International Journal on Applications in Information and Communication Engineering
Volume 1: Issue 12: December 2015, pp 4-8. www.aetsjournal.com ISSN (Online) : 2394-6237

Design and Simulation of Distributed Canny Edge Detection Algorithm

Sreelal T, Dr. A. Kishore Kumar

Abstract-Edge detection is an important step in digital image processing and is defined as a set of mathematical operations which aim at identifying points in a digital image at which the image brightness changes sharply or has discontinuities. The points at which image brightness changes sharply are termed as edges. Edge detection is used for image segmentation and feature extraction in the areas of image processing, computer vision, and machine vision etc. The Canny edge detection algorithm is one of the edge detection algorithms that use a multi-stage algorithm to detect a wide range of edges in images. Among the edge detection methods developed so far, Canny edge detection algorithm is one of the most commonly used methods that provides good and reliable detection because it has advantages like detection of edge with low error rate, good localization, minimal response, improved signal to noise ratio and better detection especially in noise conditions. This paper proposes a method to implement the Canny edge detection algorithm at the block level without any loss of edge detection performance compared with the original frame-level Canny edge detection algorithm. The method is called distributed Canny edge detection algorithm. The proposed algorithm is designed and simulated in MATLAB. The results of the proposed distributed Canny edge detection algorithm has better edge detection performance compared to the existing frame-level Canny edge detection algorithm.

Keywords-Digital Image Processing, Canny Edge Detection Algorithm, MATLAB

I. INTRODUCTION

Interpretation of contents in an image is an important objective in image processing, computer vision and machine vision applications and it has received much attention of researchers during the last few decades. An image contains different information of a scene, such as object's shape, size, color and orientation etc. But the identification of the objects from their background is the first essential task that should be performed before any interpretation. In order to extract the shape of an object, we have to detect the edges forming that object, and this fact shows the importance of edge detection in computer vision and image processing. Edge detection results have wide range of applications such as image enhancement, image recognition, image segmentation, feature extraction and feature detection etc.

Sreelal T, PG Scholar, Department of Electronics and Communication Engineering, Hindusthan College of Engineering and Technology, Coimbatore, Tamil Nadu, India.

Dr. A. Kishore Kumar, Associate Professor, Department of Electronics and Communication Engineering, Hindusthan College of Engineering and Technology, Coimbatore, Tamil Nadu, India.

An edge consists of a connected sequence of edge pixels. An edge pixel is a pixel at which the intensity of an image function changes abruptly. Edge detectors are designed to identify and locate the edge pixels. Edge detectors can be simple or complicated depending on how well they localize the edge and how well they handle the noise content and false responses. A sharp image has stronger edges where as a blurred image has weak edges. Hence boundaries of objects are more clearly visible in sharp images. A strong edge represents a sharp change in intensity. Such discontinuities in the intensity profile can be detected using the derivative operator. So edge detection algorithms are broadly classified into two categories based on the derivatives of the image, first order derivative/gradient based edge detector and second order derivative/Laplacian based edge detector.

Roberts Cross, Sobel and Prewitt operators are different types of first order classical edge operators. Here the input image is convolved by a mask to generate the gradient of the image in which edges are detected by thresholding operation. The set of works in [2], [4] deals with the implementation of classical edge detectors. The idea behind the Roberts Cross edge detection algorithm is to find the gradient of an image through discrete differentiation, achieved by computing the sum of the squares of the differences between diagonally adjacent pixels. One of the most attracting aspects of this method is its simplicity; the mask used is small and contains only integers as shown in [6]. However with the speed of computers today this advantage is negligible and also the Roberts Cross method is highly sensitive to noise. Sobel operator is a discrete differentiation operator which computes the gradient of an image intensity function. At each point in the image, the result of the Sobel operator is either the corresponding gradient vector or the norm of this vector. The Sobel operator is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical direction. The Prewitt operator is also a first order derivative edge operator which functions similar to the Sobel operator, by computing the gradient for the image intensity function. As compared to Sobel, the Prewitt masks are simpler to implement but are very sensitive to noise.

Another classification of edge detectors is based on second order derivative/Laplacian operator. Edges can be identified using the first order derivative; however, to localize the edge we need to identify the peaks in the derivative magnitude. It is explained in [10] that the magnitude of second order derivative of the intensity profile crosses zero at the location of the peak of the first order derivative. Zero crossing is flanked by positive and negative peaks which indicate, respectively, the dark and the bright side of the edge. This property is used in detection of edges in the images. Marr-

Hildreth edge detector is an example for second order edge detector which uses zero-crossing property to detect edges. But the limitations of Marr-Hildreth edge detector are detection of false edges and localization error is severe at curved edges.

II. CANNY EDGE DETECTION ALGORITHM

The Canny edge detection algorithm was developed by John F. Canny in 1986. It is one of the most powerful edge detection methods. The Canny edge detection method uses two different thresholds to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be fooled by noise, and more likely to detect true weak edges. Canny proposed a list of criteria to improve the performance of the existing edge detection algorithms.

The first criterion is low error rate. It is important that the probability of detecting real edge points should be maximized and the probability of detecting false edges should be minimized. This corresponds to maximizing the signal-to-noise ratio. The second criterion is that the edge points be well localized. In other words, the detected edges should be as close as possible to the real edges. A third criterion is to have only one response to a single edge. This was implemented because the first two were not substantial enough to completely eliminate the possibility of multiple responses to an edge. Based on these criteria Canny developed his edge detection algorithm.

Figure.1 shows the block diagram of the Canny edge detection algorithm.

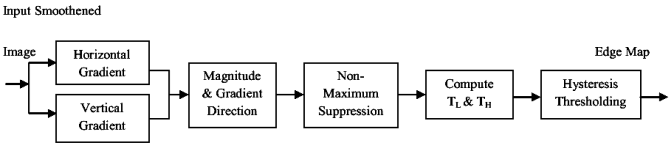

Figure.1 Block diagram of the Canny edge detection Algorithm

The Canny edge detection algorithm works on frame-level statistics and consists of the following steps.

1) Apply Gaussian filter to smooth the image in order to remove the noise.

2) Find the intensity gradients G_x and G_y by convolving with gradient masks.

3) Computing the gradient magnitude G and direction θ_G at each pixel location. The direction is rounded to one of four possible angles (Namely 0^o, 45^o, 90^o or 135^o)

4) Applying Non-Maximum Suppression (NMS) to thin edges. This removes pixels that are not considered to be part of an edge.

5) Hysteresis Thresholding: This is the final step. It uses two thresholds called upper and lower thresholds: If a pixel gradient is higher than the upper threshold, the pixel is accepted as an edge. If a pixel gradient value is below the lower threshold, then it is rejected. If the pixel gradient is between the two thresholds, then it will be accepted only if it is connected to a pixel that is above the upper threshold.

The Canny edge detector is predominantly used in many real-world applications due to its ability to extract significant edges with good detection and good localization performance. Unfortunately, the Canny edge detection algorithm contains extensive pre-processing and post-processing steps and is more computationally complex than other edge detection algorithms such as Sobel, Prewitt and Roberts Cross algorithms but it also has a higher latency because it is based on frame-level statistics. This also results in a decreased edge detection performance. Since the Canny edge detection algorithm operates on the whole image and has a latency that is proportional to the size of the image.

III. DISTRIBUTED CANNY EDGE DETECTION ALGORITHM

To improve the edge detection performance of the original frame-level Canny edge detection algorithm, a distributed Canny edge detection algorithm is proposed in this project work. A block diagram of the proposed distributed Canny edge detection algorithm is shown in Figure.2

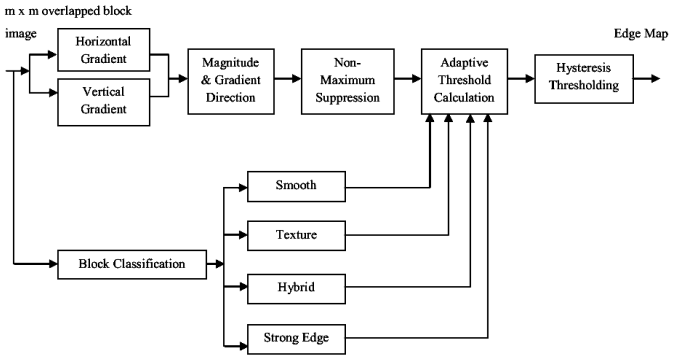

Figure.2 Block diagram of the proposed distributed Canny edge detection algorithm

The superior performance of the frame-level Canny algorithm is due to the fact that it computes the gradient thresholds by analyzing the histogram of the gradients at all the pixel locations of an image. Though it is purely based on the statistical distribution of the gradient values, it works well on natural images which consist of a mix of smooth regions, texture regions and high-detailed regions. By applying same high and low threshold values for each block in an image will lead to more number of edges in smooth regions and loss of important edges in highly detailed regions. In order to overcome this problem the high and low threshold values are chosen according to block-level statistics instead of frame-level statistics.

The proposed distributed Canny edge detection algorithm computes the low and high thresholds according to the block level statistics which detects more number of edges than the original frame-level Canny edge detection algorithm with reduced latency. So, the method can be used for the edge detection in real time applications.

IV. METHODOLOGY

The proposed distributed Canny edge detection algorithm follows the same steps as in original Canny edge detection algorithm except those steps are applied for each block in the image.

1) Image Smoothening

Edge detection results can be easily affected by noise, so it is essential to filter out the noise to avoid false detection caused by noise. To smooth the image, a Gaussian filter is used and is convolved with the image. This step will slightly smooth the image to reduce the effects of the noise on the edge detector. Figure.3 shows a 5 × 5 Gaussian filter, used for image smoothening, with standard deviation $\boldsymbol{\sigma}$ = 1.4. The input image convolved with this Gaussian filter to get the smoothened image.

$$\frac{1}{115}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Figure.3 A 5 × 5 Gaussian smoothening filter with standard deviation $\boldsymbol{\sigma}$ = 1.4

Before going to next step the smoothened image has to divide in to m × m overlapping blocks. In order to do this, first the image is divided into n × n non-overlapping blocks where n<m. For an L×L gradient mask, each block can be extended by (L-1)/2 along left, right, up and down. The non-overlapping blocks need to be extended in order to prevent edge errors and loss of edges at boundaries while computing the gradients and due to the fact that non maximum suppression operation at boundary requires gradient values of the neighboring pixels. So the adjacent blocks will overlap by (L − 1)/2 pixels for an L × L gradient mask. For a 3 × 3 gradient mask, the block has to be extended by 1 (where (L-1)/2 = 1) pixel on all sides in order to generate a block of size (n+2) × (n+2). Thus m=n+2 for this example.

2) Gradients and Gradient Magnitude Calculation

This stage calculates the vertical and horizontal gradient of the image using gradient masks. Sobel operator is used as gradient mask in this project work. The gradient masks used for Sobel operator is shown in the Figure.4. These gradient operators convolved with each blocks in the image and returns a value for the first order derivative in the horizontal direction, G_x and the vertical direction, G_y.

The magnitude of the gradient is given by the equation,

$$|G|=\sqrt{G_x^2+G_y^2}$$

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Figure.4 Sobel operator

3) Directional Non-Maximum Suppression

Since arctangent is a very complex function and also requires floating point numbers, it is very difficult to implement such functions for real time applications. So in this paper instead of finding the gradient direction by calculating the arctangent of vertical gradient to the horizontal gradient, i.e. θ = arctan (dy/dx), the value and sign of the components of the gradient is analyzed to calculate the direction of the gradient as given in [5].

Once the direction of the gradient is known, the values of the pixels found in the neighborhood of the pixel under analysis are interpolated as given in [5]. The pixel that has no local maximum gradient magnitude is eliminated. The comparison is made between the actual pixel and its neighbors, along the direction of the gradient. Let current pixel is $P_{x,y}$ and the derivative values at that pixel are dx and dy. Then for example, if the approximate direction of the gradient is between 0^0 and 45^0, the magnitude of the gradient at $P_{x,y}$ is compared with the magnitude of the gradient at adjacent points as shown in Figure. 5, where $P_{x,y} = |dx_{x,y}| + |dy_{x,y}|$.

The values of the gradient at the point P_a and P_b are defined as follows:

$P_a = (P_{x+1,y-1}+P_{x+1,y})/2$
Where $P_{x+1,y-1} = |dx_{x+1,y-1}| + |dy_{x+1,y-1}|$ and
$P_{x+1,y} = |dx_{x+1,y}| + |dy_{x+1,y}|$

$P_b = (P_{x-1,y+1}+P_{x,y+1})/2$
Where $P_{x-1,y+1} = |dx_{x-1,y+1}| + |dy_{x-1,y+1}|$ and
$P_{x,y+1} = |dx_{x,y+1}| + |dy_{x,y+1}|$

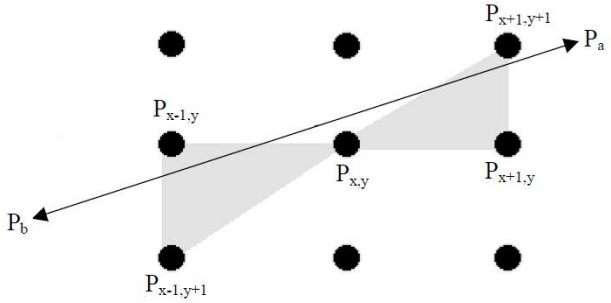

Figure.5 Pixel interpolation

The centre pixel $P_{x,y}$ is considered as an edge, if $P_{x,y} > P_a$ and $P_{x,y} > P_b$. If both conditions are not satisfied then the centre pixel is removed.

4) Pixel Classification

In this step we classify pixels in each block as uniform, texture and edge pixels using the criteria given in [1],

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

Where var(x,y) is the local 3 × 3 variance at pixel (x,y). T_u and T_e are two thresholds as given in [7].

5) Block Classification

After pixel classification, we classify each block into six types as, uniform, uniform/texture, texture, edge/texture, medium edge and strong edge block, by adopting the criteria [1] shown in Table.1.

Table.1 Criteria for block classification

Block type	No. of pixels of 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)

Total_Pixel: The total number of pixels in the block.
$N_{uniform}$: The total number of uniform pixels in the block.
N_{edge}: The total number of edge pixels in the block.

6) Adaptive Threshold Calculation

The method adopted for adaptive threshold calculation [1] is as follows. Let P1 be the percentage of pixels, in a block, that would be classified as strong edges,

```
Step 1: If smooth block type
        P1 = 0;        /*No edges*/
           else if texture block
        P1 = 0.03;     /*Few edges*/
           else if texture/edge block type
        P1 = 0.1;  /*Some edges*/
           else if medium edge block type
        P1 = 0.2;   /*Medium edges*/
           else
        P1 = 0.4;  /*Many edges*/

Step 2: Compute High_Threshold = 1-P1
Step 3: Compute Low_Threshold = 0.4*High Threshold
```

High thresholds and low thresholds are required for thresholding with hysteresis. Hysteresis thresholding is applied to each block using the values of T_H and T_L in an image to determine the edge image, instead of applying to the whole image as in original Canny edge detection algorithm.

V. MATLAB EXPERIMENTAL RESULTS

The proposed distributed Canny edge detection algorithm is designed in MATLAB. All images taken are gray scale images of size 512 × 512. The input image is smoothened with the help of a 5 × 5 Gaussian filter. The horizontal and vertical gradient of the image is computed with help of 3 × 3 Sobel operators. Since 3 × 3 Sobel operator is used the overlapping block size will become 66 × 66.

Figure.6 shows the output of the proposed distributed Canny edge detection algorithm at different stages. Figure.7 shows the comparison of edge detected output using existing frame-level Canny edge detection algorithm and proposed distributed Canny edge detection algorithm and Figure. 8 shows the comparison of edge detected output of the various existing edge detection algorithms.

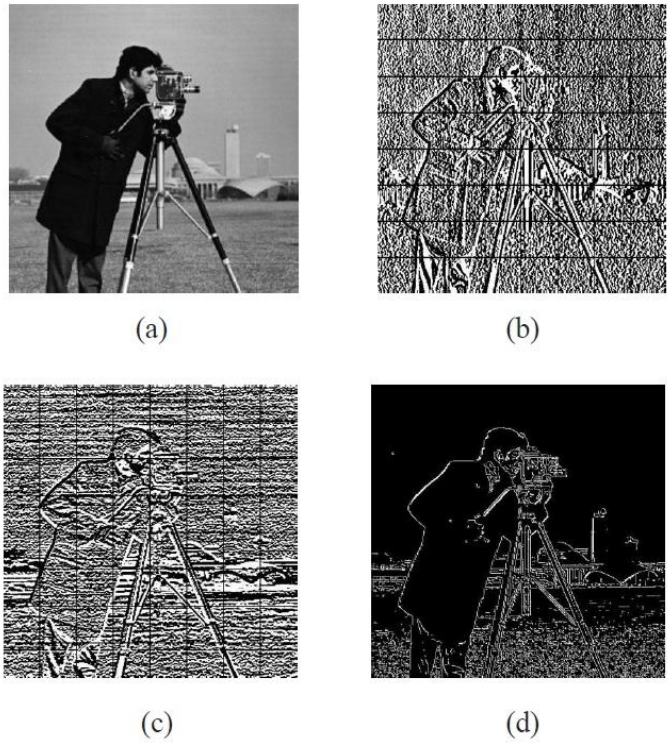

Figure.6 (a) Smoothened 512 × 512 Cameraman image (b) X-Gradient of the image (c) Y- Gradient of the image (d) Output of the proposed distributed Canny edge detection algorithm.

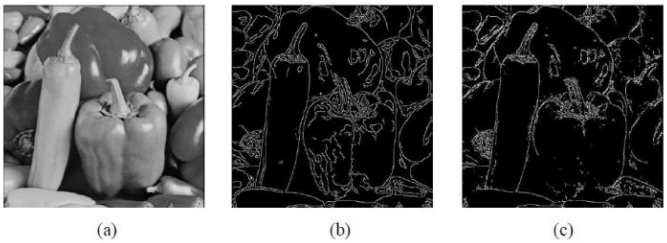

Figure.7 (a) Input pepper image; Final edge map of (b) Existing Canny edge detection algorithm (c) Proposed distributed Canny edge detection algorithm.

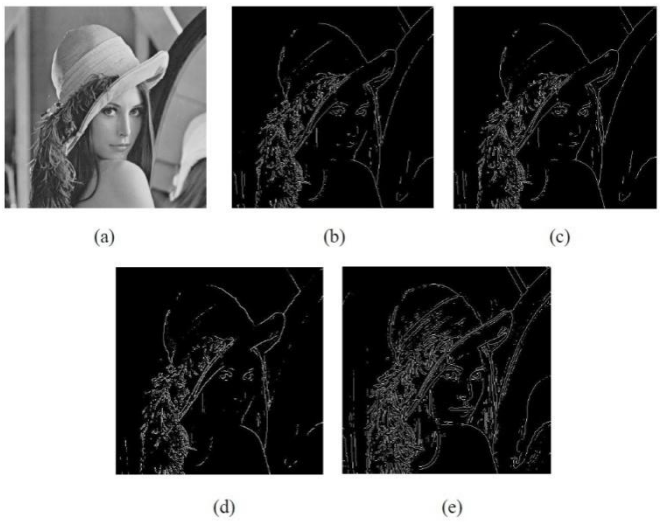

Figure.8 (a) Input Lena image; Final edge map using (b) Sobel (c) Prewitt (d) Roberts Cross (e) LoG edge operators.

VI. CONCLUSION

The original Canny edge detection algorithm computes high and low thresholds based on the frame-level statistics. In this paper, a novel distributed Canny edge detection algorithm is designed in MATLAB. The proposed algorithm calculates high and low threshold values for each block in an image. To support this, block classification and adaptive threshold selection schemes are proposed. Block classification scheme classify each block in the input image as smooth, texture, edge/texture, medium edge and strong edge blocks. Adaptive threshold selection scheme predicts the low and high threshold values by processing the pixels of an individual block. The proposed algorithm has better edge detection performance than original frame-level Canny edge detection algorithm and many other existing edge detection algorithms.

REFERENCES

[1] Qian Xu, Srenivas Varadarajan, Chaitali Ckakrabarti and Lina J. Karam, "A distributed Canny edge detector: Algorithm and FPGA implementation", IEEE Transactions on Image Processing, Vol.23, No.7, July 2014, PP. 2944-2960.

[2] Shaifali Pande, Vivek Singh Bhadouria and Dibyendu Ghoshal "A study of edge marking scheme of various standard edge detectors", International Journal of Computer Applications,Vol.44, No.9, April 2012.

[3] Qian Xu, Chaitali Chakrabarti and Lina J. Karam, "A distributed Canny edge detector and its implementation on FPGA", Digital Signal Processing Workshop and IEEE Signal Processing Education Workshop, January 2011, PP. 500-505.

[4] Mohammadreza Asghari Oskoei and Huosheng Hu "A survey on edge detection methods", School of Computer Science & Electronic Engineering, University of Essex, U.K., February 2010.

[5] Daggu Venkateshwar Rao, Shruti Patil, Naveen Anne Babu and V Muthukumar, "Implementation and evaluation of image processing algorithms on reconfigurable architecture using C-based hardware descriptive languages", International Journal of Theoretical and Applied Computer Sciences, Vol.1, Number 1,2006, PP. 9-34.

[6] Mitra Basu, "Gaussian based edge-detection methods-A survey", IEEE Trans. on Systems, man and cybernetics-Part C: Applications and Reviews, Vol. 32, No. 3, August 2002, PP. 252-260.

[7] J.K. Su and R. M. Mersereau, "Post-processing for artifact reduction in JPEG-compressed images", IEEE ICASSP, Vol. 4, May 1995, PP. 2363–2366.

[8] J.F. Canny, "A computational approach to edge detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, No. 6, 1986, PP. 679-697.

[9] W.E. Grimson and E. C. Hildreth, "Comments on digital step edges from zero crossings of second directional derivatives'' IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-7, Vol. 1, 1985, PP. 121-129.

[10] D. Marr, E. Hildreth, "Theory of edge detection", Proceedings of the Royal Society of London, Series B, Biological Sciences, Vol. 207, No. 1167, February 1980, PP.187-217.