Signal Processing: Image Communication 97 (2021) 116331

Contents lists available at ScienceDirect

Signal Processing: Image Communication

journal homepage: www.elsevier.com/locate/image

Image storage onto synthetic DNA

Melpomeni Dimopoulou [a,*], Marc Antonini [a], Pascal Barbry [b], Raja Appuswamy [c]

[a] Université Côte d'Azur, I3S, CNRS, UMR 7271 Sophia Antipolis, France
[b] Université Côte d'Azur, IPMC, CNRS, UMR 7275, Sophia Antipolis, France
[c] EURECOM, Sophia Antipolis, France

ARTICLE INFO

Keywords:
DNA coding
Image coding
Robust encoding
Long-term storage

ABSTRACT

Living in the age of data explosion, the research of solutions for efficient long term storage of the infrequently used "cold" data is becoming of great interest. However, even if existing storage systems suggest efficiency in capacity, they are lacking in durability. Hard disks, flash, tape or even optical storage have limited lifespan in the range of 5 to 20 years. Interestingly, recent studies have proven that due to its biological properties, the DNA is a strong candidate for the storage of digital information allowing also data longevity. The DNA's biological properties allows the storage of a great amount of information into an extraordinary small volume while also promising efficient storage for centuries or even longer with no loss of information. However, the biological procedures of DNA synthesis and sequencing are expensive while also introducing important restrictions in the encoding process. More precisely the encoding of digital data onto DNA is not obvious, because when decoding, we have to face the problem of sequencing noise robustness. This work proposes a coding solution for the storage of digital images onto synthetic DNA. We developed a new encoding algorithm which generates a DNA code robust to biological errors coming from the synthesis and the sequencing processes. Furthermore, we compare this new algorithm to the state of the art encoding techniques analyzing the advantages of using the proposed method.

1. Introduction

Digital evolution has caused an immersive increase in the amount of data that is being generated and stored. The digital universe is forecast to grow to over 160 zetabytes in 2025. At the same time studies show that after storage, 80% or more of this data might not be needed for months, years, decades, or maybe ever. Old photographs stored by users on Facebook is one such example of cold data; Facebook recently built an entire data center dedicated to storing such cold photographs. Unfortunately, all current storage media used for cold data storage (Hard Disk Drives or tape) suffer from two fundamental problems. First, the rate of improvement in storage density is at best 20% per year, which substantially lags behind the 60% rate of cold data growth. Second, current storage media have a limited lifetime of five (HDD) to twenty years (tape). As data is often stored for much longer duration (50 or more years) due to legal and regulatory compliance reasons, data must be migrated to new storage devices every few years, thus, increasing the price of data ownership. Several projects, for instance at the University of Southampton [1] or at Hitachi [2], are currently considering new forms of very long term digital storage, using molding silica glass, which estimated storage length time in the range of the 100 million year. However, these projects are currently stymied by an important problem related to space: both developed at most a storage capacity that does not exceed 40 MBytes per inch, i.e. a very low value compared to the one Terabyte per square inch capacity reached by any standard hard disk.

An alternative approach may stem from the use of DNA (deoxyribonucleic acid), the support of heredity in living organisms. DNA possesses three key properties that make it relevant for archival storage. First, it is an extremely dense three-dimensional storage medium that has the theoretical ability to store 455 Exabytes in 1 gram; in contrast, a 3.5″ HDD can store 10 TB and weighs 600 grams today. Second, DNA can last several centuries even in harsh storage environments; HDD and tape have life times of five and twenty years. Third, it is very easy, quick, and cheap to perform in-vitro replication of DNA; tape and HDD have bandwidth limitations that result in hours or days for copying large Exabyte-sized archives. DNA is a complex molecule corresponding to a succession of four types of nucleotides (nts), Adenine (A), Thymine (T), Guanine (G), Cytosine (C). It is this quaternary genetic code that inspired the idea of DNA data storage which suggests that any binary information can be encoded into a DNA sequence of A, T, C, G. The main challenge lies in the restrictions imposed by the biological procedures of DNA synthesis (writing) and sequencing (reading) which are involved in the encoding process and introduce significant errors in the encoded sequence while also being relatively costly (several dollars for writing and reading a small strand of nucleotides).

* Corresponding author.
E-mail address: dimopoulou@i3s.unice.fr (M. Dimopoulou).

https://doi.org/10.1016/j.image.2021.116331
Received 15 October 2019; Received in revised form 25 March 2021; Accepted 15 May 2021
Available online 27 May 2021

In this paper we make a very first step in introducing image compression techniques for long term image storage onto synthetic DNA. One of our main goals is to allow the reduction of the cost of DNA synthesis which nowadays can be very high for storage purposes. Unlike previous works that have been transcoding directly binary sequences onto DNA, our coding algorithm is applied on the quantized wavelet coefficients of an image. To this end, the proposed solution is optimized thanks to a source allocation process, which from now on will be called nucleotide allocation, across the different wavelet subbands by taking into account the input data characteristics. The desired compression rate can then be chosen, allowing to control the number of nucleotides to generate and thus, the DNA synthesis cost. Furthermore, we have developed a new encoding algorithm which generates a DNA code robust to biological errors coming from the synthesis and the sequencing processes.

In Section 2, we are presenting some existing pioneer works on the topic of DNA data storage which is relatively new to the field of digital data storage. The difficulties and constraints are discussed in Section 3 while in Section 4 we present the general procedure and the main parts of the DNA coding of digital data. In Section 5, we describe analytically the algorithm proposed in this work, including the construction of the code as well as the mapping of the source values to DNA codewords. Then, in Section 5.4, we compare our proposed encoder to some state of the art works and evaluate the performance of our algorithm in Section 6. Finally, in Section 7, we analyze all the processes needed to create the DNA strands and pass from the digital world to the biological form of DNA providing information about the wet lab experiment that has been carried out during this study.

2. Existing works

Recent works tackle the problem of digital data storage onto DNA still leaving room for further improvements that could finally bring the idea of DNA data storage into practice. In [3] there has been a first attempt to store data into DNA while also providing a study of the main causes of biological error. In order to deal with errors, previous works in [4] and [5] have suggested dividing the original file into overlapping segments so that each input bit is represented by multiple DNA sequences (oligos). However, this procedure introduces extra redundancy and is poorly scalable. Other studies [6,7] suggest the use of Reed–Solomon code in order to treat the erroneous sequences while in [8] a new robust method of encoding has been proposed to approach the Shannon capacity. Finally, latest works in [9] have introduced a clustering algorithm to provide a system of random-access DNA data storage. Nevertheless, all these approaches mainly try to convert a binary bit stream onto a DNA sequence without considering the original input data characteristics. In addition to this, as the DNA synthesis cost can be really high it is extremely important to take full advantage of the optimal compression that can be achieved before synthesizing the sequence into DNA. Although previous works have used compressed data such as images in a JPEG format, the final encoding has been carried out on the compressed bit stream without interfering to the compression procedure.

In this work, parts of which have been also presented in [10] and [11], we present a new efficient encoding algorithm which is highly robust to the sequencing error and deals with the ill-case sequences containing pattern repetitions. The construction of the code is strongly linked to some biological constraints presented in Section 3. This code is embedded in a coding workflow for the storage of digital images onto synthetic DNA. The proposed solution optimizes the trade-off rate–distortion thanks to a nucleotide allocation algorithm and is able to compress an input image without affecting strongly its visual quality. Even though the compression scheme presented in this work uses very simple compression techniques and might not show the highest compression efficiency, with this paper we prove that controlling the end-to-end process allows to minimize the DNA synthesis cost.

3. A constrained problem

DNA data coding is the process of encoding a stream of data into a quaternary sequence using the alphabet $\{A, T, C, G\}$ to be synthesized into DNA and safely stored for many years without loss of information. The stored DNA can be retrieved using the biological process of DNA sequencing and then decoded to reconstruct the stored digital data. It is consequently a multidisciplinary subject which involves the numerical processing of encoding and decoding of digital data and the biological procedures of DNA synthesis (writing) and sequencing (reading). Those two last mechanisms are very error-prone imposing fundamental restrictions to the encoding workflow. DNA synthesis is an error-free procedure as long as the DNA strands to be synthesized do not overpass the length of 150–200 nts. For longer sequences the synthesis error increases exponentially. Consequently to eliminate this error to zero the DNA sequences to be synthesized need to be cut into short pieces and formatted in such a way that the initial sequence can be reconstructed in the decoding part. Detailed explanation for the formatting of the DNA sequence will be given in Section 7.1. On the contrary, the biological procedure of DNA sequencing introduces much error which cannot be neglected and therefore there is a need for dealing with the erroneous oligos produced by the sequencer. Studies have shown that the two main factors causing errors in the sequenced oligos are the following:

- **Homopolymers:** Consecutive occurrences of the same nucleotides should be avoided.
- **G, C content:** The percentage of G and C in the oligos should be lower or equal to the one of A and T.
- **Pattern repetitions:** The codewords used to encode the oligos should not be repeated forming the same pattern throughout the oligo length.

Taking into account all the above rules the sequencing error can be reduced. Consequently, in this work we propose in Section 5.1 a novel efficient encoding algorithm which encodes the quantized wavelet coefficients using codewords that respect those biological constraints.

4. Overview of the general coding process

As shown in Fig. B.1, our proposed workflow consists of 4 main parts. The first part is image compression where the data has to be compressed using a DWT and quantizing each subband of the wavelet decomposition independently. In order to optimize the compression we use an optimal nucleotide allocation algorithm, which works similar to the bit allocation algorithms described in [12–14], designed to serve the restrictions of the DNA coding process as described in Section 3. This allocation serves to the choice of an optimal quantization step for each wavelet subband for storing the maximum possible bits in one nucleotide for a given encoding rate. Consequently, the number of oligos to be synthesized is minimized for a specified image quality. In the second part of the workflow the quantized subbands are independently encoded into a quaternary code of A, T, C and G using a novel encoding algorithm robust to sequencing error (see Section 5.1). The encoded sequences are then cut into chunks and formatted adding headers to produce the final oligos. Those oligos are then processed biologically in vitro so that in the third part of the process they are being synthesized into DNA and stored. After amplification (PCR process) and sequencing, a set of oligos is retrieved. Finally, the last part of the scheme corresponds to the decoding and reconstruction of the initial image from the sequenced oligos. All those sub-parts of our encoding workflow are described in detail in the following sections.

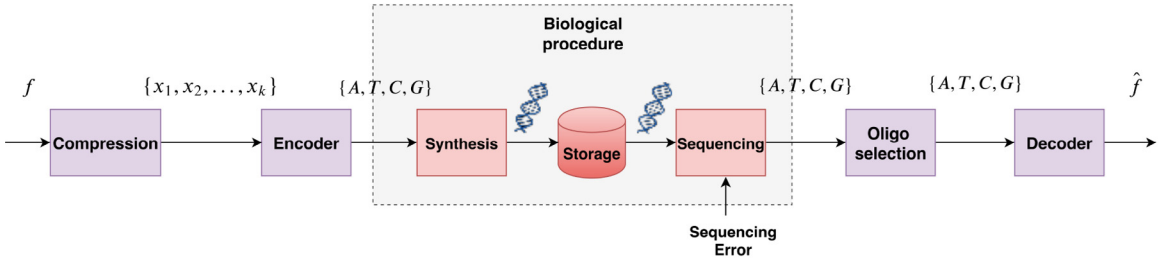

Fig. B.1. The general encoding workflow.

5. Our proposed encoding algorithm

5.1. Construction of the code

The purpose of the DNA data coding is the encoding of some input data using a quaternary code composed by the alphabet $\{A, T, C, G\}$ to be later synthesized into DNA. Let $\Sigma = \{s_1, s_2, \ldots, s_k\}$, with $|\Sigma| = k$, be a set of symbols to be encoded into a set $C^* = \{c_1, c_2, \ldots, c_L\}$ of L quaternary codewords of length $l \in \mathbb{N}\setminus\{0, 1\}$. The goal of this encoding algorithm is to generate the code Γ where $\Gamma : \Sigma \to C^*$. We denote $\Gamma(s_i) = c_i$ the codeword associated with the symbol $s_i \in \Sigma$.

The algorithm for the construction of the code C^* is guided by the restrictions imposed by the biological procedures included in the process of DNA data coding. The main idea is the creation of codewords from a set of duplets (pairs of symbols) which create an acceptable sequence when assembled in a longer strand. This means that this assembly creates no homopolymer runs and contains a percentage of G and C which is lower or equal to the percentage of A and T. More precisely, the codewords are constructed by selecting elements from the following dictionaries:

- $C_1 = \{AT, AC, AG, TA, TC, TG, CA, CT, GA, GT\}$
- $C_2 = \{A, T, C, G\}$

Dictionary C_1 is composed only by pairs of symbols that when assembled in a longer strand will respect some biological restrictions. To ensure that the code does not create homopolymers, the dictionary C_1 does not contain pairs of the same symbol. This means that the pairs AA, TT, CC, and GG are omitted. Furthermore, to keep the C and G percentage lower or equal to the one of A and T the pairs GC and CG are also not included. To verify this last claim we have computed the evolution of the A, T and G, C percentages created using our code for a source of symbols that follows a Gaussian distribution, in function of the dynamic of the source. The result is illustrated in Fig. B.2. Consequently until this point it is clear that our encoding process respects the first two restrictions described in 3.

Then, codewords of an even length l are constructed by selecting $\frac{l}{2}$ pairs from dictionary C_1. Codewords of an odd length are constructed by selecting $\frac{l-1}{2}$ pairs from C_1 also adding a symbol from C_2 at the end of the codeword. More precisely, for the construction of a codeword of length l the first $\lfloor \frac{l-1}{2} \rfloor$ nucleotide pairs will be filled by choosing symbol pairs from C_1. There are $10^{\lfloor \frac{l-1}{2} \rfloor}$ different choices for filling each those first nucleotide pairs. Then, to fill in the last nucleotide in the case of and odd length codeword, or equivalently the last pair of nucleotides in the case of an even length codeword, one should choose a symbol from C_2 or a last pair of symbols from C_1 respectively. At this point it is necessary to mention that if the number k of different symbols s_k to be encoded is lower or equal to 4 we avoid an encoding using only symbols from dictionary C_2. This is due to the fact that in the case where the same symbol s_k is repeated many times consecutively in the input sequence, the encoding can create either homopolymer runs or pattern repetitions. Consequently, even though it is feasible, an encoding rate $R = 1$ nucleotide/symbol is avoided in order to ensure robustness of the code. As a result the codebook size L is given by the following relation.

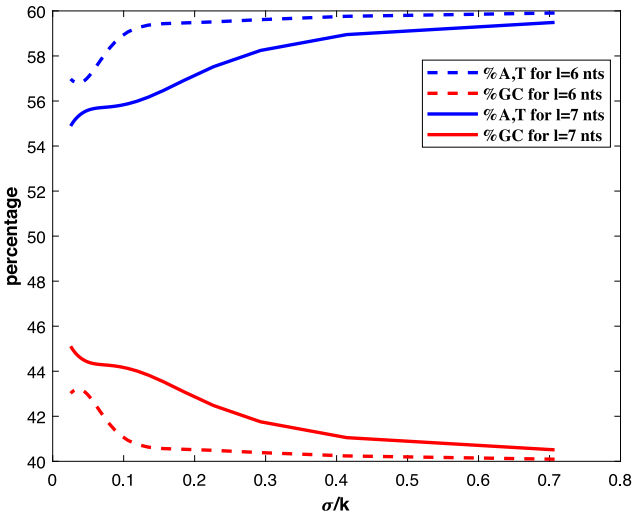

Fig. B.2. Evolution of the percentages of A, T (blue curves) and G, C (red curves) for a centered Gaussian source of variance σ^2 and mean $\mu = \frac{k}{2}$ (k denoting the number of different source symbols) in function of the source dynamic normalized by k. The full line and dashed curves represent two different cases of codeword length l of 6 and 7 nucleotides respectively.

$$L = \begin{cases} 10^{\frac{l}{2}}, & \text{if } l \text{ is even} \\ 10^{\frac{l-1}{2}} * 4, & \text{if } l \text{ is odd} \end{cases}$$

It is obvious that the codebook length increases at an order of $O(n)$. More specifically, in the worst case where one needs to add an extra pair of symbols to the codeword length to cover the needed size k of symbols to be encoded into quaternary, the codebook size L will be multiplied by 10. This codebook extension can be relatively big compared to the encoding needs and can possibly leave a big part of codewords unused. However, this unused part can be exploited to deal with the last biological restriction of pattern repetitions which can occur in the case where the same symbol is repeated many times in the initial sequence. The idea is to replicate m times k into L so that each symbol in Σ is represented by more than one codewords in C^* in such a way so to make use of the full codebook length. Using this method, which we will call *replication step*, in every repetition of the same symbol a different codeword will be used for the representation, not creating long pattern repetitions in the final encoded sequence. The replication step of the mapping algorithm can cause an increase in the encoding cost but will provide an encoded sequence which will be more robust to the biological error. Thus, depending on the needs and the purposes of the encoding, the user can select whether the replication stage is necessary or not. There are three different mapping algorithms each one using a different replication method. Further explanation about this encoding step will be given in the following section.

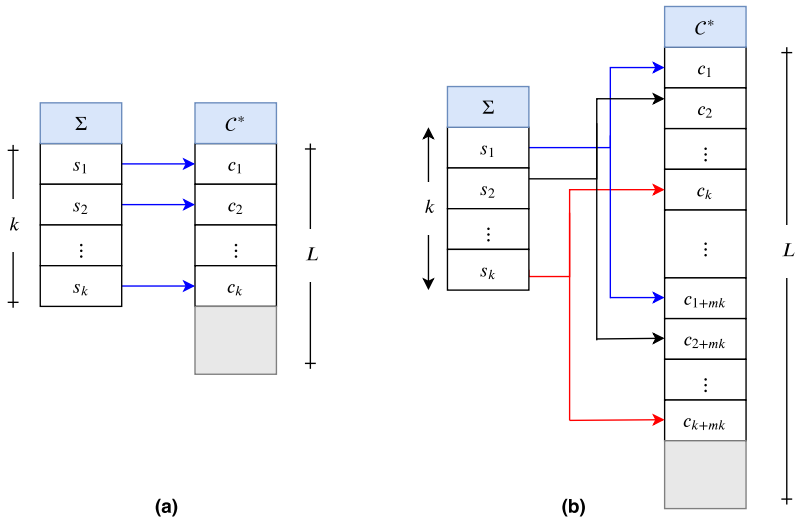

Fig. B.3. Mapping of quantized values into quaternary code. The gray area corresponds to not used codewords, if they exist.

5.2. Mapping

The mapping step of the encoding algorithm is based on a pseudorandom association of a symbol in Σ to one or more possible codewords. As explained in the previous section the mapping algorithm is different according to the needs of the encoding. In the case where one wishes to have a more robust encoding they can use a replication step in which the algorithm ensures that the set of symbols Σ can fit more than one times into the codebook C^*. The replication step increases the robustness of the encoded sequence by inserting more randomness which can also be used to give some information about possible errors that may occur in the sequencing part. However, a drawback of the replication step is the increase in the encoding cost. It is obvious that there is a trade-off between the encoding robustness and the final encoding cost so it is up to the user to select if this replication step is useful. In the case where the replication phase of the mapping is not used, the mapping algorithm is trivial and described in Fig. B.3a.

In the case where the replication step should be used, the mapping algorithm works as illustrated in Fig. B.3b. In this case the code Γ is constructed so that each symbol in Σ is mapped to a set of different m non-empty quaternary codewords in C^* following a one-to-many relation in such a way that it is uniquely decodable. Since we ensure $L \geq 2 * k$, the pseudorandom mapping can at least provide two possible codewords for one input symbol. The value of m is given by computing the number of times that the dictionary Σ fits into the code C^* and is given by $m = \lfloor \frac{L}{k} \rfloor$. More precisely, the mapping is described by the following steps:

1. Build the corresponding code C^* of size L using all possible codewords of length l which can be built following the rules described in the previous paragraph,
2. Compute the number of times m that k can be replicated into the total size L of the code C^*:

   $$m = \lfloor \frac{L}{k} \rfloor$$

3. The mapping of the quantized value s_i to a codeword c_i is given by:

   $$\Gamma(s_i) = C^*(i + rand(0, m-1) * k)$$

The encoding procedure is explained by Algorithm 1 (see Annex). It is obvious to prove that the code produced by this algorithm is uniquely decodable.

5.3. Discussion

Selection of one of the m codewords mapped at the same symbol. The number of times m that the dictionary Σ can fit into the code C^* can be selected either using a random generator, as described in the algorithm above, or using a deterministic function which will be known to the decoder. This option can potentially be used for correcting errors that might occur during the sequencing of the data. More precisely when encoding each symbol in the input data the encoder can select using a known function one of the m codewords representing this same symbol. We can this way assume that there are m different codeword classes. Consequently if an error occurs in the decoder turning a codeword belonging to one class into a codeword that belongs to another class, one will be able to detect that an error has occurred in this position correcting it to the closest codeword belonging in the correct class.

Reducing the codebook length. In Section 5.2, we have proposed the construction of a codebook C^* of length L requiring $L \geq k$. On the one hand this requirement ensures the construction of a robust code which will avoid the creation of patterns but on the other hand it creates a long codebook reducing the efficiency of the encoding algorithm in terms of encoding rate. Consequently there is a trade-off between the code robustness to errors and the coding potential. When increasing the codeword length according to the needs, our proposed algorithm multiplies the length of the code by a factor of at least 4. In the binary encoding, the addition of an extra bit in the codewords increases the code length by 2. Consequently, it is clear that when our encoder is used to transcode binary files, the length of the generated codebook is increasing faster than the number of symbols to encode. As an example we take the case of encoding digital bytes (8 bits) into DNA codewords using our proposed encoder. While there are $k = 256$ different values, one would need at least a code of $L = 400$ to encode the values into nucleotides. This applies to the case where one does not constrain the codeword length to allow double representation of each symbol into the created code C^*. However, even without this restriction, there are still 144 codewords that corresponds to 36% of the code's length that will be unused.

However, in the case where the source symbols do not follow a uniform distribution (DWT subband coefficients for example), one could map the most 144 most frequent symbols to those unused codewords and therefore balance in a way the robustness/potential trade-off. At this point we recall the fact that our encoding system is not constructed to only encode binary files but could be applied in any type of source data. Nevertheless this is just a simple example to demonstrate the flexibility of this algorithm to optimization according to the encoding needs.

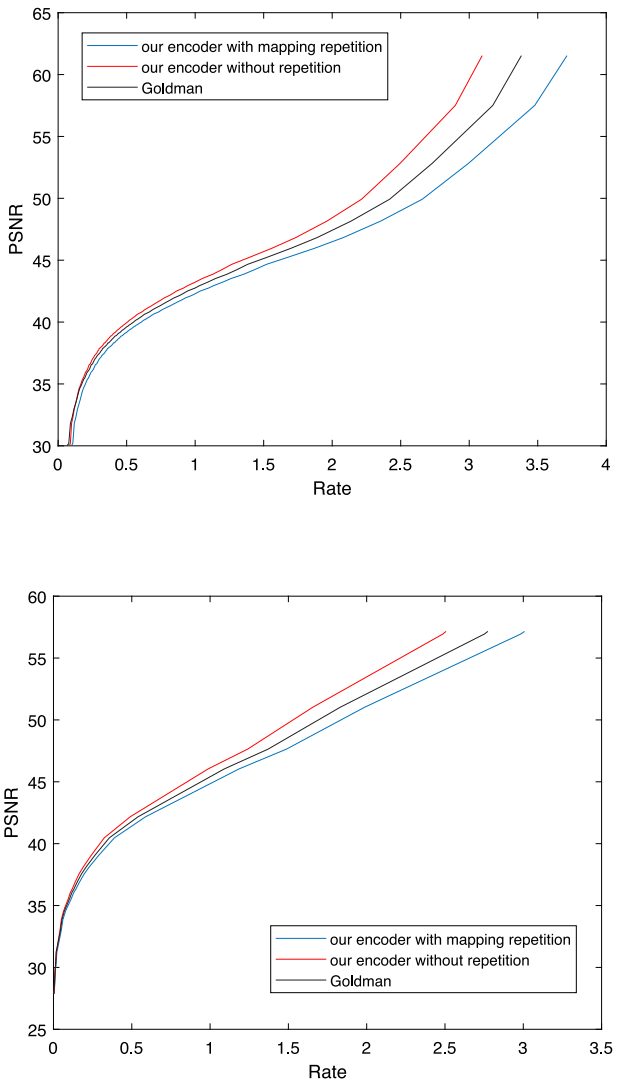

Fig. B.4. Comparison of the encoder proposed by Goldman et al. in [5] and our encoder. For the encoding we consider each byte (8 bits) as a symbol to be encoded and the figures show the evolution of the PSNR (dB) in function of the rate (nts/pixel) for different compression qualities using the JPEG codec (results on the left figure) and the JPEG2000 (on the right figure).

5.4. Comparison to the state of the art

DNA data storage is a new field of research which is expected to make a breakthrough in the domain of "cold" digital data archiving. As briefly described in Section 2, some existing pioneering works suggest different algorithms for encoding the digital information into a quaternary sequence of A, T, C, G. In this section we describe the advantages of the encoding algorithm proposed in this work in comparison to existing encoding methods. The first attempt of encoding digital data into DNA is described in [3] by the works of Church et al. In this work each binary bit is encoded to one nucleotide giving a total coding potential of 1 bit/nucleotide. To improve the coding potential as well as the robustness of the encoding to errors following works have adopted some more complicated encoding algorithms. More precisely Goldman et al. in [5], have proposed an algorithm that respects the constraint from Section 3 of avoiding homopolymer runs to improve the quality of sequencing. This encoding applies a ternary Huffman algorithm to compress the binary sequence into a ternary stream of three symbols (trits). Then each of the trits is encoded into a symbol from the dictionary $\{A, T, C, G\}$ each time avoiding the symbol that has been previously used. However, this encoding method does not allow controlling the balance of the G, G percentage in the produced strands. Furthermore, unlike our proposed algorithm, in the encoding of a quantized or a sparse signal (as for instance the wavelet coefficients of a DWT transform where a same quantized value can be consecutively repeated many times), this encoding algorithm can create pattern repetitions which is an ill-case leading to a higher error probability at the phase of sequencing [15]. A later study in [16], introduces the use of addressing fields to allow random access in the reading and writing of the DNA oligos. As the addressing primers contain fundamental information which should be correctly retrieved the authors propose a novel encoding for DNA data storage which is built such that secondary structure is avoided in the encoded DNA strands. More precisely the DNA code differs for each oligo and is constructed according to the oligo's address field. The code is constructed ensuring that there is no strong correlation between the encoding codewords and the addressing header which could lead to the oligo binding on itself and therefore leading to important loss during sequencing. According to a later publication [17], this encoding can reach a coding potential of 1.57 bits/nt which is slightly lower than the potential of our proposed solution. While this encoding avoids undesirable cross-hybridization problems during the process of oligo selection and amplification and can allow some limited error correction one possible drawback is the fact that the code is varying according to the addressing primer and it is not fixed throughout the encoding process. Thus, in contrast to our proposed algorithm, it cannot be embedded to any encoding workflow. Another interesting work has been proposed by Blawat et al. [7]. This encoding proposes the use of 5 nucleotides to encode 8 bits of information using a method for avoiding homopolymers. Furthermore the encoding inserts some randomization in the selection of the codewords which can be exploited for avoiding pattern repetitions as well as for correcting some types of errors that may occur. The coding potential of this method is 5 nucleotides per 8 bits of binary sequence which is equivalent to 1.6 bits/nt. To encode 8 bits (255 different symbols), our proposed algorithm also needs 5 nucleotides. Nevertheless, a strong advantage of our algorithm is the fact that it can be extended to the encoding of more than 8 bits of information and it can be applied to any type of input data (binary or not). In the works of Grass et al. [6], the encoding is performed using Reed–Solomon codes. This encoding achieves a coding potential of 1.187 bits/nt introducing redundancy in order to introduce error correction but similarly to [7] it is being applicable only in a binary stream. Bornholt et al. in [4] have applied the same encoding as in [5] improving the encoding scheme and avoiding the fourfold redundancy which is suggested by the latter and synthesizes each DNA chunk in 4 shifted copies of the initial sequence. For further information about the fourfold redundancy the reader can refer to [5]. Finally, Erlich et al. [8] have implemented an encoding using Fountain codes to reach a high coding potential. Similarly to most of the previously mentioned works, despite the efficiency in terms of information density, this type of encoding is only applicable to binary information while also being very expensive in computational cost. To evaluate the efficiency of our encoding algorithm we have compared it to the one proposed by [5]. The choice of this work for the comparison is for two main reasons. Firstly, the work proposed by [5] is one of the most popular ones and is to our knowledge the most widely used until this day. Secondly, similarly to our encoding, the algorithm proposed by this work can be applied to any type of symbols and is not limited to the encoding of binary data. For the comparison we have used the JPEG and JPEG2000 codecs to compress an image of 512 × 512 pix to different compression rates and have built the curves of coding potential in nts/pixel in function of the PSNR. The results of this comparison are illustrated in Fig. B.4. More precisely, in our results we compare the encoder of [5] to the one proposed in this work for two different cases of mapping. The first case is the mapping of one symbol to at least two different codewords as proposed in Section 5.2, which we will call mapping with repetition, and a second case where

Table 1
Comparison to previous works — Coding potential: maximal information content of each nucleotide before indexing or error correcting. Redundancy: excess of synthesized oligos to provide robustness to dropouts. Error correction/detection: the presence of error-correction code to handle synthesis and sequencing errors. Full recovery: DNA code was recovered without any error. Net information density: input information in bits divided by the number of synthesized DNA nucleotides (excluding primers).

Parameter	Church et al. [3]	Goldman et al. [5]	Grass et al. [6]	Bornholt et al. [4]	Blawat et al. [7]	Erlich et al. [8]	Our work (with compression)	Our work (raw data)
Input data (Mbytes)	0.65	0.75	0.08	0.15	22	2.15	0.26	0.26
Coding potential (bits/nt)	1	1.58	1.78	1.58	1.6	1.98	2.14	1.6
Redundancy	1	4	1	1.5	1.13	1.07	1	1
Error correction	No	Yes	Yes	No	Yes	Yes	No	No
Full recovery	No	No	Yes	No	Yes	Yes	Yes	Yes
Net information density (bits/nt)	0.83	0.33	1.14	0.88	0.92	1.57	1.71	1.31
Number of oligos	54,898	153,335	4991	151,000	1,000,000	72,000	13,426	17,712

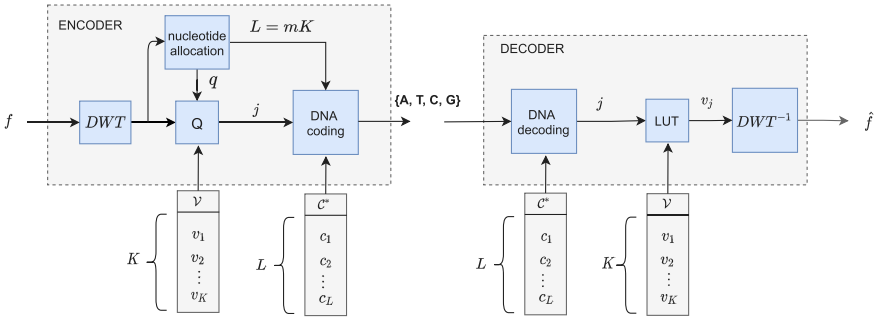

Fig. B.5. Image compression encoder and decoder used in our experiments. Encoder: The input image f is decomposed into different subbands using a DWT. The produced subbands are quantized using an optimal quantization step-size q selected by a nucleotide allocation algorithm to produce a sequence of quantization indices j. The indices are then encoded into DNA using our proposed algorithm. Decoder: The DNA sequence is decoded to the corresponding quantization indices j. The indices are used to retrieve the corresponding quantized coefficients using a Look Up Table (LUT). Finally, the quantized image is reconstructed using the inverse DWT.

one can make use of the codewords that are left unused to be mapped to the most frequent symbols as proposed in the discussion in 5.3. Those results reveal the fact that our proposed encoder's efficiency in terms of coding potential in the encoded stream of nucleotides is comparable to the encoder proposed by Goldman et al. and even slightly better. It is also very interesting to point out again that in the case of mapping repetition our encoder might perform slightly worse than the Goldman encoder but it ensures that the encoded sequence of nucleotides does not contain pattern repetitions which endanger the reliability of the sequencing and can therefore produce sequencing errors. Furthermore, in contrast to the encoder proposed in [5], our algorithm does not make use of the Huffman code. To the contrary, it is a simple fixed length encoder allowing easier error correction in case of an insertion or deletion error. Hence, given also the fact that the proposed algorithm is flexible to modifications according to the encoding needs, those results are very encouraging while proposing a highly robust code.

6. Evaluation of the proposed image codec

6.1. Description of the simulations

This work presents a very first attempt of minimizing the DNA synthesis cost by controlling the part of compression to provide an optimal performance. In this first step, we implemented a basic end-to-end compression scheme which uses classical compression methods and while it may not provide the best compression efficiency it is enough to prove the fact that the DNA synthesis cost can be controlled optimally. Other ways of compressing the image such as non-uniform quantization could significantly improve the compression performance and can therefore be used in our proposed encoder. The structure of the global encoding scheme is presented in Fig. B.5.

6.2. Efficiency of the codec

Our experiments use a 3-level 9/7 DWT decomposition for the image compression, quantizing each subband with a uniform scalar quantizer of quantization step size that is being determined by a source allocation algorithm. The source allocation, called nucleotide allocation in Fig. B.5, consists in minimizing the total quantization mean squared error (mse) under a constraint on the maximum number of nucleotides used for the encoding. It determines the value of the set of optimal quantization steps to be used by the quantizers in each subband. For more details on source allocation, readers should refer to [12,13] and [14]. This part ensures the maximum possible compression of the input image at a given rate while keeping the best visual quality.

The quantized image subbands are then being encoded using the proposed algorithm described in Section 5.1 respecting the restrictions imposed by the biological procedures. At this point it is important to highlight the fact that the state of the art encoding procedures do not take into consideration the last restriction of pattern repetitions. In our case however, as we wish to use quantization of wavelet subbands in order to achieve high compression and decrease the synthesis cost, it is possible that after quantization we get a long sequence of repeated values in the quantized coefficients. The DNA coding of such a sequence can produce pattern repetitions which is an ill case which is more likely to introduce higher percentages of sequencing errors. Thus, our encoding algorithm tackles this problem by applying some kind of randomness as described in Section 5.2. For the evaluation of our encoder's efficiency we carried out two simulations. A first one compressing the input image as described above and a second one which transcodes directly the bitstream coming from the output of a JPEG2000 codec. For the comparison to the state of the art we used some compression measures that have been used also in [8].

Simulations have been carried out on Lena image of size 512 × 512 pix. For the case in which compression is used, one can see on Fig. B.6 the evolution of the Peak SNR (PSNR) in function of the coding rate in bits per nucleotide. In Table 1 is reported the

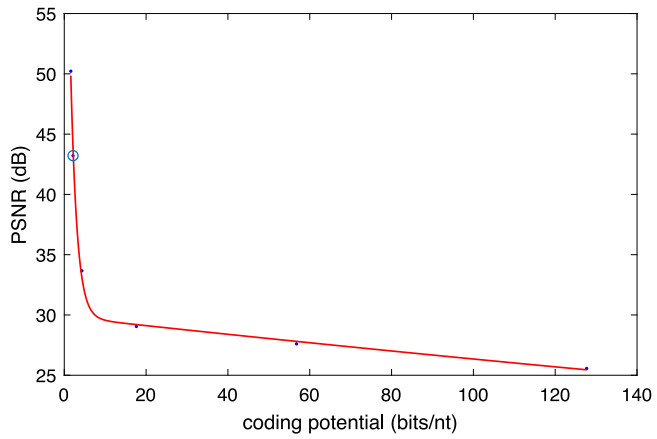

Fig. B.6. PSNR in function of the coding rate in bits per nucleotide for the image Lena of size 512 × 512 pix. The selected point (at 2.14 bits/nt) is the one represented in our results in Table 1.

coding result at 2.14 bits per nucleotide which corresponds to a nearly lossless compression, allowing a fair comparison with the state-of-the art approaches. The corresponding PSNR is equal to 43.21 dB providing a perfect reconstructed image without any visual artifacts. The visual quality of the compressed image is shown in Fig. B.7. The coding results using the proposed solution are presented in the last two columns of Table 1. Those results show that when appropriately controlled, the slightest compression can provide good results without important visual loss. Furthermore the theoretical performance of our encoder when used for transcoding a binary stream (called "raw data" in the Table) might not outperform all the existing encoding methods but is still comparable. At this point it is necessary to highlight the fact that unlike the works in [6] and [8] which outperform this work, our encoder is fixed length in order to ensure higher robustness to the sequencing error and therefore cannot take advantage of further compression allowed by variable length codecs.

7. From digital image to biological data

As described in 3, DNA digital data storage is a multidisciplinary process which is aimed to allow efficient, long-term storage of digital data. To create a robust code which encodes any input data into a quaternary stream composed by the 4 DNA bases A, T, C, and G, we have proposed in 5.1 a new encoder that respects the restrictions imposed by the biological procedure of DNA sequencing. DNA synthesis is an error-free biological procedure under the condition that the DNA strands that are being synthesized into DNA are not longer than 150–200 nts. In the case where longer DNA strands are created biologically in vitro, the synthesis error is no longer negligible and increases exponentially with the increase in the oligo length. To maximize the reliability of the DNA synthesis process and to ensure the accuracy of the synthesized DNA strands, in our experiments the encoded data needs to be cut into smaller chunks of base-pairs to be piece-wise synthesized into DNA. This yields the need for inserting some special headers in each synthesized oligo which contains information about the position of each DNA chunk in the initial encoded sequence as also some headers which contain information for the encoding parameters. Those headers are necessary for the correct decoding of the stored data. This formatting procedure is described in detail in the following section.

7.1. Formatting the oligos

In our experiments we have formatted our oligos according to the needs of the compression that has been used for the encoding. More precisely, special headers should be inserted in each oligo containing information about the position of the encoded information forming a data oligo (DO). Furthermore, as each one of the B subbands is quantized using a different optimal quantization step, we also need to encode some oligos containing information for the decoding. More specifically we create B subband-info oligos (SIO) containing information about the quantization of each type of subband as also a global-info oligo (GIO) that contains general information about the format of the encoded image and the levels of DWT that have been used. The detailed description of the formatting are depicted in Fig. B.8.

7.2. DNA synthesis and storage

The third part of the encoding workflow consists of three sub arts of biological procedures. More precisely when the digital information is encoded into a quaternary sequence and formatted into oligos, it has to be sent to a biological laboratory to be synthesized "in vitro" into DNA.[1] This DNA is synthetic which means that the created sequences do not contain any real genes and thus they do not serve in the existence of any living organism. However, while it is composed by real DNA nucleotides that have been assembled in a strand, the synthetic DNA shares the properties of a real DNA molecule which is capable

[1] For our experiment, DNA synthesis has been done by the Company Twist Bioscience.

Reconstructed image Quantization error

Fig. B.7. Visual quality of the compressed Lena image of size 512 × 512 pix that has been compressed and encoded into DNA at 2.14 bits/nt, PSNR=43.21 dB.

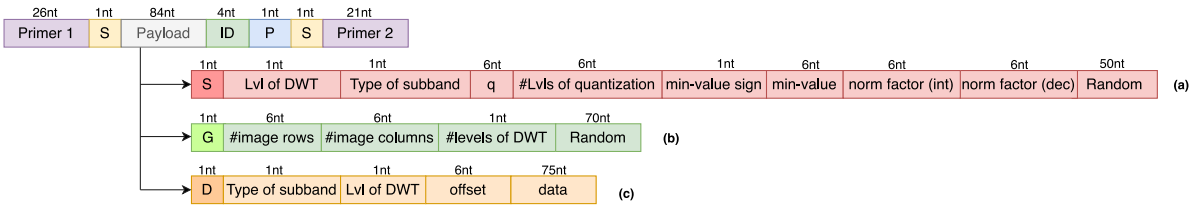

Fig. B.8. Format of the different oligos that must be used to allow image decoding and reconstruction. Three kinds of oligos are created: (a) global-info oligo (GIO), (b) subband-info oligos (SIO) and (c) data oligo (DO).

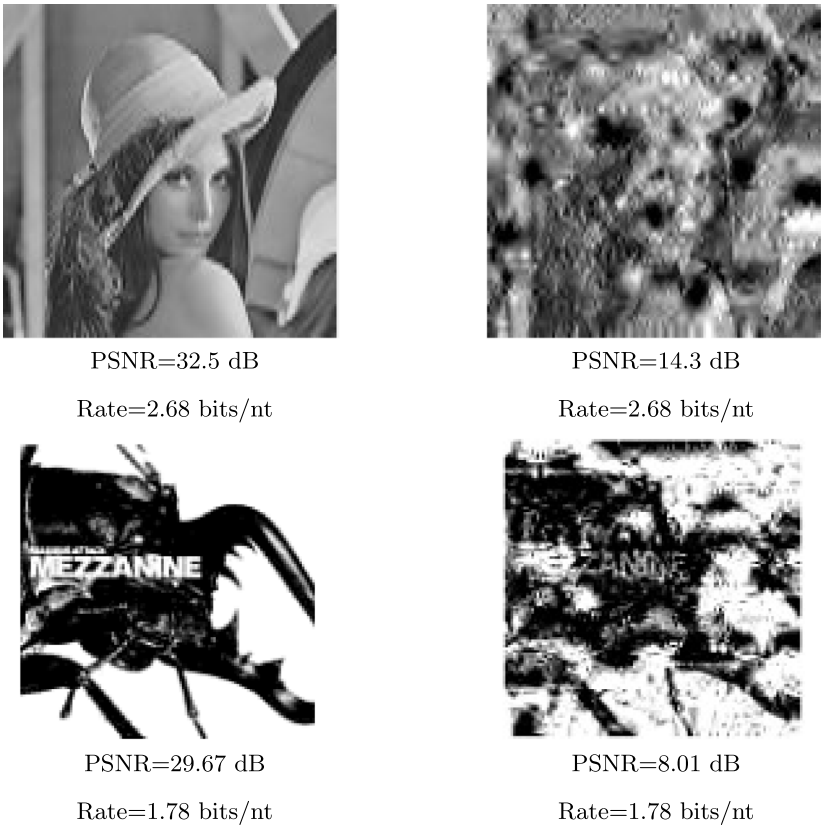

Fig. B.9. Visual results for two different cases of reconstruction: using the most frequent oligos (left column) and random selection (right column). For the left images the PSNR value is only due to the error inserted by the quantization process as we have managed to get a reconstruction without any sequencing noise, for the right images (random selection), both quantization and sequencing error appear.

of carrying a great amount of information into a tiny volume (as the volume of a cell nucleus), for a very long time. Stability of the synthesized oligos can be good for several months at −20 °C. For longer storage DNA can be encapsulated in DNAshell[2] capsules, a storage that protects it from contacts with oxygen and water. This kind of special capsules is ideal for DNA storage and can keep the encoded data safe from corruption from years to centuries, and probably over.

7.3. DNA sequencing

To decode the encoded data and retrieve the stored information one needs to read the stored DNA fragments that have been encapsulated in the storage capsule. This process is performed by special machines called sequencers; in our experiment we utilized the Illumina NextSeq 500 instrument. As mentioned in previous sections, sequencing is an error prone procedure and can cause significant errors as insertion, deletion or substitution of nucleotides. The use of a robust encoding scheme, like the one proposed by this work, which takes into consideration the biological restrictions that have been analytically described in Section 3, may reduce the probability of error occurrences. However, in order to ensure the accuracy of the decoded data, the stored oligos are firstly cloned using PCR amplification, a biological process which uses enzymes to create many copies of a DNA sequence. Then the copies can be read by a sequencer using the method of bridge amplification (BA) which allows reading the oligos while cloning them into more copies [18]. This redundancy is very important for the reduction of the sequencing error. One can imagine the role of the PCR and BA like the one of classical repetition coding used for transmission over a noisy channel that may corrupt the transmission in various positions. As in repetition coding, PCR and BA produces during sequencing many copies of the oligos hoping that only a minority of these copies will be corrupted by the sequencing noise.

7.4. Experiment

In this study we have carried out a real biological experiment for storing two small images into DNA. More specifically, one image of Lena of size 128 × 128 pixels and a 120 × 120 pixels cover image from

[2] For our experiment, encapsulation has been done by the Company Imagene, Evry, France.

the famous Massive Attack album Mezzanine have been transformed into synthetic DNA. The choice of the size of the images was constrained by the high expenses of the biological procedures involved in the experiment. Here, the coding leads to a PSNR equal to 32.5 dB at 2.68 bits/nt for the image of Lena and a PSNR equal to 29.67 dB at 1.78 bits/nt for the image of Mezzanine. As shown in Fig. B.9 (left image), this experiment proves the feasibility of correctly retrieving back the stored image from DNA. The values of PSNR has been computed by comparing the encoded images to the ones before compression. Here, The sequencer provided us with a data set containing many copies of each stored oligo which also contained sequencing errors. In order to test the sequencer's reliability we tested two different ways of selecting the oligos for reconstructing the initial image. First we tested the reconstruction using the oligos with the highest frequencies which we assume to be the most representative ones. Then we also tested the case of random oligo selection. The visual results are presented in Fig. B.9 (right images).

It is clear that by choosing the most frequent oligos from the different copies provided by the sequencing, it is more probable to achieve the best possible reconstruction of the image.

8. Conclusions

In this work we have proposed a new encoding algorithm for the robust encoding of digital images into DNA. Our proposed solution has important advantages compared to previous state of the art works as it can be applied on any kind of input data format (binary, symbols, quantized samples...) and tackles the problems of biological constraints, including pattern repetitions that according to some studies may affect the performance of the sequencing procedure by introducing more errors in the decoded DNA strands. In a comparison of our work with a popular state of the art encoding algorithm for DNA data storage we have evaluated the performance of our proposed encoder and received some very promising results. Furthermore, we discussed the importance of minimizing the DNA synthesis cost by introducing a compression method which is optimally controlled at the encoder side. Using some classical compression techniques, we made a very first attempt to store an image, optimally compressed at a given rate, into DNA proving the feasibility of our proposal. The outcome of this study is very encouraging suggesting that the introduction of more sophisticated quantization solutions at the encoder side, capturing the source statistic distribution for example, could provide even more interesting results.

CRediT authorship contribution statement

Melpomeni Dimopoulou: Concept, Design, Analysis, Writing, or revision of the manuscript. **Marc Antonini:** Concept, Design, Analysis, Writing, or revision of the manuscript. **Pascal Barbry:** Concept, Design, Analysis, Writing, or revision of the manuscript. **Raja Appuswamy:** Concept, Design, Analysis, Writing, or revision of the manuscript.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Annex A. Our proposed algorithm

Algorithm 1 Encoding Algorithm

Compute length l of codewords needed for encoding all k levels of quantization:
if $\log_{10} k$ not an integer **then**
  if $10^{\lfloor \log_{10} k \rfloor} * 4 \leq k$ **then**
    $l = \lfloor \log_{10} k \rfloor * 2 + 1$
  else $l = \lceil \log_{10} k \rceil * 2$
  end if
else $l = \log_{10} k * 2$
end if
Build code $\mathcal{D}$ of L different codewords:
if l is even **then**
  Construct all possible codewords of length l using $\frac{l}{2}$ choices from $\mathcal{D}_1$
else if l is odd **then**
  Construct all possible codewords of length l by using $\frac{l}{2}$ choices from $\mathcal{D}_1$ adding one symbol from $\mathcal{D}_2$
end if
Mapping of index values of quantization to codewords from $\mathcal{D}$
Compute: $m = \lfloor \frac{L}{k} \rfloor$
Compute: $\Gamma(\hat{x}^i) = \mathcal{D}^*(i + rand(1, m-1) * k)$

Annex B. Figures

See Figs. B.1–B.9.

References

[1] Y. Lei, M. Sakakura, L. Wang, Y. Yu, R. Drevinskas, P.G. Kazansky, Low-loss geometrical phase elements by ultrafast laser writing in silica glass, in: CLEO: Applications and Technology, Optical Society of America, 2019, pp. ATu4I–4.
[2] Y. Shimotsuma, K. Miura, H. Kazuyuki, Nanomodification of glass using fs laser, Int. J. Appl. Glass Sci. 4 (3) (2013) 182–191.
[3] G.M. Church, Y. Gao, S. Kosuri, Next-generation digital information storage in DNA, Science (2012) 1226355.
[4] J. Bornholt, R. Lopez, D.M. Carmean, L. Ceze, G. Seelig, K. Strauss, A DNA-based archival storage system, Oper. Syst. Rev. 50 (2) (2016) 637–649.
[5] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E.M. LeProust, B. Sipos, E. Birney, Towards practical, high-capacity, low-maintenance information storage in synthesized DNA, Nature 494 (7435) (2013) 77.
[6] R.N. Grass, R. Heckel, M. Puddu, D. Paunescu, W.J. Stark, Robust chemical preservation of digital information on DNA in silica with error-correcting codes, Angew. Chem. Int. Ed. 54 (8) (2015) 2552–2555.
[7] M. Blawat, K. Gaedke, I. Huetter, X.-M. Chen, B. Turczyk, S. Inverso, B.W. Pruitt, G.M. Church, Forward error correction for DNA data storage, Procedia Comput. Sci. 80 (2016) 1011–1022.
[8] Y. Erlich, D. Zielinski, DNA fountain enables a robust and efficient storage architecture, Science 355 (6328) (2017) 950–954.
[9] L. Organick, S.D. Ang, Y.-J. Chen, R. Lopez, S. Yekhanin, K. Makarychev, M.Z. Racz, G. Kamath, P. Gopalan, B. Nguyen, et al., Scaling up DNA data storage and random access retrieval, 2017, bioRxiv, 114553.
[10] M. Dimopoulou, M. Antonini, P. Barbry, R. Appuswamy, DNA coding for image storage using image compression techniques, in: CORESA 2018, 2018.
[11] M. Dimopoulou, M. Antonini, P. Barbry, R. Appuswamy, A biologically constrained encoding solution for long-term storage of images onto synthetic DNA, in: EUSIPCO 2019, 2019.
[12] M. Kaaniche, A. Fraysse, B. Pesquet-Popescu, J.-C. Pesquet, A Bit Allocation Method for Sparse Source Coding-Extended Version, Report, 2013.
[13] K. Ramchandran, A. Ortega, M. Vetterli, Bit allocation for dependent quantization with applications to multiresolution and mpeg video coders, IEEE Trans. Image Process. 3 (1994) 533–545.
[14] Y. Shoham, A. Gersho, Efficient bit allocation for an arbitrary set of quantizers (speech coding), IEEE Trans. Acoust. Speech Signal Process. 36 (9) (1988) 1445–1453.
[15] T.J. Treangen, S.L. Salzberg, Repetitive DNA and next-generation sequencing: Computational challenges and solutions, Nature Rev. Genet. 13 (1) (2012) 36.
[16] S.H.T. Yazdi, Y. Yuan, J. Ma, H. Zhao, O. Milenkovic, A rewritable, random-access DNA-based storage system, Sci. Rep. 5 (2015) 14138.
[17] S.H.T. Yazdi, R. Gabrys, O. Milenkovic, Portable and error-free DNA-based data storage, Sci. Rep. 7 (1) (2017) 1–6.
[18] I. Illumina, An introduction to next-generation sequencing technology, 2015.