Chapter 7
Locally Adaptive Processing of Color Tensor Images Represented as Vector Fields

Lakhmi C. Jain, Roumen K. Kountchev, and Roumiana A. Kountcheva

Abstract A new approach for locally-adaptive processing of color RGB images represented as tensors of size M × N × 3, is offered in this work. Unlike the famous similar methods of the kind, the processing here is executed on a single matrix only, which comprises the modules of the vectors, corresponding to the image pixels' colors. A group of related basic algorithms for locally-adaptive processing is presented, which have lower computational complexity than that of the algorithms, applied individually on each of the RGB components. As it is known, in the famous color RGB transform models of the kind YCrCb, HSV, HSI, Lab, KLT, etc., the processing is applied on the most powerful transformed color component only, and after inverse operation, the original RGB model is restored. In contrast, the idea for locally-adaptive processing does not need direct and inverse transform of the color model. Together with this, the brightness and the color hue of the processed image pixels, are retained. The characteristics of the proposed basic algorithms for contrast enhancement, linear and non-linear sharpness filtration, noise suppression, and texture segmentation, are defined. Some examples for locally-adaptive processing of color medical images are given, which illustrate the related algorithms. The presented approach could be also used for other kinds of color images, where the visibility of their local structure is of high importance.

L. C. Jain
KES International, Selby, UK

R. K. Kountchev
Technical University of Sofia, Sofia 1000, Bulgaria

R. A. Kountcheva (✉)
TK Engineering, Sofia 1582, Bulgaria
e-mail: kountcheva_r@yahoo.com

R. Kountchev et al. (eds.), *New Approaches for Multidimensional Signal Processing*,
Smart Innovation, Systems and Technologies 385,
https://doi.org/10.1007/978-981-97-0109-4_7

7.1 Introduction

The initial form for color image representation is a tensor of size M × N × 3, which has three sections—the matrices R,G,B each of size $M \times N$, whose pixels have $m = 2^b$ intensity levels in the range (0 to $m - 1$), coded through 24 bpp, for $b = 8$. The objective of this work is to present and analyze some basic algorithms for locally-adaptive processing of color tensor images represented as vector fields, and to evaluate the advantages in some basic operations for various applications, such as: contrast enhancement, noise filtration, contours extraction, areas segmentation, etc. In many publications [1–6] related to color images processing, used the approach in which the RGB color model is transformed by using some other models: YCrCb, HSV, HSI, Lab, KLT, etc. [7]. In these cases, the chosen operations are applied on the most powerful component of the new model, after which the original RGB model is inversely restored. Such approach implies the use of direct and inverse transform of the color model, which increases significantly the computational complexity of the processing. In [8], a method is described to improve the quality of RGB images through saturation increase and retaining the hue, but without brightness preserving.

In this work, a new approach is offered for locally-adaptive processing of color images, represented as matrices of the modules of the color vectors, framed by a sliding window. This approach is highly efficient and adaptive, which opens wide abilities in various application areas. The proposed algorithms do not need direct and inverse transform of the color model and ensure hue and brightness preservation in the processed image. These properties of the introduced algorithms are based on the new form for color tensor image presentation, i.e.—the vector field, which comprises the colors of all pixels. The paper is structured as follows: in Sect. 7.2, the tensor image representation as a vector field is explained; in Sect. 7.3—the essence of the proposed new approach is given; in Sect. 7.4, the image color vectors' modules are defined through 2D-DFT; in Sect. 7.5, the local histograms of color vectors' modules are calculated; Sect. 7.6 is about the local cumulative histograms application; in Sect. 7.7 are given the details of the locally-adaptive filtering for tensor images, represented as vector fields; and the conclusions are in Sect. 7.8.

7.2 Color Image Tensor Representation as a Vector Field

7.2.1 Color Vectors Presentation in Orthogonal Coordinate System Framed by a RGB Cube

In Fig. 7.1a, an RGB cube is shown, into which are defined the color vectors $\mathbf{C}_{k,i}$ for the pixels (k,i) of the color tensor image of size $M \times N \times 3$, 24 bpp. One example vector placed in the color cube is shown, denoted as $\mathbf{C}_0$. The color vector $\mathbf{C}_{k,i}$ in the orthogonal RGB coordinate system, is defined by the relation:

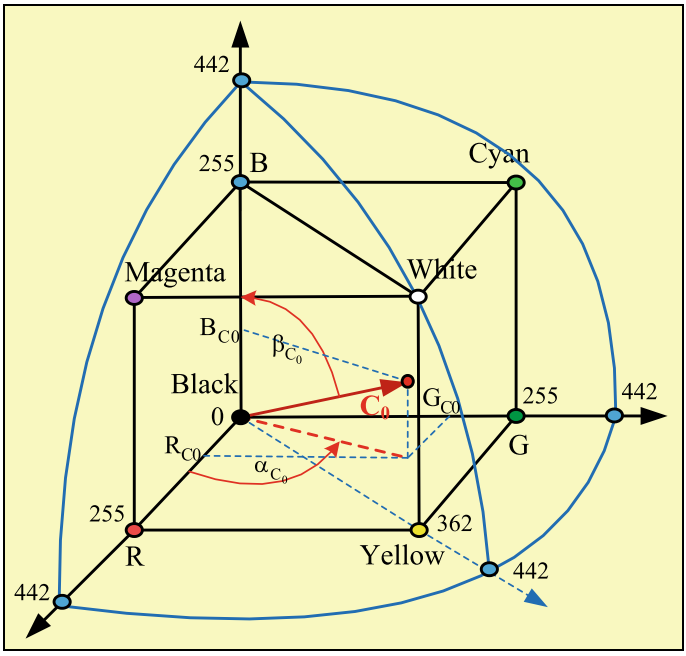

Fig. 7.1 RGB cube in a sphere with a radius MC = 442 and the color vector

$$\mathbf{C}_{k,i} = [R_{k,i}, G_{k,i}, B_{k,i}]^T \text{ for } k/i = 0, 1, 2, \ldots, (M-1)/(N-1). \tag{7.1}$$

7.2.2 RGB Image Representation as a 2D Vector Field

The tensor RGB image representation as a 2D vector field in the 3D spherical coordinate system, is defined by the modules and the phase angles of the color vectors, $\mathbf{C}_{k,i}$.

The color vector $\mathbf{C}_{k,i}$(denoted as $\mathbf{C}_0$ on Fig. 7.1) could be represented in the spherical polar coordinate system, in accordance with the relation below:

$$\mathbf{C}_{\mathbf{k,i}=}[M_C(k,i), a_C(k,i), b_C(k,i)]^{\mathrm{T}} \text{for } M_C(i,j) = 0, 1, 2, \ldots, 255\sqrt{3}, \tag{7.2}$$

where $(255\sqrt{3} \approx 441.6 \approx 442)$ and $0 \le \alpha_C(k,i) \le \pi/2$; $0 \le \beta_C(k,i) \le \pi/2$.

$\mathbf{C}_0 = [R_{C0}, G_{C0}, B_{C0}]^T$, colored in red (in the RGB cube).

The module, and the orientation angles of the color vector $\mathbf{C}_{k,i}$, are defined by the relations:

$$M_C(k,i) = \|\mathbf{C}_{k,i}\| = \sqrt{R_{k,i}^2 + G_{k,i}^2 + B_{k,i}^2} \tag{7.3}$$

$$\alpha_C(k,i) = \arcsin[B_C(k,i)/M_C(k,i)] \tag{7.4}$$

$$\beta_C(k,i) = \arcsin\{G_C(k,i)/\sqrt{[R_C(k,i)]^2 + [G_C(k,i)]^2}\} \tag{7.5}$$

As a result, the matrices $[R(k,i)]$,$[G(k,i)]$, and $[B(k,i)]$, each of size $M \times N$, are replaced by two matrices: of the modules $[M_C(k,i)]$, and of the color vectors' angles $[\theta_C(\mathrm{k,i})] = [\alpha_C(k), \beta_C(i)]$, both of size $M \times N$. The relationship between the orthogonal and the spherical coordinate systems (R, G, B) and (M_C,α_C,β_C), is:

$$R_C = M_C \cos\ \alpha \sin\beta;\ \mathrm{G_C} = M_C \cos\ \alpha \cos\ \beta;\ \mathrm{B_C} = M_C \sin\alpha; \tag{7.6}$$

$$M_C = \sqrt{R_C^2 + G_C^2 + B_C^2};\ \alpha_C = \arccos(R_C/\sqrt{R_C^2 + G_C^2});\ \beta_C = \arcsin(B_C/M_C). \tag{7.7}$$

Then, the color vector could be represented as follows:

$$\mathbf{C}_{k.i} = M_C(k,i)e^{j\,\theta_C(\mathrm{k,i})} = M_C(k,i)e^{j\,[\alpha_C(k),\beta_C(i)]}$$

7.3 The Essence of the New Approach

The new approach is based on the replacement of the three matrices of the tensor image RGB components by a single matrix, which comprises the modules of the color vectors of the pixels. The locally-adaptive processing is applied on the module of the vector, corresponding to the pixel placed in the center of a sliding window. The processing is based on the well-known algorithms used for halftone matrix images, which are the particular case of the color tensor images, when the vertices of all color vectors are placed on the diagonal between the black and white areas in the RGB cube. For such vectors, the three components are equal, i.e. $R = G = B$, $= \alpha\beta = \pi/4$, and their modules are in the range from 0 up to $(m-1)\sqrt{3}$ (here m denotes the number of values for each component R, G, and B). Depending on the processing algorithm used, the magnitudes of part of the so calculated color vectors could get too large values, and as a result, their ends will be placed out of the RGB cube: this happens if at least one of the R, G, B components is larger than $(m-1)$. The colors of the corresponding image pixels could not be reproduced accurately, and produced noticeable color distortions. To solve the problem, the largest color component should be corrected, so as to get a new value, equal to $(m-1)$. Then, to avoid additional color distortions in the restored image, all color vectors placed in and out of the cube, should be corrected accordingly. As a result, the brightness is changed too. To retain the pixel brightness, additional correction is needed (which

follows the color correction). Both corrections (color and brightness) are needed only in case that the number of vectors placed out of the color cube is higher than a pre-defined threshold. In the text below are given the specific features of the basic algorithms for locally-adaptive processing of color images, applied on the matrices of the color vectors' modules, framed by the sliding window.

7.4 Definition of the Image Color Vectors' Modules Through 2D-DFT

To define the image color vectors' modules, forward 2D Discrete Fourier Transform (2D-DFT) is executed for the modules $M_C(k.i)$ of the color vectors $\mathbf{C}_{k,i}$, in correspondence with the relation:

$$\mathbf{s}_C(q,\,p) = \frac{1}{M \times N} \sum_{k=0}^{M-1} \sum_{i=0}^{N-1} M_C(k,i) e^{-2\pi j\left(\frac{qk}{M}+\frac{pi}{N}\right)} \quad \text{for } q/p = 0, 1, .., (M-1)/(N-1). \tag{7.8}$$

Here, $\mathbf{s}_C(q,\,p)$ is the spectrum color coefficient. The inverse 2D-DFT is defined by the relation:

$$M_C(k,i) = \sum_{q=0}^{M-1} \sum_{p=0}^{N-1} \mathbf{s}_C(q,\,p)\, e^{2\pi j\left(\frac{qk}{M}+\frac{pi}{N}\right)} \text{ fork/i} = 0, 1, .., (\mathrm{M}-1)/(\mathrm{N}-1). \tag{7.9}$$

Each spectrum coefficient $\mathbf{s}_C(q,\,p)$ is represented as a vector in the complex space:

$$\mathbf{s}_C(q,\,p) = \mathrm{Re}(\mathbf{s}_C(p,q)) + j\mathrm{Im}(\mathbf{s}_C(p,q)) = \|\mathbf{s}_C(p,q)\|\, e^{j\phi(p,q)} \tag{7.10}$$

where $\|s_C(p,q)\| = \sqrt{\mathrm{Re}^2(\mathbf{s}_C(p,q)) + \mathrm{Im}^2(\mathbf{s}_C(p,q))}$ is the amplitude 2D spectrum, and $\phi(p,q) = arctg\left[\frac{\mathrm{Im}(\mathbf{s}_C(p,q))}{\mathrm{Re}(\mathbf{s}_C(p,q))}\right]$ is the phase 2D spectrum of the matrix $[M_C(k,i)]$, which comprises the color vectors' modules.

In general, the 2D-DFT in a sliding window could be used for homomorphic locally-adaptive filtering of the existing multiplicative or convolutional noises in the color image, retaining the saturation transitions. To accelerate the 2D-DFT calculation, the well-known algorithm for 2D fast Fourier transform (2D-FFT) is used [9].

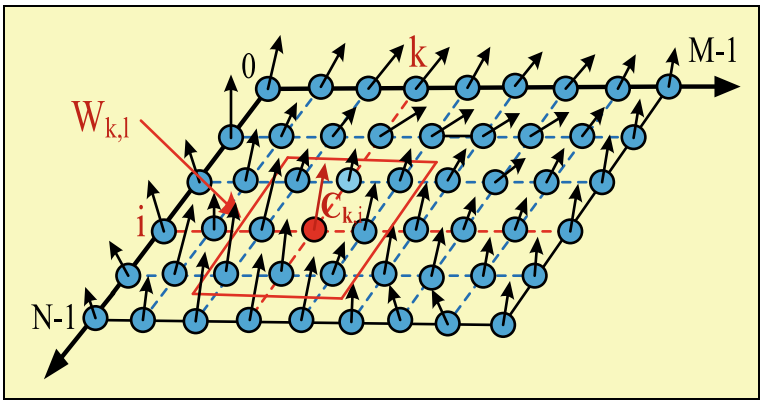

Fig. 7.2 The vector field Ck,i for the pixels of the color tensor image of size $M \times N \times 3$, and the sliding window $W_{k,i}$, used for the local processing

7.5 Calculation of the Local Histograms of Color Vectors' Modules

7.5.1 Local Histogram

The local histogram of the modules of image color vectors $\mathbf{C}_{k,i}$ in the sliding window $W_{k,i}$ of size $(2b + 1) \times (2b + 1)$ (framed in red on Fig. 7.2 for the case $b = 1$), is defined by the relation:

$$h_{k.i}(r) = \frac{N_{M_C}^{k,i}(r)}{(2b+1)^2} \text{ for} r = 0, 1, 2, \ldots, \tag{7.11}$$

where $m'' = \left\lfloor 255\sqrt{3} + 0.5 \right\rfloor = 442$, when $m - 1 = 255$; $\lfloor \bullet \rfloor$ denotes the "rounding" operator; $N_{M_C}^{k,i}(r)$ is the number of pixels, for which the value of the module $M_C(k, i)$ for the corresponding color vector $\mathbf{C}_{k,i}$, is equal to r.

7.5.2 Local Modified Histogram

The local modified histogram of the modules of color vectors $\mathbf{C}_{k,i}$ of the tensor RGB image framed by the sliding window $W_{k,l}$, is defined as follows:

– the adaptive threshold $CL_{k,i}$ is calculated, which limits the local histogram, $h_{k,i}(r)$. The area of the histogram, which is above this limiting value, is represented as a rectangle of same area, which has one side of length $m' = \left\lfloor m\sqrt{3} + 0.5 \right\rfloor$. The

so calculated rectangle area is added to the part placed under the threshold and is obtained from the so-called "modified local histogram", $h^M_{k,i}(r)$. The equalization of this modified histogram is much more accurate than that of the initial local histogram, $h_{k,i}(r)$. The value ($\mathrm{CL_{k,i}}$) is automatically calculated following the conditions to equalize the histogram areas placed above, and below the threshold [10], i.e.:

$$\sum_{r=0}^{m''} h'_{k,i}(r) = \sum_{r=0}^{m''} h''_{k,i}(r) = 0.5 \text{ for } h'_{k,i}(r) + h''_{k,i}(r) = h_{k,i}(r). \tag{7.12}$$

Accordingly, the parts of the histogram $h_{k,i}(r)$, which are above and below the threshold value, are defined by the relations:

$$\begin{aligned} h'_{k,i}(r) &= \begin{cases} h_{k,i}(r) - CL_{k,i} & \text{for } h_{k,i}(r) \geq CL_{k,i}; \\ 0 & \text{for } h_{k,i}(r) < CL_{k,i}; \end{cases} \\ h''_{k,i}(r) &= \begin{cases} \mathrm{CL}_{k,i} & \text{for } h_{k,i}(r) \geq CL_{k,i}, \\ h_{k,i}(r) & \text{for } h_{k,i}(r) < CL_{k,i} \end{cases} . \end{aligned} \tag{7.13}$$

To calculate $CL_{k,i}$ in accordance with Eq. (7.12), the following iterative algorithm is proposed [10]:

Let $CL_{k,i} = x$, with initial values $x = 0$, and $= \delta 0.01$(experimentally set).

Step 1.$x = x + \delta$;.

Step 2.$D(x) = \sum_{r=0}^{m''} [h_{k,i}(r) - x]$ for $h_{k,i}(r) \geq x$;

Step 3.If $D(x)$ $\begin{cases} > 0.5 & \text{return in step 1,} \\ < 0.5, \text{ then } x = x - \delta \text{ and return in step 2,} \\ \approx 0.5 & \text{go to step 4;} \end{cases}$.

Step 4. Stop and set $\mathrm{CL}_{k,i} = x$..

– to accelerate the calculation of $CL_{k,i}$, the image is divided into square sub-blocks. For each sub-block are calculated the local histogram $h_{k,i}(r)$ of the color vectors' modules and the adaptive threshold $CL_{k,i}$, regarding the corresponding central element, $M_C(k, i)$. For each of the remaining matrix elements $[M_C(k, i)]$, an individual threshold is calculated through bilinear interpolation, by using the thresholds calculated for the central elements of the neighbor sub-blocks in horizontal and vertical directions.
– after the threshold $CL_{k,i}$ is calculated, the modified local histogram is defined, in accordance with the relation:

$$h^M_{k,i}(r) = (1/m') \sum_{i=0}^{m''} h'_{k,i}(i) + h''_{k,i}(r) \tag{7.14}$$

where $m' = \left\lfloor 256\sqrt{3} + 0.5 \right\rfloor = 443$, for $m = 256$.

7.5.3 Local Cumulative Histogram

The local cumulative histogram of the vectors' modules $\mathbf{C}_{k,i}$ framed by the window, is calculated:

$$H_{M_C}^{k,i}(r) = \sum_{l=0}^{r} h_{k,i}^{M}(l) = [(r+1)/m'] \sum_{l=0}^{m''} h'_{k,i}(l) + \sum_{l=0}^{r} h''_{k,i}(l) = \begin{cases} (r+1)/m' \text{ for } h_{k,i}(r) \geq CL_{k,i}; \\ \sum_{l=0}^{r} h_{k,i}(l) \text{ for } h_{k,i}(r) < CL_{k,i}; \end{cases} \tag{7.15}$$

for $r = 0,1,2,\ldots, m''$

From the above relation, it follows that the local cumulative histogram $H_{M_C}^{k,i}(r)$ is a linear function of the current value of r. This is why, for the above-threshold area, full equalization of the modified local histogram $h_{k,i}^{M}(r)$ is achieved, which does not depend on its distribution. In the sub-threshold area, however, the equalization of $h_{k,i}^{M}(r)$ depends on the histogram $h_{k,i}(r)$ and sometimes it is not full.

7.5.4 Computational Cost

The computational cost [10] of the operations, needed to define the cumulative histogram $H_{M_C}^{k,i}(r)$, is reduced through recursive calculation of $h_{k,i}^{M}(r)$:

$$h_{k+1,i}^{M}(r) = h_{k,i}^{M}(r) - h_{k-d,i}^{M}(r) + h_{k+d+1,i}^{M}(r) \text{ for r} = 0, 1, 2, .., \tag{7.16}$$

After summing up for both sides of Eq. (7.16), is obtained:

$$H_{M_C}^{k+1,i}(r) = H_{M_C}^{k,i}(r) - H_{M_C}^{k-d,i}(r) + H_{M_C}^{k+d+1,i}(r) \tag{7.17}$$

7.6 Applications of the Local Cumulative Histograms

7.6.1 Local Contrast Enhancement

The local contrast of the color image is enhanced by using the local cumulative histogram $H_{M_C}^{k,i}(r)$, in correspondence with the relation:

$$g_r(k,i) \;=\; \begin{cases} 442 \text{ for } \left\lfloor 442\ H_{M_C}^{k,i}(r) + 0.5 \right\rfloor \;>\; 442; \\ \left\lfloor 442\ H_{M_C}^{k,i}(r) + 0.5 \right\rfloor \text{ - in all other cases,} \end{cases} \tag{7.18}$$

for $r = 0,1,\ldots,442$, when $m = 256$.

Here $g_r(k,i)$ is the new value for the element (k,i), which replaces the original value r.

To avoid false contours appearance in the processed image, it is supposed here to increase the number of bits used for image elements' coding: for example, if the values r of the elements $M_C(k,i)$ in the original matrix were coded with 10 bits, after the processing, 12-bits coding for the new elements, $z_r(k,i)$ is supposed to be used:

$$\begin{aligned} &z_r(k,i) = \left\lfloor m'' \times H_{M_C}^{k,i}(r) + 0.5 \right\rfloor \\ &\text{for } r = 0, 1, 2, \ldots, m''; \ \left(m'' = \lfloor 1.73(m-1) + 0.5 \rfloor\right) \end{aligned} \tag{7.19}$$

In this case, the number of levels $m = 2^{12} = 4096$ and $m'' = \lfloor 1.73.4095 + 0.5 \rfloor = 7086$, correspondingly.

7.6.2 Calculation of the RGB Vectors of the Enhanced Image

For this, the following relation is used:

$$\mathrm{C}_{\kappa,i}^{E} = \left[R_{\kappa,i}^{E}, G_{\kappa,i}^{E}, B_{\kappa,i}^{E}\right]^{T} \text{ for k/i = 1, 2, } \ldots, \text{M/N} \tag{7.20}$$

where $R_{\kappa,i}^{E} = \frac{z_r(k,i)}{M_C(k.i)}\mathrm{R_{k,i}}; G_{\kappa,i}^{E} = \frac{z_r(k,i)}{M_C(k.i)}\mathrm{G_{k.,}}; B_{\kappa,i}^{E} = \frac{z_r(k,i)}{M_C(k.i)}\mathrm{B_{k,i}}$,

for $r = 0,1,\ldots, m''$

7.6.3 Adaptive Color Correction

The adaptive color correction of vectors $\mathbf{C}_{k,i}^{E}$ is done on the basis of the vectors, whose ends are out of the RGB cube. For this, the following steps are performed:

Step 1. For all $\mathbf{C}_{\kappa,i}^{E}$ vectors is checked if their ends are out of the RGB cube, and if there is at least one of their components, whose magnitude is larger than the maximum value, $m - 1$. After that is checked if the condition $\mathrm{N_C} \geq \delta$ is satisfied, in which $\mathrm{N_C}$ is the number of color vectors $\mathbf{C}_{\kappa,i}^{E}$ which are out of the RGB cube, and δ is the pre-selected threshold. If these two conditions are satisfied simultaneously, the vector $\mathbf{C}_{\kappa_0,i_0}^{E} = [R_{k_0,i_0}^{E}, G_{k_0,i_0}^{E}, B_{k_0,i_0}^{E}]^{T}$ is detected, which has at least one component larger than all other components.

Step 2. In case, that the component $R^E_{k_0,i_0}$ of the vector $\mathbf{C}^E_{\kappa_0,i_0}$ is larger than the maximum value, then $R^E_{k_0,i_0} = \max(R^E_{k,i}, G^E_{k,i}, B^E_{k,i}) > m-1$ for $k/i = 0, 1, 2, \ldots, (M-1)/(N-1)$. For the components of this vector, the following correction is done:

$$\begin{aligned} &R^E_{k_0,i_0}(cor) = m-1,\ G^E_{k_0,i_0}(cor) = G^E_{k_0,i_0}/(R^E_{k_0,i_0}/m-1), \\ &B^E_{k_0,i_0}(cor) = B^E_{k_0,i_0}/(R^E_{k_0,i_0}/m-1). \end{aligned} \tag{7.21}$$

The brightness of the pixel (k_0,i_0), in which color vector $\mathbf{C}^E_{k_0,i_0}(cor)$ is corrected, is defined by the relation:

$$\begin{aligned} &Y^E_{i_0,j_0}(cor) = 0.21 \times (m-1) + 0.72\,(m-1)\,(G^E_{i_0,j_0}/R^E_{i_0,j_0}) \\ &+ 0.07\,(m-1)\,(B^E_{i_0,j_0}/R^E_{i_0,j_0}) \end{aligned} \tag{7.22}$$

Taking into account that (in accordance with [10, 11]), the brightness of the pixel (k_0,i_0) in the original image is defined by the relation: $Y_{k_0,i_0} = 0.21\ R_{k_0,i_0} + 0.72\ G_{k_0,i_0} + 0.07\ B_{k_0,i_0}$, to retain the brightness of this pixel in the improved image, must satisfy the equation:

$$Y_{k_0,i_0} = Y^E_{k_0,i_0}(cor) = 0.21R^{\mathrm{E}}_{k_0,i_0}(cor) + 0.72G^{\mathrm{E}}_{k_0,i_0}(cor) + 0.07B^{\mathrm{E}}_{k_0,i_0}(cor), \tag{7.23}$$

from which follow the relations:

$$\begin{aligned} &R^{\mathrm{E}}_{k_0,i_0}(cor) = m-1, G^{\mathrm{E}}_{k_0,i_0}(cor) = (m-1)(G^E_{k_0,i_0}/R^E_{k_0,i_0}), \\ &B^{\mathrm{E}}_{k_0,i_0}(cor) = (m-1)(B^E_{k_0,i_0}/R^E_{k_0,i_0}). \end{aligned} \tag{7.24}$$

For the remaining vectors $\mathbf{C}^E_{k,i} = [R^E_{k,i}, G^E_{k,i}, B^E_{k,i}]^T$, to retain the brightness of their pixels (k,i) for $k/i = 0,1,2,..,(M-1)/(N-1)$, the components of the corresponding corrected vectors $\mathbf{C}^E_{k,i}(cor)$ must be calculated accordingly:

$$\begin{aligned} &R^E_{k,i}(cor) = (m-1)(R^E_{k,i}/R^E_{k_0,i_0}),\ G^{\mathrm{E}}_{k,i}(cor) \\ &= (m-1)(G^E_{k,i}/R^E_{k_0,i_0}),\ B^{\mathrm{E}}_{k,i}(cor) = (m-1)(B^E_{k,i}/R^E_{k_0,i_0}). \end{aligned} \tag{7.25}$$

Step 3. In case, that the maximum value of one of the components $G^E_{k_0,i_0}$ or $B^E_{k_0,i_0}$ is higher than $m-1$, it should be corrected in a way, similar to the correction done for the maximum component, $R^E_{k_0,i_0} > m-1$. The correction of the components of the remaining color vectors $\mathbf{C}^E_{k,i}$ is done by analogy with Eq. (7.25).

To illustrate the presented algorithm, in Fig. 7.3a are shown the original medical tensor R, G, B images, and in Fig. 7.3b—same images, after local contrast enhancement with a sliding window of size 33 × 33 (experimentally set).

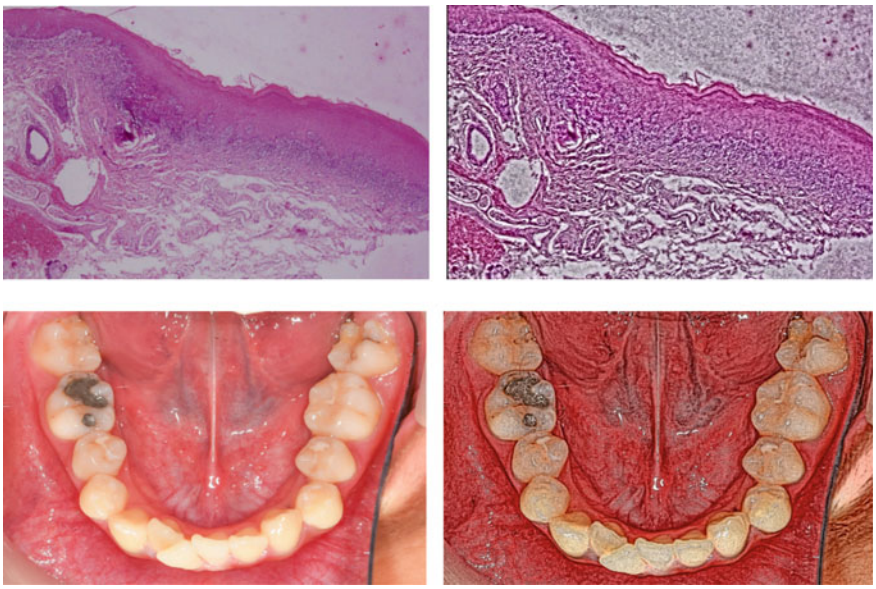

Fig. 7.3 Medical R, G, B images: **a** originals; **b** after local contrast enhancement with a sliding window $W_{k,i}$ of size 33 × 33

7.7 Locally-Adaptive Filtering for Tensor Images, Represented as Vector Fields

In this section, some of the well-known algorithms [1–6, 9, 12–14] for locally-adaptive linear and non-linear filtering are investigated for the case, when they are applied on the matrix $[M_C(k,i)]$ of size $M \times N$, if its elements are quantized at m' levels.

7.7.1 Linear 2D Filtering

The linear 2D filtering of color tensor image is represented through the modules of the color vectors framed by the sliding window $W_{k,l}$, of size $(2b+1) \times (2h+1)$:

$$F_C(k,i) = M_C(k,i) * f(k,i) = \sum_{s=-b}^{b} \sum_{l=-h}^{h} M_C(k+s, i+l) f(s,l) \qquad (7.26)$$

for $k/i = 0,1,\ldots, (M-1)/(N-1)$, where $f(s,k)$ denotes the kernel of the 2D filter, defined in the window $W_{k,i}$; $F_C(k,i)$ is the filtered value of the elements $M_C(k,i)$,

framed by the window, and "$*$" denotes the operator for 2D convolution of $M_C(k, i)$ and $f(s,k)$. The kind and the size of the kernel $f(s,k)$ determine the filtration result.

- To achieve higher sharpness (saturation changes) in the image, is used the algorithm, based on the Laplacian operator. The following assumptions are set.

  Let:

$$\begin{aligned} F_C(k,i) &= (1+4\alpha)M_C(k,i) - \alpha\,[M_C(k-1,i) \\ &+ M_C(k+1,i) + M_C(k,i-1) + M_C(k,i+1)], \end{aligned} \tag{7.27}$$

where the window $W_{k,i}$ is of size 3×3 (for $b = h = 1$). If $= \alpha 1$ is set, the filter kernel is defined by the corresponding matrix,

$$[f_\alpha(k,i)] = \begin{bmatrix} 0 & -\alpha & 0 \\ -\alpha & (1+4\alpha) & -\alpha \\ 0 & -\alpha & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}.$$

In the general case, the sharpness increase is achieved through the algorithm for adaptive unsharp masking, in accordance with which:

$$F_C(k,i) = \begin{cases} F'_C(k,i) & \text{for} |M_C(k,i) - \overline{M_C(k,i)}| \geq \delta, \\ M_C(k,i) & \text{in other cases,} \end{cases} \tag{7.28}$$

where $F'_C(k,i) = (1+\alpha)M_C(k,i) - \alpha\overline{M_C(k,i)}$; $0 < \alpha \leq 1$; δ is the threshold value, and $\overline{M_C(k,i)}$ is the mean value of the elements, framed by the sliding window $W_{k,i}$ of size $(2h+1)(2b+1)$:

$$\overline{M_C(k,i)} = \frac{1}{\beta}\sum_{s=-b}^{b}\sum_{l=-h}^{h} M_C(k+s, i+l) \text{ for } \beta = (2h+1)(2b+1) \tag{7.29}$$

The algorithm, described above, is illustrated by Fig. 7.4: in Fig. 7.4a is shown the original image, and in Fig. 7.4b—the result of the local sharpness enhancement with a sliding window of size 3×3.

To accelerate the calculation of $\overline{M_C(k,i)}$, two-dimensional recursion is used, based on the relation:

$$\begin{aligned} \overline{M_C(k,i)} &= \overline{M_C(k-1,i)} + \overline{M_C(k,i-1)} \\ &- \overline{M_C(k-1,i-1)} + (1/\beta)[M_C(k+b,i+h) \\ &- M_C(k+b,i-h-1) - M_C(k-b-1,i+h) \\ &+ M_C(k-b-1,i-h-1)]. \end{aligned} \tag{7.30}$$

As a result, the mean value $\overline{M_C(k,i)}$ is recursively calculated by only 7 operations instead of adding all pixels in the sliding window. In the last case, the number of

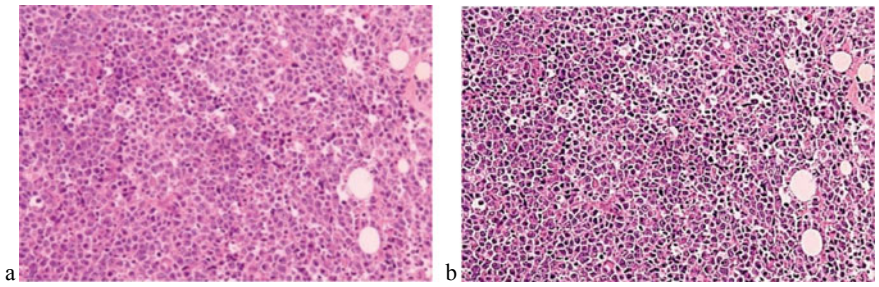

Fig. 7.4 Medical R,G,B image: **a** original; **b** same image, after local sharpness enhancement for a filter kernel of size 3×3

needed operations is $(2b+1) \times (2h+1) - 1$ (for example, if $b = h = 5$, we get $(2b+1) \times (2h+1) - 1 = 120$, and then the acceleration of the calculations $\overline{M_C(k,i)}$ is 120/7 = 17.14 times).

The recursive calculation of $F'_C(k.i)$ in the relation (7.28) for the unsharp masking, is executed in correspondence with the equation:

$$\begin{aligned} F'_C(k.i) &= (1+\alpha)M_C(k,i) - \alpha[\overline{M_C(k-1,i)} \\ &+ \overline{M_C(k,i-1)} - \overline{M_C(k-1,i-1)}] - (\alpha/\beta)[M_C(k+b,i+h) \\ &- M_C(k+b,i-h-1) - M_C(k-b-1,i+h) \\ &+ M_C(k-b-1,i-h-1)]. \end{aligned} \tag{7.31}$$

- The locally-adaptive image filtering aimed at the additive Gaussian noise reduction, is performed in accordance with the relation:

$$F_C^f(k,i) = \begin{cases} \overline{M_C(k,i)} + \frac{\sigma^2_{M_C}(k,i) - v^2}{\sigma^2_{M_C}(k,i)}[M_C(k,i) - \overline{M_C(k,i)}] & \text{for } \sigma^2_{M_C}(k,i) \geq v^2; \\ \overline{M_C(k,i)} & \text{for } \sigma^2_{M_C}(k,i) < v^2, \end{cases} \tag{7.32}$$

where: $F_C^f(k,i)$ is the filtered element $M_C(k,i)$; $\overline{M_C(k,i)}$ denotes the mean value of the element $M_C(k,i)$ in the sliding window $W_{k,i}$ of size $(2b+1) \times (2h+1)$; $\sigma^2_{M_C}(k,i) = [(1/\beta)\sum_{s=-b}^{b}\sum_{l=-h}^{h} M_C^2(k+s,i+l)] - \overline{M_C(k,i)}$ is the local variance of the element $M_C(k,i)$ in the sliding window $W_{k,i}$; $v^2 = (1/M \times N)\sum_{i=1}^{M}\sum_{j=1}^{N}\sigma^2_{M_C}(k,i)$ is the mean noise variance in the input matrix $[M_C(k,i)]$. To accelerate the calculation of the mean value $\overline{M_C(k,i)}$ for all elements, except those on the first row and first column of the input matrix $[M_C(k,i)]$, recursive relation is used in accordance with Eq. (7.30). When compared to the well-known adaptive Wiener filter [12], the main advantage of the new filter is, that it is adaptive to the relation of the local variation to the global one, in result of which the transitions in the image are retained, and the noise is suppressed.

7.7.2 Weighted Median Filtering

The weighted median filtering of pulse noises in the color tensor image, represented through the matrix $[M_C(k,i)]$ of the color vectors' modules, framed by the sliding window $W_{k,l}$ of size $(2b+1)\times(2h+1)$, is executed in accordance with the relation:

$$F_C^{WM}(k,i) = Med_W M_C(k,i) = Med\,[t(s,l)\times M_C(k+s,i+l) : s,l \in W_{k,i}], \tag{7.33}$$

where s = - b, - b + 1,.., 0,..,b − 1,b; l = - h, - h + 1,.., 0,..,h − 1,h. The coefficients $t(s,l)$ show how many times appears the corresponding element $M_C(k+s, i+l)$ in the monotonic increasing sequence x_p for $p = 1,2,\ldots,P$, from which is defined the weighted median function, x_{WM}:

$$x_1 \le x_2 \le \ldots \le x_{WM} \le \ldots x_{P-1} \le x_P \text{ forP} = (2b+1)'(2h+1). \tag{7.34}$$

The filtered element (k,i) corresponds to the weighted median $F_C^{WM}(k,i) = x_{WM}$. In accordance with Eq. (7.33), the sum of all elements $t(s,l)$ is an odd number, and the elements x_p in Eq. (7.34) are defined by the modules $M_C(k+s, i+l)$ framed by the window $W_{k,l}$, after rearrangement into an increasing monotonic sequence. The size of the window is defined so as to frame the noise elements of average size, which should be filtered.

The algorithm is illustrated in Fig. 7.5: in Fig. 7.5a is shown the noise image, and in Fig. 7.5b—the filtered image.

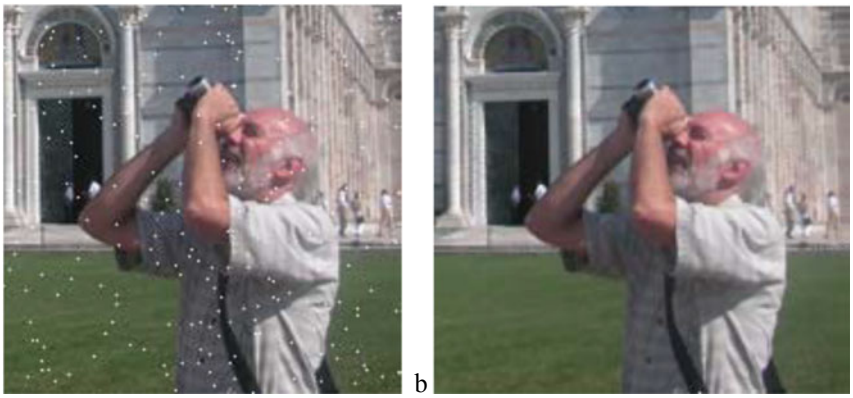

Fig. 7.5 Test R, G, B image: **a** The image with 1% additive pulse noise; **b** same image, after weighted median filtering by using a window of size 3 × 3 (for $b = h = 1$)

7.7.3 Suppression of Additive Pulse and Gaussian Noises

To suppress the additive pulse and Gaussian noises in a color image, and to retain the existing transitions, the algorithm for vector median filtering is used [13]. For this, from the color vectors placed in the sliding window $W_{k,l}$ (Fig. 7.1), the sequence $\mathbf{C}_1, \mathbf{C}_2, .., \mathbf{C}_p, .., \mathbf{C}_{P-1}, \mathbf{C}_P$ is composed. For each vector from the sequence $\mathbf{C}_p = [R_p, G_p, B_p]^T$ are calculated the distances D_p to all remaining vectors, $\mathbf{C}_j = [R_j, G_j, B_j]^T$, framed by the window $W_{k,l}$, i.e.:

$$D_p = \sum_{j=1}^{P} \sqrt{(R_p - R_j)^2 + (G_p - G_j)^2 + (B_p - B_j)^2} \text{ for p} \neq \text{j and p/j} = 1, 2, .., \text{P}. \tag{7.35}$$

The so calculated distances D_p are arranged as an increasing monotonic sequence, $D_1 \leq D_2 \leq \ldots \leq D_P$. The index p_0 of the filtered color vector $\mathbf{C}_{p_0} = [R_{p_0}, G_{p_0}, B_{p_0}]^T$, which replaces the vector $\mathbf{C}_{k,i}$ in the center (k,i) of window, is defined by the condition:

$$D_{p_0} = \min\{D_j\} \text{ forj} = 1, 2, .., \text{P}. \tag{7.36}$$

7.7.4 Morphological Filtration

For the morphological filtration of the color image A (represented as the matrix $[M_C(k, i)]$), is used the "flat" structuring element B with components $b(s,l)$, defined in a sliding window of size $(2b + 1) \times (2h + 1)$. The filtration is executed by using the following basic morphologic operators [6, 14]:

- morphological dilatation and erosion:

$$D(A, B) = \max [M_C(k - s, i - l) + b(s, l)] = A \oplus B \tag{7.37}$$

$$E(A, B) = \min [M_C(k + s, i - +l) - b(s, l)] = A \Theta B \tag{7.38}$$

where $s = -b, -b + 1, .., 0, .., b - 1, b$; $l = -h, -h + 1, .., 0, .., h - 1, h$.

- morphological opening and closing:

$$OP(A, B) = (A \Theta B) \oplus B = AoB = D\{E(QA, B), B\} \tag{7.39}$$

$$CL(A, B) = (A \oplus B) \Theta B = A \cdot B = E\{D(QA, B), B\} \tag{7.40}$$

7.7.4.1 Operator for Morphological Noise Suppression

The operator for morphological noise suppression is represented as:

$$F_{sm} = Morph\ \ Smooth\ (A, B) = CL\{OP(A, B), B\} = (AoB) \cdot B \qquad (7.41)$$

7.7.4.2 Morphological Gradient Operators for the Outer and Inner Contours

The morphological gradient operators for detection of the outer and inner contours of the objects are represented as follows:

$$F_{ext} = DG(A) = D(A, B) - A;\ \ and\ \ F_{\text{int}} = EG(A) = A - E(A, B). \qquad (7.42)$$

7.7.4.3 Morphological Gradient Operators of Laplace, Bother, Li^1 and Li^2

The morphological gradient operators of Laplace, Bother, Li^1, and Li^2 for contours detection are represented as follows:

$$F_{Lap} = DG(A) - EG(A);\ \ F_{Both} = DG(A) + EG(A); \qquad (7.43)$$

$$F_{Li}^1 = \min[DG(A), EG(A)];\ \ F_{Li}^2 = \max[DG(A), EG(A)]. \qquad (7.44)$$

7.7.4.4 Morphological Operators Top Hat and Bot Hat

The morphological operators Top Hat and Bot Hat for detection of dark/light objects on an irregular background, are defined as given below:

$$Top\ Hat\ (A, B) = CL(A, B) - A;\ \ Bot\ Hat\ (A, B) = A - OP(A, B). \qquad (7.45)$$

7.7.4.5 Morphological Operator for Sharpness Enhancement

The morphological operator for sharpness enhancement and for contrast enhancement of the small details in the color image, respectively, is defined by the relation below:

$$F_{ms} = Morph\ \ Sharpness\ \ (A, B) = A + [A - (AoB)] - [(A \cdot B) - A]. \tag{7.46}$$

7.7.4.6 Segmentation of Color Textures

The segmentation of color texture images is based on morphological filtration. The following assumptions are set:

Let the color image A, represented by the matrix $[M_C(k, i)]$ contains two different textures, each built by repetitive elements of different average sizes. To detect the border, the following three steps are performed:

$$\text{Step 1}: \ S_1^n = E(A, nB_1) = (((A\Theta B_1)\Theta B_1)....)\Theta B_1 = A\Theta nB_1 \tag{7.47}$$

where n denotes the number of erosions, and B_1 is the structuring element. Its shape is chosen so that after the n-th erosion, the elements of one of the two textures disappear, merging into a homogenous area, i.e.:

$$\text{Step 2}: \ S_2^m = D(S_1^m, mB_2) = (((S_1^m \oplus B_2) \oplus B_2)....) \oplus B_2 = S_1^m \oplus mB_2, \tag{7.48}$$

where m denotes the number of dilatations, and B_2 is the structuring element. Its shape is chosen so that after dilatations, the elements of the second texture disappear, merging into a homogenous area, i.e.:

$$Step\ 3: \ S = S_2^m - (S_2^m \Theta B_1), \tag{7.49}$$

where S is a color image, which contains the border between both textures, of width one pixel only.

The shape and the size of the structuring elements B_1 and B_2 conform to the mean values of the corresponding elements in the first and second textures. In this way, the number of erosions and dilatations needed to execute steps 1 and 2, is reduced. In particular, if $B_1 = B_2 = B$, the values of n and m increase, but the operations needed for steps 1 and 2, are simpler.

In case that the color image contains more than two different textures, the presented morphological algorithm for detecting the border between neighbor textures is applied repeatedly, depending on the number of textures.

After each dilatation, needed for the morphological filtration or segmentation, should be evaluated the number of color vectors, whose ends are out of the color cube. In case that this number is higher than the preset threshold, color and brightness correction of the vector field is needed, similar to that from Sect. 7.6 (the algorithm for local contrast enhancement).

The algorithm for morphological filtration is illustrated in Fig. 7.6, where in Fig. 7.6a is shown the original color image. It contains two textures with similar

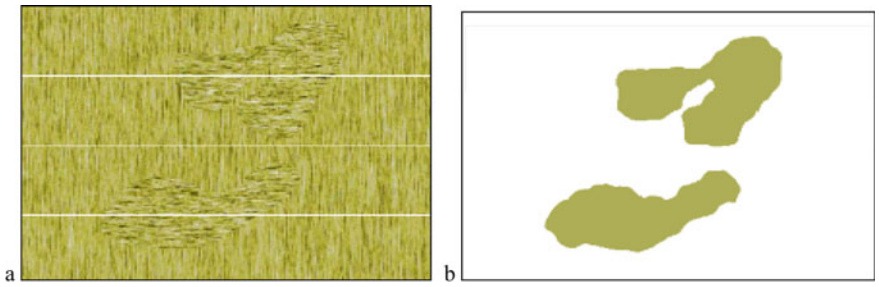

Fig. 7.6 Color texture image: **a** original; **b** after morphological segmentation by using a flat structuring element B, shaped as a horizontal/vertical line and placed in a window of size 9 × 9

color characteristics, but with different orientations of their elements (horizontal and vertical). In Fig. 7.6b is shown the result obtained after a morphological segmentation with a flat structuring element B, whose shape is a line (horizontal/vertical), defined in a window of size 9 × 9. As a result of the segmentation, the elements of both textures merge into two homogenous areas. If the color image contains two textures, which comprise similar elements, distinguished by their hue, similar algorithm could be used. However, in this case, the algorithm for morphological segmentation should be applied on the matrix $[\theta_C(k,i)]$ of the angles (orientation) of the color vectors $\mathbf{C}_{k,i}$ instead of the matrix $[M_C(k,i)]$, composed by the modules of these vectors.

7.8 Conclusions

The objective of this work is to formulate one new approach for locally-adaptive processing of color third-order tensor RGB images represented in a vector form, and to analyze the characteristics of the corresponding basic algorithms so that to exploit efficiently the spatial and inter-channel correlation. These algorithms need only operations executed on a single matrix image, which comprises the modules of the corresponding color vectors. As a result, triple reduction of the computational complexity is achieved, compared to algorithms executed on each color component individually. Together with the reduced computations, the hue and the brightness of the original image are retained.

The presented locally-adaptive algorithms are extremely efficient in the processing of medical images, which have variable color characteristics (brightness, saturation and hue). The new approach will be further investigated and extended, aiming at applications in various multidisciplinary areas. The future development of the locally-adaptive algorithms will be mainly aimed at the integration with deep neural networks with different architectures, so as to achieve higher flexibility in the adaptive processing of color images.

Acknowledgements This work was funded by the Bulgarian National Science Fund: Project No. KP-06-H27/16: "Development of efficient methods and algorithms for tensor-based processing and analysis of multidimensional images with application in interdisciplinary areas".

References

1. Thyagarajan, K.: Digital Image Processing with Application to Digital Cinema. Focal Press, Elsevier (2006)
2. Celebi, M., Lecca, M., Smolka, B. (eds.): Color Image and Video Enhancement. Springer, Heidelberg (2015)
3. Jain, A.: Fundamentals of Digital Image Processing. Prentice Hall, Englewood Cliffs, NJ (2018)
4. Gomez-Agis, J., Kober, V.: Local adaptive image processing in a sliding transform domain. In: Proc. SPIE 6696, Applications of Digital Image Processing, 669623, 24 (2007). https://doi.org/10.1117/12.735044
5. Pratt, W.: Digital Image Processing. John Wiley & Sons Inc., Publication (2007)
6. Gonzales, R., Woods, R.: Digital Image Processing, 4th edn, Pearson Education (2019)
7. Renhard, E., Ward, G., Pattanaik, S., Devebec, P., Heidrich, W., Myszkowski, K.: High Dynamic Range Imaging: Acquisition, Display, and Image-based Lighting. Morgan Kaufmann Publications, Elsevier (2010)
8. Inoue, K., Jiang, M., Hara, K.: Hue-preserving saturation improvement in RGB color cube. J Imaging **7**, 150 (2021). https://doi.org/10.3390/jimaging7080150
9. Rao, K., Kim, D., Hwang, J.: Fast Fourier transform: algorithms and applications. Springer (2010)
10. Kountchev R, Bekiarsky A, Mironov R, Bekiarska S.A.: Method for local contrast enhancement of endoscopic images based on color tensor transformation into a matrix of color vectors' modules using a sliding window. MDPI Symmetry **6**, 14(12), 2582, Open access. https://doi.org/10.3390/sym14122582
11. Recommendation ITU-R BT.709-5 (04/2002): Parameter values for the HDTV standards for production and international programme exchange. BT Series Broadcasting service (television). https://www.itu.int/pub/R-REC/en
12. Jin, F., Fieguth, P., Winger, L., Jernigan, E.: Adaptive wiener filtering of noisy images and image sequences. In: Proceedings of the International Conference on Image Processing, 14–17 Sept. 2003, Barcelona, Spain, https://doi.org/10.1109/ICIP.2003.1247253
13. Lukas, R., Smolka, B.: Application of the adaptive center-weighted vector median framework for the enhancement of cdna microarray images. Int. J. Appl. Math. Comput. Sci. **13**(3), 369–383 (2003)
14. Dougherty E. (Ed.) Mathematical morphology in image processing. CRC Press (2018)