JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 36, 535-546 (2020)
DOI: 10.6688/JISE.202005_36(3).0004

High-Efficient Low-Cost VLSI Implementation for Canny Edge Detection

DA-HUEI LEE[1], PEI-YIN CHEN[2], FU-JHONG YANG[2] AND WAN-TING WENG[2]
[1]*Department of Electronic Engineering*
Southern Taiwan University of Science and Technology
Tainan, 71005 Taiwan
E-mail: dhlee@stust.edu.tw
[2]*Digital Integrated Circuit Design Laboratory*
Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, 70101 Taiwan
E-mail: pychen@csie.ncku.edu.tw; kyo0623@hotmail.com; aiimo286@gmail.com

For real-time image processing applications in consumer electronic products, high-speed preprocessing algorithms are necessary and have been widely investigated. This article presents a highly efficient very large scale integrated (VLSI) circuit implementation of Canny edge detection. We employed an approximation method that reduces hardware costs without affecting computation results. Additionally, we divided the whole image into several blocks for processing to obtain superior detection performance. It can efficiently prevent missing the real edge in low-contrast regions. The VLSI architecture of our design yields a processing rate of approximately 250 MHz using the Xilinx Virtex-5 field-programmable gate array. The simulation result shows that the proposed circuit takes 0.14ms for processing 512×512 test image database and requires the least number of operations compared with previous techniques; therefore, it is suitable for low-cost high-performance system on chip systems.

Keywords: Canny edge detector, pipeline architecture, low-cost design, FPGA, VLSI

1. INTRODUCTION

As technology advances, computer vision plays an increasingly central role in our lives. Edge detection is a widely used algorithm in image processing for applications such as image segmentation, feature extraction, and object tracking. It significantly reduces the number of processing steps, excludes irrelevant messages, and retains key information objects. Many edge detector algorithms have been proposed, such as the Roberts detector [1], Kirsch detector [2], Gauss-Laplace detector [3], and Canny detector [4]. Among existing algorithms, the Canny algorithm exhibits a more favorable performance because of its ability to detect edges in images that are contaminated by noise [5]. The advantage of the Canny algorithm is that it processes edge detection through hysteresis thresholding, which computes a high and a low threshold to reduce the edge detection error rate. However, the Canny algorithm engenders higher computational complexity than do other edge detectors and it cannot be employed in real-time applications. Moreover, Deep learning-based methods for edge detection (or contour detection) have better detection performance than the Canny algorithm [6], however the implementation hard-

Received March 5, 2018; revised October 10, 2018 & January 8 & August 1, 2019; accepted October 24, 2019.
Communicated by Jing-Ming Guo.

ware required by these methods are far more than the Canny algorithm. For real-time application, the Canny algorithm may be a better choice than deep learning based methods.

The Canny circuit may be applied in end-user camera equipment or equipped in medical imaging systems; hence, how to implement it with a lower hardware cost is a vital concern. For this reason, previous studies [7-11] concerning the very large scale integration (VLSI) architecture of Canny algorithms have been conducted. A Field-Programmable Gate Array (FPGA) implementation of an edge detection algorithm using handle-C by Rao and Venkatesan [7] applies system-level hardware design tools to translate the software design directly into Very high-speed integrated circuit Hardware Description Language (VHDL) or Verilog and increase the time performance. In [8], a parallel design of a real-time Canny implementation is presented. The design benefits from 4-pixel parallel computations to achieve high throughput with the cost of an increased number of memory accesses. In [9], He and Yuan proposed a self-adaptive threshold Canny edge detection algorithm, whereby low and high threshold values are computed for each input image. A multiresolution FPGA-based architecture which also uses adaptive thresholds was proposed by Possa *et al.* [10]. The key component in this architecture is the neighborhood extractor that can be parameterized 'on-the-fly' on the basis of the image characteristics. Some simplifications in the algorithms that reduce mathematical complexity, latency, and memory requirements are also presented in this paper. A distributed Canny edge detector proposed by Xu *et al.* [11] computes the edges of multiple blocks at the same time. To support this, an adaptive threshold selection method is proposed that predicts the high and low thresholds of the entire image while only processing the pixels of an individual block.

To achieve lower costs and higher efficiency, we used approximation methods to replace the complex operations and adopted a pipelined architecture to implement a Canny edge detector. Our proposed design can be implemented at a low cost and can deliver high throughput. The experimental results indicate that the proposed mechanism exhibits a superior performance compared with existing mechanisms in terms of subjective testing.

The rest of this paper is organized as follows: Section 2 briefly introduces the original Canny edge detector algorithm. The proposed VLSI implementation of Canny edge detection is introduced in Section 3. Section 4 presents the simulation results. The conclusion is provided in Section 5.

2. ORIGINAL CANNY EDGE DETECTION ALGORITHM

This section outlines the original Canny edge detection algorithm. The original Canny edge detection algorithm's process includes five steps, namely (1) Apply a Gaussian filter to smooth the image to remove noise; (2) For each pixel located, calculate the horizontal gradient value (G_x) and the vertical gradient value (G_y); (3) Calculate the magnitude (M_s) using the Euclidean distance formula, and the direction ($\theta_{i,j}$) using the arctangent function (inverse trigonometric function); (4) After receiving the gradient direction of the central pixel, use Non-Maximum Suppression (NMS) to 'thin' the edge, which compares the central pixel magnitude with its two neighbors along the gradient direction;

the gradient magnitude is set to zero if it does not correspond to a local maximum. Canny classifies the gradient direction into eight main direction masks at degrees 0°, 45°, 90°, 135°, 180°, 225°, 270°, and 315°. This method not only thins the edge to achieve higher object recognition but also filters some noise to reduce false edges. (5) Perform hysteresis thresholding to determine the edge map. If the pixel magnitude is larger than the high threshold, the pixel is labeled as a strong edge pixel and is instantly considered as a definite edge. If the pixel magnitude is lower than is the low threshold, the pixel is labeled as a nonedge pixel, and the value of the pixel is set to zero. If the gradient magnitude of a pixel is between the low threshold and high threshold, the pixel is labeled as a weak edge. The weak edge pixel is classified as an edge pixel if and only if it is connected to a strong edge pixel.

Among the existing implementation mechanisms, they require extensive and complex computations, such as huge matrix multiplication and division, square roots, and exponents. Thus, we propose a more efficient and low-complexity implementation mechanism that can obtain favorable results and satisfy the requirements of real-time applications.

3. PROPOSED MECHANISM & VLSI ARCHITECTURE

The flow diagram of our efficient implementation mechanism for the Canny edge detection algorithm is shown in Fig. 1. The operating procedure of the proposed mechanism comprises four principal parts: block division, optimization of Canny operation, calculation of thresholds, and hysteresis. First, a block division technique is used for

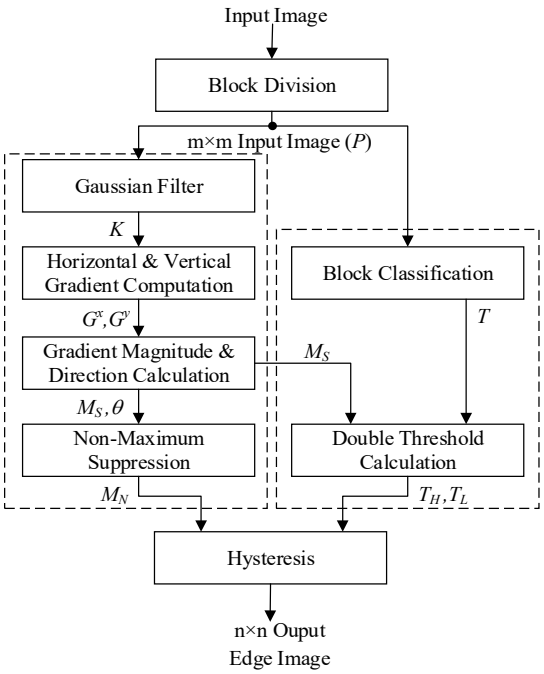


Fig. 1. Flow diagram of the Canny mechanism.

improving edge detection performance. Second, we employ a low-cost implementation for optimization of Canny operation to decrease computational complexity. Third, thresholds are calculated by block classification type and gradient magnitude. Therefore, we can detect detail edges in the low-contrast region. Finally, hysteresis thresholding is performed to determine the edge image. For some real-time image and video applications in consumer electronic products, fast execution time is required. Thus, we employ pipeline scheduling to implement the flowchart (see Fig. 1). We have developed an 11-stage pipelined hardware architecture as shown in Fig. 2. The architecture is divided into eight units: the line buffer and register bank units, Gaussian filter unit, Sobel operator unit (horizontal and vertical gradient computation, and gradient magnitude and direction calculation), Non-Maximum Suppression unit (NMS), block classification unit, double thresholds calculator unit (DTC), hysteresis unit, and controller unit. The details of each process are described as follows.

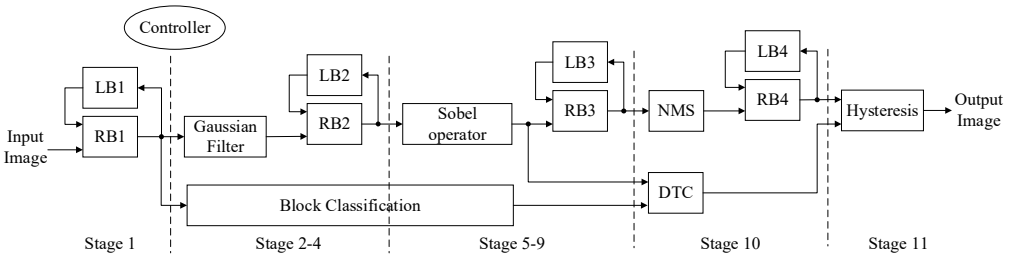


Fig. 2. Block diagram of hardware architecture for our Canny mechanism.

3.1 Block Division

For more accurate edge detection performance, a block division module is employed. It divided the input image into $m \times m$ overlapping blocks. To handle these blocks independently of each other to prevent image block artifacts, we should divide the input image into $n(64) \times n(64)$ non-overlapping blocks and then enlarge each block by three pixels on the left, right, top, and bottom sides. Generally, a larger mask size will have a better result. However, it will increase the hardware cost. Hence, we choose 3×3 mask size in our mechanism. For instance, an image of size 512×512 is divided into 8×8 blocks, where each block has a size of 64×64. With a 3×3-sized Gaussian mask, a 3×3-sized gradient mask, and a 3×3-sized NMS mask, each block has a size of $m(64+2+2+2+2) \times m(64+2+2+2+2)$.

3.2 Optimize Canny Operation

As discussed in Section 2, the original Canny edge detection algorithm [4] requires complex computation, and requires a long execution time. To mitigate the disadvantage of the algorithm presented in [4], we proposed a low-cost implementation mechanism of the Canny method, which is suitable for VLSI implementation. The mechanism contains four steps: Gaussian filter, horizontal and vertical gradient computation, gradient magnitude and direction calculation, and non-maximum suppression. The details of each step are described as follows:

3.2.1 Gaussian filter

As a consequence of edge detection being easily affected by noise, finding the optimal edges in an untreated image proves difficult. Hence, a preprocessing step before the edge detection of the image is required, the Gaussian filter preserves more real edges than other uniform blurring filters. Canny uses a 2-D convolution operator to blur the image and remove the noises. The Gaussian distribution in 2-D has the form:

$$g(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{1}$$

Where x is the distance from the horizontal axis, y is the distance from the vertical axis, and σ is the standard deviation of Gaussian distribution. Hence, it adopts a 3×3 mask and the value of σ set as 3. To implement actual weighted value, it is replaced by an approximation of the weighted value, as shown in Table 1. They are very similar to each other in the experimental results.

Table 1. Approximate Gaussian weighted value.

Actual Weighted Value	Approximate Weighted Value
0.106997	$2^{-4}+2^{-5}+2^{-6}$ (0.109375)
0.11311	$2^{-4}+2^{-5}+2^{-6}+2^{-8}$ (0.113281)
0.119572	$2^{-4}+2^{-5}+2^{-6}+2^{-7}$ (0.117187)

3.2.2 Horizontal & vertical gradient computation

As mentioned in [3], the Sobel operator detects edges more accurately than do the Prewitt and Roberts operators. Therefore, we use the Sobel operator to calculate the horizontal and vertical gradient magnitude. Subsequently, this information is used in next step to calculate gradient magnitude and direction.

3.2.3 Gradient magnitude & direction calculation

Based on the Euclidean distance formula, two square operations and one square root operation are required to calculate the magnitude of one target pixel. In hardware implementation, the cost for the calculation of the square and square root is high. Therefore, we utilize the square root approximation (SRA) technique suggested in [12] to decrease the hardware cost. The equation for gradient magnitude is expressed as follows:

$$M_S=\sqrt{G_{i,j}^{x\ 2}+G_{i,j}^{y\ 2}}\cong\max((0.875a+0.5b),a), \tag{2}$$

$$a=\max(|G_{i,j}^{x}|,|G_{i,j}^{y}|), \tag{3}$$

$$b=\min(|G_{i,j}^{x}|,|G_{i,j}^{y}|). \tag{4}$$

As introduced in Section 2, the horizontal and vertical values are used to calculate the gradient direction (θ). This step uses an arctangent function which distinguishes the gradient direction masks. The required hardware cost is high, so a coordinate rotation digital computer (CORDIC) module [13] was considered. Such modules are widely used for

calculating various trigonometric and transcendental functions by rotating vectors. However, the main drawback to the CORDIC module is that many iterations are needed to obtain acceptable calculation precision. To achieve low computational complexity, we eliminate the calculation of the arctangent function by using the definition of the trigonometric function. The equations are expressed as follows [14]:

$$G_{i,j}^{x} \times \tan\theta_i \le G_{i,j}^{y} < G_{i,j}^{x} \times \tan\theta_{i+1}. \tag{5}$$

Because we only need to determine the direction bins instead of calculating the real value of the arctangent, the θ can be classified into one of the four bins as shown in Fig. 3 by comparing the values of tangent of the degrees of 22.5°, 67.5°, 112.5°, and 157.5°. The $G_{i,j}^{x}$ located at the second quadrant can be mapped into the first quadrant by $G_{i,j}^{x} = -G_{i,j}^{x}$. A signal 'sign bit' is used to further reduce the size of the lookup table. If the sign bits of the two values are identical, we can determine that the gradient direction is within the first quadrant; otherwise, the gradient direction is within the second quadrant. Furthermore, we use the shift-and-add function to replace the multiply function. The shift-and-add coefficient values of tan 22.5° and tan 67.5° are listed in Table 2. Fig. 4 is the implementation of $G_{i,j}^{x} \times \tan 22.5°$, and Fig. 5 is the architecture used to consider bins 0 to 3 and to determine in which bin θ actually belongs.

Table 2. Approximate value of tanθ.

Tangent	Approximate value
tan22.5°	$2^{-2} + 2^{-3} + 2^{-5} + 2^{-7}$
tan67.5°	$2 + 2^{-2} + 2^{-3} + 2^{-5} + 2^{-7}$

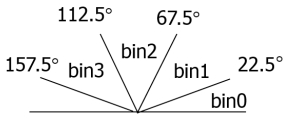


Fig. 3. Bins of gradient direction.

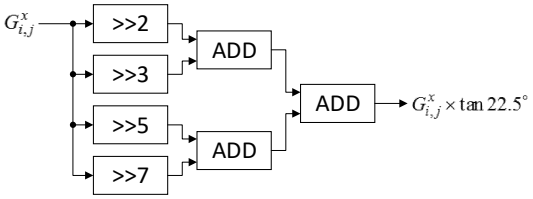


Fig. 4. Architecture of $G_{i,j}^{x} \times \tan 22.5°$.

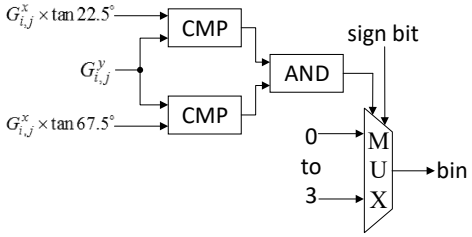


Fig. 5. Architecture of $G_{i,j}^{x} \times \tan 22.5° \le G_{i,j}^{x} \times \tan 67.5°$.

3.3 Calculate Thresholds

To enhance the performance of the proposed implementation mechanism of the Canny method, we divided the whole image into several blocks and classified each block into three types. The high and low thresholds of each block were then determined according to the type and gradient magnitude of the block. These steps are described briefly in the following sections.

3.3.1 Block classification

On the basis of [15], we classify the block by its data standard deviation. Because the data standard deviation is a measurement that is calculated from the amount of variation within a set of data values, the more concentrated the values are, the smaller is the standard deviation. The type of each block is determined by its maximum data standard deviation, calculated from all 3×3 masks. It means to find a maximum value in the 64 standard deviations. To decrease hardware costs, we use variance instead of standard deviation and replace nine (3×3) masks with eight. Afterwards, the blocks are classified into three types in the basis of the value of their maximum variance (*max_v*): Smooth, Texture, and Edge. According to our experimental results, we set coefficients as 1000, and 4100, and the type value *T* is used to represent the block type, as shown in Table 3.

Table 3. Block type classification.

Block type	Type value (*T*)	Maximum Variance (*max_v*)
Smooth	0	$max\_v \le 1000$
Texture	1	$1000 < max\_v \le 4100$
Edge	2	$max\_v > 4100$

3.3.2 Double thresholds calculation

This step takes advantage of the type and the gradient magnitude at the block to compute the high threshold. After a series of experiments, we know that if the maximum gradient magnitude is lower than a threshold T_m, the block has implied excessive noise or no edges. Therefore, if the maximum gradient magnitude is smaller than T_m, the high threshold is set to be the maximum gradient magnitude. Hence, the value of T_m set as 70. Otherwise, if the maximum gradient magnitude is greater than a threshold T_m, the high threshold is set to be the value of the maximum gradient magnitude multiplied by a proportion value (P_T) according to the block type. The formula is shown as

$$T_H = \begin{cases} max\_mag, & \text{if } max\_mag < T_m \\ max\_mag \times P_T, & \text{otherwise} \end{cases}. \tag{6}$$

Where $P_T \in \{0.5, 0.25, 0.125\}$. After calculating the high threshold, we set the low threshold value to be half of the high threshold value in the basis of Section 2.

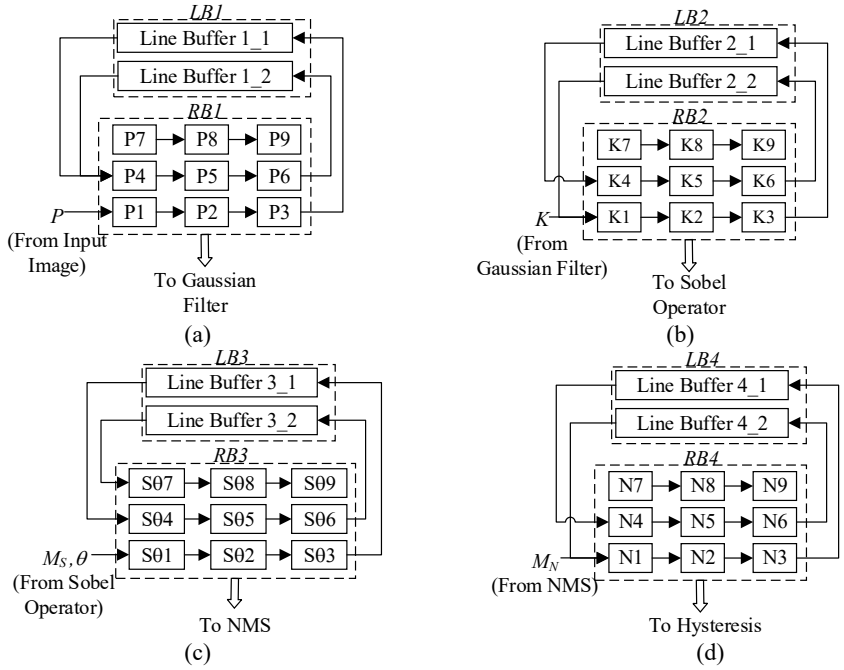


Fig. 6. Architecture of line buffers and the register banks: (a) for Gaussian filter; (b) for Sobel operator; (c) for NMS; and (d) for Hysteresis.

3.4 Line Buffer & Register Bank

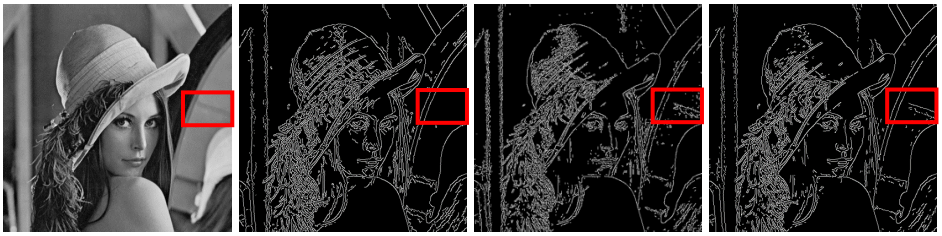
We realize a module with two line buffers and a register bank to concurrently produce nine values of a 3 × 3 window (Fig. 2). The line buffer outputs one pixel to the register bank and replaces the output pixel with a new pixel from the register bank at every cycle. The register bank is designed to feature a shift register style that saves the nine pixels of a 3 × 3 window and can output these pixels concurrently at every cycle. Then, the nine pixels are used at the next stage for the Gaussian filter unit, and block classification unit. Similarly, the three other line buffer and register bank modules are applied to store the previous Gaussian filter, Sobel operator, and NMS results, respectively, as shown in Fig. 6.

4. IMPLEMENTATION RESULTS AND COMPARISONS

To verify the performance of the Canny edge detection algorithm implemented using various mechanisms, several simulations were conducted on the USC SIPI Image Database [16] and the Standard Test Image Database [17].

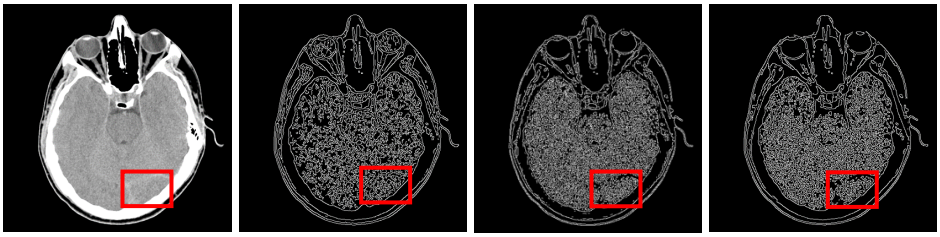
Figs. 7-8 show the simulation results compared with Canny's work [4], Xu's method [11], and our work, respectively. The simulation result of the proposed mechanism preserves more details than [4] in the cropped regions, and is similar to [11].

In order to explore the similarity of two-edge maps, three quality metrics including P_{co} (percentage of edge pixels detected by both implementations), P_{nd} (percentage of edge pixels detected by the original Canny edge detection algorithm that were not de-



(a) Original image. (b) Canny's result. (c) Xu's result. (d) Our result.

Fig. 7. Results of various Canny mechanisms and ours.



(a) Original image. (b) Canny's result. (c) Xu's result. (d) Our result.

Fig. 8. Results of various Canny mechanisms and ours.

tected by the proposed low-cost Canny edge detection algorithm also referred as false negatives), and P_{fa} (percentage of edge pixels detected by the proposed low-cost Canny edge detection algorithm that were not detected by the original Canny edge detection algorithm, also referred as false positives), as proposed in [18]. For all database, the detected edge pixel percentage P_{co}, P_{nd}, and P_{fa} were 92.81%, 3.51% and 3.67%, respectively. Obviously, the proposed low-cost algorithm detects almost all edges that are detected by the original Canny edge detection algorithm.

The hardware architecture of our design is implemented using Verilog HDL, verified using ModelSim, and synthesised using the Xilinx ISE tool, where the FPGA family is the Virtex-5 (XC5VSX240T) [19]. The Canny results of the proposed method with a software programme and VLSI circuit were identical (Fig. 9). It operated at 4 ns and achieved a processing rate of approximately 250 megapixels per second (MP/s) which is quick enough to process a video resolution of Full HD (1920×1080) at 60 fps in real time.

To explore the quality advantages of the proposed Canny mechanism, it was compared with the four Canny chips [7-9, 11]. Table 4 lists the detailed comparisons for various Canny mechanisms. The normalization time is done with respect to an image of size 512×512 and an FPGA clock frequency of 100 MHz. Although the proposed method required more slices than [8], it yielded a much faster computation time. Moreover, it has less cost than [11], and decreases by 2752 slices, 128 slice registers, 20960 slice LUT, and 299KB memory.

To show the actual computational load, we also compared the computation complexity of [11] in terms of actual number of adder, subtractor, shifter, multiplier, divider, multiplexer, comparator, square root, and exponential, as shown in Table 5. The number of operations for each pixel in the original image is specified. The result indicates that our method requires less computational load; therefore, it can be used in many real-time applications.

(a) (b) (c)

Fig. 9. Proposed Canny results with software programme and VLSI circuit; (a) Original "image01"; (b) "image01" by software; and (c) "image01" by circuit.

Table 4. Comparison of the proposed circuit and previous hardware implementation.

	[7]	[8]	[9]	[11]	Proposed
Image Size	256×256	512×512	360×280	512×512	512×512
FPGA Device	Xilinx Vertex-E	Xilinx Virtex-5	Altera Cyclone	Xilinx Virtex-5	Xilinx Virtex-5
Slices	□	4553/71680	□	23904/37440	21152/37440
Slice Registers	□	□	□	40640	**40512**
Slice LUTS	□	□	□	82496	**61536**
DSP48Es	□	□	□	224	224
Used Memory(KB)	□	192/5328	□	16184/18576	**15885/18576**
Frequency(MHZ)	16	292.8	27	250	250
Time(ms)	4.2	0.57	2.5	0.15	**0.14**
Norm. Time(ms)	2.688	1.669	0.72	0.372	**0.35**

Table 5. Comparison in terms of actual number of operations per pixel for Xu's method.

	[11]	**Proposed**
Adder	65	**59**
Substractor	16	**15**
Shifter	12	42
Multiplier	39	**9**
Divider	5	**0**
Multiplexer	4	9
Comparator	47	**37**
Square-root	1	**0**
Exponential	2	**0**

5. CONCLUSIONS

In this paper, a low-complexity pipelined implementation mechanism for Canny edge detection was proposed. Using approximation methods to replace complex operations, our mechanism efficiently reduces hardware costs. The extensive experimental results show that our mechanism detects low-contrast edges. Additionally, the cost of the proposed design is more affordable than previous methods. Therefore, our low-cost circuit is a promising solution for low-cost VLSI implementation for consumer electronic products and can be integrated with other hardware modules in system on chip devices for real-time image applications.

REFERENCES

1. L. G. Roberts, "Machine perception of three-dimensional solids," Ph.D. dissertation, Department of Electrical Engineering, Massachusetts Institute of Technology, 1963.
2. R. A. Kirsch, "Computer determination of the constituent structure of biological images," *Computers and Biomedical Research*, Vol. 4, 1971, pp. 315-328.
3. A. Rosenfeld, "Picture processing by computer," *ACM Computing Surveys*, Vol. 1, 1969, pp. 147-176.
4. J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, 1986, pp. 679-698.
5. Z. Hocenski, S. Vasilic, and V. Hocenski, "Improved canny edge detector in ceramic tiles defect detection," in *Proceedings of IEEE 32nd Annual Conference on Industrial Electronics*, 2006, pp. 3328-3331.
6. Y. Liao, S. Fu, X. Lu, C. Zhang, and Z. Tang, "Deep-learning-based object-level contour detection with CCG and CRF optimization," in *Proceedings of IEEE International Conference on Multimedia and Expo*, 2017, pp. 859-864.
7. D. V. Rao and M. Venkatesan, "An efficient reconfigurable architecture and implementation of edge detection algorithm using handle-c," in *Proceedings of IEEE International Conference on Information Technology: Coding and Computing*, 2004, pp. 843-847.
8. C. Gentsos, C.-L. Sotiropoulou, S. Nikolaidis, and N. Vassiliadis, "Realtime Canny edge detection parallel implementation for fpgas," in *Proceedings of IEEE 17th International Conference on Electronics Circuits and Systems*, 2010, pp. 499-502.
9. W. He and K. Yuan, "An improved canny edge detector and its realization on fpga," in *Proceedings of IEEE 7th World Congress on Intelligent Control and Automation*, 2008, pp. 6561-6564.
10. P. R. Possa, S. A. Mahmoudi, N. Harb, C. Valderrama, and P. Manneback, A multi-resolution fpga-based architecture for real-time edge and corner detection," *IEEE Transactions on Computers*, Vol. 63, 2014, pp. 2376-2388.
11. Q. Xu, S. Varadarajan, C. Chakrabarti, and L. J. Karam, "A distributed canny edge detector: algorithm and fpga implementation," *IEEE Transactions on Image Process*, Vol. 23, 2014, pp. 2944-2960.
12. D. D. Gajski, *Principles of Digital Design*, Upper Saddle River, NJ, Prentice-Hall, 1997.
13. H. T. Ngo and V. K. Asari, "A pipelined architecture for real-time correction of barrel distortion in wide-angle camera images," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 15, 2005, pp. 436-444.
14. P. Y. Chen, C. C. Huang, C. Y. Lien, and Y. H. Tsai, "An efficient hardware implementation of HOG feature extraction for human detection," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 15, 2014, pp. 656-662.
15. H.-S. Kong, Y. Nie, A. Vetro, and H. Sun, "System and method for classifying pixels in images," Patent, 2008, US Patent 7,346,224.
16. USC SIPI Image Dataset, http://sipi.usc.edu/database/database.php?volume=misc/, 2013.
17. Standard Test Image Dataset, http://www.imageprocessingplace.com/, 2013.
18. L. H. A. Lourenco, "Efficient implementation of canny edge detection filter for ITK using CUDA," in *Proceedings of the 13th Symposium on Computer Sys-*

tems, 2012, pp. 33-40.
19. Xilinx vertrix-5 family datasheet, http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf, 2015.

Da-Huei Lee (李大輝) received the B.S., M.S. and Ph.D. degrees in Electrical Engineering from National Cheng Kung University, in 1999, 2001 and 2008, respectively. He then joined the National Chip Implementation Center (CIC), which is affiliated with the National Applied Research Laboratories, Taiwan. In CIC, he worked on the development of readout circuits for MEMS-sensor and bio-sensor. In 2012, he was recruited by the Department of Electronic Engineering, Southern Taiwan University of Science and Technology, as an Assistant Professor. He is currently an Associate Professor and serves as the Director of Microcontroller Design and Application Technique Research Center. His research interests involved VLSI circuits, mixed-signal integrated circuits, bio-sensors, MEMS, and systems for green energy.

Pei-Yin Chen (陳培殷) received the B.S. degree from National Cheng Kung University, Tainan, Taiwan, in 1986, the M.S. degree from Pennsylvania State University, University Park, in 1990, and the Ph.D. degree from National Cheng Kung University, in 1999, all in Electrical Engineering. He is currently a Professor with the Department of Computer Science and Information. His research interests include very large-scale integration chip design, video compression, fuzzy logic control, and gray prediction.

Fu-Jhong Yang (楊富仲) received the B.S. degree in Electronic Communication Engineering from National Kaohsiung Marine University, Kaohsiung, Taiwan in 2011, the M.S. degree in Electronic Engineering from National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan in 2013, and the Ph.D. degree from National Cheng Kung University, in 2008. He is now with Raydium Inc., Tainan, Taiwan. His research interests include image processing, VLSI chip design, and data compression.

Wan-Ting Weng (翁婉婷) received the B.S. degree in Computer Science and Information Engineering from Yuan Ze University, Taoyuan, Taiwan in 2014, and the M.S. degree in Computer Science and Information Engineering from National Cheng Kung University, Tainan, Taiwan in 2016. She is now with AndesTech Inc., Hsinchu, Taiwan. Her research interests include very large-scale integration chip design, and embedded systems.