





## Research Article

# Design and Implementation of Local Threshold Segmentation Based on FPGA

Shangshang Gao <sup>1</sup>, Yuanyuan Wang,<sup>1</sup> Zhaofeng Chen <sup>1</sup>, Feng Zhou <sup>1</sup>,  
Rugang Wang <sup>1</sup> and Naihong Guo<sup>2</sup>

<sup>1</sup>School of Information Technology, Yancheng Institute of Technology, Yancheng 224051, China

<sup>2</sup>Yancheng Xiongying Precision Machinery Company Limited, Yancheng 224006, China

Correspondence should be addressed to Rugang Wang; [wrg3506@ycit.edu.cn](mailto:wrg3506@ycit.edu.cn)

Received 19 March 2022; Revised 26 June 2022; Accepted 1 July 2022; Published 22 July 2022

Academic Editor: Mohamed Louzazni

Copyright © 2022 Shangshang Gao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In the process of the development of image processing technology, image segmentation is a very important image processing technology in the field of machine vision, pedestrian detection, medical imaging, and so on. However, the traditional image segmentation technology cannot solve the problems of reflection and uneven illumination. This paper presents a local threshold segmentation method based on FPGA, which can automatically select the optimal threshold according to different gray levels of images. First, the image is processed by mean filtering to remove noise interference in the image. Then, the idea of the mean value of the local neighborhood block and the Gaussian weighted sum in the local neighborhood is used to deal with the reflective and uneven light on the image. The process is designed and realized on FPGA. Finally, the design algorithm is verified by ModelSim simulation software and QT5 software. The experimental results show that the algorithm can effectively solve the problems of reflection and uneven illumination on the image surface, and the segmentation effect is significantly improved compared with the fixed threshold algorithm and Otsu algorithm. It also has certain reference value in medicine, agriculture, engineering, and other fields.

## 1. Introduction

With the development of image processing technology, digital image capture and processing technology is developing towards a higher level. The traditional image processing system based on software platforms has been difficult to meet the needs, so people put forward new requirements for the image processing system. Due to the natural parallelism of image processing algorithms, the addition of field-programmable gate array (FPGA) hardware platforms has brought new vitality to image processing [1–6]. In addition, image segmentation is a very important image processing technology, especially in the medical field. For example, the lung image was segmented into cancer and noncancer parts by a superpixel algorithm [7]. Therefore, image segmentation technology has been concerned and valued. Thousands of kinds of image segmentation

algorithms have been proposed, and new image segmentation algorithms are constantly being born. Common image segmentation methods include threshold-based segmentation methods, edge-based segmentation methods, and region-based segmentation methods [8–13]. In recent years, Bo et al. [14] proposed a new deformable contour model for ultrasonic image sequence segmentation, which can resist the influence of misleading or weak boundary in ultrasonic image segmentation. Hongyu et al. [15] proposed a rib segmentation framework based on unpaired sample enhancement and a multiscale network, which has good segmentation performance for multiorgan overlapping regions and fuzzy regions. Zhao et al. [16] proposed an automatic segmentation method for small organs based on limited training data, which is superior to cutting-edge deep learning methods, traditional forest-based methods, and multiatlas methods in small organ segmentation. Li and Zou

et al. [17] proposed a portrait image segmentation method based on the combination of an improved genetic algorithm and threshold image segmentation, which solved the shortcomings of unsatisfactory segmentation effect and low segmentation accuracy when traditional algorithms were applied to portrait image segmentation. For the threshold segmentation method, if the image is interfered by external factors, such as reflection and uneven illumination, resulting in a large gap in the gray level of the image surface, then how to solve this phenomenon has a certain challenge. Local threshold segmentation can be used for multitarget segmentation and local threshold selection, but the target segmentation in the local threshold has poor connection and contains noise, which is suitable for close recognition of the target image. Therefore, this paper proposes a design based on FPGA local threshold segmentation, using the local neighborhood block means and the idea of Gaussian weighted sum in the local neighborhood; in the case of the image surface, the gray level gap is too large, completing the selection of adaptive local threshold, through Modelsim code simulation, and Visual Studio QT5 is connected to the platform for debugging and running display.

## 2. Local Adaptive Threshold Algorithm

The size of the threshold determines the accuracy of the image information. If the threshold is too high, some detailed images will be filtered and the edges are seriously interrupted. If the threshold is too low, more false edges of the images will cause the judgment error of the image information. Local thresholding algorithms can enable local image regions such as brightness, contrast, and texture to have corresponding optimal thresholds.

Common local threshold segmentation algorithms include the mean value of the local neighborhood blocks and

the Gaussian weighted sum of the local neighborhood blocks. Based on this, the processing window is set as an odd-number square window, assuming  $\mu$  is the mean value of the pixels in the processing window,  $\sigma^2$  is the pixel variance in the processing window,  $I(x, y)$  is the input pixel value, and  $g(x, y)$  is the output pixel value. The calculation formulas of  $\mu$  and  $\sigma^2$  and  $g(x, y)$  are as follows:

$$\mu = \frac{1}{(2r+1)^2} \sum_{i=-r}^r \sum_{j=-r}^r I(x+i, y+j), \quad (1)$$

$$\sigma^2 = \frac{1}{(2r+1)^2} \sum_{i=-r}^r \sum_{j=-r}^r [I(x+i, y+j) - \mu]^2, \quad (2)$$

$$g(x, y) = \begin{cases} 1, & \Delta(I(x, y) - \mu)^2 > K^2 \sigma^2 \text{foreground}, \\ 0, & \Delta(I(x, y) - \mu)^2 \leq K^2 \sigma^2 \text{background}, \end{cases} \quad (3)$$

where “ $r$ ” is the radius of the processing window and  $K$  is a constant greater than 0. When  $K=1$ , the  $(I(x, y) - \mu)^2 > \sigma^2$  inequality is always true, which indicates that the result of segmentation is the same whether the foreground is high brightness or low brightness. The schematic diagram of the segmentation effect is shown in Figure 1.

Before segmentation, the center point of the picture represents the foreground of different brightness, and the gray level represents the background. After the segmentation algorithm, black represents the foreground and white represents the background. The segmentation results of different brightness prospects are consistent. Therefore, the problem caused by uneven light to the image can be solved by (3). To understand and reduce the difficulty of writing the FPGA code, let the current output pixel is  $din(x, y)$ , and the pixel treated by the algorithm is  $dout(x, y)$ . Convert equation (3) to the transformed expression as follows:

$$dout(x, y) = \begin{cases} 8'hf f \Delta (2r+1)^2 \times (din(x, y) - \mu)^2 > K^2 \sum_{i=-r}^r \sum_{j=-r}^r [I(x+i, y+j) - \mu]^2, \\ 8'h00 \Delta (2r+1)^2 \times (din(x, y) - \mu)^2 \leq K^2 \sum_{i=-r}^r \sum_{j=-r}^r [I(x+i, y+j) - \mu]^2. \end{cases} \quad (4)$$

In this experiment, the radius “ $r$ ” of the processing window is 7, and the constant  $K$  is 1. Bring them into formula (4) to obtain the expression as follows:

$$dout(x, y) = \begin{cases} 8'hf f \Delta 225 \times (din(x, y) - \mu)^2 > \sum_{i=-r}^r \sum_{j=-r}^r [I(x+i, y+j) - \mu]^2, \\ 8'h00 \Delta 225 \times (din(x, y) - \mu)^2 \leq \sum_{i=-r}^r \sum_{j=-r}^r [I(x+i, y+j) - \mu]^2. \end{cases} \quad (5)$$

## 3. FPGA Implementation

**3.1. Overall Design.** At present, pipelined video positioning systems are based on universal processors and adopt OpenCV machine vision technology, which has shortcomings such as long response time, poor real-time performance, high cost, and insufficient flexibility, while FPGA

has rich logic and storage resources and unique parallel processing advantages. Therefore, this paper adopts the pipeline way to achieve the local adaptive threshold on FPGA. According to the requirements of equation (5), the following steps are required: (1) calculate  $\mu$  in the processing window; (2) make the difference between the center pixel of the window  $din(x, y)$  and  $\mu$  and calculate the square of the

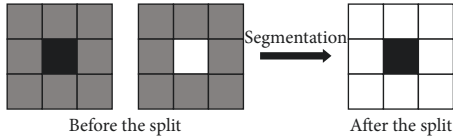


FIGURE 1: Schematic diagram of segmentation effect.

difference  $(d_i n(x, y) - \mu)^2$ ; (3) multiply the result of step (2) by 225 to complete the calculation of the left inequality; (4) calculate the 225-pixel values in the processing window and complete the calculation of the inequality on the right with the square sum of  $\mu$ ; (5) compare steps (3) and (4) to complete the local adaptive threshold segmentation; (6) align the rows and columns, and complete the boundary processing. The overall design of implementing locally adaptive threshold segmentation in FPGA is shown in Figure 2.

Among them, the window cache module and the mean filtering module are used inside the data superposition module. In order to ensure that the mean in the processing window and the pixels of the current processing window cache differ in the same timing, that is, the timing of the mean filtering module and the window cache module should be consistent, so a delay of 11 beats is required before the current window cache module. Since the data superposition module port does not support array entry, only the pixel data after the square can be assembled into a vector and input to the superposition module port. At the same time, the square value of the center pixel of the left inequality does not be recalculated and can be extracted directly from the new vector. After calculation, the multiplication circuit consumes only 2 beats, while the data superposition module uses 8 beats, so it needs to delay 6 beats after the multiplication circuit. The timing alignment is compared to complete the local adaptive threshold segmentation. Finally, the boundary is reset according to the valid data flag bit. Some of the main programs are as follows:

```

/* The squared pixel data is spliced into a vector */
assign square [(i + 1) * 2 * Bit W - 1: i * 2 * Bit W] =
square_pixdiff [i]
/* Extract the square value of the difference between
the center pixel and the mean */
parameter med_pix = num_all >> 1
assign square_med_pix = square[(med_pix + 1) * 2 * Bit
W - 1: med_pix * 2 * Bit W]

```

**3.2. Design of the Window Cache Module.** Sliding window cache design is the basic operation commonly used in FPGA when processing images, which is suitable for real-time and efficient pipeline processing on FPGA. In this paper, the size of the sliding window cache is set to  $15 \times 15$  rectangular boxes, which requires delaying 14-row direction and 14-column direction image pixels, namely, consuming 14 lines of FIFO to complete the line cache and 225 registers to complete the column cache. When FPGA caches one line of the image, the new line of image data is transferred to

Line\_FIFO1, and Line\_FIFO1 caches the image data. When the Line\_FIFO1 cache data reaches one row, Line\_FIFO1 reads the cached image data before the next row arrives and passes it to the next Line\_FIFO. At the same time, the new row of data is cached again into Line\_FIFO1, until the image data cache is complete. The  $15 \times 15$  window cache structure is shown in Figure 3.

**3.3. Design of Data Stack Module.** The function of the data stack module is to add up all the pixel values in the processing window. This experiment used two data superposition modules, one is seeking the cached data sum in the processing window; the other is stacking the square sum of the difference between the pixels and the mean in the processing window. However, stacking 225 pixels in turn is not only heavy work but also difficult for code readability and maintainability. Therefore, this paper describes the data superposition circuit by the recursive method. By summarizing the rules, it is found that if the current number to be added is  $n$  (odd number),  $n - 1/2$  as an adder and  $n - 1/2 + n\%2$  as a register are required, that is, the number to be added next time is  $n - 1/2 + n\%2$ . The bisection recursive calling formula is shown in

$$\text{sum}(n, x) = \sum_{i=0}^{(n-1)/2} [x(2i) + x(2i+1)]. \quad (6)$$

The recursive summation block diagram is shown in Figure 4. In Figure 4, the input of `din_vector` is the sum of squares of the difference between all pixels and the mean in the 225 processing window. The `first_vector` is the pixel vector after the first superposition and serves as the input vector for the next superposition, and the `last_vector` is the pixel vector after the last superposition, completing the superposition of 255-pixel data in this cycle.

**3.4. Design of the Mean Filter Module.** The mean filter module is a linear filtering method with a simple algorithm and high smoothness and has a good inhibitory effect on periodic interference. Its main function is to reduce the sharp change of the image gray value to achieve the purpose of reducing noise. If  $r$  is used to represent the radius of the processing window,  $f(x, y)$  represents the input pixel value of the current window and the output pixel value of  $g(x, y)$  the processing window; the mean filter can be expressed by

$$g(x, y) = \frac{1}{(2r+1)^2} \sum_{i=-r}^r \sum_{j=-r}^r f(x+i, y+j). \quad (7)$$

In image filtering processing, when the image processes the boundary pixels, the convolution core and the image use area cannot match, which will cause calculation problems, so the boundary should also be processed. Common boundary processing methods include boundary filling of 0, boundary filling of the nearest pixel values, and no boundary processing. Boundary filling 0 is to expand each boundary of the image and set the extended boundary to 0. Boundary filling the nearest pixel value is also to expand each boundary of the

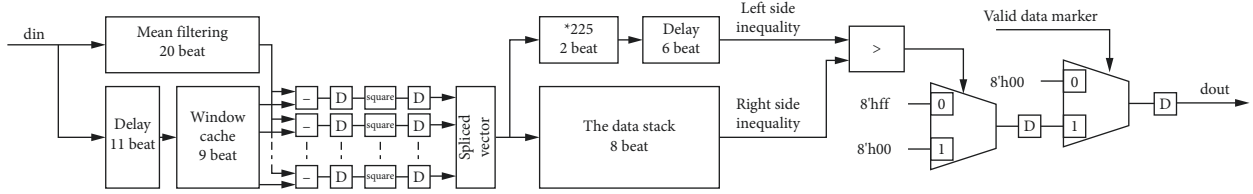


FIGURE 2: Overall design.

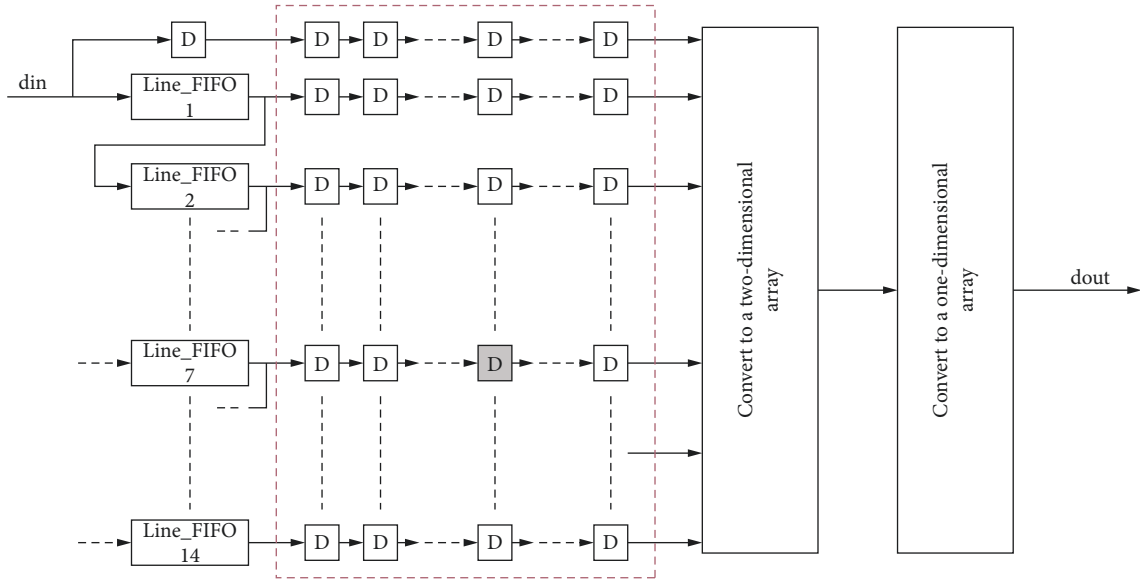


FIGURE 3: 15×15 window cache structure.

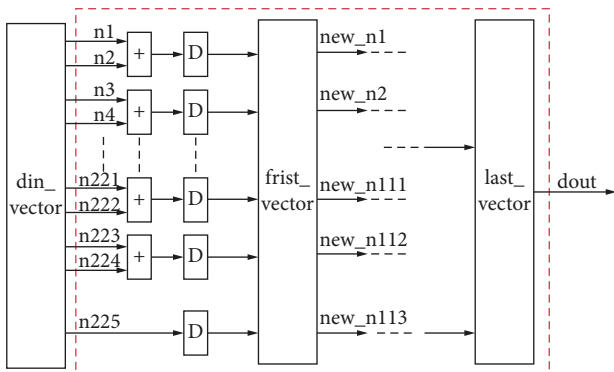


FIGURE 4: Recursive summation block diagram.

image and set the extended boundary to the value of adjacent pixels. In this paper, only the upper boundary is treated, and the rest is not treated. The block diagram of the mean filter module is shown in Figure 5.

### 4. System Test

4.1. Experimental System Design. To verify the effectiveness of the algorithm, the system mainly uses the QT5 software design program to convert the image and text files to each



FIGURE 5: Block diagram of the mean filter module.

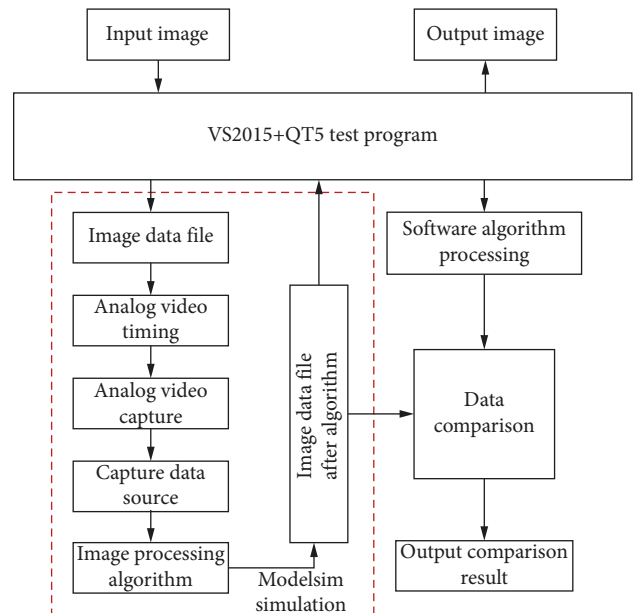


FIGURE 6: System block diagram.

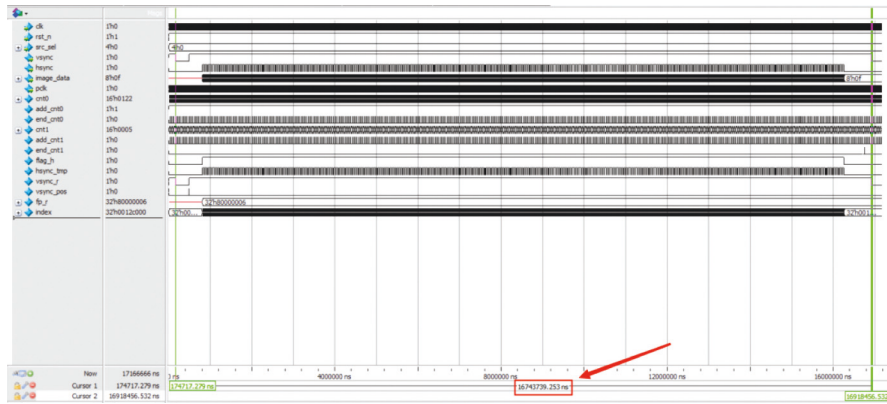


FIGURE 7: Simulation diagram of the video stream.

other, and the designed algorithm is simulated and tested by the Modelsim software. Because it is a simulation test, first simulate a video clock and then simulate the acquisition of image data, according to the clock requirements to realize the capture of image data. Finally, the captured image data is transmitted to the designed algorithm for the simulation test. Modelsim will output text data after the algorithm in this article, which can be converted into image data and displayed on the host computer through the QT5 software. In addition, this paper also designs some commonly used image processing algorithms for global threshold segmentation through VS2015; compared with the algorithm used in this paper, the final comparison results are output. The block diagram of the experimental test system is shown in Figure 6.

**4.2. Video Streaming Test.** In this paper, images with a resolution of  $640 \times 480$  and a scanning frequency of 60 HZ are tested. Then, the number of clock cycles required for one frame of image is 1\_050\_000, and the total amount of data in 1 second is 63\_000\_000, that is, only when the clock frequency is at least 63MHZ,  $640 \times 480@60$  video can be scanned. Therefore, the clock period of the video is about 15.87 ns. The scan time used to measure a single frame image is the product of the clock period and the number of clock periods, about 16\_666\_666 ns. The simulation diagram of the video stream is shown in Figure 7. It can be measured from Figure 7 that the scan time of one frame of image is 16\_743\_739 ns and just enough can meet the scanning frequency of 60 HZ.

**4.3. Video Capture Test.** Before video capture simulation, we need to calculate the local clock, that is the capture clock. Generally speaking, the local clock cycle is greater than 1/3 times the video clock cycle. The video clock cycle calculated from the previous section is about 15.87 ns, and then the period of the local clock is less than 47.61 ns. Therefore, for the convenience of observation, the clock period of the system video is 16 ns, and the local clock period is 40 ns. At the same time, the asynchronous FIFO is used to solve the problem of inconsistency between the local clock and the pixel clock. The video capture simulation diagram is shown in Figure 8. The right side of Figure 8 is the image data file

processed by QT5, and the left side of Figure 8 is the data captured by the video. After comparison, it can be seen that the captured data and image data are consistent.

## 5. Analysis of Test Results

This paper selects 4 images for testing, which are the paper image; carve image; watch image; and cup cover image; their size is  $640 \text{ pixel} \times 480 \text{ pixels}$ . The experimental results are shown in Figure 9. The first row of paper images and the second row of carved images in Figure 9 are both affected by uneven lighting. The threshold of the fixed-threshold algorithm is tested continuously, and the final threshold is 128. The global threshold segmentation of the fixed threshold algorithm and the Otsu algorithm is absolute, and a large amount of noise is segmented in the bright part, resulting in blurred images; the target in the dark part is even more unrecognizable. The algorithm in this paper effectively performs threshold segmentation on the images of the bright part and the dark part. It can not only segment the small objects in the paper image clearly and accurately but also effectively segment the outline of the figure in the dark part of the upper right corner of the statue image. The third row of the watch and the fourth row of cup cover in Figure 9 are both reflected by strong light. The threshold of the fixed-threshold algorithm is tested continuously, and the final threshold is 100. The fixed threshold algorithm and the Otsu algorithm lost the digital information of "5" and "6" on the dial of the reflective part in the lower right corner of the watch image and the overall letter information of the cup lid image during the segmentation process. The algorithm solves the interference of the reflection phenomenon and accurately retains the digital information of the dial in the reflective part in the lower right corner of the watch image and the letter information in the cup cover image. Compared with the fixed threshold algorithm, the proposed algorithm does not need to manually set the threshold with experience, but can automatically set the optimal threshold according to the brightness and darkness of the image. Compared with the Otsu algorithm, the proposed algorithm can better deal with the influence of uneven illumination. However, in the carve of Figure 9, we can see that although the local threshold algorithm can recognize the edge information of



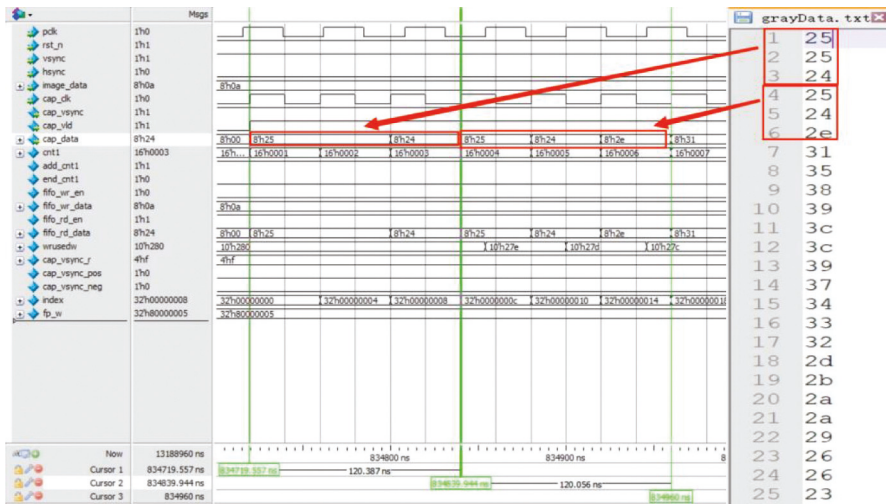


FIGURE 8: Video capture simulation diagram.

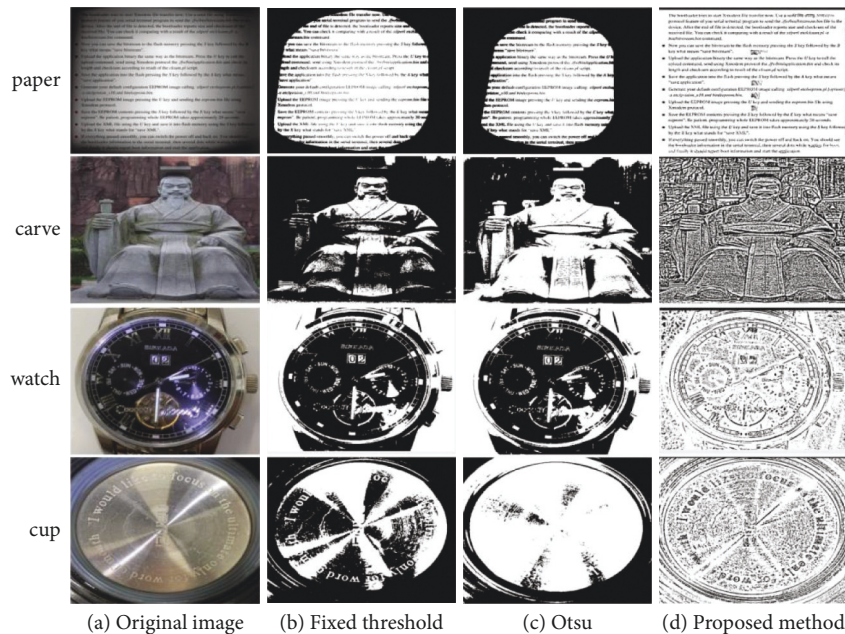


FIGURE 9: Experiment comparison of various algorithms.

the image, there will be a lot of interference. Therefore, the local threshold segmentation has certain limitations for prospective images.

### 6. Conclusion

The main purpose of this paper is to solve the problem of reflection and uneven illumination in the process of image processing. A local threshold segmentation algorithm based on FPGA is proposed. The algorithm adopts the mean value of local neighborhood blocks and the Gaussian weighted sum design idea in the local neighborhood. First, mean filtering is used to remove the noise interference in the image. Then, a local threshold segmentation algorithm is designed on FPGA to solve the interference caused by reflection and uneven illumination.

Finally, verify the designed algorithm through Modelsim simulation software and design a fixed threshold algorithm and Otsu algorithm on VS2015 to test the image. After comparing the experimental results, the algorithm can effectively reduce the image interference caused by uneven illumination and reflection and improve the segmentation effect.

### Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

### Conflicts of Interest

The authors declare no conflicts of interest.

## Acknowledgments

This work was supported by the Jiangsu Graduate Practical Innovation Project (nos. SJCX21\_1517 and SJCX22\_1685), Major Project of Natural Science Research of Jiangsu Province Colleges and Universities (no. 19KJA110002), Natural Science Foundation of China under Grant no. 61673108, and Yancheng Institute of Technology High-Level Talent Research Initiation Project (no. XJR2022001).

## References

- [1] F. J. Iniguez-Lomeli, Y. Bornat, S. Renaud, J. H. Barron-Zambrano, and H Rostro-Gonzalez, "A real-time FPGA-based implementation for detection and sorting of bio-signals," *Neural Computing & Applications*, vol. 33, no. 18, Article ID 12121, 2021.
- [2] S. Ullah, S. Rehman, M. Shafique, and A Kumar, "High-performance accurate and approximate multipliers for FPGA-based hardware accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 2, pp. 211–224, 2022.
- [3] I. Westby, X. Yang, T. Liu, and H Xu, "FPGA acceleration on a multi-layer perceptron neural network for digit recognition," *The Journal of Supercomputing*, vol. 77, no. 12, Article ID 14356, 2021.
- [4] H. Jing and X. Xiaoqiong, "Sports image detection based on FPGA hardware system and particle swarm algorithm," *Microprocessors and Microsystems*, vol. 80, 2021.
- [5] F. Zhang, N. Wang, Z. Hu et al., "A study of UDP and TCP FPGA implementation for data acquisition system," *Journal of Instrumentation*, vol. 16, no. 07, Article ID P07044, 2021.
- [6] D. D. Nguyen, A. El Ouardi, S. Rodriguez, and S Bouaziz, "FPGA implementation of HOOFR bucketing extractor-based real-time embedded SLAM applications," *Journal of Real-Time Image Processing*, vol. 18, no. 3, pp. 525–538, 2021.
- [7] F. Shafiei and S. Fekri-Ershad, "Detection of lung cancer tumor in CT scan images using novel combination of super pixel and active contour algorithms," *Traitement du Signal*, vol. 37, no. 6, pp. 1029–1035, 2020.
- [8] Z. Ye, H. Li, and W. Zha, "A visual detection method of tool damage using local threshold segmentation," *Hsi-An Chiao Tung Ta Hsueh/Journal of Xi'an Jiaotong University*, vol. 55, no. 4, pp. 52–60, 2021.
- [9] M. Han and A. Wang, "Multi-threshold segmentation of remote-sensing image based on genetic algorithm," *International Journal of Earth Sciences and Engineering*, vol. 8, no. 1, pp. 86–91, 2015.
- [10] L.-S. Wu, W. Cheng, and Y. Hu, "Image segmentation of multilevel threshold based on improved cuckoo search algorithm," *Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)*, vol. 51, no. 1, pp. 358–369, 2021.
- [11] X. Zhai, M. Eslami, E. S. Hussein et al., "Real-time automated image segmentation technique for cerebral aneurysm on reconfigurable system-on-chip," *Journal of Computational Science*, vol. 27, pp. 35–45, 2018.
- [12] R. Ratnakumar and S. J. Nanda, "A high speed roller dung beetles clustering algorithm and its architecture for real-time image segmentation," *Applied Intelligence*, vol. 51, no. 7, pp. 4682–4713, 2021.
- [13] S. B. Alagarsamy and S. Kondappan, "Ear recognition system using adaptive approach Runge-Kutta (AARK) threshold segmentation with ANFIS classification," *Neural Computing & Applications*, vol. 32, no. 15, pp. 10995–11006, 2020.
- [14] N. Bo, L. Zhiyuan, C. Xiantao, N. Michele, and W. Shaohua, "Segmentation of ultrasound image sequences by combing a novel deep siamese network with a deformable contour model," *Neural Comput & Application*, Springer, Berlin, Germany, 2022.
- [15] W. Hongyu, Z. Dandan, D. Songtao, G. Zhanyi, F. Jun, and W. Shaohua, "Rib segmentation algorithm for X-ray image based on unpaired sample augmentation and multi-scale network," *Neural Comput & Application*, Springer, Berlin, Germany, 2021.
- [16] Y. Zhao, H. Li, S. Wan et al., "Knowledge-aided convolutional neural network for small organ segmentation," *IEEE Journal of Biomedical and Health Informatics*, vol. 23, no. 4, pp. 1363–1373, 2019.
- [17] M. Li and C. Zou, "Research on threshold image segmentation method based on improved genetic Algorithm," *Software Engineering*, vol. 25, no. 01, pp. 37–40, 2022.