IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 April 2025, accepted 29 May 2025, date of publication 2 June 2025, date of current version 9 June 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3575854

RESEARCH ARTICLE

A High-Speed, Multi-Channel Lossless Compression Algorithm for High-Resolution Video on FPGA

XINGKAI DU[1], LONGHUA XIE[1], HONGCHUAN HUANG[1], HUAWEI CHEN[1], XIAOLONG GUO[2], AND TINGYU ZHAO[1]

[1]Zhejiang Key Laboratory of Quantum State Control and Optical Field Manipulation, Department of Physics, Zhejiang Sci-Tech University, Hangzhou 310018, China

[2]Hangzhou ToupTek Photonics Company Ltd., Hangzhou 310030, China

Corresponding author: Tingyu Zhao (zhaotingyu@zstu.edu.cn)

ABSTRACT With the increasing image resolution, data bit depth, and frame rate of industrial cameras, the signal output speed needs to be improved to keep pace with the rapidly developing image sensor technologies. The USB 3.0 interface has become a common transmission method for industrial cameras due to its high versatility. However, limited bandwidth often requires data compression techniques to address transmission bottlenecks. This paper proposes a four-channel high-speed lossless compression algorithm based on FPGA (Field-Programmable Gate Array). The algorithm uses the difference between adjacent pixels, which has fewer data bit widths, to replace the original pixels for transmission. This approach enhances the transmission frame rate while maintaining image quality. Additionally, a novel ''first-in, last-out'' data inversion technique is employed to solve the problem of real-time processing of four-channel data, significantly reducing FPGA hardware resource usage. The performance of the proposed algorithm is evaluated through on-board FPGA experiments. When transmitting 4096 × 4096 resolution video via USB 3.0, the frame rate reaches 18.3 fps with a data size reduced by up to 49.98% (corresponding to a compression ratio of 50.02%), with an encoding overhead delay of 20.76 μs, resulting in a frame rate increase of approximately 71.03%. Additionally, decompression speeds for a large resolution image set (5568 × 3712) are improved by at least 40% compared to mainstream algorithms. The algorithm demonstrates low FPGA resource utilization and high processing speed, ensuring efficient compression and transmission even for complex image scenes. These results provide a viable high-speed lossless compression solution for industrial camera applications.

INDEX TERMS Field programmable gate arrays, cameras, image compression, compression algorithms, universal serial bus.

I. INTRODUCTION

Field-Programmable Gate Array (FPGA) achieves efficient pipeline processing and data stream optimization through custom logic circuits, offering significant advantages in image processing efficiency. Industrial cameras based on FPGA, known for their high reliability and intelligent integration, are playing an increasingly important role in the rapidly developing industrial production sector. With the continuous advancements in image sensor technologies, the output signal speed of industrial cameras must be continuously improved to match the increasing resolution, data bit depth, and frame rate of the output images. Camera Link and USB 3.0 are the commonly used transmission interfaces for industrial cameras. Camera Link, a high-bandwidth transmission protocol, requires specific PCIe (Peripheral Component

The associate editor coordinating the review of this manuscript and approving it for publication was Md. Moinul Hossain.

Interconnect Express) slots and cables, limiting the system's versatility and portability [1]. The USB 3.0 interface, widely available in personal computers, is highly flexible and versatile, making it a popular transmission interface for industrial cameras. However, the theoretical maximum transmission rate of USB 3.0 is only 5 Gbps [2], which is insufficient to effectively support the uncompressed transmission of high-resolution and high-frame-rate video. Therefore, image compression technology has become an essential solution to address this transmission bottleneck and enable faster image data transfer with guaranteed image quality.

Data compression can be classified into lossy and lossless compression based on fidelity [3]. Lossy compression refers to a method in which the data decoded at the receiver end differs from the original data, leading to a loss of some information. Common lossy compression standards include JPEG [4] and JPEG2000 [5] for static images, and MPEG-4 [6], H.264 [7], and H.265 [8] for dynamic videos. Compression ratio, the ratio between the size of the compressed data and the size of the original data, is an important metric to measure the effectiveness of data compression. Lossy compression typically achieves lower compression ratios but may lose some image details, making it unsuitable for industrial cameras that require high image resolution.

Lossless compression, while achieving higher compression ratios than lossy methods [9], preserves the integrity of the original data and is the preferred compression technique for industrial cameras. Classical algorithms include Huffman coding [10], LZ77 [11], and LZW [12]. While some of these algorithms result in significant data reduction, they are complex to implement, have long overall runtime, and require significant computational resources, making them unsuitable for deployment on FPGA-based industrial cameras, which have limited processing power and strict real-time video requirements [13]. For example, Huffman coding, and the PNG [14] algorithm derived from LZ77, along with lossless compression algorithms for ultra-large-scale integrated circuits [15], can be inefficient for real-time video processing. Huffman coding, for instance, relies on fixed datasets to generate encoding trees. As the data stream changes continuously in video, the initial encoding tree becomes unsuitable, increasing the complexity of algorithm implementation. Therefore, computational efficiency is a crucial factor for the feasibility of image compression algorithms on FPGA platforms. For example, JPEG-LS [16], an operation mode of JPEG, is a lossless or near-lossless image compression standard. Its low complexity makes it an ideal candidate for deployment on FPGAs [17]. However, it may cause banding artifacts in regions with smooth images, which affects image quality [18]. LZW compression coding is an efficient encoding method that can be combined with predictive coding to further reduce data size. However, this encoding method, which is based on dictionaries or predictive models, often requires additional RAM (Random-Access Memory) to store the dictionary. This means it either needs a relatively resource-rich FPGA chip or external memory chips, leading to extra hardware costs. Furthermore, its reliance on the data stream and the complexity of the encoding/decoding process limit compression performance.

Mainstream FPGA-friendly compression algorithms often rely on look-up tables or dictionaries, requiring extra off-chip storage and increasing resource use, which hurts processing speed [19].

To address the data dependency issues in compression algorithms, video stream compression algorithms based on the high correlation between pixels [20] have been developed. These algorithms achieve low resource usage while ensuring real-time image processing. However, due to the complexity of the encoding process, they can only handle dual-channel data, with no suitable methods for processing or extending to four-channel data. This limitation severely restricts the output frame rate of the sensor and, consequently, the video bitrate after decoding.

This paper proposes a multi-channel real-time lossless compression algorithm based on FPGA, offering the following key contributions.

First, the approach divides the image into subregions and processes pixel data in parallel across multiple channels. This design significantly improves the camera's processing speed and output capacity.

Second, the algorithm employs differential transmission to exploit the high similarity between adjacent pixels. Instead of transmitting the original pixel values, it transmits only the difference values, which require fewer bits. This method effectively reduces data size while ensuring lossless reconstruction at the receiver end.

Third, the design incorporates a FILO (First-In-Last-Out) structure combined with a ping-pong operation to efficiently handle multi-channel data. This strategy reduces the logical processing load of the algorithm and optimizes

FPGA hardware resource utilization. The proposed algorithm demonstrates excellent compression performance when transmitting video at 4096 $\times$ 4096 resolution via USB 3.0. The encoding overhead delay is only 20.76 μs, with a minimum compression ratio of 50.02%. Additionally, the frame rate reaches 18.3 fps, representing a 71.03% improvement over the original 10.7 fps. Experimental results also indicate that the proposed algorithm exhibits excellent compression performance across different resolutions and objects, making it applicable in fields such as machine recognition [21], [22]. By improving the video frame rate and providing more image data, the algorithm enhances the accuracy of recognition.

II. COMPRESSION ALGORITHM PRINCIPLE

In industrial cameras, the basic unit commonly used for video image transmission is the byte. When higher image quality is required, image data with pixel bit depths of 10-bit, 12-bit, or even 16-bit are typically transmitted. However, since one byte contains only 8 bits, transmitting a complete pixel requires at least two bytes.

We observe that due to the strong correlation between adjacent pixels in an image, the difference between neighboring pixels often does not exceed 8 bits (i.e., 2^8 =256). Therefore, instead of transmitting the original pixel values directly, we propose transmitting only the differences between adjacent pixels. This approach enables the transmission of two bytes of information using just one byte, effectively reducing the total data required to transmit a single frame. Consequently, under the same transmission bandwidth, this differential transmission method can achieve a higher frame rate while maintaining the same image resolution and quality.

Reference [20] introduces an image compression algorithm that applies differential encoding by selecting a fixed reference pixel and computing the difference between each subsequent pixel and this reference. However, in this approach, as new pixels are progressively processed, their distance from the reference pixel increases, thereby weakening their correlation. This limitation prevents the algorithm from fully exploiting the redundancy between neighboring pixels. To address this, we propose an improved method that considers only two adjacent pixels during compression encoding, linking pixels sequentially.

Theoretically, this approach better preserves the correlation between neighboring pixels, leading to improved compression performance. Additionally, the proposed algorithm performs multi-channel compression processing and implements a ping-pong FILO (First-In-Last-Out) operation to realize the core counting function.

The smallest processing unit in this encoding relationship is called the compression kernel, which consists of a kernel length, a reference, and the differences between adjacent pixels. A single row of an image can contain multiple compression kernels. The process of constructing a compression kernel is shown in Algorithm 1: First, a compression threshold (*threshold*) is set (line 1). The starting pixel of each row is always used as the reference pixel for creating a compression kernel (lines 8-9).The difference (*dif*) between the previous pixel (*prev*) and the current pixel (*p*) is then computed (line 12). If the difference falls within the threshold range, the difference is stored as an element of the compression kernel and the length (*len*) of the kernel is increased by one (lines 13-15). Otherwise, the length (*len*) of the previous compression kernel is recorded, and the pixel that exceeds the threshold is used as the reference pixel for a new compression kernel, creating a new kernel (lines 17-19). These steps are repeated until the entire row has been processed. This algorithm effectively compresses video images by sequentially comparing pixels and storing the shorter data differences instead of the original pixel values, thereby achieving image compression.

Algorithm 1 Compression Algorthim

```
1: Input: Image data, threshold
2: for each row in the image do
3:     prev ← None
4:     difs ← empty array
5:     ref ← None
6:     len ← 0                                   ▷ Kernel length
7:     for each pixel p in the row do
8:         if prev = None then                   ▷ First pixel of the row
9:             ref ← p
10:            len ← 1
11:        else
12:            dif ← | p − prev |                ▷ Compute difference
13:            if dif ≤ threshold then
14:                Append dif to difs
15:                len ← len + 1
16:            else   ▷ Reset reference, length, and diff array
17:                ref ← p
18:                len ← 1
19:                diffs ← empty array
20:            end if
21:        end if
22:        prev ← p
23:    end for          ▷ Handle the last kernel of the row
24: end for
```

III. IMAGE COMPRESSION ENCODING MODULE

The compression kernel, as the smallest compression unit, serves to store and transmit a row of image data in a predefined compression format. The image compression encoding module consists of three submodules: the calibration submodule, the length statistics submodule, and the encoding submodule. The compression kernel length statistics submodule primarily consists of the FILO (First-In-Last-Out) component and the counting component. The structure diagram of the image compression encoding module is shown in FIGURE 1.

A. THE CALIBRATION SUBMODULE

Each compression kernel has an independent reference pixel, and the calibration of the compression kernel refers to determining the position and the value of the reference pixel, which are then processed by the subsequent modules. The input to this module is the raw image data.

The dual-channel calibration mechanism proposed in Reference [20] is extended to a multi-channel architecture to enhance transmission frame rate, and a calibration logic based on adjacent pixels has been introduced. The calibration logic is systematically elaborated through a representative 4-channel configuration as a case study.

FIGURE 2 illustrates the multi-channel and calibration signal indicators. The original pixel data is valid on the rising edge of the clock signal and changes on the falling edge. The pixel data for each clock cycle consists of data from several pixel data transfer channels. For example, the pixel data Data_n at the n^{th} clock cycle is composed of pixel data from k data channels, from data_1ch to data_$k$ch. A binary calibration sequence register flag[k-1:0] with a bit width of k is set up. The bits of the register correspond one-to-one with

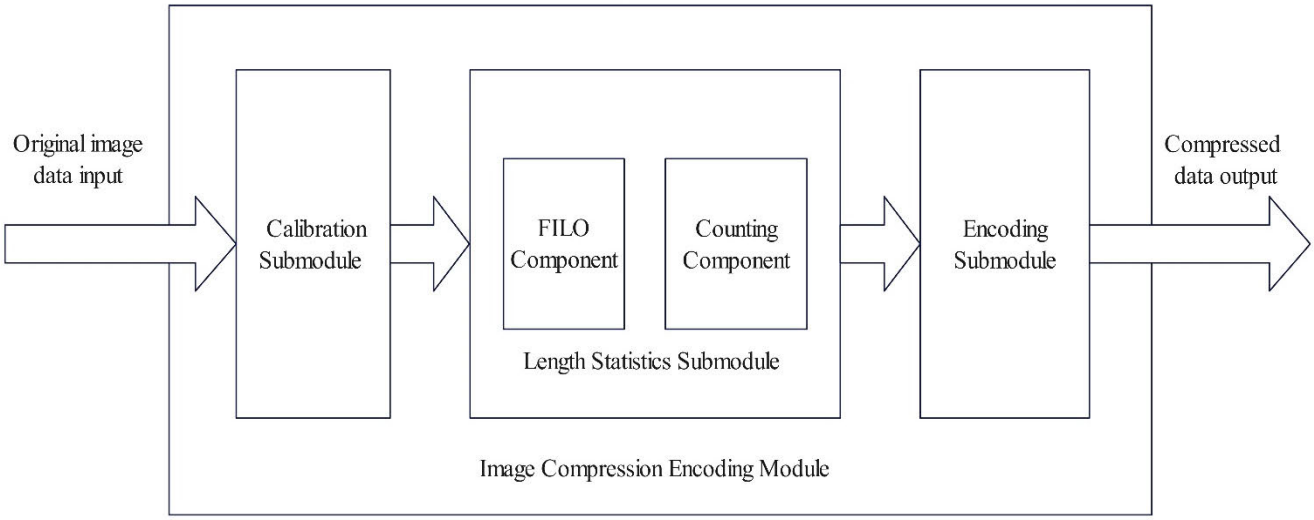

FIGURE 1. The structure diagram of the compression encoding module.

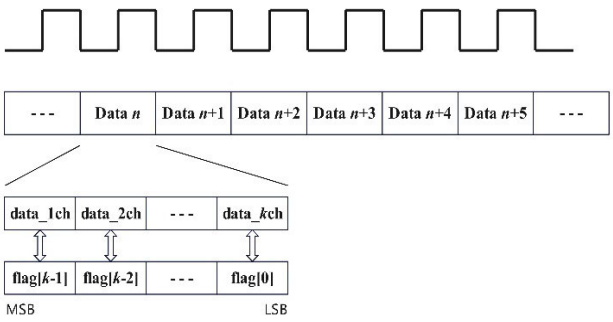

FIGURE 2. Schematic diagram of the multi-channel and calibration signal indicators.

the data channels, where the least significant bit (LSB), which is the rightmost bit flag[0] corresponds to the k^{th} data channel data_$k$ch, and the most significant bit (MSB), which is the leftmost bit flag[k-1] corresponds to the first data channel data_1ch.

Four-channel transmission is used as an example to balance the processing speed, the resource consumption, and the bandwidth limitations of USB 3.0 [23], [24], [25]. A 4-bit register is utilized as the marking sequence as shown in FIGURE 2. In the marking sequence, "1" represents the reference pixel, and "0" represents the ordinary element of the compression kernel. When a row of raw image data arrives, the first pixel of each row is forced to be marked as "1". The next pixel is determined by the difference between the next pixel value and the current pixel value: if the difference exceeds the threshold, the next pixel, marked as "1", become the reference pixel of the new kernel; otherwise, it is marked as "0".

The threshold for the difference in the module is set to 127. An example of the simulated output of the compression kernel calibration submodule is shown in FIGURE 3, where "clk" is the driving clock signal, and "lv" is the valid signal of the row data. "data_1ch", "data_2ch", "data_3ch", and "data_4ch" represent the data from the first to the fourth data channels in one clock cycle, with each pixel having a data width of 12 bits, displayed in hexadecimal format. "flag" is the compression kernel calibration signal, with a data width of 4 bits, displayed in binary format.

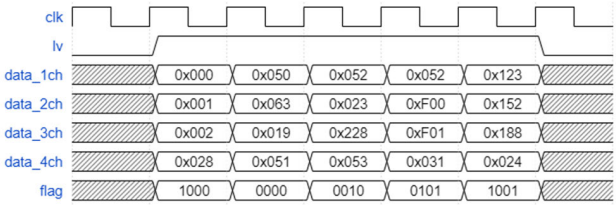

FIGURE 3. A data processing example of the calibration submodule.

When the row valid signal "lv" arrives, the first pixel has a value of "0 × 000". According to the calibration rule, the highest bit of the "flag", i.e., the third bit, is set to "1". The second pixel has a value of "0 × 001", and the difference between the first and second pixels is within the threshold range, so the second bit of "flag" is set to "0". The third pixel has a value of "0 × 002", and the difference between the second and third pixels is also within the threshold range, so the first bit of "flag" is set to "0". The fourth pixel has a value of "0 × 028", and the difference between the third and fourth pixels is within the threshold range, so the least significant bit of "flag", i.e., the 0^{th} bit, is set to "0". Therefore, the output signal "flag" for the first clock cycle is "1000".

This process continues until the "lv" signal goes low at the end of the row, marking the completion of processing for that

row of data. The system then waits for the next row of data to arrive.

B. LENGTH STATISTICS SUBMODULE

A hybrid methodology integrating FILO (First-In-Last-Out) and ping-pong operations is proposed to address the computational challenges associated with compressed kernel length statistics. Compared to conventional FIFO (First-In-First-Out)-based approaches, this method is demonstrated to overcome the inherent counting limitations of kernel length quantification, thereby resolving inefficiencies in memory space utilization and achieving significant conservation of logic resources.

As shown in FIGURE 1, the length statistics submodule, consisting of the FILO component and the counting component, is responsible for calculating the length of each compression kernel to be used during the decoding process. The input is the marking sequence output by the calibration submodule.

The length of the compression kernel is calculated by counting the number of "0" following "1" in the marking sequence. Due to the unpredictability of the data stream, it is impossible to foresee the length of the compression kernel. Therefore, FILO is innovatively used to reverse the data flow, turning the problem of predicting the compression kernel length into counting the number of zeros that appear before "1". In this way, the length of the compression kernel can be sequentially calculated based on the reversed data flow.

FIFOs can be customized in capacity to save resources depending on the amount of data. A ping-pong operation [26] is incorporated on top of the FILO structure to achieve continuous data flow and further increase the data transfer speed. More specifically, two FILOs are constructed, where one is written with data in odd rows while the other is simultaneously read from for data in even rows, vice versa.

The reversed sequence, used for counting the compression kernel length, is subject to a ping-pong operation through FILO. Therefore, the output data of the length statistics submodule will be the marking sequence containing the compression kernel length information.

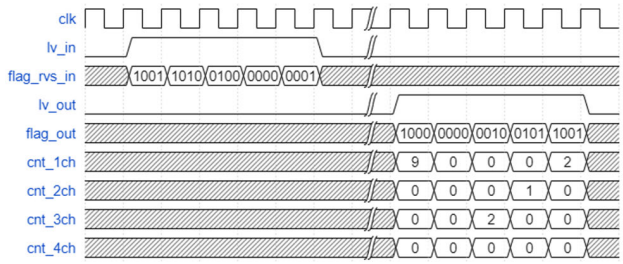

FIGURE 4. A data processing example of the length statistics submodule.

FIGURE 4 demonstrates an output example for four channels. The signal "flag_rvs_in" is the reverse-marked sequence after the operation of FILO. The output marking sequence "flag_out" is the forward-marked sequence after several clock cycles of data processing. The compressed kernel lengths corresponding to the four channels of data are output simultaneously.

As shown in FIGURE 4, the output marking sequence "flag_out" is "1000", "0000", "0010", "0101", "1001", when the row valid signal "lv_out" is high. The first "1" is followed by nine "0"s, thus the counter for channel 1 "cnt_1ch" output "9" and the rest of the counters output "0" during the first clock cycle. The output signal sequence is "0000" in the second clock cycle, i.e., all four-pixel data come from the difference sequence in the compression kernel. Therefore, all the counters output "0". Similarly, the output marking sequence "0010" causes the "cnt_3ch" counter to output "2" in the third clock cycle. Following the same logic, the values of each counter during other clock cycles can also be obtained as shown in FIGURE 4. On one hand, the output value for the corresponding channel should be ignored when the output marking sequence is "0". However, this output is defined as "0" for aesthetic reasons. On the other hand, the corresponding channel's length value is "0" if the output marking sequence is "1". It means that the compression kernel only contains a reference pixel but no pixel coming from the difference sequence. This can occur if the difference between the reference pixel and its adjacent pixels exceeds the threshold, or if the pixel is at the end of the row. There is a difference in the meaning of the "0" output in the above two cases.

By combining the ping-pong operation with FILO, the unpredictable data flow is reversed and becomes statistically manageable, allowing for fast data processing. The compression kernel length sequence corresponding to each "1" in the output marking sequence indicates the length of the respective compression kernel.

The output of the length statistics submodule is output a marking sequence and a compression kernel length sequence.

C. ENCODING SUBMODULE

This module integrates the data processed by the previously described submodules based on the characteristics of the proposed algorithm and proposes a new compact encoding sequence. The encoding module fully utilizes the characteristics of low-bit-width data and encodes the data by identifying the input marker sequence.

FIGURE 5 illustrates the encoding structure for a row. There are N ($N \geq 1$) compression kernels in one row processing time. The lengths of these compression kernels may not be equal as reflected in the varying kernel widths in FIGURE 5. Each compression kernel consists of three parts: the compression kernel length, the reference pixel, and the pixel differences, where the pixel difference data has a shorter bit width than the original pixel data. The start of a new compression kernel, i.e., "1" in the marking sequence, is detected, the length sequence (Length M) is placed as the first data of the compression kernel. The current pixel, i.e., the reference pixel (Pix), is placed as the second data of the compression kernel. The pixel difference data (Diff 1 to Diff M) are sequentially

placed in the subsequent encoding sequence when M "0" is detected in the marking sequence.

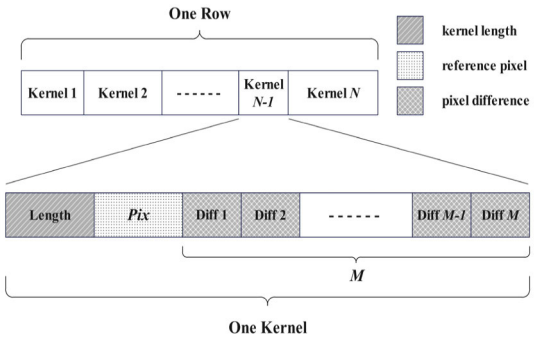

FIGURE 5. Diagram of the encoding submodule for a row.

Based on the previously described compression principle, we now proceed with the calculation of the compression ratio. Since the proposed algorithm performs encoding on a row-by-row basis, the compression ratio is also evaluated at the row level.

As shown in Figure 5, suppose a row contains m pixels, each with a bit width of p-bit, and consists of N kernels. Each kernel is composed of a data header and pixel difference values. The data header includes a compression kernel length value w and a reference pixel value, where the bit width of each Kernel's data header is uniform, occupying a total of $(p + w) \times N/8$ bytes in the row (1 byte = 8 bits).

Notably, each kernel has exactly one reference pixel value, meaning that N kernels correspond to N reference pixels, while the remaining $m - N$ pixels are represented as pixel difference values. Since each pixel difference value has a bit width of 1 byte, the total bit width occupied by all pixel difference values in the row is $(m - N)$ bytes.

In summary, after compression, the total data size in bytes of the entire row is:

$$(p + w) \times N/8 + (m - N)$$

Taking a RAW12 format image with a resolution of 4096 × 4096 as an example, the pixel bit width p is 12, and the number of pixels per row m is 4096. In the optimal case, since $2^{12} = 4096$, setting $w = 12$ is sufficient to cover the entire row of pixels. Substitute these data into the aforementioned equation, the total data size for one compressed row is 4098 bytes. For comparison, the uncompressed data size of a single row is $4096 \times 2 = 8192$ bytes. Thus, the compression ratio is 4098 / 8192. the compression ratio can approximate 50.02% in the optimal scenario.

However, the algorithm also encounters another extreme case: The total bit-width of the compression kernel length and reference pixel may be greater than the original pixel bit-width if the image exhibits highly frequent variations. This leads to an excessive number of compression kernels per row. Then the total data size after compression could exceed that of the original image. In such cases, the algorithm will fall back to transmitting the original image data, ensuring the effectiveness and stability of the compression scheme.

IV. EXPERIMENTAL RESULTS

A. EXPERIMENTAL RESULTS OF NOISE ROBUSTNESS

The experiment uses a self-captured image, as shown in the left image of FIGURE 6. The image has a high resolution of 5568 × 3712 and contains various lighting and gradient areas. It also includes complex edges and shapes, such as glass reflections, walls, and windows, making it suitable as the test image for this study. The salt-and-pepper noise, which randomly generates black and white noise points, simulates the interference caused by industrial cameras during image transmission and data processing.

The salt-and-pepper noise is gradually increased by percentage, and the corresponding changes in image compression ratio are recorded and plotted as a curve as shown in FIGURE 7. A compression ratio of "1" indicates that the encoded data size is greater than the original image data size, in which case the original image data is used. It can be observed that when the salt-and-pepper noise ratio reaches 16%, the encoded data size of the proposed algorithm exceeds that of the original image data. The right image in FIGURE 6 shows the image when the salt-and-pepper noise ratio is 16%. Compared to the original image on the left, it is evident that the image contains a significant amount of noise at this level, which severely interferes with the original image information. These noises are likely introduced by hardware design, indicating that the captured image is in an unusable state at this point. Therefore, it can be concluded that the proposed algorithm is capable of achieving effective compression when the salt-and-pepper noise ratio is relatively low and the original image is not severely affected.

It should be noted that while the relationship between compression ratio and noise rate is exemplified in FIGURE 7 using the image from FIGURE 6, this correlation is demonstrated to exhibit a universal trend across other compressed images.

B. COMPARATIVE EXPERIMENTAL RESULTS

To further evaluate the performance of the proposed algorithm in terms of compression ratio and decoding speed, comparative experiments with several classical algorithms were conducted. Since the proposed algorithm is a lossless compression method, the compared algorithms include the Xie's algorithm [20], PNG [14] algorithm, and JPEG-LS [16].

To assess the general applicability of the proposed algorithm across different scenarios, an open-source image dataset consisting of 856 images was obtained from the Roboflow website [27]. The dataset includes images of various categories, such as streets, buildings, people, objects, and landscapes, all with a resolution of 640 × 640 pixels. The proposed algorithm, Xie's Algorithm, PNG, and JPEG-LS algorithms were used to compress and decompress these images, and the compression ratios and decoding speeds

FIGURE 6. Test experiments (left) and the image with 16% salt-and-pepper noise added (right).

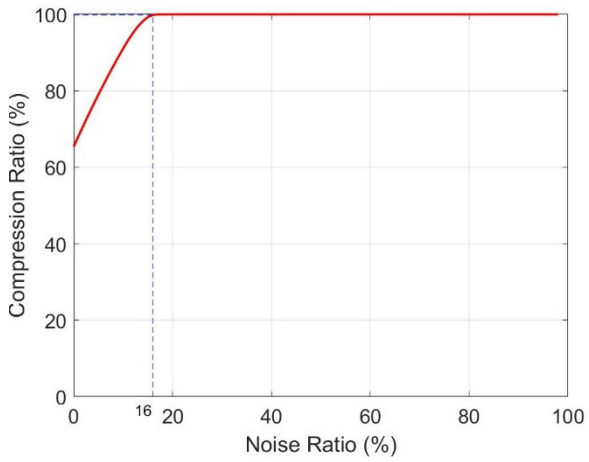

FIGURE 7. Compression ratio variation with noise level.

were compared. For the PNG compression algorithm, the *imencode()* function from the OpenCV library was used. To accelerate the processing speed, the compression ratio parameter for the.png format was set to "1", resulting in the shortest compression time and the fastest compression and decompression speeds. This setting was maintained throughout the subsequent experiments. The results of these experiments are summarized in Table 1.

TABLE 1. Comparison of compression ratio and PC-side decompression speed for original 640 × 640 resolution.

Algorithm	Average compression ratio(%)	Average decompression frame rate(fps)
Proposed Algorithm	76.80	274.53
Xie's Algorithm[20]	82.91	284.60
PNG[14]	49.91	134.42
JPEG-LS[16]	24.63	154.47

As shown in Table 1, for images with smaller original sizes, although PNG and JPEG-LS achieve lower compression ratios, they sacrifice decoding speed, with their decoding speeds being approximately half of that of the proposed algorithm. Xie's Algorithm has the fastest decoding speed, but the proposed algorithm is only 3.54% slower, while achieving a 7.37% improvement in average compression ratio.

Given that deploying video compression algorithms for industrial cameras to process high-resolution video streams is more meaningful, and since there are fewer publicly available datasets for high-resolution images, this study uses a self-captured image dataset for testing. The images, with a resolution of 5568 × 3712, suitable for high-resolution image requirements. The dataset covers a wide range of subjects, including people, animals, plants, architecture, streets, and objects. Efforts were made to avoid large areas of blank space or uniform regions to better assess the compression capabilities of the algorithms. This dataset includes 120 images, which were processed and compared while maintaining the original resolution. The experimental results are summarized in Table 2.

TABLE 2. Comparison of compression ratio and PC-side decompression speed for high-resolution images.

Algorithm	Average compression ratio(%)	Average decompression frame rate(fps)
Proposed Algorithm	61.87	5.98
Xie's Algorithm[20]	69.94	5.39
PNG[14]	53.06	4.26
JPEG-LS[16]	22.92	3.30

As shown in Table 2, when the image resolution is increased, the proposed algorithm achieves a slightly lower average compression ratio compared to PNG and JPEG-LS. However, it demonstrates the highest average decompression

frame rate: a 40.38% improvement over the PNG algorithm and an 81.21% improvement over the JPEG-LS algorithm. Compared to Xie's algorithm, the proposed algorithm has achieved significant improvements in both compression and decompression: the average decompression frame rate has increased by 10.95%, and the average compression ratio has decreased by 11.54% (i.e., resulting in more efficient compression).

In the comparative experiments with two image datasets, the proposed compression algorithm did not result in any instances where the compressed image size exceeded the original size, indicating that effective compression was achieved for all images in both datasets.

It can be concluded that the proposed compression algorithm can effectively compress images with minimal noise interference, exhibiting a certain degree of noise immunity. It also provides stable and efficient compression across different scenes and objects, and demonstrates good compression performance for high-resolution images. Regardless of the image resolution, the proposed algorithm offers significant decompression speed improvements over the well-established PNG and JPEG-LS algorithms. Compared to the Xie's Algorithm, our algorithm shows a 7.37% improvement in compression ratio for low-resolution images while maintaining a similar decompression speed. For high-resolution images, the proposed algorithm achieves a 115.4% increase in decompression speed and a 10.59% improvement in compression ratio.

C. BOARD-LEVEL VERIFICATION

Board-level verification is conducted to evaluate the proposed compression algorithm's ability to handle video streams. An Efinity-based T85 series FPGA hardware platform is used for the test, with the Onsemi PYTHON 25K image sensor selected for capturing the video data. DDR3 is employed as the video data cache, and the compressed data are transmitted via a USB 3.0 interface to a PC, where the data are decoded and displayed.

The camera's video resolution is set to 4096 × 4096 in the FPGA code, with compression and encoding handled over four channels. The video transmission frame rate is 10.7 fps (frames per second), and the compression rate is defined as the ratio of the compressed data to the data volume of raw video transmitted in RAW 12 format. Table 3 compares the resource utilization and video frame rate between two-channel and four-channel image compression encoding. The video frame rate doubled with a small increase in FPGA resource usage with four channels compared to two channels.

Table 4 compares the resource usage of the original data and the deployment of the compression algorithm on the FPGA. Compared to the scenario without deploying the compression algorithm, the logical resource utilization rate and the storage block resource utilization rate of the compression algorithm have increased by 8.42% and 12.37% respectively. The overall increase in resource usage is minimal.

Table 5 presents FPGA clock cycles consumed by each module of the proposed compression algorithm applied to processing and encoding video streams with a resolution of 4096 × 4096. Since the proposed algorithm processes data

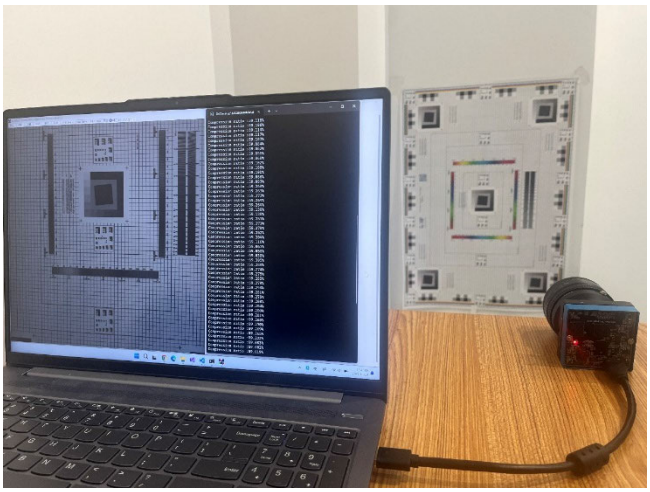

FIGURE 8. Experimental setup for image capture.

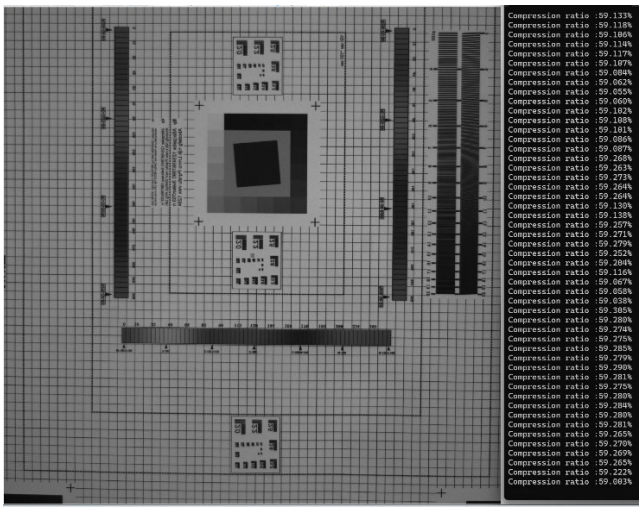

FIGURE 9. Image captured by the camera and real-time compression rate.

TABLE 3. Resource utilization and video frame rate comparison of 2-channel and 4-channel processing.

Number of Channels	Processing Clock Frequency (MHz)	Resource Utilization		Frame Rate (fps)
		Logic Elements	Memory Blocks	
2	100	20377	269	5.3
4	100	22044	329	10.7

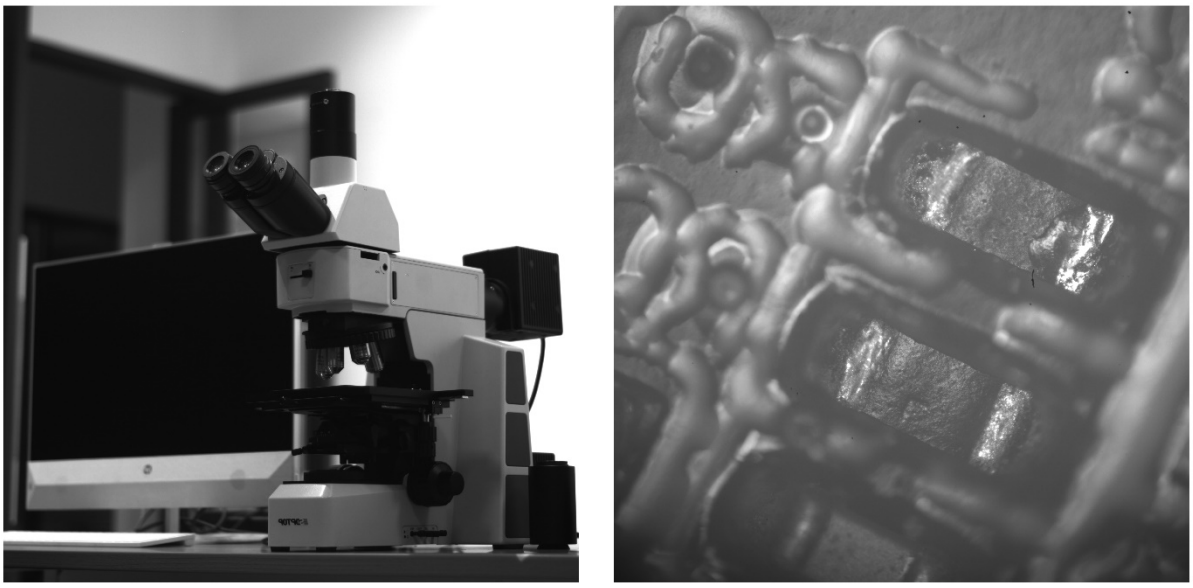

FIGURE 10. Experiment table desktop scene (left)and microscope view of the circuit board components (right).

TABLE 4. FPGA resource utilization.

Resource	Total Available Resources	Original Deployment (as a Percentage of Total Resources)	Compression Algorithm Deployment (as a Percentage of Total Resources)
Logic Elements	84096	14960 (17.79%)	22044 (26.21%)
Memory Blocks	792	231 (29.17%)	329 (41.54%)

TABLE 5. FPGA clock cycles consumed by each module for encoding one frame of a 4096 × 4096 image.

Encoding Process Module	Time / FPGA Clock Cycles	Occupied Proportion(%)
Compression Kernel Calibration	3	1.445
Compression Kernel Length Statistics	2059	99.181
Compression Kernel Encoding	14	6.744
Total Time	2076 (20.76μs)	

based on row signals within a single frame, the processing time is measured per row. The FPGA data processing clock is set to 100MHz. The encoding overhead delay for processing a single row of video at this resolution is 20.76μs, demonstrating the algorithm's ability to meet the requirements for fast processing. Since the proposed compression algorithm performs encoding on a row-by-row basis, the compression time is approximately proportional to the number of pixels per row. As a result, when processing lower-resolution images, the algorithm exhibits a shorter encoding overhead, leading to improved efficiency.

FIGURE 8 shows the experimental setup and environment for the image capture. FIGURE 9 displays the 4096 × 4096 video frame captured by the camera along with its real-time compression rate. The proposed algorithm effectively achieves a compression rate of approximately 59.2% when dealing with ultra-high-resolution images and complex scenes such as resolution test charts, which feature dense black-and-white distributions.

Additionally, several scene captures were selected for testing. The left image in FIGURE 10 shows the laboratory desk scene, which features a simple background and a single subject. In this case, the compression rate reached 50.8%. The right image in FIGURE 10 simulates a scenario where a microscope is used to inspect the soldering quality of circuit board components. Under the microscope, the surface of the solder, with its uneven contours and reflections from light, causes high-frequency variations in the image. Even in this more complex scene, where there are many fine details, the camera with the proposed compression algorithm achieves a compression rate of 51.6%.

The experimental results demonstrate that the decoded images retain high resolution, making the lossless algorithm suitable for applications in industrial inspection and other scenarios where high image quality is required.

V. CONCLUSION

This paper proposes a novel FPGA-based high-speed lossless video compression algorithm tailored for the evolving USB 3.0 industrial cameras. The algorithm takes advantage of

the similarity between adjacent image pixels by replacing the original pixel values with shorter bit-depth differences between adjacent pixels. It also addresses the issue of real-time data processing for four-channel data by employing a FILO data buffer and processing method, thus improving the video transmission frame rate on industrial cameras. The proposed algorithm requires minimal FPGA resource usage and does not rely on external memory devices for dictionary storage or look-up, enabling high-speed encoding. Software simulation experimental results show that the algorithm provides stable compression rates across various images and exhibits a certain level of noise immunity. For high-resolution images of 5568 × 3712, the proposed algorithm outperforms established algorithms with a 40.38% improvement in decompression speed compared to the PNG algorithm, an 81.21% improvement compared to the JPEG-LS lossless compression algorithm, and an 11.54% improvement over the algorithm proposed in [20]. Additionally, the proposed algorithm achieves a 10.95% improvement in compression ratio over the method in [20]. In FPGA-based onboard experiments, the algorithm demonstrates good compression performance for both common and complex images at a resolution of 4096×4096. The theoretical minimum compression ratio for a single frame is 50.02%, with an encoding overhead delay of 20.76μs, providing both strong compression performance and high-speed encoding/decoding capabilities, offering significant practical engineering application value.

REFERENCES

[1] Y. Li, K. Song, K. Zhong, J. Liang, and H. Liu, "Design of high-speed data transmission system based on USB3.0," *IEEE Trans. Nucl. Sci.*, vol. 70, no. 6, pp. 1090–1095, Jun. 2023, doi: 10.1109/TNS.2023.3240542.

[2] J. Huang and Y. Wang, "Research and application of high-speed and adjustable synchronous data transfer system based on USB 3.0 peripheral controller," *J. Circuits, Syst. Comput.*, vol. 30, no. 7, Jun. 2021, Art. no. 2150118, doi: 10.1142/s0218126621501188.

[3] R. M. Thanki and A. Kothari, "Classification in data compression," in *Hybrid and Advanced Compression Techniques for Medical Images*. Cham, Switzerland: Springer, 2019, pp. 17–29, doi: 10.1007/978-3-030-12575-2_2.

[4] M. Liu, H. Fan, K. Wei, W. Luo, and W. Lu, "Adversarial robust image steganography against lossy JPEG compression," *Signal Process.*, vol. 200, Nov. 2022, Art. no. 108668, doi: 10.1016/j.sigpro.2022.108668.

[5] Z. Zhang, Q. Sun, W.-C. Wong, J. Apostolopoulos, and S. Wee, "An optimized content-aware authentication scheme for streaming JPEG-2000 images over lossy networks," *IEEE Trans. Multimedia*, vol. 9, no. 2, pp. 320–331, Feb. 2007, doi: 10.1109/TMM.2006.886281.

[6] P. He, X. Jiang, T. Sun, and S. Wang, "Detection of double compression in MPEG-4 videos based on block artifact measurement," *Neurocomputing*, vol. 228, pp. 84–96, Mar. 2017, doi: 10.1016/j.neucom.2016.09.084.

[7] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003, doi: 10.1109/TCSVT.2003.815165.

[8] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012, doi: 10.1109/TCSVT.2012.2221191.

[9] V. Muddhasani and M. D. Wagh, "Bilinear algorithms for discrete cosine transforms of prime lengths," *Signal Process.*, vol. 86, no. 9, pp. 2393–2406, Sep. 2006, doi: 10.1016/j.sigpro.2005.10.022.

[10] D. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952, doi: 10.1109/jrproc.1952.273898.

[11] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Inf. Theory*, vol. IT-23, no. 3, pp. 337–343, May 1977, doi: 10.1109/TIT.1977.1055714.

[12] Welch, "A technique for high-performance data compression," *Computer*, vol. 17, no. 6, pp. 8–19, Jun. 1984, doi: 10.1109/mc.1984.1659158.

[13] J. Cardoso. (2019). *Verilog PNG Encoder*. Accessed: Dec. 8, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Verilog-PNG-Encoder-Cardoso/cf1082be8c5501f89ac6111e6ce69e1a4ff08fc9#paper-topics

[14] R. Kachouri and M. Akil, "Hardware design to accelerate PNG encoder for binary mask compression on FPGA," *Proc. SPIE*, vol. 9400, pp. 15–27, Feb. 2015, doi: 10.1117/12.2076483.

[15] T.-H. Tsai, Y.-H. Lee, and Y.-Y. Lee, "Design and analysis of high-throughput lossless image compression engine using VLSI-oriented FELICS algorithm," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 1, pp. 39–52, Jan. 2010, doi: 10.1109/TVLSI.2008.2007230.

[16] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," *IEEE Trans. Image Process.*, vol. 9, no. 8, pp. 1309–1324, Jan. 2000, doi: 10.1109/83.855427.

[17] Y. Zheng, T. Li, W. Li, F. Lei, J. Liu, and Y. Fan, "A low-complexity algorithm for JPEG-LS-Based RAW domain compression," in *Proc. IEEE 15th Int. Conf. ASIC (ASICON)*, Nanjing, China, Oct. 2023, pp. 1–4, doi: 10.1109/asicon58565.2023.10396052.

[18] M. Liu, L. Tang, L. Fan, S. Zhong, H. Luo, and J. Peng, "CARNet: Context-aware residual learning for JPEG-LS compressed remote sensing image restoration," *Remote Sens.*, vol. 14, no. 24, p. 6318, Dec. 2022, doi: 10.3390/rs14246318.

[19] X. Chen, L. Yang, R. P. Dick, L. Shang, and H. Lekatsas, "C-pack: A high-performance microprocessor cache compression algorithm," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 8, pp. 1196–1208, Aug. 2010, doi: 10.1109/TVLSI.2009.2020989.

[20] L. Xie, Z. Zhang, and T. Zhao, "A real-time image row-compression method for high-definition USB cameras based on FPGA," *IEEE Access*, vol. 12, pp. 64663–64671, 2024, doi: 10.1109/ACCESS.2024.3393946.

[21] Z. Ma, F. Zhang, Z. Nan, and Y. Ge, "Intention action anticipation model with guide-feedback loop mechanism," *Knowl.-Based Syst.*, vol. 292, May 2024, Art. no. 111626, doi: 10.1016/j.knosys.2024.111626.

[22] D. Mo, W. K. Wong, X. Liu, and Y. Ge, "Concentrated hashing with neighborhood embedding for image retrieval and classification," *Int. J. Mach. Learn. Cybern.*, vol. 13, no. 6, pp. 1571–1587, Jan. 2022, doi: 10.1007/s13042-021-01466-7.

[23] Q. Zhuang, "USB 3.0 machine vision camera hardware design and FPGA implementation," M.Eng. thesis, School Eng. Sci., Simon Fraser Univ., Burnaby, BC, Canada, 2017.

[24] L. Zhou and F Yu, "Design of high-performance industrial camera system based on FPGA and DDR3," *Proc. SPIE*, vol. 11568, pp. 351–356, Nov. 2020.

[25] Z. Zhang, L. Xie, and T. Zhao, "Design of a low-cost and high-dynamic USB3. 0 domestic ultraviolet industrial camera based on field programmable gate array," *Proc. SPIE*, vol. 12917, pp. 32–42, Nov. 2023.

[26] T. Xiao, T. Xu, and G. Wang, "Real-time detection of track fasteners based on object detection and FPGA," *Microprocessors Microsyst.*, vol. 100, Jul. 2023, Art. no. 104863, doi: 10.1016/j.micpro.2023.104863.

[27] estudoskennedy. (2024). *Project-Classification-Image Dataset*. Accessed: Dec. 8, 2024. [Online]. Available: https://universe.roboflow.com/estudoskennedy/project-classification-images

XINGKAI DU received the bachelor's degree in communication engineering from Zhejiang Sci-Tech University, in 2022, where he is currently pursuing the master's degree in optoelectronic information engineering. He is proficient in Verilog programming. His current research focuses on FPGA-based industrial camera development. His research interests include digital circuits, analog circuits, and signal processing.

LONGHUA XIE received the B.Sc. degree from Fujian University of Technology, in 2021. He is currently pursuing the master's degree in optoelectronic information engineering with Zhejiang Sci-Tech University. He is interested in logic and image processing. His current research interests include camera development and processing.

HONGCHUAN HUANG received the bachelor's degree in electronic information engineering from Zhejiang University of Technology, in 2022. He is currently pursuing the master's degree in optoelectronic information engineering with the University. He is interested in hardware development in the field of electronic information. His current research interests include camera development and debugging.

HUAWEI CHEN is currently pursuing the master's degree in optoelectronic information engineering with Zhejiang Sci-Tech University. He is interested in image processing on embedded devices.

XIAOLONG GUO received the master's degree in test technology and instruments from Zhejiang University, in 2011. He has been with Hangzhou ToupTek Photonics Company Ltd., specializing in the framework design of industrial and scientific cameras. He is proficient in USB3 data transmission protocols and high signal-to-noise ratio circuit design.

TINGYU ZHAO received the Ph.D. degree from the College of Information Science and Electronic Engineering, Zhejiang University, in 2009. She is currently a Professor with the School of Science, Zhejiang Sci-Tech University. Her research interests include the integrated design of optomechanical computing, including CAD design of imaging optical systems, mechanical structure design of optical instruments, and digital image processing algorithms.