

# A DISTRIBUTED PSYCHO-VISUALLY MOTIVATED CANNY EDGE DETECTOR

Srenivas Varadarajan<sup>1</sup>, Chaitali Chakrabarti<sup>1</sup>, Lina J. Karam<sup>1</sup> and Judit Martinez Bauza<sup>2</sup>

<sup>1</sup>School of Electrical, Computer, & Energy Engineering, Arizona State University, Tempe, AZ 85287-5706

<sup>2</sup>Qualcomm Incorporation, 5775 Morehouse Drive, San Diego, CA 92121  
svarada2@asu.edu, chaitali@asu.edu, karam@asu.edu, juditm@qualcomm.com

## ABSTRACT

This paper proposes a distributed Canny edge detection algorithm which can be mapped onto multi-core architectures for high throughput applications. In contrast to the conventional Canny edge detection algorithm which makes use of the global image gradient histogram to determine the threshold for edge detection, the proposed algorithm adaptively computes the edge detection threshold based on the local distribution of the gradients in the considered image block. The efficacy of the distributed Canny in detecting psycho-visually important edges is validated using a visual sharpness metric. The proposed distributed Canny edge detection algorithm has the capacity to scale up the throughput adaptively, based on the number of computing engines. The algorithm achieves about 72 times speed up for a 16-core architecture, without any change in performance. Furthermore, the internal memory requirements are significantly reduced especially for smaller block sizes. For instance, if a 512\*512 image is processed in 64\*64 blocks using the proposed scheme, the memory is reduced by a factor of 70 as compared to the original Canny edge detector.

**Index Terms** — Canny Edge detector, Distributed Processing, Internal Memory, Sharpness Metric

## 1. INTRODUCTION

Edge detection is a very important first step in many algorithms used for segmentation, tracking and image/video coding. The Canny edge detector is predominantly used due to its ability to extract significant edges. Unfortunately, the Canny edge detection algorithm is computationally very expensive and cannot be used in real-time implementations. Furthermore, the algorithm operates on the whole image and so the computations on an entire image have to be completed before the next algorithm of an image processing application can be initiated. Also, to support frame-level processing, a large internal memory is required. Some approaches have been suggested to alleviate this problem. In [1], a recursively implementable edge detection algorithm is suggested and optimized using retiming techniques. But its performance is quite poor in images with low SNRs. The approach of [2] combines the derivative and smoothing operations of the Canny algorithm into a single mask to reduce computations. The pipelined implementation suggested in [3] is a block-based approach with a block-size of 2 rows of pixels. It overcomes the dependencies between the blocks by fixing high and low thresholds to a constant value. In both of these approaches [2, 3], gradient thresholds are not adapted to the image characteristics, and their performance is not guaranteed for blurred images and images with low SNRs. Generic parallel approaches, such as the method of [4], which do not alter the original sequential algorithm, are only suboptimal because the inherent dependency between the various blocks is not

removed, and these approaches are thus not amenable to distributed processing. Also the overhead in achieving run-time parallelism might become critical in low powered cost effective systems with low memory.

The original Canny algorithm computes the higher and lower thresholds for edge detection based on the entire image statistics, which prevents the processing of blocks independent of each other. In this paper, a new threshold selection algorithm based on the distribution of pixel gradients in a block of pixels is proposed. This new algorithm enables the Canny algorithm to be implemented at the block level. This results in the following benefits: 1) an increase in the throughput when multiple core architectures are available, 2) a decrease in expensive on-chip memory for a fixed throughput, and 3) the possibility of pipelining the Canny edge-detector with other image processing modules of an application.

In this paper, a novel distributed Canny edge detector is presented, including a novel distributed threshold selection algorithm. In order to assess the performance of the proposed distributed algorithm, the increase in throughput is computed as well as the total required operations. Furthermore, the impact of the size of the gradient mask in detecting psycho-visually significant edges is analyzed by means of a no-reference perceptual-based sharpness metric [6].

The paper is organized as follows. An overview of the frame-based Canny edge detection algorithm is given in Section 2 together with the analysis of the complexity of various blocks. The details of the proposed distributed Canny algorithm is given in Section 3. In Section 4, performance results are presented to illustrate the efficacy of the proposed algorithm to discover perceptually important edges, as well as the resulting overall increase in throughput and the reduction in internal memory requirements. A conclusion is given in Section 5.

## 2. CANNY ALGORITHM COMPLEXITY ANALYSIS

The original Canny algorithm [5] shown in Fig 1, consists of the following steps executed sequentially:

1. Low pass filtering the image with a Gaussian mask
2. Computing horizontal and vertical gradients at each pixel location
3. Computing the gradient magnitude at each pixel location
4. Computing a higher and lower threshold based on the histogram of the gradients of the entire image
5. Suppressing non-maximal strong edges
6. Performing hysteresis thresholding for connected weak edges
7. Applying morphological thinning on the resulting edge Map.

The complexity of each of these modules is quantified in this paper in terms of Operations Per Pixel (OPP). These operations include the operations for for-loop counters, arithmetic, conditional, logical, relational and bitwise operations.

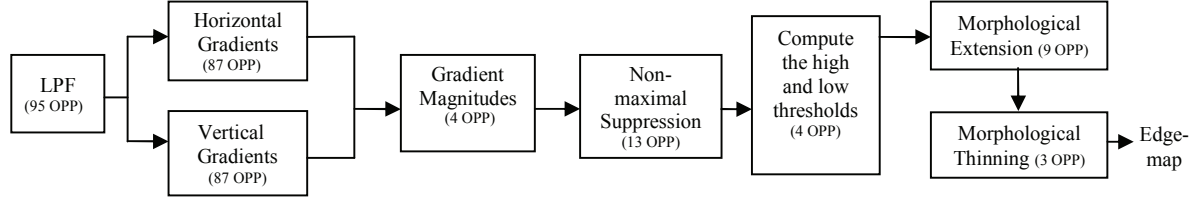


Fig 1. Block diagram of the Canny edge detector, showing the complexity of each block

The low pass filtering is achieved by taking the weighted average of the  $9 \times 9$  neighborhood pixels. By exploiting the redundancy in the mask, this weighted average can be computed using 15 multiplications and 80 additions, resulting in a total of 95 OPP. The horizontal gradient is obtained by multiplying and averaging the neighborhood pixels using another  $9 \times 9$  horizontal gradient mask. Exploiting the redundancies in the gradient mask, the computations can be performed using 68 additions and 19 multiplications per pixel. Similarly, the vertical gradient can be computed by using a  $9 \times 9$  vertical gradient mask and also consumes 87 OPP. The magnitude of the gradient at each pixel is computed by squaring the horizontal and vertical gradients, summing them and taking the square root, which corresponds to 4 OPP. Non-maximal suppression involves computing the gradient direction at each pixel, quantizing the direction to one of 8 possible main directions and comparing with two of its immediate neighbors along the gradient direction and making the gradient zero if it does not correspond to the local maximum. The total arithmetic, relational and logical operations that are required for performing this step is estimated as 7 OPP. For the gradient directions that do not coincide with one of the 8 possible main directions, an interpolation is done to compute the neighboring gradients. These steps together contribute to about 13 OPP. The calculation of the high and low thresholds includes a gradient histogram computation operation and involves quantizing and incrementing the corresponding bin's counter giving 2 OPP. The comparison with the high and the low thresholds account for 2 OPP. The morphological extension involves convolution with a  $3 \times 3$  mask. The morphological thinning involves bitwise 'AND' ing of the extended mask with the modified strong edge map. They together require 12 OPP. Hence, the total complexity of the original Canny is about 302 OPP.

### 3. PROPOSED DISTRIBUTED CANNY ALGORITHM

The superior performance of the frame-based 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. Directly applying the frame-based Canny at a block-level would fail because such a mix of regions may not be available locally in every block of the frame. This would lead to excessive edges in texture regions and loss of significant edges in high-detailed regions.

In the proposed distributed version of the Canny algorithm, the input image is divided into  $m \times m$  overlapping blocks, and the blocks are processed independent of each other. To prevent edge artifacts and loss of edges at the boundaries, adjacent blocks overlap by  $(L-1)/2$  pixels for a  $L \times L$  gradient mask. However, for each block, only edges in the central  $n \times n$  (where  $n = m - L + 1$ ) non-overlapping region are included in the final edge map. In the

proposed algorithm, Steps 1 to 3 and Steps 5 to 7 are the same as in the original Canny algorithm except that these are now applied at the block level. The high and low gradient threshold selection step of the original Canny (Step 4) is modified to enable block-level processing. Analysis of natural images showed that a pixel with a gradient magnitude of 4 corresponds to a psycho-visually significant edge. Also, a pixel with a gradient magnitude of 2 and 6 corresponds to blurred edges and very sharp edges, respectively. The proposed threshold selection algorithm was designed based on these observations and is summarized in Fig. 2.

Let  $G_t$  be the set of pixels with gradient magnitudes greater than a threshold  $t$ , and let  $\aleph_{G_t}$  denote the number of elements in  $G_t$ . For a considered  $m \times m$  block, the proposed threshold selection algorithm starts by computing the sets  $G_t$ , for  $t = \{2, 4, 6\}$ , and their corresponding  $\aleph_{G_t}$ . Using  $\aleph_{G_t}$ , an intermediate classification threshold  $C$  is computed as shown in Fig. 2, and is used to indicate whether the considered block is high-detailed, moderately-edged, blurred, or textured. Consequently, the set  $G_C = G_{t=C}$  is selected for computing the high and low thresholds. The high threshold is computed such that a percentage  $x$  of the total pixels in the selected set  $G_C$  would be classified as strong edges. The lower threshold is computed as a percentage  $y$  of the higher threshold as in the original canny algorithm. In our implementation,  $x$  and  $y$  were set as 20% and 40% respectively.

In order to reduce the computations without sacrificing performance, some changes are made to the original Canny algorithm. The low pass filtering is done with a  $3 \times 3$  mask, (M1 in Fig 3) instead of a  $9 \times 9$  mask. This helps in not blurring the faint edges, in addition to reducing the computations of this block from 95 OPP to only 11 OPP. The size of masks for horizontal and vertical gradients can be varied from  $3 \times 3$  to  $9 \times 9$ , depending on the sharpness of the image. It is shown in the results section (Section 4) that a  $3 \times 3$  gradient mask (M2 in Fig. 3) is sufficient for

```

Step 1: Determine  $G_t$  for  $t = 2, 4, 6$  and  $\aleph_{G_t}$  for  $t = 2, 4$ .

Step 2: If ( $\aleph_{G_4} > 0.25 * \text{Total\_block\_pels}$ )
     $C = 6$  /* High-detailed */
Else if ( $\aleph_{G_4} > 0.05 * \text{Total\_block\_pels}$ )
     $C = 4$  /* moderately-edged */
Else if ( $\aleph_{G_2} < 0.25 * \text{Total\_block\_pels}$ )
     $C = 2$  /* blurred */
Else
    Exit; /* textured */

Step 3: Compute the normalized histogram of  $G_{t=C}$  and the corresponding cumulative distribution function  $F(G_{t=C})$ .

Step 4: Compute High_threshold as  $F(\text{High\_threshold}) = 0.8$ 

Step 5: Compute Low_threshold =  $0.4 * \text{High\_threshold}$ 

```

Fig. 2: Pseudo-code for the proposed distributed threshold selection scheme.

$p^2$	$pq$	$p^2$
$pq$	$q^2$	$pq$
$p^2$	$pq$	$p^2$

M1

$u$	$0$	$-u$
$v$	$0$	$-v$
$u$	$0$	$-u$

M2

Fig. 3: Masks for Low Pass Filtering (M1) and Horizontal Gradient (M2), with  $p = 0.0437$ ;  $q = 0.9947$ ;  $u = 0.0038$ ;  $v = 0.0874$ .

detecting the significant edges and, compared to the original 9x9 gradient mask, the 3x3 mask reduces the computations from 87 OPP to only 7 OPP. The new distributed threshold selection calls for 2 additional gradient histograms per block and 2 additional comparisons per pixel. As a result, the total computations for the proposed distributed Canny algorithm, varies from 50 OPP (with a 3x3 gradient mask) to 130 OPP (with a 9x9 gradient mask) depending on the size of the gradient mask.

#### 4. SIMULATION RESULTS

The efficacy of the proposed algorithm and architecture is evaluated in terms of 3 parameters, namely edge detection performance (Section 4.1), increase in throughput (Section 4.2) and reduction in memory requirements (Section 4.3). We show that the proposed scheme results in high throughput as well as significant savings in memory as compared to the original non distributed Canny edge detector.

##### 4.1 Edge Detection Performance Analysis

The edge detection performance of the proposed distributed approach is analyzed by comparing the perceptual significance of its resulting edge map with the one produced by the original frame-based Canny edge detector. Figs. 4(a) and 4(b) show, respectively, the edge maps that are obtained for the 512x512 Man image using the original Canny edge detector and the proposed distributed Canny edge detector with a 3x3 gradient mask and block size of 64. Figs. 5(a) and 5(b) show the edge maps that are obtained for the Gaussian blurred ( $\sigma = 3$ ) 512x512 Man image using the proposed distributed Canny edge detector with a 3x3 and 9x9 gradient masks, respectively. As shown in Figs. 5(a) and 5(b), the 3x3 gradient mask is insufficient for detecting edges in highly blurred images and, hence, a larger mask like 9x9 needs to be used. Even though there are some differences between the obtained edge maps, the proposed method detects all the psycho-visually important edges. To illustrate this, the perceptual visual quality of the obtained edge maps is assessed using the psycho-visual sharpness metric of [6]. This metric primarily depends on the edge map to compute the amount of blurriness/sharpness in an image. It measures the spread of the edge pixels in an image and normalizes it with a ‘Just Noticeable Blur’ (JNB) that depends on the local contrast in order to compute the perceived sharpness. As in [6], four images (Man, Houses, Peppers and Flowers) with varying image characteristics were chosen, and were blurred using Gaussian masks with six different  $\sigma$  values (in range of 0.5 to 3.5) to generate a set of 24 images. The images were displayed one after another in a random order and subjects were asked to rate them on a scale of 1 to 5 to form a Mean Opinion Score. The Pearson’s correlation coefficient (PCC) was used to measure how well the sharpness metric commensurate with the actual perceived sharpness. Fig. 6 depicts the effect of replacing the frame-based edge detection with the distributed edge detection, on the sharpness metric of [6], for gradient masks of different sizes. Note that a edge detector block size of 512 corresponds to the original frame-based

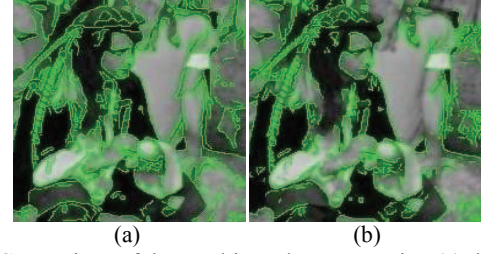


Fig. 4: Comparison of the resulting edge-maps using (a) the frame-based Canny edge detector, and (b) the proposed distributed Canny edge detector with a block-size of 64, for the 512x512 Man image without blur.

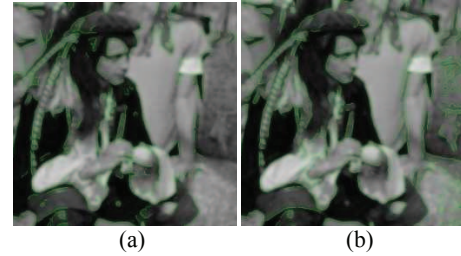


Fig. 5: Comparison of the resulting edge-maps using the proposed distributed Canny edge detector with a block-size of 64 and with (a) a 3x3 gradient mask and (b) a 9x9 gradient mask, for the 512x512 Man image blurred using a Gaussian with a standard deviation  $\sigma = 3$ .

Canny. It can be seen that, for a block size greater or equal to 32, the decrease in the PCC is insignificant. This demonstrates the ability of the distributed approach to discover edges even in blurred images. As shown in Fig 6, the PCC increases with an increase in the size of the gradient mask at the cost of extra computations.

##### 4.2 Increase in Throughput

The distributed Canny algorithm has a small computational overhead due to the fact that an  $m \times m$  overlapping block has to be processed to generate the edges of a  $n \times n$  block, where  $m = n + L - 1$ . The processing of extra pixels per block results in corresponding extra operations as given below:

$$\text{Extra ops per block} = (m^2 - n^2) * \text{OPP} \quad (1)$$

The extra control operations per block, arising due to the new threshold selection algorithm are computed as follows:

$$\text{Extra control ops per block} = 4 * m^2 \quad (2)$$

The total number of operations for processing the non-overlapping pixels of a frame is given by:

$$\text{Total non-overlapped ops per frame} = N^2 * \text{OPP} \quad (3)$$

Results from (1), (2) and (3) are combined to form *Total ops per frame*. In spite of this overhead, since the computational engines are operating in parallel, the amount of time taken for processing an entire frame equals the time taken by a single computational unit to complete processing a  $m \times m$  block. Also note that the OPP in (1) and (3) varies from 57 OPP to 217 OPP depending on the size of the gradient mask, rather than 302 OPP in the original Canny algorithm. With a 16-core architecture (128 x128 block size for a 512 x512 image) and the computing engines clocked at 300 MHz and executing 1 instruction per cycle on average, it would

Table 1	
Year	Value
2010	100
2011	105
2012	110
2013	115
2014	120
2015	125
2016	130
2017	135
2018	140
2019	145
2020	150