mathematics

MDPI

Article

VSD: A Novel Method for Video Segmentation and Storage in DNA Using RS Code

Jingwei Hong [1,2,†], Abdur Rasool [2,3,†], Shuo Wang [1,4], Djemel Ziou [2] and Qingshan Jiang [2,*]

[1] College of Mathematics and Information Science, Hebei University, Baoding 071002, China; jw.hong@siat.ac.cn (J.H.); shuowang@hbu.edu.cn (S.W.)
[2] Shenzhen Key Laboratory for High Performance Data Mining, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China; rasool@siat.ac.cn (A.R.); djemel@siat.ac.cn (D.Z.)
[3] Shenzhen College of Advanced Technology, University of Chinese Academy of Sciences, Shenzhen 518055, China
[4] Key Laboratory of Machine Learning and Computational Intelligence, Hebei University, Baoding 071002, China
* Correspondence: qs.jiang@siat.ac.cn; Tel.: +86-755-8639-2340
† These authors contributed equally to this work.

Abstract: As data continue to grow in complexity and size, there is an imperative need for more efficient and robust storage solutions. DNA storage has emerged as a promising avenue to solve this problem, but existing approaches do not perform efficiently enough on video data, particularly for information density and time efficiency. This paper introduces VSD, a pioneering encoding method for video segmentation and storage in DNA, leveraging the Reed–Solomon (RS) error correction code. This method addresses these limitations through an innovative combination of segmentation and encoding, accompanied by RS coding to bolster error resilience. Additionally, the method ensures that the GC-content of the resultant DNA sequences remains around 50%, which further enhances the storage robustness. The experimental results demonstrate the method has commendable encoding efficiency and offers a solution to the prevailing issue of time inefficiency and error correction rates in DNA storage. This groundbreaking approach paves the way for the practical and reliable storage of large-scale video data in DNA, heralding a new era in the domain of information storage.

Keywords: DNA data storage; error correction; video storage; RS code; bio computing

MSC: 68U35

Citation: Hong, J.; Rasool, A.; Wang, S.; Ziou, D.; Jiang, Q. VSD: A Novel Method for Video Segmentation and Storage in DNA Using RS Code. *Mathematics* **2024**, *12*, 1235. https://doi.org/10.3390/math12081235

Academic Editors: Dariusz Świetlik, Mariusz Baran and Narcyz Knap

Received: 18 March 2024
Revised: 15 April 2024
Accepted: 17 April 2024
Published: 19 April 2024

1. Introduction

Over the last two decades, rapid advancements in information technology have triggered an unprecedented surge in data, creating a storage crisis. Scientific research and social services generate large video datasets, with the proliferation of personalized video creation and sharing on platforms like social networks contributing to this data deluge. In the digital age, video content dominates, constituting 53.72% of global data traffic. Predictions suggest that global data volume will reach 3×10^{21} bits by 2040 [1], exceeding the capacity of existing silicon-based storage, which has a limited lifespan and is unsuitable for long-term (~20 years) storage. Preserving valuable information for future generations requires regular transfers to updated storage media. Hence, there is an urgent need for alternative solutions to address the challenges posed by storing vast amounts of data efficiently [2–4].

As a potential solution to the escalating data storage challenges, researchers have turned to DNA (deoxyribonucleic acid), the carrier of biological and genetic information, due to its advantages of significantly higher storage density (1 petabyte per gram of DNA), an extended storage period of centuries, and notably lower maintenance costs [4,5]. Unlike

traditional magnetic storage, which relies on binary encoding, DNA storage utilizes quaternary encoding, employing specific combinations of A, C, G, and T to encode computer files. The DNA data storage system encompasses several processes [6–9], including binarization (transformation of digital data into binary code), encoding (translation of binary code into DNA code), DNA synthesis for data storage, DNA sequencing (to retrieve the nucleobase code), and decoding for the restoration of the original digital data, as depicted in Figure 1.

The impetus for DNA storage research increased, particularly post-2012, when Harvard University's Church group successfully stored a 650 KB book in DNA [10]. In 2013, Goldman [11] introduced the rotation encoding model, utilizing ternary Huffman compression coding to prevent homopolymer-induced nucleotide repeats in DNA sequences while maintaining a 50% GC content. In 2015, Grass [12] pioneered the application of Reed–Solomon (RS) error correction codes in DNA storage, introducing a ternary encoding model based on Galois fields. In 2017, Erlich [13] presented the DNA fountain code, incorporating Luby Transform for XOR operations on binary information with specific random number seeds and selecting sequences meeting biochemical constraints. In 2022, Ping [14] proposed the Yin-Yang dual encoding model, transforming two binary subsequences into one DNA subsequence based on the "Yin" and "Yang" rotation rules. These innovative models enhance coding density, advancing DNA data storage technology and its practical applications.

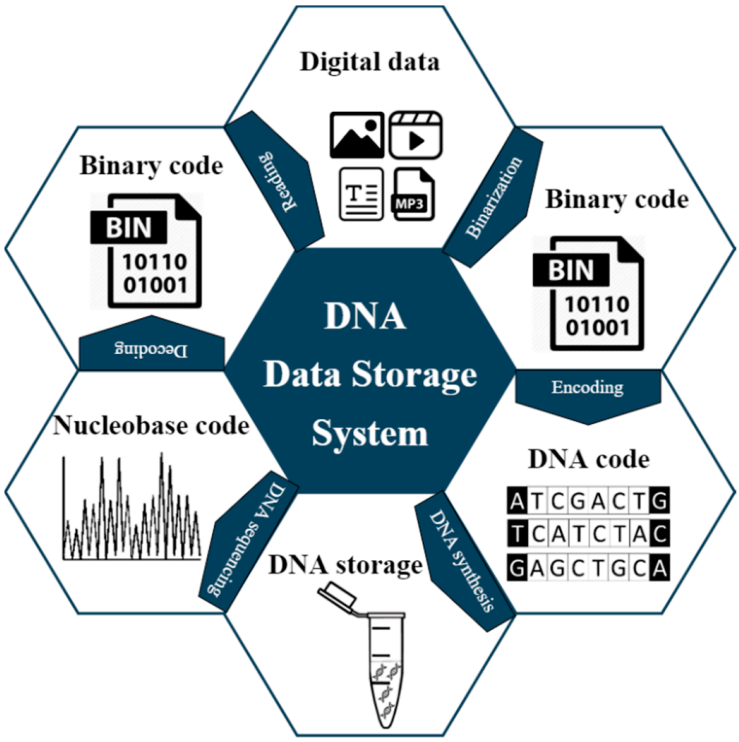

Figure 1. The key stages of DNA data storage system [14].

Despite these advancements, numerous state-of-the-art encodings [11,12,15] faced errors such as base deletions, mutations, etc., due to the limitations of DNA synthesis and sequencing technologies. To ensure DNA storage reliability [8], the error correction mechanism plays a crucial role, and researchers have developed various error correction techniques to correct these errors and enhance the stored data's accuracy and integrity. For instance, Goldman et al. [11] used quadruple overlapping redundancy, which means that each sequence generates three redundant sequences with 25% overlap. However, this method has a storage density of only 0.33 bits/nt, and the synthesis cost is too high. Bornholt et al. [15] utilized the method of heterodyne generation of a third sequence using two consecutive binary sequences, which solved the sequence-loss error, but the method

has extensive redundancy. Grass et al. [12] applied error correction techniques from the traditional communication field to DNA storage, using a combination of internal and external codes to improve the robustness of DNA storage; however, the method could not balance the GC content of the sequences. Since then, various error correction codes, such as RS codes [13,14,16–19], Hamming codes [20], LDPC codes [21,22], etc., have been used to correct base substitution errors within DNA sequences. Among these, RS codes have been applied widely. The error correction capability of these codes is positively correlated with the redundancy level because they are error-corrected at the binary level. Therefore, problems such as biochemical constraints [23] need to be considered when converting these codes to quadrature. Typically, researchers perform the binary segmentation step by adding RS codes before or after the segment, which are then converted together into DNA sequences. Erlich's code [13] can reproduce erroneous sequences with a storage density of 1.57 bits/nt; however, the decoding complexity is high and requires a lot of time for large files. Press et al. [19] developed "HEDGES" to handle addition and deletion errors in DNA synthesis and sequencing by using RS and convolutional codes for encoding and a tree structure for decoding, but the encoding rate of this method is not sufficient. Therefore, there is still a crucial demand for further improvements in error correction techniques to balance storage density, encoding rate, and time efficiency.

Apart from this crucial scientific problem, this work also emphasizes the types of data stored in DNA. As the source and importance of video data mentioned earlier, video is a data-intensive media type containing many images and sound information. For videos such as historical archives, documentaries, and important news reports, complete preservation can leave a valuable historical and cultural heritage for future generations, and thus, it requires long-term storage. However, traditional storage media usually require regular maintenance and replacement, which consumes many resources. Although DNA storage shows excellent potential, previous research has mainly focused on static data such as text documents and images. To the best of our knowledge, none of the DNA encoding models have efficiently stored the video data so far.

On the one hand, video contains a large amount of data, which requires efficient coding strategies and robust error correction mechanisms to handle the large amount of information, and the existing coding models are less time-efficient [13,14]. On the other hand, video needs to ensure continuity. The video quality needs to be maintained during storage and retrieval. The existing research encodes the video files in binary segments [13,16,24]. Due to the continuity of the video data, the binary error in a small segment may lead to catastrophic error propagation such that the video file is corrupted. Therefore, developing a segmentation strategy that guarantees video continuity is crucial.

Overall, previous studies have reported many advanced coding methods; however, when faced with large-volume data such as video, some coding models (Fountain [13] and Yin-Yang [14]) require exponential filtering steps, and the time required for coding is too long. In addition, it is critical to answer about how to add error correction codes more efficiently. Although Grass et al. [12] combined RS code coefficients into the coding process, which improved the efficiency of error correction to a certain extent, the method cannot realize the regulation of GC content when the data volume is large, which triggered more errors.

This study designs and tests a new coding method to deliver an effective solution for Video Storage in DNA (VSD) by addressing the challenges of previous coding methods. The proposed VSD method offers a novel video segmentation strategy that equates videos at certain time intervals to address the problem of error propagation under binary data segmentation. Additionally, VSD develops an innovative quadratic coding model based on RS error-correcting codes to convert video files into hexadecimal codes and DNA sequences, which significantly satisfies the biochemical constraints of DNA coding by derived theorems and improves the computational runtime. Meanwhile, it also introduces an efficient indexing mechanism for random video access. In the experiment, historically significant video data (approximately 300 MB) were used for testing, with an overall

encoding density of 1.75 bits/nt and a time efficiency improvement of over 30%, and it performs better than previous works when the base substitution rate is 0.05%. Through simulation sequencing and error correction, the original video is successfully decoded. By extending the capabilities of DNA storage to encompass videos, the proposed VSD opens up new possibilities for reliable video archival and storage applications. The data and codes for this work are available at https://github.com/jork07/VSD (accessed on 12 March 2024). The significant contributions of this study are summarized as follows:

- A novel VSD method that relies on an innovative video segmentation strategy and a quadratic coding model and uses efficient indexing to construct a video-based DNA data storage system is proposed.
- The proposed encoding model based on the RS error-correcting code efficiently balances storage density, combinatorial bio-constraints, and time efficiency, reducing overhead costs.
- The practicality of the proposed VSD method is evaluated by computer simulation, which reports the significant performance over previous work.

The structure of the remaining paper is as follows: Section 2 elaborates on the literature work on the overview of DNA storage and MPEG-4 format. Section 3 introduces the proposed video segmentation strategy and DNA transcoding model, Section 4 delivers experiments and results evaluations, and Section 5 concludes this study.

2. Literature Review

2.1. Error-Correcting DNA Codes

Error-correcting codes (ECCs) are crucial for DNA-based information storage due to the high rate of errors that arise during DNA synthesis and sequencing. These errors include substitutions of one base by another, as well as spurious insertions or deletions of nucleotides in the DNA strand (indels).

Grass et al. [12] introduced RS codes from the information field into the DNA storage system and designed a DNA data encoding method that includes inner and outer layers of error correction codes to improve the processing ability for single base mutations and whole sequence loss. This code can correct about 0.4% of sequences. Blawat et al. [16] proposed a forward error correction mechanism based on RS code, which protects DNA storage data blocks and block addresses by mapping binary to nucleotide sequences through modulation strategies. They successfully stored a 22 MB compressed video. Erlich et al. [13] explored the potential application of fountain coding in DNA storage by segmenting binary data into multiple droplets and discarding droplets with high GC and homopolymer content. They successfully converted 2.14 MB of data into 72,000 nt nucleotide chains stored in DNA without error recovery. Bornholt et al. [15] proposed the "XOR" redundant encoding scheme, which obtains a third payload by taking two payloads that are XOR each other. By recovering any two of the three strands, the third strand can be restored, greatly reducing error correction redundancy.

Press et al. [19] proposed the "HEDGES" error correction encoding algorithm, which uses RS and convolutional codes for encoding and a tree structure for decoding. They synthesized 5865 nucleotides with a length of 300 bp, then artificially introduced errors into these DNA nucleotides and sequenced them on the Illumina platform. The decoding results showed that HEDGES was able to handle a total of approximately 1.2% addition and deletion errors at the expense of a certain coding density.

2.2. Potential Challenges in Prior Studies

In contrast to text and image data, video data possess a greater and more intricate capacity. Previous approaches [12–14,16] frequently involved converting video data into binary format, employing encoding techniques that mirrored solutions for other datasets. However, such uniform approaches have proven to be problematic. The following provides succinct descriptions of these methodologies:

1. Blawat [16] transformed a compressed video of 22 MB into binary and then used their defined "bit-base" rules to convert it into a DNA sequence. However, this kind of storage is not practical for large-capacity video data. Firstly, the ability to play a video relies on accurate metadata. However, errors such as base substitutions, insertions, and deletions can occur during DNA data storage. If the video metadata are misread or lost, it would directly result in the video being unplayable by a video player. Therefore, it is necessary to provide special protection for metadata during the DNA storage process.
2. Secondly, existing high-density encoding methods, such as Fountain [13] and Yin-Yang [14], have slow processing speeds, especially when encoding large files, due to the "Screening" step involved. Since video files typically have a large amount of data, using these methods for video storage would lead to low time efficiency. Hence, there is a need for a compromise method that balances coding density and time efficiency.
3. During the DNA storage process, the error rate has led researchers to introduce error correction codes from the field of information transmission into the encoding process. In this context, the work of Grass [12] is worth mentioning. They incorporated Reed–Solomon error correction codes into the encoding and decoding algorithms. They transformed the digital sequence into coefficients in the Galois field GF(47) and assigned each coefficient to a triplet of bases. Finally, these triplets were concatenated to form a DNA sequence. This method effectively avoids the occurrence of single-base repeat sequences since it specifies that the last two nucleotides of each triplet cannot be the same. This ensures that the repeated bases during concatenation do not exceed three. It is worth noting that this method may not maintain a constant GC content in certain cases. This could affect DNA data storage's stability and read accuracy, for instance, when encountering secondary structures or errors in the sequencing process. Excessive errors may render the recovery of the original data impossible, even with the addition of error correction codes. Therefore, exploring a suitable solution to generate DNA sequences with a constant GC content is necessary.

Therefore, to ensure metadata integrity during the DNA storage process, it is essential to implement specialized protection measures. Balancing coding density and time efficiency is a critical compromise that must be sought, driving the exploration of optimal methods. Specifically, investigating solutions for maintaining a consistent GC content in DNA sequences is indispensable for the practical archival of video data within DNA storage systems. These initiatives represent significant advancements in the development of secure and efficient methodologies for data storage.

3. Proposed Methodology

In this research endeavor, we present an innovative methodology to address the identified challenges in Section 2.2. Our approach aims to enhance the practical archival of video data within DNA storage, ensuring metadata integrity, balancing coding density and time efficiency, and satisfying the bio-coding constraints. The structured delineation of our proposed methodology (depicted in Figure 2), Video Storage in DNA (VSD), unfolds across three pivotal stages.

1. Dynamic Video Parsing: The dynamic video file is segmented into multiple independent video segments; each video segment is parsed individually to separate the video metadata from the media information,
2. DNA Quaternary Coding: Hexadecimalizing the data and using a quaternary coding model based on RS codes to encode and decode, and satisfy the constraints with derived theorems,
3. DNA Synthesis, Storage, and Sequencing: DNA sequences are synthesized using existing techniques and preserved in a suitable environment, and the original sequence is obtained using sequencing techniques.

The comprehensive details are explicitly presented in the following subsections. It must be noted our novelty and primary contribution lie within the initial two stages. The third stage serves the purpose of providing a brief elaboration on end-to-end DNA data storage systems.

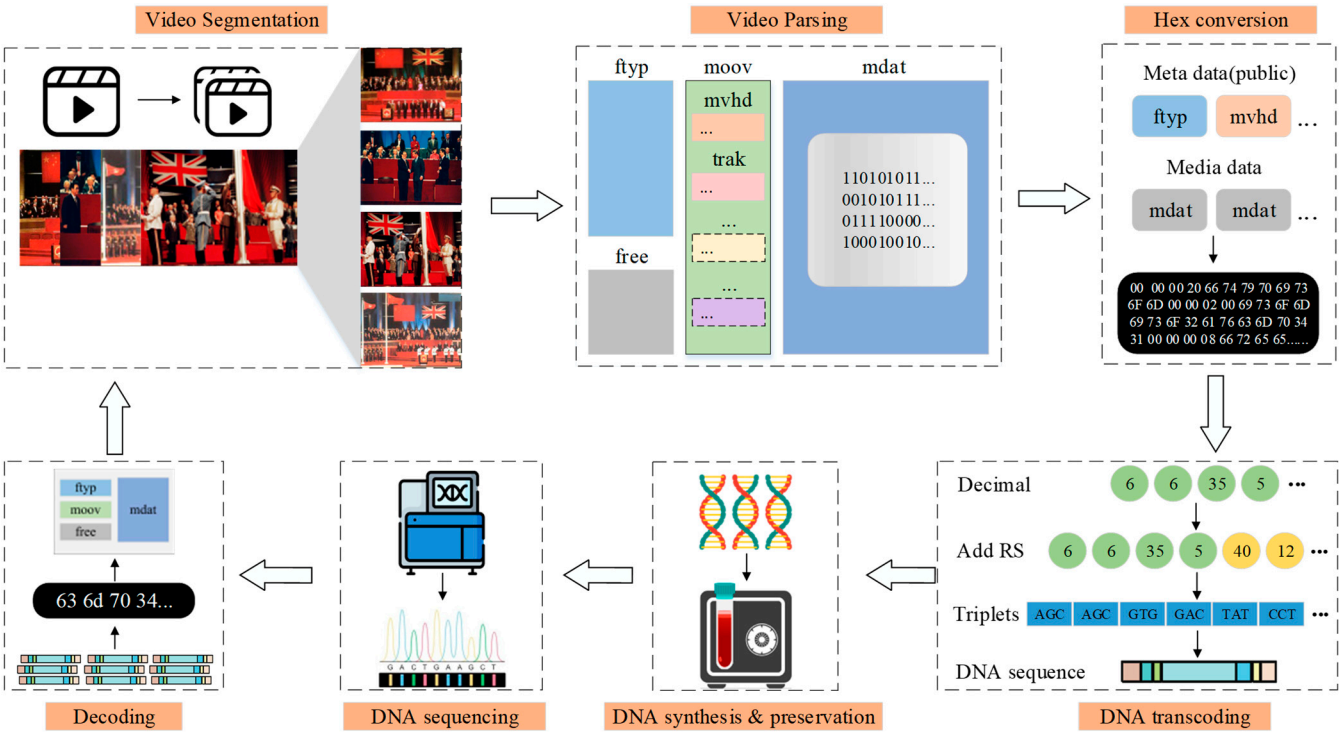

Figure 2. The systematic structure of our proposed VSD method incorporates novel dynamic video parsing and associated new quaternary encoding to synthesize, store, and retrieve the video data.

3.1. Dynamic Video Parsing

Most existing video data are compressed and stored based on MPEG-4 format. MPEG-4 (Moving Pictures Expert Group 4) is a digital multimedia compression standard designed to provide efficient methods for compressing, transmitting, and storing multimedia content. Compared to previous MPEG standards, MPEG-4 provides excellent compression capabilities and offers diverse functionalities such as multi-view video, transparent video, object recognition and coding, dynamic scene description, interactivity, and more. It makes MPEG-4 widely used in various fields [25,26]. In order to store video data efficiently in DNA, the VSD strategy dynamically parses videos, which mainly includes three steps: video segmentation, video parsing, and hexadecimal conversion.

3.1.1. Video Segmentation

Video segmentation is a crucial process in handling large amounts of video data. Traditionally, the video segmentation strategy divides videos into binary information segments [24,27]. However, this study proposes a novel approach to video segmentation by leveraging the organizational characteristics of MPEG-4.

Unlike traditional methods [24,27], the segmentation strategy employed in this study divides the video into segments based on time. These segments are not binary information segments but rather independently playable small segments akin to clips. By utilizing the inherent structure of MPEG-4, the video is divided into smaller units that retain their playability individually, offering advantages in terms of storage and retrieval. This innovative approach to video segmentation has several benefits: Firstly, it allows for more efficient storage and retrieval of video content. Instead of dealing with a single large file, the video

is divided into smaller segments, enabling faster access to specific parts of the video. Additionally, based on the organizational characteristics of MPEG-4, each segmented segment has common information that can be stored only once during the quaternary transcoding process. Each segment can reuse this information, improving storage density and storing videos more efficiently.

Overall, the proposed video segmentation strategy based on MPEG-4 organizational characteristics offers a novel and efficient approach to handling large video data. Dividing the video into independently playable segments enhances the storage and retrieval of video data in DNA.

3.1.2. Video Parsing

The MPEG-4 format uses a box-based structure where different types of data and metadata are organized into a series of nested boxes or containers, as shown in the left half (MP4 file) of Figure 3; an MPEG-4 file consists of four main boxes: file type (ftyp), movie fragment (moof), movie (moov), media data (mdat), and free space (free). This structure enables MPEG-4 to package and transmit various media content effectively [28,29].

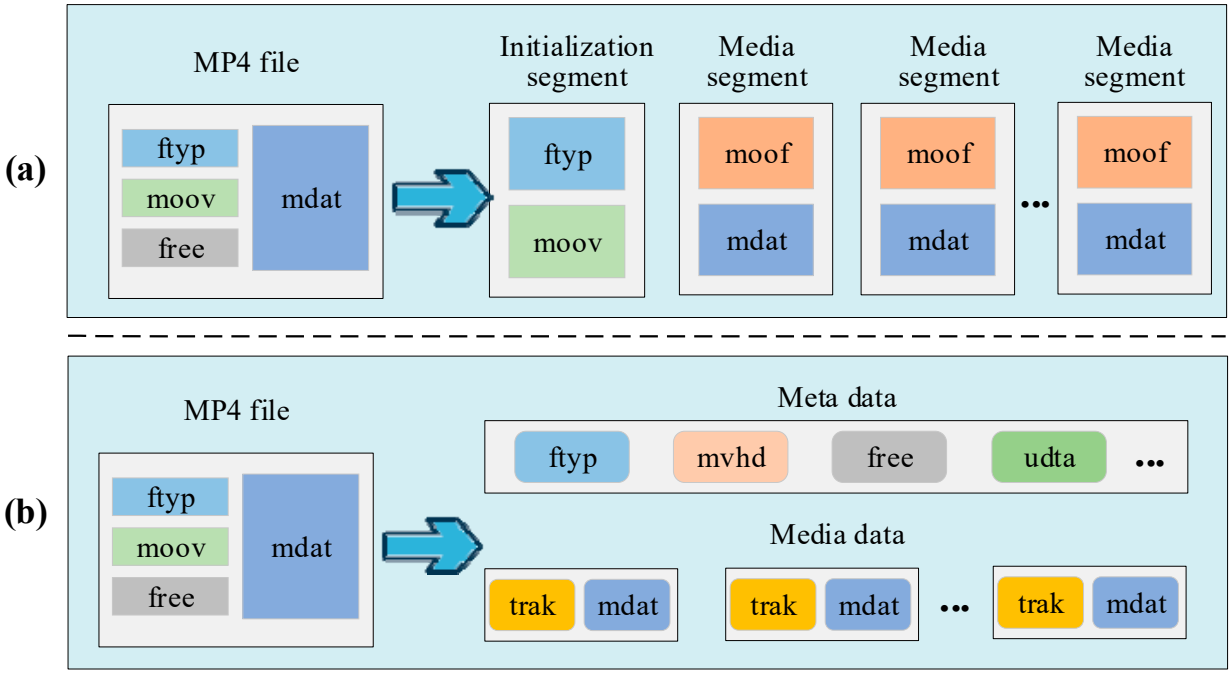

Figure 3. Fragmented MP4 stream consists of an initialization segment and a sequence of media segments (**a**), and VSD parsing strategy consists of metadata and a sequence of media data (**b**).

The FMP4 format (Fragmented MP4) is a video streaming format that extends the MPEG-4 Part 12 standard [30]. In contrast to the traditional MP4 format, the FMP4 format divides media files into several fragments. Each fragment constitutes a complete MP4 file containing media data, metadata, and index information, as illustrated in Figure 3a. Specifically, the initialization segment includes fundamental video information such as ftyp and moov, while subsequent multiple media segments encapsulate individual video streams, encompassing moof (movie fragment) and mdat boxes. Drawing inspiration from this format, our study proposes a novel strategy for parsing video data for DNA storage. Our aim is to bolster the resilience of video data storage and enhance data access efficiency.

Specifically, after the segmentation in the previous stage, each small segment is parsed into the form of the left half (MP4 file) of Figure 3. These segments all contain initialization information for the original video, so some metadata remain consistent. Storing these metadata (e.g., ftyp, mvhd (movie header), udat (user data), etc.) only once is sufficient to

decode and access these small segments, even the original video. In order to increase the coding density, all the small segments are divided into metadata and media data, as shown in Figure 3b, which are converted into hexadecimal data to be applied to the subsequent coding process.

3.1.3. Hexadecimal Conversion

Hexadecimal conversion is a common practice when working with MPEG-4 files, as it provides an intuitive way to observe the structure and content of the video files. Representing MPEG-4 data in hexadecimal format allows easy identification of file headers, data blocks, metadata, and other components. By examining the hexadecimal representation of a file, researchers and developers can analyze its features, parse data structures, and perform further processing and operations.

In the context of this study, we first perform hexadecimal conversion on the source data of the MPEG-4 video. Let X be the source binary data of the MPEG-4 video. The hexadecimal representation $H(X)$ is obtained by converting the binary data X to hexadecimal as follows:

$$H(X) = Hex(X) \tag{1}$$

This conversion transforms the binary data into a hexadecimal representation, making it easier to identify specific elements. For example, let B_i denote the ith box identifiers through hexadecimal representation. The hexadecimal identifier for each box is denoted by $I(B_i)$,

$$I(B_i) = HexId(B_i) \tag{2}$$

By using hexadecimal identifiers, such as "66 74 79 70" for "ftyp", we can locate different boxes within each video segment. If M_i represents the metadata data and D_i represent the media data within the $i-$ th segment, the separation can be represented as:

$$M_i, D_i = Sep(B_i) \tag{3}$$

Identifying these boxes through their hexadecimal representations can effectively separate the metadata and media data within each segment. The metadata and media data are separated and segmented into fixed lengths for subsequent DNA quaternary transcoding. The segmented metadata and media data are denoted by $\{I, j\}$ and $\{I_i, j\}$, respectively, where j represents the j-th segment within i-th segment.

$$\{M_i, j\}, \{D_i,\ j\} = Seg(M_i, D_i, L), \tag{4}$$

where L is the fixed length for segmentation.

3.2. DNA Quaternary Coding

This study improves the mapping rules from digital data to bases based on Grass et al.'s encoding and decoding method. It addresses the issue of the previously uncontrolled GC content by achieving similar coding density. In the original encoding process, a 16-bit binary bit sequence from every 2 bytes was converted into a numeric sequence in a base-47 system ($47^3 > 2^{16}$). Then, a simple mapping between the base-47 sequence and triplet bases was used to transform the binary bit sequence into a DNA sequence. This study first redesigned the mapping rules for triplet bases to maintain a constant GC content in the base sequence as much as possible. Tables 1–3 illustrate the partitioning of the original set of 48 triplets. Table 1 collects triplets with an equal count of A and T, while Table 2 includes triplets with an equal count of G and C. Moreover, this study switched to using Reed–Solomon (RS) error correction codes based on the Galois field GF(41) for encoding [31]. The choice of GF(41) was motivated by two reasons. Firstly, 41 is the nearest prime number to 40, which is the total number of elements in the AT and GC tables. This allows for a cross-mapping between the AT and GC (Tables 1 and 2) to control the GC content of the DNA sequence around 50%. Secondly, GF(41) satisfies $41^3 > 2^{16}$, ensuring

that two bytes can still be represented by three base-41 numbers while maintaining the same coding density.

Table 1. Mapping of A and T nucleobases.

Index	0	1	2	3	4	5	6	7	8	9
Triplet	ACA	TCA	AGA	TGA	CTA	GTA	AAC	TAC	ATC	TTC
Index	10	11	12	13	14	15	16	17	18	19
Triplet	AAG	TAG	ATG	TTG	CAT	GAT	ACT	TCT	ACT	TGT

Table 2. Mapping of G and C nucleobases.

Index	0	1	2	3	4	5	6	7	8	9
Triplet	CCA	GCA	CGA	GGA	CAC	GAC	AGC	TGC	CTC	GTC
Index	10	11	12	13	14	15	16	17	18	19
Triplet	CAG	GAG	ACG	TCG	CTG	GTG	CCT	GCT	CGT	GGT

Table 3. Remaining triplet for DNA encoding.

Remain_AT_triplet	ATA	TTA	AAT	TAT
Remain_GC_triplet	CGC	GGC	CCG	GCG

The pseudo-code for the quaternary coding process is presented in Algorithm 1, which has the following significant steps:

1. **Step 1:** The video data, divided in the previous stage, are converted into a hexadecimal sequence data H. To control the length of the DNA sequence, these sequence data are divided into several equally sized segments of hexadecimal sequences h_i .
2. **Step 2:** For each h_i , starting from the beginning, every 4 hexadecimal digits are converted into 3 base-41 numbers by performing a modulo operation, resulting in a base-41 sequence d_i.
3. **Step 3:** k RS error correction blocks are added to the base-41 sequence d_i. The error-correcting capability of the sequence is determined by k, which means the sequence can tolerate $k/2$ errors.
4. **Step 4:** Calculate the median of the base-41 sequence d_i and use that median as the offset. Define boolean indicator variables F_1 and F_2. Set up the triplet E_O, E_e, Z_O, and Z_e. Define the default mapping Table 4.
5. **Step 5:** For each element n in the base-41 sequence d_i, if $n == 40$, perform alternating mapping using an indicator variable F_1 to add additional base triplet E_O or E_e to the result sequence s_i's end, we have designed an integrated Algorithm 2, which can be stated as follows:
   - When the $offset$ is 0, if n equals 0, perform alternating mapping using the indicator variable F_2 to add additional base triplet Z_O or Z_e to the end of s_i; if n belongs to the range $(0,\ \ 20)$, add the n-th element from table AT to the end of s_i. Otherwise, add the $(n-20)$th element from table GC to the end of s_i.
   - When the $offset$ is less than 20, if n belongs to the range $[0,\ \ offset)$, add the nth element from table AT to the end of s_i. If n belongs to the range $[offset+20,\ 39]$, add the $(n-20)$th element from table AT to the end of s_i. Otherwise, add the $(n-offset)$th element from table GC to the end of s_i.
   - When the $offset$ is greater than 20, if n belongs to the range $[offset,\ 39]$, add the $(n-20)$th element from table GC to the end of s_i. If n belongs to the range $[0,\ \ offset-20)$, add the n-th element from table GC to the end of s_i. Otherwise, add the $(offset-n-1)$th element from table AT to the end of s_i.
   - When the $offset$ is 20, if n belongs to the range $[0,\ \ 20)$, add the n-th element from table AT to the end of s_i. Otherwise, add the $(n-20)$th element from table GC to the end of s_i.

6. **Step 6:** Process the offset by adding the offset-th element from the default mapping table D to the end of the result sequence s_i. Output the resulting sequence s_i

Algorithm 1. DNA encoding method.

Input: Video hexadecimal data $H(h_0, h_1, h_2, \ldots, h_n)$, RS error correction blocks k;
Output: DNA sequence set of one video $S(s_0, s_1, s_2, \ldots, s_n)$.

for h_i in H **do**
  for each 4 hexadecimal numbers in h_i **do**
    convert 4 hexadecimal numbers to a decimal number.
    perform 3 remainder operations on n over 41 to obtain 3 residues a, b, c.
    add 3 residues a, b, c to the decimal list d_i in order.
  end for
  add k RS error correction blocks for d_i.
  $offset = median(d_i)$.
  $s_i = Mapping\_triplet(d_i\ ,\ offset)$.
  $S.append(s_i)$
end for
return S

Algorithm 2. Mapping triplet algorithm.

Input: Decimal list d, $offset$, odd triplet table AT, even triplet table GC, default triplet table D, odd extra triplet E_O, even extra triplet E_e, odd zero triplet Z_O, even zero triplet Z_e, extra flag F_1 = True, zero flag F_2 = True.
Output: DNA sequence set s.

for each number n in d **do**
  if $n == 40$ **then**
    if $F_1 == True$ **then** $s_i.append(E_O)$, $F_1 = False$
    else $s_i.append(E_e)$, $F_1 = True$
  else
    if $offset == 0$ **then**
      if $n == 0$ **then**
        if $F_2 == True$ **then** $s_i.append(Z_O)$, $F_2 = False$
        else $s_i.append(Z_e)$, $F_2 = True$
      else if $n < 20$ **then** $s_i.append(AT[n])$
      else $s_i.append(GC[n-20])$
    else if $offset < 20$ **then**
      if $n < offset$ && $n \geq 0$ **then** $s_i.append(AT[n])$
      else if $n \geq 20 + offset$ **then** $s_i.append(AT[n-20])$
      else $s_i.append(GC[n-offset])$
    else if $offset > 20$ **then**
      if $n \geq offset$ **then** s_i.append(GC[n − 20])
      else if n $< offset - 20$ **then** $s_i.append(GC[n])$
      else $s_i.append(AT[offset-n-1])$
    else
      if $n < 20$ **then** $s_i.append(AT[n])$
      else $s_i.append(GC[n-20])$
end for
$s.append(s_i)$
return s.

The computational complexity of the proposed VSD strategy is $\mathcal{O}$(n*m), where n is the number of hexadecimal data sequences H, and m is the number of 4-digit hexadecimal numbers in each sequence.

Table 4. Default triplet table.

Index	0	1	2	3	4	5	6	7	8	9
Triplet	ACA	CCA	TCA	GCA	AGA	CGA	TGA	GGA	CTA	CAC
Index	10	11	12	13	14	15	16	17	18	19
Triplet	GTA	GAC	AAC	AGC	TAC	TGC	ATC	CTC	TTC	GTC
Index	20	21	22	23	24	25	26	27	28	29
Triplet	AAG	CAG	TAG	GAG	ATG	ACG	TTG	TCG	CAT	CTG
Index	30	31	32	33	34	35	36	37	38	39
Triplet	GAT	GTG	ACT	CCT	TCT	GCT	AGT	CGT	TGT	GGT

3.2.1. Integration of RS Codes

Our study's proposed DNA quaternary coding uses RS codes to add redundancy and facilitate error correction. The use of RS codes based on the Galois field GF(41) allows for efficient mapping between the digital data and the four nucleotides of DNA while maintaining a balanced GC content. Our method encodes the data into hexadecimal form, then decimal, and finally into residues of 41 to comply with the RS code over GF(41). The core idea of RS codes is to encode data into polynomials over a finite field (also known as a Galois field). The key parameters of an RS code are denoted as RS$(n,\ k)$ with n symbols, meaning that the encoder takes k data symbols and adds $n-k$ parity symbols to make an n-symbol codeword. Each symbol can be seen as a base-41 value, and therefore, the RS code operates over GF(41). A k-symbol message is represented as a polynomial $M(x)$ of degree $k-1$ [32]:

$$M(x) = m_{k-1}x^{k-1} + m_{k-2}x^{k-2} + \cdots + m_1 x + m_0 \tag{5}$$

where each coefficient m_i represents a symbol from the field GF(41).

The generator polynomial $G(x)$ is used to encode the message and is of degree $n-k$:

$$G(x) = \left(x - \alpha^1\right)\left(x - \alpha^2\right)\cdots\left(x - \alpha^{n-k}\right) \tag{6}$$

where $\alpha^1, \alpha^2, \ldots,\ \alpha^{n-k}$ are called the roots of the generator polynomial and are chosen from the field GF(41).

The encoding process involves creating a generator polynomial, which is multiplied by x raised to the power of $n-k$ (to make space for the parity symbols) and then divided by the generator polynomial $G(x)$.

$$C(x) = \left(M(x) * x^{n-k}\right) mod\ G(x) \tag{7}$$

where the resulting polynomial $C(x)$ represents the encoded message.

When errors occur during transmission, the received polynomial $R(x)$ differs from the transmitted polynomial $C(x)$. The decoder uses various methods, such as the Berlekamp–Massey algorithm [33] or the Euclidean algorithm [34], to correct the error locations. The decoder calculates the syndrome polynomial $S(x)$ by evaluating $R(x)$ at the roots of the generator polynomial:

$$S(x) = R\left(\alpha^1\right) + R\left(\alpha^2\right)x + \cdots + R\left(\alpha^{n-k}\right)x^{n-k} \tag{8}$$

By finding the roots of the syndrome polynomial, the decoder identifies the error locations and corrects the received polynomial $R(x)$ accordingly. Finally, the errors are corrected, and the original message is recovered.

3.2.2. DNA Coding Constraints

Furthermore, the generated DNA sequences undergo validation based on biological coding constraints, specifically focusing on GC and homopolymer constraints in our study. We applied newly derived theoretical models from our previous work [23], emphasizing

the limitations of GC content ω within the range of 40–60% and homopolymer ≤ 4 based on benchmark studies [35–38]. Analogously, GC can be represented as $A_4^{GC}(n, d, \omega)$ for the four nucleobases, ensuring that all DNA segments adhere to the desired code ω and constraints. The ensuing lower bounds constraints for constructing the DNA library are articulated in Theorem 1 [23,39], incorporating variables for sequence length n and Hamming distance d. The concise proof is provided in this work due to these variables.

Theorem 1. *For DNA sequence having the number of segments $n > 0$, with constraint $0 \leq d \leq n$ and $0 \leq \omega \leq n$ for the bound,*

$$A_4^{GC}(n, d, 0) = A_2(n, d) \tag{9}$$

$$A_4^{GC}(n, d, \omega) = A_4^{GC}(n, d, n - \omega) \tag{10}$$

$$A_4^{GC}(n, 1, \omega) = \begin{cases} \frac{1}{2}\left(\binom{n}{\omega}2^n - \binom{n/2}{\omega/2}2^{n/2}\right), if\ n\ is\ even\ and\ \omega\ is\ even \\ \frac{1}{2}\binom{n}{\omega}2^n \qquad\qquad\qquad\qquad , if\ n\ is\ odd\ or\ \omega\ is\ odd \end{cases} \tag{11}$$

Proof. Equation (9) signifies an equivalence between DNA codes A_4^{GC} with zero GC content and binary codes A_2. It illustrates that transforming binary codes by replacing 0s with As and 1s with Ts maintains Hamming distance and thus establishes a bijection between these two sets. This transformation establishes a one-to-one correspondence, demonstrating the interchangeability of binary and DNA codes with specific GC content, which is crucial in DNA storage. In Equation (10), the interchange operation is demonstrated to uphold the equality between the sets with different GC-contents (n and $n - \omega$). Swapping complementary nucleobases in a DNA sequence preserves the constant GC content, illustrating symmetry within the set of DNA codes. Equation (11) establishes a formula for the count of DNA codes A_4^{GC} with specific GC-content ω and Hamming distance of 1 $(n - 1)$. The proof considers two scenarios:

1. When n and ω are both even: in this case, the Equation calculates the count of words with GC-content ω that are their own reverse complements.
2. When either n and ω is odd: the formula simplifies to $\frac{1}{2}\binom{n}{\omega}2^n$, signifying that no DNA codes possess the property of being their own reverse complement.

It succinctly states that when both n and ω are even, there are $\binom{n/2}{\omega/2}2^{n/2}$ words with GC content that are their own reverse complements; otherwise, there are none. □

Eventually, we demonstrated an example of quaternary transcoding, in which we define E_O = TAT, E_e = CGC, Z_O = ATA, Z_e = GCG; Figure 4 is illustrated. The original video is segmented into a fixed-length sequence after hexadecimal conversion, using Equation (4). In this sequence, four hexadecimal bits are used as a unit, and each unit is first converted to a decimal number. Then, the remainder of 41 is calculated one by one to obtain three residues. For example, 60223/41 = 1468 (35), 1468/41 = 35 (33), 35/41 = 0 (35), where the remainder is enclosed in parentheses. By analogy, when all units are converted into three residues, RS codes are added later to obtain a base-41 sequence. Next, calculate the median of the sequence as the offset. In the above example, the median is 30 (round down). According to Algorithm 2, number 35 corresponds to line 23 and is therefore converted to a triplet (GTG) with GC table index 15; number 33 conforms to line 25 and is converted to a triplet (CTA) with an AT table index of 4; number 3 conforms to line 24 and is converted to a triplet (GGA) with GC table index 3; number 40 conforms to line 7 and is converted to a triplet E_O(TAT). Finally, the triad was integrated into a

sequence called "GTGTCGGTGCTAGGAGAGCGTCGCTTGATCTTCTCAGAT", which showed homopolymer < 3, and a GC content of 54%.

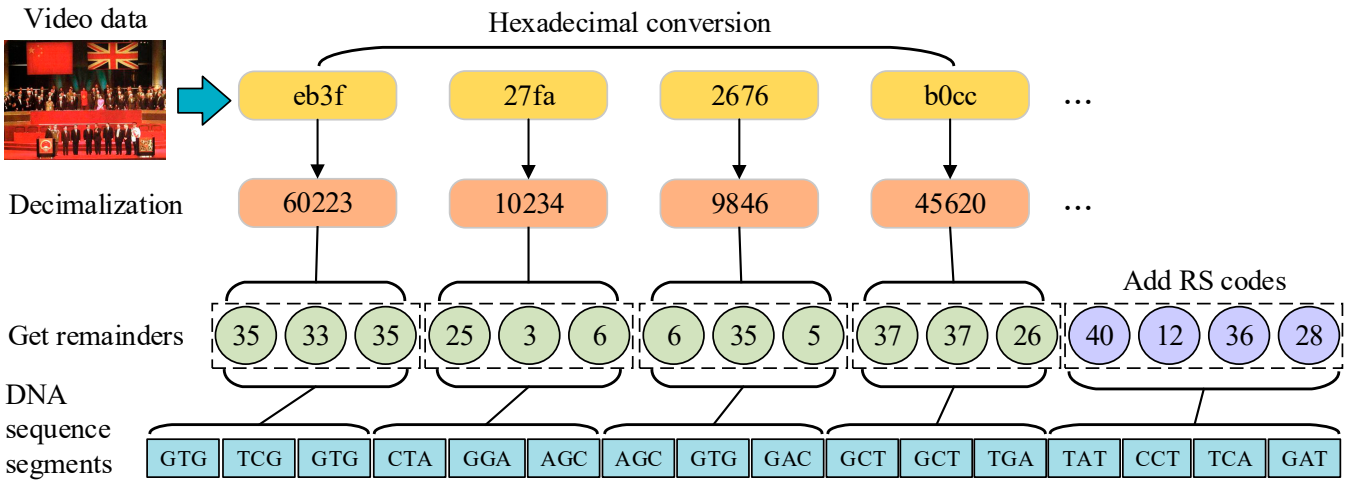

Figure 4. An example of the proposed quaternary transcoding process to convert video data to DNA sequence data.

3.3. DNA Synthesis, Storage and Sequencing

Our primary focus is on computer simulations and computational biology, and it is important to clarify that our work does not directly involve the experimental processes of DNA synthesis, storage, and sequencing. Nevertheless, we present a concise overview of these procedures for the benefit of early researchers and readers seeking a general understanding of DNA manipulation techniques.

In the synthesis phase, oligo pools are synthesized by companies to provide DNA gel for subsequent sequencing. When considering storage options, DNA fragments intended for in vivo storage undergo careful segmentation into subfragments and further division into blocks. These blocks consist of synthesized 80-nt oligos and are assembled using polymerase cycling assembly. Subsequently, they are cloned into vectors for Sanger sequencing, validating their sequence accuracy. The sequencing-verified blocks are then further assembled into subfragments using overlap extension PCR. Finally, the complete full-length DNA fragment, approximately 10 MB in size, is obtained by introducing the subfragments into yeast through homologous recombination, enabling in vivo storage.

However, in addition to in vivo storage, an alternative approach is in vitro storage. For in vitro storage, DNA fragments are not transferred into living organisms but instead preserved in laboratory conditions. This method offers advantages such as ease of manipulation and reduced reliance on living systems. In this context, the storage process entails carefully preserving the DNA fragments in a controlled environment, typically using techniques like cryopreservation or freeze-drying. These methods ensure the long-term stability and integrity of the DNA molecules, allowing for extended storage periods without the need for living organisms.

The choice between in vivo and in vitro storage depends on various factors. In vivo storage is preferred when DNA fragments need to be maintained and propagated within a living organism. It allows researchers to utilize the host organism's cellular machinery, making it useful for studying complex genetic interactions. In contrast, in vitro storage is selected when the primary goal is preserving DNA fragments themselves without immediate utilization in a living system. It offers long-term stability and easy accessibility, enabling straightforward manipulation such as retrieval and amplification. In vitro storage is more suitable for archiving. These methodologies are crucial in genetic research, synthetic biology, and biotechnological applications.

4. Experiments and Results Validation

All experiments were conducted under Windows 10 × 64, Intel Core i7 3.41 GHz, RAM 16 GB, and Python 3.8.13v language. To begin with, the data used in the experiments were obtained from the Internet, and the segmentation and integration of video data were realized using ffmpeg 6.1 (a cross-platform audio/video processing framework) [40]. Additionally, the data processing involved utilizing two Python packages, codecs (https://docs.python.org/3/library/codecs.html, accessed on 1 March 2024), and binascii (https://docs.python.org/3/library/binascii.html, accessed on 1 March 2024), which played a crucial role in data processing tasks. For RS code generation and error correction, we use the galois (https://galois.readthedocs.io/en/v0.3.6/, accessed on 1 October 2023) package.

In order to evaluate the performance of the proposed VSD method, we first encoded a series of digital files and analyzed the biological constraints on the encoding results. We tested the encoding on video files and other types of files and compared them comprehensively with previous studies. (For other types of files, we apply VSD method without the video segmentation and parsing steps). We have also compared the efficiency of error correction and encoding time efficiency.

4.1. Biological Constraint Validation

Most errors that occur during DNA storage are caused by sequences that do not conform to biological constraints. Researchers are most concerned with homopolymer and GC content among the many biological constraints. For this reason, we performed extensive tests using proposed quaternary coding with computational theorems to satisfy the constraints.

For homopolymers, we encoded several digital files, totaling 177,360 sequences of length 100 nt, and counted the number of repetitions of homopolymers of three different lengths; as shown in Figure 5, there is no homopolymer of length > 3 in the resultant sequences encoded by the VSD method. In addition, the number of times that homopolymers of length 2 or 3 occur in the four bases does not differ much. The results show that the VSD method can avoid the generation of longer homopolymers.

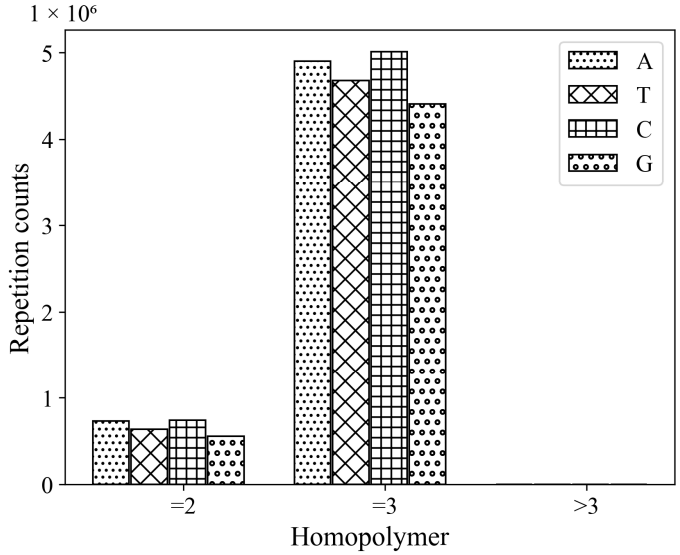

Figure 5. Validation of homopolymer constraints with varying repetition counts.

For GC content, 2000 DNA sequences were randomly selected from the encoding results for comparison. Figure 6 compares the GC content results of the generated sequences with Grass et al.'s encoding model (a) and VSD encoding model (b). The VSD method exhibits superior regulation of the GC content in the generated DNA sequences when compared to Grass et al.'s encoding model [6]. This improvement can be attributed to the

VSD method's effective base triplet mapping, which ensures a more balanced distribution of G and C nucleotides throughout the sequences. By avoiding longer homopolymers and achieving a more balanced GC content, the VSD method can reduce non-specific hybridization during DNA sequence synthesis, thereby improving the robustness of DNA storage.

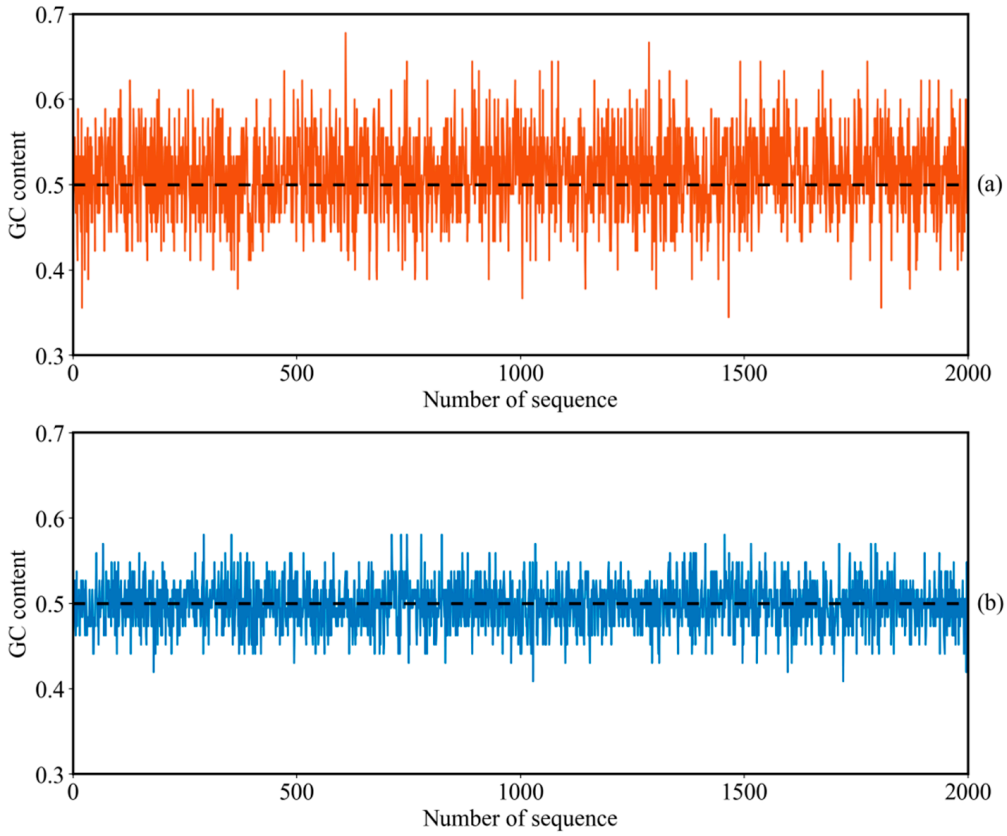

Figure 6. Comparison between the GC content of Grass [11] encoding method (**a**) and our proposed VSD encoding (**b**).

4.2. Encoding Efficiency Validation

The evaluation of the computer simulations (Table 5) reveals insightful findings regarding the proposed VSD method for video data storage. Notably, the density of DNA sequences, a crucial metric for assessing storage efficiency, exhibits varying trends across different file types and sizes.

In our test video file, the maximum duration of a single video is "Macao1999", which is 12 min and 18 s. The total duration of all videos is 25 min and 51 s. To our knowledge, there has been no research on encoding videos of this duration. The highest density is observed in the "FoundingCeremony" MP4 file at 1.79 bits/nt, while the lowest density is recorded in the "Les Miserables" TXT file at 1.73 bits/nt. The average density across all files is consistently competitive, showing the robustness of the VSD method. Intriguingly, the results illustrate that density tends to increase with larger file sizes, emphasizing the scalability of the proposed approach. The observed fluctuations in density among different file types underscore the nuanced impact of file format on storage efficiency. The VSD method demonstrates efficacy in achieving favorable density levels, coupled with reduced computational time, affirming its potential as a promising solution for video data storage. This is particularly noteworthy given the method's unique features, including the novel video segmentation strategy, quadratic coding model, and efficient indexing mechanism, all contributing to improved storage density and computational complexity.

Table 5. Comparative analysis of VSD encoding efficiency across multiple parameters after video segmentation.

	Data	File Type	Size (MB)	Sequence	Length	Density	GC	Time (s)
1	FoundingCeremony	MP4	3.97	152109	123	1.79	49	14.334
2	BandungConference	MP4	9.68	372998	123	1.77	51	35.412
3	Macao1999	MP4	36.8	1402112	123	1.79	50	135.155
4	HongKong1997	MP4	79.9	3061339	123	1.78	50	298.812
5	Sample_12x720	FLV	4.71	183752	123	1.75	49	14.226
6	Ocean with audio	MKV	16.5	646835	123	1.74	51	51.275
7	Surfing with audio	AVI	25.4	989928	123	1.75	50	65.422
8	Les miserables	TXT	1.70	67217	123	1.73	50	5.978
9	The Beatles	MP3	5.36	210192	123	1.74	52	15.652
10	Mona Lisa	JPG	1.41	55383	123	1.74	49	1.953

Furthermore, a 3D representation (Figure 7) offers valuable insights into the intricate relationships between file size, internal complexity, and embedded information across diverse media formats. The graph reveals several intriguing relationships. We observe a general trend of size increasing with density. This suggests that denser files, packing more information per unit space, tend to be larger in size. However, deviations from this trend are also evident. For instance, according to the data in Table 5, "Mona Lisa" (.JPG) is positioned at the far left on the "Size" axis. Despite having a high density, it occupies minimal space due to its inherent compression efficiency. Interestingly, the sequence values appear unrelated to either size or density. They act as independent identifiers for each file type, potentially encoding specific media formats or internal structures. Further investigation would be needed to uncover the nature and role of these sequence values.

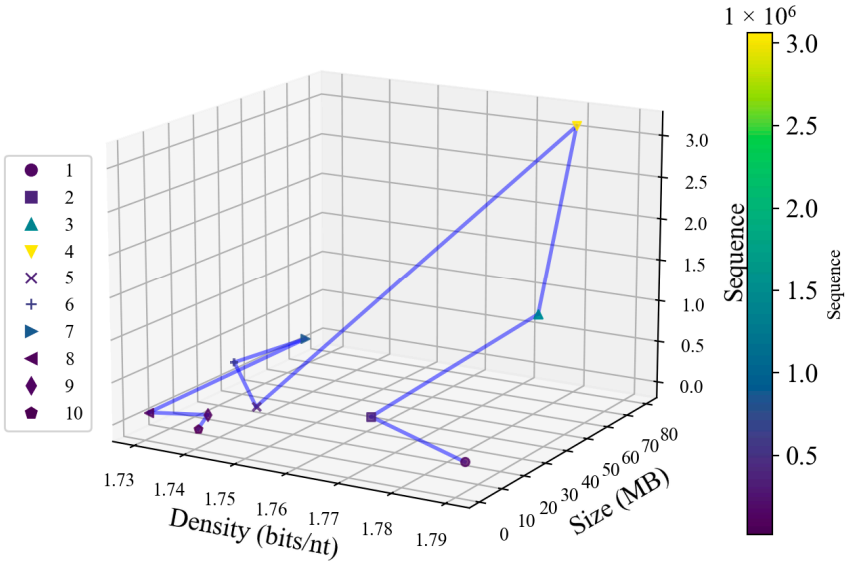

Figure 7. Exploring the interplay of file size (MB), density (bits/nt), and DNA sequence values based on Table 5 by utilizing the VSD method.

4.3. Error Correction Performance

The realm of DNA data storage faces inherent challenges due to base mutations, predominantly arising during DNA synthesis and sequencing. Factors such as the concentration or temperature of base addition reagents, the quality of DNA templates, and the sequencing instrument's light source intensity or detection sensitivity, significantly influence the incidence of base mutations. These mutations, particularly base substitutions, can lead to data corruption, posing a substantial obstacle in accurate data decoding. In

our VSD method, leveraging RS error correction codes, emerges as a pioneering solution. Figure 8 illustrates the efficiency of VSD in comparison to other benchmark studies like Erlich [13], Grass [12], and Tong [41]. The chart distinctly shows VSD's superior performance in maintaining high successful decoded rates, even at elevated base substitution rates. For instance, at a 0.05% error rate, VSD demonstrates an impressive ability to correct 30% of errors, a feat not mirrored by its counterparts.

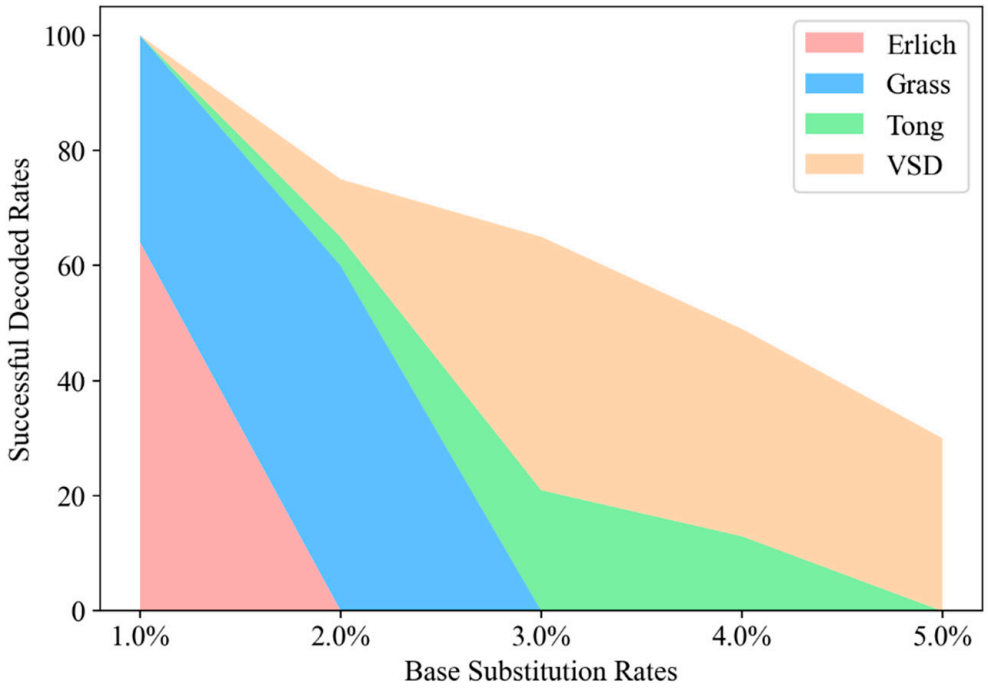

Figure 8. Comparative analysis of successful decoding rates at varied base substitution levels in DNA data storage.

The VSD method's success in DNA data storage is due to its unique blend of video segmentation and encoding, enhanced by robust RS coding. This combination improves error correction, particularly for substitution errors, and strengthens data integrity and reliability. As shown in Figure 8, VSD demonstrates remarkable error resilience and efficiency, establishing a new standard in the field and addressing key challenges of capacity and accuracy.

4.4. Large-Scale Compatibility

The VSD robustness is assessed with large-scale public video datasets (https://github.com/VSD/video_data, accessed on 12 March 2024). The experiments are conducted in a high-performance environment compressed with Ubuntu 22.04 system, AMD EPYC 7763@3.53Ghz CPU, 1024 GB memory. Figure 9 is a compelling visual affirmation of the VSD method's proficiency, particularly in handling and encoding large-scale video datasets efficiently. The proportional relationship between video file size and processing time, depicted by the ascending bars, evidences the VSD's adept handling of increased data volumes without significant time penalties. Notably, the density trend, illustrated by a brown line and scaled on a secondary y-axis, indicates a consistent density increase with video file sizes. This suggests the VSD method's capability to enhance encoding quality, possibly due to higher video resolutions and bitrates, without sacrificing biochemical compatibility or efficiency. These observations underscore the VSD's innovative coding model and indexing mechanisms' success, which notably outstrips prior models in both time efficiency and storage density. The graph thus solidifies the VSD method's potential

for revolutionizing DNA-based video archival systems, aligning with the study's aim to establish a fast, reliable, and scalable video data storage solution.

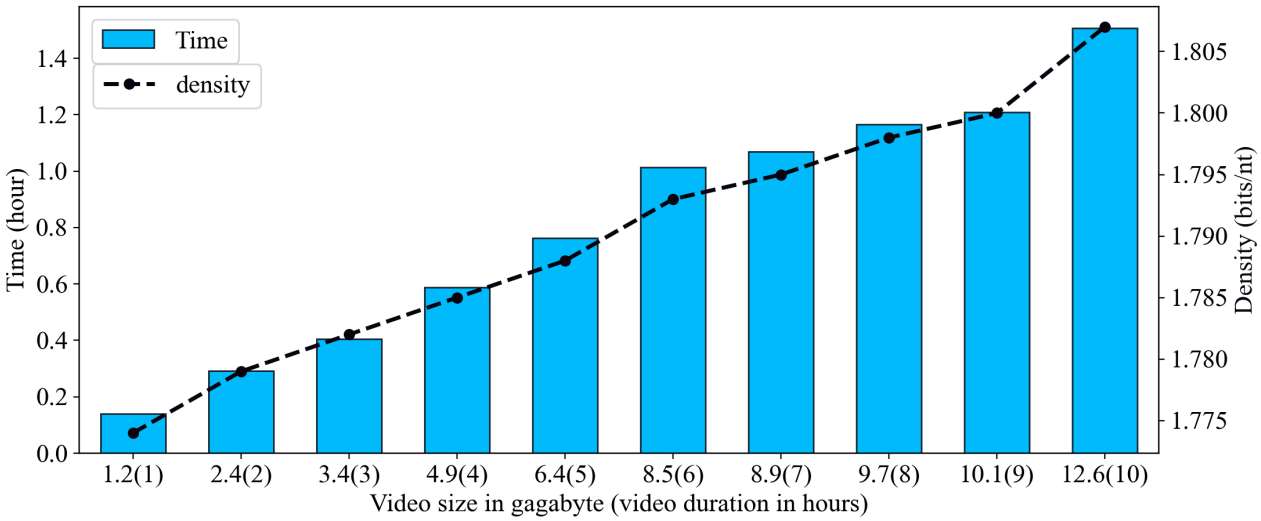

Figure 9. Comparative analysis of large-scale video datasets with computation time and density.

4.5. Computational Time

In addition to the aforementioned analysis, we compared the encoding efficiency of the proposed encoding strategy. This evaluation aimed to assess how efficiently the VSD method performs compared to previous research work. The results of this comparison are presented in Figure 10.

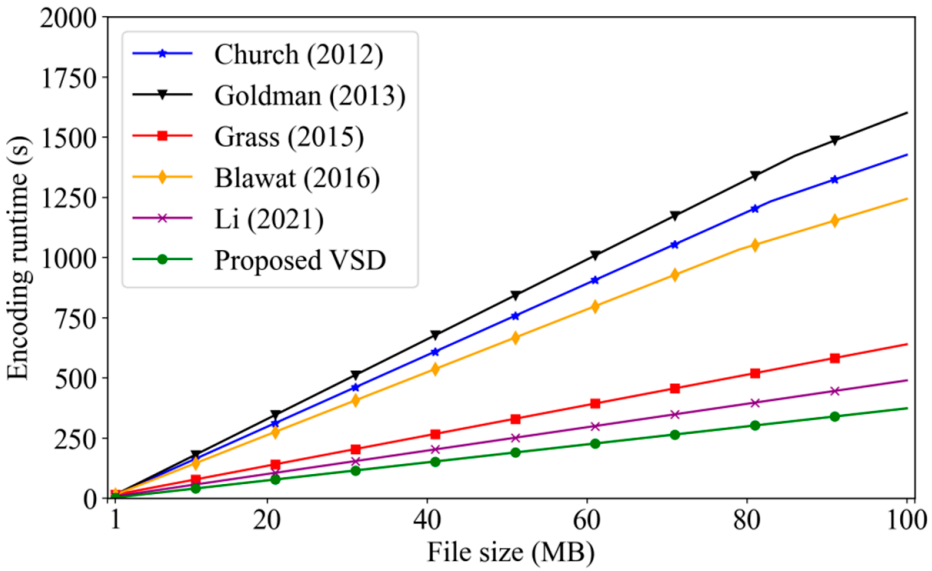

Figure 10. Comparative trend encoding computation time of proposed VSD method with state-of-the-art methods [10–12,16,42].

Figure 10 clearly illustrates the encoding time for the proposed VSD method and the encoding methods employed in previous research studies. Upon examining the graph, it becomes evident that the VSD method offers a substantial improvement in terms of encoding efficiency. The encoding time required by the VSD method is significantly lower when compared to the encoding times reported in previous research efforts. For example,

Goldman's model requires 1601 s/min to encode a 100 MB file, while VSD accomplished this task within 373 s/min.

The observed enhancement in encoding efficiency achieved by the VSD method is significant. It implies that the proposed approach outperforms existing encoding techniques [10–12,16] and streamlines the overall encoding process.

4.6. Ablation Studies

The comparison between the proposed VSD method and existing state-of-the-art methods [10–14,16] reveals notable advantages of our approach (Table 6). It should be noted that the experimental data of each method in the table are based on the video files given in Table 5 (numbered 1–7). In terms of GC content ratio, the VSD method consistently demonstrates a superior balance, ensuring optimal biochemical compatibility. The innovative quadratic coding model, leveraging RS error-correcting codes, stands out for its effectiveness in converting video files into DNA sequences with enhanced error-correction capabilities. This unique approach not only meets biochemical constraints but also addresses the complexities of computational processing. Furthermore, the VSD method introduces an efficient indexing mechanism for random video access, adding a layer of versatility to the storage system. Through extensive simulations and error correction procedures, the VSD method consistently outperforms existing methods, showing its robustness, efficiency, and success in accurately decoding the original video content.

Table 6. Comparative analysis of key factors in DNA data storage, highlighting the effectiveness of the proposed VSD method with state-of-the-art ablation studies.

Author	Church [10]	Goldman [11]	Grass [12]	Blawat [16]	Erlich [13]	Ping [14]	Li [42]	VSD
Year-Refs.	2012	2013	2015	2016	2017	2018	2021	2023
GC content(%)	2.5–100	22.5–82.5	12.5–100	27–69	40–60	40–60	34–66	40–60
Homopolymer	≤3	≤3	≤3	≤3	≤4	≤4	≤4	≤3
Random access	No	No	No	No	No	No	No	Yes
Data size (MB)	0.65	0.63	0.08	22	2.11	0.24	20	177
Error Correction	Repetition	Repetition	RS	Forward EC	Fountain	RS	Repetition	RS
Density *	0.83	0.33	1.16	1.08	1.57	1.75	1.56	1.75

* bits/nucleotides.

5. Conclusions

This study introduces the video storage in DNA (VSD) method, a pioneering solution for efficient and robust storage of large-scale video data. The innovative combination of video segmentation and Reed–Solomon error correction in the VSD method addresses the limitations of existing approaches, showing commendable encoding efficiency and a 30% improvement in time efficiency, as well as better successful decoded rates of base substitutions compared to previous works. The proposed quadratic coding model efficiently balances storage density and bio-constraints (GC ratio ~50% and homopolymer), marking a significant stride in reliable DNA storage. This work opens avenues for reliable video archival, with promising implications for information storage. In future work, adaptive coding for the VSD method and dynamically adjusting strategies based on video content promise enhanced versatility. Optimization for diverse error types will maximize VSD potential, solidifying its role as a premier DNA-based video storage solution. Further analysis, considering coding constraints like RC constraints and data redundancy, aims for a comprehensive understanding of information storage and encoding in varied digital entities.

Author Contributions: J.H.: Data curation, Conceptualization, Writing—original draft, Visualization, Validation, Methodology, Writing—review and editing, Experiments, Software. A.R.: Conceptualization, Methodology, Formal analysis, Writing—review and editing. S.W.: Validation, Supervision. D.Z.: Validation. Q.J.: Resources, Supervision, Project administration, Validation, Funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Key Research and Development Program of China under fund numbers 2021YFF1200100, and 2021YFF1200104, the grants from the Natural Science Foundation of Hebei Province (F2021201055), and the Innovation Capacity Enhancement Program-Science and Technology Platform Project, Hebei Province (22567623H).

Data Availability Statement: The data and codes used in this work are available at https://github.com/jork07/VSD (accessed on 12 March 2024).

Acknowledgments: The authors would like to thank all the anonymous reviewers for their insightful comments and constructive suggestions that have obviously upgraded the quality of this manuscript.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal circumstances that could have appeared to influence the work reported in this manuscript.

References

1. Reinsel, D.; Gantz, J.; Rydning, J. *Data Age 2025: The Evolution of Data to Life-Critical Don't Focus on Big Data*; Focus on the Data That's Big; Seagate: Dublin, Ireland, 2017.
2. Carpenter, K.P.; Siddiqi, A.; Chase, J. *Science & Tech Spotlight: Alternative Data Storage Technologies*; U.S. Government Accountability Office: Washington, DC, USA, 2022.
3. Sun, F.J.; Dong, Y.M.; Ni, M.; Ping, Z.; Sun, Y.H.; Ouyang, Q.; Qian, L. Mobile and Self-Sustained Data Storage in an Extremophile Genomic DNA. *Adv. Sci.* **2023**, *10*, 2198. [CrossRef] [PubMed]
4. Pan, C.; Tabatabaei, S.K.; Tabatabaei Yazdi, S.M.H.; Hernandez, A.G.; Schroeder, C.M.; Milenkovic, O. Rewritable two-dimensional DNA-based data storage with machine learning reconstruction. *Nat. Commun.* **2022**, *13*, 2984. [CrossRef] [PubMed]
5. Ceze, L.; Nivala, J.; Strauss, K. Molecular digital data storage using DNA. *Nat. Rev. Genet.* **2019**, *20*, 456–466. [CrossRef] [PubMed]
6. Bencurova, E.; Akash, A.; Dobson, R.C.J.; Dandekar, T. DNA storage-from natural biology to synthetic biology. *Comput. Struct. Biotechnol. J.* **2023**, *21*, 1227–1235. [CrossRef] [PubMed]
7. Li, X.Y.; Chen, M.X.; Wu, H.M. Multiple errors correction for position -limited DNA sequences with GC balance and no homopolymer for DNA-based data storage. *Brief. Bioinform.* **2023**, *24*, bbac484. [CrossRef] [PubMed]
8. Rasool, A.; Hong, J.; Jiang, Q.; Chen, H.; Qu, Q. BO-DNA: Biologically optimized encoding model for a highly-reliable DNA data storage. *Comput. Biol. Med.* **2023**, *165*, 107404. [CrossRef] [PubMed]
9. Rasool, A.; Jiang, Q.; Qu, Q.; Dai, J. Evolutionary approach to construct robust codes for DNA-based data storage. *Front. Genet.* **2023**, *14*, 1158337. [CrossRef]
10. Church, G.M.; Gao, Y.; Kosuri, S. Next-generation digital information storage in DNA. *Science* **2012**, *337*, 1628. [CrossRef] [PubMed]
11. Goldman, N.; Bertone, P.; Chen, S.; Dessimoz, C.; LeProust, E.M.; Sipos, B.; Birney, E. Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. *Nature* **2013**, *494*, 77–80. [CrossRef]
12. Grass, R.N.; Heckel, R.; Puddu, M.; Paunescu, D.; Stark, W.J. Robust chemical preservation of digital information on DNA in silica with error-correcting codes. *Angew. Chem. Int. Ed.* **2015**, *54*, 2552–2555. [CrossRef]
13. Erlich, Y.; Zielinski, D. DNA Fountain enables a robust and efficient storage architecture. *Science* **2017**, *355*, 950–954. [CrossRef] [PubMed]
14. Ping, Z.; Chen, S.; Zhou, G.; Huang, X.; Zhu, S.J.; Zhang, H.; Lee, H.H.; Lan, Z.; Cui, J.; Chen, T. Towards practical and robust DNA-based data archiving using the yin–yang codec system. *Nat. Comput. Sci.* **2022**, *2*, 234–242. [CrossRef] [PubMed]
15. Bornholt, J.; Lopez, R.; Carmean, D.M.; Ceze, L.; Seelig, G.; Strauss, K. A DNA-based archival storage system. In Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems, Atlanta, GA, USA, 2–6 April 2016; pp. 637–649.
16. Blawat, M.; Gaedke, K.; Huetter, I.; Chen, X.-M.; Turczyk, B.; Inverso, S.; Pruitt, B.W.; Church, G.M. Forward error correction for DNA data storage. *Procedia Comput. Sci.* **2016**, *80*, 1011–1022. [CrossRef]
17. Meiser, L.C.; Antkowiak, P.L.; Koch, J.; Chen, W.D.; Kohll, A.X.; Stark, W.J.; Heckel, R.; Grass, R.N. Reading and writing digital data in DNA. *Nat. Protoc.* **2020**, *15*, 86–101. [CrossRef] [PubMed]
18. Jeong, J.; Park, S.-J.; Kim, J.-W.; No, J.-S.; Jeon, H.H.; Lee, J.W.; No, A.; Kim, S.; Park, H. Cooperative sequence clustering and decoding for DNA storage system with fountain codes. *Bioinformatics* **2021**, *37*, 3136–3143. [CrossRef] [PubMed]
19. Press, W.H.; Hawkins, J.A.; Jones Jr, S.K.; Schaub, J.M.; Finkelstein, I.J. HEDGES error-correcting code for DNA storage corrects indels and allows sequence constraints. *Proc. Natl. Acad. Sci. USA* **2020**, *117*, 18489–18496. [CrossRef] [PubMed]
20. Takahashi, C.N.; Nguyen, B.H.; Strauss, K.; Ceze, L. Demonstration of end-to-end automation of DNA data storage. *Sci. Rep.* **2019**, *9*, 4998. [CrossRef]
21. Deng, L.; Wang, Y.; Noor-A-Rahim, M.; Guan, Y.L.; Shi, Z.; Gunawan, E.; Poh, C.L. Optimized code design for constrained DNA data storage with asymmetric errors. *IEEE Access* **2019**, *7*, 84107–84121. [CrossRef]
22. Chen, W.; Han, M.; Zhou, J.; Ge, Q.; Wang, P.; Zhang, X.; Zhu, S.; Song, L.; Yuan, Y. An artificial chromosome for data storage. *Natl. Sci. Rev.* **2021**, *8*, nwab028. [CrossRef]

23. Rasool, A.; Qu, Q.; Wang, Y.; Jiang, Q.S. Bio-Constrained Codes with Neural Network for Density-Based DNA Data Storage. *Mathematics* **2022**, *10*, 845. [CrossRef]
24. Chen, W.; Huang, G.; Li, B.; Yin, Y.; Yuan, Y. DNA information storage for audio and video files. *Sci. Sin. Vitae* **2020**, *50*, 81–85. [CrossRef]
25. Nam, S.-H.; Ahn, W.; Kwon, M.-J.; Yu, I.-J. Detection of Double Compression in MPEG-4 Videos Using Refined Features-based CNN. *arXiv* **2021**, arXiv:2107.08939v1.
26. Nam, S.-H.; Ahn, W.; Kwon, M.-J.; Kang, J.; Yu, I.-J. DHNet: Double MPEG-4 Compression Detection via Multiple DCT Histograms. *IEEE MultiMedia* **2022**, *29*, 11–22. [CrossRef]
27. Koch, J.; Gantenbein, S.; Masania, K.; Stark, W.J.; Erlich, Y.; Grass, R.N. A DNA-of-things storage architecture to create materials with embedded memory. *Nat. Biotechnol.* **2020**, *38*, 39–43. [CrossRef] [PubMed]
28. Gllavata, J.; Ewerth, R.; Freisleben, B. Tracking text in MPEG videos. In Proceedings of the 12th Annual ACM International Conference on Multimedia, New York, NY, USA, 10–15 October 2004; pp. 240–243.
29. Park, S.-h.; Jeong, J.; Kwon, T. Contents distribution system based on MPEG-4 ISMACryp in IP set-top box environments. *IEEE Trans. Consum. Electron.* **2006**, *52*, 660–668. [CrossRef]
30. Wamser, F.; Iffländer, L.; Zinner, T.; Tran-Gia, P. Implementing application-aware resource allocation on a home gateway for the example of YouTube. In Proceedings of the Mobile Networks and Management: 6th International Conference, Würzburg, Germany, 22–26 September 2014; pp. 301–312.
31. Jirón, I.; Soto, S.; Marín, S.; Acosta, M.; Soto, I. A new DNA-based model for finite field arithmetic. *Heliyon* **2019**, *5*, 2901. [CrossRef] [PubMed]
32. Costello, D.J.; Lin, S. *Error Control Coding: Fundamentals and Applications*; Prentice Hall: Hoboken, NJ, USA, 1982.
33. Jeng, J.-H.; Truong, T.-K. On decoding of both errors and erasures of a Reed-Solomon code using an inverse-free Berlekamp-Massey algorithm. *IEEE Trans. Commun.* **1999**, *47*, 1488–1494. [CrossRef]
34. Lee, H. An area-efficient Euclidean algorithm block for Reed-Solomon decoder. In Proceedings of the IEEE Computer Society Annual Symposium on VLSI, Tampa, FL, USA, 20–21 February 2003; pp. 209–210.
35. Welzel, M.; Schwarz, P.M.; Löchel, H.F.; Kabdullayeva, T.; Clemens, S.; Becker, A.; Freisleben, B.; Heider, D. DNA-Aeon provides flexible arithmetic coding for constraint adherence and error correction in DNA storage. *Nat. Commun.* **2023**, *14*, 628. [CrossRef] [PubMed]
36. Song, L.; Geng, F.; Gong, Z.-Y.; Chen, X.; Tang, J.; Gong, C.; Zhou, L.; Xia, R.; Han, M.-Z.; Xu, J.-Y.; et al. Robust data storage in DNA by de Bruijn graph-based de novo strand assembly. *Nat. Commun.* **2022**, *13*, 5361. [CrossRef] [PubMed]
37. Cao, B.; Shi, P.J.; Zheng, Y.F.; Zhang, Q. FMG: An observable DNA storage coding method based on frequency matrix game graphs. *Comput. Biol. Med.* **2022**, *151*, 269. [CrossRef]
38. Rasool, A.; Qu, Q.; Jiang, Q.S. A Strategy-based Optimization Algorithm to Design Codes for DNA Data Storage System. In *International Conference on Algorithms and Architectures for Parallel Processing*; Springer International Publishing: Cham, Switzerland, 2021; pp. 284–299.
39. King, O.D. Bounds for DNA codes with constant GC-content. *Electron. J. Comb.* **2003**, *10*, 1077. [CrossRef]
40. Tomar, S. Converting video formats with FFmpeg. *Linux J.* **2006**, *2006*, 10.
41. Tong, J.; Han, G.; Sun, Y. An Improved Marker Code Scheme Based on Nucleotide Bases for DNA Data Storage. *Appl. Sci.* **2023**, *13*, 3632. [CrossRef]
42. Li, B.; Ou, L.; Du, D. DP-DNA: A Digital Pattern-Aware DNA Storage System to Improve Encoding Density. *arXiv* **2021**, arXiv:2108.04123.