IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 24, 2020, accepted March 9, 2020, date of publication March 16, 2020, date of current version March 25, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2980947

A New Real-Time Lucky Imaging Algorithm and Its Implementation Techniques

JINLIANG WANG[1], **BINHUA LI**[1,2], **AND KAI XING**[1]
[1]Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, China
[2]Key Laboratory of Applications of Computer Technologies of the Yunnan Province, Kunming University of Science and Technology, Kunming 650500, China

Corresponding author: Binhua Li (lbh@kust.edu.cn)

This work is supported by the National Natural Science Foundation of China (Grant No. 11673009).

ABSTRACT Lucky imaging is a high-angular resolution astronomical image reconstruction technique that can effectively reduce the impact of atmospheric turbulence on image quality and improve the imaging resolution of ground-based telescopes. Its key steps include image selection, registration and superposition. However, the lucky imaging algorithms based on a central processing unit (CPU) encounter difficulty accomplishing real-time processing; thus, they are post-processing methods and cannot meet the needs of on-site observers. Taking advantage of the parallelism and flexibility of the field programmable gate array (FPGA), this paper presents a new real-time lucky imaging algorithm that features real-time processing and dynamic updating and displaying. The algorithm consists of a dynamic sorting-free image selection algorithm, an improved registration and storage method, a parallel superposition algorithm, a parallel preprocessing method for noise suppression and cosmic ray removal, and a dynamic multithreshold display scheme. The simulation results show that the algorithm is feasible, effective and efficient. Compared with other lucky imaging algorithms based on FPGAs, this algorithm shows great advantages in clock consumption and on-chip resource consumption. Furthermore, it can be implemented on a small or medium-size development board of an FPGA. Moreover, the implemented FPGA system can perform real-time and dynamic lucky imaging for more than 10,000 frames of short-exposure images with an original format of 512×512 pixels continuously. The experimental results not only show the validity of the proposed algorithm but also demonstrate the feasibility of the proposed implementation techniques for the FPGA-based algorithm.

INDEX TERMS Astronomical observation, data reduction, FPGA, lucky imaging, real-time processing.

I. INTRODUCTION

Atmospheric turbulence is an irregular and random movement in the atmosphere. Physical properties such as pressure, velocity and temperature at each point of the turbulence fluctuate randomly, making the distribution of the atmospheric refractive index non-uniform. After the starlight is transmitted through the atmosphere, its wavefront is distorted, causing star twinkling and swaying and, more seriously, starlight spreading, which makes the star a blob rather than a point. Finally, a long-exposure image of the observed star becomes a Gaussian image with large full width at half-maximum (FWHM), while the continuously short-exposure images of the star are composed of speckles of different shapes with random changes that have their center within a certain range.

The associate editor coordinating the review of this manuscript and approving it for publication was Cihun-Siyong Gong.

The atmospheric turbulence makes the actual resolution of the telescope much lower than the diffraction limited resolution of the telescope, which is a major factor limiting the spatial resolution of the ground-based medium-sized and large optical telescope [1], [2].

To obtain star images with resolution close to the diffraction limit of ground-based telescopes, several active and passive techniques are employed, i.e., sophisticated and expensive adaptive optics (AO) [3] and low-cost image reconstruction techniques. Lucky imaging (LI) is a simple and effective astronomical image reconstruction technique for reducing or even removing effects of the atmospheric turbulence [4], [5]. The classical lucky imaging technique based on a CPU is an image post-processing method. An electron multiplying charge-coupled devices (EMCCD) camera [6], [7] is usually used to capture a large number of short-exposure astronomical images, and a computer is used to save these

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

images. After the observation, all the images are processed by a computer equipped with a software package of the lucky imaging algorithm, and finally a high-angular resolution astronomical image can be obtained. The classical lucky imaging algorithm consists of three basic processes, namely, image selection, registration and superposition. By analyzing the high-resolution images, astronomers can discover new dim celestial bodies, calculate their precise positions and determine their orbital elements [8], [9].

It is effective to use CPU-based platforms to study the lucky imaging algorithm. Due to the serial processing characteristics of a CPU, its computing power is limited, so the lucky imaging systems based on CPUs suffer from problems of long running time and no real-time performance. During the short-exposure observations using EMCCD, due to the atmospheric turbulence effect, observers do not know much about the real-time states (i.e., position, shape and brightness) of the celestial bodies in the captured images, especially the information of those dimmer celestial bodies. This makes it difficult for observers to detect and correct possible deviations or errors in the observation in time, which may result in invalid observational data and waste valuable observation time. This is especially serious for observers with less experience in lucky imaging observations. Therefore, it is an important research field to build a high-speed and even real-time lucky imaging system by using new processing hardware with great computing power and to study new fast lucky imaging algorithms [2], [5], [10], [21].

In the recent decade, the graphics processing unit (GPU) and field programmable gate array (FPGA) with powerful parallel processing capabilities have been introduced into the field of image processing as hardware accelerators [11]–[13]. Although the GPU has great parallel processing capability, its real-time performance is not as good as that of FPGA. A GPU only has data parallelism, while an FPGA has both data parallelism and pipeline parallelism. In addition, a GPU is far less flexible than an FPGA. Once a system based on a GPU is built, its hardware resources cannot be adjusted according to actual needs. However, an FPGA can be programmed according to the actual needs [14] to realize reconfigurable computing [15]. This is the main reason why an FPGA is chosen as the hardware accelerator in this paper.

The main contributions of this study include the following two aspects:

(1) The authors propose a new real-time lucky imaging algorithm that consists of three main processes: real-time image preprocessing (Gaussian filtering, cosmic ray removal), real-time lucky imaging (dynamic sorting-free image selection, improved image registration and storage, and parallel superposition), and multiple threshold display.

(2) The authors design and implement the new algorithm on a small and medium-scale FPGA, during which several new techniques and methods were proposed and successfully tried, such as a sorting-free image selection method based on on-chip RAM, a parallel image registration and storage method based on DDR3, a parallel image superposition technique, and an on-site adjustable multithreshold image display scheme.

II. RELATED WORK

The key of astronomical lucky imaging with FPGA is not to propose a whole new complex imaging algorithm but rather to determine how to improve and modify the existing CPU-based software programming algorithm according to the characteristics of FPGA, describe lucky imaging with hardware description language (HDL), and implement the modified algorithm on an FPGA device of extremely limited on-chip resources. At present, there are only two lucky imaging systems based on FPGA; one is FastCam, which has been put into actual observation, and the other is a test system developed by Zhao, Li *et al.* in our laboratory.

In 2008, the team of Oscoz and Piqueras first implemented the lucky imaging using an FPGA and built an FPGA-based lucky imaging system–FastCam, with which astronomical observations were carried out and real-time high-resolution images were obtained [16]. The FastCam system employs a high-performance Vertex-4 FPGA and its development board manufactured by Xilinx. The front-end EMCCD camera outputs images of 512×512 pixels that are transmitted to the FPGA development board through Gigabit Ethernet. These images are cut into small images of 128×128 pixels and processed by a real-time lucky imaging algorithm. Then, the reconstructed high-resolution image is obtained and output to a video graphics array (VGA) displayer. At the same time, the resulting image is transmitted to a personal computer (PC) through Gigabit Ethernet and saved. FastCam also has its limitations: (1) Because the FastCam system only uses the FPGA on-chip random access memory (RAM) as the image storage unit instead of synchronous dynamic random access memory (SDRAM) on the development board, the FastCam can only process cropped small images with 128×128 pixels. As Piqueras reported, the real-time algorithm can handle large images up to 512×512 pixels, but in fact, due to the limitations of on-chip RAM resources, even using a newly launched Vertix-7 FPGA does not enable the FastCam system to process the original image of 512×512 pixels. (2) The number of real-time processing frames is fixed to 2048. (3) Unlike most classical lucky imaging algorithms, the FastCam does not use a constant percentage but uses a real-time threshold to select good images. If the peak flux count of the brightest speckle in the current image is greater than the real-time threshold, this image is selected for registration and superposition, and at the same time, the real-time threshold will be updated; otherwise, this image will be discarded. This may result in the selected and superimposed images that are not exactly the images with the maximum instantaneous Strehl ratio (i.e., peak flux count) in the 2048 frames. (4) Because the FastCam does not remove the images containing cosmic rays, the FastCam will directly select these images for registration and superposition, and the real-time threshold will become very large, causing subsequent good images to be selected no longer. These shortcomings will limit the processing ability and imaging effect of the system. Moreover, their papers only introduce the system structure and the basic framework of the algorithm and the experimental results and do not describe the specific algorithms and implementation techniques. Especially, no methods for determining or calculating three key parameters (i.e., two configurable input thresholds and one real-time threshold) for image selection are introduced. All of these make it impossible for others to reliably simulate the FastCam algorithm on PCs and repeat their experiments.

In 2018, Zhao *et al.* in our laboratory built an FPGA-based lucky imaging system using a low-cost Spantan-6 FPGA manufactured by Xilinx and its development board. This system can only process 1,000 frames of cropped images of 128 × 128 pixels, and only use a single threshold for display processing. Zhao's paper details the specific design and implementation of each processing module in an FPGA [10]. This is just a test system for validating the FPGA-based algorithm. Although the test system realizes the function of lucky imaging, there are also some shortcomings, such as a serial peripheral interface (SPI) for a secure digital memory (SD) card that is used in the image transmission link, the overly simple and direct hardware description of the lucky imaging algorithm, and the few parallel processing links in the design and implementation process. Although the function and timing sequence of each subalgorithm and its module are divided clearly and independently, the comprehensive resource utilization and clock efficiency are low, resulting in a processing capability and speed of the test system that cannot meet the real-time requirements with a performance far from the FastCam system.

In 2019, Duan *et al.* in our laboratory built an improved version of Zhao's system [21]. In Duan's system a Gigabit Ethernet link is used instead of the SPI link for image transmission, but Duan's lucky imaging algorithm is the same as Zhao's, and main function modules related to the algorithm are transplanted from Zhao's system. Compared with Zhao's system, the total running time of Duan's system is greatly reduced, but the processing speed of the lucky imaging algorithm itself is not improved substantially.

To break through the limitations of the FastCam system and solve the problems of Zhao's and Duan's system, we propose a new FPGA-based lucky imaging algorithm different from the existing algorithms (i.e., the classical algorithm, the FastCam and Zhao's algorithm). Compared with the existing algorithms, the proposed algorithm has 5 key innovations: (1) real-time Gaussian filtering and cosmic ray removal, (2) parallel and dynamic image selection without sorting, (3) parallel and dynamic registration with off-chip memory, (4) parallel superposition, and (5) multithreshold displaying. Compared with the FastCam and Zhao's system, the system based on the proposed algorithm has 5 new functions: (1) by using double-data-rate three synchronous dynamic random access memory (DDR3 SDRAM, short for DDR3) as the registered image storage unit, the original EMCCD image of 512 × 512 pixels can be processed; (2) real-time processed images are up to 10,000 frames or more; (3) the real-time image selection rate is a constant 1% to ensure that the selected images are the ones with the highest peak ranking in all images; (4) the original images are filtered with a Gaussian filter and the frames with cosmic rays are removed; and (5) nine real-time thresholds are used to binarize the reconstructed images that are displayed on a VGA displayer and transmitted back to the PC via Gigabit Ethernet.

This paper describes the proposed algorithm, the overall framework of the system implemented, and the specific design and implementation of each FPGA processing module. Some simulation or test experiments are designed to verify the specific algorithms and modules. The FPGA system was tested by using a large number of short-exposure images observed on the astronomical telescope with a diameter of 2.4 m at the Lijiang Observatory, Yunnan Observatory, Chinese Academy of Sciences, and the performance of the system is analyzed and discussed.

III. REAL-TIME LUCKY IMAGING ALGORITHM, ITS FPGA DESIGN AND IMPLEMENTATION

Different from the algorithm design and system implementation based on the PC platform, those based on FPGAs are closely related to the hardware platform. Because of the continuity of the research, the hardware implementation platform and software development environment for testing the new real-time lucky imaging algorithm are the same as that of Zhao's and Duan's system in our laboratory [10], [21], that is, an ALINX 516 FPGA development board has a XC6SLX16 device in the Xilinx Spartan-6 family, a Dell workstation has a Xeon-E5620 CPU, the FPGA design and development environment is ISE Design Suite 14.7, and the verification and debugging tools for hardware design are Xilinx ChipScope Pro 14.7 and Mentor Graphics ModelSim SE 10.5, a simulation tool for algorithm testing is a PC with a Core i5-3230M, Windows 7 and MATLAB R2016a. This section introduce the new real-time lucky imaging algorithm, and then introduce the algorithm implementation techniques on the FPGA.

A. NEW REAL-TIME LUCKY IMAGING ALGORITHM

1) DESCRIPTION OF THE ALGORITHM

This novel FPGA-based real-time lucky imaging algorithm is different from the conventional CPU-based lucky imaging algorithm [1], [4], [18], and also different from FastCam and Zhao's FPGA-based real-time lucky imaging algorithms [2], [10], [16]. Firstly the proposed algorithm uses a Gaussian filter and a cosmic ray removal process to denoise the input images, and then uses a constant percentage of image selection for dynamic image selection that does not require any sorting operations. After that, it only registers and stores the selected images by DDR3 ping-pong access operations, and dynamically superimpose the registered images to obtain high-resolution resulting images; finally, uses 9 piecewise thresholds to binarize the resulting images and dynamically updates the VGA display.

Let n stand for the current number of input frames, s for the constant percentage of image selection, m for the current number of selected images, nr and nc for the number of rows and columns of the registered images, respectively. The m is the nearest integer greater than or equal to $n \times s$, that is, $m = \text{ceil}(n \times s)$. The proposed real-time lucky imaging algorithm can be described as follows:

(1) Initialization: the initial values of n and m are 1, and the initial peak flux count x_0 is the median of the peak counts of the three pre-captured images;
(2) Receive serial input astronomical images continuously, and perform Gaussian filtering with a 3×3 window for the current frame (nth frame);
(3) Find the peak information of the nth image, that is, the peak flux count and its location;
(4) Calculate an adaptive threshold according to Equation (1) given in Section III-C;
(5) By comparing the peak count with the adaptive threshold, determine whether the nth image contains cosmic rays. If yes, discard the current frame and go to step (2);
(6) Create an FPGA on-chip RAM area of $m + 1$ units (called peak information register, 32 bits per unit) to store the peak information of the selected images and the current image; and create a DDR3 memory area of $nr \times nc \times (m+1) \times 2$ bytes to store the selected images after registration;
(7) Stored the current image peak information in RAM Unit $m + 1$.
(8) Compare the $m+1$ peak counts in these RAM units to find an address of RAM Unit k (i.e., address k) where the minimum value is located;
(9) Write the current peak information from RAM Unit $m + 1$ to RAM Unit k;
(10) Crop the current image with its peak pixel as the center, and store the resized image of $nr \times nc$ pixels in Block k of the DDR3 memory;
(11) If $n = 1$, go to step (13);
(12) Compare the value of the current address k with that of the last address k. If they are the same, that is, the k is not updated, go to step (16);
(13) Superimpose all registered images stored in m DDR3 blocks in parallel;
(14) Binarize the superimposed image by using several thresholds that form an arithmetic progression;
(15) Integrate these binary images into one image and then sent it to a VGA monitor for display;
(16) Update the values n and m, that is, $n = n + 1$, $m = \text{ceil}(n \times s)$, and go to step (2).

In the above procedure, step (1) is designed for initialization of the algorithm, steps (2) to (5) for receiving the input images and performing image pre-processing to suppress noise, steps (6) to (9) for selecting good images, step (10) for registering and storing the selected images, and steps (11) to (16) for dynamically updating and displaying the resulting image, of which step (12) for triggering the update of the resulting image according to the update of the address k, step (13) for reading and superimposing the selected images, and steps (14) to (15) for enhancing and displaying the resulting image.

It can be seen from the above algorithm flow that the proposed image selection algorithm is performed according to a fixed selection ratio. This is similar to the conventional algorithm, but it does not perform the global comparison and sorting like the conventional algorithm. Because the selection percentage s is small, only a small number of comparison operations are required in the selection process. As a result, the calculation amount is greatly reduced. Due to the fixed selection ratio, it is also different from the real-time threshold selection scheme of the FastCam. This sorting-free image selection algorithm is the biggest highlight of the proposed real-time lucky imaging algorithm.

If the proposed algorithm is implemented on a FPGA, after the system is powered on, as long as there is an image input, the above process will continue. Since only a small number of selected images need to be stored in the DDR3, it can continuously process up to 10,000 frames of input images without limiting the number of input images like the existing real-time lucky imaging systems. Since the number of selected images m increases gradually with the increase of the number of input images n, the amount of the FPGA on-chip RAM and the off-chip DDR3 is gradually increased; at the same time, the address k changes with the result of the selected images, thus the lucky imaging results are dynamically updated. It can be said that with the continuous input of the images, the lucky imaging process is dynamically adjusted.

2) VERIFICATION OF THE ALGORITHM

As seen from the above process, although the image selection in this algorithm is based on a fixed percentage, due to the parallel processing characteristics of FPGA, the image selection can be performed simultaneously with the image input. With the image input, therefore, the selection is carried out gradually or dynamically through simple comparison operations (that is, no sorting is required); while in the conventional post-processing algorithm, due to the requirements of global image selection, the image selection is only started after all images are input, in which a full comparison sorting algorithm is used. It is because of the improvement of the selection method that the amount of computation is greatly reduced, making the proposed algorithm suitable for real-time lucky imaging. However, whether this change is feasible and effective needs to be tested with observed data.

The test is carried out in three steps: (1) write a simulation program on a PC and MATLAB platform according to the above algorithm; Note that the RAM units and DDR3 memory blocks in the FPGA-based algorithm are represented by some arrays and matrices in the MATLAB program; (2) use the observed three data sets of astronomical HDS70 images

TABLE 1. Comparison of the results obtained by the 2 algorithms.

	1st set	2nd set	3rd set	Average
Num of the diff. frames	5	11	7	7.6667
Original PSNR	66.0286	63.1994	64.5845	64.6042
Normalized PSNR	54.8267	60.5771	54.4779	56.6272
Original SSIM	0.9648	0.9239	0.9538	0.9475
Normalized SSIM	0.9999	0.9997	0.9998	0.9998
Total AET of the conv.	146.8251	144.4013	145.9168	145.7144
Total AET of the proposed	137.9042	135.2006	136.5635	136.5561
AET for image reading	126.9696	125.4656	126.0042	126.1465
AET of the conv. LI	19.8555	18.9357	19.9126	19.56793
AET of the proposed LI	10.9346	9.735	10.5593	10.40963
Overall time eff. improved	6.08%	6.37%	6.41%	6.29%
LI time eff. improved	44.93%	48.59%	46.97%	46.83%

(10,000 frames in each set) to conduct the simulation experiments at a 1% selection ratio; (3) compared the simulation results with the results obtained by the conventional post-processing algorithm without GPUs described in Ref. [18]. The simulation results obtained by the two algorithms are shown in Table 1.

The first row of Table 1 shows the difference between the proposed algorithm and the conventional algorithm in the results of image selection. With 1% selection, for each data set of 10,000 frames, 100 good images are selected. Obviously, the results of the proposed dynamic image selection method are different from those of the conventional global image selection method, but only the last few frames in the 100 lucky frames are different. It can be said that the difference in selection is relatively small, about 7.67%. In other words, the effectiveness of the new selection method is about 92.33%.

Since the two algorithms have differences in image selection, the imaging results are inevitably different. However, because the differences in the selection results are small, it is difficult for the human eye to distinguish the differences between the two resulting images, so it needs to be illustrated with image data. To this end, we take the high-resolution image obtained by the conventional algorithm as the reference, and calculate the image quality indexes of the resulting image of the proposed algorithm, namely a peak signal-to-noise ratio (PSNR) and a structure similarity (SSIM). Rows 2 to 5 of Table 1 show the PSNR and SSIM in two cases, that is, the PSNR and SSIM of the original and the normalized resulting images. In general, a PSNR is greater than 40, indicating that the image is very good. Therefore, the data in Rows 2 and 3 of Table 1 show that the resulting image of the proposed algorithm is very similar to the resulting image of the conventional algorithm.

As seen from Rows 4 and 5 of Table 1, all the original SSIMs are greater than 0.92, and all the normalized SSIMs are very close to 1, which further illustrates that the resulting image of the proposed algorithm is very similar in structure to the resulting image of the conventional post-processing algorithm.

Another important index of algorithm evaluation is the average execution time (AET) in second. The AET listed in Table 1 is the average of results recorded in two simulations. Rows 6 and 7 represent the total AETs of the two algorithms, Row 8 the AETs for reading the images from a hard disk to the computer memory, and Rows 9 and 10 the AETs of the two lucky imaging algorithms themselves (i.e., Row 6 and Row 7 minus Row 8, respectively). Rows 11 to 12 are the time efficiencies improved by the proposed algorithm.

In terms of total AET, the difference between the two algorithms is not very large, mainly because the time for reading 10,000 frames of images from the hard disk to the computer memory (about 126s) is much greater than the AET for lucky imaging (10 to 20s). In general, the proposed algorithm is 9.1583 seconds faster than the conventional algorithm, and the time efficiency is improved by 6.29%. If we do not consider the time taken to read the images from the hard disk to the computer memory, that is, simply compare the AETs of the two algorithms, the proposed algorithm is still 9.1583 seconds faster than the conventional algorithm, but the time efficiency is improved by 46.83%. These show that the proposed algorithm is more efficient in execution time.

In summary, the new real-time lucky imaging algorithm, in terms of imaging effect, is very similar to the conventional algorithm, while in terms of time efficiency, it is better than the conventional algorithm, can be used as a feasible and effective alternative to the conventional lucky imaging algorithm.

In addition, since we are not clear about the specific method of determining the real-time threshold in the FastCam image selection [16], we cannot compare the proposed algorithm with the FastCam, which is a pity.

Any good algorithm must be implemented on a suitable platform and with proper technical methods. Otherwise, the expected goal cannot be achieved. Since the proposed algorithm is feasible and effective, as long as the FPGA design of the above algorithm is reasonable, the processing time can be reduced to a short enough level, so that the system can complete the process of image selection, registration, superposition and display before the next frame of image arrives, thereby meeting the requirements of real-time lucky imaging. The FPGA design and implementation of the new algorithm or the new real-time lucky imaging system, therefore, becomes the next key issue.

B. OVERALL DESIGN OF THE SYSTEM

For real-time applications, the lucky imaging system is required to have capabilities of high-speed image inputting, processing and outputting. Similar to FastCam, the new system uses Gigabit Ethernet as the image input interface, the outputs directly display on a VGA monitor, and the image processing is carried out by logic units inside the FPGA. Therefore, the real-time lucky imaging system will mainly include a Gigabit Ethernet receiving/transmitting module, an image preprocessing module, a lucky imaging module, a multithreshold binary module, and a VGA control/display

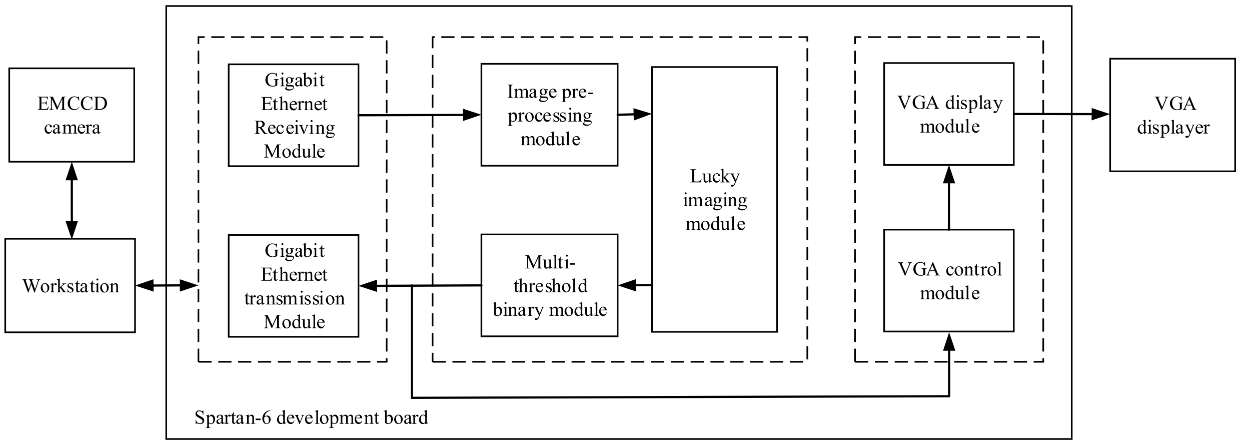

EMCCD camera
Workstation
Gigabit Ethernet Receiving Module
Gigabit Ethernet transmission Module
Image pre-processing module
Multi-threshold binary module
Lucky imaging module
VGA display module
VGA control module
VGA displayer
Spartan-6 development board

FIGURE 1. Overall structure of the real-time lucky imaging system.

module. All modules are designed and implemented with Verilog HDL. The overall structure of the system is shown in Figure 1. Among them, the four modules of Gigabit Ethernet, image preprocessing, multithreshold binary and VGA control are not available in Zhao's system, and the lucky imaging module is redesigned according the proposed algorithm. The other modules are approximately the same as those of Zhao's and Duan's system. The workflow of this system can be summarized as follows:

(1) After the system is powered on, the short-exposure astronomical image data taken in real time by the EMCCD and stored temporarily in the workstation is continuously transmitted to the Gigabit Ethernet receiving module through a gigabit medium independent interface (GMII). Then, the received image data are sent to the image preprocessing module for Gaussian filtering and removal of cosmic rays. After that, the preprocessed data are sent to the lucky imaging module.

(2) The resulting image generated by the lucky imaging module is sent to the multithreshold binary module to enhance the visibility of the target area and then divides the image data into two channels: one of which carries the resulting image and nine binary images to the Gigabit Ethernet transmitting module that then transmits these images back to the workstation; the other carries the nine binary images to the VGA control/display module to drive the FPGA off-chip interface circuit and display the binary images on the VGA monitor.

(3) In the process of continuous transmission of astronomical images, whenever there is a lucky image update, it returns to step (2) to update high-angular resolution resulting image dynamically in real time.

Scientific EMCCD cameras with high speed and low noise are usually used for astronomical lucky imaging observations [17]. At present, the system can dynamically select lucky images from 100 to 10,000 frame or more of continuously input short-exposure images with dimensions of 512×512 pixels on the FPGA development board, and cut the selected original image into a small image of 64×64 pixels in the process of image registration, then stack the registered images to produce a resulting lucky image, finally display dynamically the resulting image of 64×64 pixels on a VGA monitor.

The Gigabit Ethernet receiving/transmitting module is similar to that of Duan's system [21]. Other modules similar to Zhao's system can refer to Zhao's paper [10] and are not introduced in this paper. The following briefly describes the image preprocessing module, focuses on the lucky imaging algorithm redesigned with new ideas, as well as the multi-threshold binary display scheme.

C. IMAGE PREPROCESSING ALGORITHM AND ITS IMPLEMENTATION

The image preprocessing algorithm consists of two specific algorithms, one for Gaussian filtering of astronomical images, another for removing the frames of cosmic rays. At the same time, the peak flux count of the brightest speckle and its position coordinates (row and column) are obtained.

There are three reasons for using Gaussian filters for noise suppression. (1) Small and medium-sized FPGAs (such as Spantan 6 used in our experiments) have limited on-chip resources and cannot support complicated denoising algorithms, such as CNN-based algorithms. (2) Through our calculation experiments, we find that the Gaussian filter has a better effect on astronomical images acquired by EMCCD cameras. (3) The Gaussian filtering algorithm is relatively simple, occupies less FPGA on-chip resources, and is easy to be implemented on FPGAs.

Therefore we use a standard 3×3 Gaussian filter for noise suppression. When processing with this Gaussian template of 3×3, only 3 rows of image data need to be cached. The filtering for the image is composed of serval simple algebraic operations, such as $\times 2$ and $\times 4$, and $\div 16$, which are equivalent to the corresponding pixel value (digits) in a register being shifted 1 bit and 2 bits to the left, and shifted

4 bits to the right, respectively. It is very consistent with the characteristics of the FPGA logic operation.

Cosmic rays are highly energetic particles from outer space. When these particles hit an EMCCD sensor, a specific cosmic ray image is formed, in which the counts of several pixels are much higher than the peak count of the bright stars in normal astronomical images. Therefore, when image quality evaluation is based on instantaneous Strehl ratio or an image's peak count, it is easy to mistake the cosmic ray image as the reference star (bright star) used for image registration, eventually causing errors in image selection and directly affecting the final results. However, the treatment of cosmic rays has not been mentioned in the relevant papers (such as [2], [10], [16], [21], etc.). Although the probability of cosmic rays appearing in an EMCCD image is very small, once they occur, the result obtained by a classical image selecting method will be wrong.

This system specially designs the cosmic ray removal algorithm used to eliminate the cosmic ray images in real time. It is easy to search and remove cosmic rays by programming on a PC. However, this method has not been implemented on an FPGA. According to the method for eliminating cosmic rays on a PC, there should be two processes of searching and removing when it is implemented on an FPGA. Therefore, the cosmic ray removal algorithm designed in this paper consists of two steps: peak searching and cosmic ray estimating and eliminating, corresponding to the two FPGA processing modules. The peak searching module is implemented by a counter architecture, which searches the peak count by comparison and counting, and determines its location. The cosmic ray removal module is implemented by the state machine architecture. By comparing the peak count searched in the current image with an adaptive real-time threshold, the cosmic ray images are estimated and then removed. The initial count x_0 is the median of the peak counts of the three precaptured images. The adaptive threshold $\mathrm{T_n}$ for the nth frame of astronomical images is determined by x_0 and the peak counts of the first n frames of astronomical images $(x_1, x_2, \ldots, x_\mathrm{i}, \ldots, x_\mathrm{n})$ jointly, that is,

$$T_n = \frac{a}{n}\sum_{i=0}^{n-1} x_i, \tag{1}$$

where n $\geq$ 1, a is a coefficient related to imaging parameters of the used EMCCD camera, and usually between 2 to 2.5 for our observations. To facilitate the implementation of an FPGA, a is set to 2 in this algorithm. An iterative form of Formula (1) for FPGA logic design is

$$T_n = \frac{1}{n}\left[(n-1)\,T_{n-1} + a x_{n-1}\right], \tag{2}$$

If $x_n > T_n$, it can be considered that the current astronomical image contains cosmic rays and should be removed. There is a division operation during the calculation of the adaptive threshold. Considering that there is a time interval for data transmission of one image in the process of finding peaks of two neighboring images, the interval is long enough to complete the division operation. This also reflects the parallel advantage of the FPGA.

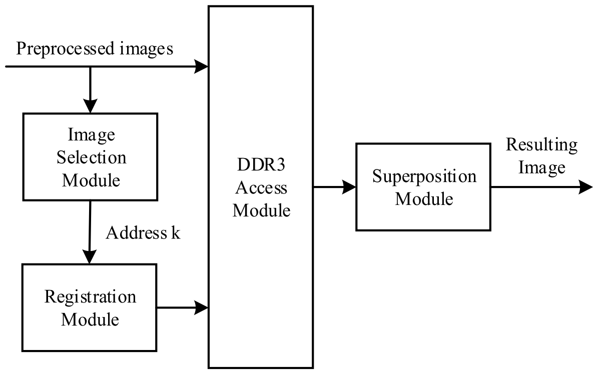


FIGURE 2. Architecture of the lucky imaging module.

D. FPGA-BASED LUCKY IMAGING ALGORITHM AND ITS IMPLEMENTATION

1) ARCHITECTURE OF THE LUCKY IMAGING MODULE

The classical lucky imaging algorithm consists of three independent algorithms and their processing procedures: image selection, registration and stack of short-exposure images. To embed the lucky imaging algorithm in an FPGA, these three algorithms must be modified and integrated by using new ideas, and their hardware processing behaviors must be described in a reasonable way by using Verilog HDL. To meet the requirements of real-time and dynamic processing, we propose a new FPGA-based lucky imaging algorithm in Section III-A, and design a complex lucky imaging module to implement the new algorithm in this section. An architecture of the complex module is shown in Figure 2.

It can be seen that the lucky imaging module is composed of four modules: image selection, registration, DDR3 access and superposition. The image selection module is used to perform the sorting-free image selection process shown in steps (6) to (9) of the proposed algorithm flow in Section III-A, while the registration, DDR3 access and superposition modules jointly complete the image registration and superposition process shown in steps (10) to (13). The following subsections describe the key algorithms or methods and main functions of their modules in Figure 2, as well as the design methods and implementation techniques.

2) IMAGE SELECTION ALGORITHM AND ITS IMPLEMENTATION

The traditional idea of image selection is to compare and sort the peaks in all images and then to select a fraction of top-ranking images according to a certain selection percentage. When the number of input images is large, the image selection algorithm based on comparison and sorting not only has a huge amount of computation but also consumes a considerable amount of valuable on-chip logic resources. It is not suitable for small or medium-size FPGA devices, nor for

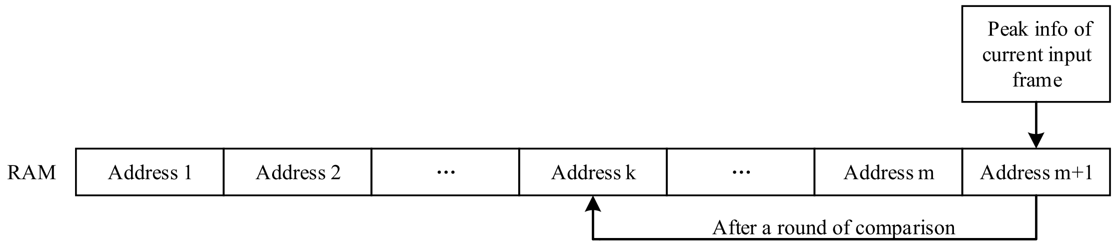


FIGURE 3. Schematic diagram of the image selection algorithm.

real-time lucky imaging systems. For this reason, considering that the ranking order of selected images does not affect the subsequent operations of registration and superposition, we propose a new idea of image selection suitable for FPGA logic implementation, that is, comparison but no sorting, which can reduce the computation and save a considerable amount of on-chip logic resources.

A schematic diagram of the sorting-free image selection algorithm is shown in Figure 3, where each block represents an information register of 36 bits used to store the peak information (i.e., count and location), and m is the number of selected images. In the whole lucky imaging process, m frames of images are selected from n frames of input original images for registration and superposition according to the 1% image selection [2], [18], that is,

$$m = \mathrm{ceil}(n \times 1\%), \tag{3}$$

where ceil() is an integer-valued function, rounding its element to the nearest integer greater than or equal to n × 1%. Note that n increases with the increase of input images; thus, m changes dynamically.

The algorithm flow can be described as follows:

(1) The top-ranking m peak counts and locations from the image preprocessing module are temporarily stored in an on-chip RAM of address 1 to m.
(2) The peak information of the current frame from the image preprocessing module is temporarily stored in the RAM of address m + 1.
(3) Read address 1 to m + 1 in the RAM sequentially, compare the m + 1 peak counts with each other and record the RAM address k of the minimum peak count. If k = m + 1, go back to step (2); otherwise, go to the next step.
(4) Write the peak information temporarily stored in the RAM of address m + 1 into the RAM of address k.
(5) The address k and the location (row and column number) of the peak of the current frame are sent to the registration module for registration.
(6) Go to step (2) until there is no longer an input image.

The image selection algorithm can ensure that the data in RAM address 1 to m is the top m maximum counts among the n image peaks.

3) REGISTRATION ALGORITHM AND ITS IMPLEMENTATION

The registration is to shift the selected image center to the brightest pixel of the reference star. This can be done by cutting the original 512 × 512-pixel images into a small-size one in which the brightest pixel is located in the image center. For a real-time lucky imaging system, it is necessary to process the full-size (512 × 512 pixels) input image, but the area of interest to be displayed is usually a small one around the target star. The proposed algorithm does not limit the size of the displayed area. The area can be larger than 64 × 64 pixels, such as 128 × 128 or 256 × 256 pixels. Note: the nth power of 2 is selected for the convenience of FPGA operation. However, in order to facilitate comparison with the FastCam and Zhao's system, we use a small image of 64 × 64 pixels for comparative experiments.

The registration algorithm presented in this paper crops each selected image centered on the peak location to form a small image of 64 × 64 pixels. Finally, the cropped image will be sent to the DDR3 access module. A key issue in the FPGA logic design is how to address the first pixel of the 64 × 64-pixel image in the DDR3 memory. According to the data storage principle of the DDR3 memory and the requirement of image registration, a registration formula consistent with the logic implementation of FPGA is obtained, as shown in equation (4).

$$\mathrm{addr\_r} = \left[(\mathrm{axis\_x} - 32) \times 512 + \mathrm{axis\_y} - 32\right] \times 4, \tag{4}$$

where addr_r is the address of the first pixel of the 64 × 64-pixel image in the DDR3 memory, and axis_x and axis_y represent the row/horizontal coordinate and column/vertical coordinate stored in the peak information register, respectively. The reason for multiplication of 4 in equation (4) is that one pixel of data occupies 4 address units in the DDR3 memory. All the operations of multiplication of 4 related to the DDR3 memory discussed in the next section are due to this reason.

By comparing this registration formula with the one adopted by Zhao in her paper [10], both of them have achieved the same goal. However, because each image is stored in a one-dimensional way, the recorded position of the peak pixel is a one-dimensional subscript in Zhao's algorithm. Therefore, Zhao uses the image serial number and the subscript position of the peak pixel for image registration, and its registration formula is very complex and hard to

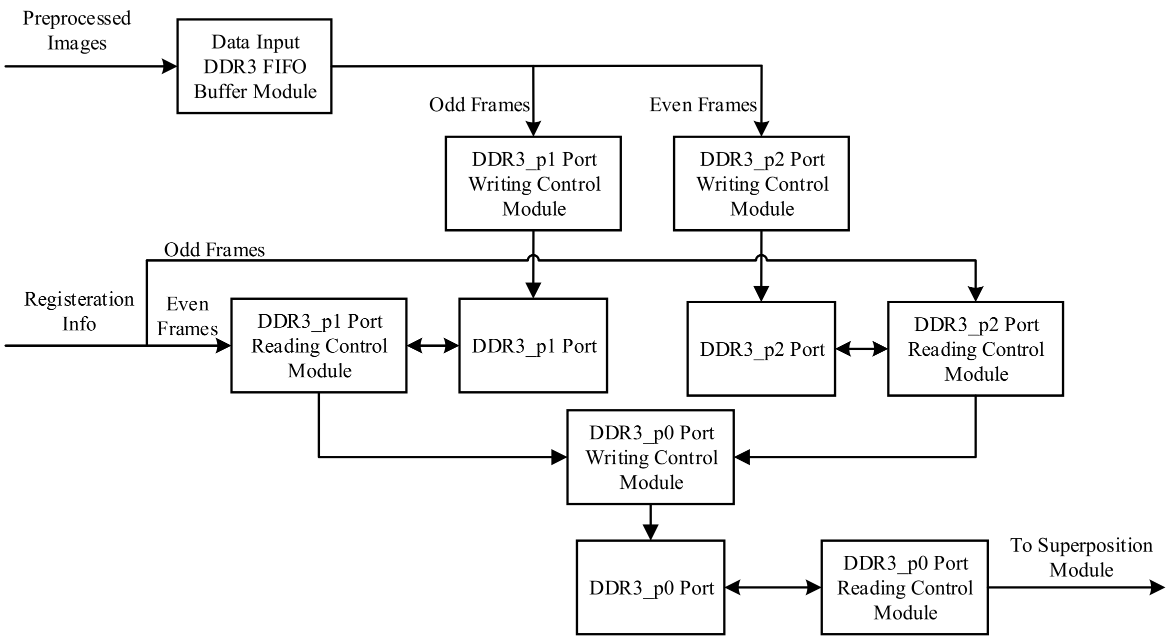

Preprocessed Images
Data Input DDR3 FIFO Buffer Module
Odd Frames
Even Frames
DDR3_p1 Port Writing Control Module
DDR3_p2 Port Writing Control Module
Odd Frames
Registration Info
Even Frames
DDR3_p1 Port Reading Control Module
DDR3_p1 Port
DDR3_p2 Port
DDR3_p2 Port Reading Control Module
DDR3_p0 Port Writing Control Module
DDR3_p0 Port
DDR3_p0 Port Reading Control Module
To Superposition Module

FIGURE 4. DDR3 access module.

understand. In contrast, equation (4) in this paper is concise and easy to understand, and the registration process is clear. This is more suitable for characteristics of the FPGA logic design and can greatly simplify the code structure and reduce the amount of code written in the actual HDL design.

4) IMAGE STORAGE METHOD AND ITS IMPLEMENTATION

In Zhao's algorithm, all astronomical images are directly stored in the DDR3 memory in a one-dimensional way and then registered and superimposed. This image storage method leads to a large consumption of resources (i.e., the RAM, DDR3 and clock) and increases the difficulty of design. Especially when the number of input images is large, the consumption of DDR3 resources becomes very large. Zhao's method is not suitable for any small or medium-sized FPGAs and their development boards, or for construction of any real-time lucky imaging system. For this reason, the proposed image storage method ingeniously divides the DDR3 memory into three memory areas, namely, DDR3_p0 port, DDR3_p1 port and DDR3_p2 port, which are used to write or read the registered images through the corresponding writing or reading control modules, as shown in Figure 4. Since the rate of image input is different from that of data writing in the DDR3, a first-in-first-out (FIFO) data buffer module is designed.

A ping-pong operation for data access is adopted between the DDR3_p1 port for the odd frames and the DDR3_p2 port for the even frames to accomplish the buffer of high-speed real-time image data. The ping-pong operation is implemented by switching the reading and writing operations between two memory areas. When the DDR3_p1 port is used for the writing operation, the DDR3_p2 port is used for the reading operation. When a frame of an image is read out, the state is switched, that is, the DDR3_p1 port is used for the reading operation, and the DDR3_p2 port is used for the writing operation at the same time. The ping-pong operations go on continuously until no images are input. The DDR3_p1 port and DDR3_p2 port only store one frame of the original astronomical image of 512×512 pixels at once.

While the DDR3_p1 port or DDR3_p2 port is read, the DDR3_p0 port stores the cropped images after registration. The storage mode of DDR3_p0 port is similar to the image selection algorithm, as shown in Figure 5. Each storage block stores one cropped image, which occupies $64 \times 64 \times 4 = 16384$ address units. When a cropped image is stored in storage block i, the first address of the storage block is $k \times 16384$, and k is the parameter coming from the image selection module. In this way, it can be guaranteed that the cropped images in storage block 1 to m of the DDR3_p0 port are exactly the cropped images corresponding to the top m maxima of n peaks of all the effective input images.

5) SUPERPOSITION ALGORITHM AND ITS IMPLEMENTATION

The superposition is simply to add all the images registered, and usually it is last step of the classical lucky imaging algorithm. However, for a real-time processing algorithm on an FPGA, it is not a simple problem because the superposition speed and the on-chip resources are also issues that must be considered in addition to the superposition method (e.g., the data flow and the cumulative manner). A solution to the problem is shown in Figure 6.

There is an adder of 32 bits and two RAM modules of 32 bits in Figure 6. RAM1 is a true dual-port RAM with two writing ports and two reading ports, while RAM2 is a simple dual-port RAM with one writing port and one reading port. To accomplish seamless buffering and stacking of image data and to save the buffer space, a ping-pong operation for

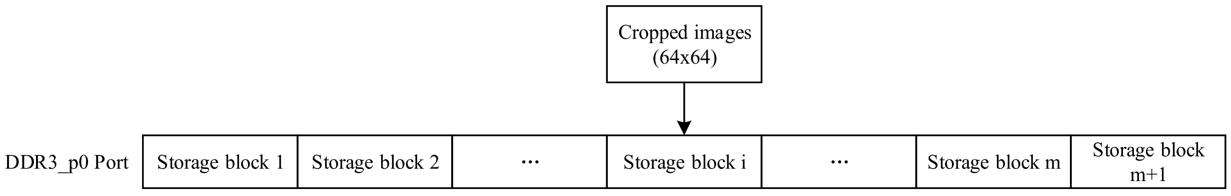


FIGURE 5. **Storage mode of DDR3_p0 port.**

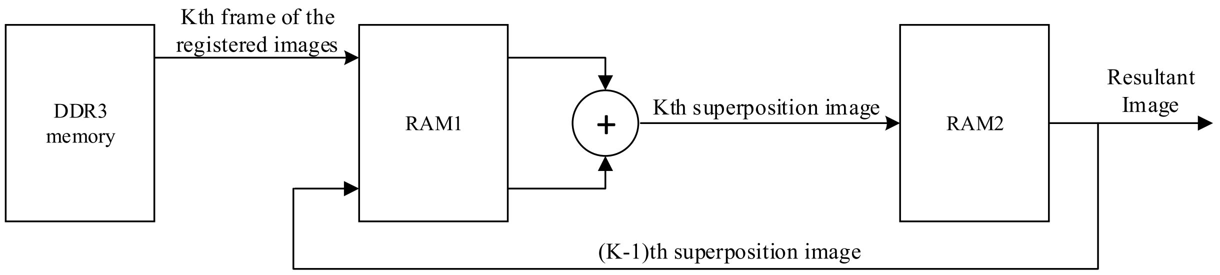

Kth frame of the registered images
DDR3 memory
RAM1
+
Kth superposition image
RAM2
Resultant Image
(K-1)th superposition image

FIGURE 6. **Superposition scheme.**

data access between RAM1 and RAM2 is also used to achieve image superposition. The ping-pong operation is realized by switching the reading and writing operations between two RAMs, that is, when RAM1 is writing, RAM2 is reading. After a frame of an image is added, the read-write state is switched, that is, RAM1 is reading and RAM2 is writing. This superposition process goes on until all the registered images in the DDR3_p0 port are added. At this time, a resulting lucky image is produced and stored in RAM2. In the process of image transmission, as long as a new good image arrives, the superposition module will be reset to start a new round of stacking.

E. DISPLAY SCHEME AND ITS IMPLEMENTATION

Usually, the primary star in a binary or multistar system is much brighter than the companion stars, and it is used as a reference star for lucky imaging. The main purpose of lucky imaging is to distinguish and measure dim companion stars in a binary or multistar system. When a real-time resulting image is displayed directly on the VGA displayer according to the gray value of the image, it is hard to show the dim companion stars. Moreover, for different observation objects, the brightness differences between bright reference stars and dim stars are changed and can even be unknown when searching for new faint celestial bodies. To make the constructed real-time lucky imaging system suitable for different observational targets, we proposed a multithreshold or piecewise-threshold scheme to enhance visibility of the target area.

In this method, nine thresholds are used to binarize the high-angular resolution resulting image. For a threshold T, the basic formula is as follows:

$$I_o(x,y) = \begin{cases} 1 & \text{When } I_i(x,y) > T \\ 0 & \text{When } I_i(x,y) \le T, \end{cases} \tag{5}$$

Where I_i(x,y) is the pixel count of the resulting image after superposition, I_o(x,y) is the pixel value of the output binary image.

Since the resulting image after superposition is updated dynamically in real time, the threshold T cannot be fixed. Two groups of buttons on the FPGA board are therefore set for observers to regulate the threshold T in the field, one for fine tuning and the other for coarse tuning, so that an appropriate threshold is obtained. As the threshold can track the optimal threshold of the resulting image in real time, a good display effect is achieved. However, its initial value T_0 should be set. In the new system T_0 is equal to the average pixel value of the first selected image.

To obtain a better display effect, on the other hand, 8 thresholds are added to binarize the resulting image after superposition. Therefore, there are 9 thresholds distributed according to arithmetic progression with a step of 20, in which the adjustable threshold is located in the middle of the arithmetic progression. As a result, 9 binary images responding to the 9 thresholds are obtained, in which at least two or more can show the dim celestial bodies clearly.

To show the 9 binary images of 64 × 64 pixels on a VGA monitor, a VGA control module is designed to control the positions of these images on the VGA screen. The actual resolution of VGA display is 1024 × 768 (i.e., extended graphics array, XGA). The 9 binary images are displayed on the VGA screen in a 3 × 3 arrangement.

IV. EXPERIMENTATIONS

To verify the proposed real-time lucky imaging algorithm based on an FPGA and its implementation techniques, a real-time processing system is built on an ALINX 516 FPGA development board [10], [21]. The system accomplishes lucky imaging with 1% image selection [2], [18]. A series of the short-exposure astronomical image used in the

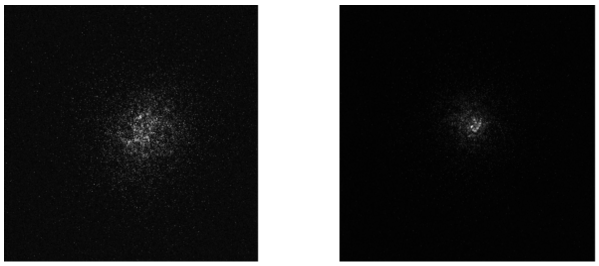


FIGURE 7. Two frames of observed images of HDS 70.

experiment were the observed images of the astronomical binary star system HDS 70 at the Lijiang Observatory, Yunnan Observatory of the Chinese Academy of Sciences on October 20, 2016. There are three data sets, with 10,000 frames in each set. The observational conditions and imaging parameters of the EMCCD camera are listed in Mao's paper [18]. A random frame in a data set is shown in Figure 7(a), and the best frame is shown in Figure 7(b).

To test functions of the system, 10,000 frames of original images are continuously transmitted to the FPGA system via Gigabit Ethernet at the acquisition speed (35 frames per second) of the EMCCD camera, and the FPGA is used for lucky imaging. After the first 100 effective astronomical images are transmitted, the VGA monitor starts to display 9 piecewise-threshold binary images. As the images continue to be transmitted to the FPGA system, the threshold images on the VGA screen are updated dynamically. This shows that the system realizes the function of real-time processing and dynamic updating.

To evaluate the performance of the proposed algorithm, a resulting image return module is designed. When 5,000 frames have been processed, the resulting image and the 9 piecewise-threshold binary images are transmitted back to the PC via Gigabit Ethernet and then saved to a hard disk. The 9 images are displayed by calling several functions in MATLAB, as shown in Figure 8, where Figure 8(a) and 8(b) are the results of piecewise-threshold processing for 5,000 and 10,000 frames respectively. The visual effect of the 9 images on PC screen is the same as that on the VGA screen in the real-time lucky image system. Each small image in the figure shows the resulting lucky image binarized by a corresponding threshold. The number above each small image indicates the serial number for the threshold.

However, after using 9 segmented thresholds, there are four small images in Figure 8(a) and six in Figure 8(b), in which the dim star can be seen directly by human eyes. It shows that the image enhancement effect of the proposed display scheme is good.

The resulting images from 5,000 and 10,000 frames of original images are used determined positions of the primary and the companion star with two-dimensional-modified moment centering algorithm [19]. Obtained results are listed in Table 2. According to the scale of the telescope imaging system 0.043 arcsecond/pixel [18], the separation angles of

TABLE 2. Positions and separation of the primary and the companion star (in pixels).

	Primary star	Companion star	Separation
Results from 5,000 frames	(32.820, 32.356)	(37.512, 26.424)	7.563 (0.325")
Results from 10,000 frames	(32.625, 32.228)	(37.220, 26.382)	7.436 (0.32")

the binary stars are 0.325 and 0.320 arcsecond respectively. This results are close to the data of HDS70 in the binary catalog (0.3 arcsecond) [20]. The error comes mainly from the difference between the position of the binary star at the catalog ephemeris and the position of binary star at our actual observation time, and partially from the lucky imaging and the centering calculation.

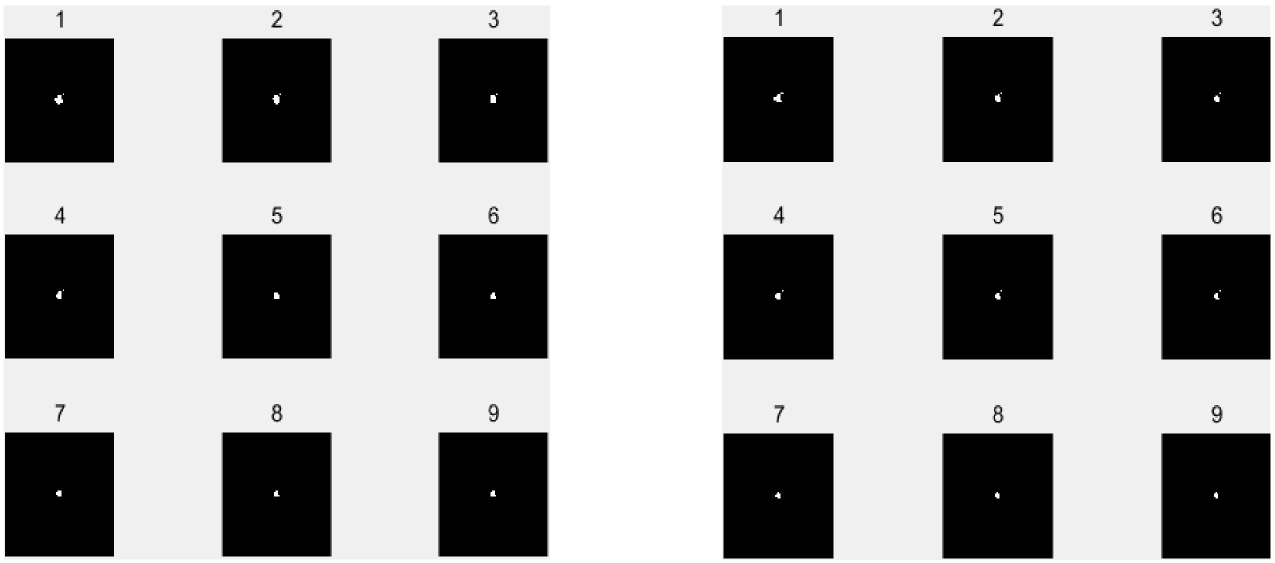
In addition, we have also tested the new system with two other data sets of astronomical images of binary stars HDS 70 and obtained similar results. All of these demonstrate that the proposed real-time lucky imaging algorithm based on FPGA is effective once more, and its implemented system are correct.

V. COMPARATIVE ANALYSES AND DISCUSSIONS

During the research, we not only implement the proposed lucky imaging algorithm and build a new system but also replicate Zhao's experiments, that is, we also implement her lucky imaging algorithm and build Zhao's system to gain comparable experimental data. Because we do not know the specific algorithm and HDL codes used in FastCam [16], we cannot compare the results of the proposed system with those of the FastCam system. The only comparison between the two systems is the final on-chip logic resource consumptions. Since the main modules for lucky imaging in Duan's system are transplanted from Zhao's system, during the comparative analysis of the algorithm, we did not give the Duan's data, except for the system running time [21].

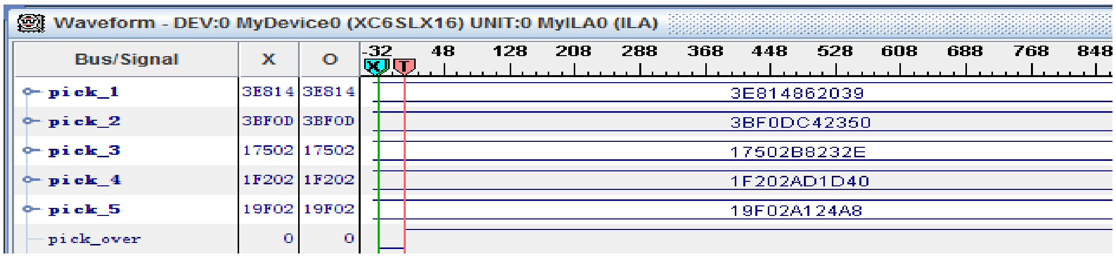
A. COSMIC RAY REMOVAL ALGORITHM

For the three data sets of 10,000 frames of short-exposure images, there were several frames (3 to 8 frames) of cosmic ray images in each set. To make a comparative study, two astronomical images of cosmic rays were inserted into the original image set of 1,000 frames, given that Zhao's system can only process images up to 1,000 frames of 128 × 128-pixel images. The serial numbers of the two frames are 959 (i.e., 3BF in Hex) and 1,000 (i.e., 3E8 in Hex), and their peak flux counts are 3524 (i.e., 0DC4 in Hex) and 5254 (i.e., 1486 in Hex) respectively. The top five peak counts selected from the 1,000 astronomical images were captured by the ChipScope tool. The results of images selected by Zhao's algorithm and by the proposed algorithm are shown in Figure 9, where pick_1 to pick_5 are five peak information registers, and pick_over represents a signal indicating the end of image selection at which the signal is pulled to a high level.



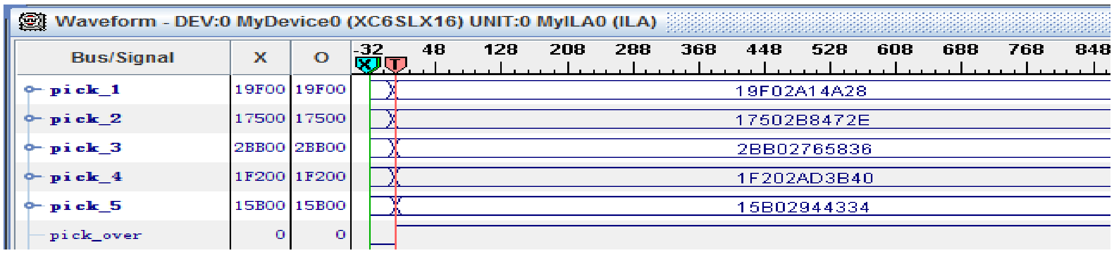
(a) Resultant Images from 5,000 Frames (b) Resultant Images from10,000 Frames

FIGURE 8. Nine piecewise-threshold binary images on a VGA screen.



```
Waveform - DEV:0 MyDevice0 (XC6SLX16) UNIT:0 MyILA0 (ILA)
Bus/Signal     X      O      -32  48  128  208  288  368  448  528  608  688  768  848
pick_1       3E814  3E814    3E814862039
pick_2       3BF0D  3BF0D    3BF0DC42350
pick_3       17502  17502    17502B8232E
pick_4       1F202  1F202    1F202AD1D40
pick_5       19F02  19F02    19F02A124A8
pick_over        0      0
```

(a) Results of image selected by Zhao's algorithm



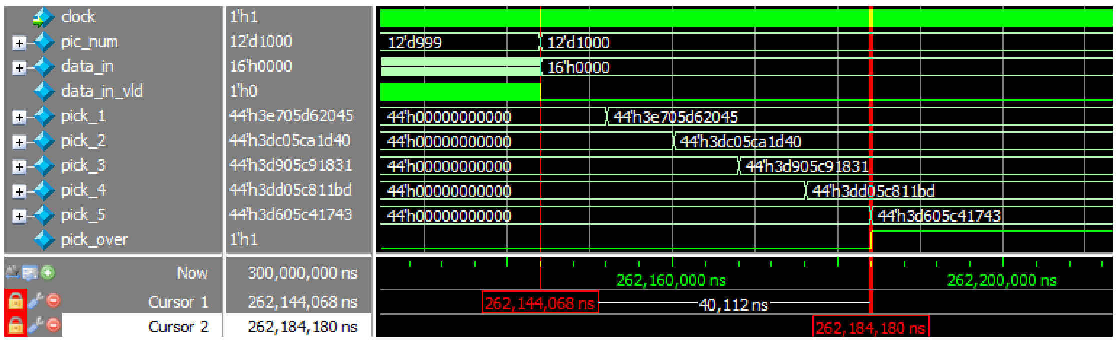
```
Waveform - DEV:0 MyDevice0 (XC6SLX16) UNIT:0 MyILA0 (ILA)
Bus/Signal     X      O      -32  48  128  208  288  368  448  528  608  688  768  848
pick_1       19F00  19F00    19F02A14A28
pick_2       17500  17500    17502B8472E
pick_3       2BB00  2BB00    2BB02765836
pick_4       1F200  1F200    1F202AD3B40
pick_5       15B00  15B00    15B02944334
pick_over        0      0
```

(b) Results of image selected by the proposed algorithm

FIGURE 9. Comparison of image-selecting results.

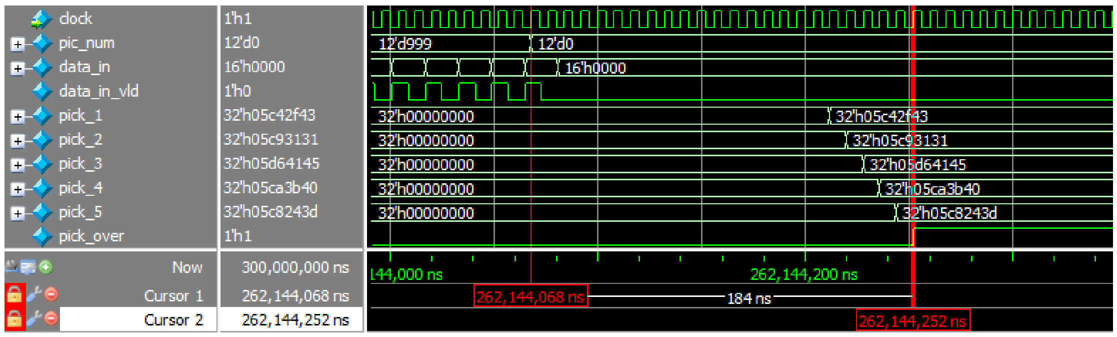
Each peak information register is of width of 44 bits (i.e., 11 hexadecimal digits). The value of the register is expressed in hex, which is listed on the right side of Figure 9. The first 3 hexadecimal digits represent the image serial number, the middle 4 digits represent the image peak count, and the last 4 digits in Zhao's algorithm are used to record the subscript position of the image peak, while in this system they are used to record the horizontal and vertical coordinates of the image peak (note that the first 2 hexadecimal digits represent the horizontal coordinate, and the last 2 digits represent the vertical coordinate). The difference between the two recording methods is introduced in Section III of this paper.

By comparing the data in Figure 9, it can be seen that Zhao's algorithm selects the two images containing cosmic rays as lucky images (pick_1 and pick_2) because her algorithm has no function of filtering and removing cosmic rays. Therefore, her algorithm mistakes the cosmic rays as brightest speckle of the primary star in the binary star system, which would directly affect the subsequent registering and stacking. In contrast, this system does not select any images containing cosmic rays. Moreover, the five frames selected by



```
clock          1'h1
pic_num        12'd1000        12'd999 | 12'd1000
data_in        16'h0000        16'h0000
data_in_vld    1'h0
pick_1         44'h3e705d62045  44'h000000000000 | 44'h3e705d62045
pick_2         44'h3dc05ca1d40  44'h000000000000 | 44'h3dc05ca1d40
pick_3         44'h3d905c91831  44'h000000000000 | 44'h3d905c91831
pick_4         44'h3dd05c811bd  44'h000000000000 | 44'h3dd05c811bd
pick_5         44'h3d605c41743  44'h000000000000 | 44'h3d605c41743
pick_over      1'h1
Now            300,000,000 ns   262,160,000 ns   262,200,000 ns
Cursor 1       262,144,068 ns   262,144,068 ns
Cursor 2       262,184,180 ns   262,184,180 ns
                                40,112 ns
```

(a) zhao's algorithm based on comparison and sorting



```
clock          1'h1
pic_num        12'd0           12'd999 | 12'd0
data_in        16'h0000        16'h0000
data_in_vld    1'h0
pick_1         32'h05c42f43    32'h00000000 | 32'h05c42f43
pick_2         32'h05c93131    32'h00000000 | 32'h05c93131
pick_3         32'h05d64145    32'h00000000 | 32'h05d64145
pick_4         32'h05ca3b40    32'h00000000 | 32'h05ca3b40
pick_5         32'h05c8243d    32'h00000000 | 32'h05c8243d
pick_over      1'h1
Now            300,000,000 ns   144,000 ns   262,144,200 ns
Cursor 1       262,144,068 ns   262,144,068 ns
Cursor 2       262,144,252 ns   262,144,252 ns
                                184 ns
```

(b) Proposed sorting-free algorithm only based on comparison

FIGURE 10. Simulation results of the two image selection algorithms.

this algorithm include 3 frames selected by Zhao's algorithm. All of these show that the cosmic ray removal algorithm is effective, and its implemented module is correct.

B. IMAGE SELECTION ALGORITHM

To make a comparative analysis, the FPGA implementation of the proposed sorting-free image selection algorithm and Zhao's image selection algorithm based on comparison and sorting are simulated by using the ModelSim tool. The results are shown in Figure 10, where the clock used refers to the Gigabit Ethernet clock with a frequency of 125 MHz and a period of 8 ns, pic_num is the number of frames (1,000 frames) from which top 5 peak counts are selected, data_in represents the simulated input data, and data_in_vld is an indicator for valid input data. The input data are generated by a random function, pick_1 to pick_over are the peak information registers similar to those illustrated in Section V-A. However, for the proposed algorithm does not need to record information about the serial numbers of selected images, these registers in this proposed system are of width of only 32 bits (i.e., 8 hexadecimal digits). The first 4 hexadecimal digits represent the peak count of the image, and the last 4 digits record the horizontal and vertical coordinates of the image peak (where the first 2 digits represent the horizontal coordinate and the last 2 digits represent the vertical coordinate). The simulation results for each register or signal are listed on the right side of Figure 10, where two red vertical lines and the data below the lines are the beginning and ending time of the image selecting. The span between the two lines is the time consumed by the image selection algorithm, also shown in the figure.

The following are comparative analyses and discussions about the two image selection algorithms from three aspects:

(1) By comparing two group of results of image selection (i.e., the 4th to 7th hexadecimal digits of pick_1 to pick_5 in Figure 10(a) and the 1st to 4th hexadecimal digits of pick_1 to pick_5 in Figure 10(b)), it can be seen that the results of the two algorithms are exactly the same except for the different order of image selection. This shows that the proposed algorithm can obtain the correct result of image selection, as also accomplished by Zhao's algorithm, and shows once again that the proposed algorithm has no problems in image selection. The order of image selection results does not affect the subsequent image registration and superposition. That is, it is not necessary to sort all the peak flux counts during image selection, as long as the images responding to the top m peak counts can be selected from the n frame of input images. This is the innovative point of the image selection algorithm in this paper.

(2) By analyzing the clock resources consumed in Figure 10(a), it can be seen that Zhao's image selection algorithm based on comparison and sorting takes considerable clock resources, more specifically, 5,014 (i.e.,40,112 ns/8 ns) clock cycles, to complete the image selecting after reading all the images. From Figure 10(b), it can be seen that the proposed

algorithm carries out image selection while reading images. Therefore, after reading all images, only a very small amount of clock resources, 23 (i.e., 184 ns/8 ns) clock cycles, are consumed to finish the image selection. Compared with Zhao's image selection algorithm, the clock resource consumption of the proposed algorithm decreases 99.54% (i.e., |5,014-23|/5,014), which greatly improved the clock utilization efficiency and fully demonstrates the advantages of parallelism and flexibility of the FPGA.

(3) By comparing and analyzing the consumed on-chip resources, it can been seen that Zhao's algorithm needs to store all the peak flux counts in the RAM on the FPGA chip in the process of receiving images. All the information about the serial numbers, peak counts and their locations of all received images also needs also to be registered at the same time. After all the images are received, the peak counts need to be read out from the RAM for sorting in turn. Finally, the images in which the peak counts are top-m ranked are selected. The serial number of an image is represented by a 12-bit binary number in the FPGA, and the peak count and peak location in an image are represented by 16-bit binary number. Because the peak counts of n frames of input images in Zhao's algorithm are sorted directly, the on-chip RAM resource required is $y_{11} = (12 + 16 + 16) \times n$. However, the proposed algorithm only needs to store the peak counts and locations of $m + 1$ frames, and the on-chip RAM resource required is $y_{12} = (16 + 16) \times (m + 1)$. Compared with Zhao's algorithm, therefore, the RAM resource consumption of implementing the proposed algorithm is as follows

$$\eta_1 = \frac{y_{12}}{y_{11}} = \frac{8\,(m+1)}{11n}, \tag{6}$$

Because n is usually greater than 1,000 and m is only 1% of n, η_1 is less than 1%, and when n becomes continuously larger, η_1 tends to 0.7273%. To intuitively compare the on-chip RAM consumption of the two algorithms, the ratio η_1 is plotted by using the MATLAB tools, as shown in Figure 11. It can be seen from the figure that the RAM resource consumption of implementing the proposed image selection algorithm is far less than that of implementing Zhao's algorithm. This demonstrates that the proposed sorting-free image selection algorithm has a great advantage over Zhao's implementation.

In fact, when Zhao's algorithm based on a comparison and sorting is used to select images for more than 1,000 frames of input images, the RAM resources on the Spantan-6 FPGA are not sufficient. As a result, Zhao's system can only accomplish lucky imaging for 1,000 frames of 128 × 128-pixel astronomical images at most. However, the RAM resource needed by the proposed image selection algorithm is only related to the frame number m of selected images, but not to the total number n of input images, which can greatly save the on-chip logic resource. This is one of the reasons why the proposed system can accomplish real-time dynamic lucky imaging for more than 10,000 frames of 512×512-pixel images on a small and medium-scale FPGA device.

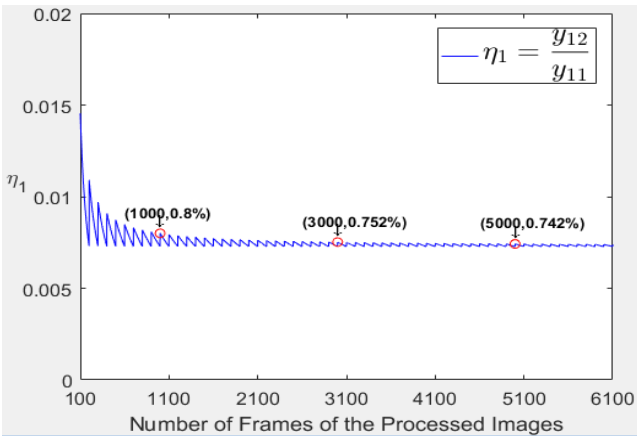


FIGURE 11. Comparison of RAM resource consumption in the two image selection algorithms.

C. IMAGE STORAGE METHODS AND RESOURCE CONSUMPTION OF DDR3

Here, the image storage methods used in Zhao's lucky imaging algorithm and the proposed algorithm are compared and analyzed from the perspective of hardware resource consumption. Zhao's image storage method needs to store all the input images in the DDR3 memory, so the n frames of input astronomical images with the format of 512×512 pixels occupy the DDR3 resource given by $y_{21} = 512 \times 512 \times 4 \times n$. However, the proposed method needs only to store two input images with the format of 512 × 512 pixels in the ports DDR3_p1 and DDR3_p2 using ping-pong access operation and $m + 1$ frames of small registered images with 64 × 64 pixels in the port DDR3_p0; thus, the DDR3 resource spent by this storage method is $y_{22} = (512 \times 512 \times 4) \times 2 + (64 \times 64 \times 4) \times (m + 1)$. Compared with Zhao's image storage method, therefore, the DDR3 resource consumption of implementing the proposed method is as follows:

$$\eta_2 = \frac{y_{22}}{y_{21}} = \frac{m+129}{64n}, \tag{7}$$

Since n is usually greater than 1,000 and m is only 1% of n, η_2 is less than 0.22%, and when n becomes continuously larger, η_2 tends to 0.0156%.

To more intuitively compare the DDR3 resource consumption of the two storage methods, the ratio η_2 is plotted by using the MATLAB tools, as shown in Figure 12. As seen from the figure, the DDR3 resource consumed by the proposed algorithm is far less than that by Zhao's algorithm. If there are 10,000 frames of short-exposure images for lucky imaging, Zhao's algorithm takes up 10 GB of DDR3 resources, which is an amount far beyond the total DDR3 resources on the development board. With 1% image selection, however, the proposed algorithm takes up only 3.61 MB. By contrast, the algorithm has a great advantage in image storage. This is another important reason why the proposed system can accomplish real-time dynamic lucky imaging for more than 10,000 frames of 512 × 512-pixel

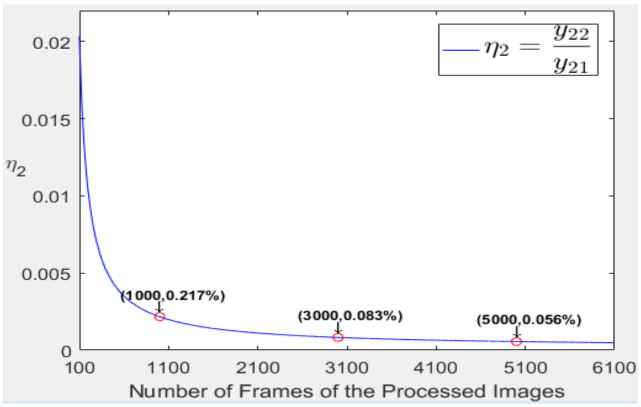


FIGURE 12. Comparison of DDR3 resource consumption.

images on a small and medium-scale FPGA development board. In principle, this system can directly process the original images of 1k × 1k pixels.

Even if the original format images without cropping during the registration are directly used for storage, the memory resource for the DDR3_p0 port taken up by the proposed method is only (512 × 512 × 4) × (m + 1). Compared with Zhao's method, therefore, the DDR3 resource consumption of implementing the proposed method is as follows:

$$\eta_2 = \frac{y_{22}}{y_{21}} = \frac{m+3}{n}, \tag{8}$$

Since n is very large, η_2 is approximately equal to the fixed percentage of image selection (1%). This shows that in this situation, the demand of the algorithm for DDR3 is still far less than that of Zhao's method. When processing 10,000 frame of input images, only 100 MB is needed, and thus a DDR3 memory of 128 or 256 MB on a small or medium-scale FPGA development board can meet the requirements of the proposed lucky imaging algorithm.

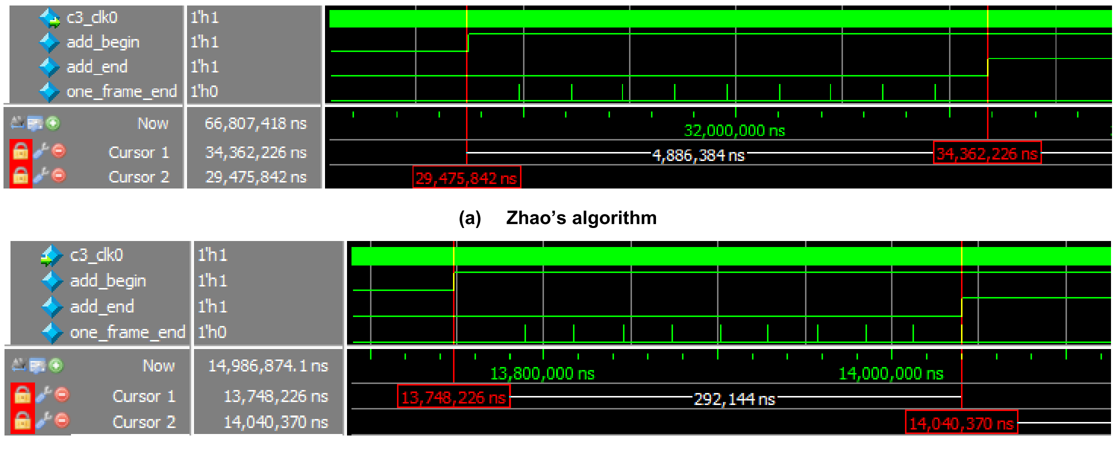
D. SUPERPOSITION ALGORITHM

To make a comparative study, Zhao's superposition algorithm and the proposed one are simulated by using the ModelSim tool. The simulation experiment is to select 10 frames from 20 randomly generated images and superpose them. The results are shown in Figure 13, where c3_clk0 is a DDR3 working clock of 312.5 MHz (i.e., a period of 3.2 ns), add_begin is a superposition start signal, add_end is an end signal that jumps from 0 to 1 after all images are added together, and one_frame_end is an end signal that jumps from 0 to 1 after one frame of registered images is superposed. The simulation waves or data for these signals are listed on the right side of Figure 13, where two red vertical lines and the data below the lines are the beginning and ending time of the image superposition. The span between the two lines is the time consumed by the superposition algorithm, also shown in the figure. Note that the proposed algorithm is not limited to the frequency of 312.5MHz. In fact, the system can work properly at 625MHz that is the maximum operating frequency of the DDR3 supported by the FPGA development board we used.

As seen from Figure 13, in terms of clock resource consumption, a total of 1,526,995 (i.e., 4,886,384ns/3.2ns) clock cycles are consumed by Zhao's superposition algorithm, while the proposed superimposed algorithm consumes 91,295 (i.e., 292,144ns/3.2ns) clock cycles. Compared with the Zhao's algorithm, the clock resource consumption in the proposed algorithm is reduced by 94.02% (i.e., |1,526,995-91,295|/1,526,995). The proposed superposition algorithm greatly improves the efficiency of clock utilization. This result fully embodies the advantage of using the ping-pong operation to realize image superposition on an FPGA.

E. CLOCK CONSUMPTION IN FPGA

Using the FPFA-based lucky imaging algorithm proposed in this paper and the algorithms in Zhao's and Duan's paper, lucky imaging experiments were performed for 1,000 frames



(a) Zhao's algorithm

c3_clk0	1'h1
add_begin	1'h1
add_end	1'h1
one_frame_end	1'h0
Now	66,807,418 ns
Cursor 1	34,362,226 ns
Cursor 2	29,475,842 ns

32,000,000 ns
4,886,384 ns
34,362,226 ns
29,475,842 ns

(b) Proposed algorithm

c3_clk0	1'h1
add_begin	1'h1
add_end	1'h1
one_frame_end	1'h0
Now	14,986,874.1 ns
Cursor 1	13,748,226 ns
Cursor 2	14,040,370 ns

13,800,000 ns
14,000,000 ns
13,748,226 ns
292,144 ns
14,040,370 ns

FIGURE 13. Simulation results of the two superposition algorithms.

TABLE 3. Running times (in seconds).

Computing platform	Running time for lucky imaging	Time for image transmission	Total
Zhao's system[10]	2.45	41.94	44.39
Duan's system[21]	2.20	7.26	9.46
Proposed system	0.08	4.99	5.07

TABLE 4. FPGA on-chip resource consumption of lucky imaging for 2048 frames of cropped images.

Computing platform	Slices	Block RAMs
FastCam system[16]	3778	133
Proposed system	1350	21

of input images (128 × 128 pixels) at a rate of 1% image selection [10]. Running times for the three lucky imaging algorithms, for image transmission, and the total running times are listed in Table 3. Note: (1) Due to the limited processing capacity of Zhao's system, only 1,000 frame images are used to process for comparison here; (2) Since Zhao's system does not meet the real-time processing requirements, the comparison experiments for running time are conducted under the condition of simulated data transmission.

As seen from Table 3, the processing speed of the proposed lucky imaging algorithm is 2.45/0.08 = 31 times faster than that of Zhao's algorithm. This is because a new image selection algorithm and a new image storage method are adopted, and the data parallel and pipeline parallel characteristics of the FPGA are fully utilized in the design and implementation of the algorithm. The pixel receiving speed of this system is 41.94/4.99 = 8.4 times faster than that of Zhao's system. This is because this system adopts the GMII for image transmission instead of the SPI used in Zhao's system. However, due to the limitation of the FPGA development board used in the experiment, the GMII transceiver can only pack/unpack up to 1,500 bytes of packets at a time, resulting in a large number of interpacket time delays in the transmission process. It can be seen from Table 3 that it still needs 4.99 s to transmit 1,000 frames of 128×128 pixels. A solution to this problem is to use an FPGA development board with higher performance. The average running speed of this system is 44.39/5.07 = 8.76 times faster than that of Zhao's system.

With the help of high-speed data transmission of Gigabit Ethernet, compared with Zhao system, the total running time of Duan's system is greatly reduced, but the processing speed of lucky imaging is not improved substantially. This is because Duan's lucky imaging algorithm is the same as Zhao's algorithm and his main FPGA-based function modules are transplanted from Zhao's system. It can be seen from Table 3 that the lucky imaging algorithm and related implementation techniques proposed in this paper still have obvious advantages in the same image transmission mode.

F. FPGA ON-CHIP RESOURCE CONSUMPTION

The on-chip resource consumption of implementing the proposed algorithm is compared with that of FastCam [16]. When the FastCam system performs lucky imaging, the input images processed by FPGA are not the original format (512×512 pixels) images directly from the EMCCD camera, but cropped images of 128 × 128 pixels, and the registered images are selected from 2,048 frames of the small-size images. For comparison purposes, a special test system based on the proposed lucky imaging algorithm was implemented to ensure the two compared systems have the same image input. The experimental results are shown in Table 4.

As seen from Table 4, under the same processing conditions, the on-chip slices and block RAMs consumed by the proposed lucky imaging algorithm are only 35.73% and 15.79% of those by the FastCam. Obviously, the FPGA on-chip resource consumption has been greatly reduced when using the proposed algorithm.

From the above analysis and discussion, we can see that the proposed FPGA-based lucky imaging algorithm and the implemented system have good performance, but it is not perfect, and there are still some limitations. For example, although the input images are the original format (512 × 512 pixels) images of the EMCCD camera, in the registration process, the original images are still cropped and resized into small images of 64 × 64 pixels for superposition processing, and the resulting lucky image is finally displayed in a window of 64 × 64 pixels. In fact, because the proposed algorithm uses a DDR3 memory to store images temporarily, it can process EMCCD images with a larger format (such as 1k × 1k pixels). In addition, in the registration process, the images can be cropped into images of 128 × 128 or 256 × 256 pixels for stacking. This will display a larger target area and be more effective for observing dim objects in a multistar system or cluster. To achieve this goal, the superposition algorithm needs to be partially modified, that is, given that the block RAM resource on the FPGA is not sufficient, the images after superposition are stored in the DDR3 memory.

VI. CONCLUSION

On the basis of carefully analyzing the advantages and disadvantages of the FastCam, Zhao's lucky imaging algorithm and their implemented systems based on FPGAs, and taking full advantage of the data parallel and pipeline parallel characteristics of FPGA, this paper proposes a new real-time lucky imaging algorithm that contributes to real-time lucky imaging in the following five aspects: a dynamic sorting-free image selection algorithm with only a few comparisons, a registration and storage method for storing only a few registered images, a parallel superposition algorithm, a parallel preprocessing method for noise suppression and cosmic ray removal, and a dynamic multithreshold display method. In the specific implementation of the algorithm, parallel processing techniques and FPGA off-chip resources are adopted as much as possible, which affords the algorithm have a great advantage over the existing algorithms in terms of clock resources and on-chip resource consumption so that it can perform lucky imaging for more than 10,000 frames of the original images in real time. The experimental results show that the proposed algorithm is feasible and effective.

In addition, the design method and implementation techniques for the proposed algorithm based on an FPGA also have a certain universality, which can provide a reference for the design and implementation of other FPGA-based algorithms.

ACKNOWLEDGMENT
The authors would like to thanks Dr. X. Zhang, Prof. K. Ji, and Prof. Z. Jin from Yunnan Observatory, Chinese Academy of Sciences for their valuable comments and suggestions for this study. The original short-exposure images of the binary star HDS 70 used in this study are provided by Dr. Zhang.

REFERENCES
[1] W. Brandner and F. Hormuth, *Lucky imaging in astronomy* (Astrophysics and Space Science Library), vol. 439. Berlin, Germany: Springer, 2016, pp. 1–16.
[2] A. Oscoz, R. Rebolo, and R. Lopez, "FastCam: A new lucky imaging instrument for medium-sized telescopes," *Proc. SPIE*, vol. 7014, Jul. 2008, Art. no. 701447.
[3] O. Guyon, "Extreme adaptive optics," *Annu. Rev. Astron. Astrophys.*, vol. 56, no. 1, pp. 315–355, Sep. 2018.
[4] N. M. Law, C. D. Mackay, and J. E. Baldwin, "Lucky imaging: High angular resolution imaging in the visible from the ground," *Astron. Astrophys.*, vol. 446, no. 2, pp. 739–745, Feb. 2006.
[5] C. Mackay, "High-efficiency lucky imaging," *Monthly Notices Roy. Astronomical Soc.*, vol. 432, no. 1, pp. 702–710, Jun. 2013.
[6] C. D. Mackay, T. D. Staley, D. King, F. Suess, and K. Weller, "High-speed, photon-counting CCD cameras for astronomy," *Proc. SPIE*, vol. 7742, Jul. 2010, Art. no. 774202.
[7] C. D. Mackay, K. Weller, and F. Suess, "Photon counting EMCCDs: New opportunities for high time resolution astrophysics," *Proc. SPIE*, vol. 8453, Sep. 2012, Art. no. 845302.
[8] F. Faedi, T. Staley, Y. Gómez Maqueo Chew, D. Pollacco, S. Dhital, S. C. C. Barros, I. Skillen, L. Hebb, C. Mackay, and C. A. Watson, "Lucky imaging of transiting planet host stars with LuckyCam," *Monthly Notices Roy. Astronomical Soc.*, vol. 433, no. 3, pp. 2097–2106, Aug. 2013.
[9] M. Cortés-Contreras, V. J. S. Béjar, and J. A. Caballero, "CARMENES input catalogue of M dwarfs: II. High-resolution imaging with FastCam," *Astron. Astrophys.*, vol. 597, p. A47, Jan. 2017.
[10] P. Zhao, B. Li, L. Mao, and Y. Tao, "Implementation of lucky imaging algorithm base on field programmable gate array," (in Chinese), *Astronomical Res. Technol.*, vol. 16, no. 02, pp. 236–243, vol. 2019.
[11] C. Guo, H. Chen, and H. Sun, "Parallelism of in-memory sorting algorithm on modern hardware," (in Chinese), *Chin. J. Comput.*, vol. 40, no. 9, pp. 2070–2092, 2017.
[12] T. Browning, C. Jackson, and F. Cayci, "Hardware acceleration of lucky-region fusion (LRF) algorithm for high-performance real-time video processing," *Proc SPIE*, vol. 9451, Jun. 2015, Art. no. 94512G.
[13] J. Hegarty, R. Daly, and Z. Devito, "Rigel: Flexible multi-rate image processing hardware," *ACM Trans. Graph.*, vol. 35, no. 4, p. 85, vol. 2016.
[14] R. Stewart, K. Duncan, G. Michaelson, P. Garcia, D. Bhowmik, and A. Wallace, "RIPL: A Parallel Image Processing Language for FPGAs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 11, no. 1, p. 7, 2018.
[15] S. Wei, L. Liu, and S. Yin, *Reconfigurable computation*. Beijing, China: Science Press (in Chinese), 2014, pp. 185–188.
[16] J. Piqueras, L. Rodriguez-Ramos, and Y. Martin, "FastCam: Real-time implementation of the lucky imaging technique using FPGA," in *Proc. 4th Southern Conf. Program. Logic*, Mar. 2008, pp. 155–160.
[17] B. Hu and B. Li, "Electron multiplication model of EMCCD in low temperature," (in Chinese), *Acta Electronica Sinica*, vol. 41, no. 09, pp. 1826–1830, vol. 2013.
[18] L. Mao, B. Li, X. Zhang, K. Ji, and Z. Jin, "Experimental investigation of lucky imaging algorithm based on 2m astronomical telescope," (in Chinese), *Opt. Technique*, vol. 44, no. 05, pp. 542–548, 2018.
[19] K. Ji and F. Wang, "Two dimensional modified moment centering algorithm in CCD images," (in Chinese) *Acta Astronomica Sinica*, vol. 37, no. 01, pp. 85–90, 1996.
[20] S. Doppie. *00317+1929 HDS 70*. Accessed: Jun. 21, 2019. [Online]. Available: http://stelledoppie.goaction.it/index2.php?iddoppia=2129
[21] C. Duan, B. Li, Z, Chen, "Lucky imaging system on FPGA by using Ethernet transmission," *Proc. SPIE*, vol. 11189, Nov. 2019, Art. no. 111891Y.

JINLIANG WANG received the B.S. degree in electronic science and technology from the Hunan Institute of Engineering, in 2017. He is currently pursuing the master's degree with the Faculty of Information Engineering and Automation, Kunming University of Science and Technology. His current research interests include image processing and lucky imaging.

BINHUA LI received the B.S. degree in industrial automation of electrical engineering from the Jiangxi Institute of Technology (now Nanchang University), in 1984, the M.S. degree from the Department of Electrical Engineering, Chongqing University, in 1987, and the Ph.D. degree in astrophysics from the Graduate School, Chinese Academy of Sciences (Yunnan Observatory), in 2002. He is currently a Professor with the Faculty of Information Engineering and Automation, Kunming University of Science and Technology. His research interests include imaging technology and image processing, FPGA design and modern electronic systems, and astronomical telescope control and data acquisition.

KAI XING received the B.S. degree in electrical engineering and automation from Shanxi Agricultural University, in 2016. He is currently pursuing the master's degree with the Faculty of Information Engineering and Automation, Kunming University of Science and Technology. His current research interests include imaging technology and image processing.