IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Traitement du Signal
Vol. 39, No. 4, August, 2022, pp. 1421-1427
Journal homepage: http://iieta.org/journals/ts

A Novel Sobel Edge Detection Accelerator Based on Reconfigurable Architecture

Li Xu*, Dechun Zheng

School of Electronic and Information Engineering, Ningbo University of Technology, Ningbo 315211, China

Corresponding Author Email: xuli@nbut.edu.cn

https://doi.org/10.18280/ts.390436

Received: 5 May 2022
Accepted: 25 July 2022

Keywords:
reconfigurable architecture, Sobel edge detection, accelerator

ABSTRACT

A novel Sobel edge detection accelerator based on reconfigurable architecture is proposed to solve the problem of low power-to-performance ratio of traditional Sobel edge detection algorithm in CPU processing. The accelerator adopts pixel level fine grain image data parallel processing and row buffer storage architecture to improve the processing efficiency of edge detection. At the same time, a reconfigurable architecture based on FPGA is built. Through experiments, it can be found that the acceleration effect of the edge detection accelerator on video data is superior to that of the CPU software. Compared with similar accelerators, the acceleration performance of the novel accelerators improves by 10%. The results show that the proposed edge detection accelerator can be used in embedded systems to provide edge detection processing capability with high performance power consumption ratio.

1. INTRODUCTION

The edge is the collection of all pixels in the image area where the gray level changes suddenly, containing a large number of effective parameters of the image. Image edge detection is usually used as a preprocessing step for image analysis and understanding, and the purpose of image segmentation is achieved by extracting the boundary lines of different regions. In the fields of image feature extraction, target recognition and tracking, image edge detection plays an important role. Edge detection methods can be divided into spatial domain detection and transform domain detection. The commonly used Robert, Prewit, LOG, Canny, Sobel and other algorithms all belong to spatial detection [1]. Among them, Robert and Prewit have low edge positioning accuracy. LOG operators cannot identify the direction of edges and are sensitive to noise. Canny operators have superior functions but are complex to implement [2]. It is difficult to use them in real-time hardware systems. Although traditional Sobel algorithms need to manually specify detection thresholds, they have the advantages of simple detection principle and easy hardware implementation.

With the increasing complexity of the algorithm and the amount of data, the reconfigurable system based on FPGA uses a reconfigurable structure to implement the edge detection algorithm [3], which can meet the requirements of big data, high speed and high stability. And the implementation of Sobel edge detection algorithm with reconfigurable system can improve the processing speed [4], so that the operation of Sobel algorithm is no longer the bottleneck of the whole system efficiency, and the algorithm can be applied to the local computing environment with high real-time requirements. In addition, in some applications with high security and confidentiality, such as radar monitoring [5] and remote sensing identification [6], the application of reconfigurable architecture to build special embedded chips can also improve system security and reduce system power consumption.

Therefore, in recent years, a lot of related work has been devoted to the research of edge detection accelerator based on reconfigurable architecture. Menaka et al. [7-10] designed a high-speed and low-power edge detection accelerator. However, during system verification, some images are stored in memory in advance, others are transferred from the upper computer to the system for edge extraction, and some do not use actual images. At the same time, the detection only aims at the image without optimizing the video. In addition, in order to effectively implement edge detection, the algorithm becomes more and more complex, which is a huge challenge to the edge detection accelerator with reconfigurable architecture. Jiang et al. [11-15] put forward sequence stream processor and pipeline processing method to build the edge detection accelerator based on reconfigurable architecture with the purpose of simplifying computing components and improving parallelism, and achieved certain results. However, these systems did not optimize the architecture for video streams, nor did they consider the parallel acceleration of memory in the reconfigurable architecture, but only achieved processing efficiency by improving the performance of computing components.

In the method proposed in this paper, firstly, RGB video frame data is converted into gray scale. Then, according to the characteristics of video frame data and Sobel algorithm, the image is parallelized with pixel level fine granularity. And according to the characteristics of pixel level fine granularity data, the row buffer storage architecture is designed. Finally, Sobel edge detection accelerator based on reconfigurable architecture is designed to accelerate the processing capability of edge detection.

2. SOBEL ALGORITHM

The common method for edge detection is image gradient

value detection, which uses discrete differential operators to calculate the approximate gradient of pixel gray level for the pixel points in the image edge area. The greater the gradient value is, the higher the possibility of edge is. This algorithm is called gradient calculation method, also known as Sobel algorithm. For a digital image, the gradient vector at a pixel point can represent that the directional derivative at the point gets the maximum value along the direction, as shown in Formula (1), that is, the image changes fastest along the gradient direction at the point, that is, the so-called edge. The gradient is the sum of X direction (horizontal) and Y direction (vertical), and the continuous form of X direction of gradient at a point is shown in Eq. (2). The same is true for the Y direction. In the discrete system, the differential is approximated by subtracting the previous result from the current one.

$$\text{grad } f(\text{x,y}) = \nabla f(x,y) = \{\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\} \tag{1}$$

$$\frac{\partial f}{\partial x} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x, y) - f(x,y)}{\Delta x} \tag{2}$$

The core of Sobel algorithm is two 3*3 anisotropic gradient matrices: X and Y, namely Sobel operator. X is used to calculate the horizontal gradient and Y is used to calculate the vertical gradient, as shown in Figure 1 and Figure 2. The matrix is obtained by weighting the gradient in four directions including horizontal, vertical and two diagonals [5].

Sobel algorithm conducts convolution operation between image pixel value and Sobel operator [16], and obtains the gradient value of image gray level using difference calculation method. According to the gradient value, it can be judged whether the pixel point is a boundary. The operator in X direction and Y direction is convolved with the matrix to obtain the corresponding result, and then the gradient value is obtained through the gradient calculation formula of the image gray function. Finally, the edge information is determined by the threshold operation. It is assumed that A is the 3*3 gray matrix of the image to be detected. Let the X direction and Y direction operators conduct convolution operation with A matrix, and the results obtained are represented by G_x and G_y, as shown in the following Eqns. (3) and (4).

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A = [-1 \quad 0 \quad 1] * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * A \tag{3}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A = [1 \quad 2 \quad 1] * \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * A \tag{4}$$

According to the gradient templates (a) and (b) in Figure 2, they are convolved with the pixels of the image, and the formula is as follows:

$$G_x = (z1 + 2*z2 + z3) - (z7 + 2*z8 + z9) \tag{5}$$

$$G_y = (z3 + 2*z6 + z9) - (z1 + 2*z4 + z7) \tag{6}$$

Then the gradient value is obtained through the gradient calculation formula of the image gray function. The total gradient $G = \sqrt{G_x^2 + G_y^2}$, which is actually simplified to the absolute value form $|G| = |G_x| + |G_y|$, and the gradient direction $\theta = \arctan\frac{G_y}{G_x}$. According to the θ value, the direction angle of the gradient change of the pixel point is obtained, so as to determine the position of the next boundary pixel point. The threshold operation can be used to determine the edge information. Assuming that the threshold value is M, if it is greater than M, the pixel is a boundary value. If it is less than M, it is not a boundary value. Because Sobel operator weights the influence of pixel position, it can lower down the degree of edge blur, and has better detection effect than the traditional Prewitt operator and Roberts operator.

z1	z2	z3
z4	z5	z6
z7	z8	z9

Figure 1. 3*3 Area

−1	0	1
−2	0	2
−1	0	1

(a)

1	2	1
0	0	0
−1	−2	−1

(b)

Figure 2. Sobel operator horizontal and vertical gradient template

3. IMPLEMENTATION OF EDGE DETECTION ACCELERATOR BASED ON SOBEL

3.1 Overall implementation scheme of Sobel algorithm

As shown in Figure 3, Sobel algorithm is optimized in this paper, which is suitable for accelerator implementation through hardware. First, the received 24 bit RGB format data information of each pixel is converted into 8-bit gray image, and the gray images are stored in Cache. At the same time, according to the properties of two-dimensional convolution, the convolution window is slid to accumulate the index values below the current row. The offset value of the convolution window relative to the initial position is calculated when convolving the 3*3 convolution window and the image in the cache region. After the matrix multiplication is completed by multiplying the convolution kernel and image matrix, whether the index values of the current row and column of the pixel are greater than or equal to 2 can be judged. The judgment of the index value is used to carry out the zero filling processing to ensure that the output image pixel is consistent with that of the source image, and other data are sent into the subfunction of the operation window to sum the multiplication result window. The accelerator decomposes two-dimensional discrete convolution into matrix multiplication and window

summation. The two-dimensional discrete convolution formula is shown in Formula (7), where f [x, y] refers to the image matrix and g [x, y] refers to the operator matrix.

$$f[x,y]*g[x,y]=\sum_{n1=-\infty}^{+\infty}\sum_{n2=-\infty}^{+\infty}f[x,y]\cdot g[x,y] \tag{7}$$

3.2 Pixel level fine grain image parallel data processing

The original data of the accelerator is the video picture captured by the camera in real time. Each frame of the video is an image. When the video image processing is implemented, different granularity of processing interval can be adopted for images to achieve different computing power. For an image or a frame of a video, there are three different processing granularities: from small to large, pixel level, block level, and frame level. In order to achieve the optimal efficiency of accelerator and the simplest design, the advantages and disadvantages of each processing granularity are discussed and the processing granularity is determined. In a video, a static image is a frame, and several static images form a video. Frame level processing is to read a whole picture or a frame of video into memory at one time for overall calculation. Block level processing refers to that a frame of video or an image can be divided into several blocks, one image block is calculated each time, and the whole image is finally processed. Pixel is the basic unit of the image, and the pixel level processing is to read in each pixel at one time and process them in turn.

The processing granularity in the code of traditional software implementation algorithm is frame level processing, which has two problems: first, low efficiency and high resource consumption appear. When a frame of video image is processed, the whole image needs to be read into the memory and then calculated. The resources in the reconfigurable hardware are very limited. When the amount of data is large, the image stored in the memory consumes resources. In addition, the calculation is not performed until a whole image is read in, causing low efficiency. In fact, when the image is read in a small part, the calculation can be performed to reduce the waiting time. Second, the simultaneous design of signal is complex.

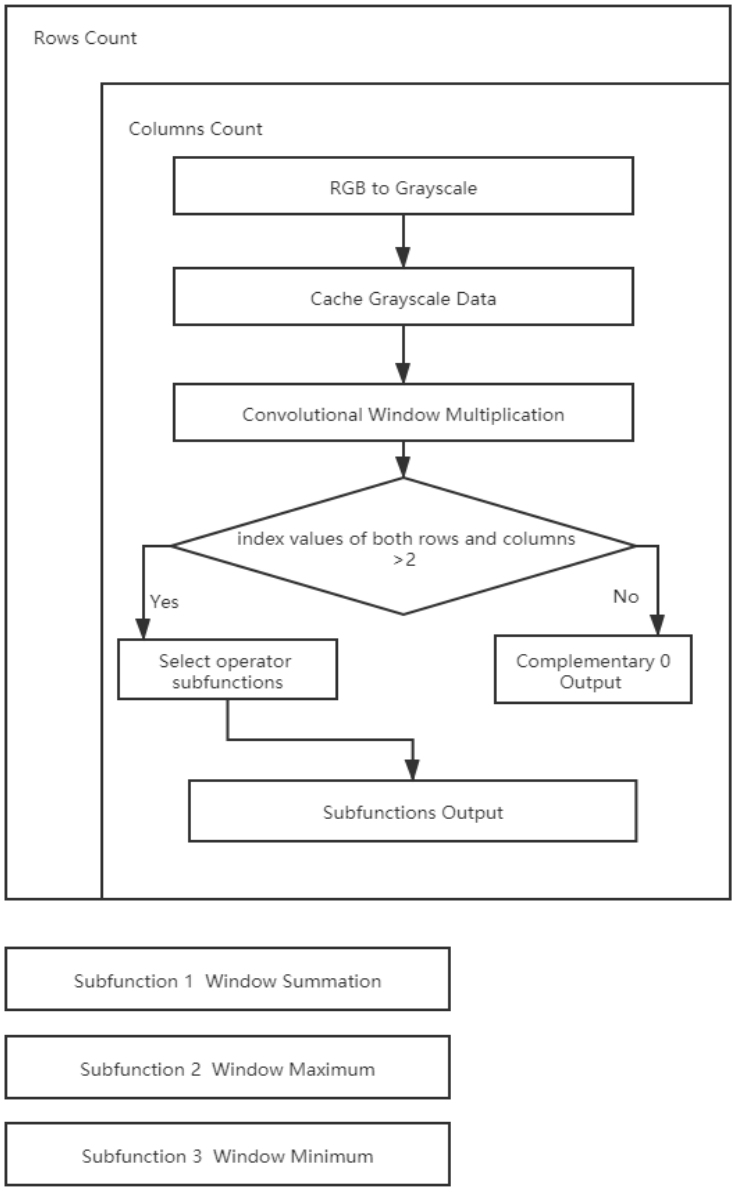

Figure 3. Sobel algorithm implementation flowchart

As shown in Figure 4, all video frames have three areas: vertical blanking area, horizontal blanking area and active video area. The converting of optical signals to electrical signals starts from the upper left corner and moves horizontally forward. At the same time, the scanning point moves downward at a certain rate. When the scanning point reaches the far right, it quickly returns to the left, generating a horizontal blanking area in the process gap. After scanning an image, the scanning point returns to the upper left corner from the lower right corner of the image to start scanning the next frame. This time interval generates a vertical blanking area [17]. Therefore, when processing a frame of a video image, it is necessary to avoid the horizontal and vertical blanking areas. When it is in the active area, it is necessary to carry out the algorithm operation of edge extraction, which is more complex. Such processing leads to the need to increase the processing of synchronization signals, the addition of which will lead to the upper design.

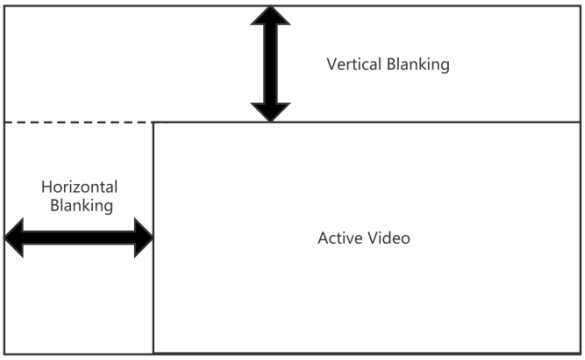

Figure 4. Video frame area

Using block level processing granularity, we need to consider the image blocking strategy. Since Sobel operator is a 3*3 area, we can define the hardware only for gradient calculation part, that is, gradient operator only acts on a 3*3 area and does not process the whole image. If hardware modules are created in this way, additional hardware modules are required. Before calculation, an image is divided into several small windows in the size of 3*3, and additional modules are added, which increases the difficulty of system design and debugging. It can also be considered to divide a row into a module to process data in one row at a time to reduce processor interaction. However, Sobel operator needs multiple rows of data for edge extraction, and complex control mechanism synchronization signals need to be designed to synchronize multiple rows of data.

To sum up, this paper adopts pixel level fine-grained image data processing, which is suitable for large-scale image processing, saves storage space, simplifies synchronization mechanism, and improves the efficiency of edge detection.

In the design of the accelerator, the receiver converts the RGB24 format image into 8-bit gray-scale image. The gray-scale conversion is to facilitate the design of cache, reduce the amount of computation, and facilitate circuit integration. A parallel pipelining storage system is designed to accelerate the algorithm of architecture. The gray-scale conversion process is shown in Formula (8). RGB is the 8bit value of the red, green and blue components of the RGB 24 format.

$$Y = 0.229R + 0.587G + 0.114B \tag{8}$$

3.3 Row buffer storage architecture

The 9 pixels required by Sobel algorithm for a calculation are not in the same line. As shown in the left one of Figure 5, when calculating the gradient value of F point, it is necessary to read all the data in the row of A and E pixel points, as well as the data in the row of I pixel point to K pixel point. The row buffer can provide simultaneous access to multiple different lines in a clock cycle. Therefore, the line buffer is required to provide a cache of multiline pixels. In addition, since Sobel operator is a 3*3 data area, a 3*3 sliding window is required to cache the data that needs to be convolved with Sobel, in addition to a row buffer for caching 3 rows of data.

As shown in Figure 5, this paper designs a row buffer storage architecture that can buffer three rows of pixels. The length is the width of the original image. The row buffer can store three rows of complete data of the original image. First, the pixels of ROW0 line in the original image enter ROW2 line of the line buffer in turn for caching. Since ROW0 line is the boundary line of the image, it will not be calculated with Sobel calculation, but will directly process the boundary gradient and directly assign the gradient value. After the ROW2 area of the line buffer is filled, move up to the ROW1 line of the line buffer, and the pixels of the original image ROW1 line enter the ROW2 line of the line buffer in turn. Since ROW1 line of the original image is a sub boundary line, it will not be operated with Sobel operator, but directly enters into boundary gradient processing and directly assigns gradient values. The data of ROW1 and ROW2 in the row buffer are moved up into ROW0 and ROW1 rows. Pixels of ROW2 row in the original image are stored in ROW2 row of the row buffer in turn. When the three rows of the row buffer are full, the row buffer enters the sliding window in sequence according to COL0, COL1... COLn columns. Finally, as shown in Figure 6, the pixel in the sliding window is convolved with Sobel operator to obtain the result.

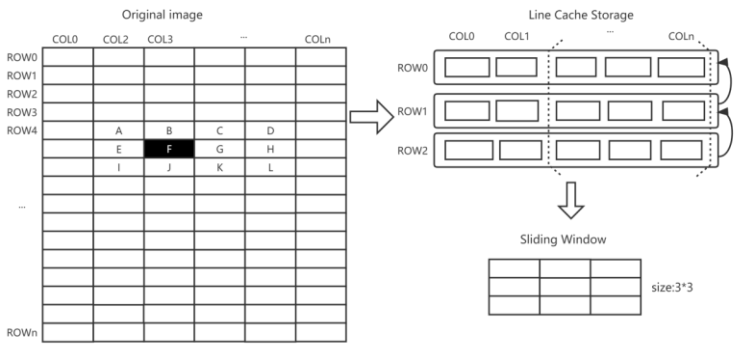

Figure 5. Line cache storage structure

Figure 6. Sliding window and Sobel operator convolution

After all the pixels in ROW0, ROW1, and ROW2 lines of the line buffer are processed, the line buffer moves the pixels upward, discarding the pixels in ROW0 line, moving the pixels in ROW2 line into ROW1 line, moving the pixels in ROW1 line into ROW0 line, and the data in ROW3 line of the original image will be successively cached in ROW2 line of the line buffer. And the rest is done in the same manner, until the whole image is processed.

4. RESULTS AND ANALYSIS

4.1 Validation test platform

The verification test platform is mainly based on Zynq-7000, which is internally equipped with an ARM Cortex-A9 processor and FPGA [7]. Its peripheral hardware mainly consists of HDMI interface, 7-inch IPS screen and keyboard. The verification test platform has PS, PL and other peripheral interfaces, such as Ethernet, HDMI, GPIO, USB, UART, etc. The specific architecture of the verification test platform is shown in Figure 7. The Zynq based SOC is divided into PS end and PL end. The PS end of the system is responsible for image acquisition, the PL end is responsible for video image processing and display, and the AXI protocol is used for data transmission at PL and PS ends. The PL end is equipped with FPGA, Sobel accelerator is implemented on FPGA, and VDMA is the video transmission module in Zynq. After the PS end of the verification test platform conducts video capture, the video stream is directly sent to the image processing module at the PL end through the pin for image processing. The overall architecture is simple and easy to implement.

Test process

The input test data is the real-time video data stream captured by the camera. Based on the verification test platform, Sobel algorithm can be smoothly run under 1024 * 768 (60fps) video input. The effect of Sobel edge extraction is shown in Figure 8, where (a) is one frame of the original video image, and (b) is the result of edge extraction. The system has good real-time performance when using hardware to accelerate processing with Sobel algorithm, and video images do not suffer from jamming and tearing.

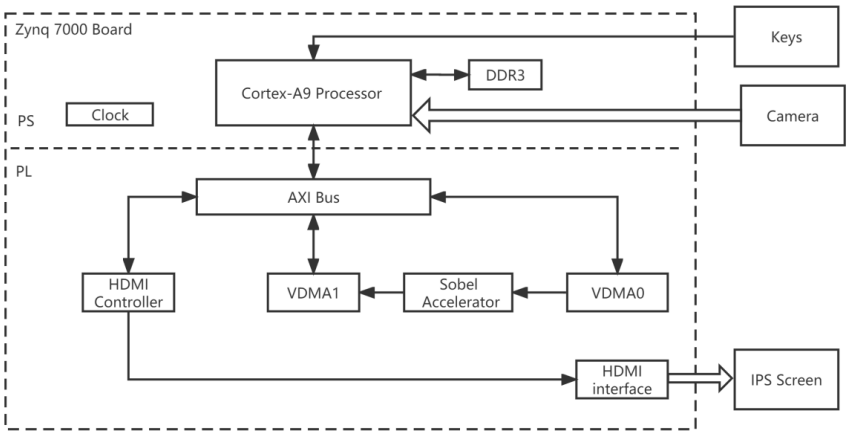

Figure 7. Test and verification platform

Figure 8. Sobel edge detection results

4.2 Comparative experiment and analysis

The efficiency of video image edge extraction is measured by the time of a frame processing. The shorter the time is, the higher the efficiency of edge processing is. In order to obtain more comprehensive results, this paper uses images with different resolutions to verify one by one. The horizontal comparison of Sobel edge extraction efficiency by using hardware means that the time taken by FPGA-based edge detection systems in different papers to detect images is compared with that of this system. The results are shown in Table 1.

The image edge detection systems are all based on static images [18], and the processing time is calculated according to the processing time of an image, while Monson et al. [19] directly used FPGA to process video stream data without the help of a processor. Taslimiet et al. [20] and Jiang et al. [21] enhanced the processing capacity by improving the computing components. However, Kumar et al. [22] uses a low efficiency FPGA, so its processing capacity is relatively poor.

At the same time, through the analysis of experimental results, we can see that the FPS of each method has been more than 30, which meets the real-time requirements of the human eye, and is one order of magnitude higher than the human eye in terms of real-time requirements. This means that the three algorithms all meet the video post-processing flow and the characteristics of non-jamming. It can be seen that although the architecture of each accelerator is different, this article still has certain advantages in processing time.

In addition, this paper also compares the performance of hardware and software in processing different pixel images, as shown in Table 2. The software is based on Intel i5-6300HQ processor.

It can be seen that the efficiency of the reconfigurable accelerator for edge detection is much higher than that of the software. With the increase of image pixels, the time growth rate of the software for edge detection is more obvious than that of the hardware for edge detection. That is to say, with the increase of image pixels, the FPGA-based acceleration effect is more significant. High-performance CPU can achieve the same performance as reconfigurable accelerator [23], but the power consumption of CPU is 6 times that of FPGA. The GPU is selected as the processing unit and the edge detection accelerator is designed. Although the processing capacity is slightly higher than that of CPU and FPGA, the power consumption is higher than that of CPU [24, 25]. This further shows that the role of the high-performance CPU or GPU in enhancing the system performance is limited even though they continue to improve their performance or add more multi-cores, while the reconfigurable accelerator based on FPGA is an ideal choice for accelerating embedded graphics and image processing, which can meet the stringent requirements of power consumption and resources.

Table 1. FPGA based edge detection accelerator comparison

Papers	FPGA	Resolution	Time(ms)
[18]	Xilinx Virtex-5	640*480	6.41
[19]	Xilinx Virtex-7	640*480	2.58
[20]	Xilinx Virtex-6	512*512	2.2
[21]	Xilinx Virtex-5	512*512	2.62
[22]	Xilinx xc7z020-1clg484	320*320	80.47
This paper	Zynq 7000	640*480	2.03

Table 2. Hardware and software edge detection comparison

Resolution	Hardware(ms)	Software(ms)	Acceleration ratio
640*480	2.0	12.2	6.1
1280*720	8.2	58.7	7.2

5. SUMMARY

In this paper, a novel Sobel edge detection accelerator based on reconfigurable architecture is designed. The accelerator uses pixel level fine grain image parallel data processing and row buffer storage architecture to improve the processing ability of the accelerator. Video streams with different resolutions are input, and Sobel edge detection contrast experiments are carried out through software and hardware, respectively, to verify the acceleration effect of the accelerator proposed in this paper. At the same time, comparing the accelerator with other similar accelerators, it can be seen that the accelerator proposed in this paper has more acceleration advantages than similar accelerators, and can improve the processing efficiency by 10%.

ACKNOWLEDGMENT

This research was supported by Zhejiang Provincial Natural Science Foundation of China (Grant No.: LY19F020008) and Zhejiang Province Commonweal Projects of China (Grant No.: LGG19F03007).

REFERENCES

[1] Shah, B.K., Kedia, V., Raut, R., Ansari, S., Shroff, A. (2020). Evaluation and comparative study of edge detection techniques. IOSR Journal of Computer Engineering, 22(5): 6-15.

[2] Gaurav, K., Ghanekar, U. (2018). Image steganography based on Canny edge detection, dilation operator and hybrid coding. Journal of Information Security and Applications, 41: 41-51. https://doi.org/10.1016/j.jisa.2018.05.001

[3] Menaka, R., Janarthanan, R., Deeba, K. (2020). FPGA implementation of low power and high speed image edge detection algorithm. Microprocessors and Microsystems, 75: 103053. https://doi.org/10.1016/j.micpro.2020.103053

[4] Zhang, K., Zhang, Y., Wang, P., Tian, Y., Yang, J. (2018). An improved sobel edge algorithm and FPGA implementation. Procedia Computer Science, 131: 243-248. https://doi.org/10.1016/j.procs.2018.04.209

[5] Buzzi, S., Grossi, E., Lops, M., Venturino, L. (2021). Radar target detection aided by reconfigurable intelligent surfaces. IEEE Signal Processing Letters, 28: 1315-1319. https://doi.org/10.1109/LSP.2021.3089085

[6] Aziz, S.M., Hoskin, D.H., Pham, D.M., Kamruzzaman, J. (2022). Remote reconfiguration of FPGA-based wireless sensor nodes for flexible Internet of Things. Computers and Electrical Engineering, 100: 107935. https://doi.org/10.1016/j.compeleceng.2022.107935

[7] Menaka, R., Janarthanan, R., Deeba, K. (2020). FPGA implementation of low power and high speed image edge detection algorithm. Microprocessors and Microsystems, 75: 103053. https://doi.org/10.1016/j.micpro.2020.103053

[8] Ravivarma, G., Gavaskar, K., Malathi, D., Asha, K.G., Ashok, B., Aarthi, S. (2021). Implementation of Sobel operator based image edge detection on FPGA. Materials Today: Proceedings, 45: 2401-2407. https://doi.org/10.1016/j.matpr.2020.10.825

[9] Sangeetha, D., Deepa, P. (2019). FPGA implementation of cost-effective robust Canny edge detection algorithm. Journal of Real-Time Image Processing, 16(4): 957-970. https://doi.org/10.1007/s11554-016-0582-2

[10] Hagara, M., Stojanović, R., Bagala, T., Kubinec, P., Ondráček, O. (2020). Grayscale image formats for edge detection and for its FPGA implementation. Microprocessors and Microsystems, 75: 103056. https://doi.org/10.1016/j.micpro.2020.103056

[11] Wang, Y., Xie, W., Chen, H., Li, D.D.U. (2021). Multichannel time-to-digital converters with automatic calibration in Xilinx Zynq-7000 FPGA devices. IEEE Transactions on Industrial Electronics, 69(9): 9634-9643. https://doi.org/10.1109/TIE.2021.3111563

[12] Wang, P., McAllister, J. (2016). Streaming elements for FPGA signal and image processing accelerators. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 24(6): 2262-2274. https://doi.org/10.1109/TVLSI.2015.2504871

[13] Vourvoulakis, J., Kalomiros, J., Lygouras, J. (2016). Fully pipelined FPGA-based architecture for real-time SIFT extraction. Microprocessors and Microsystems, 40: 53-73. https://doi.org/10.1016/j.micpro.2015.11.013

[14] Amiri, M., Siddiqui, F.M., Kelly, C., Woods, R., Rafferty, K., Bardak, B. (2017). FPGA-based soft-core processors

for image processing applications. Journal of Signal Processing Systems, 87(1): 139-156. https://doi.org/10.1007/s11265-016-1185-7

[15] Nausheen, N., Seal, A., Khanna, P., Halder, S. (2018). A FPGA based implementation of Sobel edge detection. Microprocessors and Microsystems, 56: 84-91. https://doi.org/10.1016/j.micpro.2017.10.011

[16] Han, L., Tian, Y., Qi, Q. (2020). Research on edge detection algorithm based on improved sobel operator. InMATEC Web of Conferences, 309: 03031. https://doi.org/10.1051/matecconf/202030903031

[17] Vieri, C., Lee, G., Balram, N., Jung, S.H., Yang, J.Y., Yoon, S.Y., Kang, I.B. (2018). An 18 megapixel 4.3″1443 PPI 120 Hz OLED display for wide field of view high acuity head mounted displays. Journal of the Society for Information Display, 26(5): 314-324. https://doi.org/10.1002/jsid.658

[18] Chaple, G., Daruwala, R.D. (2014). Design of Sobel operator based image edge detection algorithm on FPGA. In 2014 International Conference on Communication and Signal Processing, 788-792. https://doi.org/10.1109/ICCSP.2014.6949951

[19] Monson, J., Wirthlin, M., Hutchings, B.L. (2013). Optimization techniques for a high level synthesis implementation of the Sobel filter. In 2013 International Conference on Reconfigurable Computing and FPGAs (ReConFig), 1-6. https://doi.org/10.1109/ReConFig.2013.6732315

[20] Taslimi, S., Faraji, R., Aghasi, A., Naji, H.R. (2020). Adaptive edge detection technique implemented on FPGA. Iranian Journal of Science and Technology, Transactions of Electrical Engineering, 44(4): 1571-1582. https://doi.org/10.1007/s40998-020-00333-5

[21] Jiang, J., Liu, C., Ling, S. (2018). An FPGA implementation for real-time edge detection. Journal of Real-Time Image Processing, 15(4): 787-797. https://doi.org/10.1007/s11554-015-0521-7

[22] Kumar, V., Asati, A., Gupta, A. (2017). Hardware implementation of a novel edge-map generation technique for pupil detection in NIR images. Engineering Science and Technology, an International Journal, 20(2): 694-704. https://doi.org/10.1016/j.jestch.2016.11.001

[23] Georgis, G., Lentaris, G., Reisis, D. (2019). Acceleration techniques and evaluation on multi-core CPU, GPU and FPGA for image processing and super-resolution. Journal of Real-Time Image Processing, 16(4): 1207-1234. https://doi.org/10.1007/s11554-016-0619-6

[24] Iqbal, B., Iqbal, W., Khan, N., Mahmood, A., Erradi, A. (2020). Canny edge detection and Hough transform for high resolution video streams using Hadoop and Spark. Cluster Computing, 23(1): 397-408. https://doi.org/10.1007/s10586-019-02929-x

[25] Du, C., Yuan, J., Dong, J., Li, L., Chen, M., Li, T. (2020). GPU based parallel optimization for real time panoramic video stitching. Pattern Recognition Letters, 133: 62-69. https://doi.org/10.1007/s10586-019-02929-x