whatever component is specified at point $P5$ (green, red, or blue), we always compute differences between the intensities of the same color.

If, at point $P5$, a value of green is given, we can determine the derivatives a little bit more accurately by setting

$$D_{xd}(P_5) = \max\left\{\left|\frac{P_3 - P_5}{\sqrt{2}}\right|, \left|\frac{P_7 - P_5}{\sqrt{2}}\right|\right\},$$

$$D_{yd}(P_5) = \max\left\{\left|\frac{P_1 - P_5}{\sqrt{2}}\right|, \left|\frac{P_9 - P_5}{\sqrt{2}}\right|\right\}.$$

In this case, the weight function is given by

$$E_i = \frac{1}{\sqrt{1 + D^2(P_5) + D^2(P_i)}},$$

where $D(P_i)$ is the derivative along P_i.

2.2.2.2. Interpolation of red and blue by using the known green color. The red and blue colors are recovered by means of the above-described method of the

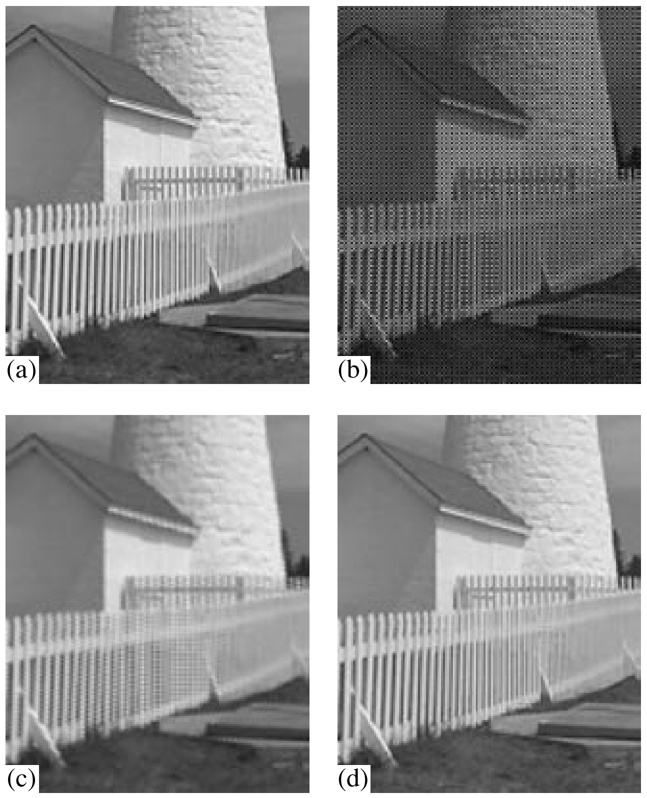


Fig. 4. Original image (a), Bayer pattern (b), bilinear interpolation (c), and the suggested method (d).

interpolation of the color ratios. The ratios themselves are interpolated in the same way as the green pixels were interpolated at the first stage with the use of the weight function *E* defined above.

The interpolation formula for the blue color is similar.

2.2.2.3. Correction in the Kimmel algorithm. This is the key stage of the algorithm, since the majority of the artifacts (e.g., the color moire) are suppressed just at this stage. The basic idea of this stage is as follows.

In the interpolation of the red (blue) color, we assumed that the ratio of the red (blue) color to the green color is fixed within one object. This implies that the ratio of the green to the red (blue) must also be fixed in this region. Therefore, having the interpolation of the red and blue completed, one can correct the green pixels to meet this assumption. However, this will change the original values of the ratios of the red (blue) to the green, so that one will have to modify the values of the red (blue) color. The authors of this method [3] suggest repeating this process three times alternately modifying values of the green and red (blue) colors.

Thus, the correction stage can formally be described as follows:

• Repeat three times:

• Modify the values of the green color in accordance with the values of the ratios of the green to the blue and red by the formulas

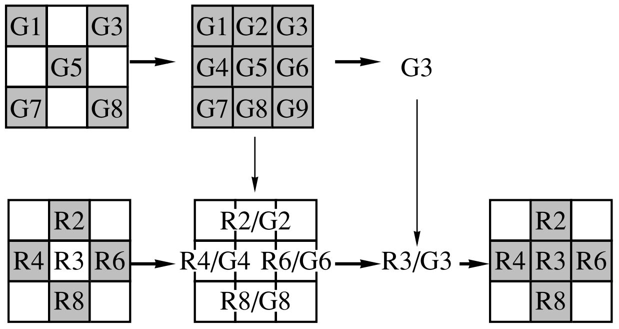


Fig. 5. Interpolation of color ratios.

$$G_5^B = B_5 \frac{E_2\dfrac{G_2}{B_2} + E_4\dfrac{G_4}{B_4} + E_6\dfrac{G_6}{B_6} + E_8\dfrac{G_8}{B_8}}{E_2 + E_4 + E_6 + E_8},$$

$$G_5^R = R_5 \frac{E_2\dfrac{G_2}{R_2} + E_4\dfrac{G_4}{R_4} + E_6\dfrac{G_6}{R_6} + E_8\dfrac{G_8}{R_8}}{E_2 + E_4 + E_6 + E_8},$$

$$G_5 = \frac{G_5^R + G_5^B}{2}.$$

• Modify the values of the red and blue colors in accordance with the values of their ratios to the green by the formulas

$$B_5 = G_5 \frac{\sum E_i \dfrac{B_i}{G_i}}{\sum E_i}, \quad i \neq 5,$$

$$R_5 = G_5 \frac{\sum E_i \dfrac{R_i}{G_i}}{\sum E_i}, \quad i \neq 5.$$

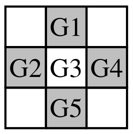
• End of the loop.

2.3. Mathematical Methods

2.3.1. Optimal recovery. To better recover the green color (which affects the quality of the entire interpolation), one can apply the well-known and approved methods, such as, for example, NEDI [7] or the optimal recovery method [4], which is an extension of the former.

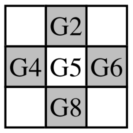
For example, in [5], it is suggested to replace the interpolation of the green in the above-described Kimmel algorithm [3] by the interpolation by means of the optimal recovery method, and the interpolation of the color ratios, by the interpolation of the color residuals.

The idea of the optimal recovery method is to extend the NEDI method, which is described below, by introducing additional functionals that impose bounds on the values of the pixels obtained. This makes it possible



```
(1) Horizontal gradient
        $H = |G2 - G4|$
(2) Vertical gradient
        $V = |G1 - G5|$
(3) If H>V
        G3=(G1+G5)/2
    Else If V>H
        G3=(G2+G4)/2
    Else
        G3=(G1+G2+G4+G5)/4
```

Fig. 6. Edge-directed interpolation. $G1$, $G2$, $G4$, and $G5$ are known values of the green, and $G3$ is the value being interpolated.



$$G_5 = \frac{E_2G_2 + E_4G_4 + E_6G_6 + E_8G_8}{E_2 + E_4 + E_6 + E_8}$$

E_i is weight function

Fig. 7. Interpolation of the green color in the Kimmel algorithm.

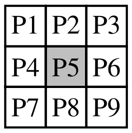


Fig. 8.

to obtain better results owing to the suppression of the artifacts inherent in the NEDI method by means of the bounding functionals. Let us briefly describe the theory of the optimal recovery method.

Let x be an unknown vector and F be a linear functional. Suppose that the values of a set of L linear functionals of x are known: F_i ($i = 1, \ldots, L$). Further, we assume that the desired vector x belongs to the ellipsoidal class

$$K = \{x \in R^n : x^T Q x \le \varepsilon\},$$

where Q is a given matrix.

Note that the functionals F_i determine a hyperplane X in the n-dimensional space, and its intersection with the ellipsoid K is a hyperdisk C_x. The optimal recovery problem consists in finding a vector $\hat{x}$ $x \in C_x$ that minimizes the approximation error of the desired functional:

$$\delta = \max_{x^T Q x} |\hat{y} - y|,$$

where $\hat{y} = (\hat{x})$ and $y = F(x)$. The solution of the problem is the so-called Chebyshev center, the vector $\bar{u} \in X$ with the minimal Q-norm. It has been shown in [4] that the desired vector $\bar{u}$ can be represented as a linear combination of representatives ϕ_i of the linear functionals F_i,

$$\bar{u} = \sum_{i=1}^{L} \alpha_i \phi_i.$$

In the problem of the pattern interpolation, for the vector x, a subset of this pattern that contains the interpolated pixel in the position j is taken. In this case, $F(x) = x_j$, and, for the known functionals F_i ($i = 1, \ldots, L$), the known values of the neighboring pixels are taken, $F_i(x) = x_i$, where x_i are given in the pattern. Then, $F(\bar{u})$ is the desired value of the interpolated pixel.

Clearly, the basic problem here is to construct the class K, i.e., to select such a Q that most adequately represents the image in the neighborhood of the interpolated pixel. It was shown in [4] that $Q = (SS^T)^{-1}$, where S is a training set written as a matrix consisting of m learning column vectors $x \in R^n$.

In [5], the recovery of the green color proceeds in the two following stages:

• Coarse recovery. The class K roughly models the image in the neighborhood of the pixel being interpolated, since the learning vectors are taken only at given points of the pattern (i.e., on the original sparse grid).

• Accurate recovery. The learning vectors are taken from the image interpolated at the first stage; therefore, they have the same scale as that of the vector x being estimated.

At both stages, the training set S consists of all possible learning vectors falling within the window of size 15×15 with the center located at the pixel being interpolated.

The second stage can often be omitted. It considerably improves the quality only in the high-frequency areas, i.e., in the places of repeating fine-textured patterns.

2.3.2. Alternating projections. The method of alternating projections [6] is classified among the so-called POCS (projection onto convex sets) methods. The basic idea of the method consists in obtaining an initial approximation of the interpolated image and subsequently improving it by means of the alternate satisfaction of two sets of constraints.

The first constraint set is the set of the color values specified in the pattern: the results of the interpolation must coincide with these values at the points where the latter are specified.

The second constraint set is obtained by using the assumption that the high frequencies (details) of all three color components at any point are close to one another. This can be achieved by allowing the wavelet coefficients of red and blue differ from the wavelet coefficients of green not more than by a given threshold. The threshold can be either unique for the entire image or different for each point being a function of some local characteristics of the neighborhood of this point. It describes the proximity of the high frequencies corresponding to the different color planes. If they are very close, the threshold must be low; if a large difference between them is expected (allowed), it must be high. For grayscale images, the threshold may be set equal to zero.

Thus, the algorithm scheme looks as follows.

• Initial interpolation of the green color by some method (for example, using the Edge-Adaptive method).

• Repeat several times:

— Perform the wavelet transform of all color planes.

— Modify the wavelet coefficients for red and blue such that they differ from those for green by not more than a given threshold.

— Perform the inverse wavelet transform.

— Substitute the initial values of the colors at the points where they were specified in the pattern.

— End of loop.

2.4. *Estimation of the Algorithm Quality*

To estimate the quality of the algorithms for the interpolation of the Bayer patterns, the PSNR estimate is widely used. To calculate it, a full-color original image is used. From this image, the corresponding Bayer pattern is artificially created and is supplied to the input of the interpolation algorithm. The interpolated image obtained is compared with the original full-color image by means of the PSNR. The difference between the two images can also be measured by applying other techniques, such as the calculation of the color differences in perceptually uniform color models (for example, ΔE_{00}, [9]) and the use of human perception models and distortion masking. In this paper, we use the PSNR and ΔE_{00} estimates, along with the visual assessment of the interpolation artifacts.

The basic artifacts of the interpolated images that are important from the visual estimate standpoint are presented in Fig. 10.

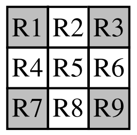
3. SUGGESTED ALGORITHM

The suggested algorithm consists of the following stages:

(1) Accurate interpolation of the green color with the use of the gradient interpolation and NEDI methods.

(2) Application of a modified Kimmel algorithm (adaptively variable number of iterations plus some other insignificant modifications).

(3) Projection onto the input data.



$$R_5 = \frac{E_1\frac{R_1}{G_1} + E_3\frac{R_3}{G_3} + E_7\frac{R_7}{G_7} + E_9\frac{R_9}{G_9}}{E_1 + E_3 + E_7 + E_9}$$

E_i is weight function

Fig. 9. Interpolation of the red color in the Kimmel algorithm.

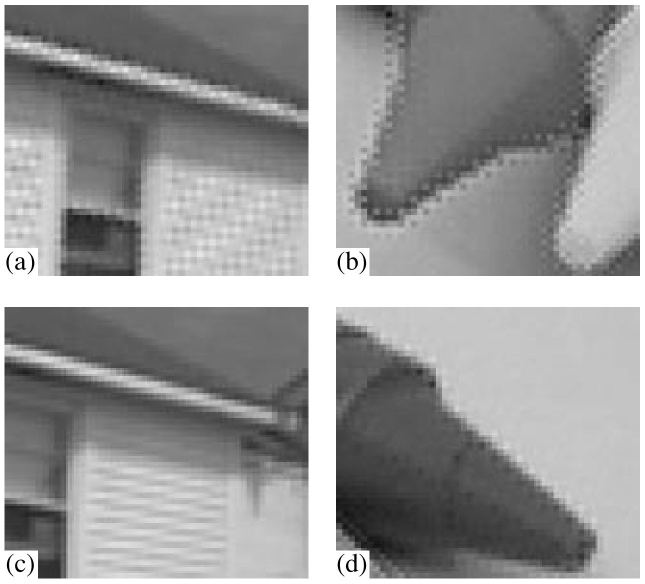


Fig. 10. Color moire (a), zipper effect (b), loss of sharpness (c), and jagged edges (d).

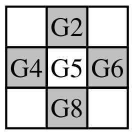
(4) Repetition of the above stages with the use of the interpolated image.

Let us consider these stages in more detail.

3.1. *Interpolation of the Green Color*

In our method of the interpolation of the green color, we determine the direction and weights of the interpolation based on only the green component (the red and blue components are not used at all). This is explained by the facts that the red and blue components in the Bayer pattern have worse resolution and the aliasing arising in these components may result in an incorrect determination of the interpolation direction.

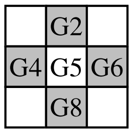
3.1.1. NEDI. The NEDI (New Edge-Directed Interpolation) method has been suggested in [7] (see also [5, 8]). The NEDI method is characterized by the accurate interpolation of the image edges without the effect of jagged edges. However, it has a number of disadvantages as well, such as high computational complexity, watercolor artifacts in fine-textured regions, and the algorithm instability in smooth regions of the image [10]. Therefore, we suggest combining the NEDI method with simpler edge-adaptive interpolation methods.



(1) Computation of the horizontal and vertical gradients H and V

```
(2) If H ≥ V
        G3=(G2+G8)/2
    Else
        G3=(G4+G6)/2
```

Fig. 11. Gradient interpolation of the green color (simplified variant).



(1) $E_2 = E_8 = 1/(\varepsilon + V^8)$

(2) $E_4 = E_6 = 1/(\varepsilon + H^8)$

(3) $G3$ is computed in accordance with Fig. 9

Fig. 12. Gradient interpolation of the green color (simplified variant).

In our method, we use a modification of NEDI suggested in [8] (intensity map blurring, spatial aperture extension). To make the operation of the matrix inversion in the calculation of the optimal weights more stable, a random white noise of small amplitude is added to the intensity map.

3.1.2. Gradient interpolation. For the second (simpler) interpolation method, we use the gradient interpolation. In the simplified gradient interpolation method, we select either the vertical or horizontal interpolation directions by comparing local vertical and horizontal gradients of the green component of the image. The interpolation direction is selected as that corresponding to the smaller gradient.

All gradients are averaged over the neighborhood of the current pixel. At the first step, the gradient H is averaged over a small neighborhood (formula 1):

$$D[i, j] = |G[i, j] - G[i, j+2]|,$$

$$H = D[i, j-1] + D[i-1, j-2]$$
$$+ D[i+1, i-2] + D[i-1, j] + D[i+1, j]$$
$$+ D[i, j-3] + D[i, j+1] + D[i-2, j-1]$$
$$+ D[i+2, j-1] + (D[i, j-5] + D[i, j+3]$$
$$+ D[i-2, j-3] + D[i+2, j-3]$$
$$+ D[i-2, j+1] + D[i+2, j+1]$$
$$+ D[i-1, j-4] + D[i+1, j-4]$$
$$+ D[i-1, j+2] + D[i+1, j+2])/2.$$

Here, D[] is the absolute value of one horizontal difference.

Next, the vertical and horizontal gradients are compared. If their values are close, it is concluded that the interpolation direction is not well defined. In this case, the gradients are computed in a wider neighborhood:

```
If |H − V| < β · (H + V)
 then Use bigger window
```

We use the following three neighborhoods: the least (in accordance with formula 1) neighborhood of size about 4×4 pixels, an average neighborhood of size 10×10 pixels, and a large neighborhood of an approximate size 22×22 pixels. After the direction has finally been selected, the interpolation is performed by averaging two neighboring pixels in the given direction (see Fig. 11).

In a more complicated interpolation method, instead of selecting one of the two directions, the interpolation weights are computed by the formula presented in Fig. 12.

The large power of the gradients in the denominator of the fractions makes it possible to avoid mixing of the vertical and horizontal interpolation directions. This, in turn, allows us to avoid the color moire, which often results from such a mixing.

The next modification improves the quality of the interpolation in smooth regions. It consists in reducing the power of the gradients in the denominators for smooth regions of the image. In such regions, the image noise plays an important role in forming the gradients, and the resulting interpolated image may have a "watercolor" artifact, whereas the ordinary bilinear interpolation does not result in such an artifact. Therefore, the reduction of the power of the gradients in the denominator increases the mixing of the vertical and horizontal interpolation and makes the method similar to the bilinear interpolation method. In our algorithm, depending on the region smoothness, the power of the gradients in the denominator reduces from 8 to 4 or 2.

As a result of the gradient interpolation, we obtain an interpolated green component and an additional array storing the coordinates of the pixels for which the window of the maximum size was used for the computation of the gradient. This array will be used in what follows for identifying complicated ("problematic") regions of the image where the direction of the interpolation might be computed incorrectly.

3.1.3. Combination of the gradient interpolation and NEDI. Our method of the interpolation of the green component combines the NEDI and gradient interpolation methods, which allows us to get rid of the artifacts inherent in both methods and to speed up the NEDI method (since NEDI is now applied to a small portion of the image pixels). To this end, after the gradient interpolation of the green color, we apply a special classification algorithm, which determines the proportions of mixing the NEDI and gradient interpolation results for each pixel, to the resulting image. The purpose of the classification algorithm is to divide the image pixels into two classes. The first class includes edge regions of the image. The second class includes smooth regions of the image, high-frequency domains (i.e., repeating small patterns), and fine-texture regions (grass, leaves, and the like). To the first class, we apply the NEDI algorithm, since it yields nice results on edges. To the second class, the gradient interpolation is applied, since NEDI, in this case, results in artifacts.

The classifier used constructs a solution taking into account a combination of the following criteria:

(1) *Presence of details in the neighborhood of the given pixel.* The criterion is computed as a linear filter (3 × 3) approximating the second derivative ((0, 1, 0), (1, –4, 1), (0, 1, 0)). This criterion allows us to classify smooth regions as those belonging to the second class.

(2) *Ratio of the low-frequency energy to the high-frequency energy in the neighborhood of the given pixel.* The criterion is computed by means of the local windowed two-dimensional Fourier transform (window size is 8 × 8 pixels). The total energy of the coefficients for the high-frequency and low-frequency regions is computed. Low frequencies here are frequencies ranging from 0 to 0.25 periods per pixel (the zero frequency is excluded from the consideration since it corresponds to the average block luminance and has no effect on the edge determination). Based on this criterion, regions with the prevailing high-frequency information (small repeating patterns) fall into the second class, and the regions near the boundaries, into the first class (due to the fact that the low-frequency information dominates in the edges).

(3) *Complexity of the two-dimensional image structure in the neighborhood of the given pixel.* The criterion is computed as follows. First, the 8 × 8 region around the given pixel is transformed to a two-color palette (by the K-means method with a very small number of iterations). Then, the number of the connected regions in the binary image obtained is determined (we consider the 8-connectivity; i.e., each pixel has eight neighbors related to it). By means of this criterion, regions with simple structure (near the edges, there usually exist only two connected regions: one from one side of the edge, and the other, from another side) and fine-texture regions (leaves, grass, and the like) are classified as regions of the first and second classes, respectively.

A combination of these criteria means that, to each pixel, the "probability" of its occurrence in the first or second classes is assigned. It is this number that determines the proportions of mixing the NEDI and gradient interpolation results (Fig. 13a).

Since the classification is done before the NEDI interpolation, the NEDI may be applied to only those points where the corresponding weight is not equal to zero (to further speed up the interpolation, the threshold can be increased from 0 to 0.3 without any loss of accuracy).

3.2. Modification of the Kimmel Algorithm

Having done the interpolation of the green color, we interpolate the red and blue by means of a modified Kimmel algorithm. The basic distinction of the modified algorithm from the original one is that the number of the iterations is adaptively varied.

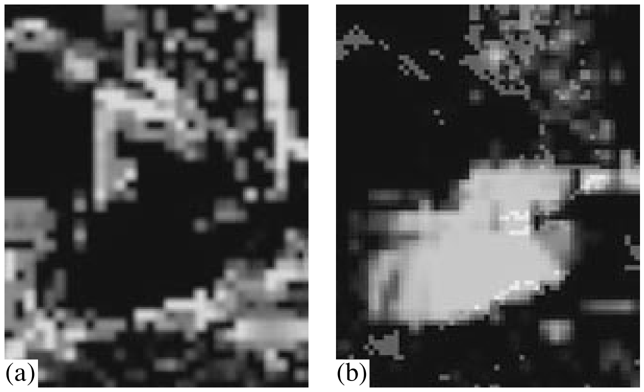


Fig. 13. Classification of pixels for switching the interpolation methods (a) and the "problem map" (b) (the original image is shown in Fig. 4).

3.2.1. Variable number of iterations. The standard Kimmel algorithm uses three iterations for recovering the red and blue colors. We carried out experiments to study the dependence of the resulting image on the number of the iterations. The results of the experiments showed that, as the number of the iterations grows, the color moire is better suppressed owing to better mutual matching of the three color components. However, simultaneously, the color saturation in the regions of the color changeover worsens [11]. Three iterations are usually quite sufficient for the majority of the images and yield the best values of the PSNR. However, the results can considerably be improved by adaptively controlling the number of the iterations.

The basic goal of the Kimmel algorithm is to remove the color moire, which appears because of the aliasing of the red and blue components, i.e., in the regions where, because of the low sampling rate, the high spatial frequencies cannot be captured. The basic idea of our approach is to find such regions and to use an increased number of the Kimmel iterations in these regions. The search for such regions is based on two criteria.

The first criterion is an uncertainty of the determination of the gradient the information about which is stored in the auxiliary array (see Section 3.1.2).

The second criterion is the image regions classifier similar to that described in Section 3.1.3. The distinction of the former from the latter consists in that the second criterion is used in the inverse manner (since the regions with the prevailing high-frequency information must be classified as those from the second class) and the third criterion is not used (in practice, these classifiers are combined, since they work with the same data).

The result of the classifier operation is a distribution map showing the probabilities that the pixels belong to the second class. This map is further referred to as the "problem map," since it shows the regions where the color moire may probably appear (Fig. 13b).

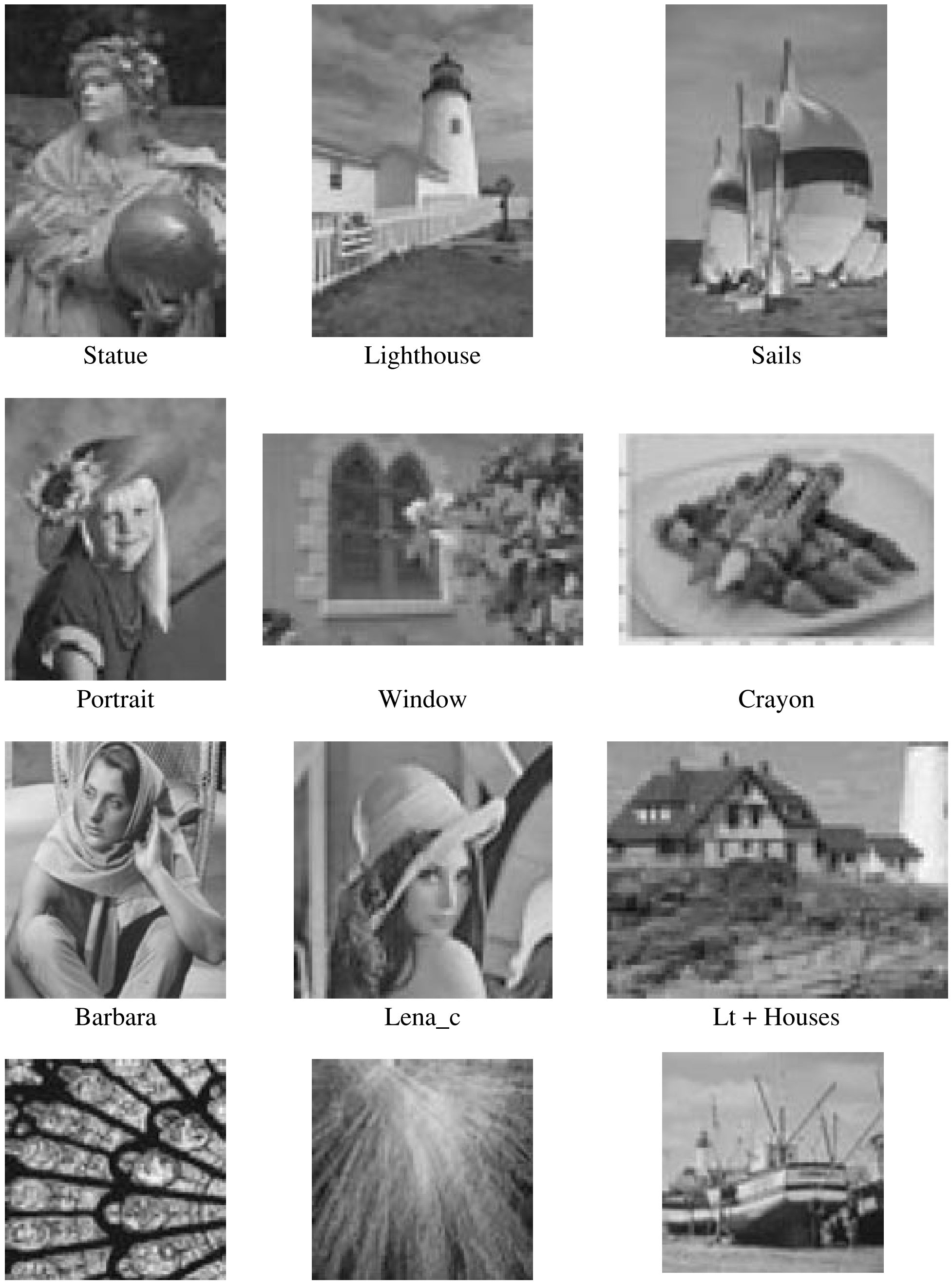


Fig. 14. Test set of images.

The "problem map" is used to select the number of the iterations in the Kimmel algorithm: in "no problem" regions, the number of the iterations may be small, e.g., one or two iterations. In the "problematic" regions, the number of the iterations increases up to 10–12.

3.2.2. Extending the window for the gradient calculation. When calculating the interpolation weights in the Kimmel algorithm, the gradients along several directions are computed. Each gradient is found as a difference of two pixels: $Grad = |G[i, j] - G[k, m]|$.

In our algorithm, the computation of the gradient is modified through the calculation of differences in a larger spatial window:

$$\begin{aligned} Grad = {} & 8 \cdot |G[i, j] - G[k, m]| \\ & + |G[i+1, j] - G[k+1, m]| \\ & + |G[i-1, j] - G[k-1, m]| \\ & + |G[i, j+1] - G[k, m+1]| \end{aligned}$$

$$+|G[i, j-1] - G[k, m-1]|.$$

This resulted in a slight increase in the PSNR.

3.2.3. Semi-recursive color update. In the original Kimmel algorithm, the new values of the corrected colors are calculated by means of one array, and the new values themselves are written to another array. At the next iteration, the arrays swap over. In our algorithm, we write the new values of the pixels to the original array and use them when computing weighted values of the subsequent interpolated pixels. The array of the pixels is looked through from left to right and from top to bottom in the course of the odd iterations and in the reverse order in the course of the even iterations. This small modification also improved the resulting value of the PSNR.

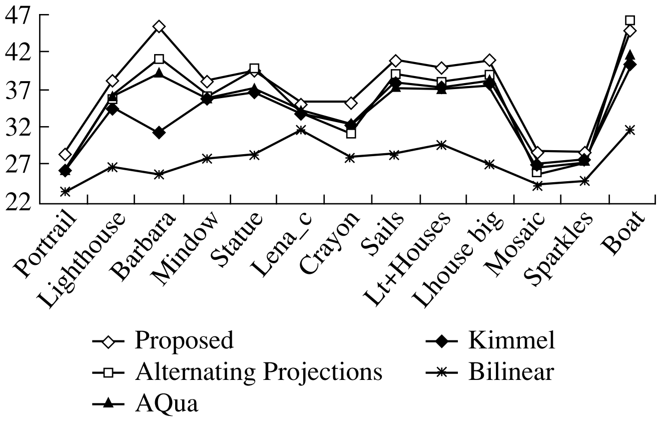


Fig. 15. The values of the PSNR–RGB for different images.

3.3. Projection onto the Input Data

The operation of the Kimmel algorithm results in matching three color components and reduces the color moire. On the other hand, the algorithm modifies values of the color components at the points where they were initially given. Obviously, the substitution of the original values of the color components for the interpolated values at these points improves the PSNR. It turns out that the visual quality of the image is also improved in this case. If we consider the interpolation problem as the problem of projecting onto convex sets of constraints (POCS) [6], the substitution of the original values of the color components is the projection of the image onto the input data.

In our algorithm, we use a more complicated model of projecting images to the input data. The idea of the method is to modify two other color components of the given pixel together with the color component being substituted. These components are modified in accordance with the local model of the color variation in the image.

The local model is constructed as follows. We try to approximate all colors of the pixels from the neighborhood of the given pixel by a straight line in the three-dimensional color space. Let us represent the colors of these pixels as a "cloud" consisting of points in the three-dimensional color space. Then, we need to draw a straight line in the space that optimally approximates this cloud. In our method, we simplify the standard LMS problem by taking the line segment connecting the two points of the cloud that are most distant from one another. This interval is called a *color direction vector.* It shows the dominant direction of the color variation in the three-dimensional color space in the neighborhood of the given pixel of the image. For example, if there is a boundary between the black and white colors (including intermediate colors as well) in the neighborhood of the given pixel, then the color direction vector will have equal components R, G, and B.

In our algorithm, we use the 3 × 3 neighborhood and, after finding the color direction vector, determine the deviation of the pixels belonging to the neighborhood from this vector. This deviation may be viewed as an indicator of the consistency of the local one-dimensional color model.

When the color direction vector is found, the substitution of the original color components is made as follows: the original component being substituted determines the magnitude of the shift of the pixel color, and the shift direction is determined by the color direction vector.

This simple rule needs certain refinements, since it can result, for example, in very large variations of the components if the color direction vector is perpendicular to the color component being substituted. Therefore, in our implementation, we introduced certain restrictions on the maximum variation of the color components of the following form (based on the example of the substitution of the green component):

```
Green = OriginalGreen;
if (CDV[Green]>eps)
{
  Amount = 1 – eps/CDV[Green] +
GryscaleMeasure;
```

Average values of the PSNR on the test images

Method	Average value of PSNR–RGB	Average value of PSNR–CIEDE2000
Bilinear	27.50	35.56
Kimmel	33.50	42.57
AQua-2	34.63	42.99
Alternating projections	35.24	42.76
Suggested algorithm	37.10	44.36

```
  Red += (OriginalGreen - Green) * CDV[Red] /
                        CDV[Green] * Amount;
  Bound(Red, -Limit, +Limit);
  Blue += (OriginalGreen - Green) * CDV[Blue] /
                        CDV[Green] * Amount;
  Bound(Blue, -Limit, +Limit);
}
```

Here, *Red*, *Green*, and *Blue* are values of the color components, *OriginalGreen* is the value of the green component being substituted, {*CDV*[*Red*], *CDV*[*Green*], *CDV*[*Blue*]} is the color direction vector, and *Limit* is the restriction on the variation of the color components (may be computed with regard to the consistency of the local color model).

The variable *Amount* reduces the action of the color component being substituted on other color components of the given pixel. Its value is computed on the basis of the following two criteria:

(1) Reduction of the action to avoid instability in the case where the vector *CDV* is perpendicular to the green component (the term with *Eps*).

(2) Increase of the action when the direction of *CDV* is close to {1, 1, 1}. This criterion makes it possible to increase the relationship of the color components for black-and-white images and to improve the quality of the recovery. Simultaneously, it reduces the relationship of the color components for regions with changeover between different colors, which also improves the quality and reduces the "zipper effect" artifact.

3.4. Repeated Iteration of the Algorithm

The quality of the resulting image can additionally be improved by applying the entire algorithm once again. At this stage, in the interpolation of the green component, the image with full resolution, which was obtained after the first iteration, is used. In this way, the quality of the interpolation of the green component is considerably improved, especially in the regions where the high-frequency information dominates. In the course of the first iteration, the interpolation direction could be determined incorrectly because of the aliasing, which appeared due to an insufficient sample rate for the green color. After projecting the image onto the input data in the course of the first iteration, the desired high frequencies are partially recovered, which improves the repeated interpolation of the green component in the course of the second algorithm iteration.

4. RESULTS AND DISCUSSIONS

We compared the suggested method with the best above-described algorithms. For the test set, we used several images, which are often used in publications.

The result of the application of the algorithm to the Bayer pattern generated by a given image was compared with the original image. The quality of the recovery was measured by means of the PSNR calculated in the color space RGB and in terms of the color differences ΔE_{00} in the color space CIEDE2000 [9].

The test images are shown in Fig. 14. (Averaged) results of the application of various interpolation methods to these images are presented in the table.

The values of the PSNR for each image are plotted in Fig. 15.

Figures 16 and 17 show examples of the image interpolation obtained by means of different algorithms. Other examples and illustrations can be found in [11].

Let us describe specific features of the resulting images obtained by different algorithms.

The linear algorithms (bilinear and bicubic interpolation) demonstrate the worst quality and the maximum number of artifacts.

The Kimmel algorithm shows good results on the boundaries between different colors and almost eliminates the "zipper effect" artifact. However, it is not capable of eliminating the color moire in complex high-frequency regions of the image. As the number of the iterations grows, the suppression of the moire is improved, but the color saturation reduces.

The algorithm AQua-2, which is based on the Optimal Recovery and Kimmel algorithms, demonstrates better results owing to the use of a more complicated interpolation of the green color. The disadvantages of the method are the great dependence of the resulting image on the window size *OR* (the lack of the window size adaptation) and the insufficient sharpness of the resulting image (since the image is not projected onto the initial data after the Kimmel algorithm).

The Alternating Projections method works well in the case of images for which the color direction vector coincides with the gray color axis (including grayscale images). However, if the image does not meet this requirement (for example, near the boundaries between different colors), the algorithm results in a pronounced zipper effect.

In our method, we tried to combine the advantages of all considered algorithms and to adaptively control the size of the gradient computation window, the number of the iterations in the Kimmel algorithm, and the

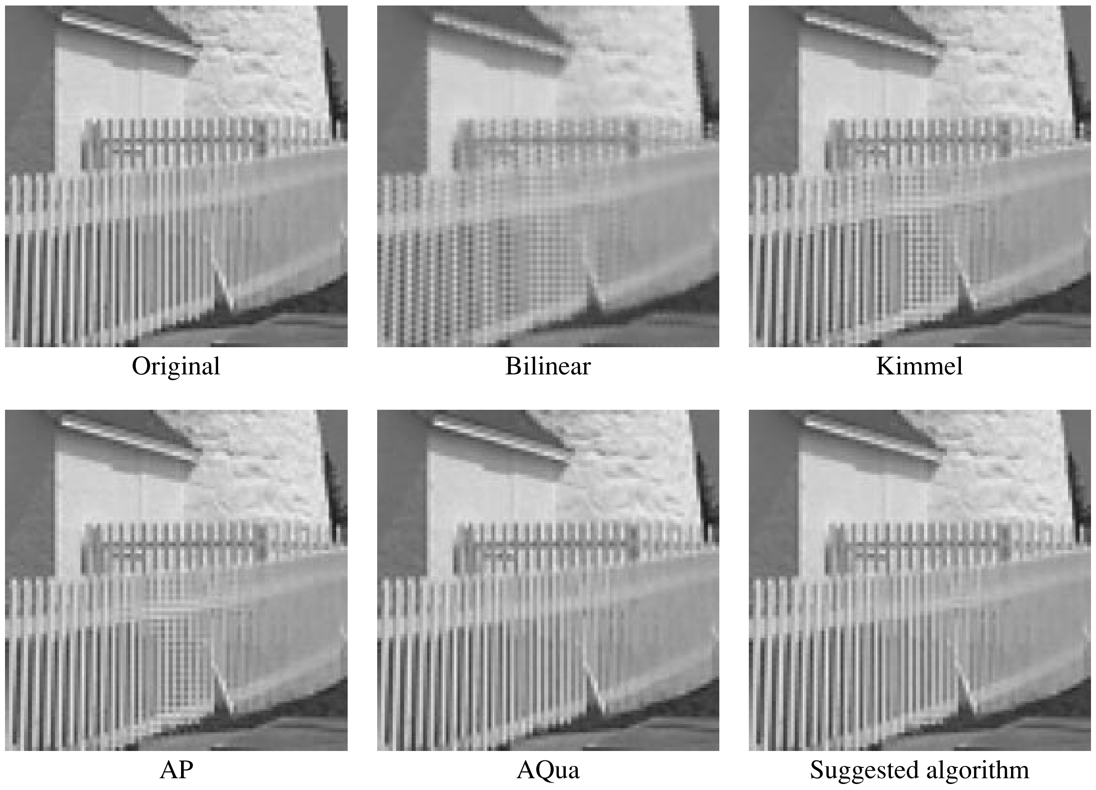


Fig. 16. Results of the application of different methods to the image Lighthouse.

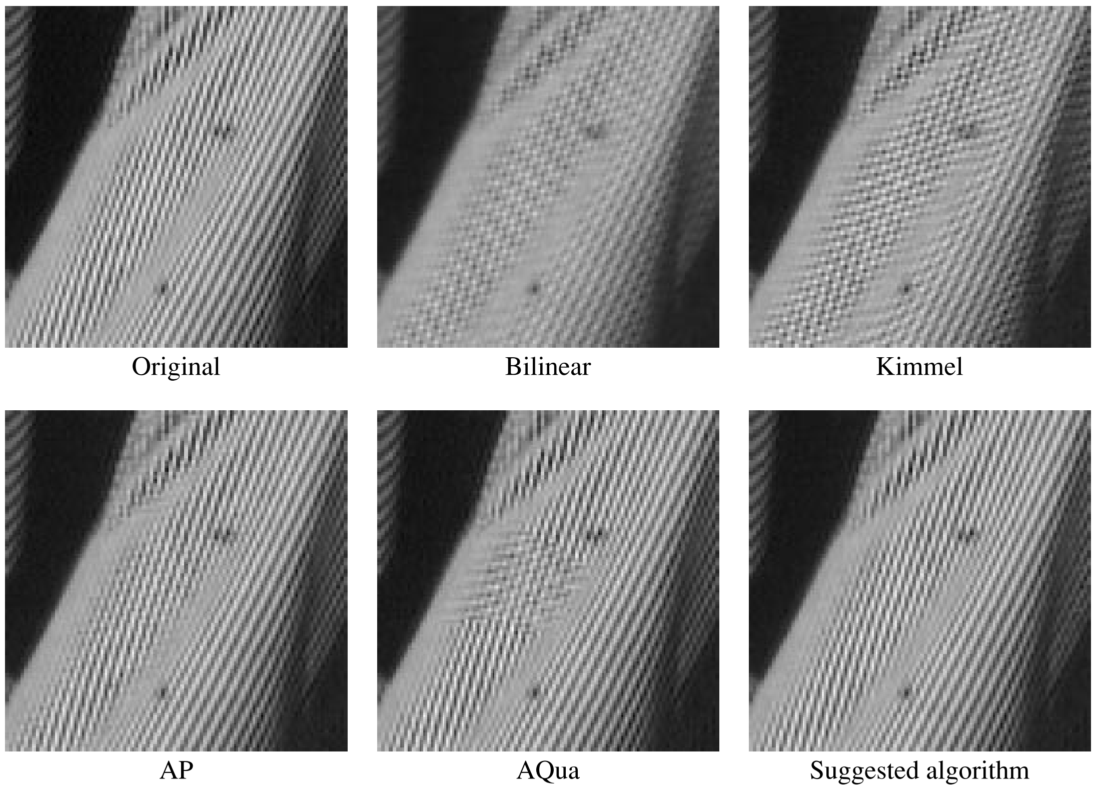


Fig. 17. Results of the application of different algorithms to the image Barbara.

projection of the image onto the input data. Based on the results of the PSNR measurements and on the visual quality of the images, we may conclude that the suggested algorithm for interpolating the Bayer patterns is superior to all other algorithms discussed in this paper. The temporal complexity of our method is about three times higher than that of the Kimmel algorithm.

REFERENCES

1. Hibbard, R.H., Apparatus and Method for Adaptively Interpolating a Full Color Image Utilizing Luminance Gradients, US Patent 5,382,976, 1995.
2. Laroche, C.A. and Prescott, M.A., Apparatus and Method for Adaptively Interpolating a Full Color Image Utilizing Chrominance Gradients, US Patent 5,373,322, 1994.
3. Kimmel, R., Demosaicing: Image Reconstruction from CCD Samples, *Proc. Trans. Image Processing*, 1999, vol. 8, pp. 1221–1228.
4. Muresan, D.D., Review of Optimal Recovery, http://dsplab.ece.cornell.edu/papers/technical_reports/review_or.pdf.
5. Muresan, D.D. and Parks, T.W., Optimal Recovery Demosaicing, *IASTED Signal and Image Processing Conf.*, Hawaii, 2002.
6. Gunturk, B.K., *et al.*, Color Plane Interpolating Using Alternating Projections, *IEEE Trans. Image Processing*, 2002, vol. 11, no. 9, pp. 997–1013.
7. Li, X. and Orchard, M.T., New Edge-Directed Interpolation, *IEEE Trans. Image Processing*, 2001, vol. 10, no. 10.
8. Leitao, J.A., Zhao, M., and de Haan, G., Content-Adaptive Video Up-Scaling for High-Definition Displays, *Proc. of IVCP 2003*, 2003, vol. 5022.
9. Luo, M.R., Cui, G., and Rigg, B., *The Development of the CIE 2000 Colour Difference Formula: CIEDE2000*, Colour & Imaging Inst., Univ. of Derby, UK.
10. Lukin, A., Image Resampling Algorithms (demo webpage), http://audio.rightmark.org/lukin/graphics/resampling.htm.
11. Lukin, A. and Kubasov, D., Interpolation of Bayer Patterns (web-page), http://audio.rightmark.org/lukin/graphics/demosaicing.rus.htm.