2009 International Conference on Advances in Recent Technologies in Communication and Computing

A Color Edge Detection Algorithm in RGB Color Space

Soumya Dutta
Department of Electronics and Communication
Netaji Subhash Engineering College
Kolkata - 700152, India
soumya.nsec@gmail.com

Bidyut B. Chaudhuri
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute
Kolkata - 700108, India
bbc@isical.ac.in

Abstract— **Edge detection is one of the most commonly used operations in image processing and pattern recognition, the reason for this is that edges form the outline of an object. An edge is the boundary between an object and the background, and indicates the boundary between overlapping objects. This means that if the edges in an image can be identified accurately, all of the objects can be located and basic properties such as area, perimeter, and shape can be measured. Since computer vision involves the identification and classification of objects in an image, edge detection is an essential tool. Efficient and accurate edge detection will lead to increase the performance of subsequent image processing techniques, including image segmentation, object-based image coding, and image retrieval. A color image edge detection algorithm is proposed in this paper. Average maximum color difference value is used to predict the optimum threshold value for a color image and thinning technique is applied to extract proper edges. The proposed method is applied over large database of color images both synthetic and real life images and performance of the algorithm is evident from the results and is comparable with other edge detection algorithms.**

Keywords-color edge detection; color difference calculation in RGB vector field; edge thinning; smoothing by adaptive median filter

I. INTRODUCTION

Edge detector is one of the most important tools in computer vision. The edge detection process serves to simplify the analysis of images by drastically reducing the amount of data to be processed, while at the same time preserving useful structural information about object boundaries [1]. Edge detection in color image is far more challenging task than gray scale images as color space is considered as a vector space. Almost 90% of edge information in a color image can be found in the corresponding grayscale image. However, the remaining 10% can still be vital in certain computer vision tasks. [2]. Furthermore human perception of color image is more enriched than an achromatic picture [3]. Several color models are present such as RGB color model, YUV model, CMY color model, CMYK color model, HIS color model [4].

One of the earliest color edge detectors is proposed by Navatia. The image data is transformed to luminance and two chrominance components and Huckel's edge detector is used to find the edge map in each individual component independently, except for the constraint of having the same orientation [3]. Shiozaki proposed a method on the basis of entropy calculation [5]. In 1999 M. A. Ruzon and Carlo Tomasi proposed a method of detecting color edges with compass operator [6] and in 2001 with color distribution [9]. They considered a disc at each pixel location. The disc was divided into several pairs of opposite semi discs by rotating the diameter over 180^o with an interval of 15^o. The color distribution of the pixels in each such semi disc was found after doing vector quantization. The distance between two semi discs generated by a single diameter was the distance between their color distributions. The distance between two distributions was found using the Earth Mover's Distance (**EMD**). The edge magnitude at the pixel was the maximum distance among all the distances found between each pair of opposite semi discs created rotating the diameter. Cumani proposed an extension of second directional derivative approach to color images [7]. In this paper a method is discussed to detect edges in color images with a threshold process that automatically selects a value for a given picture and after that a thinning technique is applied to generate edge map.

II. STEPS OF PROPOSED METHOD

The proposed method consists of four steps. At first, the image is smoothed by median filter to suppress unwanted noise in the image. Secondly, maximum directional differences of sum of gray values (Red+Green+Blue) are calculated for each pixel. In the third step, image is thresholded with a single threshold value and finally the detected edges are thinned to get the proper edge map.

A. Smoothing by Adaptive Mudian filter

A traditional **median filter** is based upon moving a window over an image and computing the output pixel as the median of the gray values within the input window. If the window is J x K in size we can order the J*K pixels in gray level values from smallest to largest. If J*K is odd then the median will be the (J*K+1)/2 entry in the list of ordered gray values. Note that the value selected will be exactly equal to one of the existing gray values so that no round off error will be involved if we want to work exclusively with integer gray values. Median filters are quite popular because, for certain random types of noise, they provide excellent noise reduction capabilities, with considerably less blurring than linear smoothing filters of similar size which helps to preserve edges. In the proposed method we have used an **adaptive median filter**. One main reason for using adaptive median filter is that it seeks to preserve detail of the image while smoothing the non-impulse noise, something that the "traditional" median filter does not do [4].

The working of an adaptive median filter is divided into two levels and as follows:

Level A. $A1 = Z_{med} - Z_{min}$
$A2 = Z_{med} - Z_{max}$
If $A1 > 0$ AND $A2 < 0$, Go to Level B
Else increase the window size
If $S_{xy} < S_{max}$ repeat Level A
Else output Z_{xy}
Level B. $B1 = Z_{xy} - Z_{min}$
$B2 = Z_{xy} - Z_{max}$
If $B1 > 0$ AND $B2 < 0$, output Z_{xy}
Else output Z_{med}

where Z_{xy} = gray value at coordinate (x,y), Z_{max} = maximum gray value, Z_{min} = minimum gray value, S_{xy} = window size, S_{max} = maximum allowed window size. Maximum allowed window size for our experiment was 9X9 starting from 3X3 window.

B. Directional Color Difference Calculation

In the proposed method, color image is analyzed from **RGB color space** point of view. Each pixel in the image consists of three color channels known as RGB components. The range of values of each of this components lies within 0 to 255. **Edges exist in a color image where abrupt changes of RGB values occur**. So to detect proper edges, first the abrupt color differences in an image must be pointed out. Generally an edge can exist in four directions and they are 0^o, 90^o, 45^o and 135^o. To reduce the **computational overhead** we have calculated a transformed value for each pixel which converts three component valued pixels into a single valued attribute. This transformation is simply a weighted addition of three components.

$$\text{Pixel}(i,j)=2*\text{red}(i,j)+3*\text{green}(i,j)+4*\text{blue}(i,j) \qquad (1)$$

Reason of multiplying the weights with the color channels is discussed below:

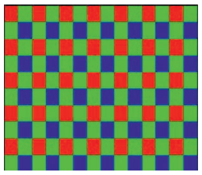


Figure 1. Checker box color image

For images like figure 1, checker box we can see that the image consists of three fundamental colors. So if we compute the sum of three channels for each pixel it will be 255. Because, for the red regions, the value of the red component is 255 and values of green and blue components are 0. Similarly, for blue regions only the value of the blue component will be 255 and red, green pixels will be 0. Same problem will arise for green regions. Using the **weighted sum technique** we can easily overcome this type of problem and can calculate exact color differences. So to calculate directional color differences we need four masks:

0	0	0	0	4	0	4	0	0	0	0	-4
4	0	-4	0	0	0	0	0	0	0	0	0
0	0	0	0	-4	0	0	0	-4	4	0	0

Figure 2. (a) A[3][3], (b)B[3][3], (c) C[3][3], (d) D[3][3] Four 3X3 directional masks are applied to the image smoothed with adaptive median filter

i,j	i,j+1	i,j+2
i+1,j	i+1,j+1	i+1,j+2
i+2,j	i+2,j+1	i+2,j+2

Figure 3. 3X3 Mask with coordinates

All four masks shown in figure 2 are moved over the transformed pixel values one by one to calculate the color differences in four directions. Mask A calculates horizontal color differences; mask B calculates vertical differences, mask C in 45^o direction and mask D in 135^o direction considering f(i+1,j+1) as the center pixel. The mathematical model for masking is presented below:

$$(\partial f(i+1,j+1)|\ 0^o) = \sum_{i=0}^{row}\sum_{j=0}^{col} \qquad (2)$$
$$[\ |\{2*R(i+1,j))+3*G(i+1,j)+4*B(i+1,j)\}*A(i+1,j) - \{2*R\ (i+1,j+2)+3*G(i+1,j+2)+4*B(i+1,j+2)\}*A\ (i+1,j+2)|\ ]$$

$$(\partial f(i+1,j+1)|\ 90^o) = \sum_{i=0}^{row}\sum_{j=0}^{col} \qquad (3)$$
$$[\ |\{2*R(i,j+1))+3*G(i,j+1)+4*B(i,j+1)\}*B(i,j+1) - \{2*R\ (i+2,j+1)+3*G(i+2,j+1)+4*B(i+2,j+1)\}*B\ (i+2,j+1)|\ ]$$

$$(\partial f(i+1,j+1)\ |\ 135^o) = \sum_{i=0}^{row}\sum_{j=0}^{col} \qquad (4)$$
$$[\ |\{2*R(i,j))+3*G(i,j)+4*B(i,j)\}*C(i,j) - \{2*R\ (i+2,j+2)+3*G(i+2,j+2)+4*B(i+2,j+2)\}*C\ (i+2,j+2)|\ ]$$

$$(\partial f(i+1,j+1)\ |\ 45^o) = \sum_{i=0}^{row}\sum_{j=0}^{col} \qquad (5)$$
$$[\ |\{2*R(i+2,j))+3*G(i+2,j)+4*B(i+2,j)\}*D(i+2,j) - \{2*R\ (i,j+2)+3*G(i,j+2)+4*B(i,j+2)\}*D\ (i,j+2)|\ ]$$

Considering each pixel at the centre of the mask the maximum directional color difference is calculated.

$$(\partial f\ (i+1,j+1)\ |\ \max) = \max((\ \partial f\ (i+1,j+1)\ |\ 0^o),(\ \partial f\ (i+1,j+1)\ |90^o),(\partial f(i+1,j+1)|135^o),(\ \partial f\ (i+1,j+1)\ |\ 45^o)) \qquad (6)$$

C. *Threshold Technique*

Threshold technique is very important task in edge detection algorithms. The accuracy of an algorithm is dependent on the choice of threshold parameters. One of the foremost criteria of thresholding is that the program should be efficient enough to automatically compute the optimum threshold parameter. The criteria of selection of a parameter for a given image are that the resultant edge map should satisfy the following [8]:

1. It should contain most of the prominent edges;
2. It should not contain too much spurious edges;
3. It should be meaningful and visibly pleasing;

The proposed scheme suggests a method of parameter selection that works satisfactorily on various types of images. We have already calculated the maximum color difference (∂f (i+1,j+1) | max) for each pixel. Next the average value of the maximum color difference is computed:

$$t = (\partial f|avg) = \sum_{i=0}^{row} \sum_{j=0}^{col} [\partial f(i,j) / (row*col)] \qquad (7)$$

The proposed method uses single threshold value T. There are several ways to obtain a fixed parameter value. A very simple way is to observe the edge maps for a set of selected images and take that value which is producing acceptable edge maps for all the selected images [8]. When such an experiment is done over more than 50 selected different kinds of images it produced acceptable edge maps at 1.2t. Thus the threshold value is set at T=1.2t.

C. *Edge thinning*

Edge map produced in this way contains thick edges. So a thinning technique is applied to create more thin edges which will be more accurate and visibly soothing. Two 3X3 masks are applied in thinning operation.

0	0	0	0	-1	0
-1	2	-1	0	2	0
0	0	0	0	-1	0

Figure 1. Thinning masks

Two masks as shown in figure 4 are used to create thin edges. These two masks are moved over the edge image produced after thresholding. Left one works in the horizontal direction and the right one works in the vertical direction. This procedure creates thin pixel edge response by suppressing a thick edge response from both sides (vertically or horizontally) and eventually keeps thin edge response. Effect of the thinning masks is clearly observable from the figures given below:

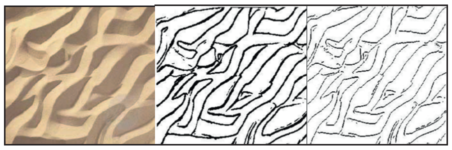


Figure 2. (a) Sand image, (b) multi pixel thick edge map after threshold, (c) final edge map after using thinning masks

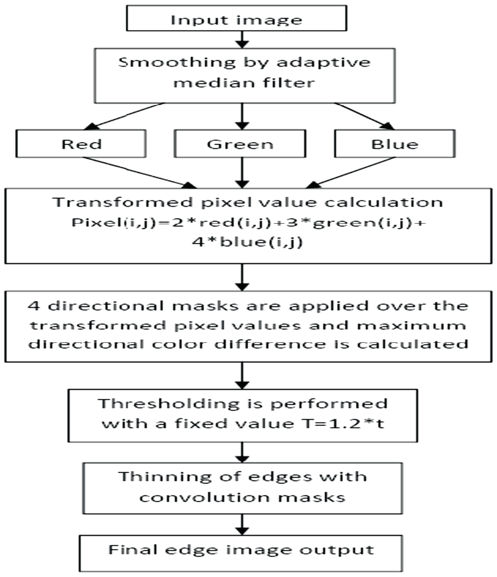
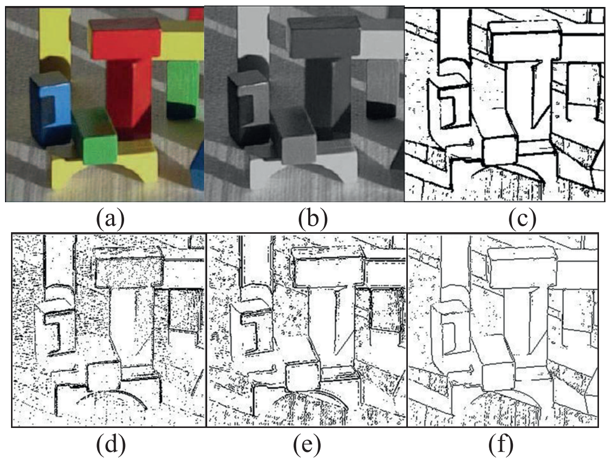

Input image
Smoothing by adaptive median filter
Red
Green
Blue
Transformed pixel value calculation Pixel(i,j)=2*red(i,j)+3*green(i,j)+ 4*blue(i,j)
4 directional masks are applied over the transformed pixel values and maximum directional color difference is calculated
Thresholding is performed with a fixed value T=1.2*t
Thinning of edges with convolution masks
Final edge image output

Figure 3. Steps of the proposed method

III. RESULTS AND COMPARISONS

Comparisons with several existing methods have been made to show the efficacy of the proposed method. Here we have considered the block image for comparison. Edge maps produced with Sobel operator, Laplace operator, Mexican hat operator, Cumani operator with different parameters, and finally with the proposed method are shown. It clearly shows that the output of the proposed method is easily comparable and in some cases better than the existing methods.



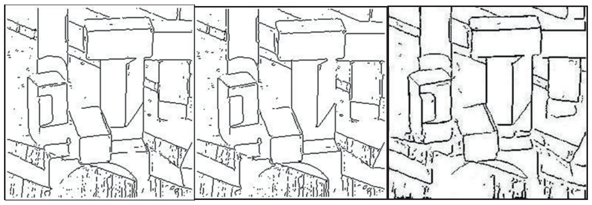


Figure 7. (a) Original block image, (b) gray image, (c) result of Sobel operator, (d)result with Laplace operator, (e) result with Mexican hat operator(σ=1.0), (f) result with Cumani operator with Gaussian mask (σ = 0.5), (g) result with Cumani operator with (σ=1.0), (h) Cumani operator with (σ= 1.0) applied to the intensity block image, (i) result of proposed method with T=1.2t.

Undoubtedly, as the above results demonstrate, the proposed method performed better than Sobel operator, Laplace operator, Mexican hat operator. In figure 7.(h) Cumani operator is applied to the intensity block image (σ= 1.0) produced good result. Proposed method in figure 7.(i) produced satisfactory output with less spurious edges than others with T=1.2t and contains most of the important edges.

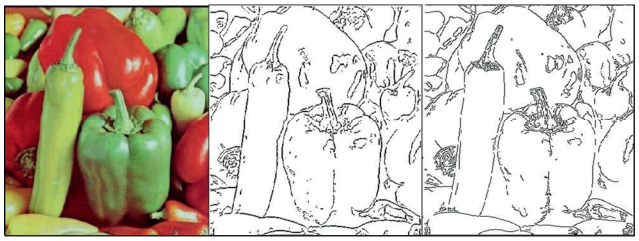


Figure 8. (a) Original pepper image, (b) proposed method output with T=1.2t, (c) openCV canny edge detector (low=70, high=120)

Figure 8.(a) is the pepper image, 8.(b) is the result is the proposed method and 8.(c) is the output of openCV canny edge detector with low=70 and high=120. This comparison clearly shows that the proposed method produces clear edge map with good continuity and easily comparable with famous canny edge detector.

IV. ACCURACY ANALYSIS OF PROPOSED METHOD

Here we have presented a qualitative method of accuracy analysis of the proposed method with the openCV canny edge detector. The accuracy of an edge detector can be measured with respect to another standard method by comparing the edge locations detected. For simplicity of analysis, we have selected a synthetic color image of dimension 139X157. Two vertical edges exist in the image one at 50th pixel location and another at 100th pixel location.

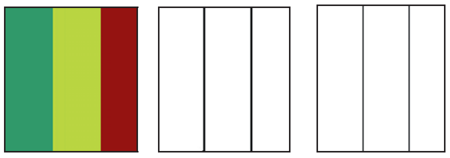


Figure 9. (a) Original image (139X157), (b) edge map by proposed method, (c) edge map by openCV canny edge detector

We have selected 50th row and plotted combined pixel values following equation 1, moving horizontally through the image shown in figure 9.(a) Also we have plotted the detected edge locations for both proposed and canny edge detector.

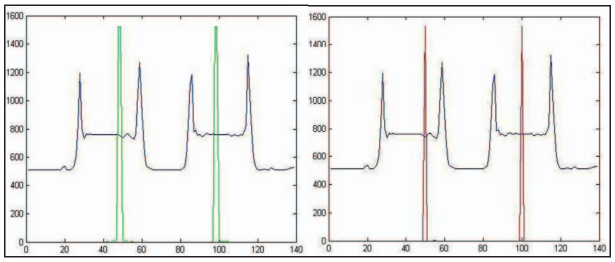


Figure 10. (a) Plot of proposed method, (b) Plot of openCV canny edge detector

Comparing figure 10.(a) and 10.(b) it is evident that the proposed method detects proper edges at proper location and also no false edge is detected.

V. CONCLUSION AND FUTURE WORK

The proposed method is tested on different images. It produced stable and fairly good results. Consistent acceptable outputs over different kinds of real life images have proved robustness of the presented scheme. Thus, the proposed method may be handy for any computer vision task where extraction of edge maps is required for a large set of images for feature extraction or for any other work. Our next venture will be comparing those algorithms with the proposed one and analyze the performance on the basis of parameters like computing time, execution complexity and accuracy of the system output in presence of noise.

REFERENCES

[1] J. F. Canny, .A computational approach to edge detection,. *IEEE Trans.Pattern Anal. Machine Intell.*, vol. 8, no. 6, pp. 679.698, Nov. 1986.

[2] C. L. Novak and S. A. Shafer, “Color edge detection,” in *Proc. DARPA Image Understanding Workshop*, 1987, pp. 35–37.

[3] R. Nevatia, “A color edge detector and its use in scene segmentation,” *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, no. 11, pp. 820–826, Nov. 1977.

[4] Rafael C. Gonzalez, Richard E. Woods, Digital Image Processing, Second Edition.

[5] A. Shiozaki, “Edge extraction using entropy operator,” *Computer Vis.,Graph., Image Process.*, vol. 36, no. 1, pp. 1–9, Oct. 1986.

[6] M. A. Ruzon and C. Tomasi, “Color edge detection with compass operator,” in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, Jun. 1999, vol. 2, pp. 160–166.

[7] A. Cumani, “Edge detection in multispectral images,” *CVGIP: Graph. Models Image Process*, vol. 53, no. 1, pp. 40–51, Jan. 1991.

[8] S. K. Naik and C. A. Murthy, “Standardization of Edge Magnitude in Color Images,” *IEEE Trans. Image processing,* vol. 15, no. 9, Sept. 2006.

[9] M. A. Ruzon and C. Tomasi, “Edge, junction, and corner detection using color distributions,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol.23, no. 11, pp. 1281–1295, Nov. 2001.00X.