Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 5270887, 14 pages
https://doi.org/10.1155/2022/5270887

Research Article

Design of Hardware Acceleration in Edge Computing Device for Bottle Cap High-Speed Inspection

Kaiyuan Liu, Yiming Liu, Chong Peng, Yiyang Chang, and Yi Zhao

State Key Laboratory of Integrated Optoelectronics, College of Electronic Science and Engineering, Jilin University, Changchun, Jilin Province, China

Correspondence should be addressed to Yi Zhao; yizhao@jlu.edu.cn

Received 27 July 2022; Revised 14 September 2022; Accepted 19 September 2022; Published 6 October 2022

Academic Editor: Kuruva Lakshmanna

High-speed product appearance inspection is crucial for modern automated industrial production such as bottle caps. The main detection method is to capture the image of bottle caps on the high-speed conveyor belt by industrial cameras and send them to the server through edge devices for various analyses. In order to improve production efficiency, it is necessary to increase the inspection rate of bottle caps. However, the transmission rate and huge data throughput of traditional inspection methods limit the inspection rate. Because of the application requirements for improving bottle cap detection efficiency, this paper proposes a hardware acceleration design based on edge devices to improve the bottle cap detection rate significantly. In this paper, an image processing module based on FPGA is designed as the edge device, improving algorithm execution speed through pipeline processing. It realizes the edge detection of the bottle cap and the fast detection of the front or back states and can send the instructions to the actuator to correct it in time when the back bottle cap is detected. It also realizes the positioning of the bottle cap area and cuts the image. Thereby, the amount of data sent to the server is significantly reduced. We have done both functional simulation and hardware implementation. Comparing with the pure software solution, the proposed design reduces the execution time of the algorithm from 16 ms to 3.07 ms, which achieves a more than four times rate increase. The amount of data that needs to be transmitted to the server per second is reduced from 7200 Kb is reduced to 25 Kb, which reduces the transmission capacity and server cache space by more than 280 times compared to the original image. In this paper, the hardware acceleration design of edge devices and the positioning and cropping of original images can greatly reduce the transmission pressure, data calculation pressure, and buffer space requirements of the central server and improve the detection rate. This design can not only be used for bottle cap inspection but also for various machine vision fields, such as defect detection of other workpieces.

1. Introduction

Nowadays, the production of various workpieces is becoming more and more automatic, and the inspection of the appearance quality involves various aspects such as defects, size, and pattern printing quality. This kind of inspection is called mechanical repetition. The early inspection mainly relied on manual observation. But since the naked eye has low accuracy [1, 2], there will inevitably be some mistakes. Besides, with the improvement of production efficiency, the rate and accuracy of manual inspection are far from meeting the requirements [3]. In recent years, machine vision technology has developed rapidly and has shown outstanding performance in appearance detection [4, 5]. High-speed industrial quality inspection systems based on machine vision can integrate the functions of collecting target images, sending collected data, analyzing data, making judgments, executing instructions, etc., which can achieve a high degree of automation. With its advantages of fast inspection speed, high accuracy, and capability to inspect without contact [6, 7], it is getting more and more attention from manufacturers and has gradually replaced manual

inspection to achieve automatic inspection, which is used to distinguish between qualified and defective products with high efficiency and accuracy [8].

Traditional machine vision inspection systems usually consist of industrial cameras, high-speed conveyors, edge equipment, and a central server [9]. Industrial cameras have a short shutter time and fast shooting speed, which can quickly capture objects in motion at high speed, and the number of photos taken per second can reach dozens to hundreds, much higher than ordinary cameras. The industrial cameras also have high-resolution and raw output data, which can be used for high precision image processing, but the data volume is also relatively high. In industrial scenarios, the requirements for real-time are relatively high, and if the data is directly uploaded to the central server, the bandwidth and cache requirements will be higher, and it will also greatly increase the amount of server computing, which will waste valuable computing power. Therefore, edge computing is used [10].

Edge computing is initially defined as a new platform that can provide cloud computing functions for mobile terminals in the vicinity of wireless access networks deployed in the mobile device task computing architecture [11, 12], and with the expansion of usage scenarios, multiple access devices are also classified as edge computing. Edge computing focuses on the shortcomings of high bandwidth requirements and high latency in cloud computing, and edge devices with capabilities of storage, transmission, and specific computing are deployed at the terminal to assume some of the computing tasks of the central server and realize the offloading of computing [13–15].

Of late, Blockchain technologies have enticed significant attention in several research areas. One of them is the Edge of Things (EoT), which is made possible by the amalgamation of edge computing and the Internet of Things (IoT). The unique features of Blockchain such as decentralization, immutability, and traceability are useful to reshape and transform traditional EoT systems with higher levels of security. The convergence of Blockchain and EoT, in particular, results in a new paradigm known as BEoT, which has been hailed as a promising enabler for future services and applications [16, 17].

Bottle cap inspection is an essential and typical application in machine vision inspection [18–20]. In bottle cap machine vision inspection, the image data is captured by the industrial camera and sent to the central server for the image algorithm analysis. The central server receives the image information transmitted from the edge and executes the appropriate algorithm to distinguish the qualified and defective products and detect the positive or negative sides of the bottle caps. Then, the server sends the instructions to the actuators near the production line, and the actuators complete the operations such as turning over the caps or rejecting the unqualified defective products according to the instructions [21, 22].

The main problems with the traditional machine vision inspection system exist in the following aspects: firstly, the edge device based on a pure software solution is not fast enough to process a large amount of image information [23], which will generate delays and cannot send the image information to the central server in time to meet the requirements of the detection speed in nowadays' industrial scenarios. Secondly, the transmission of high-resolution, high-frame image data from the edge devices to the central server requires a large amount of bandwidth. In addition, when multiple pipelines are working on detection at the same time, the number of edge devices is large, and the amount of data that needs to be sent to the central server at the same time is enormous, which will put massive pressure on the server's computation speed and cache storage space.

In this paper, we propose to use FPGA as the edge device of the machine vision bottle cap detection system to implement a hardware-based accelerated design. Because of its parallel and pipelined processing characteristics [24–26], it has a significant speed advantage compared to the pure software solution [27]. We implement the edge detection of bottle caps on an FPGA edge device; it could automatically locate and crop the bottle caps in the image and extract the cap location in the image, significantly reducing the amount of data, the bandwidth pressure on the transmission, and the computational and caching pressure on the server. It can also detect the front or back information of the cap and output the position coordinates of the cap on the conveyor belt, then send the indication signal of whether the cap needs to be turned over directly to the actuator. Therefore, compute offload and hardware acceleration for the central server are achieved in this paper by using FPGAs as edge devices.

The major contributions of this article are to:

(i) Study the various bottle cap inspection approaches comprehensively

(ii) Implement the edge detection of bottle caps on FPGA edge device

(iii) Implement a hardware-based accelerated design for bottle cap inspection

(iv) Perform an experimental analysis based on execution time and data volume

The rest of this paper is organized as follows. Section 2 discusses the recent works related to the machine vision based inspection methods. Section 3 describes our proposed method in detail. The fourth section provides simulation and experimental results. The fifth section gives the related discussion. The sixth section concludes the whole paper.

2. Related Works

This section discusses the research progress related to the bottle cap high-speed inspection. Quality inspectors aim to bridge the gap between manufacturing and assembly in the automobile industry. It ensures that components manufactured are in appropriate form to build a reliable system. However, if inspection is performed by humans, it may result in the incorrect classification of defective components as good or vice versa, compromising quality and wasting

material. Hence, machine-vision-based inspection system is proposed to automate this process. The provision of high-level quality control system based on machine vision in production lines is an imperative issue. It not only enhances the efficiency but also stems the appropriate tools to gather information about technical mistakes and faults in the target system. Prabadevi et al. [18] deliberate the inspection of bottle caps in drink factories using CNN-based algorithms. This work uses VGG-19 as an end-to-end deep learning method to enhance the accuracy.

Govindaraj et al. [2] proposed computer vision and image processing methods, namely, morphology, edge, and contour detection techniques to evaluate the errors cylindrical metallic components effectively. This work reduces the consumption of space and cost of installation since this approach does not evaluate errors stage by stage. This approach does not unite inspection of rotating components, as some errors in small components cannot be detected. Monitoring the assembly process is difficult in manual assembly of mass customization production, where the operator must change the assembly process based on the product. If an assembly error is not immediately detected during a product's assembly process, it may result in errors and loss of time and money during the subsequent assembly process and will affect product quality. Chen et al. [8] proposed two methods, three-dimensional CNN (3D CNN) and fully convolutional network (FCN), to recognize assembly action and recognize parts from complicated assembled products, respectively. This work shows reduction in computational complexity and improvement in training speed.

Smith et al. [5] presents the study of machine vision including historical aspects, perspectives, and future enhancements. This work highlighted the importance of machine vision in manufacturing (e.g., quality inspection), security (e.g., biometric applications), medicine (e.g., detecting diseases), and in agriculture (e.g., plant disease detection). A deep ANN-based approach is expressed for machine vision tasks. Machine vision enhances the efficiency, quality, and reliability of defect detection significantly. Excellent optical illumination platforms and appropriate image acquisition hardware are required for high-quality images in visual inspection. Image processing and analysis are critical technologies for obtaining defect information, and deep learning is having a significant impact on image analysis. Ren et al. [4] systematically discusses a brief history as well as the state of the art in optical illumination, image acquisition, image processing, and image analysis in the field of visual inspection.

Benbarrad et al. [6] devised machine vision model for detecting defect in the casting manufacturing products based on machine learning. This work CNN-based model named EfficientNetB0 to detect defects efficiently. This work makes use of cloud-based system and adopts edge real-time image processing. In this work, the use of an IoT gateway maintains continuous and efficient communication between the various system components that use different protocols. Model training is managed in the cloud, allowing the system to take advantage of the available computing power. Of late, image analysis method based on deep neural networks helps in product quality control. The main advantage of these approaches is fast performance and robustness. Malesa and Rajkiewicz [19] propose lightweight CNN method to perform quality control of polyethylene terephthalate (PET) bottles caps with dedicated image calibration. This work makes use of CustomKSM method for the product quality control, and it significantly reduces training time and maintains the classification accuracy.

Prabuwono et al. [20] presented fuzzy logic-based model to perform automated visual inspection of bottle caps. This work uses three classification methods such as Mamdani, Sugeno, and production rule to perform inspection. Existing works face issues, namely, an illumination and reflection problems which are resolved in this work by using feature selection algorithm and enhance accuracy. Several machine vision system for bolt inspection are implemented on PC-based environment for an embedded platform faces limited computation performance. To enjoy high power-efficiency and lower cost as well as enhanced computation power, Oh et al. [27] propose novel bolt inspection algorithms on an embedded CPU and FPGA using hardware-software (HS) codesign approaches.

Tao et al. [3] use YOLO V3 algorithm to perform surface inspection of the precision instrument. This algorithm is based on obtaining and processing the image presences. Generally, YOLO V3 detection network is used to estimate the location of the damage appearances and recognize the type of damage. Following that, the designed level set algorithm was used to acquire more precise damage locations in the image block by leveraging the characteristics of various types of damages. Frustaci et al. [23] proposed hardware–software (HW–SW) codesign approach to design robust and high-performance machine vision system for automatic quality inspection in assembly processes. The whole system is executed on a Xilinx Zynq heterogeneous system-on-chip. This approach uses the catalytic converter assembly process to demonstrate their research work. This approach achieves 23× speed-up compared to the pure software solution functioning on the Zynq-embedded processing system. The summary of related works is expressed in Table 1.

3. Methods

3.1. Architecture Design. A block diagram of the overall design of the high-speed inspection system for bottle caps in this paper is shown in Figure 1. The system consists of a transmission device, an item to be detected, an industrial camera, an actuator, an FPGA-based edge device, and a central server at the remote end. After the industrial camera acquires the picture information and passes it to FPGA for edge calculation, the FPGA completes the edge detection and the positioning and cropping of the detected items and outputs the front or back data to the central processor actuator, respectively. After the calculation, the central processor then transmits the instructions back to the actuator. In this paper, we have completed the FPGA design, simulation, and practical implement.

Table 1: Summary of various state-of-the-art approaches.

Approach	Author	Year	Description	Pros	Cons
Automatic vision-based inspection system in an olive oil bottling line	Abdelhedi et al. [1]	2012	This work uses thresholding algorithms such as global thresholding and adaptive thresholding for inspecting an oil bottling line.	It offers accurate feature extraction and on-line capabilities, considering robustness and low processing time.	This work faces scalability issues, and processing time is not convincing.
Inspection of bottle caps using CNN	Prabadevi et al. [18]	2019	This work adopts VGG-19 method to inspect bottle caps in drink factories.	It uses strong fine-tuned classifier based on the optimization of hyperparameters to improve accuracy.	The hardware implementation of this work is not proposed. This work may lead to poor scalability.
Automated vision inspection system for cylindrical metallic components	Govindaraj et al. [2]	2019	This work makes use of vision and image processing methods such as morphology, edge and contour detection techniques to assess the errors effectually.	This method is capable of detecting the presence of outer-diameter burrs. It can evaluate errors in cylindrical components of minimum size.	The number of samples tested for evaluating the performance is low.
Inspection of bottle caps using fuzzy logic	Prabuwono et al. [20]	2019	This work proposes automated visual inspection system (AVIS) using fuzzy logic-based classifiers. Three types of classification such as Mamdani, Sugeno, and production rule are used.	An illumination and reflection problem faced in bottle cap inspection is resolved and maintains acceptable accuracy.	The number of samples tested for evaluating the performance is low.
FPGA acceleration of bolt inspection algorithm	Oh et al. [27]	2019	This approach discusses about an image processing algorithm for bolt defect detection implemented on an embedded CPU.	When compared to the software-only embedded CPU platform, the proposed system could accelerate calculation speeds by up to 18.9 times and the overall system by 1.3 times.	The number of samples tested for evaluating the performance is low.
Deep learning methods to monitor assembly process	Chen et al. [8]	2020	This work devises two methods such as 3D CNN and FCN to recognize assembly action and recognize parts from complicated assembled products, respectively.	3D CNN with batch normalization reduces computational complexity, improves training speed, and maintains accuracy.	Some important studies, such as estimating the pose of each part in a product and determining whether each part is in the correct place, are not addressed.
Study on machine vision	Smith et al. [5]	2021	This work presents a state-of-the-art survey, including historical review, perspectives, and future directions in machine vision.	The significance of machine vision in manufacturing, security, and in medicine are addressed. It addresses how deep ANN used in machine vision.	Challenges faced in the design and development of CNN-based solutions are not addressed.
Approaches for defect detection based on machine vision	Ren et al. [4]	2021	This study systematically discusses a brief history as well as the state of the art in optical illumination, image acquisition, image processing, and analysis in the field of visual inspection.	The recent progresses in industrial defect detection based on machine vision are introduced.	Robust approach which balances efficiency and precision is not addressed. Machine vision based approaches faces scalability issues.
Machine learning-(ML-) based defective product inspection	Benbarrad et al. [6]	2021	Convolution neural network- (CNN-) based approach is proposed for detecting the defects in the casting manufacturing products.	This work amalgamates the classification model for product inspection and a regression model for processes prediction. Model training is managed in the cloud, allowing the system to take advantage of the available computing power.	This work does not address the overfitting and data imbalance issues associated with ML algorithms.
		2021			

Table 1: Continued.

Approach	Author	Year	Description	Pros	Cons
Quality control of PET caps using deep neural networks	Malesa and Rajkiewicz [19]		This work adopts CustomKSM method for the product quality control of PET bottle caps.	It is a very lightweight network architecture, and it significantly reduces the training time.	The hardware implementation of this work is not proposed. This work does not address the overfitting and data imbalance issues.
Machine vision and robotics technologies in assisting quality inspection and testing	Tao et al. [3]	2022	This research utilizes YOLO V3 algorithm to perform surface inspection of the precision instrument.	A deep architecture capable of conducting damage detection intelligently. The proposed method demonstrated a high tolerance for unknown types of damage as well as exceptional flexibility and adaptability.	The hardware implementation of this work is not proposed.
Automatic quality inspection in assembly processes	Frustaci et al. [23]	2022	This paper addresses the issue of automatic quality inspection in assembly processes by delving into the creation of a computer vision system using a heterogeneous multiprocessor system-on-chip.	The proposed system attains a 23× speed-up compared to the pure software solution running on the Zynq-embedded processing system.	The proposed system is not addressed the implementation of more complex image processing routines.

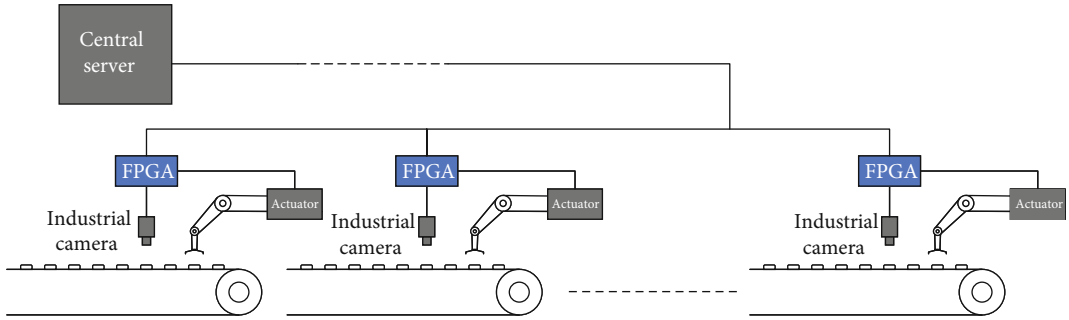

Figure 1: High-speed bottle cap detection system based on FPGA edge computing.

3.2. Workflow. The whole process is divided into three stages: data acquisition, FPGA processing, and postprocessing. The camera captures the data and stores it in SDRAM, after which it is read into the image processing part of the FPGA. Then, it goes through the RGB to grayscale module, a median filter module, and the Sobel edge detection module and cache in SDRAM. The processed data also goes to the position detection module, after which it is read out and cut by the cut control module. After processing by FPGA, the front or back data, the coordinates of the items in the image, and the cropped image data are transmitted to the host PC for postprocessing, and the host PC converts the data to image via Python. A simplified block diagram of the workflow of this experience is shown in Figure 2.

3.3. Grayscale Conversion Module. This module is used to convert the 24-bit RGB data to grayscale format, and the grayscale is converted by

$$Y = R * 0.299 + G * 0.587 + B * 0.114, \tag{1}$$

where R, G, and B is the red, green, and blue channel values for each pixel point of the 24-bit RGB format image, while R corresponds to the first eight bits of the input data, G to the middle eight bits, and B to the last eight bits. Y is the component of the YUV format, which means grayscale. Because the calculation process of the formula includes a floating-point calculation, which FPGA is not good at, then we expand the floating-point number in the formula by 256 times for calculation, that is, the form of binary left shift by eight bits, after participating in the calculation to take the first ten bits of the 18-bit result and determine whether the high two bits are 0 if not means that the calculation result is beyond 0xFF, the output result is 0xFF, else then the output of the last eight bits as grayscale. The main function flow of the module is shown in Figure 3.

3.4. Median Filter Module. Median filtering is a nonlinear sorted smoothing filter that can effectively remove noise from images. We use a $3 * 3$ convolution kernel to implement fast median filtering in FPGA. Because the image data stream is a serial input, nine columns of parallel data are required to perform the convolution operation. Therefore, we use three sets of shift registers with the same width as the image to output the data in parallel. As shown in Figure 4(a), the serial data is input from the first position of the first row and then shifted to the right in order. After moving to the rightmost of each row, the next time, it will move to the first position of the next row and output when moving to the last position. The last position data of each row is also read out to realize the simultaneous parallel call of data of three pixels in the same column of the image. Therefore, if the pixel data of the whole frame is input in sequence, the convolution operation of the whole image is realized, and the simulation waveform of the module function to generate a $3 * 3$ matrix is shown in Figure 4(b).

The specific operation of median filtering is to design and instantiate a sorting module in the median filtering module. The sorting process is shown in Figure 4(c), the three grey values of each row are sorted in descending order in a $3 * 3$ matrix around each pixel, and then, the grey values of each column are sorted in descending order. The minimum value in the first column, the median value in the second column, and the maximum value in the third column are sorted to take the median of these three values, which is the median of the matrix as the grey value of that point after median filtering.

3.5. Sobel Edge Module. This paper uses the Sobel algorithm for edge detection of the grayscale image. The size of the convolution kernel is $3 * 3$. The Sobel algorithm consists of two convolution kernels. The Sobel algorithm consists of two convolution kernels corresponding to the horizontal and vertical directions of the original image for two-dimensional convolution to obtain the gradient of two directions and then have an average weight summed. The formula is

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A, \tag{2}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A, \tag{3}$$

$$G = \sqrt{G_x^2 + G_y^2}, \tag{4}$$

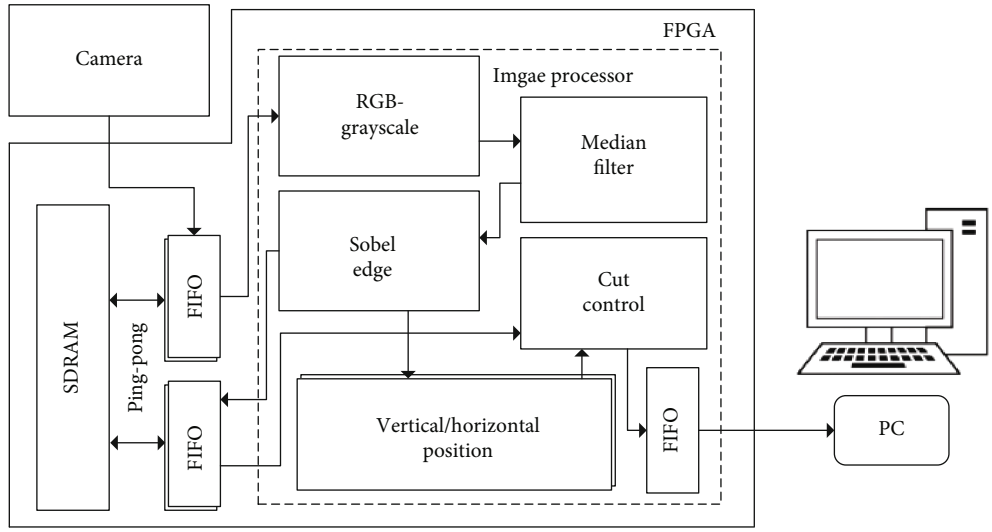

Figure 2: The simplified block diagram of the workflow.

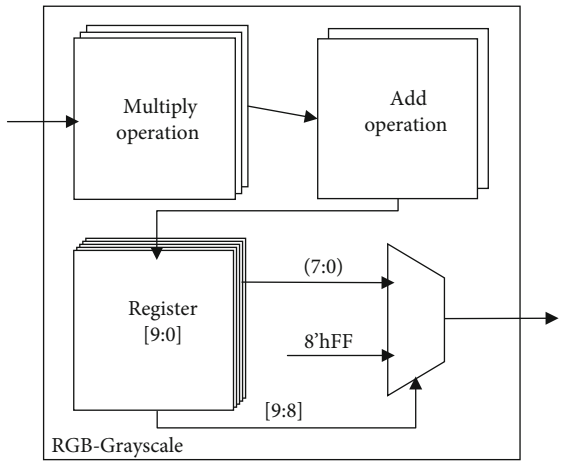

Figure 3: Flowchart of the main functions of the grayscale conversion module.

where G_x and G_y are the convolution results in horizontal and vertical directions, respectively, A is the $3 * 3$ matrix of the area around the convolution target pixel, and G is the average weight summation result. Then, it judges the relationship between the result and the preset threshold value, which is greater than the threshold value output high level. It means that there is a boundary there. Less than the threshold value output low level means no boundary. The output of the module is a 1-bit binary image. Therefore, the module first applies the method mentioned in 2.4 to convert the serial data into a parallel matrix and then performs the convolution of two Sobel convolution kernels. Then, it is needed to instantiate a squared module to calculate formula (4). The central functional flow diagram of the whole Sobel module is shown in Figure 5.

3.6. Position Detection Module. In order to determine the position of the target item in the image, we designed vertical and horizontal position detection modules. For example, in the vertical position detection module, two counters are set to traverse the entire diagram image, with the x-counter indicating the row coordinates of pixels and the y-counter indicating the column coordinates, and we set a register with the same depth as the width of the image. When the pixel data corresponding to the counter goes high level, the enabled signal is pulled high and reads the value stored in the register corresponding to the address of the x-counter, plus one, and writes it back to the original position; the address at the time of writing is the current address delay one clock cycle. When the image is transmitted to the last row, the number of boundary pixels in each column is stored in the register. Extracting the position where the boundary appears for the first time and the position where the boundary appears for the last time, the vertical position of the target location is determined. The main functional flow diagram of this module is shown in Figure 6.

3.7. Cut Control Module. The data processed by the Sobel module is written into SDRAM in ping-pong mode, which means that the data of adjacent frames is written alternately to the addresses of two areas of SDRAM. When the position determination module gets the position information, the data of the adjacent frames is read out alternately in this module in ping-pong mode, outputting the pixels of the address located within the position of the cap in turn. At the same time, we set a counter that increases when a border pixel appears and compare it with the preset threshold value after cropping is finished. Since there are more pattern boundaries on the front side of the bottle cap and fewer pattern boundaries on the backside, the counter value is greater than the set threshold for the front of the bottle cap and less than the set threshold for the back of the bottle cap.

4. Experiment and Analysis

We performed a functional simulation of the algorithm flow and analyzed the functions in detail, then we downloaded the design to FPGA and got the results.

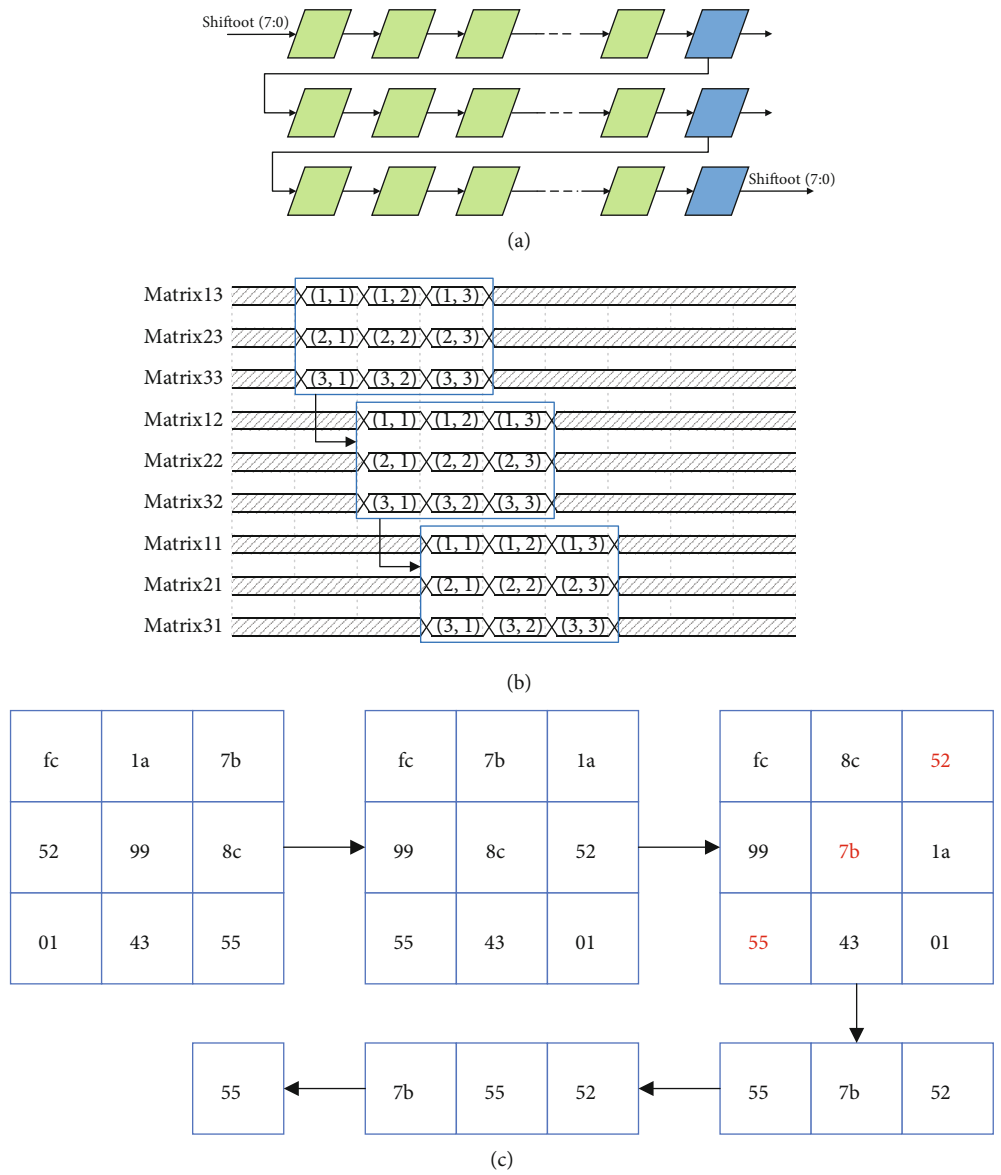

Figure 4: Principle of median filtering module algorithm. (a) Schematic diagram of shift register operation, the number of registers in each row is equal to the image width. (b) Waveform diagram, the marked parts in the figure are the same matrix. The middle clock period allows for the use of all nine matrix elements. (c) Median filter module sorting flow, the gray values in the example are all 8-bit wide hexadecimal.

4.1. Image Generation. The images for the simulation are preprocessed by Python, and this module is used in the simulation to capture the image data after preprocessing, and after this module, the picture information is turned into electrical signal output. The output timing is also imitated, as Figure 7 shows the output timing of one line, including the vertical synchronization signal and horizontal synchronizing signal.

4.2. Grayscale Conversion. The before-and-after comparison of grayscale conversion is shown in Figure 8(a), and a few of the pixel points are selected to show the key signal waveforms of the module. As shown in Figure 8(b), the data of three channels are converted to 18-bit results after two cycles, and 10-bit results are obtained after another clock cycle, and then, the grayscale values are output by combined logic. The parameter in the formula has three decimal places and multiply by 256 times then take the integer part for calculation, so the grayscale conversion accuracy is 99.9%.

4.3. Median Filter. This module is aimed at removing the noise from the captured image, first by the grayscale conversion module to convert the image to a grayscale image after pipeline into this module. The noise of the original image is not obvious. Therefore, we artificially added impulse noise and then simulated it to make the comparison. Impulse noise is a typical type of noise that may be generated by interference with the video signal or bit transmission errors and manifests as the random pixel in the image turning black or white. We tested from without adding impulse noise to adding 50% degree of noise, replacing 50% of random pixel points with black or white.

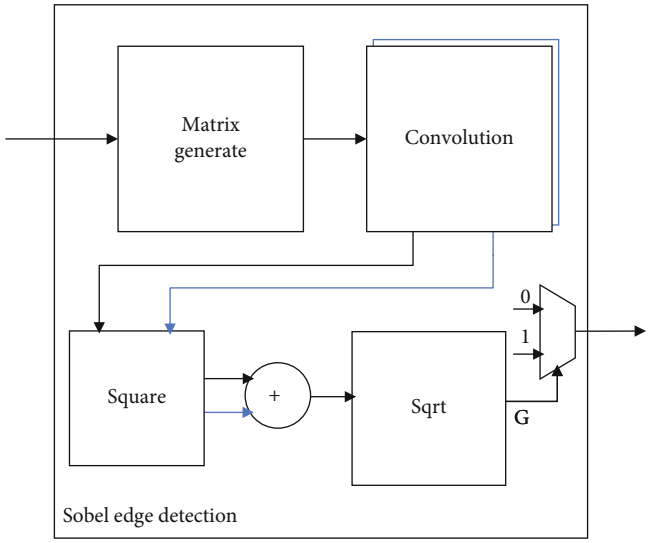

Figure 5: Flowchart of the main functions of the Sobel edge detection module.

A similar relationship between the median filtered image and the grayscale map of the original image for different impulse noise scale images is shown in Figure 9(a). When the noise ratio is less than 21%, the similarity is more significant than 0.95, and the image is clear and has a high degree of detail retention. When the noise ratio is less than 16%, the similarity reaches above 0.98, and when the noise ratio is greater than 21, the similarity is less than 0.95, and there will be a noticeable loss of details or unclear edges.

We filtered the images with 5%, 15%, and 25% of noise and the original image to obtain their grayscale histograms and put them together with misalignment as in Figure 9(b). The peak on the left side of the grayscale histogram corresponds to the background area of the image, and the peak on the right side corresponds to the bottle cap area. As can be seen in Figure 9(b), when the ratio of impulse noise is 5% and 15%, the amount of grayscale values of 0 and 255 is almost nonexistent, and there is almost no effect on the details of the original image, indicating that this median filtering module can effectively remove the noise in the image. Moreover, when the noise level is set to 25%, both peaks show the loss of detail to some extent, and the number of grey values between 0 and 255 is significantly higher. Therefore, this median filter module has an excellent filtering effect for impulse noise of less than 21%; the filtering effect of the module is shown in Figure 10.

4.4. Sobel Edge Detection. We use the Sobel algorithm for edge detection, which requires setting a suitable threshold value. If the threshold value is too small, it will lead to an unclear boundary of the obtained image, and if the threshold value is too large, it will lead to some valid information being filtered out. Therefore, we obtained Ground Truth by manual segmentation, and the results of the similarity analysis between the images obtained with different thresholds and Ground Truth are shown in Figure 11(a). The correlation is the highest when the threshold is set at 125. Therefore, 125 is considered to be the appropriate threshold for this detection. Figure 11(b) shows the original image and the images before and after the Sobel edge detection module.

4.5. Position Detection. We performed a functional simulation of the position detection module, and the key signal waveform diagrams are shown in Figure 12(a). Both the vertical and horizontal synchronization signal are at the falling edge, indicating that the transmission of the whole frame image is completed, and the position signal is updated at the rising edge of the next clock, indicating that the position signal is configured. The simulation waveform diagram of the register used to determine the boundary corresponds to the edge binary image, as shown in Figure 12(b).

4.6. Cut Control and Side Detection. After this module, the image is cropped, and the front or back judgement signal is output, the module reads the image data of the specified range from SDRAM in ping-pong mode with the address of the horizontal and vertical counter, and the waveform of the counter function simulation is shown in Figure 13(a). The position signal can be seen that the coordinate of the upper boundary is 202, the coordinate of the left boundary is 96, and the coordinate of the right boundary is 262. The horizontal direction counter at the position marked in the figure is at the right boundary of the image, and on the next clock cycle, the horizontal direction counter returns to 96, which is the left boundary, while the vertical direction counter jumps to 203, which means it moves down one line.

Figure 13(b) is the simulation waveform of the front or back judgement signal. It can be seen that the value of the front or back counter (Side_cnt) remains unchanged when the output data is low and increases when it is high. When the counter reaches the preset threshold of 3000 for this experiment, the judgement signal is pulled high, and the final output is high.

4.7. Hardware Implementation. We implement the algorithm flow on FPGA, send the data processed to the upper computer, and restore the obtained image data and related information by postprocessing through Python, then we got the position coordinates of the object, the front or back information, and the binary image. We selected different positions, patterns, sizes, shapes, and sides of the bottle caps and different colors of the conveyor belt for the simulation and showed the final results separately in Figure 14, in which the positive direction of the x-coordinate is from left to right, and the positive direction of the y-coordinate is from top to bottom.

5. Discussion

5.1. Logic Utilization Evaluation. We used Quartus II 13.1 to comply the whole design code and downloaded to Altera Cyclone IV FPGA (EP4CE10F7C8), and the logic utilization of the design is presented in Table 2. As it can be seen, the

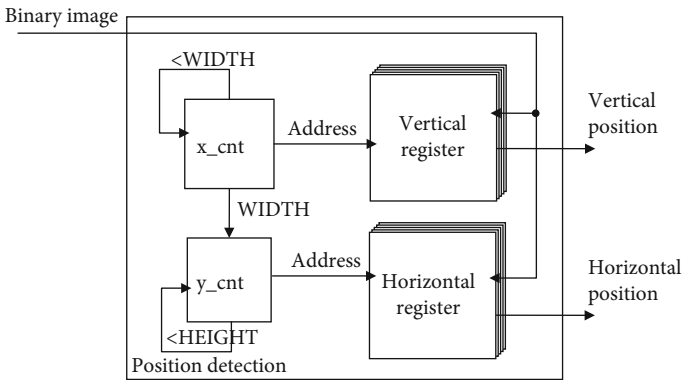

Figure 6: Flowchart of the main functions of the position detection module, the data input is binary image data, and the data output is the object position information.

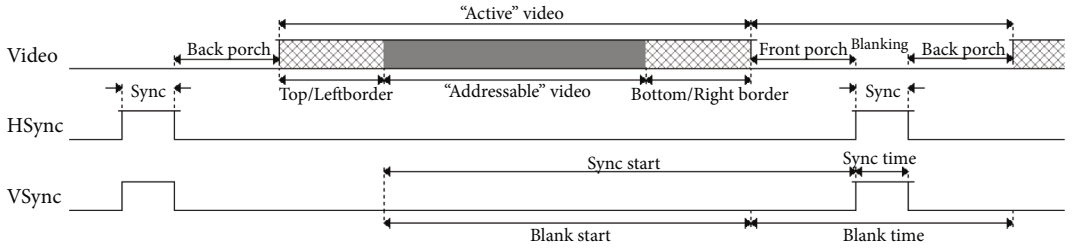

Figure 7: Waveform diagram of output by image generation module. The four output signals marked are the vertical sync signal, horizontal sync signal, image data, and enable signal.

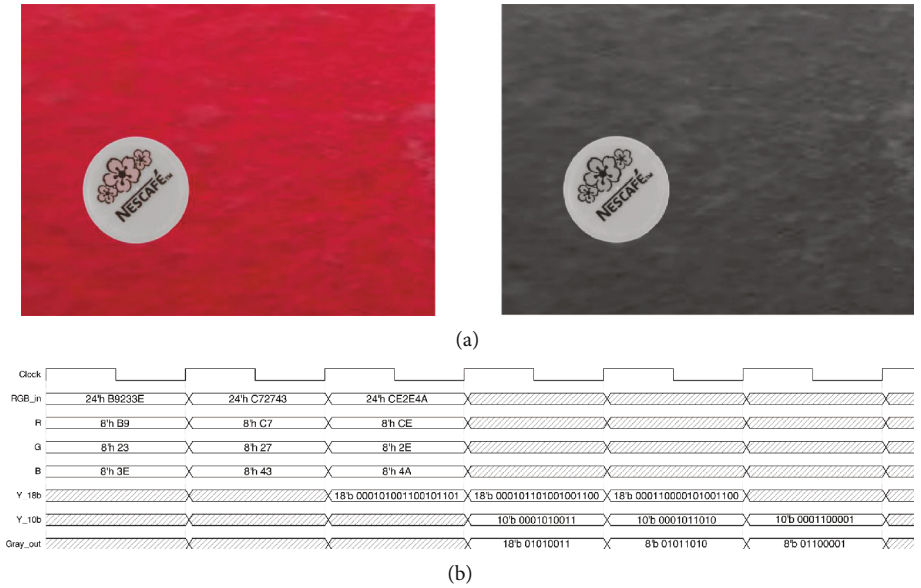

Figure 8: Grayscale conversion module results schematic. (a) Before-and-after comparison image of grayscale conversion processing by this module. (b) The simulation waveform diagram of the important signals. The top four signals in the figure are the correspondence between the three channels of RGB and the number of input data bits, and the three signals below are the 18-bit results after the formula calculation, the results of taking the high decimal, and the final output grayscale results.

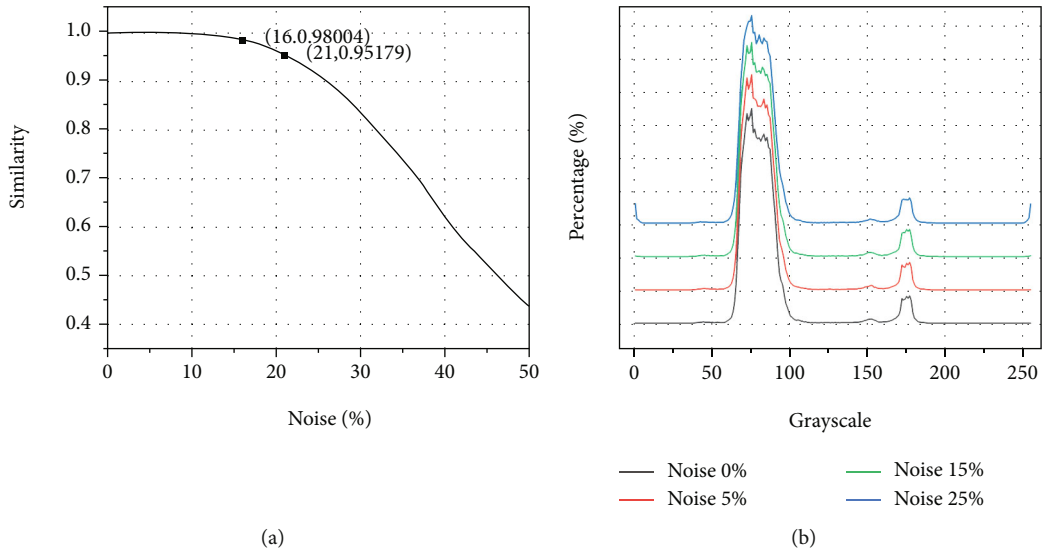

Figure 9: Analysis of the filtering capacity of the median filter module. (a) After adding different noise ratios, the similarity between the filtered image and the original image. (b) Grayscale histograms of images after filtering with different noise ratios.

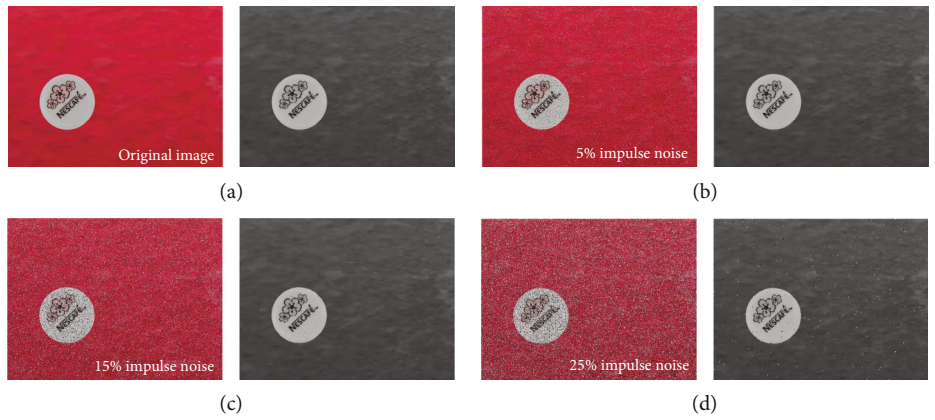

Figure 10: The filtering effect of adding different percentages of noise. (a) Original image. (b) 5% ratio of impulse noise. (c) 15% ratio of impulse noise. (d) 25% ratio of impulse noise.

Figure 11: Sobel edge module results schematic. (a) The similarity between the images obtained with different thresholds and Ground Truth. (b) The original image and the image before and after the Sobel edge detection module.

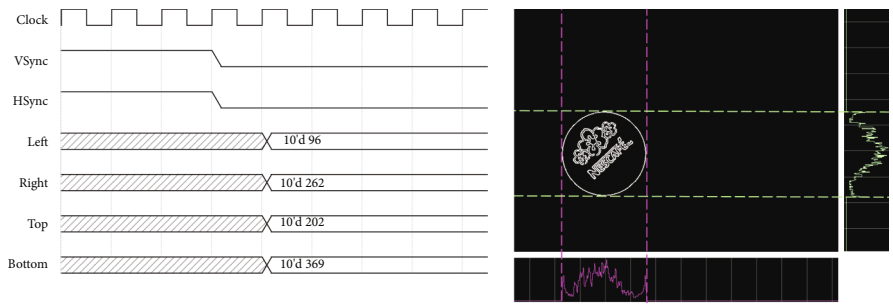

Figure 12: Position detection module results schematic. (a) Waveform diagram of the position signal. (b) Correspondence between register and binary image. The depth of the register below is the same as the image width, and the depth of the register on the right is the same as the image height.

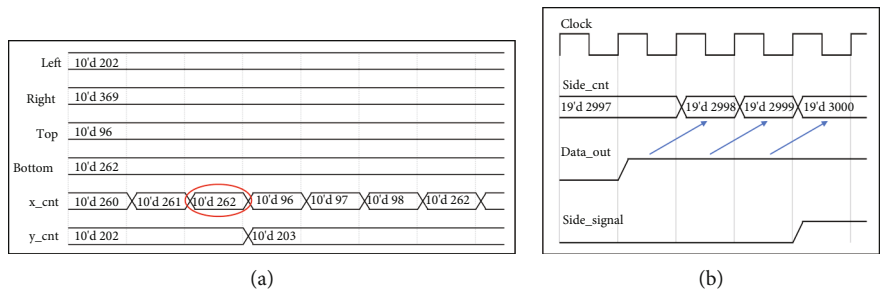

Figure 13: Timing diagrams of the cut module. (a) Horizontal and vertical counters. (b) Front or back judgment signals.

Figure 14: Comparison of the results of selecting different caps, conveyor belts, and sharpness.

Table 2: Logic utilization.

Resource	Used	Total available	Utilization
LUTs	2348	10320	23%
FFs	1415	10320	14%
Block memory bits	119808	423936	28%

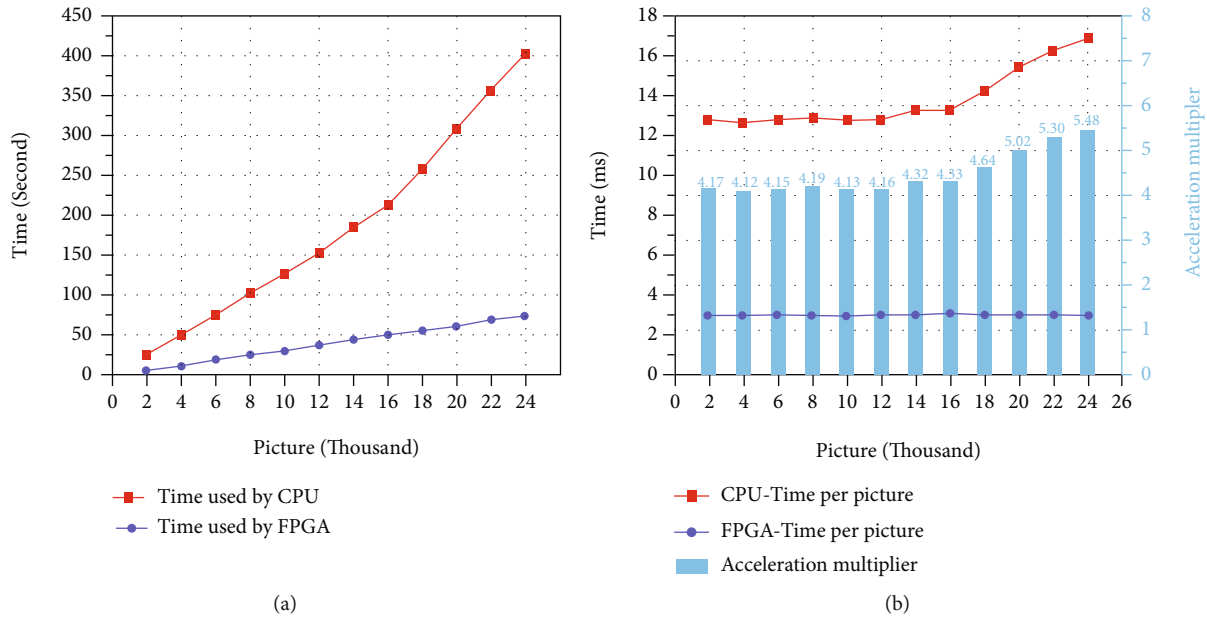

Figure 15: FPGA acceleration effect. (a) Time spent by CPU and FPGA to process the same number of images. (b) The time spent by the CPU and FPGA to process a single photo and the FPGA's acceleration multiplier.

hardware stack consumes 2348 LUTs and 1415 FFs, representing 23% and 14% of all the logic resources, respectively.

5.2. Computational Efficiency Analysis. The FPGA performs image processing in a pipelined manner, so the primary source of latency generation is the timing constraint delay generated by the timing logic calculation of each module and the timing of the acquired data. We compare the FPGA pipeline processing speed with the CPU, the clock frequency of the FPGA is 100 MHz, and the CPU is AMD Ryzen 7 5800H, with a main frequency of 3.20 GHz.

The FPGA acceleration effect is shown in Figure 15. The comparison of the time used by the FPGA and CPU to process the same number of images is shown in Figure 15(a). The software algorithm here uses the highly integrated OpenCV software library, so it is inherently faster. However, as shown in Figure 15(b), we still achieve more than four times the speedup using the FPGA as the hardware edge device, and the average execution time of the algorithm is reduced from 16 ms to 3.07 ms. Furthermore, it is clear that as the number of images increases, the CPU processing speed slows, whereas the FPGA maintains a linear trend of increasing processing time due to the pipeline processing method. The acceleration effect increases significantly when the number of images reaches 16,000 or more; it can achieve more than 5 times the acceleration multiplier when the number of images exceeds 20,000.

5.3. Data Volume Analysis. This design significantly reduces the amount of data required to be transmitted and the required capacity by locating the object on the image and sending the binarized image that retains only the boundary details of the effective area. The original image in this paper is 24 bit, the resolution is 640 ∗ 480, and the data capacity of each image is 7372800 bits or 7200 Kb. Despite the fact that the effective area is only 160 ∗ 160, the final data volume to be transmitted to the server is only 25600 bits or 25 Kb. This design reduces the volume and capacity of data transmission by more than 280 times.

6. Conclusion

In this paper, we designed a hardware image processing edge device with FPGA, and implemented fast edge detection, position detection, and front or back state of bottle caps on the edge device, and implemented area image cropping, which greatly reduces the transmission pressure, server cache, and computation pressure in industrial machine vision inspection. However, this paper does not make further detection of bottle cap shape and defects

yet, and this part still puts pressure on the processor. In the future, it is planned to refine this part by allocating more computation to edge devices to reduce the pressure on the central server.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no competing interest.

References

[1] S. Abdelhedi, K. Taouil, and B. Hadjkacem, "Design of automatic vision-based inspection system for monitoring in an olive oil bottling line," *International Journal of Computer Applications*, vol. 51, no. 21, pp. 39–46, 2012.

[2] K. Govindaraj, B. Vaidya, A. Sharma, and T. Shreekanth, *Automated Vision Inspection System for Cylindrical Metallic Components: IC3 2018[M]*, 2019.

[3] J. Tao, Q. Chen, J. Xu, H. Zhao, and S. Song, "Utilization of both machine vision and robotics technologies in assisting quality inspection and testing," *Mathematical Problems in Engineering*, vol. 2022, Article ID 7547801, 9 pages, 2022.

[4] Z. Ren, F. Fang, N. Yan, and Y. Wu, "State of the art in defect detection based on machine vision," *International Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 9, no. 2, pp. 661–691, 2022.

[5] M. L. Smith, L. N. Smith, and M. F. Hansen, "The quiet revolution in machine vision - a state-of-the-art survey paper, including historical review, perspectives, and future directions," *Computers in Industry*, vol. 130, article 103472, 2021.

[6] T. Benbarrad, M. Salhaoui, S. B. Kenitar, and M. Arioua, "Intelligent machine vision model for defective product inspection based on machine learning," *Journal of Sensor and Actuator Networks*, vol. 10, no. 1, p. 7, 2021.

[7] W. He, Z. Jiang, W. Ming, G. Zhang, J. Yuan, and L. Yin, "A critical review for machining positioning based on computer vision," *Measurement*, vol. 184, p. 109973, 2021.

[8] C. Chen, C. Zhang, T. Wang et al., "Monitoring of assembly process using deep learning technology," *Sensors*, vol. 20, no. 15, p. 4208, 2020.

[9] R. Kamal, A. Jain, and M. V. Deshpande, "Architectural design for inspection of machine objects using small DNNs as tiny ML for machine vision of defects and faults in the manufacturing processes," in *Smart Systems: Innovations in Computing*, Smart Innovation, Systems and Technologies, Springer, Singapore, 2022.

[10] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: the confluence of edge computing and artificial intelligence," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7457–7469, 2020.

[11] Z. Fan, W. Yang, and K. Tian, "An edge computing service model based on information-centric networking," in *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, Tianjin, China, 2019.

[12] W. Yu, F. Liang, X. He et al., "A survey on the edge computing for the internet of things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[13] N. Kim, Y. Lee, C. Lee, T. V. Nguyen, and S. Cho, "GPU-specific task offloading in the mobile edge computing network," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju, Korea (South), 2020.

[14] S. Chen, Q. Li, M. Zhou, and A. Abusorrah, "Recent advances in collaborative scheduling of computing tasks in an edge computing paradigm," *Sensors*, vol. 21, no. 3, p. 779, 2021.

[15] H. Ning, Y. Li, F. Shi, and L. T. Yang, "Heterogeneous edge computing open platforms and tools for internet of things," *Future Generation Computer Systems*, vol. 106, pp. 67–76, 2020.

[16] M. Bahaghighat, F. Abedini, M. S'hoyan, and A. J. Molnar, "Vision inspection of bottle caps in drink factories using convolutional neural networks," in *2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, Romania, 2019.

[17] T. R. Gadekallu, Q.-V. Pham, D. C. Nguyen et al., "Blockchain for edge of things: applications, opportunities, and challenges," *IEEE Internet of Things*, vol. 9, no. 2, pp. 964–988, 2022.

[18] B. Prabadevi, N. Deepa, Q. V. Pham et al., "Toward Blockchain for edge-of-things: a new paradigm, opportunities, and future directions," *IEEE Internet of Things Magazine*, vol. 4, no. 2, pp. 102–108, 2021.

[19] M. Malesa and P. Rajkiewicz, "Quality control of PET bottles caps with dedicated image calibration and deep neural networks," *Sensors*, vol. 21, no. 2, p. 501, 2021.

[20] A. S. Prabuwono, W. Usino, L. Yazdi et al., "Automated visual inspection for bottle caps using fuzzy logic," *TEM Journal*, vol. 8, no. 1, pp. 107–112, 2019.

[21] P. Horputra, R. Phrajonthong, and P. Kaewprapha, "Deep learning-based bottle caps inspection in beverage manufacturing and packaging process," in *2021 9th International Electrical Engineering Congress (iEECON)*, Pattaya, Thailand, 2021.

[22] R. Kulkarni, S. Kulkarni, S. Dabhane, N. Lele, and R. S. Paswan, "An automated computer vision based system for bottle cap fitting inspection," in *2019 Twelfth International Conference on Contemporary Computing (IC3)*, Noida, India, 2019.

[23] F. Frustaci, F. Spagnolo, S. Perri, G. Cocorullo, and P. Corsonello, "Robust and high-performance machine vision system for automatic quality inspection in assembly processeS," *Sensors*, vol. 22, no. 8, p. 2839, 2022.

[24] L. Alzubaidi, J. Zhang, A. J. Humaidi et al., "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, p. 53, 2021.

[25] A. Shawahna, S. M. Sait, and A. El-Maleh, "FPGA-based accelerators of deep learning networks for learning and classification: a review," *IEEE Access*, vol. 7, pp. 7823–7859, 2019.

[26] Y. Liu and X. Wu, "Parallel and pipelining design of SLAM feature detection algorithm in hardware," in *2021 IEEE 16th conference on industrial electronics and applications (ICIEA)*, Chengdu, China, 2021.

[27] S. T. Oh, J. H. You, and Y. K. Kim, "FPGA acceleration of gear inspection algorithm for high speed embedded machine vision," *Journal of Institute of Control*, vol. 25, no. 8, pp. 665–670, 2019.