Received 28 September 2022, accepted 24 December 2022, date of publication 27 December 2022,
date of current version 16 February 2023.

Digital Object Identifier 10.1109/ACCESS.2022.3232720

RESEARCH ARTICLE

Adaptive Buffer Capture and Storage for High-Speed Cameras

ZHONGHUA HONG[1], (Member, IEEE), SHIHAO SUN[1], XIAOHUA TONG[2], (Senior Member, IEEE), RUYAN ZHOU[1], HAIYAN PAN[1], YUN ZHANG[1], YANLING HAN[1], JING WANG[1], SHUHU YANG[1], AND ZHENLING MA[1]
[1]College of Information Technology, Shanghai Ocean University, Shanghai 201306, China
[2]College of Surveying and Geo-Informatics, Tongji University, Shanghai 200092, China

Corresponding authors: Xiaohua Tong (xhtong@tongji.edu.cn) and Ruyan Zhou (ryzhou@shou.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB0505400, and in part by the National Natural Science Foundation of China under Grant 41871325.

ABSTRACT The output speed of a high-resolution camera with a fast frame rate is high, reaching 800 MB/s or even more. Consequently, the abundant data recorded can easily cause data overflow during transmission and frame loss during storage. To address this problem of data overflow and frame loss while achieving reliable acquisition rates, herein, we design a high-speed video adaptive transmission and storage method based on the camera link transmission protocol. The system uses the camera link transmission protocol and high-speed cameras for data acquisition. During data transmission, memory space is employed as the data buffer pool, an adaptive memory buffer pool is designed using the hardware and acquisition parameters, and a circular read/write buffer, combined with a block device read/write method, is used inside the buffer pool for data transmission. For data storage, input/output completion port asynchronous storage mixed with frame sequence number filtering is employed to write data, and the adaptive buffer of the hard disk is used to achieve efficient data writing. The experimental results reveal that the proposed method can realize efficient and stable writing of data on conventional industrial cameras based on the camera link interface, effectively addressing the problem of frame loss during data storage. Moreover, the stored data can be transferred efficiently with less resource occupation, thereby meeting the application requirements of high-precision vision measurement systems and demonstrating broad application value.

INDEX TERMS High-speed camera, buffer, data storage.

I. INTRODUCTION

In recent years, the demand for cameras with better resolution and high frame rates has increased owing to the increased use of vision measurement systems in industrial inspection fields and the military [1]. Modern high-speed cameras can have capturing frame rates of hundreds or even thousands of frames per second. However, in vision measurement systems, the massive amount of high-speed video data generated by high-resolution and high-frame-rate cameras requires reliable acquisition, stable transmission, and efficient storage techniques. Data acquisition techniques for large amounts of data are developed based on suitable high-speed data transmission protocols; however, a mismatch between the protocols and transmission bandwidths can easily cause limitations in data acquisition. Directly transferring massive amounts of data from the data acquisition card to the storage system can potentially exert enormous transmission pressure on the computer CPU and memory, which creates a bottleneck during data transmission.

Stable storage of massive amounts of data requires the write speed of the hard disk to match the data output rate; if this is not met, the possibility of data frame loss increases, resulting in inefficient use of the large amounts of data recorded.

Generally, the high-speed camera acquisition technology is based on reliable camera interface transmission protocols and camera acquisition cards. Reliable data acquisition

The associate editor coordinating the review of this manuscript and approving it for publication was Dusan Grujić.

requires transmission protocols and transmission bandwidths that match the properties of high-speed cameras. However, transmission protocols with an excessively high rate can potentially exert enormous investment and transmission pressure on high-speed data acquisition cards and computers. Therefore, stable data transmission and efficient storage are essential for vision measurement systems. Usually, the transmission of messages contained in large amounts of data requires stable and efficient algorithms; however, the mechanisms of transfer of large amounts of data from acquisition cards to the computer storage need to overcome the issues of data overflow and frame loss. Despite the development of hard disks and the storage technology, high-speed data storage still suffers from several issues, such as frame loss due to mismatch between the hard disk writing speed and inefficient writing methods.

Typical transmission protocols for high-speed cameras are camera link, CoaXPress, GigE, etc. The camera link protocol was developed at the beginning of the 21st century as a dedicated data transmission interface for industrial cameras, and it was introduced by National Instruments with four data acquisition modes that could acquire and transmit data. Waeny et al. designed and developed a complementary metal–oxide–semiconductor industrial camera with the camera link interface, achieving a data acquisition rate of 200 MB/s [2]. Xu et al. developed a high-speed image acquisition system based on the camera link interface, which uses two static random-access memory (SRAM) units for ping-pong operations to buffer data for data acquisition and transmission with a rate of 150 fps–500 fps via serial data transfer [3]. Nieto et al. developed a data acquisition and transmission device based on field-programmable gate array (FPGA) and graphics processing unit (GPU) devices with the camera link interface to transfer data by buffering between the FPGA, data acquisition system, and GPU [4]. Chen et al. High-Speed Image Velocimetry System for Rainfall Measurement was built using CMOS cameras, capable of capturing images at 500 frames per second [5]. Although data recorded by a camera can be transferred using two SRAM buffers, as well as an FPGA and GPU, the hardware must be designed and developed as a dedicated hardware device. However, the versatility of the hardware is usually poor, and the transmission speed of the system is thereby affected. With the developments in the industrial camera field, the CoaXPress protocol gradually came into public view for coaxial cable transmission. Its transmission speed can typically reach 1.25–6 Gbps, and it covers a long transmission distance. In 2020, Waide used the CoaXPress camera interface protocol to develop a multiple camera vision system [6]. The GigE protocol is a recently developed digital interface protocol that is used along with machine vision, enabling transmission speeds of up to 1 Gbps, long-distance transmission, and simultaneous data acquisition from multiple cameras. Qi et al. used multiple GigE cameras to realize a high-speed vision system with a wide field of view [7]. Zang et al. proposed high-speed TDICCD image data acquisition and storage system, the system uses Field-Programmable Gate Array(FPGA) controlling synchronous dynamic random access memory(SDRAM) array to realize the cache of image data, then according to the received commands, sends certain channel's image data to the acquisition card to write the image data into SCSI hard disk and implements the final storage [8]. Ye et al. designed an FPGA hardware-based GigE camera acquisition system that could write the acquired data into SRAM, thereby lowering resource consumption of the computer [9].

The camera link interface protocol was the first protocol to be developed, and it presented no advantages in terms of the transmission bandwidth, transmission distance, and transmission speed compared with other transmission protocols. However, several old industrial cameras still use the camera link interface protocol. So far, researchers have not been able to suggest efficient solutions for old industrial camera acquisition and transmission storage systems equipped with the camera link interface protocol. To address this gap, in this study, the camera link interface transmission protocol is used for several old industrial cameras to develop a data acquisition module. The protocol uses a computer memory buffer to receive the acquired data to achieve efficient data transmission and storage based on its four acquisition modes.

So far, researchers have proposed two general solutions to address the problem of data overflow and the bottleneck of data transmission during the transmission of massive amounts of data; one involves the buffering of data based on hardware, and the other involves the use of the memory buffer pool space in the computer for dynamic buffering or the development of a ring buffer. Xu et al. designed a high-speed image acquisition system based on the camera link interface protocol, which uses two SRAM units for ping-pong operation during data buffering to acquire and transmit data with a rate of 150 fps–500 fps via serial data transmission [3]. Although data buffering based on hardware SRAM can collect data acquired by high-speed cameras, the hardware must be designed and developed for specific hardware devices. The resulting system is less versatile, and as the buffer pool is a space in the computer memory, using the data buffer pool to buffer the large amount of data collected can reduce the strain on the computer system, thereby providing more stable transmission. Eckart et al. developed the performance adaptive (PA)-user datagram protocol (UDP), which uses the UDP to transmit data and adopts a buffer pool management algorithm to dynamically adjust the buffer pool size, aiming to achieve data buffering [10]. In 2012, X. Wang et al. design DMA size to improve the transmission efficiency [11]. In 2013, the Rada algorithm [10] was proposed based on the PA-UDP protocol for dynamic buffer acquisition during high-speed data transmission; the algorithm uses relevant parameters to dynamically adjust the buffer size and a dynamic memory adjustment scheme to adjust the available memory. The experimental results revealed that the dynamic buffer

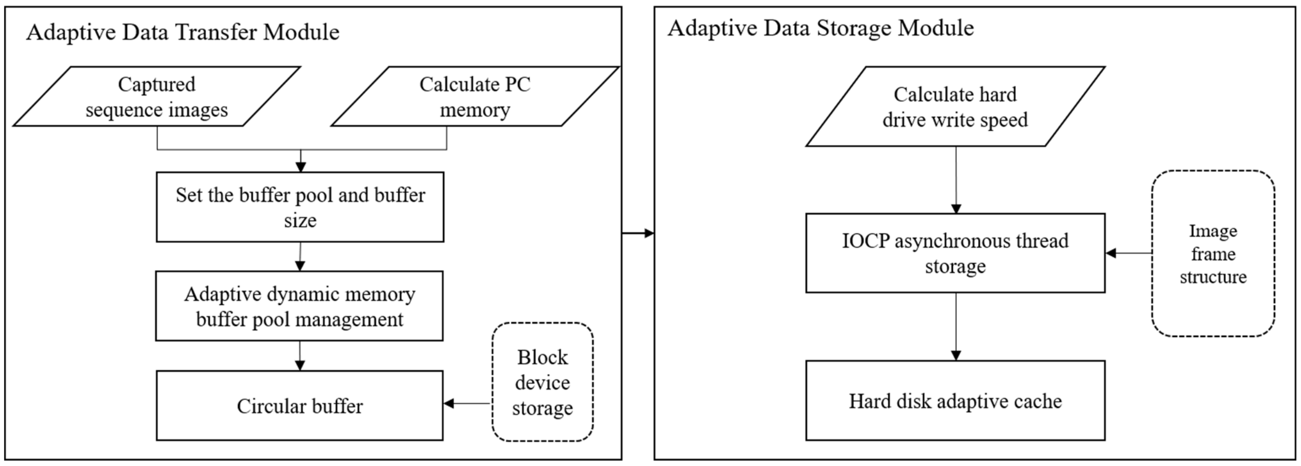

Adaptive Data Transfer Module
Captured sequence images
Calculate PC memory
Set the buffer pool and buffer size
Adaptive dynamic memory buffer pool management
Circular buffer
Block device storage
Adaptive Data Storage Module
Calculate hard drive write speed
IOCP asynchronous thread storage
Image frame structure
Hard disk adaptive cache

FIGURE 1. Framework of the camera link-based adaptive transfer storage system.

scheme outperformed the static buffer scheme in terms of the throughput and memory occupation. However, this dynamic buffer scheme uses multiple parameters for collective adjustment, and this implementation requires various data parameters and abundant memory and storage. Additionally, the implementation of the method is slightly complicated, and the method uses highly concurrent data from data centres, such as banks. Moreover, the amount of data generated is far less than that collected by high-speed cameras. Zhuo et al. adopts the computer distributed framework and the efficient network transmission technique to finish overall framework design of the network data visualization system, and realizes it based on the Web full-stack technique [12]. In 2017, Paul et al. proposed a circular buffering algorithm for high-speed data transfer through a shared memory area [13]. Inglés et al. used a standard C++ library to develop a multi-threaded circular buffer system [15]. Kasu et al. propose The layout-aware data scheduling (LADS), During data transfer, LADS can avoid congested storage elements by exploiting the underlying storage layout at each endpoint. This improves the I/O bandwidth and hence the data transfer rate across high-speed networks [16]. In 2022, H. Kim et al. propose a concurrent and reliable end-to-end data integrity verification scheme considering the internal operation of the storage devices for data transfer between high-performance computing systems with flash-based storage devices. To perform data integrity verification including data corruptions that occurred inside the storage devices, we control the order of I/O operations considering the internal operations of the storage devices [17]. Generally, shared memory, multi-threading, and ring buffers can increase the efficiency of data transfer; however, such systems do not specify the size of the buffers, which is prone to redundancy.

So far, two solutions have been proposed to address the problems of frame loss during the transfer of massive amounts of data and the mismatch between the storage efficiency and writing speed. These solutions increase the storage efficiency using a byte stream writing algorithm for data writing or the asynchronous storage strategy of input/output completion port (IOCP) and the buffer management algorithm. In 2016, researchers used a byte stream writing algorithm for data storage [18], aiming at stable and reliable data writing. However, the large amount of data generated during industrial experiments is prone to data loss and transmission instability problems when the byte stream writing algorithm of the transmission control protocol is used, which is why the transmission efficiency is significantly reduced. With the continuous development in the performance of high-speed storage media, the asynchronous storage strategy of IOCP has been adopted to significantly accelerate the efficiency of high-speed data transmission and storage. In 2014, a bicycle rental platform was developed using the IOCP mechanism to handle the high concurrency of data with reasonable allocation of resources [19]. The use of IOCP for asynchronous storage can produce a sufficient response to a large amount of data written via concurrent execution, which reduces the data processing time. Nonetheless, only limited multi-threaded storage techniques based on thread pooling have been proposed, and the massive data frame loss can also pose a major problem. Wei et al. propose a locality-aware cooperative buffer scheme referred to as FlashCoop (Flash Cooperation), which can efficiently shape the I/O request stream and improve the sequentiality of the write accesses passed to the SSD [20]. Liu et al. proposed a cache management algorithm to improve the disk cache utilization of solid-state drives (SSDs) by using random access limited. Accordingly, the data written to the stock is randomized and flushed to the hard disk to advance the efficiency of cache utilization [14]. However, this random-write algorithm causes computational stress per frame on top of frame reads. In terms of addressing the data loss, researchers have adopted hardware and software solutions to deal with the same. On the hardware side, the number of frames recorded and the number of frames stored are synchronized using a synchronized digital tube; however, its implementation is challenging. The software uses frame marker points to mark the frames in the collected data before identifying and verifying the frame markers during storage to identify frame loss. During the storage of massive amounts of data, the mismatch between the writing speed of the hard disk and the data transmission speed can lead to frame loss.

In summary, this study aims to accomplish the following objectives based on research proposals and ideas put forth in recent years. First, for the limitation on data acquisition, a vision acquisition system based on the camera link protocol is designed to efficiently add different acquisition modes of the camera link protocol under low bandwidths. An adaptive memory buffer is designed to receive the data transferred from the camera acquisition card to address the problem of data overflow and the data transmission bottleneck. Moreover, by using an adaptive dynamic memory buffer pool and a circular buffer within the buffer pool, the size of which is consistent with the size of the captured image, data redundancy is significantly reduced, and the efficiency of transmission is considerably improved. Third, the asynchronous storage strategy of IOCP is used to store data using multiple threads to address the problems of frame loss and low storage efficiency during data storage. Additionally, a new frame structure is adopted to identify the frame structure during storage to solve the problem of frame loss and to increase the storage efficiency.

II. HIGH-SPEED VIDEO ADAPTIVE TRANSMISSION AND STORAGE METHOD

In this study, the considered high-speed video adaptive acquisition and storage method based on the camera link protocol is divided into two modules: the adaptive memory buffer module and the adaptive data storage module. In the memory buffer module, a dynamic memory buffer pool is used for data transmission, and a circular buffer is adopted to achieve circular buffering of acquired data frames. The size of the buffer pool and that of the buffer are determined experimentally. In the adaptive storage module, the adaptive model of the hard disk buffer is developed based on experiments and data fitting by adopting the asynchronous multi-threaded approach of IOCP to increase the storage efficiency.

The remainder of this paper is organized as follows: Section I introduces the adaptive data transfer module, including the determination of the buffer pool size, buffer size, adaptive memory buffer pool, and circular buffering of the buffer. Section II introduces the adaptive data storage module, including the asynchronous multi-threaded storage of IOCP and the adaptive storage method of the hard disk. Finally, Section III presents the experimental results.

A. ADAPTIVE DATA TRANSFER MOUDLE

During the transfer of large amounts of data, data transfer from the camera acquisition card to the computer can easily cause data overflow because the software development kit (SDK) of a high-speed camera acquisition card does not involve the buffer pool design. Furthermore, if data are written directly through the hard disk, the writing speed of the hard disk does not match the data transfer rate, which significantly affects the stability of data storage and may even lead to the loss of a large amount of data. Therefore, reasonable use of the memory buffer space has been considered in existing studies, and consequently, it has been established that data can be stored efficiently by designing a suitable buffer pool.

Figure 2 presents the primary methods of adaptive data transmission. A high-speed camera records large amounts of high-speed video data and transmits the data to the image acquisition card through the optical fiber medium. Subsequently, the image acquisition card sends the image data to the computer. Herein, we use a dynamic buffer pool to buffer the data transmitted by the camera acquisition card. First, a circular buffer is designed in the buffer pool space based on the actual physical memory; the acquisition frame rate, image size, and other related parameters are used to determine the buffer pool size for pre-acquisition. The buffer pool size is then selected. Although the buffer pool size is adjusted with the acquisition rate, an oversized buffer pool still affects the performance of inter-process communication (IPC). Further, the maximum limit size of the buffer pool and size of each buffer are designed based on an experiment. Subsequently, the circular buffer pool is introduced to accommodate the massive amount of data transferred. Each buffer in the buffer pool can be written repeatedly, thereby addressing the problem of data overflow in the buffer pool.

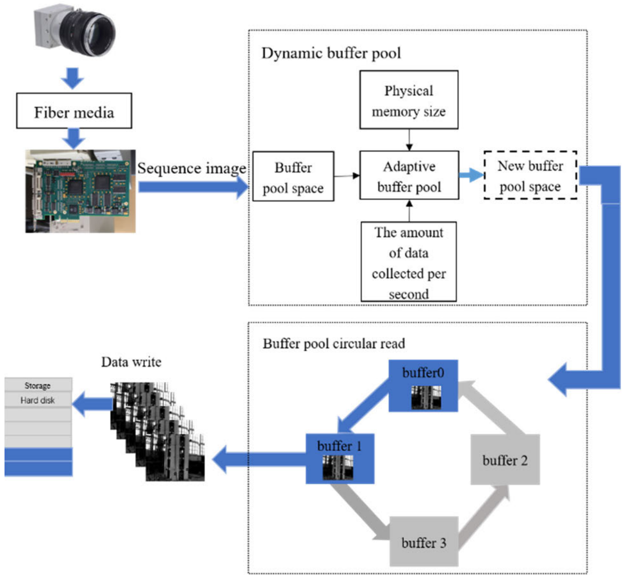


FIGURE 2. Adaptive transmission algorithm framework.

B. BUFFER POOL SIZE

The buffer pool is a part of the computer's memory (RAM) space, which is utilized for data exchange with the CPU in the Windows operating system. The system memory size is governed by the size of the physical memory, which is determined by the size of a specific memory card. However, the memory size in an operating system is not exactly equal to that of the physical memory, as part of the computer's hardware also occupies a small portion of the memory. When a program is running, the operating system allocates memory space for each process of the program and limits the memory size for each program. Consequently, excessive memory usage slows down data exchange between the CPU and memory in the operating system, which affects the performance of the computer [21]. In this study, a dynamic buffer pool is designed, whose size adapts to the amount of data collected each time. However, the maximum buffer pool size of the same IPC is determined by the memory of the IPC. The designed dynamic buffer pool is too large, which is why it requires extensive computer resources, and such high memory usage can cause system run down.

Therefore, in this study, we design experiments to determine the amount of space that can be allocated to the buffer pool for high-speed data transfer.

Note that the size of a buffer determines the size of the entire buffer pool, as indicated in the following equation.

$$s \times p = b \tag{1}$$

where s indicates the number of buffers, p indicates the buffer size, and b indicates the size of the entire buffer pool. Usually, high-speed data are transferred to the buffer pool

In a frame-by-frame manner, and during writing, the data are written in the order of each frame. When the data are not transferred and stored in frame, it is essential to determine the starting pixel position of a particular frame when the image is displayed in real time, which will result in additional computational burden. In this study, we design a storage method to reduce the computational burden by aligning the size of the area of interest of high-speed camera exposure with the buffer size. The buffer size is related to the size of the block device during the one-time write operation; the following graph presents the write speeds of two different hard drives with sizes of 0–64 MB. The graph reveals that the write speeds of the block device with sizes of 0–64 MB are approximately the same.

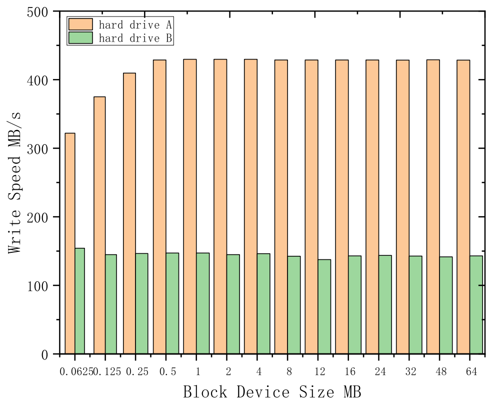


FIGURE 3. Variation of write speed with block device size (0–64 MB).

In Figure 3, although block device at a time to write the size of 64 MB will increase part of the efficiency. However, if the buffer data accumulates to 64 MB before being written to the block device, it is a cannot tolerate time, and the writing speed will be abysmal. Additionally, read the buffer after the buffer size as a write block size. Although it cannot make the block device write performance play the best, the block device writes speed can also meet the requirements of high-speed video measurement writing. The area of interest for high-speed camera exposure is adjusted during each acquisition.

In summary, the buffer size is designed to be the size of the area of interest in the high-speed camera exposure, which prevents additional data segmentation, improves efficiency, and facilitates block device writing. The block device write method will be further expanded in detail in the IOCP asynchronous write section.

C. ADAPTIVE BUFFER POOLING

A massive data throughput strains the transmission storage of a system. A fixed-size buffer pool becomes quickly occupied, and the large amounts of data cause such a buffer pool to overflow. The Rada buffer pool management algorithm mentioned earlier is a dynamic management algorithm that uses data entering and exiting a buffer pool to dynamically manage the buffer pool. However, considering the massive amounts of high-speed video data transmitted in the buffer pool in one frame, it is difficult to handle one data frame using the Rada buffer pool management algorithm. Therefore, a dynamic buffer pool management algorithm is proposed in this section to maximize the use of the memory buffer pool and enhance the transfer efficiency of massive amounts of data. The amount of data transmitted in high-speed video data transmission can be calculated using the following equation.

$$s = t \times f \times p \tag{2}$$

where t indicates the acquisition time, f symbolizes the acquisition frame rate, and p denotes the image's window size, i.e., the image's length and width, (m $\times$ n).

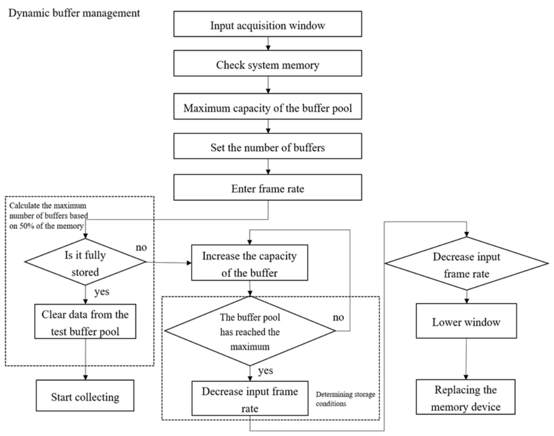

Dynamic buffer management
Input acquisition window
Check system memory
Maximum capacity of the buffer pool
Set the number of buffers
Enter frame rate
Calculate the maximum number of buffers based on 50% of the memory
Is it fully stored
no
yes
Clear data from the test buffer pool
Start collecting
Increase the capacity of the buffer
The buffer pool has reached the maximum
no
yes
Decrease input frame rate
Determining storage conditions
Decrease input frame rate
Lower window
Replacing the memory device

FIGURE 4. Adaptive buffer pool management algorithm framework.

Usually, a massive amount of data is transferred from the camera acquisition card to the memory buffer pool.

Consequently, the performance of the buffer pool is usually constrained by the amount of transferred data and physical memory conditions. Therefore, herein, a buffer pool with an adaptive size is designed using different parameters, such as the frame rate and window size of the transferred image data. The specific algorithm framework is detailed in Figure 4.

The specific process flow of this algorithm is as follows: the memory size of the system and the transfer speed of the storage device should be tested before initiating the high-speed data acquisition and transfer. First, the buffer pool size formula is used to determine the maximum memory space that can be used as buffer pool space. Following this, the acquisition and transfer test phases are executed, and the number of buffers, the acquisition frame rate, and the open window size of the simulated test section are input.

Subsequently, the algorithm checks the amount of captured data. In the first stage, it tests whether the captured frame rate satisfies the storage needs, and in the second stage, it tests whether buffer overflow occurs; in the third stage, it tests the integrity of the stored file and whether it can store all the data. Lastly, the algorithm modifies the buffer pool capacity based on the results of the three stages of testing and gradually increases the capacity of the buffer pool without exceeding its maximum capacity.

In case the buffer pool redundancy problem still persists after increasing to the maximum buffer pool capacity, the frame rate is reduced to satisfy the minimum frame rate condition for high-speed data acquisition, and data acquisition is performed. After reducing the frame frequency, in case effective data transfer is still not feasible, the memory device is chosen to be replaced for transfer.

Algorithm 1 demonstrates the pseudocode for adaptive buffer pooling, aimed at enhancing the efficiency of transmission. The input parameters are the size of the captured image and the frame rate of the capture process. First, the maximum buffer pool space is determined using memory detection, and then, the algorithm checks for buffer overflow. If overflow is detected, the buffer pool space is increased, and once the buffer pool space reaches its maximum, the frame rate of the capture process is reduced. Else, the data in the buffer pool are cleared, and acquisition is formally initiated using the same parameters.

Algorithm 1 Adaptive Buffer Pool Management Algorithm

```
→ buffer: buffer pool capacity
→ picture: Capture image window size
→ fps: Capture frame rate
→ maxbuffer: Maximum buffer pool capacity
1. Input: picture a*b, fps x
2. Output: return write finish
3. int a*b, int x
4. int maxbuffer=maximum memory capacity*50%
5. while transfer do
6.         if return finish
7.               clear data then start data collection;
8.           else
9.                  buffer→buffer1;
10.        end if
11.            if    buffer1=maxbuffer then
12.                 fps→fps1;
13.             when fps1=slow FPS
14.                  picture a*b→a1*b1 or
15.                  return
16.             end when
17.             end if
18. end while
```

D. BUFFER CYCLE READ/WRITE

While tlizing the memory buffer pool to receive the massive amount of data transferred, the speed of data transfer usually does not match the speed of data writing, which easily causes buffer pool data overflow and reduces the efficiency of high-speed data acquisition and storage and even risks data loss. In high-speed camera acquisition storage, the previously mentioned multi-threaded circular buffer system [22] employs a circular buffer through an adaptive memory buffer pool design. However, the overall buffer size is still constrained by the size of the memory. Therefore, based on the adaptive buffer pool structure under the design of the buffer circular buffer, which was previously mentioned through the buffer and block device for each frame of the data storage. According to the SDK of the acquisition card in the circular buffer method, each buffer within the buffer pool can be repeatedly written, thus carrying out repeated read and write operations, which significantly improves the efficiency of data transmission and storage, solving the problem of the limited capacity of the memory buffer pool encountered by the Rada method. The circular read/write model is illustrated in Figure 5 below.

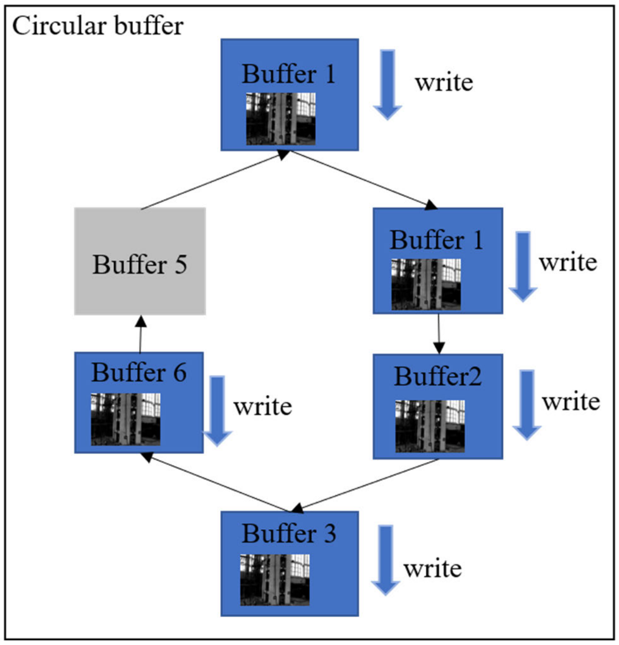


FIGURE 5. Circular buffering of buffer blocks in the buffer pool.

As shown in Figure 5, six repeatable read/write buffers exist inside the buffer pool with corresponding buffer numbers for writing and reading operations in a specific buffer. Among them, 0–4 have all been written to; if data transmission is still in progress, buffer 5 will be occupied. Subsequently, the data will be stored according to the buffer number, ensuring both the order and integrity of the block device during storage. More importantly, the data will not be lost in frames. After all the buffers are occupied, new data will still be written. The data will continue to be written to buffer 0 and so on in a cycle of reading operations, and the writing operation must be executed in real-time to the corresponding storage media. Otherwise, the data may be lost.

E. ADAPTIVE DATA STORAGE MODULE

A massive amount of data needs a stable and efficient storage mechanism for further writing. Although a massive amount of data is stored by a byte stream single thread or multi-thread in the standard byte stream writing algorithm, the writing speed of the storage device presents a challenge. Asynchronous writing algorithm multi-thread writing is highly suitable for a block device. Block devices have distinctive logical control and multi-channel data storage areas that can perform data writing tasks in multiple parallels. Herein, the asynchronous write algorithm [23] of IOCP is used to write massive amounts of data to maximize the writing efficiency of the storage device by combining the block device with multi-threading. Subsequently, the adaptive hard disk cache is introduced for writing operations with and without cache to improve the writing efficiency. Finally, frame sequence number filtering is used to prevent frame loss during storage.

Algorithm 2 IOCP Asynchronous Writing Algorithm

```
→ IO port: Asynchronous IO port
→ thread 0: Storage thread 0
→ thread 1: Storage thread 1
→ cache: Storage Devices
→ time: Preset time
1. Input: start write
2. Output: return write finish
3. while IO transfer do
4.         create IO port then
5.             IO port=cache;
6.         when start write then
7.             start thread0;
8.         if IO complete
9.             write thread0→cache;
10.              else
11.                  restart thread0;
12.              end if
13.          when thread0 complete
14.              start thread1 then
15.              write thread1→cache;
16.            if time = time(thread1→cache);
17.              return write finish
18.              end if
19. end while
```

F. IOCP ASYNCHRONOUS WRITING

First, we create the I/O completion port and bind the I/O completion port to the opened device. Then, we set up the stored block device object and pre-issue the write command with four bytes of 0, after which we initiate "start storage" and open thread 0.

Following this, we check if the storage is started, and if it is not started then it is simply ended. Else, we check the I/O completion queue. If there is no write command completion, we continue checking the I/O completion queue. Else, we read the buffer pool data block. This data block size corresponds to a single frame image size. Following this, we issue a write device command to update the value of the written offset. Determine how much data has been written. The stop storage thread needs to be triggered to terminate the real-time asynchronous I/O process. If the stop storage thread has been triggered, we open thread 1 and check if the preset time is consistent with the storage time. Here, the preset time implies the storage time that a user sets in advance. If not, the thread must wait until such consistency is ensured with the present time and then stop the storage task to loop on the massive amount of data.

G. FRAME STRUCTURE DESIGN

Further, a frame structure with the sequence number is designed to detect the sequence number during storing, and the frame structure is depicted in Figure 6. Each frame of the data image generated by the high-speed camera exposure contains a frame number, which is a binary number in the header area of each frame, and the decimal difference between the frame sequence number of each frame and the previous frame is 1. Therefore, this buffer can address frame loss.

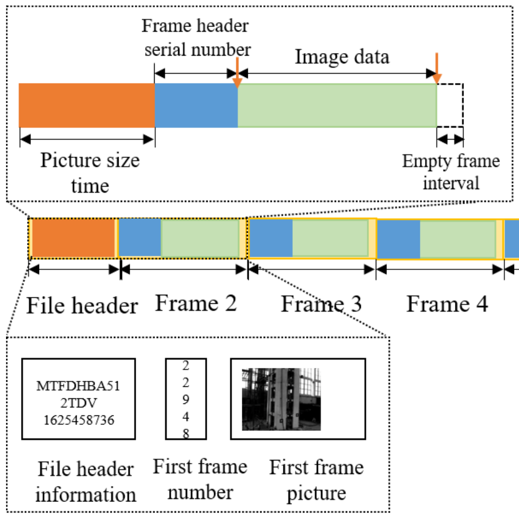


FIGURE 6. Frame structure of the acquired image.

The 8192-byte header at the head of the frame structure stores information pertaining to the time of acquisition and the size of sequence images. This is followed by the header of the first frame file, which contains the serial number of the first image and image size information, and this is followed by the first image frame, with empty data reserved at the end of the frame structure to distance it from the next image frame. For each retrieved frame, the module provides information on its resolution, number, timestamp, and so on.

H. HARD DISK CACHE ADAPTIVE STORAGE

Algorithms often encounter problems with cache applications. Without a highly efficient algorithm, the process will result in a low utilization rate of cached data in the application and will not be capable of effectively exploiting high-capacity cache. For example, when storing large amounts of data at high speeds and writing large files to the hard disk, the writing speed needs to be limited by the rotational speed of the internal drive of the hard disk. Therefore, the hard disk buffer does not necessarily improve the speed of data storage. Alternatively, when the size of the stored files is smaller than the buffer capacity, the hard disk buffer can first store the written data, after which it can store the data on the hard disk platter to enhance the storage efficiency [24].

As illustrated in Figure 7, the data writing adaptive scheme uses the size of the captured image to compare with the experimentally derived hard disk read/write speed corresponding to the open window size. When the size of the captured image is larger than the corresponding open window size, hard disk cache is bypassed to directly write to the hard disk storage unit; alternatively, when the size of the captured image is smaller than the corresponding open window size, the hard disk cache buffer is used to achieve high-speed image data. When the size of the captured image is smaller than the corresponding window size, the cache of the hard disk is used to achieve high-speed and highly efficient data writing [25].

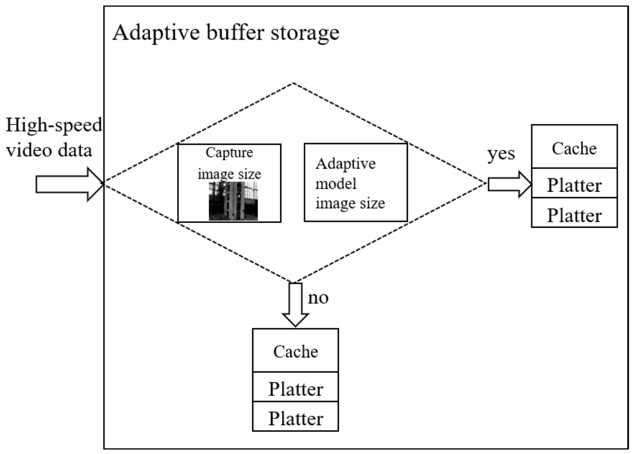


FIGURE 7. Adaptive buffer storage algorithm.

Further, based on the storage algorithm model of the captured image window size and the write speed of the hard disk, the storage and writing method with or without cache is performed according to the window size of the image. The adaptive relationship between the transfer speed of the hard disk and the image window size is expressed in the following equation.

$$d = a \times e^{(x \times b)} \tag{3}$$

where x denotes the transfer speed of the storage hard disk, d indicates the acquired window size (the length and width of the image), and a and b denote constants.

The relationship between the average write speed of the hard disk and the image window size can be derived according to the adaptive model, as presented in the following table.

TABLE 1. Average write speed of hard disk in relation to the window size.

Hard Drive	Write speed MB/s	Image size/Bite	Window opening size (bite*bite)
Samsung 980pro 512GB	438.66	1290000-1325000	1280*1024
WD10-EZEX-08M2NA0 1TB	136.25	2550000-2600000	1920*1300
WD10-EZZX-08M2NA0 2TB	180.56	2300000-2325000	1920*1200
STDR1000300 2TB	244.32	1950000-2075000	1920*1024

According to the storage adaptive model, the writing speed of the hard disk is computed in the system corresponding to the open window size, and when the acquired image size is larger than the image size corresponding to this write speed, it is stored in a cache-less mode, whereas when the acquired image size is smaller than the image size corresponding to the write speed, it is written in a cache mode.

III. EXPERIMENTAL PART

For our experiments, a Dell business desktop PC with a SATA interface was equipped with a Samsung 970 high-speed SSD with a storage capacity of 1 TB and a maximum writing speed of 2.3 GB/s. Another Dell business desktop PC with a PCIe 4.0 × 16 interface was equipped with a camera link high-speed image acquisition card. A CL600 camera was used as the data source for image acquisition. Moreover, two fiber optic media and a data interface camera link connected the CL600 high-speed camera to the high-speed image acquisition card.

A. BUFFER POLL SIZE

According to the formula presented in Section II-B, $s \times p = b$, the buffer pool size is related to the number of buffers and buffer size. In our experiment, we designed buffers with a fixed size and set the size of each buffer to 4000 KB. The experimental results are illustrated in Figure 8 below.

The first experiment involved the application of memory buffers, starting from 1000 buffers and then gradually increasing the number with a step size of 1000. It can be observed from the graph presented on the left that the memory occupation of the computer appears to be rising with the increase in the number of buffers applied, and the memory resources of the computer seem to be heavily occupied. Subsequently, when the number of buffers gradually increases to 6000, the memory occupation of the computer exceeds 75%, and the resource occupation is excessive, which can cause the system to crash. The error corresponding to 1000 buffers appears extremely high, and it is difficult to accurately estimate the maximum buffer pool occupation. Therefore, in the second experiment, the memory buffer step was set to 100, and the number of buffers gradually increased from 5000. It can be observed from the graph presented on the right that when the number of buffers gradually increases to 5900, the resource consumption system of the computer finds it difficult to satisfy the demand of normal acquisition and transmission storage. The buffer pool occupation formula can be presented as

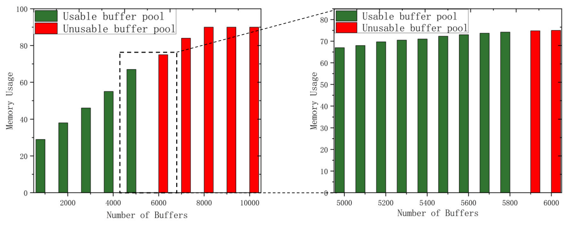


FIGURE 8. Relationship between the number of buffers and computer memory usage.

$$5800 \times 4000 = 23,200,000KB \approx 44GB \times 0.5 \tag{4}$$

According to the above-mentioned experimental method, a maximum buffer pool capacity of 16G and 64G IPCs was also verified, and the experimental results are tabulated in Table 2 below.

As mentioned above, the size of the memory buffer pool is constrained by the physical memory. Admittedly, an ample buffer pool space can store data more efficiently; however, a larger buffer space implies that it engages additional computer resources, and all the system resources are occupied by the buffer pool, which will not only cause the computer to crash but also lead to the loss of transferred data owing to the computer crash. Based on the above experimental data, we can conclude that if the buffer pool space is within 50% of the system memory, it can ensure safe and stable data transfer and prevent delay of data exchange with the CPU owing to excessive memory resource occupation [27].

TABLE 2. Relationship between memory and buffer pool size.

IPC memory capacity	Number of buffers	Buffer pool capacity
16G	2000	7.6G
44G	5800	22.1G
64G	7900	30.1G

B. ADAPTIVE MEMORY BUFFER POLLING

Rada, another buffer management algorithm mentioned in a previous paper, is an efficient dynamic buffer pool adaptive algorithm for high-speed data transfer; it primarily ensures that the size of the memory buffer pool can be adjusted according to real-time execution to realize dynamic data buffer pool buffering, after which it dynamically adjusts the buffer pool size for efficient storage according to the CPU, as well as the efficiency of the memory.

In our next experiment, buffer pooling and circular buffer algorithms were utilized for a comparison with the previously mentioned buffer management algorithms. In the first part of the experiment, memory buffer pool utilization was compared between fixed and dynamic buffer pool sizes using different amounts of collected data. Accordingly, two fixed memory buffer pool capacities of 25% and 50% were set. Subsequently, the Rada buffer pool management algorithm was used for data collection to compute the buffer pool utilization of the dynamic buffer pool. In this study, the dynamic buffer pool algorithm was used for data acquisition and storage of the same size, and it determined the buffer pool utilization independently. The experimental results are presented in Figure 9.

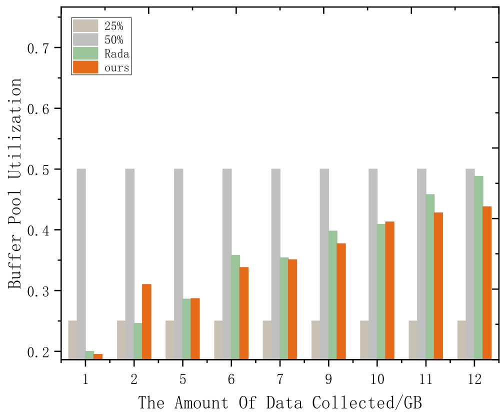


FIGURE 9. Buffer pool utilization.

The multi-threaded ring buffer-based algorithm mentioned in the previous section employs the frames captured by the high-speed camera for ring buffering. In our experiment, the multi-threaded ring buffer algorithm, along with the Rada buffer pool management algorithm, was employed to calculate the storage time for different amounts of acquired data while comparing with the adaptive buffer pool algorithm. Additionally, two fixed buffer pools with sizes of 25% and 50% were set. The experimental results are illustrated in the following Figure 10.

The figure above illustrates that the memory utilization of the two dynamic buffer pools continues to change with an increase in the amount of collected data. Under the condition that the buffer pool space does not exceed 50% of the memory size, the dynamic buffer pool has less resource occupation than the fixed-size buffer pool. Consequently, it demonstrates more efficient transmission performance for the same amount of transferred data. Figure 10.depicts the storage time of the fixed buffer pool compared with that of the dynamic buffer pool and ring buffer. The storage time of the dynamic buffer pool is shorter than that of the fixed buffer pool for the same amount of collected data. Moreover, the same dynamic buffer with a ring buffer has a higher storage time than the algorithm considered in this study. Evidently, it can be inferred from Figure 9. and Figure 10. that the dynamic buffer pool can exhibit more efficient transmission performance in the limited memory space. Among the two dynamic buffer pooling methods, the dynamic management algorithm considered in this study can operate with a low storage time and minor memory occupation.

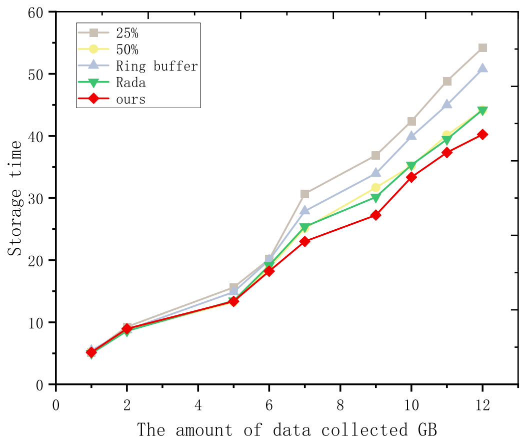


FIGURE 10. Storage time for different buffer pool sizes.

C. HARD DISK BUFFER ADAPTIVE METHOD

In the hard disk transfer speed method, adaptive storage model when after the hard disk speed and storage time data fit to get, using different sizes of data test packets, data packets according to 0.5 KB, 1.0 KB, and 2.0 KB until 8192.0 KB for reading and writing tests. Consequently, the average writing speed of the hard disk can be obtained. From the previous conclusion, it is evident that the mechanism of storage and the size of the open window have a significant influence on the actual write time of the hard disk; therefore, the storage time of different hard disks with different image sizes under the conditions of the presence of a cache area and the absence of a cache area is designed in the adaptive storage model, and the experimental results are shown in the following Figure 11.

The above figure presents the amount of storage according to the adaptive storage model in the presence of a buffer and no buffer. The independent variable indicates the open window size (size of the image in byte). The experiments verify the storage time at different transfer speeds of the hard disk with different window sizes. Based on the above data, a simple function model can be established. Subsequently, the function equation obtained by using the Newton method and the general global optimization fitting method can optimize the mean square error, root mean square error, and the coefficient of determination of the relevant results. Consequently, the relevant results are chosen for the final function equation model.

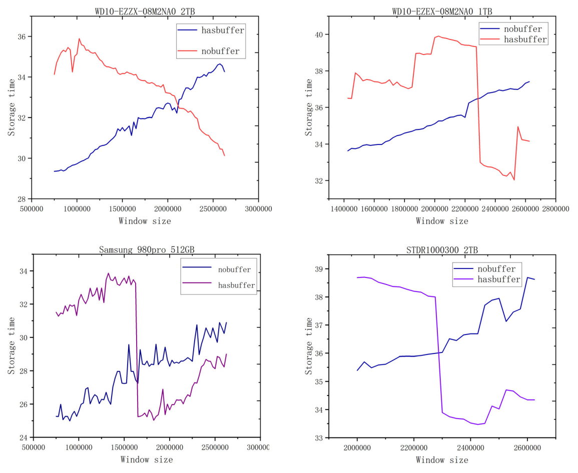


FIGURE 11. Storage time with and without cache.

In the experimental verification stage, the time required by high-speed video capturing and storage is compared between scenarios with and without the use of the adaptive storage model. The high-speed video capture rate and storage of the same hard disk, Samsung 870 1 TB, in Base, Medium, and Full modes are experimentally verified according to several different capture modes of the camera link protocol. The experimental results are presented in Figure 12 below.

Based on the experimental results, under the condition of the same storage hard disk and the same acquisition time, we can observe that as the size of the opening window gradually increases, writing of the massive amount of high-speed video data to the hard disk without the storage adaptive model will take longer than that with the storage adaptive algorithm. Furthermore, comparison tests reveal that the adaptive storage model can effectively cope with different image opening window sizes and achieve more efficient storage of high-speed video data.

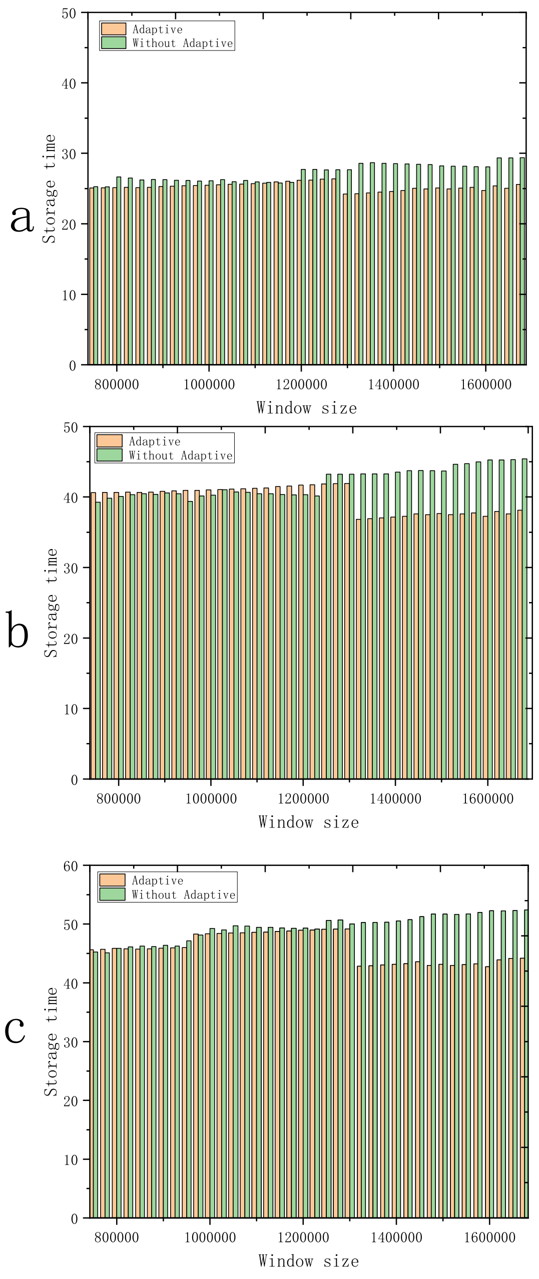


FIGURE 12. (a) Time in Base mode with or without the adaptive model. (b) Time in Medium mode with or without the adaptive model. (c) Time in Full mode with or without the adaptive model.

D. TRANSFER SPEED AND FRAME LOSS EXPERIMENTS

The buffering and storage of massive amounts of data are algorithmic problems. In a further experiment, the multi-threaded ring buffer algorithm and the Rada buffer pool management algorithm were employed to calculate the average write speed for different amounts of collected data and to compare with the adaptive buffer pool algorithm designed in this study. The experimental results are illustrated in Figure 13.

According to the graph above, the average writing speed for large amounts of data increases slowly as the volume of data increases. However, the speed will not be greater than the maximum write speed of the drive, and the writing speeds of SSDs will be greater for large amounts of data than those for small amounts of data. The adaptive buffer pool approach adopted in this study demonstrates higher writing speeds and storage efficiency than the ring buffer algorithm and the Rada dynamic buffer pool algorithm.

Note that in this experiment, the stability of data acquisition and storage is based on the frame structure designed in a previous paper, as tabulated in Table 3; here, the starting frame number and the ending frame number of the stored data are recorded, and the number of stored image frames is calculated. Moreover, the table below indicates that the method adopted in this study can effectively address the problem of frame loss and achieve stable storage for a long time compared with other methods.

TABLE 3. Frame loss of different storage methods.

Start frame number			End frame number			Image frame rate		
Ours	Rada	Ring Buffer	Ours	Rada	Ring Buffer	Ours	Rada	Ring Buffer
2	2	2	6210	6210	6210	6209	6197	6182
2	2	2	10561	10561	10561	10560	10549	10478
2	2	2	27832	27832	27832	27831	27803	27716
2	2	2	52082	52068	45000	52081	51966	44628
2	2	2	102843	98245	45000	102842	98135	44635
2	2	2	205921	150923	45000	205920	150511	44655

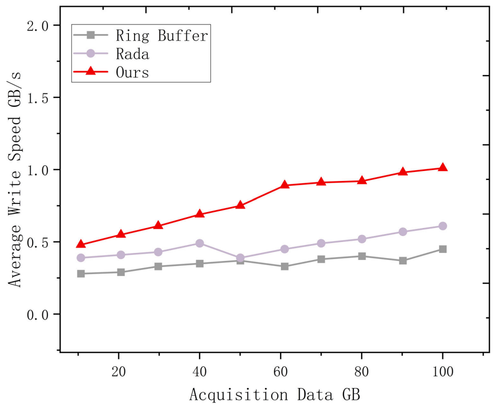


FIGURE 13. Average write speed of several methods.

IV. CONCLUSION

This paper proposes an adaptive method for transferring and storing high-speed video data to achieve stable acquisition with safe and efficient storage. Accordingly, herein, we use dynamic memory buffer pooling and circular buffering within the buffer pool to accept and process the transmitted data and achieve adaptive memory buffer pooling limiting the maximum buffer pool size. Consequently, the transfer efficiency is higher. In terms of data storage, we use IOCP-based asynchronous writing and frame number filtering to asynchronously write the transmitted data with multiple threads, which increases the utilization of storage resources and addresses the data loss caused by writing through byte streams. Combined with the frame structure in the sequence number for frame loss identification, the approach ensures the integrity of the storage process and addresses the data frame loss problem. When writing to the hard disk at the writing speed of the hard disk, we establish an adaptive buffer model according to the size of the image to write with or without a buffer, thereby increasing the storage efficiency.

REFERENCES

[1] M. Vollmer and K.-P. Möllmann, "High speed and slow motion: The technology of modern high speed cameras," *Phys. Educ.*, vol. 46, no. 2, pp. 191–192, Mar. 2011.

[2] M. Waeny and P. Schwider, "CMOS megapixel digital camera with CameraLink interface," *Proc. SPIE*, vol. 4669, pp. 137–144, Dec. 2002.

[3] Z. Xu, Y. Chen, and X. Zhang, "Design of serial image acquisition system based on camera link," in *Proc. 7th IEEE Conf. Ind. Electron. Appl. (ICIEA)*, Jul. 2012, pp. 1804–1809.

[4] D. Waide, "Applying the CoaXPress interface in multi-camera machine vision systems," *Quality*, vol. 2020, pp. 4–6, Jan. 2020.

[5] C. Chen, C. Hsieh, P. Chi, C. Lin, C. Weng, and C. Hwang, "High-speed image velocimetry system for rainfall measurement," *IEEE Access*, vol. 6, pp. 20929–20936, 2018.

[6] J. Nieto, P. Guillén, S. Esquembri, G. de Arcas, M. Ruiz, J. Vega, and R. Castro, "High performance image acquisition and processing architecture for fast plant system controllers based on FPGA and GPU," *Fusion Eng. Des.*, vol. 112, pp. 957–960, Apr. 2016.

[7] Q. Li, R. Lu, N. Liu, and Q. Yu, "Architecture and processing speed study of multi-GigE camera high-speed vision system," *J. Electron. Meas. Instrum.*, vol. 24, no. 4, pp. 4–8, May 2010.

[8] J. Zang, Y. Li, X. Xue, Y. Guo, and J. Zang, "Multi-channel high-speed TDICCD image data acquisition and storage system," in *Proc. Int. Conf. E-Product E-Service E-Entertainment*, Nov. 2010, pp. 1–4.

[9] L. Ye, K. Yao, J. Hang, P. Tu, and Y. Cui, "A hardware solution for real-time image acquisition systems based on GigE camera," *J. Real-Time Image Process.*, vol. 12, no. 4, pp. 827–834, Dec. 2016.

[10] B. Eckart, X. He, Q. Wu, and C. Xie, "A dynamic performance-based flow control method for high-speed data transfer," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 1, pp. 114–125, Jan. 2010.

[11] X. Wang and D. Tian, "Design and research of the data acquisition system based on the mass DMA transmission," in *Proc. Int. Conf. Comput. Sci. Service Syst.*, Aug. 2012, pp. 651–654.

[12] Z. Zhuo, J. Dong, Y. He, L. Guo, and B. Peng, "Design and implementation of visualization system based on network mass data," in *Proc. 2nd IEEE Int. Conf. Comput. Commun. (ICCC)*, Oct. 2016, pp. 150–154.

[13] S. Paul, "Lockless circular buffer over shared memory (high speed data transfer via shared memory)," in *Proc. Int. Conf. Adv. Comput. Commun. Syst.*, Dec. 2013, pp. 1–6.

[14] H. Liu, Y. Zhang, Y. Zhou, X. Fu, and L. T. Yang, "Receiving buffer adaptation for high-speed data transfer," *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2278–2291, Nov. 2013.

[15] R. Inglés, P. Perek, M. Orlikowski, and A. Napieralski, "A simple multithreaded C++ framework for high-performance data acquisition systems," in *Proc. 22nd Int. Conf. Mixed Design Integr. Circuits Syst. (MIXDES)*, Aug. 2015, pp. 153–157.

[16] P. Kasu, T. Kim, J.-H. Um, K. Park, S. Atchley, and Y. Kim, "FTLADS: Object-logging based fault-tolerant big data transfer system using layout aware data scheduling," *IEEE Access*, vol. 7, pp. 37448–37462, 2019.

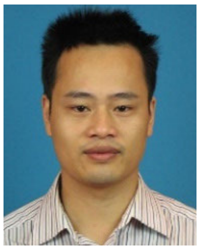
[17] H. Kim, I. Hwang, J. Lee, H. Y. Yeom, and H. Sung, "Concurrent and robust end-to-end data integrity verification scheme for flash-based storage devices," *IEEE Access*, vol. 10, pp. 36350–36361, 2022.

[18] G. R. Gowda, S. U. Shenoy, and S. Kumari, "Survey on transmission control (TCP) protocol for wired and wireless networks," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 5, no. 9, pp. 5–8, Jun. 2016.

[19] L. L. Chen and J. Huang, "Design and implementation of public bicycle rental system service platform based on IOCP and JSON," *Adv. Mater. Res.*, vols. 998–999, pp. 1100–1103, Jul. 2014.

[20] Q. Wei, B. Gong, S. Pathak, and Y. C. Tay, "FlashCoop: A locality-aware cooperative buffer management for SSD-based storage cluster," in *Proc. 39th Int. Conf. Parallel Process.*, Sep. 2010, pp. 634–643.

[21] V. H. Shah and A. Shah, "An analysis and review on memory management algorithms for real time operating system," *Int. J. Comput. Sci. Inf. Secur.*, vol. 14, no. 5, pp. 236–240, May 2016.
[22] R. Inglés, M. Orlikowski, and A. Napieralski, "A C++ shared-memory ring-buffer framework for large-scale data acquisition systems," in *Proc. 24th Int. Conf. Mixed Design Integr. Circuits Syst.*, Jun. 2017, pp. 161–166.
[23] R. K. Boggavarapu and S. Jiang, "Deduplication-aware I/O buffer management in the Linux kernel for improved I/O performance and memory utilization," in *Proc. 12th Int. Conf. Knowl. Smart Technol. (KST)*, Jan. 2020, pp. 70–74.
[24] Z. Xu, Y. Chen, and X. Zhang, "Design of serial image acquisition system based on camera link," in *Proc. 7th IEEE Conf. Ind. Electron. Appl. (ICIEA)*, Jul. 2012, pp. 1804–1809.
[25] X. Wan, B. Wang, P. Zeng, and S. Liu, "A design of data acquisition of high speed CCD camera," in *Proc. 8th IEEE Int. Conf. Dependable, Autonomic Secure Comput.*, Dec. 2009, pp. 743–745.
[26] J. Xia, B. Li, M. Guo, F. Sun, S. Yang, T. Luo, and Y. Chen, "Design and realization of four-channel image data acquisition for framing gated camera," in *Proc. Int. Conf. Multimedia Technol.*, Jul. 2011, pp. 4898–4900.
[27] D. Petrişor, C. Foşalău, and F. Măriut, "Video acquisition framework for high speed video measurements of human eyelid mobility," in *Proc. Int. Conf. Expo. Elect. Power Eng.*, Oct. 2012, pp. 857–860.
[28] M. Murugan and D. H. C. Du, "Rejuvenator: A static wear leveling algorithm for NAND flash memory with minimized overhead," in *Proc. IEEE 27th Symp. Mass Storage Syst. Technol. (MSST)*, May 2011, pp. 1–12.



ZHONGHUA HONG (Member, IEEE) received the Ph.D. degree in GIS from Tongji University, Shanghai, China, in 2014. He has been an Associate Professor with the College of Information Technology, Shanghai Ocean University, since 2019. His research interests include satellite/aerial photogrammetry, high-speed video grammetry, planetary mapping, 3D emergency mapping, GNSS-R, deep learning, and processing of geospatial big data.

SHIHAO SUN received the B.S. degree in communication engineering from the Zhengzhou University of Light Industry, Henan, China, in 2020. He is currently pursuing the M.S. degree in software engineering with Shanghai Ocean University, Shanghai, China.

His research interest includes high-speed camera processing.

XIAOHUA TONG (Senior Member, IEEE) received the Ph.D. degree from Tongji University, Shanghai, China, in 1999.

From 2001 to 2003, he was a Postdoctoral Researcher at the State Key Laboratory of Information Engineering in Surveying, Mapping, and Remote Sensing, Wuhan University, Wuhan, China. He was a Research Fellow at The Hong Kong Polytechnic University, Hong Kong, in 2006, and a Visiting Scholar at the University of California at Santa Barbara, Santa Barbara, CA, USA, from 2008 to 2009. His research interests include photogrammetry and remote sensing, trust in spatial data, and image processing for high-resolution satellite images.

RUYAN ZHOU received the Ph.D. degree in agricultural bio-environment and energy engineering from Henan Agricultural University, in 2007.

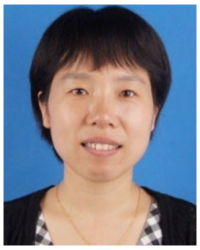
From 2007 to 2008, she worked with the Zhongyuan University of Technology. Currently, she is working with Shanghai Ocean University, Shanghai, China.

HAIYAN PAN received the Ph.D. degree in surveying and mapping from Tongji University, Shanghai, China, in 2020.

Currently, she has been a Lecturer with the College of Information Technology, Shanghai Ocean University. Her research interests include multitemporal remote sensing data analysis, change detection, and multispectral/hyperspectral image classification.

YUN ZHANG received the Ph.D. degree in applied marine environmental studies from the Tokyo University of Maritime Science and Technology, Tokyo, Japan, in 2008.

Since 2011, he has been a Professor with the College of Information and Technology, Shanghai Ocean University, Shanghai, China. His research interests include the study of navigation system reflection signal technique and its maritime application.

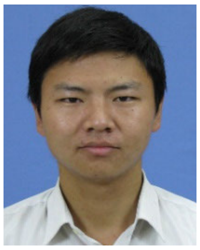


YANLING HAN received the B.E. degree in mechanical design and manufacturing and the M.E. degree in mechanical automation from Sichuan University, Sichuan, China, and the Ph.D. degree in engineering and control theory from Shanghai University, Shanghai, China.

She is currently a Professor and also working with Shanghai Ocean University, Shanghai. Her research interests include the study of ocean remote sensing, flexible system modeling, and deep learning.

JING WANG received the Ph.D. degree in biomedical engineering from the Department of Biomedical Engineering, Shanghai Jiaotong University, in 2014.

She has been a Lecturer with the College of Information Technology, Shanghai Ocean University, since 2015.



SHUHU YANG received the Ph.D. degree in physics from the School of Physics, Nanjing University. Since September 2012, he has been a Lecturer with the College of Information Technology, Shanghai Ocean University. His research interests include evolution of the antarctic ice sheet, hyperspectral remote sensing, and the use of navigational satellite reflections.

ZHENLING MA received the B.E. and M.S.E. degrees in photogrammetry and remote sensing from Central South University, Changsha, China, in 2008 and 2011, respectively, and the Ph.D. degree in geomatics from Wuhan University, Wuhan, China, in 2015.

From 2016 to 2018, she worked as a Postdoctoral Researcher at the College of Marine Sciences, Shanghai Ocean University, Shanghai, China. Since 2018, she has been a Lecturer with the College of Information Technology. Her research interests include georeferencing for satellite images and underwater photogrammetric engineering. She serves as a Reviewer for IEEE Transactions on Geoscience and Remote Sensing, IEEE Access, *Remote Sensing*, and *ISPRS International Journal of Geo-Information.*