Fast Anti-noise Compression Storage Algorithm for Big Data Video Images

Tao Lei(✉)

South China Normal University, Guangzhou 510006, China
leitao0l45@l63.com

Abstract. When calculating the traditional image compression storage algorithm, the key frames of the video image are mainly extracted by the video image feature. In the process of video image acquisition, the influence of factors such as light is detected, and the image features are changed, resulting in a large storage problem. A new anti-noise compression storage algorithm for big data video images is proposed. First, the collected big data video images are divided. The average value of the gray of the image sub-region is obtained, and then the compression process and the stored procedure are given. The actual working effect of the algorithm is verified by comparison with the traditional algorithm. The experimental results show that the improved algorithm is well stored and the error is small. The fast anti-noise compression storage method for the big data video images studied in this paper has a good storage effect, and its application range is wider and more worthy of promotion.

Keywords: Big data · Video image · Fast compression · Anti-noise compression · Storage algorithm

1 Introduction

Currently, many data center service models have undergone significant changes. In the era of big data, data sources are extremely rich. Especially such as images, the proportion of unstructured data such as video is increasing, how to efficiently compress large data images and store them has become a major problem in this field. In the current video image storage algorithm, the video image data is prone to loss due to noise interference. Because the storage speed of the same size file fluctuates continuously under the noise interference, the real-time storage of the high-speed data stream cannot be guaranteed. The traditional method uses the queue-based cache structure to solve the frame loss problem of the video implementation storage process, but the storage process cannot be guaranteed to be anti-noise. And the stability of the storage cannot be guaranteed. Compressing and storing large data video images can fully improve the accuracy and reliability of large data video image compression storage. The current compression storage algorithms used for big data video images are: optical flow analysis algorithm, MPEG-7 motion descriptor algorithm. Although the above algorithm can complete the compression and storage of video images in the smallest position, the calculation process is also relatively simple. But the effect of storage is often poor. So it takes a long storage time, the information extracted from a single

G. Gui and L. Yun (Eds.): ADHIP 2019, LNICST 302, pp. 355–362, 2019.
https://doi.org/10.1007/978-3-030-36405-2_35

video produces large errors. In response to the above problems, the algorithm is divided into two processes: compression and storage. The specific analysis of the calculation process, the experimental results and feasibility of the algorithm are verified by experiments [1].

2 Fast Anti-noise Compression Storage Algorithm for Big Data Video Images

During the rapid compression of big data video images, the user gives a key feature image of the big data video image to be queried according to different requirements. According to the sample given by the user, the compression system obtains the corresponding key characteristics of the video image key frame according to the requirements. Matching queries through the database, the results obtained are arranged in similar degrees. Effectively complete the compression of key frames of big data video images.

The compression algorithm is as follows:

$$R = \sum_{j=1}^{i} \frac{p_{ij}}{c_{ij}(i,j)} \tag{1}$$

Formula (1) represents the compression process of big data video images, a video with i frame images represented by j, p_{ij} represents a feature vector in a video image, compare the feature vector with the vector to be matched, get a compressed large video image, that is, the compression of the key information of the video image is completed [2].

After completing the fast anti-noise compression of big data videos, store it, the stored procedure is as follows:

$$L_i = \sum_{j=1}^{n} w_{ij} s_j \tag{2}$$

n in formula (2) represents the number of big data video images. j stands for video frame, i stands for key frame, w stands for data compression package, s represents the target data image storage package, L_i indicates the stored big data video.

The images stored in big data video images are different according to the storage technology. Divided into data by line storage and data by column storage, database table data storage supports both row and column storage [3]. Row storage is stored in units of records. The data page stores a complete number of records; column stores are stored in column units. All rows of data for each column are stored together, a segment stores only one column of data, and a designated page stores continuous data of a certain column.

There are several advantages to storing the list: the data of the same column is stored continuously. Can speed up the data query speed of a certain column (reduced unnecessary IO in relative memory); original interpretation: database column storage technology principle and implementation; continuously stored column data, with

greater compression unit and data similarity, can get much better compression efficiency than line storage; conditional scanning uses the statistical information of the data area for accurate filtering. Can further reduce IO, improve scanning efficiency (Smart Indexing) [4].

Column storage data video image storage table also called HUGE table, it is based on the HTS table space (full name HUGE TABLE SPACE). This table space is different from the normal table space. Ordinary table space, the data is passed through segments, cluster, page to manage, in addition, pages with fixed sizes (4K, 8K, 16K, 32K) are used as management units. HTS is equivalent to a simple file system. Create a HTS, it is actually creating an empty directory. After creating an HFS table, the database creates a series of directories and files under the specified HTS table space directory. The file system structure is shown in the following Fig. 1:

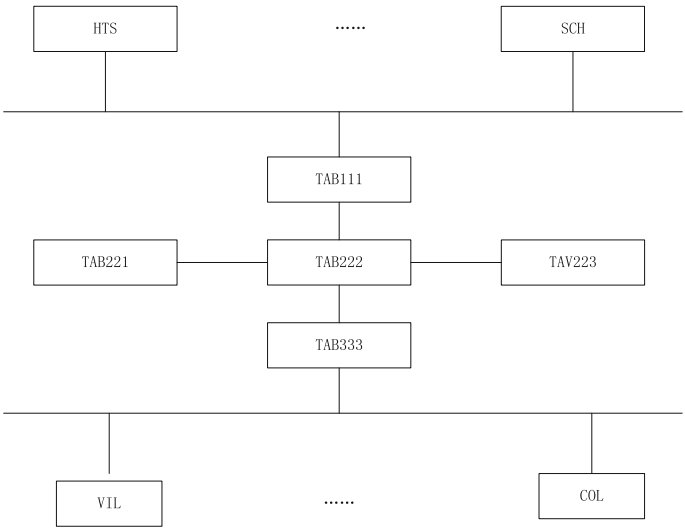

Fig. 1. Column storage file system structure

As can be seen from the above figure, the HFS table was created successfully in the HTS directory. The system needs to go through the following steps: first of all, create the schema directory corresponding to this table in the HTS directory. The directory name is "SCH + ID number 9" the composed string. Second, create the corresponding table directory [5] in the schema directory. Table catalog is the same reason, the table directory is a string consisting of "TAB + ID number of 4". The table directory contains all the files in this table. Second, when creating a new table, each column corresponds to a file with a dta suffix. The file size can be specified when the table is built. The default is 64M. The file name is "COL+ column number of 4 + file number of 10". For example, in the figure above, 0000 represents the first column, 0001 represents the second column…0000000000 represents the first file, 0000000001 represents the second file…there is only one file for the first column. As the amount of data continues

to grow, after a file has been lost, the system will automatically create new files to store the growing data [6]. For a file, its internal storage is managed by district. Area is the smallest unit of data management within the file. It is also the only unit (similar to the page that is stored). In a district, the number of rows that can hold a single column of data is specified when the table is created. For columns of the ABLE attribute, stored there is a corresponding logo. The start position and length of each zone are 4K aligned within the file. For an HFS table, the corresponding is also equipped with an auxiliary table to manage its data [7].

Because only the data is stored in the file described above, auxiliary tables are used to manage and assist system users to manipulate these data. The auxiliary table is automatically created by the system when creating the HFS table. Each record in the auxiliary table corresponds to a data area in the file. The auxiliary table includes the following 15 columns:

1. COLID: indicates the column ID of the column where the current record corresponds to.
2. SEC_ID: indicates the area ID number of the area corresponding to the current record. Each zone has an ID number and only;
3. FILE_ID: indicates the file number of the data in this area;
4. OFFSET: indicates that the data in this area is offset in the file, 4K aligned;
5. COUNT: indicates the total number of data stored in this area (which may include deleted data);
6. ACOUNT: indicates the actual number of data rows stored in this area;
7. N_LEN: indicates the length of the data stored in this area in the file, 4K aligned;
8. N_: indicates the number of rows included in the data in this area;
9. N_DIST: indicates the number of rows in this area that have different data from each other.
10. MAX_VAL: represents the maximum value in this area, the exact value;
11. MIN_VAL: represents the minimum value in this area, the exact value;
12. SUM_VAL: represents the sum of all the values in this area, the exact value;
13. CPR_FLAG: Indicates whether this area is compressed;
14. ENC_FLAG: Indicates whether this area is encrypted;
15. CHKSUM: Used to verify, this feature is not enabled yet.

The first seven columns are used to control data access. Based on this information, you can know the specific storage location of this area. Length and basic information. The last eight columns are all used for statistical analysis of this area. Among them, the key combination of COLID and SEC_ID is the clustered keyword of the auxiliary table.

In fact, the process of searching the data stored in the database is through the retrieval of auxiliary table information. The process of manipulating files under the HTS directory using auxiliary information [8].

3 Fast Anti-noise Compression Storage Process for Big Data Video Images

According to the above study, a fast anti-noise compression storage algorithm for large data video images compresses and stores images. The workflow is shown in Fig. 2 below:

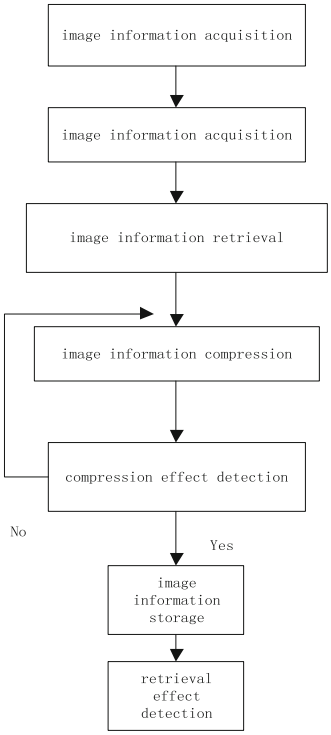

Fig. 2. Fast anti-noise compression storage process for big data video images

It can be seen from Fig. 2 that in the workflow of the fast anti-noise compression storage algorithm for large data volume video images, the video image information is first acquired, the acquired image information is retrieved, and the retrieved image information is compressed. The effect is detected, and the image information that passes the test is stored, and the image information that fails the test is re-retrieved. After the video image information is stored, the retrieval effect needs to be detected to ensure the accuracy of the retrieval.

Data storage coded frame implementation method: by storing the command, start the data storage state. There are two frame synchronization signals, the frame identifier EB90 [9] corresponding to the last two paths of each main frame. Three-way counting is included in the main frame. Respectively low counts, medium and high counts. The

low count determines the length of the sub-frame, when the low count is from 00 to 1F (hex, the same below), counting carry in low counter clearing, at the same time, the frame identification of the main frame is changed from EB90 to 146F (corresponding to the synchronization signal of the sub-frame at this time). In order to achieve a 32 × 32 full-frame data format. When the count counter reaches FF, clear the high count carry bit. You can identify whether the data record is lost by judging whether the frame count is continuous or not, wrong number. The data format for each sub-frame is as follows: when the low count is 00,01, insert the frame header. Count to 1E, record the current count and high count at 1F, the middle 28 frames records the operating status parameters in the system. The same position in the entire frame is the state of the same parameter at different times [10].

4 Experimental Study

In order to verify the practical work effect of the fast anti-noise compression storage algorithm for big data video images proposed in this paper, compared with traditional anti-noise compression, contrast experiment was set up.

4.1 Experimental Parameters

Experimental parameters are shown in Table 1:

Table 1. Experimental parameters

Project	Parameter
Compressed image size	1.0 GB–10.0 GB
The compression time	0.3 s
Storage time	0.5 s
Working frequency	300 Hz–1000 Hz
The storage system	Computer center system
Monitoring state	Hardware monitoring
Number of experimental sample video images	100n
System storage capacity	$576 \times 72 = 4.15 \times 10^4$ bits
Storage density	4.15×10^4 bits/0.01 cm^2 = 4.15×10^6 bits/cm^2

4.2 Experiment Process

Experiment according to the parameters set above, selecting the traditional anti-noise compression storage algorithm and the anti-noise compression storage algorithm studied in this paper simultaneously store a segment of the image. Record the storage effect.

4.3 Experimental Results and Analysis

The results obtained are as follows.

As can be seen from Tables 2 and 3, using an improved storage method, the storage error is 12%, storage accuracy is 95%, the storage time is 1.6 s; the traditional storage method has a storage error rate of 68.5%, storage accuracy is 71%, storage time is 4 s; it can be seen that using the improved algorithm for keyframe storage of large data video images reduces the error rate by 56.5% compared to the traditional method. Increased storage accuracy by 24%, the storage time is shortened by 2.4 s, its overall performance is better than traditional algorithms.

Table 2. Storage performance of traditional algorithms under interference

Type of interference	Error rate (%)	Accuracy (%)	Time consuming (min)
Gauss's vagueness	0.68	70	3.49
Luminance drift	0.68	72	4.31
Gamma correction	0.69	72	4.31
Video Mosaic	0.69	72	4.23

Table 3. Algorithm storage performance in this paper in the case of interference

Type of interference	Error rate (%)	Accuracy (%)	Time consuming (min)
Gauss's vagueness	0.1	95	1.21
Luminance drift	0.2	96	1.63
Gamma correction	0.2	95	1.42
Video Mosaic	0.1	96	1.69

4.4 Experimental Conclusion

Based on the above experimental results, get the following experimental results: the storage method studied in this paper has a good storage effect. Wide range of applications, more worth promoting.

5 Conclusion

For the current algorithm, there is a big error in large-data video images. A new fast anti-noise compression storage algorithm is proposed. Through the introduction of video metadata information algorithm to calculate the keyframe metadata of big data video images, the retrieval model of keyframes for big data video images can be established more accurately. Simulation results prove that, improved algorithm for keyframe storage of large data video images the error rate is reduced by 56.5%, increased storage accuracy by 24%, save time by 2.4 s, its compression storage effect is good, stable and win high, compared with the traditional method, the scope of application is wider.

References

1. Bello-Orgaz, G., Jung, J.J., Camacho, D.: Social big data: recent achievements and new challenges. Inf. Fusion **28**, 45–59 (2016)
2. Sowmya, R., Suneetha, K.R.: Data mining with big data. In: International Conference on Intelligent Systems and Control, pp. 246–250. IEEE (2017)
3. Aishwarya, K.M., Ramesh, R., Sobarad, P.M, et al.: Lossy image compression using SVD coding algorithm. In: International Conference on Wireless Communications, Signal Processing and Networking, pp. 1384–1389. IEEE (2016)
4. Masyarif, S., Kurniawan, A.: Harmony search algorithm with dynamic pitch adjustment rate and fret width for image compression. In: Multimedia and Broadcasting, pp. 66–72. IEEE (2016)
5. Bharathi, M., Janani, T.: Fractal image compression using quantum search algorithm. J. Comput. Theor. Nanosci. **14**(9), 4580–4585 (2017)
6. Gu, Y., Jiang, H., Xie, X., et al.: An image compression algorithm for wireless endoscopy and its ASIC implementation. In: Biomedical Circuits and Systems Conference, pp. 103–106. IEEE (2017)
7. Kamargaonkar, C., Sharma, M.: Hybrid medical image compression method using SPIHT algorithm and Haar wavelet transform. In: International Conference on Electrical, Electronics, and Optimization Techniques, pp. 897–900. IEEE (2016)
8. Zheng, F., Zhang, C., Zhang, X., et al.: A fast anti-noise fuzzy C-means algorithm for image segmentation. In: IEEE International Conference on Image Processing, pp. 2728–2732. IEEE (2014)
9. Krueger, J., Nicolai, M.: Sound generator for an anti-noise system for influencing exhaust noises and/or intake noises of a motor vehicle: US9374632 (2016)
10. He, G., Wei, Y.: An anti-noise fusion method for the infrared and the visible image based upon sparse representation. In: International Conference on Machine Vision and Information Technology, pp. 12–17. IEEE (2017)