Fast Color Edge Detection Algorithm Based on Similarity Relation Matrix

H.I.ALI

Electronics and Communication Engineering Department, Helwan University, Cairo, 11792, Egypt

Abstract

A fast edge detection algorithm for color images was described in this paper. In the proposed method, smoothness of each pixel in color image is firstly calculated by means of similarity relation matrix and is normalized to maximum gray level. In spite of the simplicity of the proposed method, it can be further simplified taking benefit from the symmetry of matrix components as well as the avoidance of re-computing the common matrix elements between two successive pixel windows. This will reduce the time complexity extremely. The time complexity is further reduced by the splitting the algorithm into parallel tasks.

Keywords: *Edge detection; Similarity relation matrix; Smoothness thresholding*

1. Introduction

Edge is commonly defined as a sudden change in the local color intensity of an image or a jump in intensity from one pixel to the next. On the other hand, a region in image that generally represents objects could be defined as a collection of pixels, which share similar intensities. Thus boundaries of regions or objects in image are characterized by edges. The analysis of an image can be simplified by detecting edges since; it reduces the amount of data to be processed. Usually, edge detection is performed by smoothing, differentiating and thresholding.

The gradient of an image is the most common edge detector so far. Computing the gradient of an image can be performed by obtaining the partial derivatives in x and y directions by means of many operators like Roberts, Prewitt and Sobel operators [1, 2]. The gradient-based edge detection methods suffer from some practical limitations. Firstly, they need a smoothing operation to alleviate the effect of high spatial frequency in estimating the gradient. Smoothing is applied to all pixels in the image including the edge regions. This may distort or eliminate the edge in some cases. Secondly, the gradient magnitude alone is insufficient to determine meaningful edges because of the ambiguity caused by underlying pixel pattern, especially in complex natural scenes. Thirdly, the gradient-based edge detection method increases the computational complexity because calculations, such as square root and arctangent, to produce the gradient vector are required [1].

A detailed comparison and evaluation of edge detectors has been performed by Heath et al. [3]. They employed people to evaluate performance of several edge detectors with a number of images and looked for correlations in judgments of participants. Kim and Han have described edginess of pixels in terms of fuzzy rules [4] whereas the gradient magnitude and direction with fuzzy reasoning rules have been used to locate edges by others [5–7].

On the other hand, the edge detection process of color images is another important research issue. Typically, a color image consists of RGB channels. The color edge detection process must take into account the changes in intensity, chromaticity or both. So far, several color edge detection algorithms have been developed. These schemes can be classified into two different approaches. In the first approach, the three-channel image is processed as three gray-level images. We can use any gray-level edge detection scheme to detect the edge image for each color channel separately. Therefore, three edge images can be obtained for the RGB channels. Finally, a merging procedure is executed to combine these edge images into a targeted edge image. However, these sorts of algorithms have two major drawbacks. First, the inter-channel correlation is discarded in these schemes. Second, a high computational cost is consumed [8–10].

In the second approach, a two-stage structure is imposed on the design of color edge detection schemes. In the first stage, a channel reduction technique is employed to reduce the dimensionality of each color image from three to one. Next, an edge detection procedure for the reduced one-channel image is executed to detect edge. The two-stage edge detection schemes or color image have two advantages. First, the channel reduction process is independent of the edge detection scheme. The correlation among the color channels is taken into

consideration. Second, any gray-level edge detection scheme can be applied to the detection of edge images in the second stage. In other words, the computational cost can be reduced [11–14].

Recep Demirci [15], proposed a two-stage edge detection algorithm. Firstly, the color image in three dimensional color spaces is mapped into one dimension by means of the similarity relation matrix. This transformation produces a gray level image where the similar pixels show the smooth areas and dark pixels show the dissimilar areas, noise and edges. Secondly, the thresholding could be employed if it is preferred. This work suffers from the high computational cost. So, in this paper we propose a fast edge detection algorithm that takes benefit from the symmetry of similarity matrix components as well as the avoidance of re-computing the common matrix elements between two successive pixel windows. Also a parallel implementation of the proposed algorithm is implemented which further reduces the time needed by the algorithm.

2. Color similarity

An image consists of pixels, which are neighbor to each other. Gray level differences of each color component between pixel P_1 and P_2 could be defined as follows:

$$\Delta R = |L_{R,1} - L_{R,2}| \quad (1)$$
$$\Delta G = |L_{G,1} - L_{G,2}| \quad (2)$$
$$\Delta B = |L_{B,1} - L_{B,2}| \quad (3)$$

The Euclidean distance can be computed as:

$$d_{i,j} = \frac{1}{\sqrt{3}}(\Delta R^2 + \Delta G^2 + \Delta B^2)^{1/2} \quad (4)$$

On the other hand, Wuerger et al. [16] showed in their research into proximity judgments in color space that perceptual color proximity is not Euclidean in nature. That means that distance information in Euclidean color space is not adequate for similarity judgment. The most general from of similarity measure based on the distance in color space could be given as below:

$$S_1(x_i, x_j) = 1 - \frac{d_{ij}}{D_n} = 1 - \frac{\|x_i - x_j\|}{D_n} \quad (5)$$

Where, S_1 is similarity between x_i and x_j, D_n is normalization coefficient. In Generalized Context Model, the similarity was expressed in terms of an exponential or Gaussian function of distance [17, 18]. So, the following formula could be obtained for color similarity:

$$S_2(x_i, x_j) = \exp\left(\frac{-d_{ij}^q}{D_n}\right) = \exp\left(\frac{-\|x_i - x_j\|^q}{D_n}\right) \quad (6)$$

With q=1 and 2 we obtain an exponential and a Gaussian functions respectively. The employment of Gaussian function with color distances to calculate similarity measure in term of color histograms was recently performed in applications for color image retrievals [28, 29]. Recep Demirci, showed that the usage of an exponential or Gaussian functions as color similarity functions, gives better performance in measuring similarity [15].

3. Proposed algorithm

3.1. Formation of similarity relation matrix

The similarity of any neighboring two pixels could be calculated by means of Eq.'s (5) and (6), respectively. A pixel in an image has eight neighboring pixels as shown in Fig. 1. Therefore, the similarity calculations for all the possible combinations are performed as shown in Fig. 2.

P1	P4	P7
P2	P5	P8
P3	P6	P9

Fig. 1 Pixel neighborhood

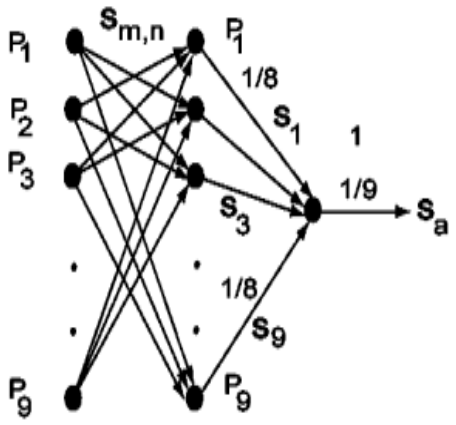

Fig. 2 Similarity network

This approach is well-suited with noisy exemplar approach proposed by Kahana and Sekular [17] where inter-stimulus similarity is used to categorize the noisy image. Consequently, similarity relation matrix is achieved as:

$$S_{m,n} = \begin{bmatrix} S_{1,1} & S_{1,2} & \dots & S_{1,9} \\ S_{2,1} & S_{2,2} & \dots & S_{2,9} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ S_{9,1} & S_{9,2} & \dots & S_{2,9} \end{bmatrix} \quad (7)$$

As the similarity of a pixel to itself is always equal to unity, there is no need to calculate them. The similarity relation matrix is symmetric. When all elements of the $S_{m,n}$, apart from diagonal, are unity, it means that the central pixel and its all neighbors have the same color level or it is perfectly smooth. The local smoothness of k^{th} pixel could be estimated as follows:

$$S_k = \frac{1}{8}\sum_{n=1}^{9} S_{k,n} \quad for \ k \neq n \quad (8)$$

As could be seen from Eq. (8) and Fig. 2, there is no need to consider the similarity of itself. Therefore, Eq. (8) gives the average of how the k^{th} pixel is similar to the others. On the other hand, the general average of the central and all neighboring pixels could be calculated as follows:

$$S_a = \frac{1}{9}\sum_{k=1}^{9} S_k \quad (9)$$

The Sa could be interpreted as the smoothness of the central pixel. Its values vary between zero and 1. On the other hand, the complement of the Sa (1−Sa) is considered to be dissimilarity or noisiness [15].

Calculating the similarity using this direct approach, as proposed in [15], needs very large time. So we proposed a modified approach for calculating the color similarity, which will reduce the computation time considerably. To show how to reduce the calculations needed to compute Sa, consider Fig. 3 that shows the elements of two successive pixel windows.

P1	P4	P7	P10
P2	P5	P8	P11
P3	P6	P9	P12

Fig.3 The elements of two successive pixel windows

If we need to compute the similarity of pixel P5, we will need to construct the similarity matrix shown in Eq. (7). Now to compute the similarity of next pixel P8, the similarities between (P4, P5, P6, P7, P8, and P9) need not be computed again. These similarities are already computed during the computation of the similarity matrix of pixel P5. So, we need to reshape this matrix to construct the similarity matrix for the next pixel (P8).

Studying the matrix in Eq. (7), we find that similarities between (P1, P2, P3) and all other pixels will not be needed again during the construction of the similarity matrix for the next pixel. So, the first step is to eliminate rows 1, 2 and 3, also the columns 1, 2 and 3. Secondly, the similarities between pixels (P10, P11, and P12) and (P4, P5, P6, P7, P8, P9) need to be computed.

Generally, the similarity matrix for pixel $P_{m+1,n}$ can be constructed as shown in Eq.(10). The lined elements will not be computed, so they will be cashed from the similarity matrix computed previously for pixel $P_{m,n}$. finally, we can summarize the steps to construct a similarity matrix for $P_{m+1,n}$ (call it $S_{m+1,n}$) using the similarity matrix for $P_{m,n}$ (call it $S_{m,n}$) as:

1- Eliminate rows 1,2,3
2- Eliminate columns 1,2,3
3- Compute the similarities between (P10,P11,P12) and (P4,P5,P6,P7,P8,P9)

$$S_{m+1,n} = \begin{bmatrix} \cancel{S_{4,4}} & \cancel{S_{4,5}} & \dots & \cancel{S_{4,9}} & S_{4,10} & S_{4,11} & S_{4,12} \\ \cancel{S_{5,4}} & \cancel{S_{5,5}} & \dots & \cancel{S_{5,9}} & S_{5,10} & S_{5,11} & S_{5,12} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \cancel{S_{9,1}} & \cancel{S_{9,5}} & \dots & \cancel{S_{9,9}} & S_{9,10} & S_{9,11} & S_{9,12} \\ S_{10,1} & S_{10,5} & \dots & S_{10,9} & S_{10,10} & S_{10,11} & S_{10,12} \\ S_{11,1} & S_{11,5} & \dots & S_{11,9} & S_{11,10} & S_{11,11} & S_{11,12} \\ S_{12,1} & S_{12,5} & \dots & S_{12,9} & S_{12,10} & S_{12,11} & S_{12,12} \end{bmatrix} \quad (10)$$

3.2 Local similarity thresholding (LST)

Kahana and Sekular [17] have described inter-stimulus dissimilarity as noise in their investigation by assuming that each stimulus is stored imperfectly in memory. Therefore they proposed to threshold the summed similarity of a stimulus to make decision. Keeping in mind that the complement of the Sa is considered as dissimilarity or noisiness and assuming that each pixel is stimulus, Recep Demirci [15], proposed a new version of similarity matrix as follows:

$$\tilde{S}_{m,n} = \begin{cases} 1 & if \ S_{m,n} \geq S_T \\ 0 & if \ S_{m,n} < S_T \end{cases} \quad (11)$$

Where, S_T is similarity threshold. The interpretation of Eq. (11) is that if the similarity of two pixels is lower than S_T, they are considered to be dissimilar. So the local smoothness of k^{th} pixel could be estimated as follows:

$$\tilde{S}_k = \frac{1}{8}\sum_{n=1}^{9} S_{k,n} \quad for \ k \neq n \quad (12)$$

The general average of the central and all neighboring pixels could be calculated as follows:

$$\tilde{S}_a = \frac{1}{9}\sum_{k=1}^{9} \tilde{S}_k \quad (13)$$

With such approach, a color image in three-dimensional color spaces is mapped into one-dimensional gray image while the noise is weakened and the edge information is preserved.

These calculations, as proposed in [15], could be reduced also, for the Euclidean, Exponential and Gaussian functions, with which we compute $S_{m,n}$. Our approach is based on rearranging the inequality (11) by putting the $\|x_i - x_j\|$ in one side, while all other variables are in the other side. In this way, we will obtain the following variations shown in table 1.

Table 1: Rearranging inequality (11) for Euclidean, Exponential and Gaussian forms

Original inequality [15]	$\tilde{S}_{m,n} = \begin{cases} 1 & if\ S_{m,n} \geq S_T \\ 0 & if\ S_{m,n} < S_T \end{cases}$
Rearranging the Euclidean form of $S_{m,n}$	$\tilde{S}_{m,n} = \begin{cases} 1 & if\ \|x_i - x_j\| \leq Dn(1 - S_T) \\ 0 & if\ \|x_i - x_j\| > Dn(1 - S_T) \end{cases}$
Rearranging the exponential form of $S_{m,n}$	$\tilde{S}_{m,n} = \begin{cases} 1 & if\ \|x_i - x_j\| \leq -Dn * \ln(S_T) \\ 0 & if\ \|x_i - x_j\| > -Dn * \ln(S_T) \end{cases}$
Rearranging the Gaussian form of $S_{m,n}$	$\tilde{S}_{m,n} = \begin{cases} 1 & if\ \|x_i - x_j\| \leq \sqrt{-Dn * \ln(S_T)} \\ 0 & if\ \|x_i - x_j\| > \sqrt{-Dn * \ln(S_T)} \end{cases}$

Studying table1 carefully, we notice that the right hand sides of the inequalities are all of constant values. So, we can compute them once at the beginning of the algorithm and compare directly with the pixels difference. This will surely reduce the computation of $S_{m,n}$, since we need not to compute $S_{m,n}$, in its different forms, and then compare with ST. All what is needed is to compute the difference between pixels and directly compare this difference with the new constants, computed once, to get $\tilde{S}_{m,n}$.

4. Simulation results and discussion

The original algorithm [15] and the modified algorithm were tested with the well-known peppers image shown in Fig. 4(a) which is a color image of size 256 × 256 pixels. Fig. 4(b)–(d) show the smoothness image obtained by means of Eq. (5) with different normalization coefficients: 32, 64 and 128, respectively. Fig. 4(e)–(g) shows the transferred images when the similarity of pixels is calculated by means of exponential similarity function, Eq. (6), with different D_n: 32, 64 and 128, respectively. Moreover, Fig. 4(h)–(j) have been achieved when the Gaussian similarity measure is employed with different D_n: 1024, 8192 and 16 384, respectively.

Table 2 shows the time needed by the modified algorithm and the original algorithm [15] using same image, Peppers image. The algorithms were implemented on a computer having the following specifications: Intel(R) Core™2 Duo CPU @ 2,26GHZ. The results show that there is a large reduction in time complexity more than 40%.

Table 2: The time needed by the modified algorithm and the original algorithm

Distance function	Time for the original algorithm (seconds)	Time for the modified algorithm (seconds)	Reduction percent
Euclidean	155.54	92.32	40.65%
Exponential	158.25	93.31	41.04%
Gaussian	156.19	93.45	40.17%

Table 3 shows the time needed by the modified algorithm when implemented as parallel tasks on two processors and the original algorithm using same image, Peppers image. The results show that there is a large reduction in time complexity more than 70%. The image data was split into two equal parts. Each part is processed by a separate task on a separate processor.

Table 3: the time needed by the original algorithm and the modified algorithm when implemented as parallel tasks

Distance function	Time for the original algorithm (seconds)	Time for parallel implementation of the modified algorithm (seconds)	Reduction percent
Euclidean	155.54	42.64	72.59%
Exponential	158.25	43.95	72.23%
Gaussian	156.19	44.64	71.42%

Fig. 5 shows the transferred image of Peppers with original LST algorithm and the modified algorithm while normalization coefficient is fixed as 32. Fig. 5(a)–(c) show the images with ST: 0.25, 0.50 and 0.75, respectively, when linear similarity function is employed. Transformation of three dimension color space into one dimension with exponential function and different ST: 0.25, 0.50 and 0.75 have been shown in Fig. 5(d)–(f), respectively. Moreover, LST images obtained with Gaussian functions with ST: 0.25, 0.50 and 0.75 are shown in Fig. 5(g)–(i), respectively. It has been noticed from applications that the three dimensional color images could be successfully transferred into one-dimension image with these algorithms. The transformed image is a

gray scale image in which the gray levels show the edginess strength.

Table 4 shows the time needed by the modified LST algorithm and the original algorithm using same image, Peppers image. The results show that there is a large reduction in time complexity around 40%.

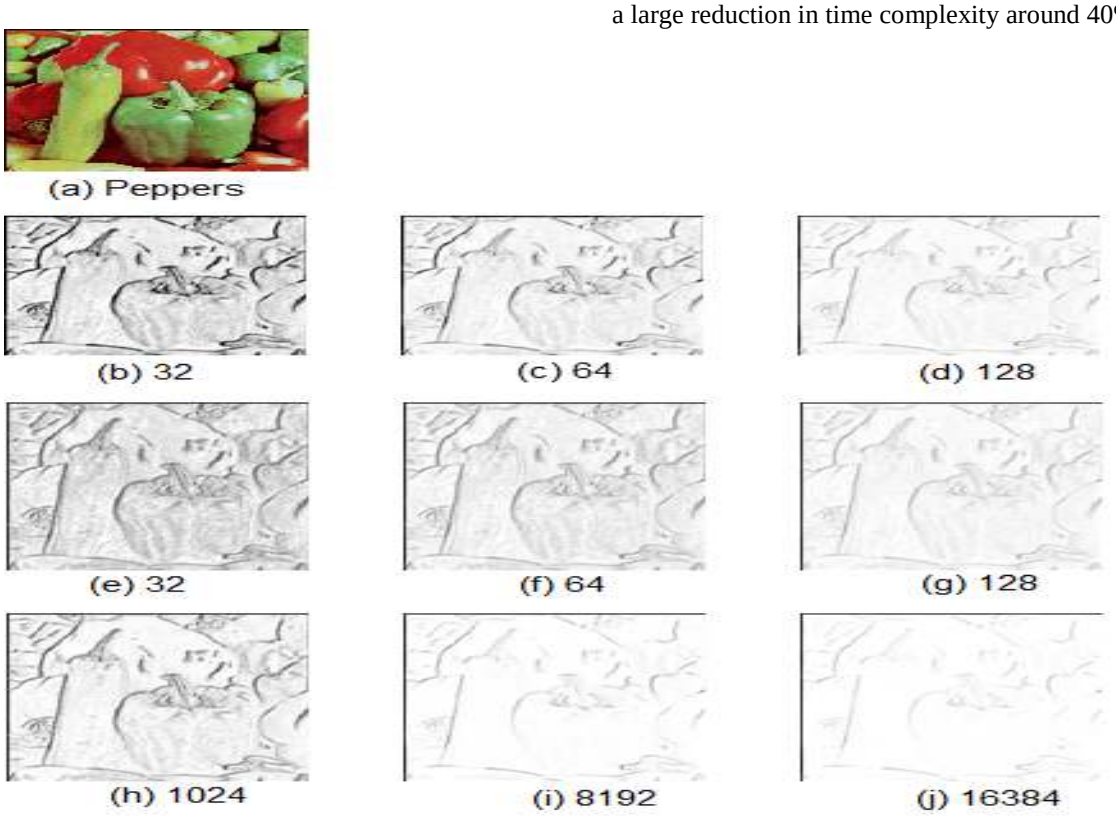

Fig. 4 Transformation of three-dimension color space into one dimension: Sa: (a) Peppers (b)–(d) By means of linear function with Dn: 32, 64 and 128. (e)–(g) By means of exponential function with Dn: 32, 64 and 128. (h)–(j) By means of Gaussian function with Dn: 1024,8192 and 16 384.

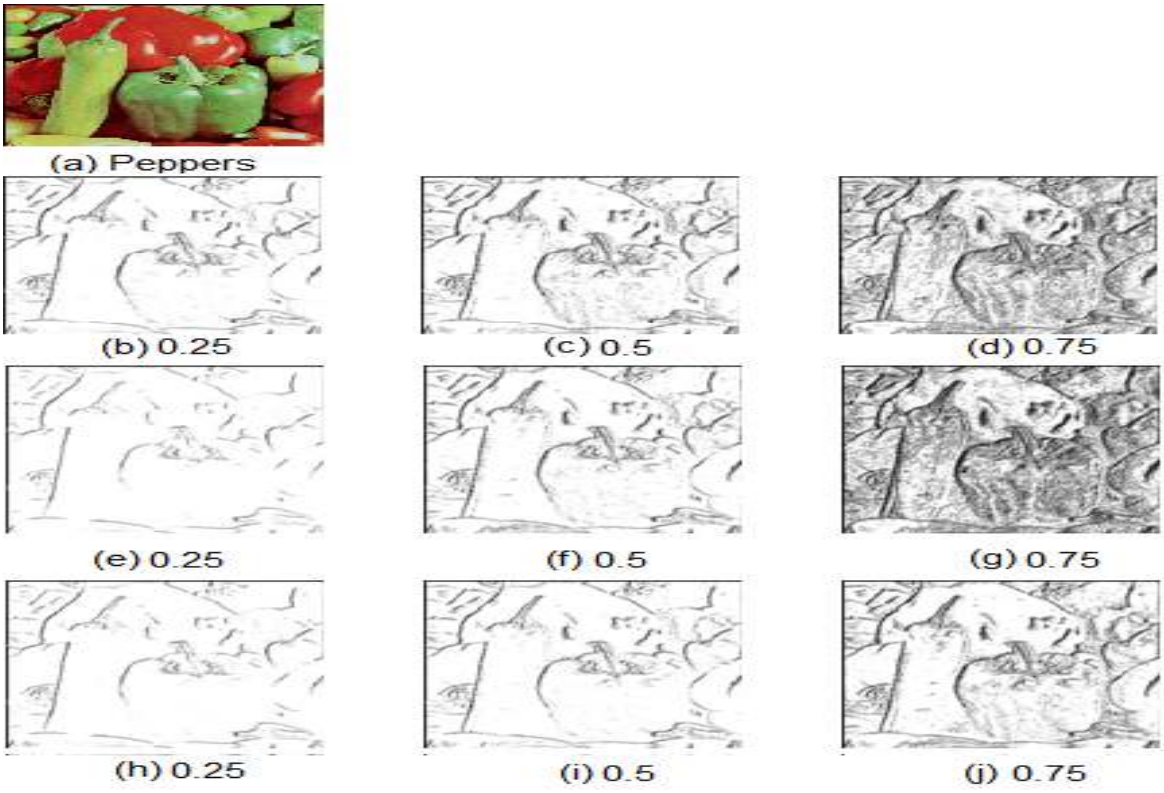

Fig. 5 Transformation of Lena image into one dimension with LST: Sa: (a) Peppers (b)–(d) By means of linear function with ST: 0.25, 0.50 and 0.75. (e)–(g) By means of exponential function with ST: 0.25, 0.50 and 0.75. (g)–(i) By means of Gaussian function with ST: 0.25, 0.50 and 0.75.

Table 4: The time needed by the modified LST algorithm and the original LST algorithm

Distance function	Time for the original LST algorithm (seconds)	Time for the modified LST algorithm (seconds)	Reduction percent
Euclidean	157.21	92.16	41.38%
Exponential	156.05	93.99	39.77%
Gaussian	157.30	93.84	40.34%

Table 5 shows the time needed by the modified LST algorithm when implemented as parallel tasks on two processors and the original algorithm using same image, Peppers image. The results show that there is a large reduction in time complexity more than 70%. The image data was split into two equal parts. Each part is processed by a separate task on a separate processor.

Table 5: the time needed by the original LST algorithm and the modified LST algorithm when implemented as parallel tasks

Distance function	Time for the original LST algorithm (seconds)	Time for parallel implementation of the modified LST algorithm (seconds)	Reduction percent
Euclidean	157.21	44.65	71.60%
Exponential	156.05	44.65	71.39%
Gaussian	157.30	43.26	72.50%

5. Conclusion

In this paper, a fast two-stage color edge detection algorithm, where the similarity relation matrix is used for the channel reduction process has been proposed. In spite of the simplicity of the original algorithm ,proposed in [15], it can be further simplified taking benefit from the symmetry of matrix components as well as the avoidance of re-computing the common matrix elements between two successive pixel windows. Accordingly, the three-dimensional color images could be successfully transferred into one-dimension images, which show color discontinuous as gray levels. The results show that there is a large reduction in time complexity more than 40%. The modified two-stage color edge detection algorithm was also implemented as parallel tasks on two processors. The results show that there is a large reduction in time complexity more than 70%.

References

[1] Gonzalez RC,Woods RE. Digital Image Processing. Reading, MA: Addison-Wesley; 1993.
[2] Canny JF. A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 1986;8(6):679–98.
[3] Heath H, Sarkar S, Sanocki T, Bowyer KW. Edge detector comparison: initial study and methodology. Comput Vision Image Understanding 1998;69:38–54.
[4] Kim TY, Han JH. Edge representation with fuzzy sets in blurred images. Fuzzy Sets Systems 1998;100:77–87.
[5] Kim DS, Lee WH, S Kweon I. Automatic edge detection using 3×3 ideal binary pixel patterns and fuzzy-based edge thresholding. Pattern Recognition Lett 2004;25:101–6.
[6] Yuksel ME,Yildirim MT. A simple neuro-fuzzy edge detector for digital images corrupted by impulse noise. AEU—Int J Electron Commun 2004;58:72–5.
[7] Liang LR, Looney CG. Competitive fuzzy edge detection. Appl Soft Comput 2003;3:123–37.
[8] Fan J, Aref WG, Hacid MS, Elmagarmid AK. An improved automatic isotropic color edge detection technique. Pattern Recognition Lett 2001;22:1419–29.
[9] Ruzon MA, Tomasi C. Edge, junction, and corner detection using color distributions. IEEE Trans Pattern Anal Mach Intell 2001;23:1281–95.
[10] Theoharatos C, Economou G, Fotopoulos S. Color edge detection using the minimal spanning tree. Pattern Recognition 2005;38:603–6.
[11] Trahanias PE, Venetsanopoulos AN. Color edge detection using vector order statistics. IEEE Trans Image Process 1993;2:259–64.
[12] Yang CK, Tsai WH. Reduction of color space dimensionality by moment-preserving thresholding and its application for edge detection in color images. Pattern Recognition Lett 1996;17:481–90.
[13] Yang CK, Tsai WH. Color image compression using quantization, thresholding, and edge detection techniques all based on the moment-preserving principle. Pattern Recognition Lett 1998;19:205–15.
[14] Tsai P, Chang CC, Hu YC. An adaptive two-stage edge detection scheme for digital color images. Real-time Imag 2002;8(4):329–43.
[15] Recep Demirci, Similarity relation matrix-based color edge detection, Int. J. Electron. Commun. (AE-) 61 (2007) 469 – 477
[16] Wuerger SM, Maloney LT, Krauskopf J. Proximity judgments in color space: tests of a euclidean color geometry. Vision Res 1995;35:827–35.
[17] Kahana MJ, Sekuler R. Recognizing spatial patterns: a noisy exemplar approach. Vision Res 2002;42:2177–92.
[18] Stewart N, Brown GDA. Similarity and dissimilarity as evidence in perceptual categorization. J Math Psychol 2005;49:403–9.

Hossam Eldin Ibrahim received his B.Sc. degree in Communications & Electronics Engineering, his M.Sc. degree in Computer Engineering, and his Ph.D. degree in Computer Engineering from Helwan University, Cairo, Egypt, in 2000, 2004, and 2009 respectively. He has three published papers at M.Sc. degree, and seven published papers at Ph.D. degree. He is currently a Teacher at Electronics, Communication & Computer Department, Faculty of Engineering, Helwan University.