Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2014) 22: 990 – 1006
© TÜBİTAK
doi:10.3906/elk-1207-7

A new edge-preserving algorithm based on the CIE- Lu'v' color space for color contrast enhancement

Mohammad ZOLFAGHARI[1,*], **Mehran YAZDI**[2]
[1]Faculty of Electrical and Computer Engineering, Zarghan Branch, Islamic Azad University, Zarghan, Iran
[2]Faculty of Electrical and Computer Engineering, Shiraz University, Shiraz, Iran

Received: 03.07.2012 • **Accepted:** 17.12.2012 • **Published Online:** 17.06.2014 • **Printed:** 16.07.2014

Abstract: Image segmentation and edge detection are the most important presteps in machine vision, and their successfulness can affect the success of the next steps. In this paper, the performance of the Trahanias edge detector in different color spaces, such as RGB, YCbCr, HSI, and CIE Lu'v', is compared in order to find the best color space for image segmentation and edge detection. We then offer an efficient edge-preserving algorithm for color contrast enhancement in the CIE Lu'v' space. The proposed algorithm can increase the color contrast, which causes a remarkable improvement in image segmentation and edge detection in the CIE Lu'v' color space. Moreover, it can efficiently reduce the number of spurious edges that may be produced during the color contrast enhancement process. The results obtained by applying the proposed algorithm, as compared with those by applying another recently introduced algorithm, demonstrate the better performance of the proposed algorithm.

Key words: Edge detection, color space, color contrast enhancement, edge preserving, image saturation

1. Introduction

In an image segmentation process, an image is usually divided into smaller regions, where the pixels of each region are similar in color, intensity, and so on. The points between 2 regions are called the edge. Thus, the edge detection process can be used as a beneficial prestep in image segmentation. Different image segmentation algorithms based on thresholding [1], splitting and merging [2], and data clustering techniques [3] have been proposed and various edge detection algorithms such as the Canny, Sobel, and Trahanias [4] have been used, but a common and perfect method does not yet exist and the methods are very sensitive to the types of images.

In addition to the choice of efficient algorithms for image segmentation and edge detection, the type of useful color space in which the algorithms are applied is also important. The most important color spaces normally used for image segmentation and edge detection are RGB, HSI, YCbCr, and CIE Lu'v' [5]. Among the mentioned color spaces, the CIE Lu'v' color space is mostly used in order to increase the image contrast to attain better image segmentation and edge detection. Although the aim of increasing the image contrast is to have better image segmentation, creating spurious edges and missing true edges may occur in this process.

Various methods have been proposed for image contrast enhancement based on using filters, image histograms, chromatic features [6,7], etc. Although these methods improve the image contrast, they are unable to preserve all of the true edges.

Recently, Chung et al. [8] proposed a method of edge-preserving based on the CIE Lu'v' color space.

[*]Correspondence: zolfaghari20@yahoo.com

Although their method has opened a new direction in CIE Lu'v' color space-based saturation, it contains some deficiencies that can be improved.

In this paper, we first compare the results of applying the Trahanias edge detector on test images represented in various color spaces in order to find the best color space for the segmentation and edge detection processes. The reason for using the Trahanias edge detector is that it is significantly more efficient than other edge detectors because of the advantages of using vector order statistics in the edge-detection process [4]. In the next step, to improve the image segmentation results, we increase the contrast of the test images by saturating them in the CIE Lu'v' color space. To do that, we propose an efficient edge-preserving algorithm and compare our algorithm with Chung et al.'s algorithm [8] to demonstrate the superiority of the proposed algorithm.

This paper is organized as follows: In Section 2, the CIE Lu'v' color space and Trahanias edge detector are introduced. Section 3 consists of 2 edge-preserving algorithms, Chung et al.'s [8] and our proposed algorithm. The experimental results are presented in Section 4 and, finally, some conclusions are addressed in Section 5.

2. Preliminaries

In this section, 2 concepts are introduced that will be used in the next sections. These are the CIE Lu'v' color space and Trahanias edge detector.

2.1. The CIE Lu'v' color space

Since color spaces like RGB, HSI, and YCbCr are well known, in this section only the CIE Lu'v' color space is discussed in detail. This color space can be obtained from the RGB color space using Eqs. (1) and (2) [8]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.49000 & 0.31000 & 0.20000 \\ 0.17697 & 0.81240 & 0.01063 \\ 0.00000 & 0.01000 & 0.99000 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}, \tag{1}$$

where R, G, and B are the red, green, and blue components of the RGB color space and X, Y, and Z are the components of the CIE XYZ color space, respectively.

$$u' = \frac{4X}{X + 15Y + 3Z} \quad v' = \frac{9Y}{X + 15Y + 3Z} \quad L = Y \tag{2}$$

L contains the brightness information. The $u'v'$ diagram, which includes the chromatic information, is illustrated as a closed curve in Figure 1, where $R' = (u'_{R'} = 0.4507, v'_{R'} = 0.5229)$, $G' = (u'_{G'} = 0.1250, v'_{G'} = 0.5625)$, and $B' = (u_{B'} = 0.1754, v_{B'} = 0.1579)$ are 3 points of a triangle and point W is defined as the white point.

The points within the triangle are displayable by a cathode ray tube monitor. The points near $W = (u_w = 0.1978,\ v_w = 0.4683)$ are achromatic ones. Hence, pixels for the image saturating process should be within the triangle and adequately far from W.

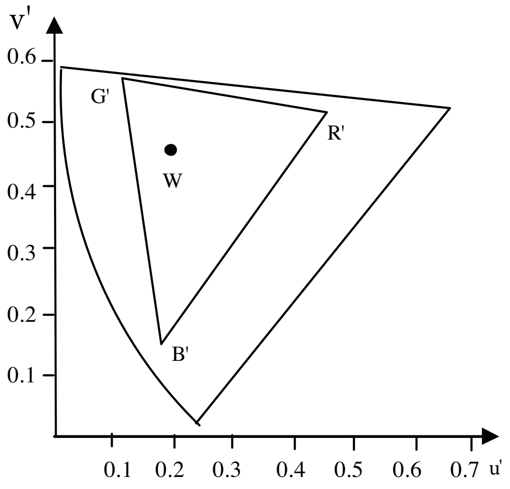

Figure 1. The u$'$v$'$ chromatic diagram.

2.2. Trahanias edge detector

One of the effective edge detectors is the Trahanias detector and we used it in our work for comparing the results. However, other edge detectors can also be used for this purpose, such as those proposed in [9,10].

To apply the Trahanias edge detector, the following steps should be applied on all of the pixels of an image [4,11]:

Step 1. A 3 $\times$ 3 window is selected to scan all of the image pixels. The selected window contains 9 pixels, where each of the pixels is represented by p.

Step 2. For each pixel within the window denoted by p_k, $k=$ 1, 2, 3, ..., 9, we sum up the norm of difference of each pixel with itself and with the other 8 pixels, which is defined as d_k and computed by the following equation:

$$d_k = \sum_{m=1}^{9} \|p_k - p_m\| \quad k = 1,2...,9, \tag{3}$$

where $\|.\|$ represents a vector norm and p_m is another pixel within the window.

Step 3. The obtained values of $d_1, d_2, d_3 \ldots, d_9$ from Eq. (3) are arranged in ascending order. It is supposed that the sorted distances are $d_{(1)}, d_{(2)}, \ldots, d_{(7)}$, where $d_{(1)}$ corresponds to $p_{(1)}$ with a minimum distance and $d_{(7)}$ corresponds to $p_{(7)}$ with a maximum distance.

Step 4. The minimum vector dispersion (MVD) is computed by applying the following equation:

$$MVD = \min_i \left\{ \left\| p_{(10-i)} - \sum_{j=1}^{n} \frac{p_{(j)}}{n} \right\| \right\} \quad i = 1,2,\ldots,k; \quad k,n < 9, \tag{4}$$

where the values of k and n are selected empirically as 3 or 4, which was suggested in [4]. Note that $P(j)$ or $P(i)$ in this equation represents the pixels whose values are sorted from lowest ($j=$ 1) to highest ($j=$ 9). If the MVD value is more than a specified threshold, then the central pixel of the window is an edge pixel; otherwise, it is not an edge pixel. The MVD can improve the efficiency of this edge detector, even in noisy images [4], and that is a reason why it is used in our work.

3. Edge-preserving algorithms

Edge-preserving algorithms can be useful to enhance the results of image segmentation and edge detection algorithms. They would be more efficient if they were applied using a proper color space. Consequently, as a first step, we should determine the more proper color space for edge detection and edge preserving.

Images are usually represented in the RGB color space, so it is necessary to transfer the RGB images to a different color space. Transferring to the CIE Lu'v' color space is done using the method explained in Section 2.1 and transferring to HSI and YCbCr is done using the methods introduced in [12,13].

In the next step, the Trahanias edge detector is used to detect the edges of an image in different color spaces.

The CIE Lu'v' color space can be chosen as an efficient color space in order to saturate the image. In this space, image saturation causes the enhancement of the image segmentation; however, the true edges may be missing and spurious edges may be created in the edge detecting process. A new method that can preserve better edges is Chung et al.'s algorithm. Although this method preserves most of the true edges, it contains 2 deficiencies. First, we explain Chung et al.'s algorithm, and then we discuss its deficiencies.

3.1. Method of Chung et al.

This method contains 3 parts: image saturation, image desaturation, and a new edge-preserving algorithm proposed by Chung et al. [8].

3.1.1. Image saturation

For increasing the image contrast in the CIE Lu'v' color space, the image saturation can be used as shown in Figure 2a. For a pixel like C, which is within the R'G'B' triangle and has adequate distance from W, the maximum amount of saturation can be obtained by moving pixel $C = (u'_c, v'_c, L)$ along line segment $\overline{WC}$ and by finding the intersection point of the line segments of $\overline{WC}$ and $\overline{R'B'}$ [14,15]. This intersection point is called $C_s = (u'_{c_s}, v'_{c_s}, L_{c_s})$, which indicates the maximum amount of saturation for pixel C. Note that this procedure does not affect the brightness of the pixel, so $L_{c_s} = L$.

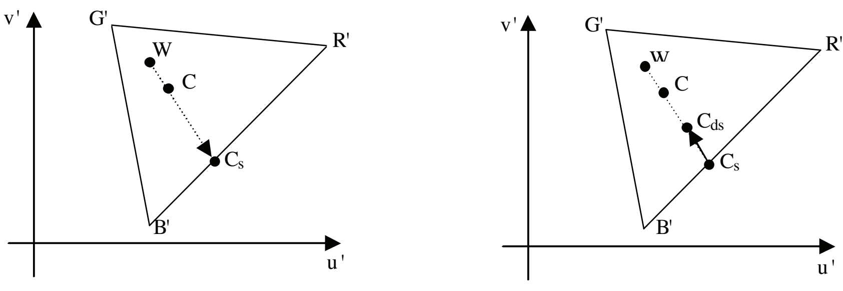

Figure 2. Saturation (a) and desaturation (b) steps.

3.1.2. Image desaturation

Maximizing the amount of saturation for the image pixels causes 2 problems: 1) missing the detection of true edges and creating a number of spurious edges, and 2) increasing the intensity of colors, which means that the image may become visually abnormal. Therefore, a desaturation process is needed. As can be seen in Figure

2b, other papers, such as [8], have determined the components of a desaturated pixel $C_{ds} = (u'_{c_{ds}}, v'_{c_{ds}}, L_{c_{ds}})$ using the central gravity law of color mixture, as defined by:

$$u'_{C_{ds}} = \frac{u'_w \frac{L_W}{v'_W} + u'_{C_s} \frac{L}{v'_{C_S}}}{\frac{L_W}{v'_W} + \frac{L}{v'_{C_S}}} \quad v'_{C_{ds}} = \frac{L_W + L}{\frac{L_W}{v'_W} + \frac{L}{v'_{C_S}}} \quad L_{c_{ds}} = L_W + L, \tag{5}$$

In this equation, L_W is computed separately by $L_W = k\bar{L}$, where $\bar{L}$ is the mean of the color image luminance and k is a factor to control the image luminance and saturation.

3.1.3. Chung et al.'s algorithm [8]

In [8], Chung et al. used the following steps to perform the procedure of edge preserving in the saturated color space.

Step 1. First, the edges of the image in the CIE Lu'v' color space are detected by the Trahanias edge detector. There are 2 cases for each pixel like C: either pixel C is an edge pixel, which is denoted by $E(C) =$ 1, or pixel C is not an edge pixel, which is denoted by $E(C) = 0$.

Step 2. Using the saturating process, each pixel in the CIE Lu'v' color space is saturated and denoted by C_s, and then each saturated pixel is desaturated by Eq. (5) and is denoted by C_{ds}.

Step 3. A 3 × 3 window (see Figure 3) is selected for scanning all of the desaturated image rows horizontally from the previous step. In this window, the symbol p means that the edge preservation procedure is done on these pixels, the symbol c shows that the edge preservation procedure is applied on the central pixel of the window, and the symbol u denotes that the procedure will be done on these pixels.

p	p	p
p	c	u
u	u	u

Figure 3. The 3 × 3 window for scanning the image rows.

In each step of moving the window, the case of the central pixel should be determined by the edge detector. Four cases can occur due to moving each pixel in the original image in the CIE Lu'v' space, like pixel C toward another pixel like C_{ds} in the desaturated image obtained from step 2: 1) $E(C)= 1$ and after transferring pixel C to pixel C_{ds}, $E(C_{ds}) = 1$. In other words, if pixel C is an edge in the original image and by transferring it to pixel C_{es} it is still an edge, no error will occur. 2) $E(C) = 0$ and after transferring, we can write $E(C_{ds})= 0$. Consequently, no error occurs in cases 1 and 2. In cases 3 and 4, the value of $E(C)$is not equal to $E(C_{ds})$. Cases 3 and 4 are noted as error cases.

Color point C_{es} is defined as a point between points C and C_s to have a saturated pixel but without error. Indeed, if transferring pixel C to pixel C_{ds} leads to an error, pixel C_{ds} must be moved to another optimum saturated pixel like pixel C_{es}, which removes the error. Finding pixel C_{es} means we have a saturated pixel with no error.

Step 4. According to the previous step, in cases 1 or 2, we can assume that $C_{es} = C_{ds}$; otherwise, for finding color point C_{es}, the binary alternative search algorithm in the next step should be used.

Step 5. If $E(C) = 1$ and pixel C is transferred to pixel C_{ds}, where $E(C_{ds}) = 0$, it causes the error. According to Figure 4, it is noted that distance d is empirically equal to $\frac{4|\overline{C_sC_{ds}}|}{5}$. For removing the error, first, pixel C$_{ds}$ should be transferred to pixel $C_{t(1)}$, where $E(C_{t(1)}) = 1$ and $|\overline{C_{ds}C_{t(1)}}| = d$. In this case, the error is removed. In the next phase, in order to reach the nearby point C_{ds}, pixel $C_{t(1)}$ should be transferred to pixel $C_{t(2)}$, where $|\overline{C_{t(1)}C_{t(2)}}| = d/2$. If $E(C_{t(2)}) = 1$, then pixel $C_{t(2)}$ is transferred to pixel $C_{t(3)}$ and $C_{es} = C_{t(3)}$; otherwise, it is transferred to pixel $C_{t'(2)}$ and $C_{es} = C_{t'(2)}$. In this step, the aim is to find pixel C_{es}, which should be as close as possible to C_{ds}.

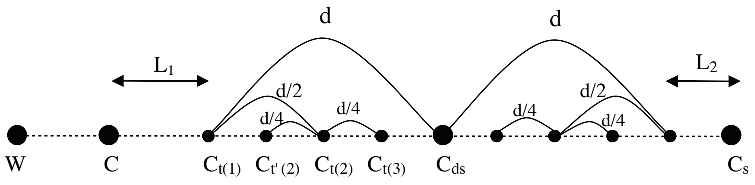

Figure 4. Binary alternative search in Chung et al.'s algorithm [8] for edge preservation.

The same procedure is applied on the pixel on the right side of color point C_{ds} and on the closest pixel to pixel C_{ds} to find C_{es}. If the above procedure cannot find the edge pixel for the central pixel of the window, then $C_{es} = C_{ds}$ is chosen. However, this procedure can be also done for the nonedge pixels that become edge pixels by the saturation step.

3.2. Proposed edge-preserving algorithm

This section presents our proposed algorithm, but before that, the deficiencies in the previous method of edge preserving are discussed briefly.

First, according to Section 3.1.3 and Figure 4, in the case of the error, distances L_1 and L_2 are not considered in the binary alternative search. If color points $C_{t(3)}$, $C_{t(2)}$, and $C_{t'(2)}$ cannot remove the error, pixel C_{es} may be found within these 2 distances (i.e. L_1 and L_2), which are not considered by the algorithm. As a result, we must choose $C_{es} = C_{ds}$, which means that the error still exists. In fact, it is possible that the point being searched for removing the error is between color points C and C_{ds} or between color points C_{ds} and C_s. Second, Eq. (5), which is used for desaturating, does not depend on color point $C = (u'_c, v'_c, L)$. This causes some pixels to be transferred from a color point like C_s to a color point like C_s, between W and C. As shown in Figure 5, in this way, it is not possible to search on the left side.

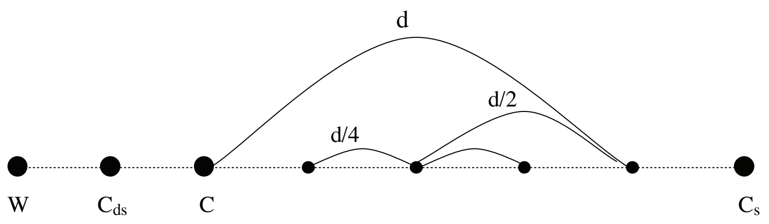

Figure 5. Example of a case where a saturated pixel is moved between W and C instead of being placed between C and C_s.

In our proposed algorithm, for desaturating the image, according to Figure 6, the origin of the coordinate system is transferred to point W, and then color point $C = (u'_c, v'_c, L)$ is transferred to color point $C_{ds} = (u'_{c_{ds}}, v'_{c_{ds}}, L_{c_{ds}})$. In the desaturating process, $L_{c_{ds}} = L$ and components $u'_{c_{ds}}$ and $v'_{c_{ds}}$ are computed by Eqs. (6) through (8):

$$\sqrt{(u'_{C_{ds}} - u'_C)^2 + (v'_{C_{ds}} - v'_C)^2} = d_1, \tag{6}$$

$$v'_{c_{ds}} = m u'_{c_{ds}}, \tag{7}$$

$$d_1 = nd \quad 0 < n < 1, \tag{8}$$

where d_1 is the rate of saturation of pixel C, d is the distance between C and C_s, n is the factor that controls the rate of desaturating, m is the slope of the line segment $\overline{WC}$, and $L_{c_{ds}} = L$. In addition, $d - d_1$ is considered as the rate of desaturation of pixel C$_s$.

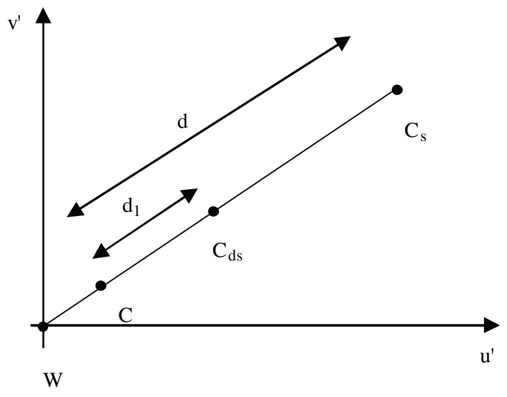

Figure 6. Origin coordinate system transferring for desaturating an image.

The advantage of the above method for desaturating the image is that we are absolutely sure that all of the image pixels move toward color point C_s at a rate of d_1. Using this method, the desaturated pixels like C_{ds} are not placed between W and C, so the second problem described above is solved. According to Figure 7, the first problem is also solved because, in our proposed algorithm, pixel C_{es} is searched for between color points C and C_{ds} and also between color points C_{ds} and C_s.

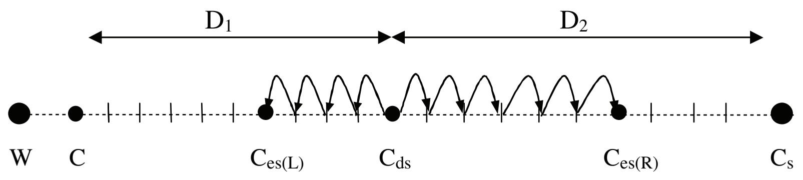

Figure 7. Direct search for edge preservation in our proposed algorithm.

The proposed algorithm can be described in the following steps:

Step 1. After transferring the original image to the CIE Lu'v' color space, edge detection is done on the image.

Step 2. The image is saturated as in Section 3.1.1 and is desaturated by Eqs. (6) through (6).

Step 3. This step is done like the third step in Section 3.1.3.

Step 4. Let us define the central pixel of the window as C. If $E(C) = 1$, it is transferred to C_{ds}, where $E(C_{ds}) = 1$. In another case, if $E(C) = 0$, pixel C is transferred to C_{ds}, where $E(C_{ds}) = 0$. For both cases, no error is produced and $C_{es} = C_{ds}$. However, if we have an error, we should go to the next step for finding color point C_{es}.

Step 5. As shown in Figure 7, color point C_{ds} should be moved from both sides to find color point C_{es}. If $\overline{|CC_{ds}|} = D_1$ and $\overline{|C_sC_{ds}|} = D_2$, distances $D_1$and D_2 are divided into 10 equal intervals experimentally, and then point C_{ds} is moved toward the right side until finding a point like $C_{es(R)}$ between $C_s$and C_{ds}, which removes the error, or reaching point C_s. The same procedure is done on the left side to find a point like $C_{es(L)}$ between C_{ds} and C, which removes the error, or to reach point C.

Four cases can occur in our searching algorithm: 1) If we reach color point C by searching the left side and C_s by searching the right side, then $C_{es} = C_{ds}$, because we could not find a pixel to remove the error; 2) if we reach color points $C$and $C_{s(R)}$, then $C_{es} = C_{es(R)}$, because $C_{es(R)}$ could remove the error; 3) if we reach points C_s and $C_{es(L)}$, then $C_{es} = C_{es(L)}$; otherwise, 4) point C_{es} is chosen between $C_{es(L)}$ and $C_{es(R)}$, which is the nearest point to color point C_{ds} and is selected as pixel C_{es}.

Step 6. All of the previous steps are repeated, but in step 3, the window as shown in Figure 8 is used and this window is moved on the image vertically. It causes a remarkable improvement in the edge preserving of our algorithm.

p	**p**	**u**
p	**c**	**u**
p	**u**	**u**

Figure 8. The 3×3 window for scanning the image columns.

The flow chart in Figure 9 illustrates the various steps of the edge-preserving algorithm. In this flow chart, it is supposed that a pixel is an edge in the original image in the CIE Lu'v' color space ($E(C) = 1$) and then becomes a nonedge pixel by the saturation step ($E(C_{ds}) = 0$). The edge-preserving algorithm is applied to find pixel C_{es}.

4. Experimental results

We performed 2 experiments. In the first part, the best color space, which is useful for the edge-detection process, is determined, and in the second part, the results of edge-preserving algorithms (that of Chung et al. [8] and the proposed algorithm) in saturated CIE Lu'v' color space are compared. For comparing the results in both parts, 6 test images are used, which are shown in Figure 10. Test image Lena is shown in Figure 10a. In Figures 10b–10d test images Peppers, House, and Jelly Beans and in Figures 10e and 10f test images Baboon and Tree are shown, respectively.

In the first part, the test images are transferred into 4 color spaces, according to Sections 2.1 and 3, and then the Trahanias edge detector is used to detect the edges. In Figure 11, the edges of test image Lena in 4 color spaces are shown. Figures 11a and 11b show the edges in the RGB and HSI spaces, respectively. The results of the edge detection process in the YCbCr and CIE Lu'v' color spaces are illustrated in Figures 11c and 11d, respectively.

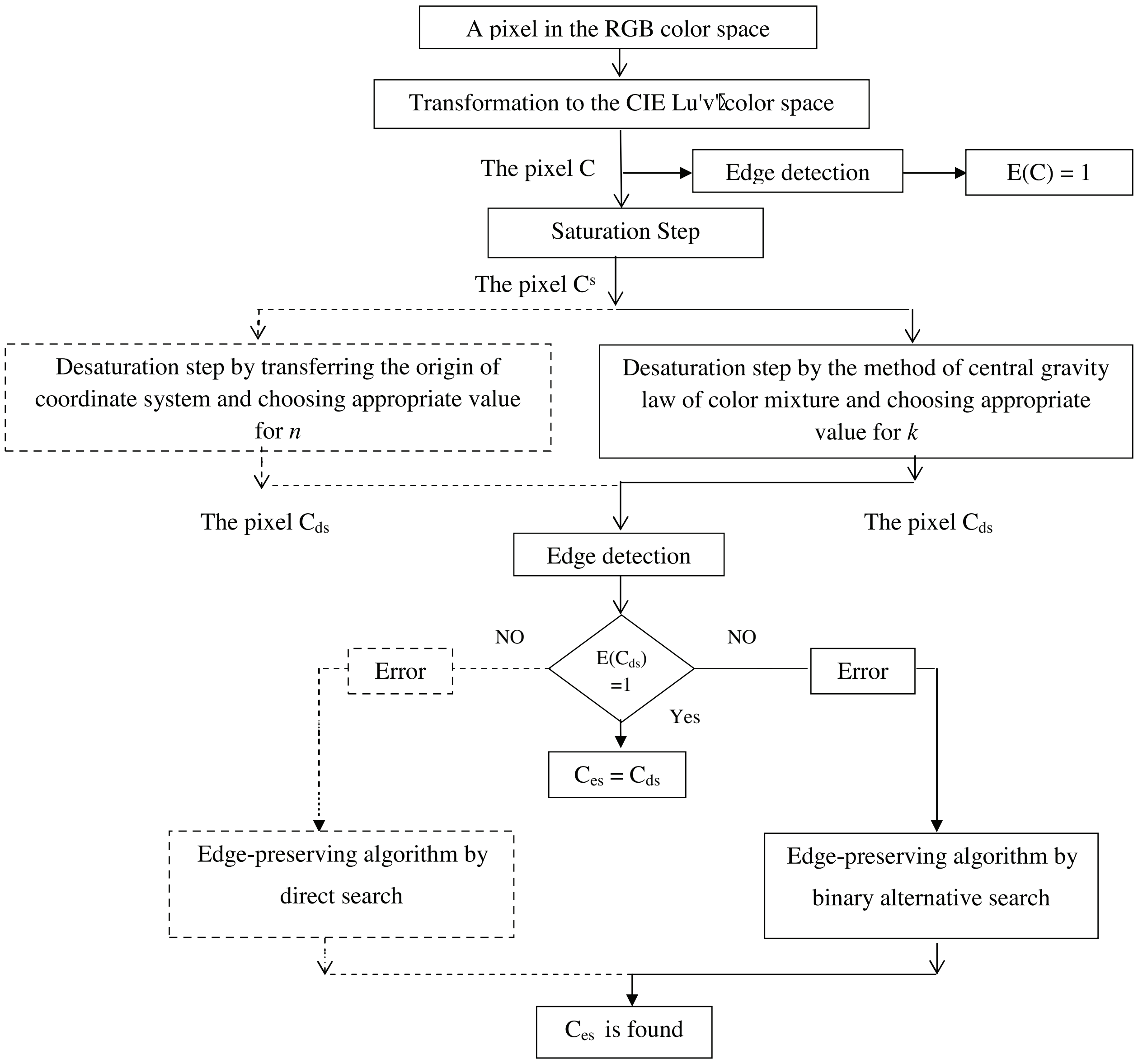
A pixel in the RGB color space
Transformation to the CIE Lu'v' color space
The pixel C
Edge detection
E(C) = 1
Saturation Step
The pixel C^s
Desaturation step by transferring the origin of coordinate system and choosing appropriate value for n
Desaturation step by the method of central gravity law of color mixture and choosing appropriate value for k
The pixel C_{ds}
The pixel C_{ds}
Edge detection
NO
NO
Error
$E(C_{ds}) =1$
Error
Yes
$C_{es} = C_{ds}$
Edge-preserving algorithm by direct search
Edge-preserving algorithm by binary alternative search
C_{es} is found

Figure 9. The edge-preserving algorithm when a pixel is an edge and will become a nonedge pixel with the saturation step; the dashed lines show our modifications with respect to the previous method.

The results show that the best edge detection process is done in the CIE Lu'v' color space, followed by the YCbCr color space. The worst results belong to the RGB color space. The edges in the HSI color space are detected accurately in some parts of the image, but in another parts, it produces very bad results, so the edge detection in the HSI color space is not reliable. Other test images also confirm this conclusion. Figures 12a–12d show the edges in the RGB, HSI, YCbCr, and CIE Lu'v' color spaces, respectively. The edges of test image Baboon in these 4 color spaces are illustrated in Figures 13a–13d and the results of applying the edge detector to test image House can be seen in Figures 14a–14d. In addition, Figures 15a–15d show a comparison between the edges of test image Jelly Beans in the RGB, HSI, YCbCr, and CIE Lu'v' color spaces, respectively. Finally, another test image, named Tree (see Figures 16a–16d), also confirms the conclusion mentioned.

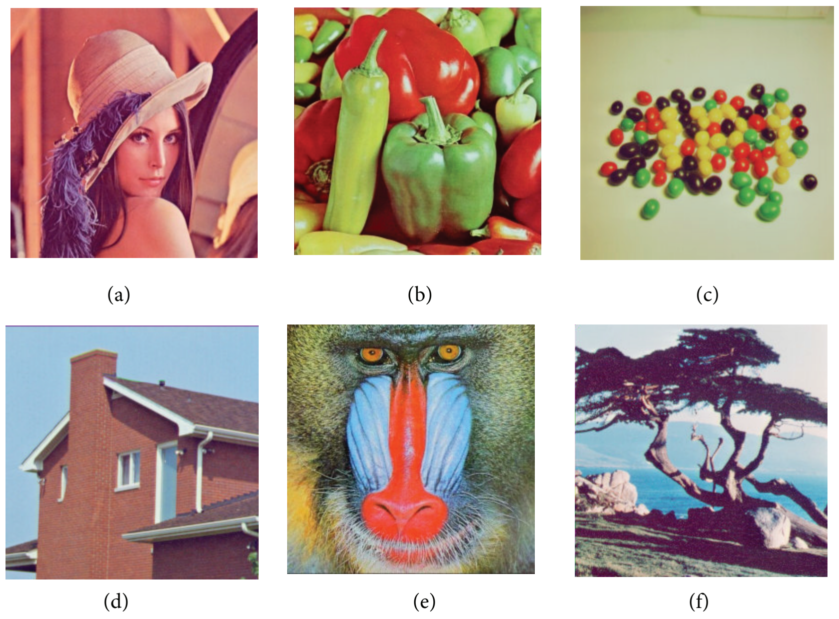

Figure 10. Test images used to evaluate the algorithms: a) Lena, b) Peppers, c) Jelly Beans, d) House, e) Baboon, and f) Tree.

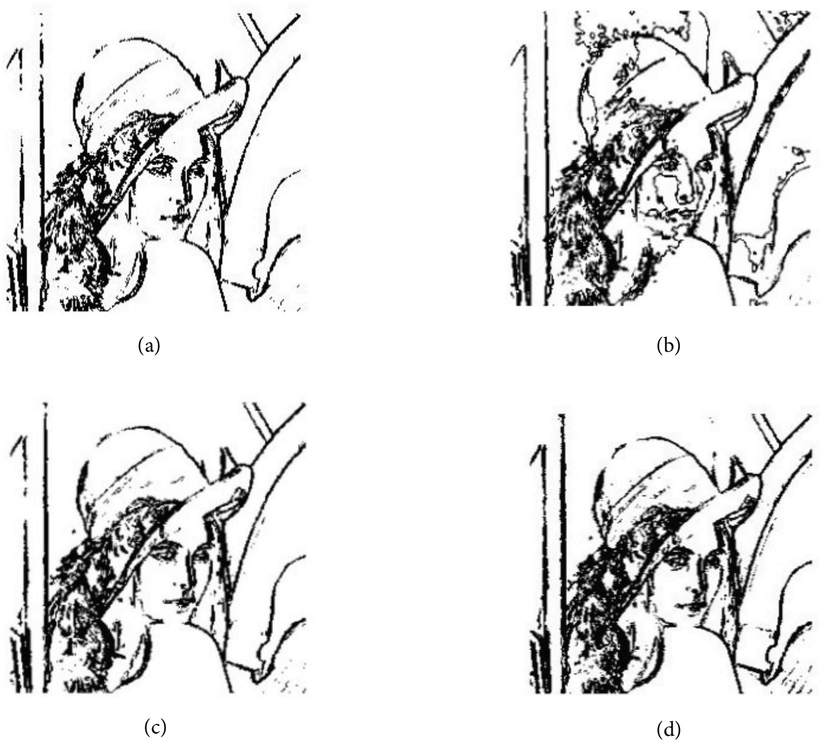

Figure 11. Edges of test image Lena in different color spaces: a) edges in the RGB color space, b) edges in the HSI color space, c) edges in the YCbCr color space, and d) edges in the CIE Lu'v' color space.

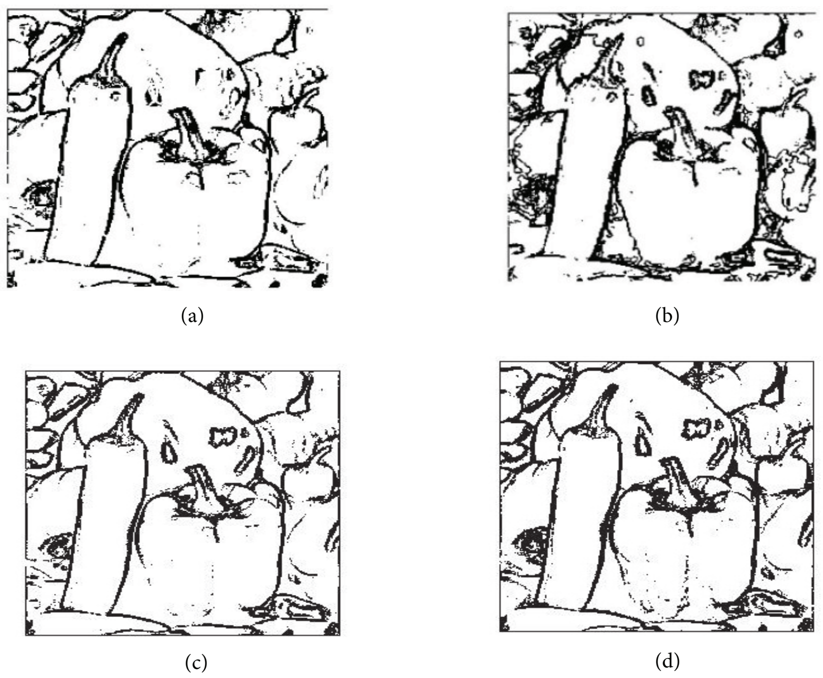

Figure 12. Edges of test image Peppers in different color spaces: a) edges in the RGB color space, b) edges in the HSI color space, c) edges in the YCbCr color space, and d) edges in the CIE Lu'v' color space.

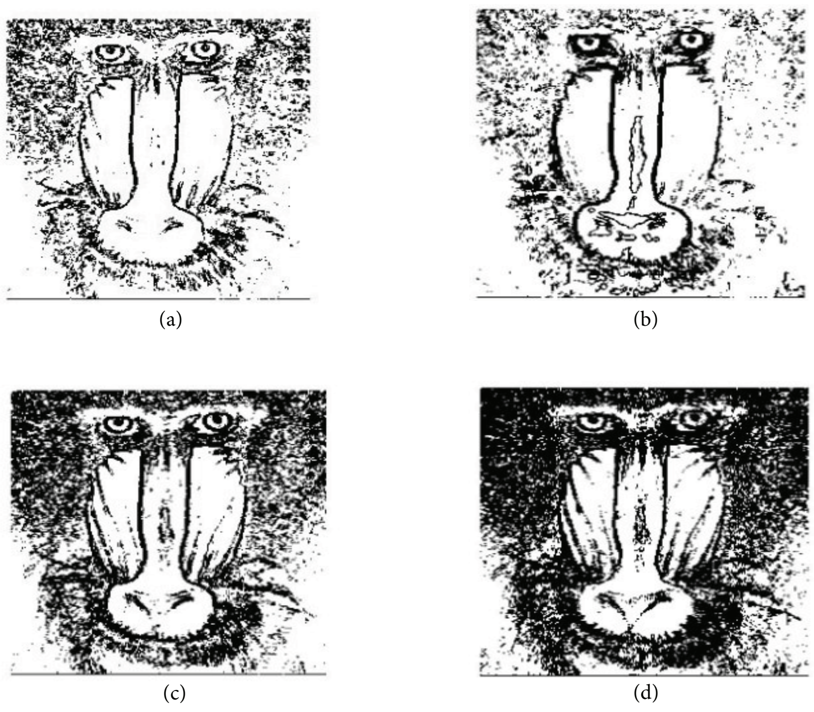

Figure 13. Edges of test image Baboon in different color spaces: a) edges in the RGB color space, b) edges in the HSI color space, c) edges in the YCbCr color space, and d) edges in the CIE Lu'v' color space.

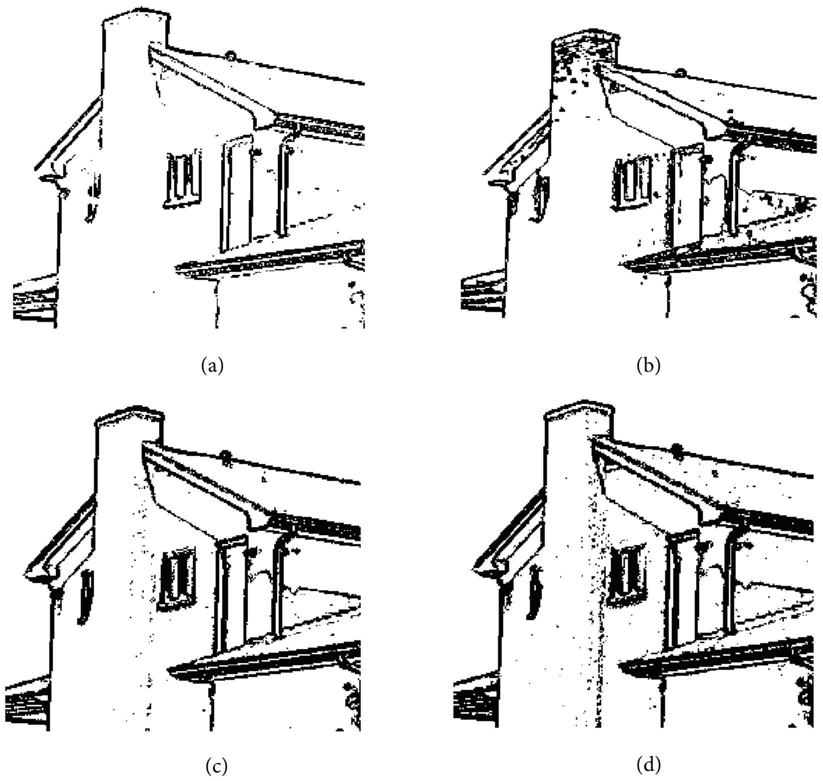

Figure 14. Edges of test image House in different color spaces: a) edges in the RGB color space, b) edges in the HSI color space, c) edges in the YCbCr color space, and d) edges in the CIE Lu'v' color space.

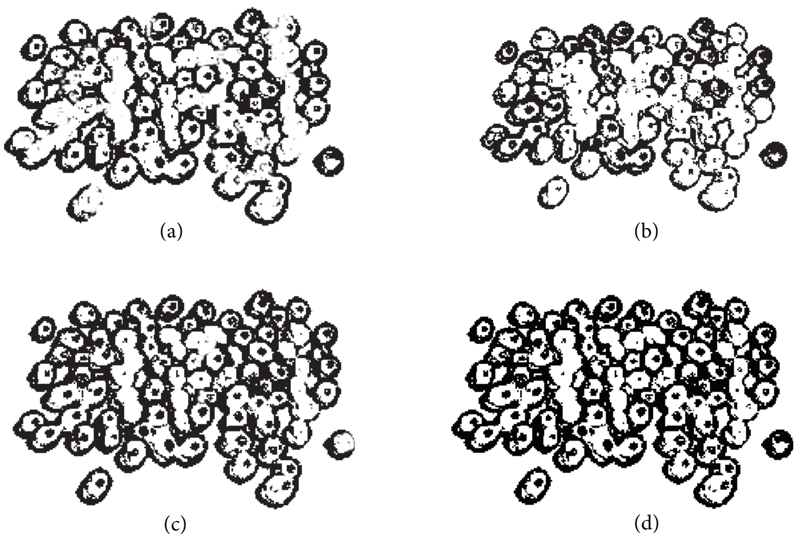

Figure 15. Edges of test image Jelly Beans in different color spaces: a) edges in the RGB color space, b) edges in the HSI color space, c) edges in the YCbCr color space, and d) edges in the CIE Lu'v' color space.

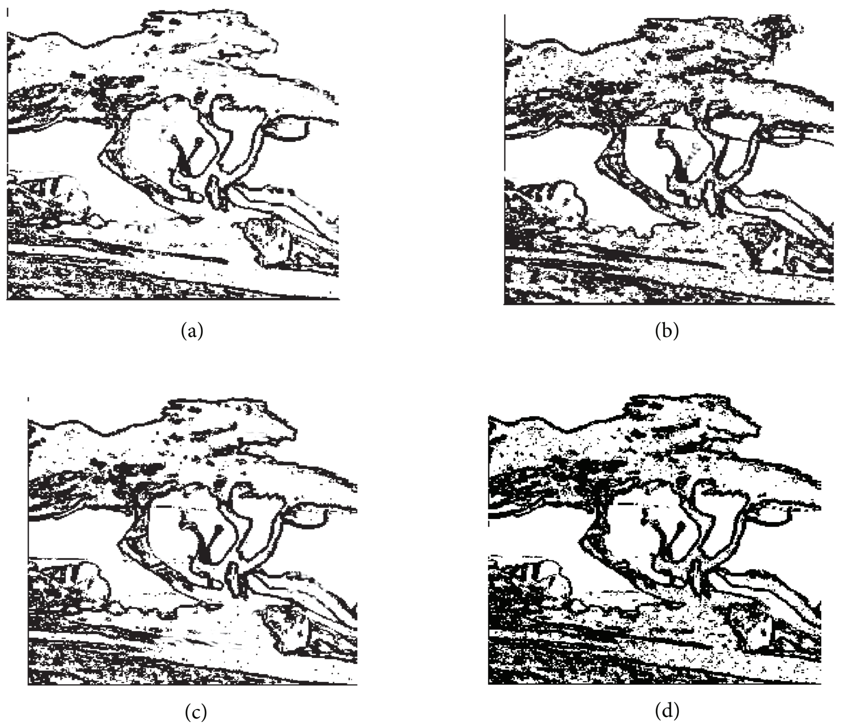

Figure 16. Edges of test image Tree in different color spaces: a) edges in the RGB color space, b) edges in the HSI color space, c) edges in the YCbCr color space, and d) edges in the CIE Lu'v' color space.

In the second part, the results of our proposed algorithm are compared with those of Chung et al.'s algorithm. The values of k in Eq. (5) and n in Eq. (8) are selected empirically so that the saturation means for each pixel of the image, which is done by 2 desaturated algorithms, become equal. We do this to fairly compare the 2 algorithms. The saturation mean of each image pixel is defined as m_s and after applying the edge-preserving algorithm it is named m_{eff}, which is computed by Eqs. (9) and (10):

$$m_s = \frac{1}{p}\sum_{j=1}^{p}\left|\overline{C_J C_{ds_J}}\right|, \tag{9}$$

$$m_{eff} = \frac{1}{p}\sum_{j=1}^{p}\left|\overline{C_J C_{es_J}}\right|, \tag{10}$$

where p is the total number of pixels, $\left|\overline{C_J C_{ds_J}}\right|$ is the distance between C and C_{ds}, and $\left|\overline{C_J C_{es_J}}\right|$is the distance between C and C_{es} for pixel j.

For assessing the results, we computed the difference (E_s) between the number of saturated image edges and the edges of the original image in the CIE Lu'v' color space, the difference (E_{ds}) between the number of desaturated image edges and edges of the original image in the CIE Lu'v' color space, and the difference (E_{eff}) between the number of image edges preserved in the algorithm and the edges of the original image. E_s, E_{ds}, and E_{eff} represent binary error images. Figure 17a shows test image Lena and Figure 17b shows this image in the CIE Lu'v' color space. The results of applying the Trahanias edge detector on Figure 17b are shown in Figure 17c. Next, the image is saturated and the Trahanias edge detector is applied on it. Figure 17d shows E_s

when the missing edges and spurious edges (or errors) are produced by the saturating process. The saturated image is then desaturated using the Eq. (6) with $k = 0.3$ and $m_s = 0.0576$. The computed E_{ds} is illustrated as an error image in Figure 17e. By applying the previous edge-preserving algorithm, the value of m_{eff} becomes 0.0541. The error image E_{eff} is also obtained and is shown in Figure 17f.

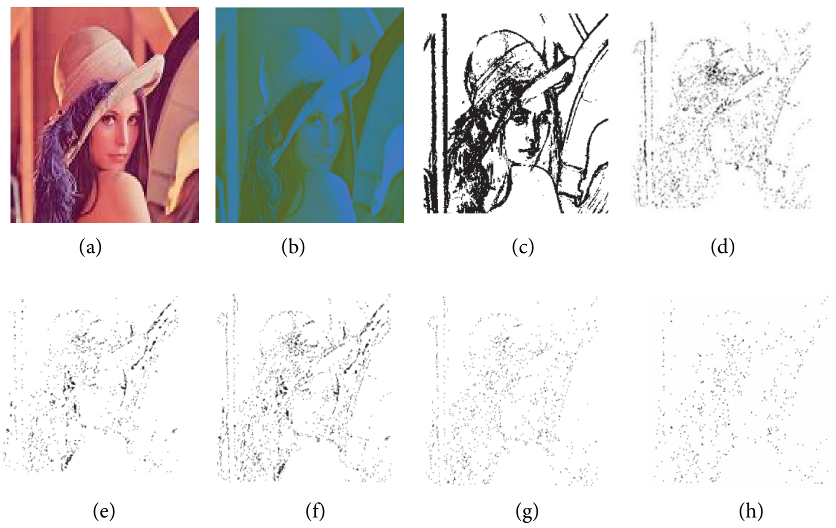

Figure 17. Obtained results for test image Lena; a) original image, b) original image in the CIE Lu'v' color space, c) edges of original image, d) image of E_s, e) E_{ds} by Chung et al.'s algorithm in the desaturation process, f) E_{eff} by Chung et al.'s algorithm in the edge-preserving process, g) E_{ds} by the proposed desaturation method, and h) E_{eff} by our proposed preservation algorithm.

By selecting $n = 0.4$ and using Eqs. (6) through (8), the saturated image is desaturated and $m_s = 0.0576$ is obtained. For this case, E_{ds} is shown in Figure 17g. We then apply the proposed edge-preserving algorithm on test image Lena with $m_{eff} = 0.0568$. The error image E_{eff} for the proposed algorithm is illustrated in Figure 17h.

The comparison of Figures 17f and 17h shows that the proposed algorithm preserves the edges much better than the previous algorithm; however, in both methods, the amount of saturation for each pixel is the same ($m_s = 0.0576$) before the edge-preserving process.

We also compare the algorithms using 3 other test images with few details, i.e. Jelly Beans, Peppers, and House (see Figure 18). We first saturate these images and then determine the differences between the edges of the original images and those of the saturated images, without using any edge-preserving algorithms (E_s). The obtained results are shown in Figures 18a–18c.

Next, we desaturate the mentioned test images using Chung et al.'s algorithm and our proposed method, and then apply both algorithms to preserve the edges. The error images (E_{eff}) using Chung et al.'s algorithm are illustrated in Figures 18d–18f. Figures 18g–18i also show the error images (E_{eff}) when our proposed algorithm is applied. As can be clearly seen, the performance of our algorithm in preserving the true edges and in avoiding the superficial edges is superior to the other algorithms.

In addition, 2 other images with more details, i.e. Tree and Baboon (see Figure 19), are used to make more comparisons. Figures 19a and 19b show the image of E_s without applying any edge-preserving algorithm.

The error images (E_{eff}) are shown in Figures 19c and 19d by applying Chung et al.'s algorithm. These images are compared with Figures 19e and 19f when our edge-preserving algorithm is applied. For these detailed images, our algorithm performs much better in comparison with the other algorithms.

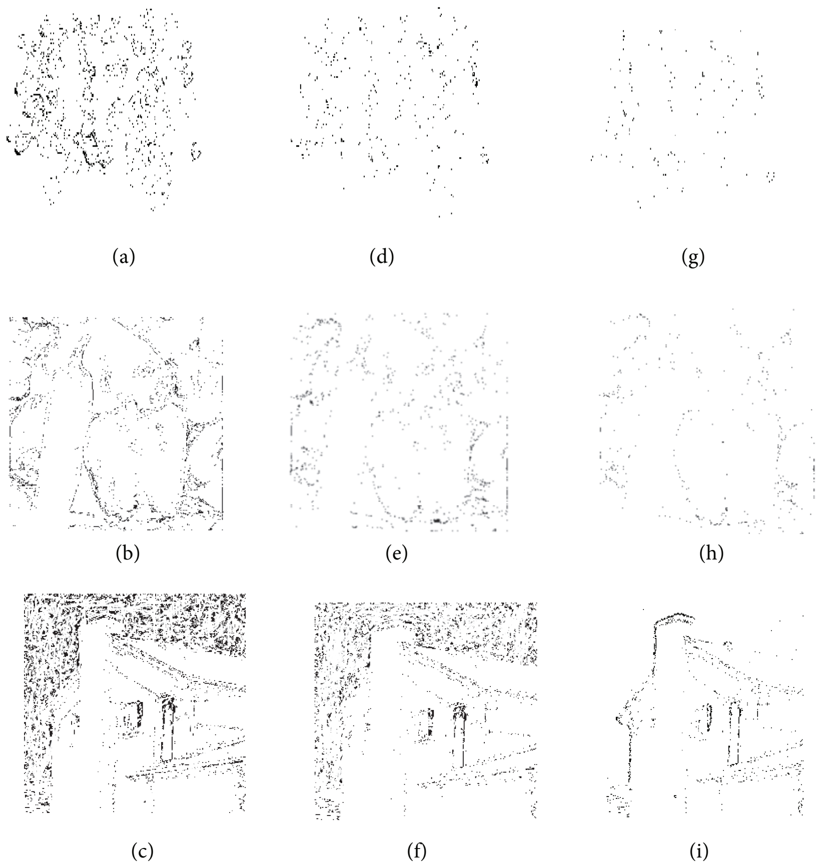

Figure 18. Obtained results for test images Jelly Beans, Peppers, and House: a), b), and c) the edge errors without using an edge-preserving algorithm (E_s) for Jelly Beans, Peppers, and House, respectively; d), e), and f) E_{eff} using Chung et al.'s algorithm for Jelly Beans, Peppers, and House with k = 0.7, k = 0.25, and k = 0.6, respectively; and g), h), and i) E_{eff} using our proposed algorithm for Jelly Beans, Peppers, and House with n = 0.2, n = 0.3, and n = 0.5, respectively.

To quantitatively evaluate the results, we use the rate of improvement in edge preservation, R, which can be computed by Eq. (11).

$$R = \frac{E_s - E_{eff}}{E_s} \times 100 \tag{11}$$

In the Table, the performance of the algorithms on the mention images is assessed statistically. As can be seen, the proposed algorithm produces a significantly better rate of improvement in the edge preserving of the 6 test images. Some conclusions can be extracted from the obtained results: 1) When we increase n from 0.2 to 0.8 in our algorithm, it leads to an augmentation in the amount of image saturation (m_s), while $k$in Chung et al.'s algorithm follows a diverse pattern. 2) We can see that for the saturated images, we can obtain the same edge improvement rate by choosing appropriate values for k and n. Hence, an image can be desaturated by 2 methods by the same rate. However, after applying the edge-preserving algorithms, m_{eff} in our algorithm is higher than m_{eff} in the previous algorithm. 3) The value of E_{ds} shows that the transferring origin coordinate system is better than applying the central gravity law of color mixture for desaturating an image. 4) Although our algorithm saturates more images, the value of E_{eff} shows that the edge preservation in our algorithm is still better than that in the previous algorithm.

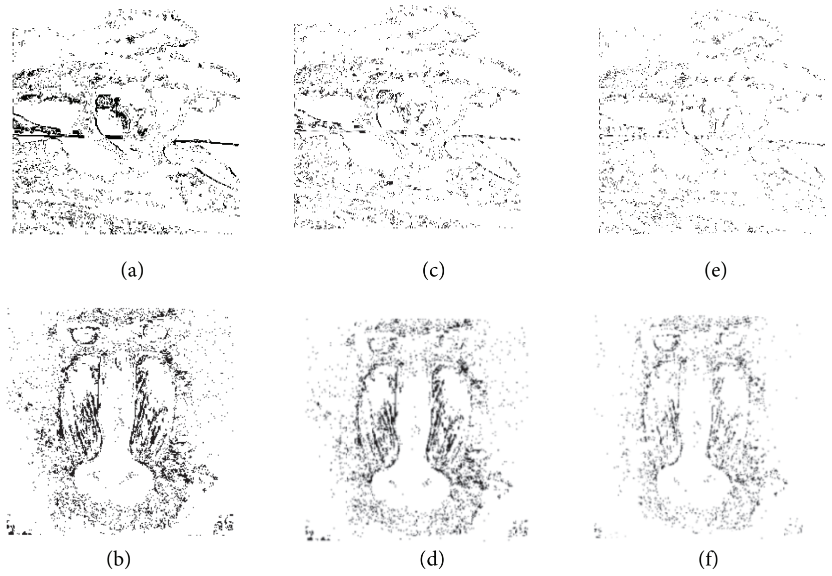

Figure 19. Obtained results for test images Tree and Baboon: a) and b) the edge errors without using an edge-preserving algorithm (E_s) for Tree and Baboon, respectively; c) and d) E_{eff} using Chung et al.'s algorithm for Tree and Baboon with $k = 0.25$ and $k = 0.1$, respectively; and e) and f) E_{eff} using our proposed algorithm for Tree and Baboon with $n = 0.65$ and $n = 0.8$, respectively.

Table. The results of applying the algorithms on the test images.

Image	Algorithm	m$_s$	E$_s$	E$_{ds}$	E$_{eff}$	m$_{eff}$	R (%)
Jelly Beans	Chung et al.'s algorithm, K = 0.7	0.0159	1225	647	370	0.0134	69.79
	Our proposed algorithm, n = 0.2	0.0159	1225	424	139	0.0160	88.65
Peppers	Chung et al.'s algorithm, k = 0.25	0.0182	2443	1799	1202	0.0181	50.79
	Our proposed algorithm, n = 0.3	0.0182	2443	1032	421	0.0221	82.76
Lena	Chung et al.'s algorithm, k = 0.3	0.0576	2960	2746	1865	0.0541	36.99
	Our proposed algorithm, n = 0.4	0.0576	2960	1193	591	0.0568	80.03
House	Chung et al.'s algorithm, k = 0.6	0.0638	4900	3283	2890	0.0611	41.02
	Our proposed algorithm, n = 0.5	0.0638	4900	931	900	0.0637	81.63
Tree	Chung et al.'s algorithm, k = 0.25	0.0799	5630	4991	4175	0.0759	25.84
	Our proposed algorithm, n = 0.65	0.0799	5630	3981	2297	0.0775	59.20
Baboon	Chung et al.'s algorithm, k = 0.1	0.0807	7131	6493	5917	0.0769	16.82
	Our proposed algorithm, n = 0.8	0.0807	7131	6257	4313	0.0775	39.51

5. Conclusion

In this paper, we introduced 4 color spaces and then showed that the best color space for the edge detection process is the CIE Lu'v' color space. In addition, this color space has a property that helps to better saturate the image for producing better segmentation and edge detection in practical cases, because pixels that are in the same region become more similar chromatically. On the other hand, image saturation causes the creation of spurious edges and the loss of true edges, so the edge-preserving algorithm is needed. To do that, first, the image is saturated in the CIE Lu'v' color space and then desaturated by transferring the origin of the coordinate system, so that fewer true edges are missing and fewer spurious edges are added in the results. Hence, we can saturate a color image more and better edge preserving can be achieved. The experimental results on well-known test images demonstrate the superiority of our proposed algorithm compared with the previous one.

Acknowledgments

The authors would like to thank the Department of Electrical and Computer Engineering at Islamic Azad University, Zarghan Branch, Iran, for the help provided for this work to be implemented.

References

[1] T.W. Ridler, S. Calvard, "Picture thresholding using an iterative selection method", IEEE Transactions on Systems, Man and Cybernetics, Vol. 8, pp. 630–632, 1978.

[2] A. Mehnert, P. Jackway, "An improved seeded region growing algorithm", Pattern Recognition Letters, Vol. 18, pp. 1065–1071, 1997.

[3] Z. Wu, R. Leahy, "An optimal graph theoretic approach to data clustering: theory and its application to image segmentation", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 15, pp. 1101–1113, 1993.

[4] P.E. Trahanias, A.N. Venetsanopoulos, "Color edge detection using order statistics", IEEE Transactions on Image Processing, Vol. 2, pp. 259–264, 1993.

[5] J. Schanda, Colorimetry Understanding the CIE System ,New York, Wiley Interscience, 2007.

[6] S.J. Sangwine, R.E.N. Horne, The Color Image Processing Handbook, London, UK, Chapman and Hall, 1998.

[7] L. Lucchese, S.K. Mitra, "Filtering color images in the xyY color space", Proceedings of the International Conference of Image Processing, pp. 500–503, 2000.

[8] K.L. Chung, Y.W. Liu, W.M. Yan, "Efficient edge preserving algorithm for color contrast enhancement with application to color image segmentation", Journal of Visual Communication and Image Representation, Vol. 17, pp. 299–310, 2008.

[9] J. Scharcanski, A.N. Venetsanopoulos, "Edge detection of color images using directional operators", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 7, pp.397–401, 1997.

[10] C. Theoharatos, G. Economou, S. Fotopoulos, "Color edge detection using the minimal spanning tree", Pattern Recognition, Vol. 38, pp. 603–60, 2005.

[11] V. Barnett, "The ordering of multivariate data", Journal of The Royal Statistical Society, Vol. 139, pp. 318–343, 1976.

[12] A. Sharma, Understanding Color Management , New York, Thomson Delmar Learning, 2003.

[13] R. Jackson, L. MacDonald, K. Freeman, Computer Generated Colour, New York, Wiley, 1994.

[14] L. Lucchese, S.K. Mitra, J. Mukherjee, "A new algorithm based on saturation and desaturation in the xy chromaticity diagram for enhancement and rendition of color images", Proceedings of the International Conference of Image Processing, Vol. 2, pp. 1077–1080, 2000.

[15] S.C. Pei, Y.C. Zeng, C.H. Chang, "Virtual restoration of ancient Chinese paintings using color contrast enhancement and lacuna texture synthesis", IEEE Transactions on Image Processing, Vol. 13, pp. 416–429, 2004.