Turkish Journal of Computer and Mathematics Education Vol.12 No. 7 (2021), 2654-2659

Research Article

Two Stage Distributed Canny Edge Detector For Images Corrupted With Gaussian Noise

[1]Dr.M.Pompapathi, [2]Smt G.Swetha

[1]Assoc.Professor, Department of IT, RVR & JC College of engineering, Chowdavarm, Guntur, Andhra Pradesh, INDIA
E-mail:Manasani.pompapathi@gmail.com
[2]Asst.Professor, Department of IT, RVR & JC College of engineering, Chowdavarm, Guntur, Andhra Pradesh, INDIA
E-mail: ursgadde@gmail.com

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 16 April 2021

Abstract- In classification and recognition of objects the most commonly used features are edges, which are the locations where the intensity values change more than a predefined threshold value. The widely used edge detection algorithms is Canny edge detection method due to its superior performance. Directly applying the original Canny detector on noisy images will identify noise content also as edge content. The proposed method is a two stage filter , will restore the image corrupted by Gaussian noise using Nonlinear filtering and then applies distributed Canny edge detection algorithm that adaptively computes the thresholds used to identify the edges based on the type of a block and the local distribution of the gradients in the image block. This proposed scheme, obtains accurate edge pixels from a denoised image with higher Figure of Merit (FOM values.

Keywords: Canny edge detector; Denoisining; Gradient; Texture; FOM;

1 Introduction

Currently in real world the visual information is acquired or communicated in the form of digital images. After acquisition/transmission image is often degraded with some unwanted error information called noise. Further before using in any applications, the received image needs to be processed by using various linear or nonlinear filters [4] in order to recover to the original content. In image segmentation the edge detection algorithms are mainly used for image sharpening. The edge detection algorithm is designed to detect and highlight the discontinuities in the image intensity values. Edge detection has acquired enormous importance in computer vision research and for classification and recognition of objects.

In real world to detect the edges from noisy image most of the methods adopt the two stage filtering mechanisms. During first stage various filters[1]-[7] applied to restore the image from corrupted noise content and then during second stage edge detectors[8]-[12] is applied to detect the discontinuities in the image which further can be used for object recognition. Most widely used simple Canny edge detector [3], [8], [10] obtains the edges from the synthetic and real images. Canny edge detector performs the following four steps to obtain the edge images from the given input image.

1. Regularization of an image with Gaussian filter
2. Calculation of gradient.
3. Apply non-maximal suppression to obtain thin edges.
4. Apply double thresholding to detect the final edges.

This paper presents a method called Two Stage Distributed Canny Edge Detector for Noisy Images (TSDCEDNI).The paper is organized as follows. The proposed TSDCEDNI method in detail step by step is given in section 2, the results and discussions are given in section 3 and finally conclusion is given in section 4.

2 Proposed Two Stage Distributed Canny Edge Detector for Images (TSDCEDNI) corrupted with Gaussian noise

In order to improve the performance and efficiency of general canny detector [3] used to detect the edges from the given input image, we proposed Two Stage Distributed Canny Edge Detector for Noisy Images (TSDCEDNI) with improved FOM Value. The block diagram Fig.1 shows the steps involved in proposed method used to find denoised image by non-linear pre-filtering during step I, and distributed block based canny edge detector applied in step II to find the edges of an image.

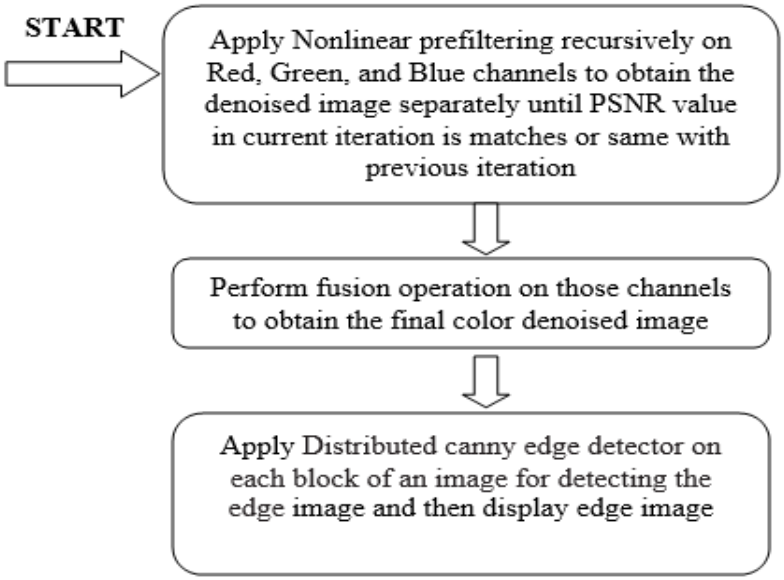

Fig.1: Block diagram for proposed method Two Stage Distributed Canny Edge Detector for Images (TSDCEDNI) corrupted with Gaussian noise

The proposed method first obtains the denoised image by using nonlinear prefiltering during step I and then applies distributed canny edge detector for improving the identification accuracy of edge location in step II based on blocks type.

Step I. Nonlinear prefiltering method to restore image - To obtain the denoised image during first step we adopt the following procedure.

1. Construct Histogram for noisy image F
2. Identify the type of noise
3. If noise is Gaussian noise then perform the following steps to obtain restored image.
4. Apply the following Non-linear Prefiltering on noisy image to produce denoised image recursively on each channel of a color noisy image until PSNR value in the current iteration is same or nearer to previous iteration.

First decompose the corrupted Gaussian noisy image into F_{Red}, F_{Green}, and F_{Blue} channels. Let F is an image corrupted with Gaussian noise and $F^{(i)}$ represents the multichannel image at iteration i, where i=0 is the corrupted noisy image. Let $F_k^{(i)}$ be the pixel value ($0 <= F_k^{(i)} <= L-1$) at the location (x,y) in the kth channel $F_k^{(i)}$ where k=1 is a Red channel F_r, k=2 is the Green channel F_g and k=3 is a blue channel F_b. The prefiltering takes into account the absolute differences between the pixel to be processed and its neighbors, differences with smaller value are considered as noise and are to be reduced and differences with large value are considered as edges and are to be continued with restored image. Let $F_k^o(x+m, y+n)$ where m,n = -1, 0, +1 and (m,n) not equal to (0,0) be group of eight neighboring pixels that belong to 3x3 window as shown in Fig.2.

$F_k^o(x-1, y-1)$	$F_k^o(x-1, y)$	$F_k^o(x-1, y+1)$
$F_k^o(x, y-1)$	$F_k^o(x, y)$	$F_k^o(x, y+1)$
$F_k^o(x+1, y-1)$	$F_k^o(x+1, y)$	$F_k^o(x+1, y+1)$

Fig.2. Neighboring pixels of $F_k^o(x, y)$ currently processing pixel in a 3 X 3 window

New intensity value for the restored image for next pass is given by Eqn.1

$$F_k^{(1)}(x, y) = F_k^{(0)}(x, y) + 1/8\Sigma_m \Sigma_n \zeta_k(F_k^{(0)}(x + m, y + n), I_k^0(x, y)) \quad (1)$$

where m=-1,0,+1 and n=-1,0,+1 and (m, n) is not equal to (0,0) and ζ_k is a parametric based nonlinear function[14] The nonlinear prefiltering mechanism during this step aims at gradually excluding pixel values that are very different from the central element, in order to avoid blurring the image details during noise removal.

The above step is applied for separated three channels Red(F_r), Green(F_g) and Blue(F_b) channels separately with p=1.

5. Then finally construct restored image in color from these channels by using fusion of Red(F_r), Green(F_g) and Blue(F_b) channels. Iterate the step-I by incrementing p value until the PSNR value of the restored image in current iteration is more compared to the previous iteration.

Step II. Distributed canny edge detector for detecting the edges- The classical Canny method sets the Maximum and minimum thresholds based on the spread of the gradients at all the pixels of an image. If directly applied original Canny algorithm to a block of the image leads to failure in detection of desired edges I nthe given block ecause the statistical features of a block will differ highly if compared to the statistics of the whole image. Here in second step we applied the distributed canny edge method [13] to obtain the edge content by processing block by block based on block's statistical features.

In the distributed canny method, the input image is decomposed into *m X m* overlapping blocks and these blocks are processed independent of each other. For an $L \times L$ gradient mask, the *m* X *m* overlapping blocks are obtained by first dividing the input image into $n \times n$ non-overlapping blocks and then extending each block by $(L + 1)/2$ pixels along the left, right, top, and bottom boundaries, respectively. This results in *m* X *m* overlapping blocks, with $m = n + L + 1$.

First divide the input image into *n* X *n* non-overlapping blocks and then blocks are classified into the following six categories, uniform, uniform/texture, texture, edge/texture, medium edge, and strong edge block, by adopting the block classification method. This classification method utilized the local variance of each pixel using a 3 × 3 window that is centered around the considered pixel in order to label it as of type edge, texture, or uniform pixel. Then, each block is classified based on the total number of edge, texture, and uniform pixels in the considered block. This process involves first we need to classify the pixel and then block as given below.

Step 1: Classification of pixel(x,y) in a blcok-

$$\text{Pixel type} = \begin{cases} \text{Uniform pixel} & - \text{var}(x,y) <= T_u \\ \text{Texture} & - Tu < \text{var}(x,y) <= T_e \\ \text{Edge} & - T_e < \text{var}(x,y) \end{cases}$$

Step 2: Classification of n X n Non-overlapping block

Block type	**No: of pixels of pixel type**	
	$N_{uniform}$	N_{edge}
Smooth	>=0.3*Total_Pixel	0
Texture	<0.3*Total_Pixel	0
Edge/ Texture	<0.65*(Total_Pixel-N_{edge})	(>0)&(<0.3*Total_Pixel)
Medium edge	>=0.65*(Total_Pixel-N_{edge})	(>0)&(<0.3*Total_Pixel)
Strong edge	<=0.7*Total_Pixel	>=0.3*Total_Pixel

Here Var(x, y) is the local (3 X 3) variance at pixel (x, y); The values of T_u and T_e are initialized to 100 and 900 respectively; Total_Pixel : total number of pixels in the block; $N_{uniform}$ is the total number of uniform pixels in the block; N_{edge} is the total number of edge pixels in the block;

The threshold in each block based on its type is calculated as follows.

Let P_1 be the percentage of pixels, in a block, that would be classified as strong edge.

Step a): If block type is smooth then P1=0; means no edge content in a block

else if block type is texture then P_1=0.03; means Few edges are in a block

else if block type is texture/edge then P_1=0.1; means Some edges in block

else if block type is medium edge then P1=0.2; means Medium edges in block

else P_1=0.4; means Many edges in a block

Step b) : Compute the 8- bin non-uniform gradient magnitude histogram and the corresponding Cumulative distribution function F (G)

Step c): Compute High threshold as F (High threshold) = 1-P_1

Step d): Compute Low threshold =0.4*High threshold

Step e): Use these thresholds to decide edge candidature for each pixel from a denoised image.

3 Results and Discussions

The results obtained by applying nonlinear prefiltering to obtain the denoised image and distributed canny edge detector applied after nonlinear prefiltering are shown from Fig.3 to Fig.5. as a subjective analysis.

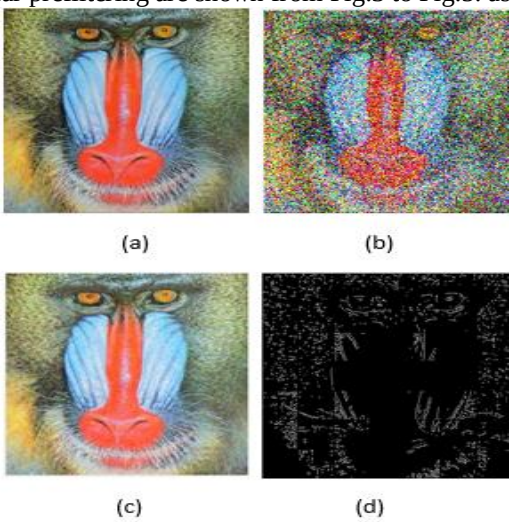

Fig.3. (a) Original baboon image (b) Image with Gaussian noise (c) Denoised image by applying nonlinear prefiltering (d) Edge image obtained by applying distributed canny edge detector block by block on denoised image in (c)

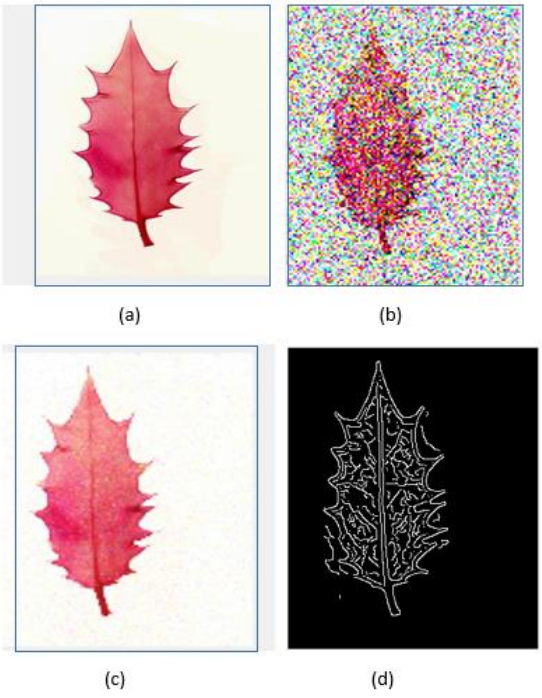

Fig.4. (a) Original leaf image (b) Image with Gaussian noise (c) Denoised image by applying nonlinear prefiltering (d) Edge image obtained by applying distributed canny edge detector block by block on denoised image in (c)

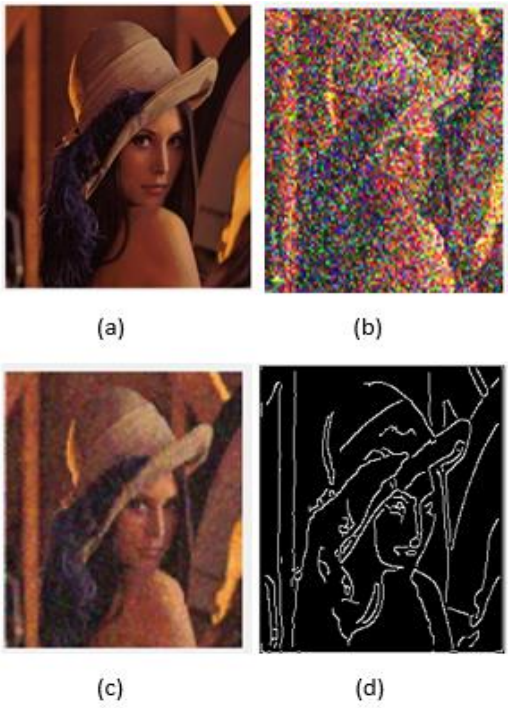

Fig.5. (a) Original lena image (b) Image with Gaussian noise (c) Denoised image by applying nonlinear prefiltering (d) Edge image obtained by applying distributed canny edge detector block by block on denoised image in (c)

To show the objective fidelity, we used the performance measure to define the exact location of edge pixels are identified by proposed method is Figure of Merit

The Figure of Merit (FOM) is an edge preserving measure:

$$FOM=1/\max\{N, N_{ideal}\} \sum_{i=1}^{N} 1/(1+d_i\lambda)$$

Where N and N_{ideal} are the numbers of detected edge pixels in the noisy and the original image, respectively. D_i is the Euclidean distance between the i^{th} detected edge pixel and the nearest original edge pixel, and λ is a constant typically set to 1/9. The dynamic range of FOM is between 0 and 1. The following TABLE I shows the calculated FOM values on a set of images.

TABLE I -FOM values on a set of images using proposed method in comparison with standard canny without blocks

Image	FOM value calculated by applying original canny edge image on denoised image without blocks	FOM value calculated by applying proposed distributed canny edge method on denoised image with block nature
Lena image	0.3589	0.7899
Baboon image	0.2566	0.7566
Leaf image	0.4611	0.8999
Coala image	0.4010	0.8710
Cameraman Image	0.3900	0.8814
Pepper image	0.4566	0.8992
Desert image	0.5621	0.8190

The proposed method outperforms well in terms of exactly allocating pixel based on its nature in each block. The results shown in Table I shows that FOM value is nearer to 1 if distributed method is applied to obtain the edge image after step I.

4 Conclusion

The proposed Two Stage Distributed Canny Edge Detector for Noisy Images obtains denoised image in by nonlinear prefiltering by preserving all edge pixels and exhibits improved FOM value by applying block type based distributed canny over denoised image to obtain the edge pixels. Proposed algorithm computes edges of more than one block in the given image at the same time. This algorithm is scalable irrespective of image size and types of blocks and has very high edge detection performance. This algorithm can detect all psycho-visually important edges in the image for various block sizes.

REFERENCES

1. Computer and Robot vision by Robert M. Haralick and Linda Shapiro.
2. Image Processing by Rafael C. Gonzalez.50
3. Image Processing Analysis, and Machine Vision by Milan Sonka, Vaclav Hlavac, Roger Boyle.
4. Computer Vision: A Modern Approach by David A. Forsyth, Jean Ponce.
5. Machine Vision by R. Jain, R. Kasturi, and B.G. Schunk.
6. Fabrizio Russo and Annarita Lazzari,(2005)," Color Edge Detection in Presence of Gaussian Noise Using Nonlinear Prefiltering", IEEE Transactions On Instrumentation And Measurement, Vol. 54, No. 1, PP:352-358.
7. I. K. Park, N. Singhal, M. H. Lee, S. Cho, and C. W. Kim, "Design and performance evaluation of image processing algorithms on GPUs,"IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 1, pp. 91–104, Jan. 2011.
8. R. Deriche, "Using canny criteria to derive a recursively implemented optimal edge detector," Int. J. Comput. Vis., vol. 1, no. 2, pp. 167–187,1987.
9. D. V. Rao and M. Venkatesan, "An efficient reconfigurable architecture and implementation of edge detection algorithm using handle-C," in Proc. IEEE Conf. ITCC, vol. 2. Apr. 2004, pp. 843–847.
10. L. H. A. Lourenco, "Efficient implementation of canny edge detection filter for ITK using CUDA," in Proc. 13th Symp. Comput. Syst., 2012, pp. 33–40.
11. J. F. Canny, "A computation approach to edge detection," IEEE Trans.Pattern Anal. Mach, Intell., vol. 8, no. 6, pp. 769–798, Nov. 1986.
12. S. Nercessian, "A new class of edge detection algorithms with performance measure," M.S. thesis, Dept. Electr. Eng., Tufts Univ., Medford, MA, USA, May 2009.
13. Qian Xu, Srenivas Varadarajan, Chaitali Chakrabarti, Fellow, IEEE, and Lina J. Karam, Fellow, IEEE "A Distributed Canny Edge Detector: Algorithm and FPGA Implementation "IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 23, NO. 7, JULY 2014, pp:2944-2960.
14. A.Sri Krishna, .B.Eswara Reddy, M.Pompapathi "Color Edge Detection for Noisy Images by Nonlinear Prefiltering and Block-by-block Rotations" IEEE ICCSP 2015 conference, 978-1-4799-8081-9/15/$31.00 © 2015 IEEE, pp:1262-1267.