ENHANCEMENT AND PERFORMANCE ANALYSIS ON DUTTA AND CHAUDHURI COLOR EDGE DETECTION ALGORITHM IN RGB COLOR SPACE

Mohamed D Almadhoun
University College of Applied Sciences
Palestine, Gaza
mdmadhoun@ucas.edu.ps

ABSTRACT

Edge detection process has a widespread usage in computer vision applications. The output detected edges of process changes when input image changes from color to grayscale. This changeability of results makes us add modifications on edge detection process procedures to correctly detect all those edges in color images that couldn't be detected in gray ones. This research reviews the proposed solution of Dutta and Chaudhuri on color edge detection algorithm that works using RGB color space, detects problem of a huge set of undetected edges by their proposed algorithm and finds solutions for that, and applies complexity and performance analysis and experiments to compare the proposed algorithm with Canny edge detection algorithm.

KEYWORDS

Computer vision, Color edge detection, Enhancement on edge detection, Complexity analysis, Performance Comparison.

1 INTRODUCTION

Edges are sudden variations in the gray level or color of image pixels [2], edge detection is an important operation for reducing processed data with preserving useful and informative object boundaries [1]. Original color edge detection uses some specific procedure consisting of classical operators to combine the RGB colors of an image to shape the original edges of image [3], then applies a traditional edge detection algorithms which passes through three steps: Firstly, it starts by reducing noise which acts like fake edges for the edge detection algorithms, various noise filters can be employed to tackle this problem like the proposed algorithm in [9] which is based on mean shift algorithm to remove noise while minimizing loss of edges. Secondly, edge enhancement is applied to strengthen edges and suppress elsewhere which is expressed as high pass filter. And finally, edge localization is used to apply threshold for deciding which pixels are edges and which are not [2]. Research has shown that 90% of edges in color images can be found in their corresponding grayscale images, and the remaining 10% of edges may not be detected in intensity images due to change in color, which may cause a fail of vital computer vision tasks [1] [4].

Dutta and Chaudhuri in [1] proposed an algorithm for edge detection of color images which starts by median filter to suppress unwanted noise in the image, then maximum directional differences of sum of gray values (Red+Green+Blue) are calculated for each pixel, then a single value auto-threshold is applied, and finally an edge thinning technique is applied to generate edge map [1]. A problem of RGB transformation procedure which was detected in Dutta and Chaudhuri algorithm makes a huge set of couple of different colors be transformed to a similar value by the RGB transformation procedure. This problem means there are lots of edges will not be detected in the color image.

By this research, an enhancement on Dutta and Chaudhuri algorithm will be proposed. In addition, a complexity analysis, and performance measurement operations in terms of execution time and number of basic operations will be shown with comparison to Canny edge detection algorithm.

2 ENHANCEMENT ON ALGORITHM

The second step of algorithm that was proposed by Dutta and Chaudhuri calculates a directional color difference. This step applied a transformation on the RGB pixel values to make it one value instead of three to reduce the computational overhead of color vector [1]. This step used Equation 1

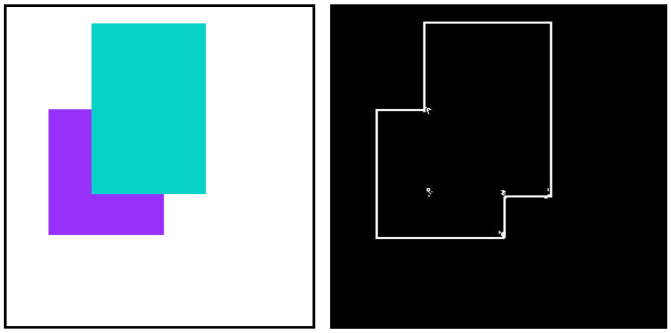
$$Pixel(i,j)=2*red(i,j)+3*green(i,j)+4*blue(i,j) \tag{1}$$

But this equation will fail in differentiating between lots of colors like the two colors (RGB=150,50,250) and (RGB=10,210,200), because the value of pixel(i,j) will be the same

$$Pixel(i,j)=2*150+3*50+4*250 = 1450$$

$$Pixel(i,j)=2*10+3*210+4*200 = 1450$$

So it will not consider the divider between two areas of those two colors as a boundary. Figure 1 images show how applying algorithm could not draw the edge between the two colors



(a) (b)

Figure 1. (a) Original input image. (b) Output image.

To solve this problem we have to consider a forth item in **Error! Reference source not found.**, this item should fulfill this differentiation. By experiment and from studying other color modes, it was found that Hue in the HSB is the most suitable item to finish this problem. Equation 2 shows the modification that can make algorithm detect this edge

$$Pixel(i,j)=2*red(i,j)+3*green(i,j)+4*blue(i,j)+2*Hue \tag{2}$$

By entering image of figure 1 (a) to the algorithm with modified calculation, we got out resulting image in Figure 2 with correct detected edges

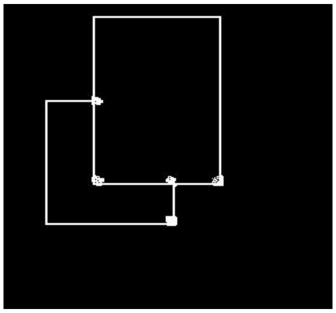


Figure 2. Output image after modification of calculating color difference

3 COMPLEXITY ANALYSIS

Basic operation is that operation which contributes most towards the running time of the algorithm. To analyze time efficiency of an algorithm, we count the number of repetitions of the basic operation as a function of input size [7].

Herewith, we can analyze Dutta and Chaudhuri algorithm by passing through its steps. Their algorithm consists of four main steps

1- Smoothing by adaptive median filter

Number of comparisons in this step is at most=number of window comparisons x n=(9+16+25+36+49+64+72)xn = 271 x n

Where n is number of image pixels

And in the best case it will be 9xn, when always med>min and med<max

So complexity is Θ(n)

2- Directional Color Difference Calculation

   This step runs two different loops, number of assignments in the first loop is n, where n=number of image pixels, and number of assignments in the second loop is at most 4xn

   So, it's Θ(n).

3- Threshold Technique

   Number of times of executing basic operation in this step is n, where n=number of image pixels, so it's Θ(n).

4- Edge thinning

   This operation has 2xn of assignments and complexity of Θ(n) when n=number of image pixels.

As for Canny edge detection algorithm, it is multi-stage process, starts by detecting edges with maximizing probability of edge points and minimizing non-edge points, then localization and resulting one real edge [6], by passing through the algorithm it follows these steps:

1- Convert to gray: costs n iterations if n is the number of image pixels

2- Normalize contrast: runs n+255+n, so its complexity is Θ (n), where n is number of image pixels.

3- Compute gradients: costs at most is 3 x (k x n)+ n iterations for the four loops, so it's Θ(n).

4- Perform hysteresis: costs at most n, so it's Θ(n).

5- Threshold edges: costs n iterations, so its complexity is Θ(n)

It's clear that both algorithms run in a cost of Θ(n), which means they are equal from perspective of time complexity, but when comparing number of basic operations repetitions, Canny repeats operations less than the proposed algorithm repetitions and this makes Canny finishes his job faster especially when applying algorithms on multi images like processing video.

4 PERFORMANCE MEASUREMENT AND ANALYSIS

Comparing different edge detection algorithms makes us do the best choice of using one of them in some application. Performance measurement experiment is needed to compare performance of the two edge detection algorithms: Canny, and the proposed by Dutta and Chaudhuri. Ten images were downloaded from Cambridge image database, image resolution was 816x616, with jpeg format, and RGB color.

In each of the two edge detection algorithms, a counter for counting the number of basic operations were added inside its code, and another statement for getting the execution time of each algorithm.

Performance measurement experiment applied the two algorithms on one image at first, then on two images, on three, until reaching to apply 10 images at a time for each algorithm. Applying algorithm in a loop on more than one image discovers the difference between them when using algorithm in video processing.

Since we are interested to measure how quickly program is executed, performance metric of computer program is the time required to execute it and finish its job completely. The basic technique to measure time of execution is by reading the difference value between clock time at the start of process and the clock time at the end of process [8].

Figure 3 shows the curves of execution time values for processing multi-images in each algorithm. This experiment was applied using PC computer with Dual core CPU of 1.73 GHz, 2 GB RAM, and 32-bit windows OS. It’s clear that the angle of canny curve is the smallest, and execution time values in all ten stages of canny experiment are the lowest, that means canny edge detection algorithm is the most suitable for video processing operation.

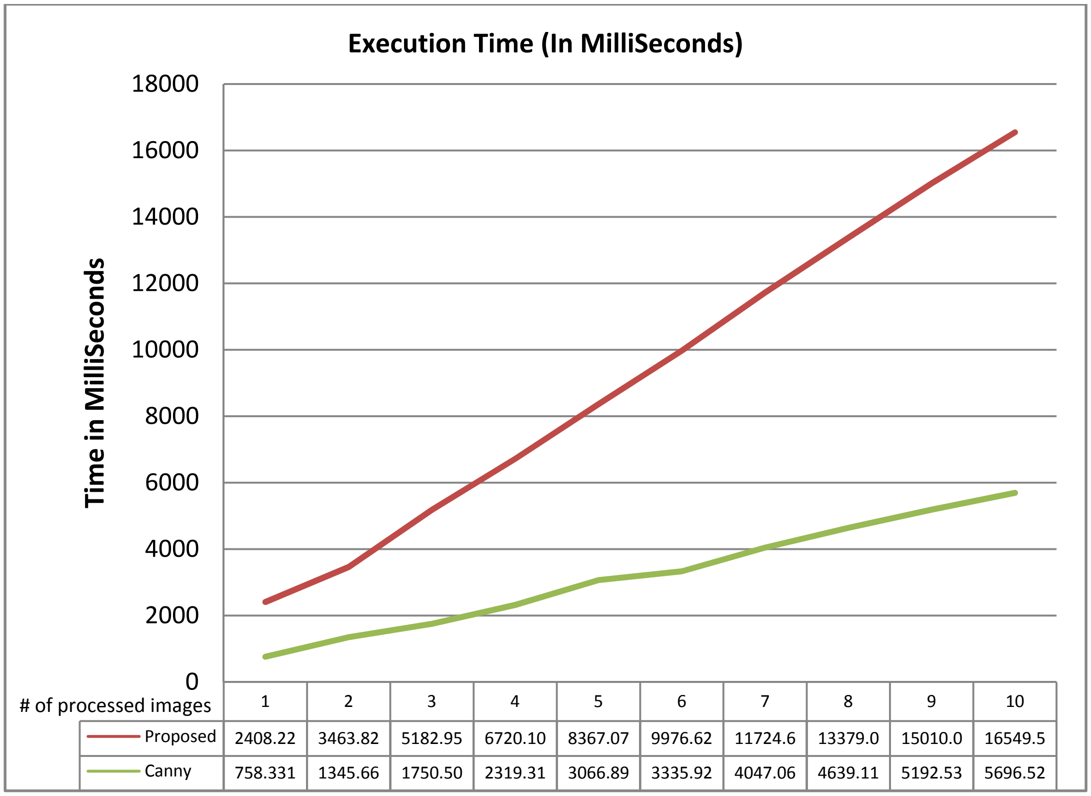
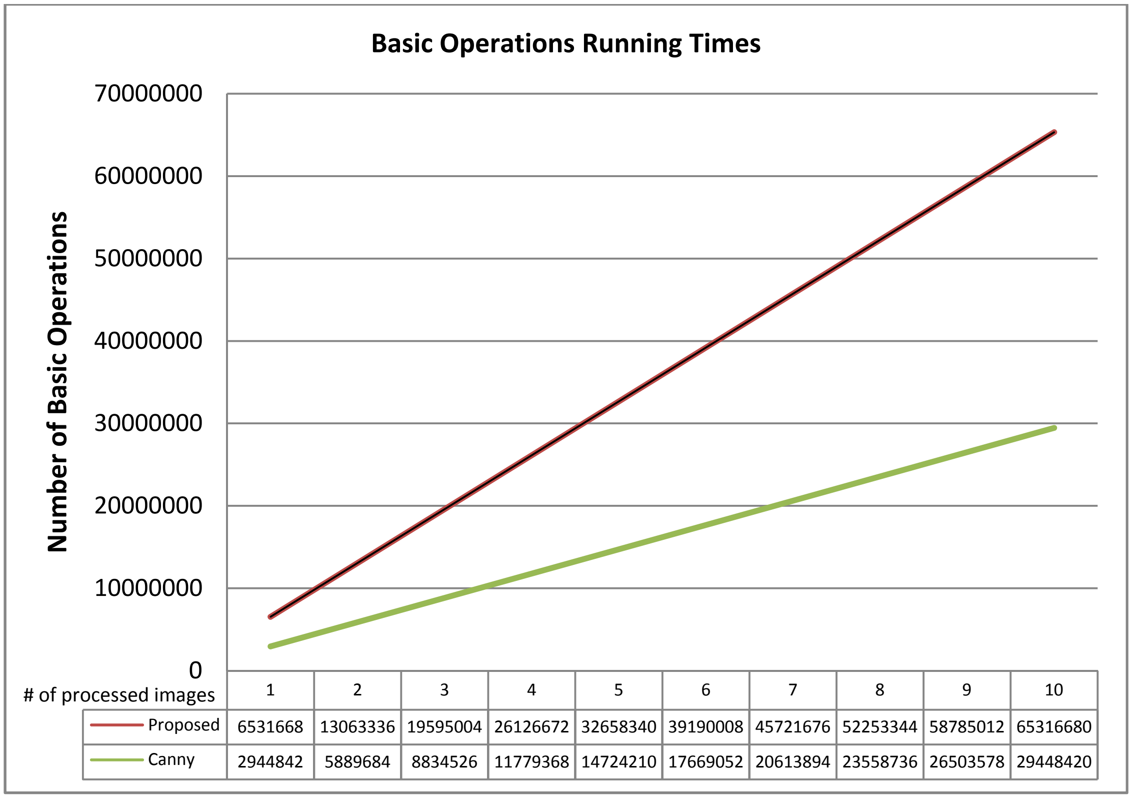


Figure 3. Curves of execution time values for processing multi-images in each algorithm

As for counting number of basic operations, Figure 4 shows that canny has the smallest number of basic operations.

5 CONCLUSION

By this research, problem of failed detection of edges between a set of colors in Dutta and Chaudhuri color edge detection algorithm in RGB color space was defined and resolved by adding Hue value from HSB color space to the procedure of transformation from RGB to single distinguishing value. In addition, a complexity analysis was applied on Dutta and Chaudhuri proposed algorithm, and compared to Canny edge detection algorithm with respect to time execution and number of basic operations. Comparison results show that Canny has better performance properties.



# of processed images	1	2	3	4	5	6	7	8	9	10
Proposed	6531668	13063336	19595004	26126672	32658340	39190008	45721676	52253344	58785012	65316680
Canny	2944842	5889684	8834526	11779368	14724210	17669052	20613894	23558736	26503578	29448420

Figure 4. Curves of number of basic operations for processing multi-images in each algorithm

6 REFERENCES

1. Dutta, S., Chaudhuri, B.: A Color Edge Detection Algorithm in RGB Color Space. In: Proc. International Conference on Advances in Recent Technologies in Communication and Computing. pp. 337 - 340, (2009).
2. Efford, N.: Digital Image Processing - A Practical Introduction Using Java. Pearson Education Limited, ISBN 0-201-59623-7, (2000).
3. Zhang, L., Mao, X., Zhou, C.: Edge Detection of Color Image Based on HI*S* Color Space. In: Proc. International Conference on Information Engineering and Applications. Lecture Notes in Electrical Engineering Volume 154, 2012, pp 1529-1534, (2011).
4. Chen, H.: A novel color edge detection algorithm in RGB color space. In: Proc. IEEE 10th International Conference on Signal Processing (ICSP), (2010).
5. Arpitha, D., Arakeri, M., Reddy, R.: An Approach for Color Edge Detection with Automatic Threshold Detection. In: Proc. International Conference, ADCONS 2011. Lecture Notes in Computer Science Volume 7135, 2012, pp 117-124, (2011).
6. Cheng, Y.: An improved Canny Edge Detection Algorithm. Lecture Notes in Electrical Engineering Volume 126, pp 551-558, (2012).
7. Levitin, A.: Introduction to the Design and Analysis of Algorithms (2nd Edition). Addison Wesley, ISBN-10: 0321358287 | ISBN-13: 978-0321358288, (2006).
8. Lilja, D.: Measuring Computer Performance: A Practitioner's Guide. Cambridge University Press, New York, NY, ISBN 0-521-64105-5, (2000).
9. Shim, S., Malik, A., Choi, T., Noise reduction using mean shift algorithm for estimating 3D shape. Imaging Science Journal, Vol. 59, No. 5, pp. 267-273, (2011).