Overview of JPEG DNA coding system for image storage on synthetic DNA

Davi Lazzarotto[a], Michela Testolina[b], Serge Kas Hanna[b], Osamu Watanabe[d], Eva Gil San Antonio[c], Xavier Pic[e], Melpomeni Dimopoulou[c], Marc Antonini[b,c], and Touradj Ebrahimi[a]

[a]Multimedia Signal Processing Group (MMSPG), École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
[b]Laboratory I3S, Université Côte d'Azur and CNRS UMR 7271, France
[c]PEARCODE, Valbonne, France
[d]Takushoku University, Tokyo, Japan
[e]EURECOM, Sophia Antipolis, France

ABSTRACT

With the exponential growth of digital data production, conventional storage technologies face increasing challenges related to scalability, durability, and environmental impact. DNA has emerged as a promising alternative storage medium due to its exceptional density, longevity, and sustainability. In this context, the JPEG Committee (ISO/IEC JTC 1/SC 29/WG 1) has developed the JPEG DNA standard (ISO/IEC 25508), the first specification for image coding on synthetic DNA, which is currently in its final stages of standardization. The goal of the standard is to enable the efficient and robust representation of digital images using DNA as a storage medium, while addressing challenges such as biochemical constraints and high error rates. This paper provides an overview of the JPEG DNA standard, describing in detail its codec-agnostic core coding system able to encode not only images but also arbitrary binary data, and presenting a performance comparison with state-of-the-art alternatives. In particular, the codec is observed to be able to encode data at 1.83 bits/nt without producing homopolymers or repeated patterns and keeping a balanced content of C and G nucleotides. In addition, a discussion is provided on ongoing developments and remaining challenges in the standardization process, including a wide discussion of the performance of the JPEG DNA standard in the presence of noise.

Keywords: JPEG DNA, DNA-based data storage, standardization, image coding, synthetic DNA

1. INTRODUCTION

The demand for data storage has been exponentially growing and is often associated with strict requirements in terms of durability. Because of its longevity and low energy demand, synthetic DNA molecules have been identified as a promising alternative for data storage. Their sequential structure, composed of a series of nucleotides, each representing one of four possible bases (A, C, G, or T), can be viewed as a quaternary code capable of storing digital information. Recent progress on biochemical procedures of synthesis, sequencing, and polymerase chain reaction (PCR) has made it possible to create, read, and copy DNA molecules designed for data storage. In this context, early works[1,2] proposed different encoders for the purpose of storing data into synthetic DNA. These works identified that the errors occurring during synthesis, PCR, and especially sequencing, were highly dependent on the encoded nucleotide sequence.

For this reason, coding constraints were proposed to reduce the number of errors occurring when encoding data into DNA. First, the nucleotide sequence should not contain repetitions of the same nucleotides several times in a row, usually more than three times. These repetitions are often referred to as *homopolymers*, and their existence was observed to deteriorate the quality of the sequencing process. Likewise, the repetition of short patterns between three and five nucleotides should also be avoided. The second constraint that was identified is the proportional content of G and C nucleotides. Previous studies[1] found that extreme GC content values have

Corresponding author: Davi Lazzarotto, davi.nachtigalllazzarotto@epfl.ch

a negative impact in the PCR process, proposing a limit between 40% and 60% of the total encoded stream. Finally, to decrease the errors introduced during synthesis, the DNA molecules should be split into short strands denominated oligonucleotides, or oligos, which should have a length between 100 and 300 nucleotides.

In addition to these biochemical constraints, DNA-based data storage also faces other challenges, such as the high cost of synthesis and sequencing, as well as the low speed for reading and writing data. For this reason, this support is usually associated with the archival of *cold* data, that is, rarely accessed data for long-term preservation. DNA is particularly attractive for related applications due to its durability, density, and passive energy profile. In this context, the JPEG DNA initiative has identified several representative use cases for DNA-based media storage,[3] including long-term cultural heritage preservation, social media cold data archiving, medical imaging retention, and large-scale biomedical repositories. Observing the need for standardized coding solutions, a Call for Proposals (CfP) was launched,[4] which resulted in the adoption of one of the submissions as its preliminary method, denominated verification model (VM). A further collaborative phase resulted in the coding system described in the JPEG DNA standard, which is scheduled to reach the status of Draft International Standard in October 2025. This paper describes the main achievements of the JPEG DNA activity and describes the encoding and decoding methods proposed in Part 1 of the standard, presenting its performance both in terms of compression efficiency and robustness to noise.

2. RELATED WORK

The work from Church et al.[1] introduced a practical framework for encoding digital information into synthetic DNA. This study also analyzed the sequencing errors and established the main constraints to respect when encoding data into DNA. The encoding method proposed by Goldman[2] notably introduced one of the first entropy-based coders tailored for DNA, combining ternary Huffman coding with a rotating base encoder. A critical challenge in DNA storage lies in managing errors introduced during synthesis, amplification, and sequencing. To address this, Grass et al.[5] embedded digital data in silica and applied error correction, pioneering robust storage strategies. This was followed by other error-resilient approaches, including the DNA Fountain by Erlich and Zielinski,[6] and robust coding schemes like Blawat et al.,[7] Yazdi et al.,[8] and recent advances such as DNA-Aeon,[9] HEDGES,[10] and GCNSA,[11] the latter incorporating machine learning techniques. More system-level solutions have emerged recently. For instance, Organick et al.[12] proposed random-access DNA storage at scale, and CMOSS[13] introduced a motif-based, columnar architecture optimized for modularity and scalability. A comprehensive overview of these techniques can be found in the recent survey by Heinis et al.[14]

Moving beyond generic binary storage, the past few years have seen the emergence of image-specific DNA coding techniques that integrate domain knowledge from traditional image compression (e.g., JPEG, JPEG 2000) with DNA biochemical constraints. One of the first targeted image codecs was introduced by Dimopoulou et al.,[15] who proposed a biologically constrained codec for storing images in synthetic DNA. This codec prioritized error robustness, constraint compliance, and low complexity with built-in scalability. Later works enhanced the compression efficiency of the proposed algorithm by using vector quantization.[16] Follow-up works focused on adapting the legacy JPEG standard to DNA storage, either by directly encoding uncompressed images[17] or by transcoding already encoded files.[18] These codecs preserved the quantized DCT structure and applied a quaternary mapping leading to DNA-ready bitstreams. Later improvements introduced more sophisticated entropy coders (e.g., Shannon-Fano[19] and MQ-inspired[20]), specifically adapted to balance entropy efficiency with biochemical constraints. Moreover, the combined use of high-efficiency coding methods such as JPEG XL and JPEG AI with DNA Raptor codes was recently proposed. This method was compared to previous joint source-channel coding solutions,[21] being selected as the first version of the JPEG DNA VM due to its superior rate-distortion performance obtained during the CfP.

In line with broader trends in image compression, several learning-based approaches have recently emerged. Pic et al.[22] proposed a pipeline that uses compressive autoencoders for lossy image compression, followed by a DNA-adapted entropy coder to ensure constraint-compliant representation. This method allows for data-driven latent representations that adapt to DNA channel conditions. Strebel et al.[23] proposed the use of already trained autoencoders combined with a Goldman[2] encoder. Franzese et al.[24] proposed Generative DNA, a representation-learning approach that allows for approximate image reconstruction from compressed DNA. Le et al.[25] introduced a compression scheme using Multiple Description Coding (MDC) and Implicit Neural Representations (INR), which generates multiple redundant yet diverse representations. This makes the system more robust to strand loss and sequencing noise. Similarly, Seo et al.[26] explored lossy compression strategies to increase information density per synthesized base.

These models are increasingly influenced by advances in end-to-end image compression,[27–32] where joint optimisation of encoder-decoder pipelines with learned image compression models allows for efficient and robust coding, traits that are well suited for DNA channels. In this context, Couvreur et al.[33] investigated the use of various learning-based image codecs for source coding within the framework of the JPEG DNA activity.

3. JPEG DNA SCOPE AND PRELIMINARY ACTIVITIES

3.1 Exploration scope and early activities

In September 2020, the JPEG Committee initiated an exploration of standardization needs on the field of DNA-based data storage. The JPEG DNA activity was defined with the scope of creating a standard for efficient coding of images that considers biochemical constraints and offers robustness to noise introduced by the different stages of the storage process based on DNA synthetic polymers.

The activity proceeded by identifying the main use cases driving the field of DNA-based data storage and the requirements related to these use cases, moving towards the definition of common test conditions which should be used to analyse and compare the performance of different image coding solutions in DNA support. These efforts resulted in two key documents[3,34] that guided the activity towards the creation of the JPEG DNA standard. The main components of these documents are outlined in this section.

3.2 Use cases and requirements

The JPEG DNA standardization effort is driven by a range of use cases,[3] each with distinct storage, quality, and access requirements:

- **Long-term media archives and cultural heritage preservation**: Focused on national archives and museums, where lossless coding, high fidelity, and long-term durability are crucial. DNA's longevity makes it ideal for preserving scanned books, 3D models, and metadata-rich digital artefacts.
- **Cold storage for social media platforms**: Massive volumes of personal media data on social networks (e.g., Facebook, Instagram) require low-access, long-duration storage. In this case, some degree of quality degradation may be acceptable.
- **Medical imaging**: Medical images demand high resolution and dynamic range. Due to legal and diagnostic constraints, lossless coding and error resilience are essential for long-term patient record storage.
- **Large-scale biomedical repositories**: DNA storage can support future bioinformatics use cases, including multi-modal, multi-scale medical datasets (e.g., MRI, fMRI, genomics). Compression of higher-dimensional data with scalable and lossless capabilities is needed.
- **Traceability**: DNA-encoded barcodes embedded in physical materials (textiles, concrete, metals) provide a long-lasting, tamper-proof method to track product origin and ingredients. Encoding must ensure biosafety and decode reliably after decades.

These diverse use cases motivate a comprehensive list of technical requirements the JPEG DNA standard aims to fulfill:

- **Compression efficiency:** The codec should significantly outperform binary-to-DNA transcoding methods not including source data compression.
- **Lossless coding:** The standard shall offer lossless coding at a state-of-the-art compression performance.
- **Composite media assets:** The codec should handle collections of media (e.g., images with metadata).

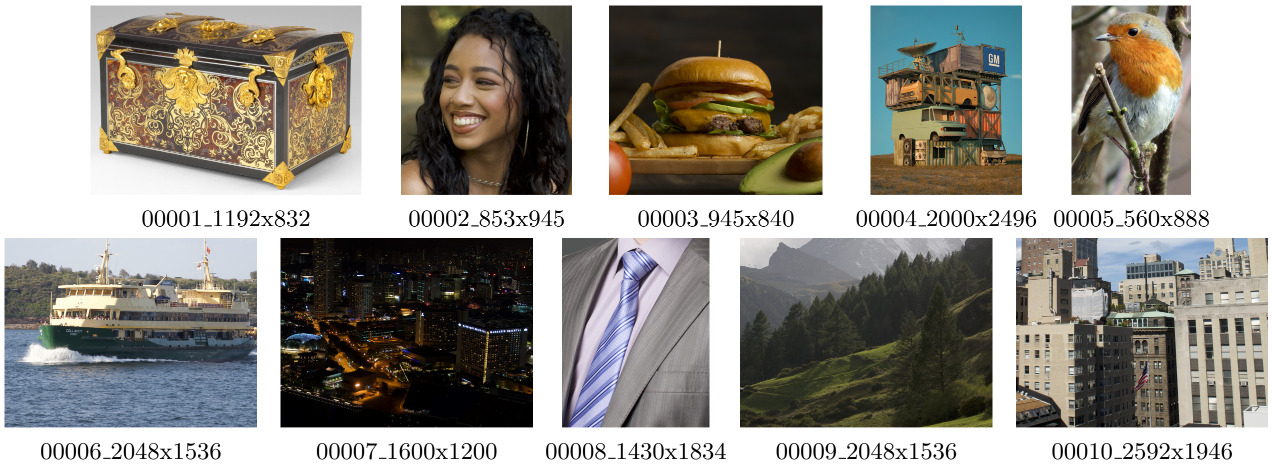


Figure 1: JPEG AIC-3 dataset, adopted in the context of the JPEG DNA CfP

- **Metadata support:** The standard must allow efficient encoding of structured metadata associated with the media content.
- **Privacy and security:** The standard shall offer the capability to efficiently code appropriate privacy and security-related data, associated with the relevant media assets.
- **Random access:** The standard must support selective decoding of parts of the content.
- **Biological constraints:** It must respect biochemical limitations (e.g., GC balance, homopolymers) to ensure molecular stability.
- **Error resilience:** It should tolerate typical synthesis and sequencing errors and offer mechanisms for robustness and recovery.
- **Scalability:** Progressive or scalable representations must be supported to allow partial reads at lower quality or resolution.
- **Ambiguity-free decoding:** Decoded outputs must be deterministic and unambiguous, ensuring consistent interpretation.
- **Artificial recognition:** The output DNA sequences must be recognizable as artificial to prevent confusion with natural DNA.
- **Biosafety:** The standard must ensure that no harmful or biologically active sequences can be produced by accident.
- **Flexible nucleotide sequence composition:** The standard shall be able to decode images based on different code constructions.

3.3 Common test conditions

The JPEG DNA Common Test Conditions[34] (CTC) outlines the criteria adopted for the evaluation of DNA-based coding tools. This document defines a rigorous evaluation framework including dataset specifications, target rates, coding constraints, performance metrics, and anchor implementations.

Anchor coding mechanisms: Three anchor methods were adopted in the context of the JPEG DNA CfP, including one encoding anchor, able to encode uncompressed images into DNA, and two distinct transcoding anchors, designed to translate JPEG bitstreams into DNA support. The selected anchors represent state-of-the-art solutions specifically designed for the use case of image coding into DNA. The first, **Transcoder Anchor 1**, is based on the method proposed by Goldman et al.[2] and performs a direct transcoding of the JPEG bitstream by converting each byte into a ternary representation, which is then encoded using the Goldman codec. The second, **Encoder Anchor 2**, also known as the JPEG DNA Benchmark Codec (BC) and described by Dimopoulou et al.,[17] encodes raw images into DNA while maintaining compatibility with legacy JPEG entropy structures. This method applies quantization to DCT coefficients followed by ternary Huffman encoding and Goldman coding, producing biologically constrained DNA sequences. Finally, **Transcoder Anchor 3**, referred to as the JPEG DNA Benchmark Transcoder (BT) and presented by Secilmis et al.,[18] provides transcoding from JPEG files to DNA by extracting the quantized DCT coefficients and encoding them using the same biologically constrained process as JPEG DNA BC.

Dataset and target rates: The dataset defined in the CTC consists of 10 RGB images (Figure 1), carefully selected to ensure diversity in content, resolution, and visual complexity. Each image is provided in multiple formats to support both encoding and transcoding scenarios: PNG images (RGB color components, non-interlaced), JPEG (including progressive and hierarchical modes), as well as other standardized formats. Image resolutions vary from 560×888 to 2592×1946 pixels, ensuring that codecs are tested on content with differing levels of detail and complexity.

To evaluate rate-distortion performance, a series of target bitrates expressed in nucleotides per pixel (nts/pixel) is defined. These rates (see Table 1) serve as common evaluation points for all submissions.

Objective quality assessment metrics: A key component of the evaluation process is objective quality assessment, which relies on a rich set of well-established objective image quality metrics. The following metrics are used employed to assess the performance of various DNA-based image coding solutions:

- PSNR (Peak Signal-to-Noise Ratio): a well-known metric in the state of the art, measuring the ratio between the maximum possible signal power and the power of the distortion.
- MS-SSIM (Multi-Scale Structural SIMilarity):[35] measures the perceptual similarity between a reference and a distorted image by evaluating structural information across multiple spatial scales.
- IW-SSIM (Information Content Weighted Structural SIMilarity Measure):[36] extends MS-SSIM by incorporating an information-weighted pooling strategy, giving greater importance to regions with higher information content when computing overall image similarity.
- VMAF (Video Multimethod Assessment Fusion):[37] a perceptual video quality metric that combines multiple quality assessment algorithms to predict subjective visual quality, with a particular focus on artifacts introduced by compression and rescaling.
- VIF (Visual Information Fidelity):[38] quantifies the loss of perceptual information by modeling natural scene statistics and evaluating how much information the distorted image retains relative to the reference.
- FSIM (Feature SIMilarity):[39] assesses image quality by combining two complementary low-level features, i.e. phase congruency and gradient magnitude, which capture salient structures and contrast information.
- NLPD (Normalized Laplacian Pyramid Distance):[40] a perceptual distance metric that models two key characteristics of the human visual system, namely local luminance subtraction and contrast gain control, through a multi-scale Laplacian pyramid decomposition.
- PSNR-HVS:[41] an enhancement of the traditional PSNR metric that integrates properties of the human visual system (HVS), such as contrast sensitivity and visual masking.

Biochemical coding constraints: The following constraints are imposed in order to reduce the error rate during DNA synthesis and sequencing:

- Strand length: the strand length should lie in an interval between 100 and 300 nucleotides, enabling compatibility with current synthesis and sequencing technologies.

	Target nucleotide rates (*npp*)						
Image	**r1**	**r2**	**r3**	**r4**	**r5**	**r6**	**r7**
00001_1192x832	1.99	1.70	1.18	1.00	0.85	0.72	0.61
00002_853x945	1.00	0.80	0.57	0.48	0.41	0.35	0.30
00003_945x840	0.83	0.65	0.51	0.40	0.32	0.25	0.20
00004_2000x2496	1.00	0.80	0.53	0.44	0.37	0.30	0.25
00005_560x888	2.00	1.50	0.84	0.68	0.54	0.44	0.35
00006_2048x1536	1.80	1.50	0.91	0.74	0.60	0.49	0.40
00007_1600x1200	1.30	0.86	0.70	0.57	0.46	0.37	0.30
00008_1430x1834	1.96	1.55	1.23	0.97	0.76	0.60	0.48
00009_2048x1536	1.97	1.46	1.08	0.80	0.59	0.44	0.32
00010_2592x1946	0.92	0.77	0.65	0.55	0.46	0.39	0.33

Table 1: Target nucleotide rates (npp) adopted in the context of the JPEG DNA CfP.

- Homopolymer runs: a homopolymer is a sequence of consecutive identical nucleotides (e.g., GGG) in a DNA strand. Homopolymers longer than 3 nucleotides should generally be avoided in DNA strands, while homopolymers exceeding 7 nucleotides in length must be strictly prevented during encoding.
- GC content balance: the proportion of G and C bases in each DNA strand should be maintained between 40% and 50% to promote molecular stability and reduce secondary structure formation.
- Repetition of patterns: short DNA motifs of 3 to 5 nucleotides should not repeat consecutively more than four times, as excessive repetition can lead to synthesis and sequencing errors caused by reduced sequence diversity.

4. THE JPEG DNA STANDARD

4.1 JPEG DNA key parts

With the issue of its first CfP[4] in April 2023, the JPEG DNA activity initiated an open benchmarking process to identify high-performance image coding solutions adapted for DNA-based data storage. The proposals were received in October 2023 at the 102nd JPEG meeting, where the JPEG DNA VM was created based on the technologies used in one of the three proposals. The primary purpose of the VM is to provide a software-based pipeline for evaluating the performance of potential technologies for the JPEG DNA standard presented during the collaborative phase, which was initiated after the CfP. At the same time, a noise simulator was also created to simulate the error characteristics of DNA channels between write and read operations in the DNA-based storage. Many experiments were conducted with the VM and the noise simulator, and results showed that the JPEG DNA project had reached a mature stage. The New Work Item Proposal (NWIP) of the JPEG DNA was submitted to ISO at the 105th meeting, and the JPEG DNA has been an official ISO project (ISO/IEC 25508-1).

The algorithm employed by the JPEG DNA VM will be standardised in JPEG DNA Part 1. Future parts of the standard are expected to include other features, as described below:

Part 1 – JPEG DNA Core Coding System: This part defines the syntax and an accompanying decompression process that is capable of representing bi-level, continuous-tone grey-scale, continuous-tone colour, or multichannel digital samples in a format representing nucleotide sequences for supporting DNA storage. Part 1 is "codec-agnostic". In other words, this part allows for user-defined image compression tools to be combined with the standardised DNA-based coding system.

Part 2 – JPEG DNA Profiles and Levels: This part defines several subsets of the syntax specified in Part 1 including profiles and levels. Some image compression technologies and error correction mechanisms might be included in Part 2 as elements defined in profiles and/or levels. It is worth noting that an error correction mechanism may be specific to a particular image compression technology.

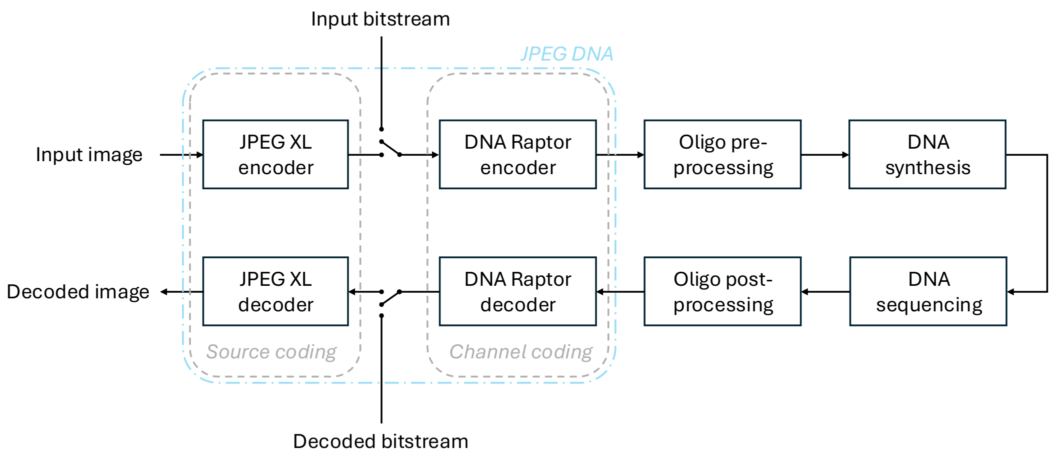


Figure 2: Image coding workflow with JPEG DNA.

Part 3 – JPEG DNA Reference Software: This part contains the reference software of the JPEG DNA series. It acts as a guideline for implementation and as a reference for conformance testing.

Part 4 – JPEG DNA Conformance: This part specifies the framework, concepts, methodology for testing, and criteria to be achieved to claim conformance to multiple parts of the JPEG DNA series. It lists the conformance testing procedures based on Parts 1 and 3, several conformance points for profiles and levels defined in Part 2.

Currently, Part 1 has reached the CD (Committee Draft) stage. The DIS (Draft International Standard) ballot is scheduled for October 2025.

4.2 JPEG DNA coding workflow

The core coding system described in JPEG DNA Part 1 defines an algorithm that encodes images into a set of oligos. This coding method is also flexible enough to support any binary data as its input in addition to raw image data encoding. The decoder is capable of retrieving the input data served to the encoder from oligo readings. JPEG DNA does not define all processes in a DNA-based data storage system, and requires to be combined with other elements within a pipeline as illustrated in Figure 2.

As represented in the diagram, the JPEG DNA standard separates source coding and channel coding into two independent processes. Considering an image as its input, the source encoder is used to reduce the coding rate by exploiting redundancies in the visual data for lossless and lossy coding. Already compressed JPEG bitstreams can also be transcoded to further reduce the coding rate. Moreover, the source encoder can also be skipped to serve the input data directly to the channel encoder, enabling the coding of arbitrary binary data. The channel coder converts the input binary representation into a set of oligos following an arbitrary set of biochemical constraints and enabling correction of errors introduced in the DNA channel.

The different steps illustrated in Figure 2 are described below:

JPEG XL encoder and decoder: The standard specifies JPEG XL as the method used for source coding, leveraging its advantages such as high coding efficiency and mathematically lossless transcoding of JPEG bitstreams. The input to this encoder may be either an uncompressed image or an already encoded JPEG bitstream. The standard also allows the user to skip the source coding module and send the input data directly to the channel coder, allowing other image compression methods to be combined with JPEG DNA. The information of whether or not the JPEG XL encoder is employed in the source data is stored as metadata in the encoded representation, allowing the decoder to determine whether the output of the channel decoder should be sent to the JPEG XL decoder or directly sent as the output data.

DNA Raptor encoder and decoder: The DNA Raptor encoder acts as the channel encoder, translating already compressed information into nucleotide strands. Inspired by DNA Fountain codes,[6] this process combines Raptor codes defined in the RFC5053[42] specification with biochemical constraint verification, producing oligos respecting an arbitrary set of imposed constraints with built-in erasure correction capabilities. This process is described in detail in Section 4.3 and is similar to techniques proposed in previous research works.[43] The DNA Raptor decoder is capable of retrieving the input information from oligos received in a shuffled order. This decoder is also able to compensate for oligos lost at the DNA storage channel if enough redundancy was produced at the encoder. The channel decoder is described in detail in Section 4.4.

Oligo post- and pre-processing: Some processes usually employed in DNA data storage systems are not included in the JPEG DNA encoder and must be conducted as a post-processing operation prior to synthesis. Sequencing methods often require a primer to be attached to both ends of the oligo, i.e. a short nucleotide sequence that physically attaches to a strand in the sequencing device and used in the context of the Polymerase Chain Reaction (PCR). Such primers can also include barcodes for selective sequencing and random access, enabling the decoding of only a specific part of the DNA pool. When reading the data, since the sequencing process results in multiple unordered copies available for each oligo, methods for grouping the copies related to the same source oligo (clustering) and for estimating the source oligo given the multiple noisy reads (consensus) are a part of the processing pipeline. These processes must be conducted as an external part of the JPEG DNA standard.

DNA synthesis and sequencing: The core components of DNA-based data storage are the synthesis and sequencing of strands, which convert between abstract representations of nucleotide bases - i.e. A, C, G, and T symbols - and physical molecules. These processes are not specified by JPEG DNA, which remains agnostic to the methods employed for these purposes as long as they share the same DNA representation.

4.3 Channel encoder

The channel encoder employed by the JPEG DNA VM combines the use of systematic Raptor codes as described in the RFC5053 specification with a proposed iterative binary-to-nucleotide encoding mechanism. The result is a FASTA file containing oligos that follow an arbitrary set of biochemical constraints. While any set of constraints is compatible with the JPEG DNA standard, the JPEG DNA VM implementation uses the list of constraints provided by the JPEG DNA CTC, which are described in Section 3.3. A block diagram of the JPEG DNA VM channel encoder is presented in Figure 3. Each process illustrated in the block diagram is described below:

Block and symbol partition: The channel encoding process starts with the partition of the input bitstream, with initial length F bytes, into Z blocks. A Source Block Number (SBN) is assigned to each block, where the first block receives an SBN value of 0 until $Z-1$. The blocks are then further partitioned into symbols with T bytes. The last symbol of the last block should be padded with zeroes if F is not a multiple of T. This partition mechanism determines that the first Z_L blocks should be partitioned into K_L symbols and that the last Z_S blocks should be partitioned into K_S symbols. These values are computed according to Equations 1 to 5.

$$K_T = \lceil F/T \rceil \tag{1}$$
$$K_L = \lceil K_T/Z \rceil \tag{2}$$
$$K_S = \lfloor K_T/Z \rfloor \tag{3}$$
$$Z_L = K_T - K_S \times Z \tag{4}$$
$$Z_S = Z - Z_L \tag{5}$$

While these equations take as input F, Z, and T, the first value is defined by the length of the input bitstream and is therefore not an independent parameter configurable by the user. Z can either be directly set by the user

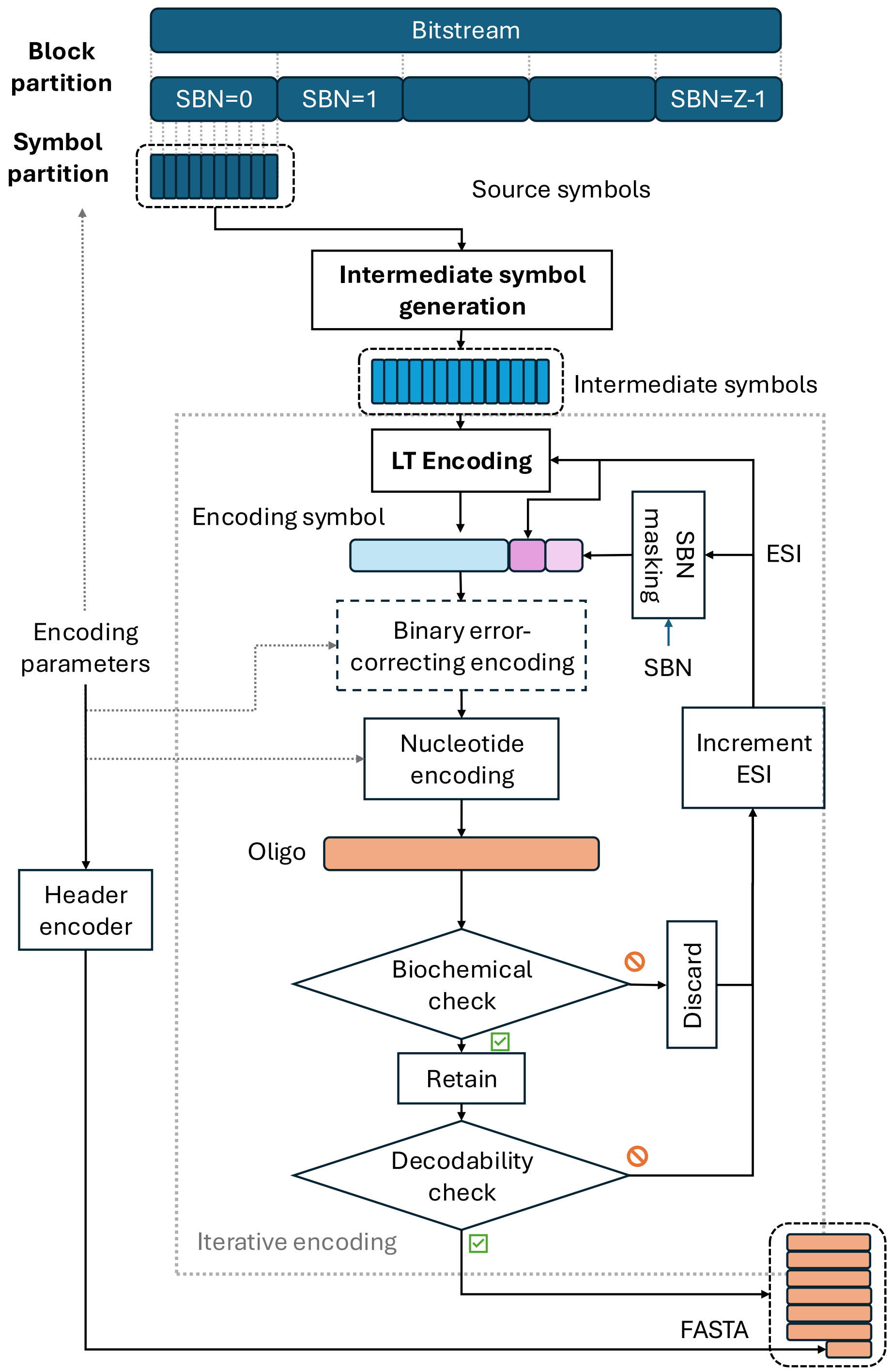


Figure 3: Block diagram of the JPEG DNA VM channel encoder. Blocks written in **bold** are defined in the RFC5053 specification.

or obtained through the equation $\lfloor K_T/K_{max} \rfloor$, with K_{max} being specified as the maximum number of symbols in a block. The RFC5053 specification imposes the limitation that K_{max} should not be larger than 8192.

Effectively, this partition scheme allows all blocks to have approximately the same number of symbols, with $K_L = K_S + 1$. The number of symbols in an arbitrary block is referred to as K in the remainder of this section to facilitate the notation.

Intermediate symbol generation: The encoding process proceeds independently for each block. The K source symbols of the block are employed to compute the $L = K + S + H$ intermediate symbols, where S denotes the number of LDPC symbols and H denotes the number of Half symbols. Both S and H can be deterministically obtained from K according to the RFC5053 specification.

The intermediate symbols $C_0, C_1, ..., C_{L-1}$ are defined as the set of symbols respecting two pre-coding relationships. First, the S intermediate symbols C_K to C_{K+S-1} must be the result of LDPC encoding applied over the first K intermediate symbols C_0 to C_{K-1}. Second, the H Half symbols C_{K+S} to C_{L-1} must be the result of Gray encoding applied over the first $K + S$ intermediate symbols C_0 to C_{K+S-1}. Both the processes of LDPC and Gray encoding are defined by the RFC5053 specification. Finally, the intermediate symbols must follow an additional set of constraints, that is, all source symbols C'_i from C'_0 to C'_K must be the result of Luby Transform (LT) encoding applied over the set of intermediate symbols, using the index i as input to the process.

In practice, these constraints can be translated into a matrix multiplication $A \times C = D$ defined over the finite field $\mathbb{F}_2$, with C being a matrix of dimension $L \times T$ containing all intermediate symbols as rows and D being a matrix of dimension $L \times T$ where the first $S + H$ rows are set to zeroes and the last K rows contain the source symbols. A is an $L \times L$ matrix fully defined by the set of constraints on the intermediate symbols previously defined. Since A and D are known by the encoder, the intermediate symbols C are computed as the solution to this linear system using Gaussian Elimination with Partial Pivoting (GEPP).

LT encoding: The encoding symbols can then be obtained by using the LT encoding function, which takes three arguments as specified in RFC5053: the number of symbols in the block K, the L intermediate symbols for that source block $C_0, C_1, ..., C_{L-1}$, and an index i for the encoding symbol to be generated which is used to produce the pseudo-random triplet d, a, b. The index of the first symbol is set to 0 and is incremented for every new encoding symbol. The generated encoding symbol is the result of the bit-wise XOR operation applied over d intermediate symbols at indices given by a deterministic process defined by the values a and b. This symbol can be identified by both the SBN and index i, which is denominated as the Encoding Symbol Identifier (ESI).

SBN masking: The encoding symbols can only be used to retrieve the input data if their SBN and ESI are known by the decoder. For this reason, these values must be attached to each symbol. While the ESI is directly appended to the symbol, the SBN value goes through a masking process before. The masked SBN is equal to the result of the bit-wise XOR operation applied over the SBN and $R(ESI, 1, 65535)$, where R is the pseudo-random number generator defined in the RFC5053 standard. While the SBN is the same across all encoding symbols in the same source block, the masked SBN is dependent on the ESI and is therefore unique for every symbol. This variability is crucial to the success of the iterative encoding mechanism, as detailed further in this section.

Binary error-correcting encoding: The JPEG DNA encoding system allows for the possibility of using binary error-correcting mechanisms to encode the packet produced by the concatenation of the encoding symbol, the ESI, and the masked SBN. However, since no specific coding mechanism is defined in JPEG DNA Part 1, this process is only defined as a placeholder for future parts of the standard.

Nucleotide pairs	Bit pair
AA, CC, GG, TT	00
AC, CG, GT, TA	01
AG, CT, GA, TC	10
AT, CA, GC, TG	11

Table 2: Nucleotide encoder B translation table for last bit pair

Nucleotide encoding: The binary packet is translated into a nucleotide-based representation. Part 1 of the JPEG DNA standard defines two possible nucleotide encoding mechanisms, i.e. nucleotide encoders A and B. Nucleotide encoder A groups each input byte into four groups of two bits and converts each group into one nucleotide according to the following function: $\{00, 01, 10, 11\} \rightarrow \{A, C, G, T\}$. This encoder achieves the highest information density, but often results in oligos not following the biochemical constraints. Nucleotide encoder B also divides each byte into groups of two bits but instead maps the byte into a sequence of five nucleotides following the mechanism proposed by Blawat et al.[7] The first three bit pairs are mapped into one nucleotide using the same function as nucleotide encoder A. These three nucleotides are placed in the first, second, and fourth positions of the sequence corresponding to that byte, respectively. The last bit pair is mapped into a pair of nucleotides according to Table 2. The first nucleotide of the pair is placed in the third position of the encoded sequence, while the second is placed at the last.

Since Table 2 enables four encoding alternatives for each input byte, the options generating sequences where either the first three nucleotides are the same or the last two nucleotides are the same are not considered. These constraints result in encoded sequences where no homopolymers longer than three nucleotides are present in the output representation. Even after the removal of the invalid sequences, at least two encoding options are available no matter the input byte. The valid nucleotide encoding sequence are grouped into two clusters, where cluster A contains all the first encoding possibilities for each byte according to Table 2 and cluster B groups all the second encoding possibilities. When encoding a sequence of bytes, bytes at odd positions (first, third, fifth, etc.) are encoded using cluster A while bytes at even positions (second, fourth, sixth, etc.) are encoded using cluster B, resulting in an alternating cluster pattern. While this mechanism could allow for the detection of insertion and deletion errors added between the encoder and decoder,[7] no such detection mechanism is specified in the normative decoder from JPEG DNA Part 1.

Biochemical check: The oligo produced by the nucleotide encoder may not follow the biochemical constraints outlined in Section 3.3. Nucleotide encoder A does not contain any mechanism to avoid biochemical constraints, while nucleotide encoder B only avoids the generation of homopolymers but does not take into account the proportion of GC content or the presence of repeated patterns. For this reason, the resulting oligo is screened and discarded if any of the biochemical constraint patterns described in 3.3 are detected. The removal of entire oligos is possible since the employed Raptor encoder is able to produce a very large number of encoding symbols given a set of K input source symbols. Meanwhile, the decoder only requires any combination of $K + \epsilon$ symbols to decode the source block, with ϵ representing a small overhead.

There is, however, a limitation on the number of encoding symbols produced for a source block. Since the ESI must be encoded with 16 bits to be included in the packet, its value cannot go above 65535. The majority of these values may result in invalid oligos depending on factors such as the symbol size T, the set of imposed constraints, and the employed nucleotide encoder. For this reason, the value of K for each block must be carefully chosen in order to allow for the existence of at least $K + \epsilon$ valid oligos for all source blocks. Otherwise, the need for ESI values higher than this limit would result in an encoding error.

This screening mechanism also clarifies the need for the SBN masking process. If the encoding of the SBN results in a section of the oligo that does not follow the biochemical constraints, e.g., due to the existence of a homopolymer, then all the oligos for the same source block would have the same issue, which would prevent the generation of any valid oligos for that source block. However, with the proposed mechanism, the encoded masked SBN is also dependent on ESI and is therefore different for every oligo. The presence of unwanted patterns in the encoded nucleotide sequence related to the masked SBN is not persistent across the source block, allowing the iterative encoder to find a set of valid oligos.

Variable	Number of bytes	Description
F	6	Length of the encoded bitstream in bytes.
Z	2	Number of source blocks.
T	2	Symbol length in bytes.
NDI	1	Nucleotide decoder indicator.
BDI	1	Binary error-correcting decoder indicator.
IDI	1	Image decoder indicator.

Table 3: Variables encoded in the header. NDI is set to 0 for nucleotide decoder A and 1 for nucleotide decoder B. BDI is always set to 0 in JPEG DNA Part 1. IDI is set to 0 for no source decoding and to 13 for JPEG XL source decoding.

Decodability check: Every time a valid oligo is produced, the encoder verifies whether the decoder would be able to retrieve the source block from the set of produced encoding symbols. Once K valid oligos were produced, the GEPP solver of the intermediate symbol generation process from the decoder (described in Section 4.4) is run on the set of encoding symbols corresponding to the generated oligos. If the solver is able to successfully retrieve the intermediate symbols, then the encoding process can be stopped and the $K + \epsilon$ generated oligos are used to represent the source block. Alternatively, additional oligos can be generated as an overhead to allow for the correction of erasure errors, i.e. lost oligos, at the decoder. In this case, the generation of oligos is continued until $K + \epsilon + \lceil \theta K \rceil$ valid oligos were produced, with θ being defined as the overhead.

Increment ESI: Whether the last produced oligo was discarded or the encoding process was not finished, the current ESI is incremented for the production of the next encoding symbol with the LT encoder. This process closes the encoding loop and enables the generation of a decodable set of valid oligos for any encoding block. However, if the previous ESI is equal to the maximum value of 65535, then the encoding process is stopped with an error. Careful configuration of the encoding parameters is required to avoid this scenario. In practice, the value of K_{max}, the value of T, the employed nucleotide encoder, or the enforced biochemical constraints should be modified if encoding fails due to an overflow of the ESI.

Header encoder: Some parameters must be encoded together with the oligos representing the source blocks to allow the decoder to successfully retrieve the input information. The values of F, Z, and T are required for the decoder to infer the number of symbols K in each block. Moreover, the decoder must be informed of which nucleotide encoding mechanism is used and whether JPEG XL decoding should be applied to the output of the channel decoder. These parameters are encoded in a header. The header encoding process first represents the variables described in Table 3 in binary form, resulting in 13 bytes.

The nucleotide encoder B is employed to translate the header bytes into nucleotides. While this encoder includes a mechanism to avoid the production of homopolymers, other biochemical constraints are not taken into account. For this reason, the binary header goes through an iterative encoding process for conversion into nucleotides. This process starts with the scrambling, which applies the XOR function between each byte at a position i in the binary header and the value $R(S_H, i, 255)$, where S_H is denominated the header seed and R is the random number generator defined in the RFC5053 specification. The iterative encoding process first sets $S_H = 0$, produces the scrambled header by applying the XOR function from $i = 0$ to 12, appends the seed encoded as an unsigned 8-bit integer, and sends the resulting binary stream to the nucleotide encoder. If the resulting sequence does not comply with the biochemical constraints described in Section 3.3, then S_H is incremented and the process is repeated iteratively until a valid encoding is found. The first valid header oligo is then added to the set of oligos generated for the main section and encoded into a FASTA file, which can be used to synthesize a set of DNA strands fully representing the input data.

4.4 Channel decoder

The channel decoder receives as input a set of oligo reads generated after sequencing and pre-processing, according to the workflow described in Section 4.2. Figure 4 illustrates how the FASTA file describing the sequenced oligos can be used to reconstruct the input binary data. The different blocks of this diagram are described below:

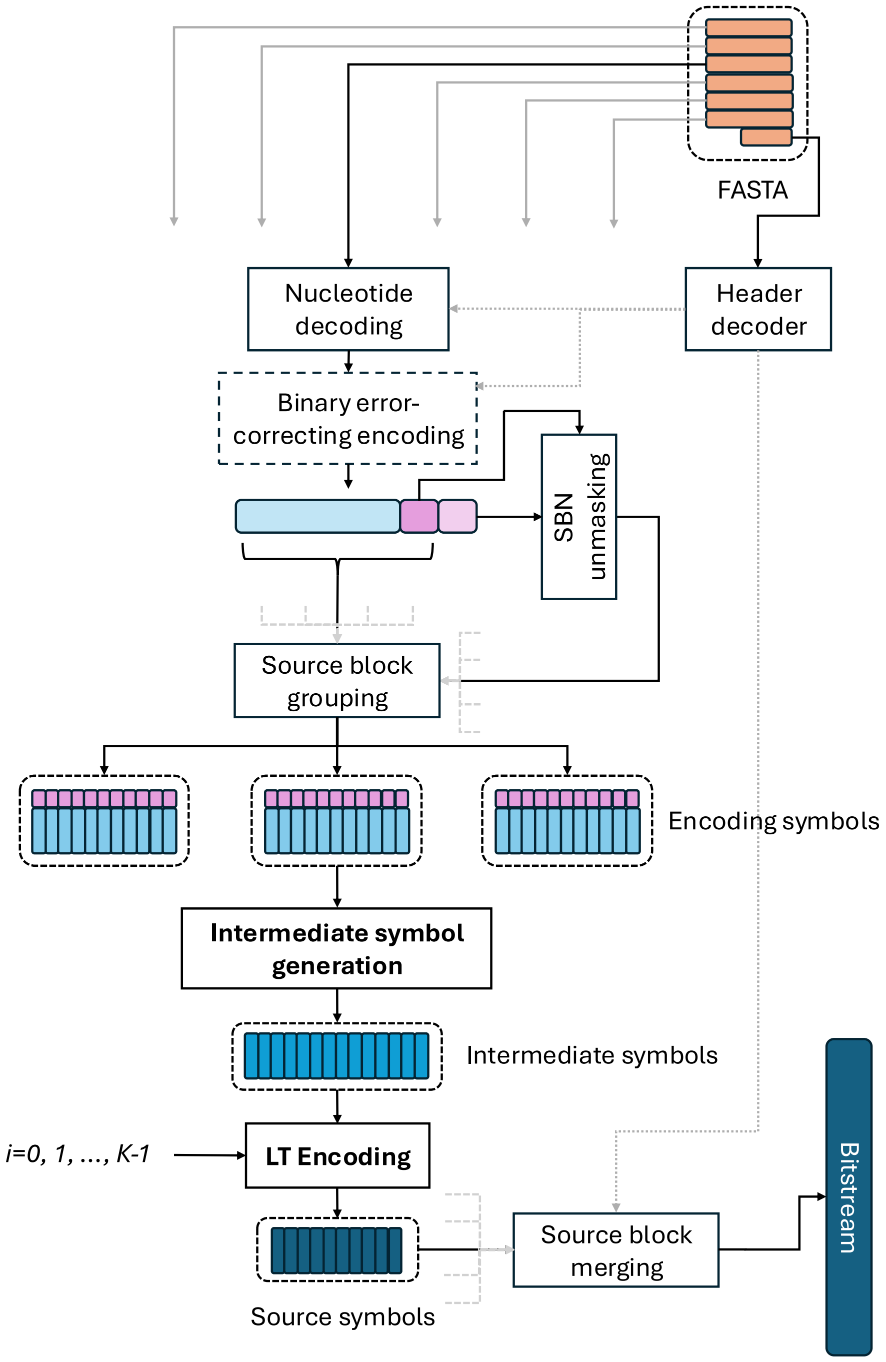


Figure 4: Block diagram of the JPEG DNA VM channel decoder. Blocks written in **bold** are defined in the RFC5053 specification.

Header decoder: The first step of the decoder consists of identifying the header by searching for the shortest oligo across the entire pool. This oligo is then decoded using nucleotide decoder B, which splits the oligo into sequences of five nucleotides and maps each short sequence into one byte. The retrieved sequence is then de-scrambled by identifying the seed S_H as the last byte of the header and applying the XOR operation between each byte and $R(S_H, i, 255)$, where R is the random number generator defined in the RFC5053 specification and i is the index of the byte in the header between 0 and 12. The de-scrambled header is then parsed according to Table 3, and the corresponding parameters are employed in the decoding of the remaining oligos.

Nucleotide decoding: Each oligo corresponding to the main section is decoded into a packet according to the nucleotide decoder indicated by the NDI parameter. In particular, if $NDI = 0$ then nucleotide decoder A is employed, while if $NDI = 1$ then nucleotide decoder B is employed. Since no binary error-correcting decoder is specified in JPEG DNA Part 1, the decoded packets are directly sent to the next stage.

SBN unmaksing: The decoded packet is parsed to obtain the encoding symbol, the ESI, and the masked SBN. The original SBN value is then retrieved by applying the XOR operation between the masked SBN and the value given by $R(ESI, 1, 65535)$, as specified in Section 4.3.

Source block grouping: Since the Raptor decoding process is conducted independently for each source block, all packets must be separated according to their SBN value. Starting from $SBN = 0$ until $Z - 1$, all packets with the same SBN value are grouped together. The encoding symbols from each source block are then sent to be independently decoded by the Raptor decoder.

Intermediate symbol generation: The Raptor decoding process for the encoding symbols of a source block starts with the intermediate symbol generation process, which is equivalent to the same process in the encoder. The symbols $C_0, C_1, ..., C_{L-1}$ can be found from a set of encoding symbols C'_i, with $i \in \mathbb{E}$ and $\mathbb{E}$ being defined as the set of ESI values found in the packets received by the decoder for a given source block. This operation is conducted by solving the linear system $A \times C = D$, where D is a matrix of dimension $M \times T$, with $M = S + H + N$ and $N = |\mathbb{E}|$ being the number of encoding symbols received by the decoder for a given source block. The first $S + N$ rows of D are set to zero, while the last N rows are set to the values of the received encoding symbols. The matrix C is the same as in the equivalent process from the encoder side: it has dimensions $L \times T$ with $L = S + H + K$ and each row corresponds to an encoding symbol. The binary matrix A with dimensions $M \times L$ is defined in the same way as in the encoder for the first $S + H$ rows. For the last N rows, the value A_{ij} is set to 1 if the j^{th} intermediate symbol was employed in the XOR operation to generate the encoding symbol at the i^{th} row of D. Note that while the encoding symbols can be arranged in any order in the matrix D, the same ordering should be applied when defining the rows of matrix A. Once A and D are defined, the intermediate symbols can be generated by solving the linear system $A \times C = D$ through GEPP. Note that the system can be solved if A has rank equal to L. In the opposite case, the system cannot be solved, and the decoding process fails.

LT encoding: Once the L intermediate symbols were generated, the source symbols can be generated with LT encoding with an equivalant process as in the encoder side. In particular, the following parameters are served to the LT encoder for the generation of a source symbol: the value K computed from the parameters received in the header, the L intermediate symbols for that source block $C_0, C_1, ..., C_{L-1}$, and the index i of the source symbol, which must be set to values ranging from 0 to $K - 1$ to produce the K source symbols.

Source block merging: The retrieved source symbols are then concatenated within each source block and across all blocks received in the FASTA file, resulting in the bitstream. If this process results in a number of bytes higher than F, then only the F first bytes are retained. This bitstream is then sent to the JPEG XL decoder if $IDI = 13$ or sent directly as the output if $IDI = 0$.

5. ERROR CORRECTION MECHANISMS

The JPEG DNA VM, as it is defined in the JPEG DNA Standard Part 1, is capable of correcting erasure errors within its Raptor decoding mechanism. Given that the amount of oligos produced by the decoder is higher than $K + \epsilon$, i.e. the minimum value of oligos required for decodability, then the decoder may be able to reproduce the input image even if oligos are lost during the synthesis, storage, and sequencing processes. The requirement for this to happen is that, for every source block, the rank of the $M \times L$ matrix A in the intermediate symbol generation process of the decoder must be equal to L.

However, errors introduced within oligos, such as nucleotide insertions, deletions, and substitutions, will be translated to the packets and are likely to either make the linear system solved in the intermediate symbol generation unsolvable or result in the generation of errors within the intermediate symbols, which will propagate to the source symbols and to the bitstream. These errors impair the source decoding process and prevent the decoder from retrieving the encoded image.

However, the encoder and decoder block diagrams illustrated in Figures 3 and 4 include the possibility of adding binary error-correcting coding mechanisms. An example of such a mechanism is Reed-Solomon (RS) codes. RS codes can be defined over a $\mathbb{F}_{2^8}$ finite field, considering every input byte as a symbol and adding redundant $2 \times t$ symbols to allow for the correction of t substitution errors. Moreover, if more than t errors are present, the decoder may also be able to detect the erroneous packet and exclude it from the decoding process to prevent errors from propagating to the Raptor decoder. Cyclic Redundancy Checks (CRC) can also be used if the sole purpose is error detection. The proposed architecture can also accommodate nucleotide coding mechanisms with built-in error-correction capabilities. Codes for correction of insertion and deletion errors, such as HEDGES,[10] could be employed. These methods were evaluated and compared in a recent work[44] using a previous version of the JPEG DNA VM.[21] Another alternative is the use of schemes that combine both binary-based and nucleotide-based codes. For instance, a decoder compatible with nucleotide encoder B could be used to translate insertions and deletions into substitutions by taking advantage of the alternating cluster pattern.[7] The generated substitution errors could then be corrected using RS codes. Such an approach would use an encoder already defined in the JPEG DNA Part 1 standard and was explored in a recent study.[45]

This paper explores the use of RS codes within the JPEG DNA VM as a binary error-correcting encoder combined with nucleotide decoder A, with results being presented in Section 6.2.

6. PERFORMANCE ANALYSIS AND DISCUSSION

The performance of the JPEG DNA VM is evaluated *in silico* under two conditions. First, the encoded FASTA file containing the oligos representing the input data is directly served as the input to the decoder. This scenario considers an errorless channel, which can be achieved in practice if the clustering and consensus mechanisms applied after sequencing are able to correct all errors introduced in the DNA storage channel. Previous studies[6] demonstrate that this scenario is achievable under realistic conditions. The second evaluation condition employs a channel simulator, adding substitution errors in a multi-draw sampling paradigm. The JPEG DNA VM is coupled with an external clustering and consensus mechanism, with RS codes being employed as error-correcting codes acting in the binary domain.

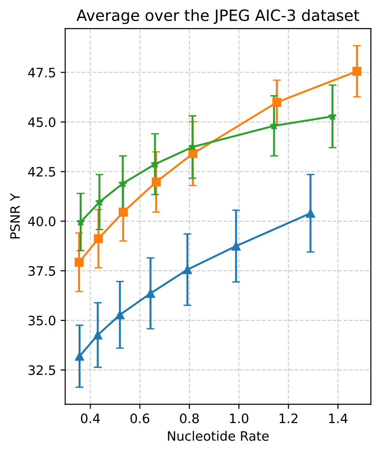
6.1 Performance in errorless channel

The performance of the JPEG DNA VM in a noise-free environment was evaluated using rate-distortion plots. In particular, two different setups were considered, namely *direct encoding* and *transcoding*. In the direct encoding setup, the uncompressed images from the JPEG AIC-3 dataset were directly encoded into DNA using various target nucleotide rates, which were achieved for the evaluated VM by varying the quality parameter of the JPEG XL compression engine. In the transcoding setup, JPEG-compressed images from the JPEG AIC-3 dataset are transcoded into DNA format without full decoding by leveraging the JPEG XL lossless transcoder.

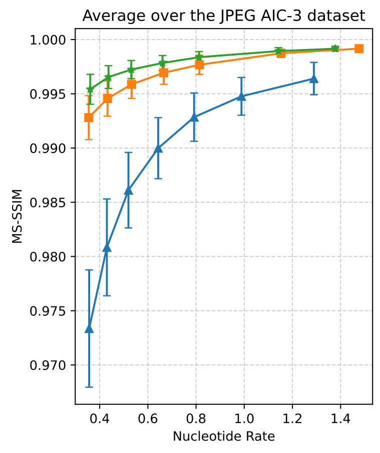
For the direct encoding setup, the performance of the following encoding methods was compared:

- **Anchor BC**: This method corresponds to the encoder anchor 2 described in Section 3.3, which encodes quantized DCT coefficients obtained according to the JPEG coding standard with a ternary Huffman coder followed by a rotating dictionary[2] mechanism for nucleotide encoding.

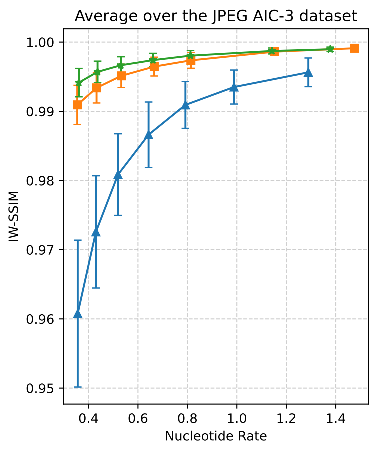
DIRECT ENCODING SETUP



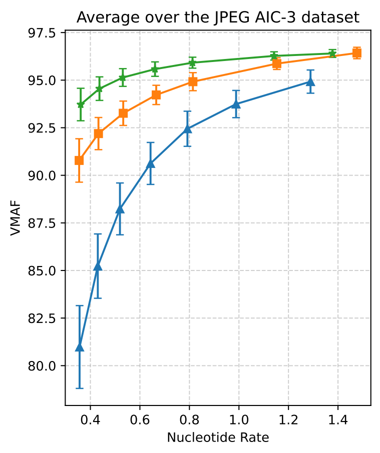
(a) PSNR Y



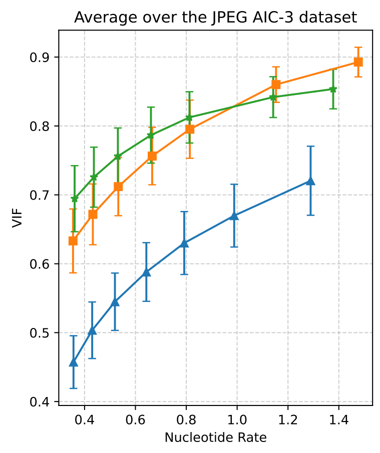
(b) MS-SSIM



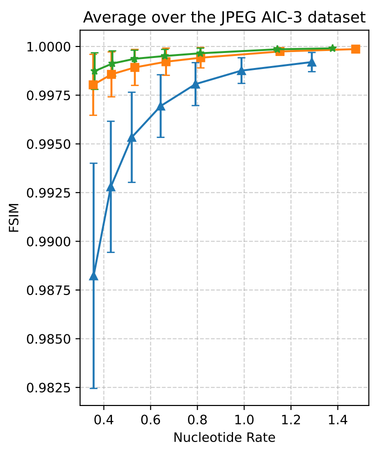
(c) IW-SSIM



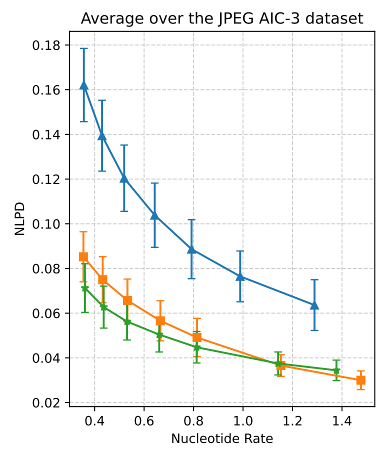
(d) VMAF



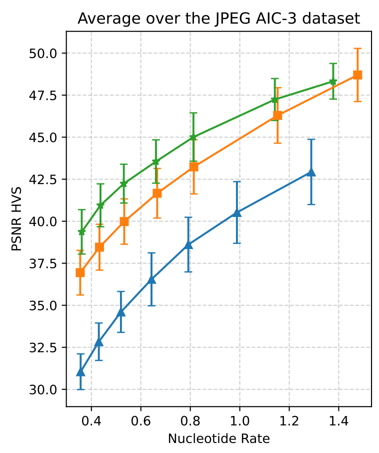
(e) VIF



(f) FSIM



(g) NLPD



(h) PSNR HVS

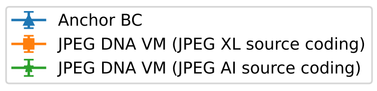


Figure 5: Rate-distortion curves for the *direct encoding* setup for all objective metrics considered in this study, computed using average nucleotide rates and average metric values across all images in the JPEG AIC-3 dataset. Error bars indicate 95% confidence intervals.

- **JPEG DNA VM with JPEG XL source coding**: The JPEG DNA VM integrating JPEG XL version 0.11.1 was adopted. The encoding symbol size was kept at $T = 46$ and the nucleotide encoder A was employed, resulting in oligos with length equal to 200 nucleotides. The maximum number of source symbol per block K_{max} was set to 2600.
- **JPEG DNA VM with JPEG AI source coding**: Images were first encoded using the JPEG AI Reference Software[*], adopting the high operation point (*hop*) model. The output encoded binary files were then encoded into nucleotides using the JPEG DNA VM, bypassing JPEG XL compression and sending the bitstream directly to the channel encoder, which employed the same configuration as the previous encoding method.

For the transcoding setup, the performance of the following methods was compared:

- **Anchor Goldman**: This method corresponds to the transcoder anchor 1 described in Section 3.3, which translates each byte in the JPEG bitstreams to ternary and encodes them into nucleotides with a rotating dictionary.[2]
- **Anchor BT**: This method corresponds to the transcoder anchor 3 described in Section 3.3, which applies the same entropy coding mechanism as the Anchor BC to DCT coefficients obtained from JPEG bitstreams.
- **JPEG DNA VM (direct transcoding of JPEG files)**: The JPEG bitstreams from the JPEG AIC-3 dataset were directly converted into nucleotides using the JPEG DNA VM bypassing the JPEG XL source encoder.
- **JPEG DNA VM (lossless transcoding of JPEG files with JPEG XL)**: The JPEG bitstreams from the JPEG AIC-3 dataset were first losslessly transcoded to JPEG XL and then converted to nucleotides using the channel encoder of the JPEG DNA VM.

Distortion was evaluated using a set of objective image quality metrics aligned with those listed in the JPEG DNA CTC. Specifically, PSNR, MS-SSIM, IW-SSIM, VIF, NLPD, and PSNR-HVS were computed on the luminance Y channel, FSIM was computed in the RGB color space, and VMAF was computed in the YUV color space. All metrics were computed using the JPEG AI Quality Assessment Framework[†].

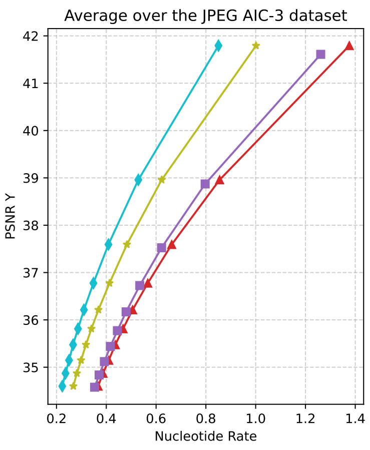
The rate-distortion curves for the *direct encoding setup* are presented in Figure 5. All images in the JPEG AIC-3 dataset were encoded using the target nucleotide rates listed in Table 1, and the plots report the average performance across the entire dataset. In particular, the metric scores and the nucleotide rates have been averaged over all the images in the dataset. The vertical bar presents the 95% confidence intervals computed across the obtained objective quality scores.

It can be observed that the Anchor BC exhibits lower rate-distortion performance compared to both variations of the JPEG DNA VM. In most cases, the confidence intervals do not overlap, indicating statistically significant differences. This suggests that the JPEG DNA VM presents improved performance compared to current state-of-the-art methods designed specifically for image coding.

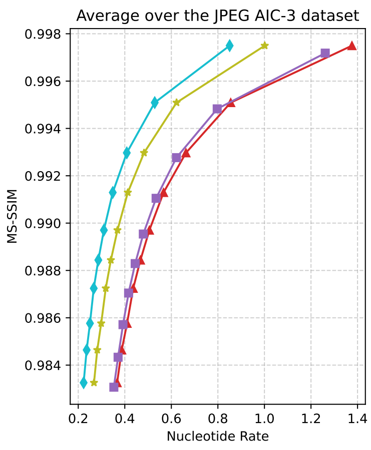
On the other hand, the two methods based on the JPEG DNA VM achieve similar results overall. Notably, the variant using JPEG AI source coding demonstrates superior performance at lower nucleotide rates within the considered range, while the JPEG XL-based variant performs comparably or better at higher nucleotide rates, depending on the objective image quality metric used. For the remaining metrics, both methods perform similarly at higher rates. However, it is important to note that confidence intervals often overlap, suggesting that those differences are not statistically significant. Additionally, the distortion range defined by the JPEG DNA activity targets nearly-visually lossless to visually lossless qualities, and therefore these differences may not be perceptible to the human eye. Consequently, a subjective visual quality assessment is necessary to validate these findings. Overall, the results highlight the flexibility of the JPEG DNA standard: even if JPEG XL is specified

[*] `https://gitlab.com/wg1/jpeg-ai/jpeg-ai-reference-software`
[†] `https://gitlab.com/wg1/jpeg-ai/jpeg-ai-qaf`

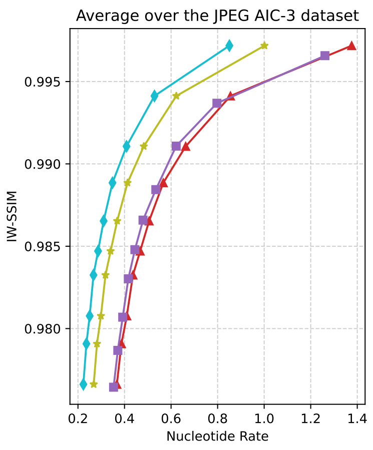
TRANSCODING SETUP



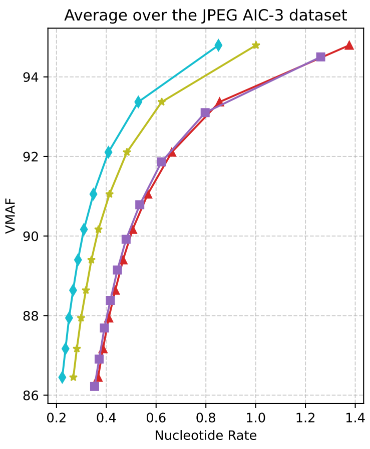
(a) PSNR Y



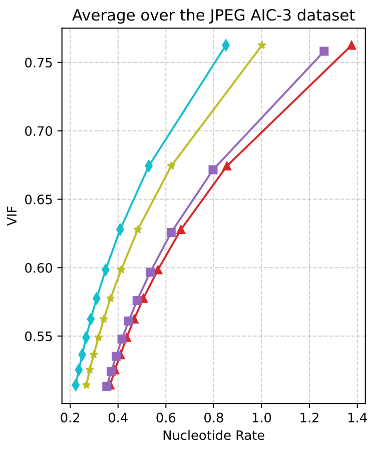
(b) MS-SSIM



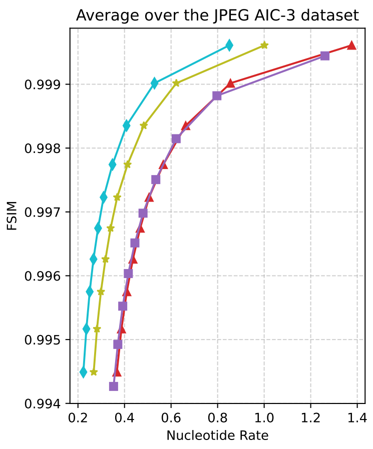
(c) IW-SSIM



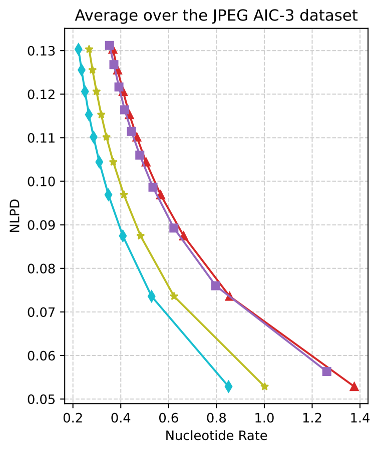
(d) VMAF



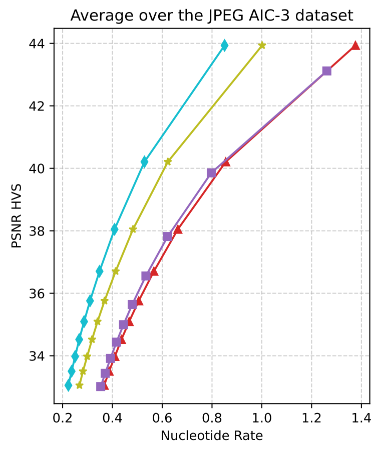
(e) VIF



(f) FSIM



(g) NLPD



(h) PSNR HVS

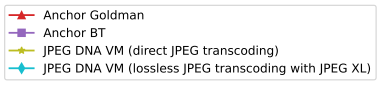

Anchor Goldman
Anchor BT
JPEG DNA VM (direct JPEG transcoding)
JPEG DNA VM (lossless JPEG transcoding with JPEG XL)

Figure 6: Rate-distortion curves for the *transcoding* setup for all objective metrics considered in this study, computed using average nucleotide rates and average metric values across all images in the JPEG AIC-3 dataset.

as its default source coding engine, bypassing this module to employ another encoder may be advantageous depending on the specific requirements of the target application. Moreover, while the coding rate of the source encoder is variable, the DNA Raptor channel encoder is observed to keep the information density at a stable level, even if small variations were observed between different files. The average ratio between the number of bits in the input bitstream and the number of nucleotides in the encoded representation is found to be 1.83 bits/nt, with a standard deviation of 0.0015 bits/nt. This value can be expected to be maintained when arbitrary binary data is encoded with the JPEG DNA VM.

Figure 6 presents the results for the *transcoding setup*. In this configuration, the JPEG-encoded images from the JPEG AIC-3 dataset were adopted. The plots display the average objective metric values and nucleotide rates computed across the entire dataset.

Results indicate that the two anchor methods exhibit similar overall performance. This behavior is expected, since, while Anchor Goldman performs direct transcoding of the JPEG bitstream by converting each byte into a ternary representation and subsequently applying the Goldman codec, Anchor BT instead parses the quantized DCT coefficients from the JPEG file and encodes them into DNA using the same procedure adopted by Anchor BC. Despite these methodological differences, both approaches ultimately preserve the same quantized transform coefficients, which carry the essential visual information, leading to comparable rate-distortion performance across the two methods.

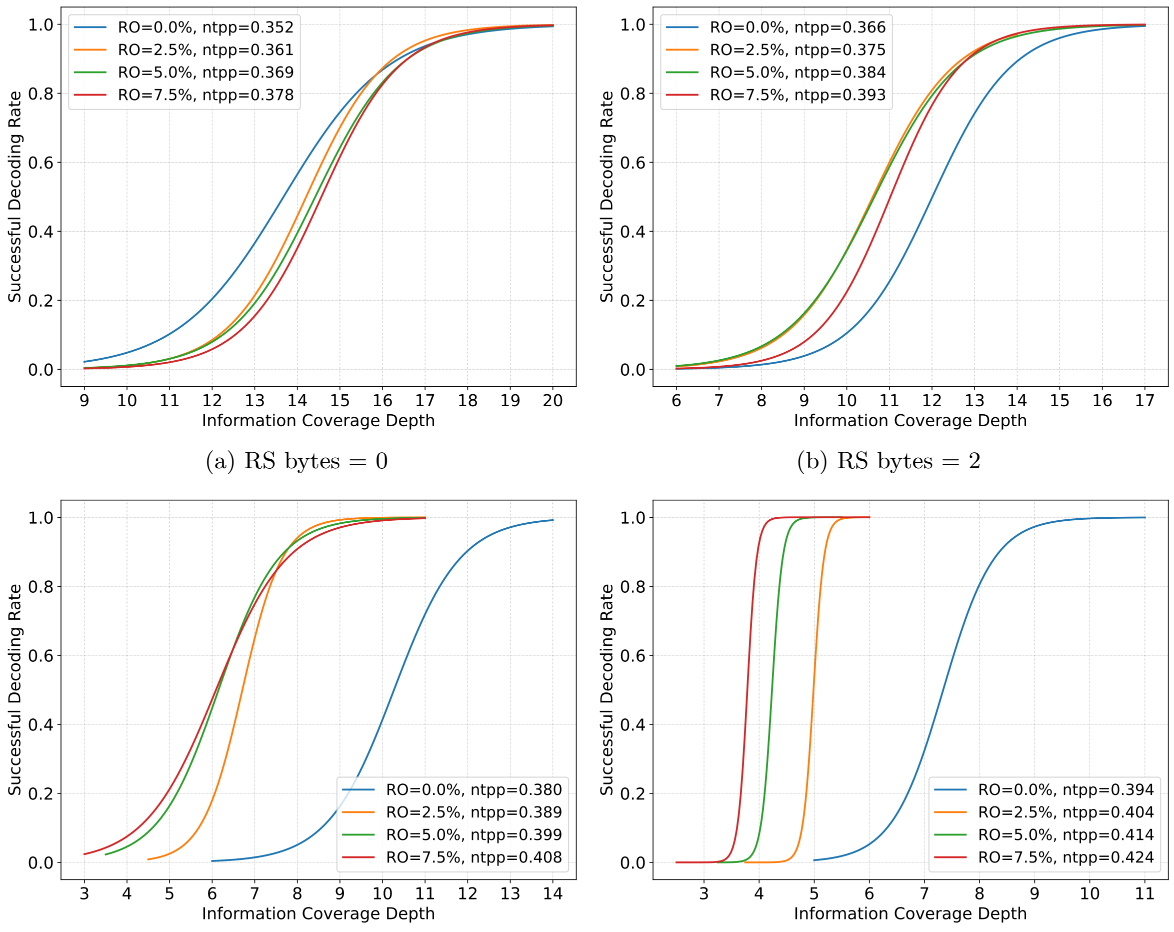
The performance of both anchor methods are nevertheless found to be less effective than the two JPEG DNA VM-based transcoding approaches. In particular, the direct transcoding method of JPEG files to quaternary format presented in the JPEG DNA VM presents enhanced performance compared to the methods based on Goldman, highlighting the effectiveness of the DNA Raptor channel coder regardless of the underlying source coding engine. Furthermore, performing a lossless transcoding from JPEG files to JPEG XL files followed by conversion to DNA using the JPEG DNA VM enables even greater reductions in nucleotide rates while maintaining matching visual quality. In fact, JPEG XL lossless transcoding allows, as the name suggests, a reduction in the file size without loss of visual information.

6.2 Performance in noisy channel

The evaluation of the JPEG DNA VM in noisy channel conditions included the use of RS codes as the binary error-correcting component, as illustrated in Figures 3 and 4. The considered channel model is motivated by practical aspects of DNA data storage, where the sequencer outputs multiple noisy reads of the encoded oligos in arbitrary order. Given a FASTA file containing a collection of N'_T oligos produced by the VM encoder, the channel generates a total of $R_{\text{all}} > N'_T$ noisy reads, each corresponding to one of the N'_T oligos selected uniformly at random with replacement. In addition, each read is affected by substitution errors, where each nucleotide is substituted, with probability ϵ_{err}, by one of the three other nucleotides chosen uniformly at random. The output of the DNA channel is therefore a FASTA file containing R_{all} unordered, noisy oligos, which is in line with the typical output of a DNA sequencer. The analysis in this section focuses on independent substitution errors with rate $\epsilon_{\text{err}} = 1\%$, consistent with recent experimental findings in which substitutions were observed to occur at this rate and independently across nucleotides.[46] The analysis of other types of errors, such as deletions and insertions, using suitable error-correcting codes designed for these errors,[9,10,45,47] is deferred to future work.

The FASTA file at the output of this channel cannot be directly handled by the JPEG DNA VM decoder and requires pre-processing. The decoding workflow considered here thus involves two additional steps prior to applying the JPEG DNA VM decoder: *clustering* and *consensus*. First, the noisy reads in the FASTA output of the channel are clustered to group together reads originating from the same oligo. Then, a consensus sequence is derived for each cluster. The resulting collection of consensus sequences is subsequently passed to the JPEG DNA VM decoder. For clustering, reads sharing the same SBN and ESI (16 nucleotides in total) are grouped together. For consensus, a simple position-wise majority vote over nucleotides is applied across the reads within each cluster. While clustering and consensus improve the quality of the reconstructed oligos, these processes are not error-free. Residual errors resulting from these steps are addressed by the JPEG DNA VM decoder through the error correction capabilities provided by the RS and Raptor codes.

In the absence of added redundancy from the RS and Raptor codes, the number of oligos in the FASTA file after JPEG DNA VM encoding is denoted by N_T, referred to henceforth as the number of *information*



(a) RS bytes = 0

(b) RS bytes = 2

(c) RS bytes = 4

(d) RS bytes = 6

Figure 7: Decoding success rate for image 00005_560x888 from the JPEG AIC-3 dataset as a function of the information coverage depth (R_{all}/N_T), for different RS byte counts and Raptor overheads (RO), under a fixed substitution rate of $\epsilon_{\text{err}} = 1\%$. The image is encoded using the JPEG DNA VM with JPEG XL source coding at quality 71, resulting in $N_T = 875$ information oligos, consistent with the rate $r7 = 0.35$ in Table 1. Each (RS, RO) configuration yields a joint source–channel coding rate above 0.35, with exact values shown in the legend.

oligos. A Raptor overhead of θ increases this number to N'_T approximately equal to $(1+\theta)N_T$, reflecting the addition of redundant oligos to recover lost ones (erasures). Furthermore, each RS byte appended to an oligo introduces 4 extra nucleotides, providing redundancy to correct nucleotide errors (substitutions) within each oligo. The decoding performance of the JPEG DNA VM under the considered channel was evaluated in terms of the decoding success rate. This metric is computed as the average, over independent channel realizations, of a binary indicator that equals one when the bitstream generated by the DNA Raptor decoder exactly matches the one originally generated by the JPEG XL encoder. The success rate depends on several factors, including channel parameters such as the substitution error rate ϵ_{err} and the number of reads R_{all}, as well as the redundancy introduced by the RS and Raptor codes. The general trade-offs between error-correction and channel parameters, and their impact on decoding success, were analyzed from a theoretical perspective in recent work.[48]

Simulations were conducted across a range of values for RS byte count (0, 2, 4, 6), Raptor overhead (0%, 2.5%, 5%, 7.5%), and information coverage depth R_{all}/N_T, where the latter defines the average number of reads per information oligo. For each configuration, the full encoding–decoding pipeline (including clustering and consensus) was executed over 50 independent channel realizations. The empirical decoding success rate was then computed and fitted using a logistic regression model to characterize the decoding performance. Figure 7 shows results for image 00005_560x888 from the JPEG AIC-3 dataset, encoded using the VM with JPEG XL source coding at quality 71. This setting yields $N_T = 875$ information oligos, aligning with the baseline source coding rate $r7 = 0.35$ listed in Table 1. Adding Raptor overhead or RS bytes increases the joint source–channel coding rate above the 0.35 ntpp baseline due to the extra nucleotides introduced by the channel codes.

The results in Figure 7 show that under the considered noisy channel, the JPEG DNA VM can achieve successful decoding with high probability, provided that the total number of reads, reflected through the information coverage depth, is sufficiently large. This outcome is expected, as multiple reads introduce redundancy analogous to repetition coding, which can be leveraged during clustering and consensus to reduce substitution errors via majority voting. The required coverage depth for reliable decoding strongly depends on the channel coding parameters. Without any channel coding redundancy (i.e., 0 RS bytes and 0% Raptor overhead), reliable decoding requires an information coverage depth of approximately 20. Introducing RS coding significantly reduces this requirement: for RS byte counts of 2, 4, and 6, the required coverage depth drops to around 16, 10, and below 5, respectively, depending on the Raptor overhead. Notably, the impact of Raptor overhead in reducing coverage requirements becomes more pronounced as the number of RS bytes increases and individual oligos become more reliable. This reduction in coverage is important in practice, as it translates to lower sequencing costs and faster data retrieval. Similar simulations conducted on other images yielded comparable results regarding the required coverage depth for successful decoding. This consistency is intuitive since the channel generates reads and introduces substitution errors independently of the oligo content; thus, the average number of reads per information oligo required for reliable decoding remains fairly stable across different images.

7. CONCLUSION

This paper provides an overview of the JPEG DNA standard for encoding images into synthetic DNA. In particular, the paper introduces a historical outline of the activities conducted within the scope of the CfP and presents a comprehensive analysis of the JPEG DNA VM's performance in comparison to state-of-the-art methods. The results show that the JPEG DNA VM consistently outperforms existing approaches in terms of rate-distortion efficiency. A major benefit of the standard is its flexibility, allowing for the integration of different source coding methods tailored to specific applications, as well as the efficient transcoding of existing files. Additionally, the error-correcting capabilities of the JPEG DNA VM were thoroughly evaluated, with simulated results showing robust decoding rates at a coverage of 20 with no added redundancy as well as a reduction in the required coverage when Reed-Solomon codes are used. Future work will focus on conducting wet-lab experiments to validate the standard's effectiveness under real-world conditions, as well as developing additional components of the standard, including the integration of different and more advanced error-correcting codes.

ACKNOWLEDGMENTS

EPFL-affiliated authors would like to thank the Swiss National Foundation for Scientific Research (SNSF) under grant number 200020_207918 for funding this research. I3S-affiliated and EURECOM-affiliated authors would like to acknowledge the support of the project PEPR MolecularXiv (ANR-22-PEXM-003) financed by France 2030.

REFERENCES

[1] Church, G. M., Gao, Y., and Kosuri, S., "Next-generation digital information storage in DNA," *Science* **337**(6102), 1628–1628 (2012).

[2] Goldman, N., Bertone, P., Chen, S., Dessimoz, C., LeProust, E. M., Sipos, B., and Birney, E., "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature* **494**(7435), 77–80 (2013).

[3] ISO/IEC JTC 1/SC29/N100252, "Use Cases and Requirements for DNA-based Media Storage version 1.0." https://jpeg.org/jpegdna/documentation.html.

[4] ISO/IEC JTC1/SC29/WG1 N100476, "Final call for proposals on digital media storage on DNA support." https://jpeg.org/jpegdna/documentation.html.

[5] Grass, R. N., Heckel, R., Puddu, M., Paunescu, D., and Stark, W. J., "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Angewandte Chemie International Edition* **54**(8), 2552–2555 (2015).

[6] Erlich, Y. and Zielinski, D., "DNA fountain enables a robust and efficient storage architecture," *science* **355**(6328), 950–954 (2017).

[7] Blawat, M., Gaedke, K., Huetter, I., Chen, X.-M., Turczyk, B., Inverso, S., Pruitt, B. W., and Church, G. M., "Forward error correction for DNA data storage," *Procedia Computer Science* **80**, 1011–1022 (2016).

[8] Yazdi, S. M. H. T. et al., "Portable and error-free DNA-based data storage," *Nature* (2017).

[9] Welzel, M., Schwarz, P. M., Löchel, H. F., Kabdullayeva, T., Clemens, S., Becker, A., Freisleben, B., and Heider, D., "DNA-aeon provides flexible arithmetic coding for constraint adherence and error correction in DNA storage," *Nature Communications* (2023).

[10] Press, W. H., Hawkins, J. A., Jones Jr, S. K., Schaub, J. M., and Finkelstein, I. J., "HEDGES error-correcting code for DNA storage corrects indels and allows sequence constraints," *Proceedings of the National Academy of Sciences* **117**(31), 18489–18496 (2020).

[11] Cao, B., Wang, B., and Zhang, Q., "GCNSA: DNA storage encoding with a graph convolutional network and self-attention," *iScience* **26**(3), 106231 (2023).

[12] Organick, L. et al., "Random access in large-scale DNA data storage," *Nature Biotechnology* (2018).

[13] Marinelli, E. et al., "CMOSS: A reliable, motif-based columnar molecular storage system," *SYSTOR '24: 17th ACM International Systems and Storage Conference* (2024).

[14] Heinis, T. et al., "Survey of information encoding techniques for DNA," *ACM Comput. Surv.* **56** (2023).

[15] Dimopoulou, M., Antonini, M., Barbry, P., and Appuswamy, R., "A biologically constrained encoding solution for long-term storage of images onto synthetic DNA," in [*2019 27th European Signal Processing Conference (EUSIPCO)*], 1–5, IEEE (2019).

[16] Dimopoulou, M. and Antonini, M., "Image storage in DNA using vector quantization," in [*2020 28th European Signal Processing Conference (EUSIPCO)*], 516–520, IEEE.

[17] Dimopoulou, M., San Antonio, E. G., and Antonini, M., "A JPEG-based image coding solution for data storage on DNA," in [*2021 29th European Signal Processing Conference (EUSIPCO)*], 786–790, IEEE (2021).

[18] Secilmis, L., Testolina, M., Lazzarotto, D., and Ebrahimi, T., "Towards effective visual information storage on DNA support," in [*Applications of Digital Image Processing XLV*], **12226**, 29–35, SPIE (2022).

[19] Pic, X. and Antonini, M., "A constrained shannon-fano entropy coder for image storage in synthetic DNA," in [*2022 30th European Signal Processing Conference (EUSIPCO)*], 1367–1371, IEEE (2022).

[20] Pic, X., Dimopoulou, M., Antonio, E. G. S., and Antonini, M., "MQ-Coder inspired arithmetic coder for synthetic DNA data storage," *2023 IEEE International Conference on Image Processing (ICIP)* (2023).

[21] Lazzarotto, D., Ramos, J. E., Testolina, M., and Ebrahimi, T., "Technical description of the EPFL submission to the JPEG DNA CfP," *arXiv preprint arXiv:2312.00560* (2023).

[22] Pic, X., Gil San Antonio, E., Dimopoulou, M., and Antonini, M., "Image storage on synthetic DNA using compressive autoencoders and DNA-adapted entropy coders," *2023 IEEE 25th International Workshop on Multimedia Signal Processing (MMSP)* (2022).

[23] Strebel, S., Monnier, N., Lazzarotto, D., Testolina, M., and Ebrahimi, T., "Towards learning-based image compression for storage on DNA support," in [*Applications of Digital Image Processing XLVI*], (2023).

[24] Franzese, G. et al., "Generative DNA: Representation learning for DNA-based approximate image storage," 01–05 (2021).

[25] Le, T. H., Pic, X., Mateos, J., and Antonini, M., "Implicit neural multiple description for DNA-based data storage," *2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2023).

[26] Seo, S. et al., "Information density enhancement using lossy compression in DNA data storage.," (2024).

[27] Ballé, J., Laparra, V., and Simoncelli, E. P., "End-to-end optimized image compression," in [*International Conference on Learning Representations*], (2016).

[28] Theis, L., Shi, W., Cunningham, A., and Huszár, F., "Lossy image compression with compressive autoencoders," *International Conference on Learning Representations* (2017).

[29] Minnen, D., Ballé, J., and Toderici, G., "Joint autoregressive and hierarchical priors for learned image compression," 10794–10803 (2018).

[30] Ascenso, J., Akyazi, P., Pereira, F., and Ebrahimi, T., "Learning-based image coding: early solutions reviewing and subjective quality evaluation," in [*Optics, Photonics and Digital Technologies for Imaging Applications VI*], Schelkens, P. and Kozacki, T., eds., **11353**, 113530S, International Society for Optics and Photonics, SPIE (2020).

[31] Ladune, T., Philippe, P., Henry, F., Clare, G., and Leguay, T., "COOL-CHIC: Coordinate-based low complexity hierarchical image codec," (2023).

[32] Kim, H., Bauer, M., Theis, L., Schwarz, J. R., and Dupont, E., "C3: High-performance and low-complexity neural compression from a single image or video," *preprint arxiv* (2023).

[33] Couvreur, C., Testolina, M., Ladune, T., Lorand, G., Philippe, P., and Antonini, M., "Benchmarking conventional and learning-based codecs for DNA image storage," in [*2025 Picture Coding Symposium (PCS)*], IEEE (2025). Manuscript submitted for publication.

[34] ISO/IEC JTC 1/SC29/WG1 N100517, "JPEG DNA Common Test Conditions version 2.0." https://jpeg.org/jpegdna/documentation.html.

[35] Wang, Z., Simoncelli, E. P., and Bovik, A. C., "Multiscale structural similarity for image quality assessment," in [*The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*], **2**, 1398–1402, Ieee (2003).

[36] Wang, Z. and Li, Q., "Information content weighting for perceptual image quality assessment," *IEEE Transactions on image processing* **20**(5), 1185–1198 (2010).

[37] Li, Z., Bampis, C., Novak, J., Aaron, A., Swanson, K., Moorthy, A., and De Cock, J., "Vmaf: The journey continues." https://netflixtechblog.com/vmaf-the-journey-continues-44b51ee9ed12 (2018). Accessed: 2025-07-29.

[38] Sheikh, H. R. and Bovik, A. C., "Image information and visual quality," *IEEE Transactions on image processing* **15**(2), 430–444 (2006).

[39] Zhang, L., Zhang, L., Mou, X., and Zhang, D., "FSIM: A feature similarity index for image quality assessment," *IEEE transactions on Image Processing* **20**(8), 2378–2386 (2011).

[40] Laparra, V., Ballé, J., Berardino, A., and Simoncelli, E. P., "Perceptual image quality assessment using a normalized laplacian pyramid," *Electronic Imaging* **28**, 1–6 (2016).

[41] Ponomarenko, N., Silvestri, F., Egiazarian, K., Carli, M., Astola, J., and Lukin, V., "On between-coefficient contrast masking of DCT basis functions," in [*Proceedings of the third international workshop on video processing and quality metrics*], **4**, Scottsdale USA (2007).

[42] Luby, M., Shokrollahi, A., Watson, M., and Stockhammer, T., "Raptor forward error correction scheme for object delivery," tech. rep. (2007).

[43] Schwarz, P. M. and Freisleben, B., "NOREC4DNA: using near-optimal rateless erasure codes for DNA storage," *BMC bioinformatics* **22**(1), 1–28 (2021).

[44] Lazzarotto, D., Testolina, M., and Ebrahimi, T., "Error correction for DNA-based image storage," in [*2025 IEEE International Conference on Image Processing (ICIP)*], IEEE (2025). Manuscript accepted for publication.

[45] Lazzarotto, D., Piguet, F., and Ebrahimi, T., "DNA-based image storage with substitution and indel error correction," in [*Picture Coding Symposium 2025*], IEEE (2025). Manuscript submitted for publication.

[46] Gimpel, A. L., Stark, W. J., Heckel, R., and Grass, R. N., "A digital twin for dna data storage based on comprehensive quantification of errors and biases," *Nature Communications* **14**(1), 6026 (2023).

[47] Kas Hanna, S., "Short systematic codes for correcting random edit errors in DNA storage," in [*2024 IEEE International Symposium on Information Theory (ISIT)*], 663–668 (2024). Extended version available at: https://arxiv.org/abs/2402.01244.

[48] Kas Hanna, S., "On the reliability of information retrieval from MDS coded data in DNA storage," in [*2025 IEEE International Symposium on Information Theory (ISIT)*], (2025). Manuscript accepted of publication; extended version available at: https://arxiv.org/abs/2502.06618.