Encoding techniques for long-term storage of digital images into synthetic DNA

Dimopoulou Melpomeni

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

Techniques de codage pour le stockage à long terme d'images numériques dans l'ADN synthétique

Melpomeni Panagiota DIMOPOULOU

Laboratoire d'Informatique, Signaux et Systèmes de Sophia-Antipolis (I3S)

Présenté en vue de l'obtention du grade docteur en Automatique Traitement du Signal et des Images **d'**Université Côte d'Azur
Dirigé par : Marc Antonini
Soutenue le : 04/12/2020

Devant le jury, composé de :
Marc ANTONINI, Directeur de Recherche, CNRS
Raja APPUSWAMY, Professeur Assistant, EURECOM
Pascal BARBRY, Directeur de Recherche, CNRS
Frédéric DUFAUX, Directeur de Recherche, CNRS
Touradj EBRAHIMI, Professeur, EPFL
Christine GUILLEMOT, Directrice de Recherche, INRIA
Thomas HEINIS, Professeur Assistant, Imperial College London
Emily LEPROUST, Directrice Generale, Twist Bioscience

Techniques de codage pour le stockage à long terme d'images numériques dans l'ADN synthétique

---

Jury :

Rapporteurs
Christine GUILLEMOT, Directrice de Recherche, Institut National de Recherche en Informatique et en Automatique (INRIA)
Thomas HEINIS, Professeur Assistant, Imperial College London

Examinateurs
Touradj EBRAHIMI, Professeur, École Polytechnique Fédérale de Lausanne (EPFL)
Frédéric DUFAUX, Directeur de Recherche, CNRS, Université Paris-Sud
Emily LEPROUST, Directrice Generale, Twist Bioscience
Pascal BARBRY, Directeur de Recherche, CNRS, Université Côte d'Azur
Raja APPUSWAMY, Professeur Assistant, EURECOM
Marc ANTONINI, Directeur de Recherche, CNRS, Université Côte d'Azur (Directeur de Thèse)

Techniques de codage pour le stockage à long terme d'images numériques dans l'ADN synthétique

---

Résumé

L'explosion de la quantité de données est l'un des plus grands défis de l'évolution numérique, entraînant une croissance de la demande de stockage à un rythme tel qu'elle ne peut pas rivaliser avec les capacités réelles des périphériques. L'univers numérique devrait atteindre plus de 175 zettaoctets d'ici 2025, tandis que le 80% de ces données est rarement consultée (données froides), mais archivée sur des bandes magnétiques pour des raisons de sécurité et de conformité réglementaire. Les dispositifs de stockage conventionnels ont une durée de vie limitée de 10 à 20 ans et doivent donc être fréquemment remplacés pour garantir la fiabilité des données, un processus qui est coûteux en termes d'argent et d'énergie. L'ADN est un candidat très prometteur pour l'archivage à long terme de données «froides» pendant des siècles voire plus à condition que l'information soit encodée dans un flux quaternaire constitué des symboles A, T, C, G, pour représenter les 4 composants de la molécule d'ADN, tout en respectant certaines contraintes d'encodage importantes. Dans cette thèse, nous présentons de nouvelles techniques de codage pour le stockage efficace d'images numériques dans l'ADN. Nous avons implémenté un nouvel algorithme de longueur fixe pour la construction d'un code quaternaire robuste qui respecte les contraintes biologiques et proposé deux fonctions de "mapping" différentes pour permettre une flexibilité par rapport aux besoins d'encodage. De plus, l'un des principaux défis du stockage des données dans l'ADN étant le coût élevé de la synthèse, nous faisons une toute première tentative pour introduire une compression contrôlée dans la solution de codage proposée. Le codec proposé est compétitif par rapport à l'état de l'art. En outre, notre solution de codage / décodage de bout en bout a été expérimentée dans une expérience de laboratoire humide pour prouver la faisabilité de l'étude théorique dans la pratique.

Mots clés: Stockage de données dans l'ADN, Codage d'images, ADN synthetique, Compression d'images, Codage robuste

Encoding techniques for long-term storage of digital images into synthetic DNA

Abstract

Data explosion is one of the greatest challenges of digital evolution, causing the storage demand to grow at such a rate that it cannot compete with the actual capabilities of devices. The digital universe is forecast to grow to over 175 zettabytes by 2025 while 80% is infrequently accessed ("cold" data), yet safely archived in off-line tape drives due to security and regulatory compliance reasons. At the same time, conventional storage devices have a limited lifespan of 10 to 20 years and therefore should be frequently replaced to ensure data reliability, a process which is expensive both in terms of money and energy. Recent studies have shown that due to its biological properties, DNA is a very promising candidate for the long-term archiving of "cold" digital data for centuries or even longer under the condition that the information is encoded in a quaternary stream made up of the symbols A, T, C and G, to represent the 4 components of the DNA molecule, while also respecting some important encoding constraints. Pioneering works have proposed different algorithms for DNA coding leaving room for further improvement. In this thesis we present some novel image coding techniques for the efficient storage of digital images into DNA. We implemented a novel fixed length algorithm for the construction of a robust quaternary code that respects the biological constraints and proposed two different mapping functions to allow flexibility according to the encoding needs. Furthermore, one of the main challenges of DNA data storage being the expensive cost of DNA synthesis, we make a very first attempt to introduce controlled compression in the proposed encoding workflow. The, proposed codec is competitive compared to the state of the art. Furthermore, our end-to-end coding/decoding solution has been experimented in a wet lab experiment to prove feasibility of the theoretical study in practice.

Keywords: DNA data storage, Image coding, Synthetic DNA, Image compression, Robust encoding

"The true sign of intelligence is not knowledge, but imagination..."

Albert Einstein

Acknowledgements

This thesis has been a great adventure for me and has helped me broaden my horizons and get to know the main aspects of research. However, nothing would have been possible without the help and guidance of my supervisor Dr. Marc Antonini who has been a great advisor, always present to guide me in any moment of doubt, encouraging me to take initiatives and feel free to express my ideas while also enlightening me with all his knowledge and experience! He has been my mentor and I couldn't be more grateful to him as anything I have achieved in the field of research is due to his constant advice and the fact that he always believed in me, sometimes even more than I did myself!

Then, I would also like to thank Christine Guillemot and Thomas Heinis for accepting to review my thesis manuscript despite the amount of work this might require. I am also grateful to Touradj Ebrahimi, Emily Leproust and Frederic Dufaux who have accepted to take part in my PhD jury. I would like to separately express my gratitude to Raja Appuswamy and Pascal Barbry for their helpful comments and advise not only as members of my PhD jury but through-out our collaboration during the past three years.

Furthermore, I would like to thank all my colleagues in the lab (new ones and old ones) for their feedback, cooperation and of course friendship. But I would like to separately mention my closest friends in the lab: Somia, Vasilina, Diana, Ninad, Cyprien, Arnaud, Gaël and Froso, thank you for all the times you were there for me to celebrate the happy moments and to encourage me during the most difficult times. A very big and special "Thank you" to my dear friend and academic alter-ego, Eva, with whom I have worked together for the last two years and she has supported me through all the difficult research moments.

In addition, I would like to express my sincere appreciation to the staff of the administration for their help and especially Nadia, our mother in the lab, for all her kindness and the last-minute favors.

At this point, I would also like to express my deep gratitude to my professors from the Computer Engineering and Informatics Department in the University of Patras in Greece, who played an important role in the construction of a strong core of basic knowledge which has been the most helpful tool in the pursue of my academic career. More precisely, I would like to thank Emmanouil Psarakis, my bachelor thesis supervisor, for guiding me, inspiring me and motivating me to explore the field of Signal and Image processing. Then, I would also like to thank Kostas Berberidis for all the knowledge, fruitful discussions and his valuable advice when I was in doubt. And last but not least, I would like to thank the person to whom I owe the beginning of this wonderful journey in France, Evi Papaioannou, for all the help, the support and encouragement to accept the challenge and take this first step in following my dreams.

Throughout this exciting adventure, I also had the chance to be surrounded by some very special people who are always there for me even if we rarely meet due to the long distance... my best friends, Maria, Natassa, Penny, Ioanna and Vassilis. Thank you for always making my day with your positive energy and encouragement along these years!

A big and very special "Thank you" goes to my family! First of all, my parents, to whom I owe everything I am today and are always by my side

filled with pride for any of my achievements, supporting all my efforts and constantly providing me with love, strength and courage, as well as my beloved sister Georgina for all her love and her continuous support since I moved to France. Also, I would like to separately thank my oldest, wisest and most wonderful professor I have ever had in my life since I was 6 years old... my grandfather Alekos, to whom I owe the biggest part of any of the knowledge I have today!

Finally, the most special thank you goes to Dimitris, my partner in life, with whom I have been sharing all my thoughts and fears and has always been supporting me through everything! I would like to thank him for his encouragement to the pursue of my dreams even if this required much more strength due to the long-distance of the first 2 years. Thank you for your patience during my complaints and all the moments we share every day.

Contents

List of Figures

List of Tables

List of Abbreviations

AC	**A**lternating **C**urrent
ACD	**A**dvanced **C**orrection **D**ecoding
BA	**B**ridge **A**mplification
DC	**D**irect **C**urrent
DCT	**D**iscrete **C**osine **T**ransform
DNA	**D**eoxyribo**N**ucleic **A**cid
DPCM	**D**ifferential **P**ulse **C**ode **M**odulation
IDC	**I**nternational **D**ata **C**orporation
DWT	**D**iscrete **W**avelet **T**ransform
JPEG	**J**oint **P**hotographic **E**xperts **G**roup
MSE	**M**ean **S**quare **E**rror
NGS	**N**ext **G**eneration **S**equencing
nt	**n**ucleo**t**ide(s)
PCR	**P**olymerase **C**hain **R**eaction
PSNR	**P**eak **S**ignal to **N**oise **R**atio
SBS	**S**equencing **B**y **S**ynthesis
SCD	**S**imple **C**orrection **D**ecoding
SQ	**S**calar **Q**uatization
VQ	**V**ector **Q**uantization
VLC	**V**ariable **L**ength **C**oding

List of Symbols

a_{sb}	Fraction of total pixels of a subband sb
$\mathcal{B}$	Set of barcodes
$\mathcal{C}$	Set of all possible quaternary words composed by the symbols $\{A, T, C, G\}$
$\mathcal{C}^*$	Constrained codebook provided by PAIRCODE
$\mathcal{C}_u$	Set of decodable codewords in the 1-ring neighborhood (space domain) of an undecodable word
c_e	Erroneous codeword
D	distortion
$\mathcal{D}$	Set of all possible viable quaternary words composed by the symbols $\{A, T, C, G\}$
d_E	Euclidean distance
d_H	Hamming distance
d_L	Levenshtein distance
d_{SL}	Sequence-Levenshtein distance
$F(v)$	empirical function used to compute the density of the neighborhood of a vector v
$H(c_k)$	Hamming sphere of c_k containing words with Hamming distance of 1 to c_k.
K	number of indices in the codebook
L	number of codewords in the code
l	length of codewords in the code
$\#\ell$	number of levels of DWT
m	number of different codewords from code assigned to each source symbol
n	length of quantization vectors in VQ
P_R	Matrix of all possible permutations of subband rates providing a target total rate
P_T	Matrix of all possible permutations of subband rates
$p(u)$	probability of a vector v
q	Quantization step size of Uniform Scalar Quantizer
R	Rate
$S(v_k)$	Neighborhood containing the closest vectors around vector v_k
SB	Number of subbands
$\mathcal{V}$	Set of input source symbols
$\mathcal{V}_{H_u}$	Set of vectors to which are mapped the codewords belonging to
$\mathcal{V}_u$	Set of vectors to which are mapped the codewords belonging to the neighborhood around an undecodable word
v_u	Vectors belonging in the set V_{H_u} of vectors to which are mapped codewords from the same Hamming sphere H_u.
$\mathcal{W}$	Set of undecodable codewords
w_u	undecodable word
$\alpha(v_k)$	Function that provides the index k of v_k
β	trade-off parameter
Γ	Mapping function
λ	Lagrange multiplier (slope of the R-D curves)
μ	number of errors

ν_u	Vectors belonging in the set V_u of vectors to which are mapped the codewords from the same neighborhood $\mathcal{C}_u$ around some undecodable word w_u.
$\phi(v)$	Summation of euclidean distances between a vector v and all vectors belonging to its neighborhood
Σ	Set of indices
ω_{sb}	weights of non-orthogonality used in source allocation

To my parents ...

Chapter 1

Introduction

1.1 Digital data storage

The problem of data explosion constitutes nowadays one of the greatest challenges of digital evolution. The continuous greedy use of the internet, including digital platforms and social networks is leading to an immersive increase in the generation of digital data which needs to be handled and stored efficiently. This information overload is massively stored in the servers of big data centers where it is being organized and archived using different technologies according to the data's nature and demand. Data centers contain a set of routers and switches that transport traffic between the servers and to the outside world. To better understand the organization of the data into server systems, figure 1.1 illustrates the various parts of current memory and storage units.

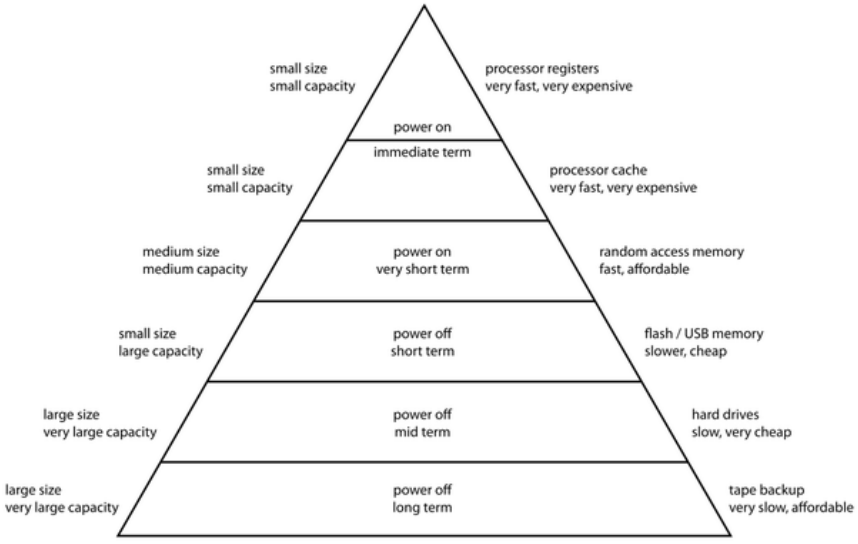


Figure 1.1: Hierarchy of data

Data storage can be hierarchically divided in 6 different layers.

- The top layer consists of small-size processor registers which handle the data which is accessed very frequently in immediate term and therefore needs to be fast and efficient. This type of storage is power-on and small in capacity.
- The second layer contains cache processors which are acting as a temporary storage area where the processors of the top layer can retrieve

data from easily. This layer is also working power-on and it is small in capacity and fast in access time.

- The third layer is the last power-on part of the pyramid. It consists of the Random Access Memory (RAM) which contains short term data that can be read in any order, to retrieve information needed by the cache processors of the previous layer. RAM memory is medium capacity and fast in access time.
- The next layer consists of the flash storage which can work off-line and contains short term data. It is large in capacity but slower in access time.
- The second power-off layer consists of the hard drives (HDD). It is used for mid term storage and contains data which is less frequently used and therefore it is slower in access time than flash storage. This type of memory has a large size and a large capacity.
- Finally, the last part of the pyramid concerns the data which is very infrequently accessed. This type of data can be stored in tape back-ups which are very slow in access time and very large in capacity.

It is easy to notice that the volume of data which is more frequently accessed is extremely small compared to the data of less frequent demand. To this end, the last two layers of the hierarchy consist of solutions of a large capacity which unfortunately also exhibit a much larger volume.

An article published in Forbes magazine [3] mentions that according to International Data Corporation (IDC), 22 Zettabytes (ZB) of digital storage will be shipped across all storage media types between 2018 and 2025, with nearly 59% of that capacity supplied by the HDD industry. Figure 1.2 depicts the predicted exponential growth in the byte shipment for the different types of storage media. As can be seen, by 2025 there is expected to be significant amounts of digital stored in HDDs, in various solid state storage as well as magnetic tape.

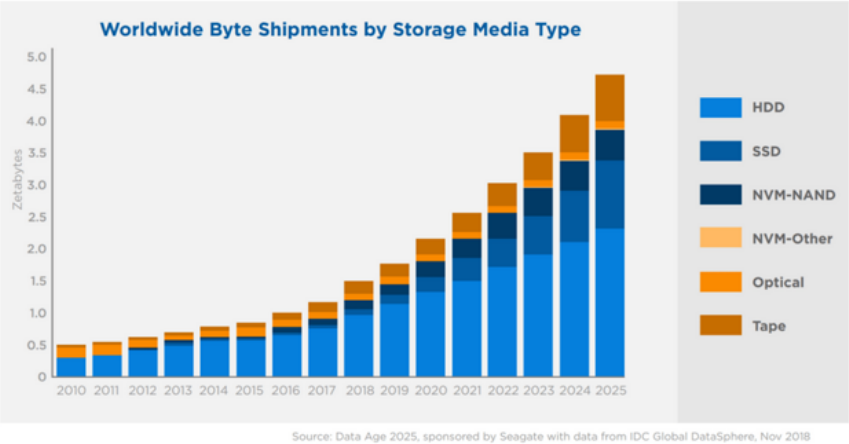


Figure 1.2: Byte Shipment as reported in [3]

It is clear that this exponential growth in the generation of digital data should not be neglected and we are about to face some serious obstacles given the existing storage resources! In the next paragraph we will provide some interesting statistics that clearly expose the impact of this data explosion in many different aspects of our everyday life.

1.1.1 Some statistics

According to an article published in [4], the rapid rise in smartphone usage, IoT adoption, and big data analytics have led to a massive growth in data centers, and they come with a cost. This article presents the following statistics as provided by IDC.

- In 2012 there existed 500.000 data centers to handle global traffic while today there exist more than 8 million.
- The yearly CO2 emissions of data centers reaches the amounts of CO2 produced by the global airline industry.
- Every year, millions of data centers worldwide are draining country-sized amounts of electricity. Several models even predict that data center energy-usage could engulf over 10% of the global electricity supply by 2030 if left unchecked.
- 90% of the existing data have been only generated in the last 2 years.
- The amount of energy used by data centers continues to double every four years.

Along with the above numbers, it is also known that storage media have a limited life-span which varies from 3-5 years for HDD drives and 20-30 years for back-up tape drives. To reassure reliability of the stored data, it is therefore necessary that data centers frequently replace the different storage units, a fact that leads to a huge hardware waste. Furthermore, the replacement of older storage units yields the need for migrating the data into the new units, a process which is expensive both in terms of money and energy. All these facts, reveal that the enormous increase in the generation of data is causing important pollution to the environment. Due to the resulting environmental impact, increased pressure has been placed on companies to follow a green policy by building green or sustainable data centers which utilize energy-efficient technologies.

1.1.2 What is cold data?

For managing, storing and re-purposing digital content, industries and data centers differentiate the data into three levels, hot, warm and cold, based on interest or access priorities. The frequency of data demand (metaphorically called data temperature) denotes the most appropriate unit to which each type of data should be stored. More precisely, hot data refers to assets that require the fastest storage as they are accessed most frequently. It is thus stored in the nearest or closest spots from the accessing points such as solid state or flash drives and CPU. Warm data represents information that is less accessible and is stored on a bigger storage capacity or file servers for relatively cost-efficient concern. Finally, the data which is very rarely or even never accessed and doesn't require on-line workflow is placed on the slowest low-cost options of storage mediums such as tape and optical discs and is termed as cold data.

Figure 1.3 depicts the different levels of data temperature as well as the most appropriate means of storage according to the access frequency. It is

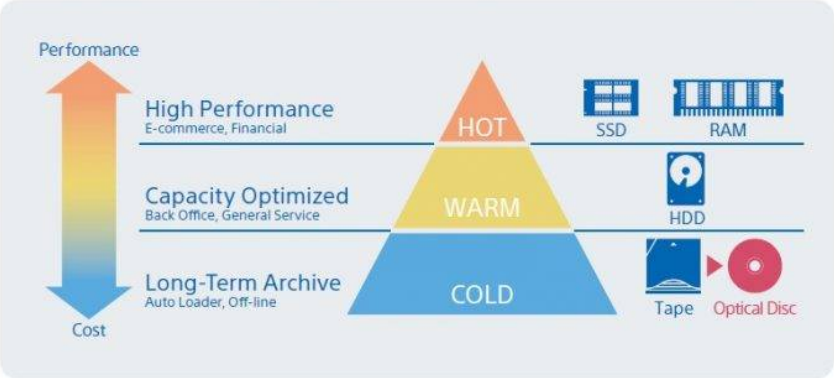


Figure 1.3: The different types of storage according to data's temperature (image taken from [5].)

clear that the largest part of digital information consists of cold data and in spite of its infrequent use, this information must be nevertheless stored in back-up tape drives due to security and regulatory compliance reasons. 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 [6]. Furthermore, as the percentage of cold data has reached the 80% over the last years, it is clear that the total cost for preserving this type of information increases significantly along time!

1.1.3 Problem definition

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. It is therefore necessary to find new resources for the storage of digital data which exhibit higher capacity and longer life-span. Some interesting solutions to this problem are presented in the following paragraph.

1.1.4 Existing solutions

Longevity of data storage is not only important for financial or environmental reasons, but it is also crucial for preserving fundamental and invaluable cultural heritage for next generations. To deal with this problem, scientists have been studying the use of alternative means of higher durability.

Several projects, for instance at the University of Southampton [7] or at Hitachi [8], are currently considering new forms of very long term digital storage, using molding silica glass, which estimated storage length time in the range of 100 million years. 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 other very interesting solution, proposes the use of the DNA molecule which is life's information-storage material, as an alternative approach for digital data storage. Interestingly enough, recent works have proven that storing digital data into DNA is not only feasible but also very promising as the DNA's biological properties allow the storage of a great amount of information into an extraordinary small volume, for centuries or even longer, with no loss of information. This thesis aims to present some novel algorithms and techniques for the storage of digital information into DNA and thus the next sections are dedicated in explaining the term of DNA data storage as also in analysing the most important assets and challenges.

1.2 What is DNA coding?

DNA (deoxyribonucleic acid), is the support of heredity in living organisms. It is a complex molecule corresponding to a succession of four types of nucleotides (nts), Adenine (A), Thymine (T), Guanine (G), Cytosine (C). DNA can be double strand if one single strand binds on a complementary one according to the complementary base pairing rule (Chargaff's rule) [9] which denotes that DNA base pairs are always adenine with thymine (A-T) and cytosine with guanine (C-G). 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.

More specifically, some important advances in the field of synthetic biology have allowed artificial synthesis of DNA strands in a laboratory (in vitro). The produced DNA is synthetic but shares the same extraordinary properties as the real one. The only difference would be the fact that artificial synthesis does not require any particular DNA templates, allowing virtually any quaternary sequence of A, T, C, G to be synthesized in the laboratory. This means that the produced DNA will not necessarily contain any genes, which are DNA sequences responsible for producing life. On the contrary any sequence of nucleotides can be assembled into a DNA strand. Consequently using this technique any digital information can be synthesized into DNA on the condition that it has been previously encoded into a quaternary representation, a process which is called DNA coding. Once synthesized into the form of DNA, the encoded sequence can be retrieved using some special machines, the sequencers. DNA sequencing is the biological process which allows reading any DNA strand and decoding it to provide their quaternary content. Those two fundamental biological processes of DNA synthesis (writing) and sequencing (reading) work similarly to a noisy channel and thus construct an encoding workflow for digital storage.

1.2.1 Advantages

DNA possesses three key properties that make it a very promising candidate for archival storage of digital data.

- 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. The decoding of the DNA of a woolly mammoth that had been trapped into permafrost for 40.000 years [10] is only one example which proves DNA's longevity in contrast to HDD and tape drives which have a life-span of five and twenty years respectively.
- 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.
- Finally, DNA is life's information storage material the main composition of which will never change. This comes in contrast to other means of storage which tend to change over the years according to the technological progress and so do the corresponding decoding devices. This means that on the long term, due to the incompatibility of the stored data with the new decoders, the stored content might not be decodable. For example almost 20 years ago, computers used to have a special input for floppy discs which is no longer the case. Consequently, any information that was stored in floppy disks is no longer accessible. On the contrary, DNA will exist forever in living organisms and even if the methods used for sequencing will further improve, the new sequencers will always be adapted for decoding the exact same molecule.

The above properties reveal that storing digital data into DNA is an extremely promising solution. According to an article published in the journal of "Nature" [11], in a very rough theoretical estimation, scientists claim that 1kg of DNA would be enough for storing all the world's digital information. Figure 1.4 depicts a table taken from [11] comparing the molecule of DNA to some widely used storage devices, the hard disks and flash memories.

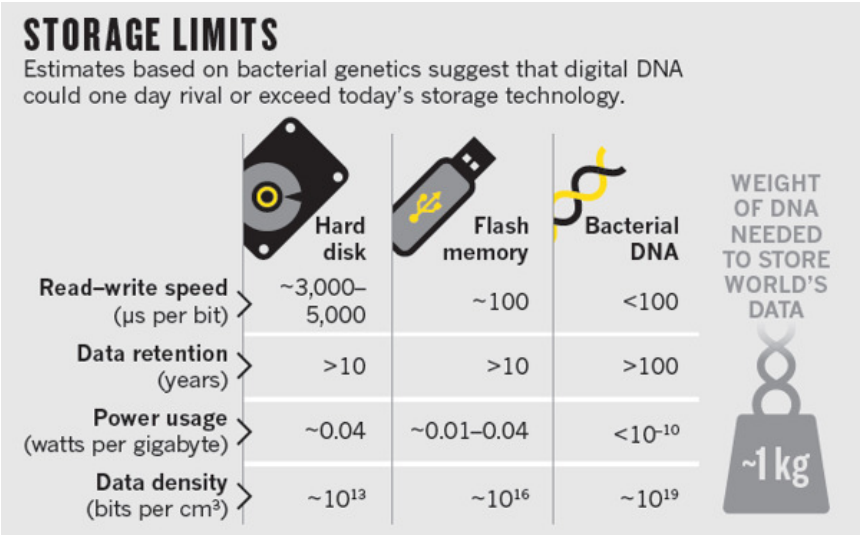


Figure 1.4: Comparison of DNA to other means of digital data storage [11]

1.2.2 The challenges

As described in section 1.2, DNA synthesis and sequencing are the key procedures which allow the archiving of digital data into DNA. While fundamental in the field of biology those two processes introduce some important challenges.

To begin with, DNA synthesis requires the construction of DNA strands (oligos) of no more than 200-300 nts. This restriction stems from the fact that the error of the synthesis increases exponentially with the increase in the length of the oligos and such short oligos have a low error probability. It is thus necessary to cut the encoded quaternary strand into smaller chunks and also yields the need for introducing some special headers to allow correct reconstruction at the decoding.

Secondly, both DNA synthesis and sequencing include some fine and delicate biological manipulations and thus those two processes are expensive and require several dollars per synthesized/sequenced oligo. It is therefore necessary to efficiently compress during the the data to be archived before it is stored into DNA.

Another important drawback rises from the process of DNA sequencing which is prone to errors creating insertions, deletions or substitutions of nucleotides in the decoded sequence. Luckily there are some special rules for the encoded strands which allow reducing the probability of error but unfortunately without eliminating it. Those rules will be described in a later section.

Finally, a last but not negligible challenge lies in the longevity of DNA. While being an important asset which allows storage of digital data for centuries and maybe over it also requires that the know-how of the decoding process should be passed on to the next generations to allow long-term decoding of data that had been stored many years ago. It is therefore very important, to safely preserve this information into durable materials while also ensuring that it is encoded in a way that will be easy for any new user to retrieve and understand. An interesting study on this particularly difficult challenge has been presented in [12]. Another interesting idea could be storing the decoding information in silica glass. Some interesting works for storing information in silica glass have been proposed in [13].

1.3 Outline

The main contribution of this thesis is the introduction of an end-to-end solution for the efficient storage of digital images into synthetic DNA. Since DNA synthesis (writing) is an expensive process that costs several dollars per synthesized strand, state of the art methods have been compressing images using the classical JPEG standard which is optimized for a binary representation and then transcoding the binary output into DNA. Since this is an open-loop solution which is not optimized for DNA's quaternary encoding, in our work we propose for the first time, a workflow which includes the process of image coding to allow controlled compression of the input image. In other words, we provide a "closed-loop" solution which optimally compresses and encodes the input aiming to the reduction and control of the synthesis cost. To this end we introduced two different encoding solutions to

be applied according to the needs of the encoding. We first implemented a fixed-length encoding solution which is optimized due to a source allocation algorithm to provide the best image quality for a given cost. This solution uses a novel constrained fixed-length quaternary code which is robust to sequencing (reading) noise. We then implemented a second variable-length encoding solution which is inspired by the classical JPEG standard to further improve the encoding performance. Finally since the encoded data needs to be synthesized into small DNA fragments which are stored in the same pool, we propose different formatting scenarios for cutting the encoded strands into smaller chunks and introduce the necessary headers for the correct reconstruction. The thesis is organized as follows:

- In this chapter (chapter:1), we have already introduced the main ideas and goals of DNA data storage and we have explained the reasons for which this field of study is expected to flourish in the following years to provide interesting solutions for digital data storage.
- In chapter 2, we provide a generalized workflow for DNA data storage by introducing each of the different sub-processes. We then present the state of the art methods by analysing the methods which have been proposed and explain our contributions compared to the existing solutions.
- In chapter 3, we present the algorithm for constructing a novel fixed length quaternary code which has been used in our proposed encoding methods. To evaluate this quaternary code we analyse its assets and weaknesses compared to state of the art algorithms for the construction of DNA codes. We also provide two different algorithms for mapping the input data to the codewords of the proposed constrained quaternary code. The first mapping algorithm exploits the code's redundancy to robustify the encoded data while the second one creates an encoding which is resistant to noise.
- The above methods have been used in chapter 4 to construct a fixed-length image coding solution which uses a source allocation algorithm to allow controlled compression.
- In chapter 5, we introduce a new variable-length encoding solution using the main workflow of the classical JPEG standard while modifying it to produce an optimized quaternary encoding. For this last encoding case we combined our code construction algorithm proposed in chapter 3 with another variable-length one proposed in the bibliography to produce an efficient entropy-coding model.
- In chapter 6, we propose multiple formatting scenarios adapted to the needs of each proposed encoding solution, for creating decodable DNA chunks than contain all the necessary header information for the decoding. Since the formatting headers contain fundamental information for the correct reconstruction of the input image, we also propose a method for robustifying those headers with the use of error-correcting barcodes.
- In chapter 7, we present a wet-lab experiment which has been carried-out for one of the proposed encoding solutions in order to prove feasibility of the end-to-end solution when also including the intermediate biological manipulations.

- Finally, in chapter 8, we provide a summary of the methods presented in this thesis by conducting conclusions, discussing some important points and presenting some interesting steps to follow in our future studies.

Chapter 2

State of the art on DNA coding

2.1 General workflow

In the previous section, we presented the reasons for which DNA is an eco-friendly solution offering the possibility of storing a great amount of information in a very small volume while also promising longevity of the stored data. We also explained that DNA coding is a multi-disciplinary subject which is inspired by the quaternary code of DNA and highly depends on the biological processes of DNA synthesis and sequencing. Those two methods are reminiscent of a digital noisy source channel which adds any type of noise to the transmitted data. Therefore, the process of DNA data storage can be thought of as a classical encoding workflow for the transmission of 4-ary data through a noisy channel. The general coding scheme for DNA data storage is depicted in figure 2.1.

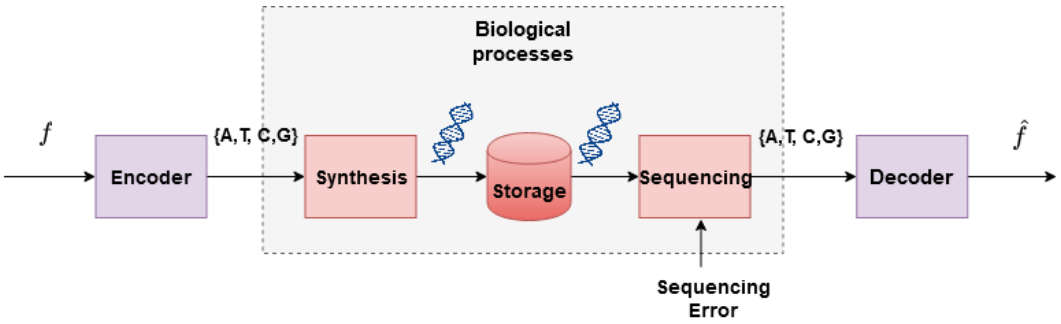


Figure 2.1: Main component parts of a typical DNA storage process.

Although this process might seem simple, it is very important to denote that this is only a very rough and simple presentation of the general DNA coding workflow. However, as briefly explained in section 1.2.2, DNA synthesis and sequencing are very delicate and complex processes which introduce some important constraints when it comes to the encoding of digital data. In the following sections we will briefly explain those complex yet fundamental biological procedures in more detail and we will present the main challenges as well as the elaborated solutions which form a more complete extended workflow.

2.1.1 The structure of DNA

DNA is a molecule composed of two strands of nucleotides forming a double helix that is carrying genetic instructions for the development, functioning, growth and reproduction of organisms. The two DNA strands are known

as polynucleotides as they are composed of simpler monomeric units called nucleotides. Each nucleotide is composed of one of four nitrogen-containing nucleobases (cytosine [C], guanine [G], adenine [A] or thymine [T]), a sugar called deoxyribose, and a phosphate group. The double helix has a fixed backbone which is composed by alternating phosphate and sugar groups. [14] The sugar in DNA is 2-deoxyribose, which is a pentose (five-carbon) sugar. The sugars are joined together by phosphodiester bonds between the third and fifth carbon atoms of adjacent sugar rings. These are known as the 3′-end (three prime end), and 5′-end (five prime end) carbons. Therefore, any DNA strand normally has one end at which there is a phosphate group attached to the 5′ carbon of a ribose (the 5′ phosphoryl) and another end at which there is a free hydroxyl group attached to the 3′ carbon of a ribose (the 3′ hydroxyl). The orientation of the 3′ and 5′ carbons along the sugar-phosphate backbone confers directionality to each DNA strand. The two opposite DNA strands are linked through hydrogen bonds between the nitrogen-containing nucleobases according to the complementary base pairing rule which states that DNA base pairs are always adenine with thymine (A-T) and cytosine with guanine (C-G). As a result the two strands are complementary and this yields the fact that the content of one strand defines the content of the opposite one. This complementary base pairing plays a great role in the creation of exact copies of a DNA molecule as one strand can be used to synthesize the corresponding complementary one. This process of creating copies is performed using special enzymes called Polymerases and the process of cloning is called Polymerace Chain Reaction (PCR)(see figure 2.3). In the DNA double helix, the direction of the nucleotides in one strand is opposite to their direction in the other strand: the strands are antiparallel. The asymmetric ends of DNA strands are said to have a directionality of five prime end (5′), and three prime end (3′), with the 5′ end having a terminal phosphate group and the 3′ end a terminal hydroxyl group. The bases lie horizontally between the two DNA strands (figure 2.2). It is important to remark that during the process of PCR amplification the creation of the clone-strands follows a direction from the 5′end to the 3′end of the original DNA strands.

During PCR the original strands of DNA are separated and a polymerase enzyme is bound on special short DNA sequences (primers) which are specific for the given enzyme. The enzyme glides upon each strand with direction from 5′end to 3′end constructing the corresponding complementary strand of each of the two original strands creating this way identical copies. PCR amplification can be repeated many times and after y PCR cycles there are 2^y new strands created.

In DNA coding this process of PCR is highly important for the introduction of some extra redundancy in the produced DNA strands, which can be useful for error correction. More precisely, in the presence of multiple copies of an oligo, some of which might carry errors of insertions, deletions or substitutions of nucleotides, one can easily compute the most representative consensus DNA sequence which is expected to be the closest to the correct one.

[1] https://commons.wikimedia.org/wiki/File:DNA_chemical_structure.svg
[2] https://creativecommons.org/licenses/by-sa/3.0/
[3] https://commons.wikimedia.org/wiki/File:Polymerase_chain_reaction.svg
[4] https://creativecommons.org/licenses/by-sa/3.0/deed.en

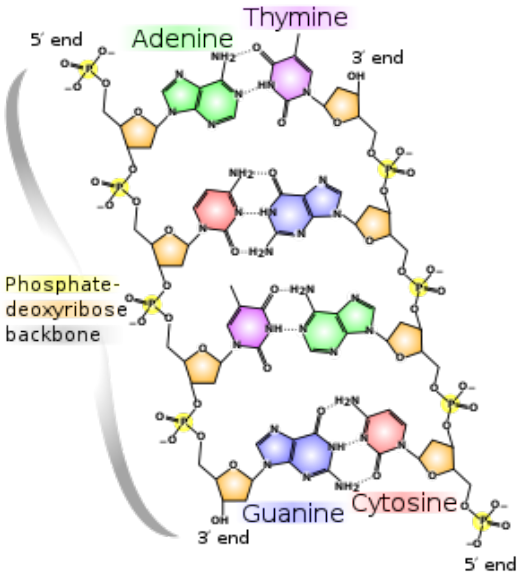


Figure 2.2: The structure of a DNA molecule. (source:wikipedia[1]. Figure under creative commons license.[2]

2.1.2 DNA synthesis

Oligonucleotide synthesis is the chemical synthesis of DNA sequences. The majority of biological research and bioengineering involves synthetic DNA. Today, all synthetic DNA is custom-built using the phosphoramidite method by Marvin H. Caruthers. Oligos are synthesized from building blocks which replicate natural bases. The process has been automated since the late 1970s and can be used to form desired genetic sequences as well as for other uses in medicine and molecular biology. However, the chemical construction of DNA sequences is impractical beyond 200-300 bases and constitutes a hazardous process. These oligos, of around 200 bases, can be connected using DNA assembly methods, creating larger DNA molecules [15]. Although information can be retrieved very quickly from DNA through next generation sequencing technologies, de novo synthesis of DNA is a major bottleneck in the process. Only one nucleotide can be added per cycle, with each cycle taking seconds, so the overall synthesis is very time consuming.

As described in [16] the commonly used phosphoramidite synthesis chemistry consists of a four-step chain elongation cycle that adds one base per cycle onto a growing oligonucleotide chain attached to a solid support matrix (Fig. 2.4).

- In the first step, a dimethoxytrityl (DMT)-protected nucleoside phosphoramidite that is attached to a solid support (usually contained within a synthesis column) is deprotected by the addition of trichloroacetic acid. This activates the support-attached phosphoramidite for chain elongation with the next phosphoramidite monomer.
- In the second step, the next base in the sequence is added in the form of a DMT-protected phosphoramidite and is coupled to the 50-hydroxyl group of the previous nucleoside phosphoramidite in the sequence forming a phosphite triester.

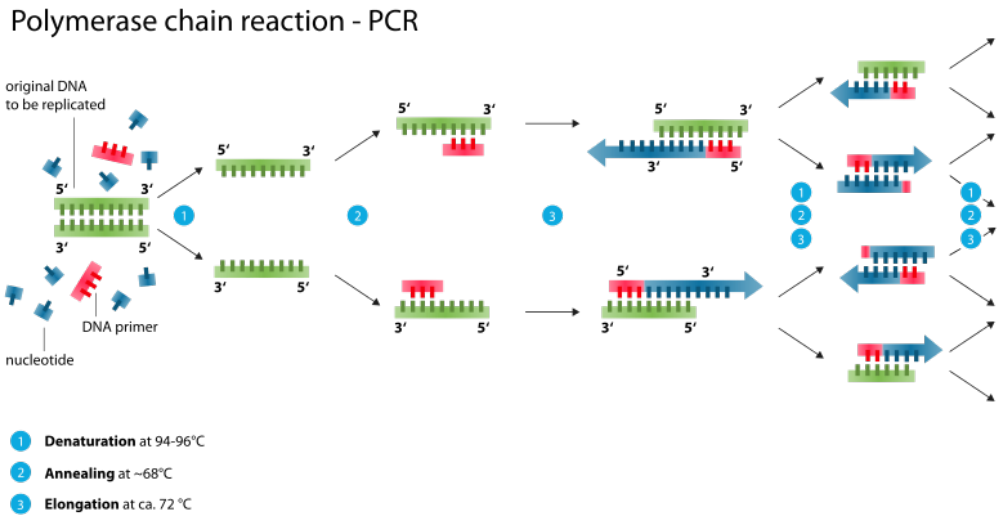


Figure 2.3: The process of Polymerace Chain Reaction (PCR) explained (source: Wikipedia [3], figure under creative commons licence [4]

- Third, any unreacted 50-hydroxyl groups are capped by acylation to render any unextended sequences inert in subsequent rounds of the chain elongation cycle and thus reducing deletion errors in the finished oligonucleotide sequences.
- In the fourth step, the phosphite triester linkage between the monomers is converted to a phosphate linkage via oxidation with an iodine solution to produce a cyanoethyl-protected phosphate backbone.
- The synthesis cycle then repeats for the next base in the sequence via the removal of the 50-terminal DMT protecting group. After the desired sequence has been synthesized from the 30 to 50, the oligonucleotide is chemically cleaved from the solid synthesis support and the protecting groups on the bases and the backbone are removed.

The above process has been automated since the late 1970s and can be used to form desired genetic sequences as well as for other uses in medicine and molecular biology. However, creating sequences chemically is impractical beyond 200-300 bases, and is an environmentally hazardous process. These oligos, of around 200 bases, can be connected using DNA assembly methods, creating larger DNA molecules.

2.1.3 DNA sequencing

As described in section 1.2, one can read the content of DNA strands with the use of some special machines, the sequencers. In practice, there are many different machines to perform sequencing. However, the most popular-ones for DNA data storage, are the Illumina sequencers which belong to the category of Next Generation Sequencing (NGS) and the Oxford Nanopore which is a more recent model and belongs to the group of third-generation DNA sequencers. To get a better idea of how sequencing works we will describe in this section the way those two models function for reading a DNA strand.

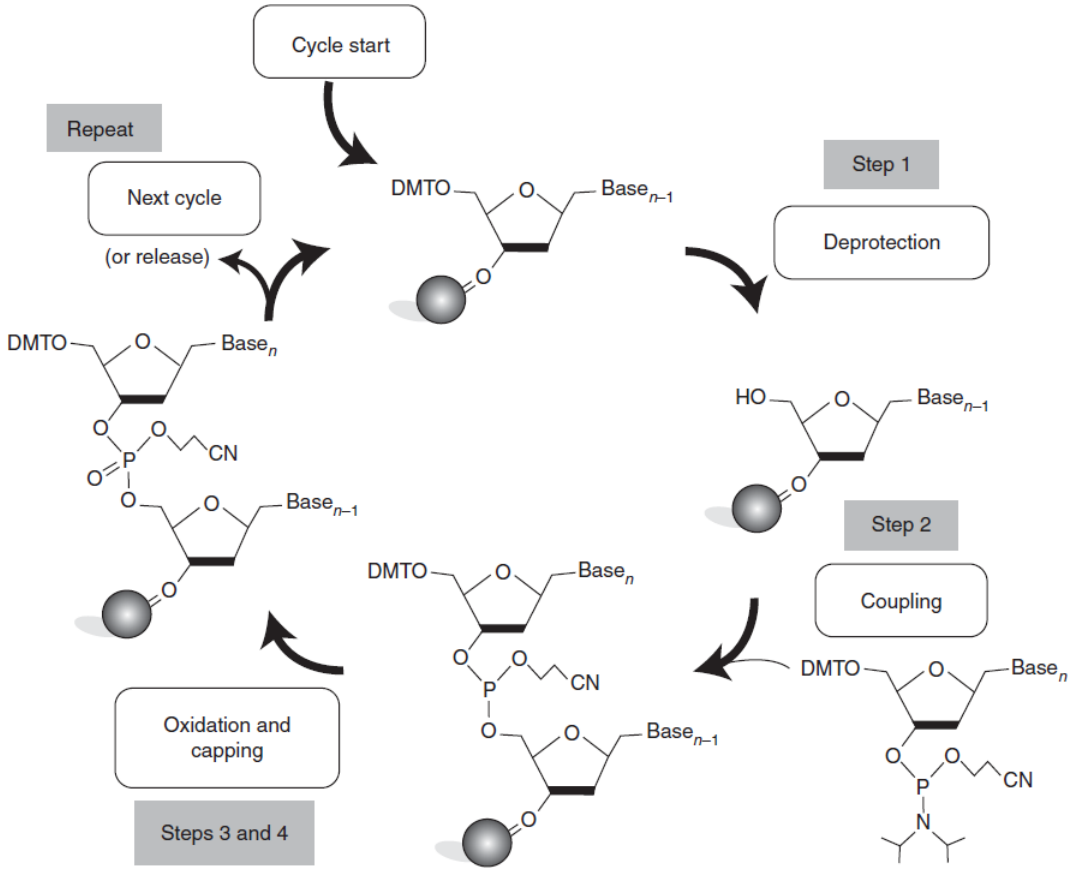


Figure 2.4: The steps of a cycle of DNA synthesis (source: [16])

Illumina: One of the most accurate sequencing machines is the one of Illumina which reads DNA strands using a method called Sequencing by Synthesis (SBS). This method works in different cycles.

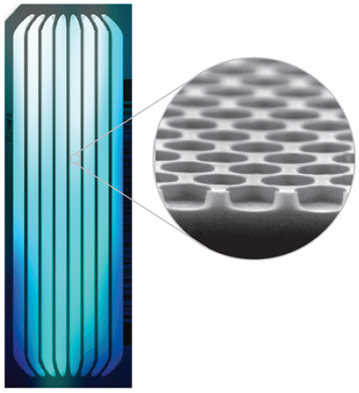


Figure 2.5: Illumina flowcell (source: [17])

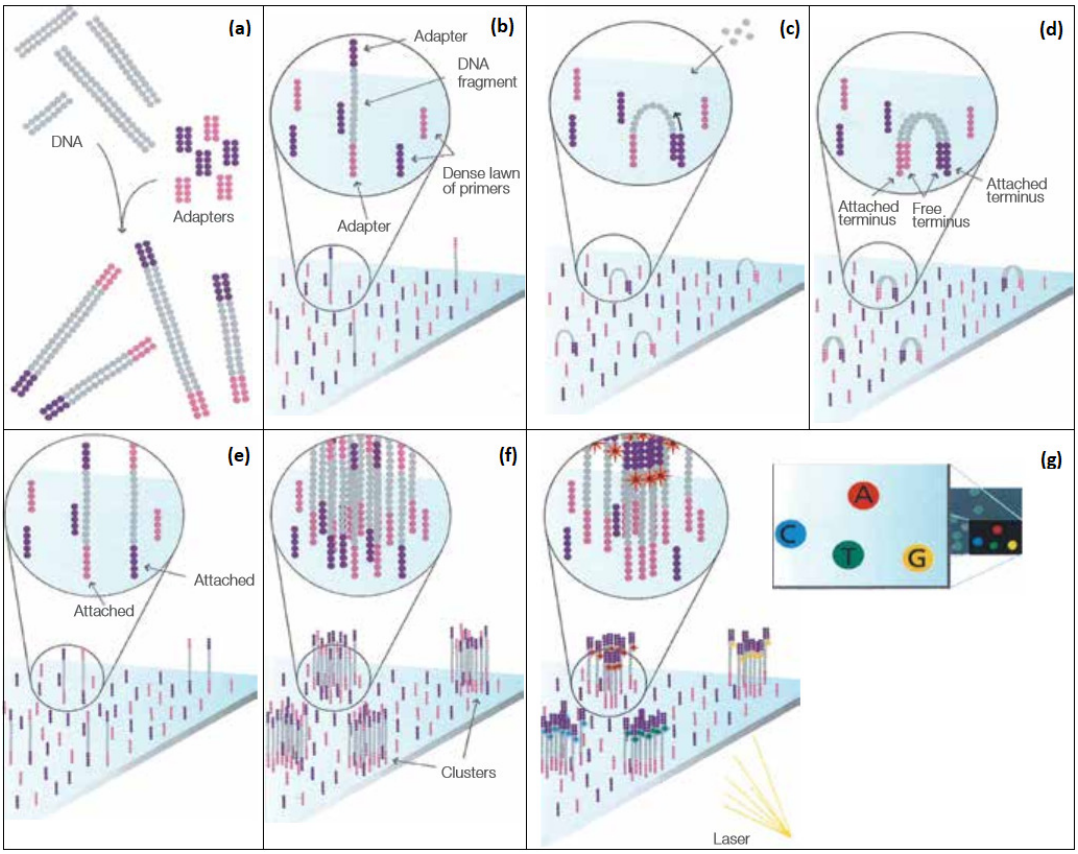


Figure 2.6: Sequencing with illumina explained (source: [17]).

To begin with, short sequencing DNA-templates (which we will call primers), are immobilized on the surface of a proprietary flow cell surface (figure 2.5) designed to present the DNA in a manner that facilitates access to enzymes. The primers that are bound on the flow-cell are complementary to the 3′ and 5′ ends (which we will call primers) of the oligos to be read allowing them to bind on the flow-cell's sequencing templates as shown in (figure 2.6 b). In the next step, the free ends of the oligos bend and bind on the corresponding complementary templates forming a bridge-like form (figure 2.6 c). In this phase of the sequencing the reading begins. Special enzymes allow the

synthesis of the complementary strands by binding the corresponding nucleotides on the "bridge single strands". The complementary nucleotides are fluorescently labeled using 4 different dyes according to the nucleotide type. Each time a nucleotide is bound in position, the fluorescent dye is imaged to identify the base and then enzymatically cleaved to allow incorporation of the next nucleotide. This is the reason why this process is called sequencing by synthesis. Once the double strands are created, they are separated again discarding the original strands and the same process will be repeated many times creating this way thousands of copies, which are frequently mentioned as reads, of each stored oligo. This process of copying the oligos by forming bridges and synthesizing the complementary strand is called Bridge Amplification (BA). BA works similarly to PCR and is adding the required redundancy to allow denoising. More precisely, amplification creates up to 1,000 identical copies of each single stored oligo in close proximity creating a blob of the same oligo which is expected to emit at every new synthesis cycle the same sequence of fluorescent colors forming blobs of the same color which are easily identified (see figure 2.7).

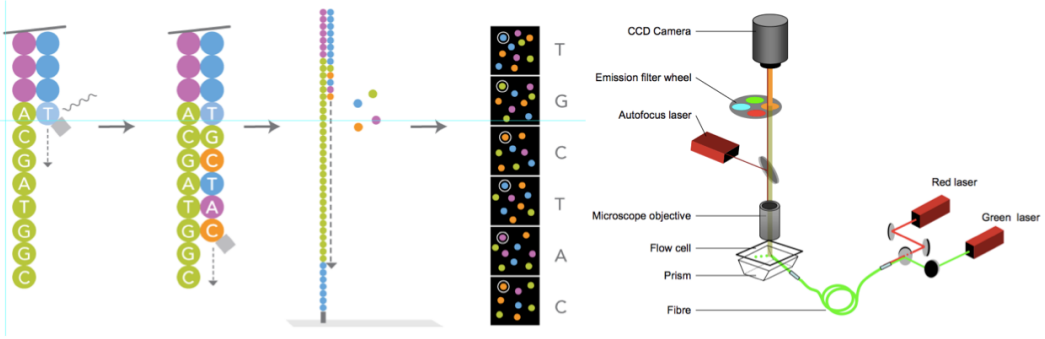


Figure 2.7: Illumina basecalling explained. Image from Illumina [17] showing how the four-colours work and the camera system in the older sequencers, the image of the cameras comes from the Bentley Nature paper of 2008 [18]

Base calls are made directly from the signal intensity measurements during each cycle and thus raw error rates greatly reduces compared to other technologies. However, in the case that different oligos which are bound to neighboring regions on the Illumina flow-cell happen to have similar structure of nucleotides the spotted color clusters might be overlapping merging the different color blobs into one, creating this way a difficulty in identifying the separate oligo structures and this way, introducing some errors in the sequenced DNA strands. An image of the Illumina sequencer is depicted in figure 2.8.

Figure 2.8: The Illumina sequencer [1]

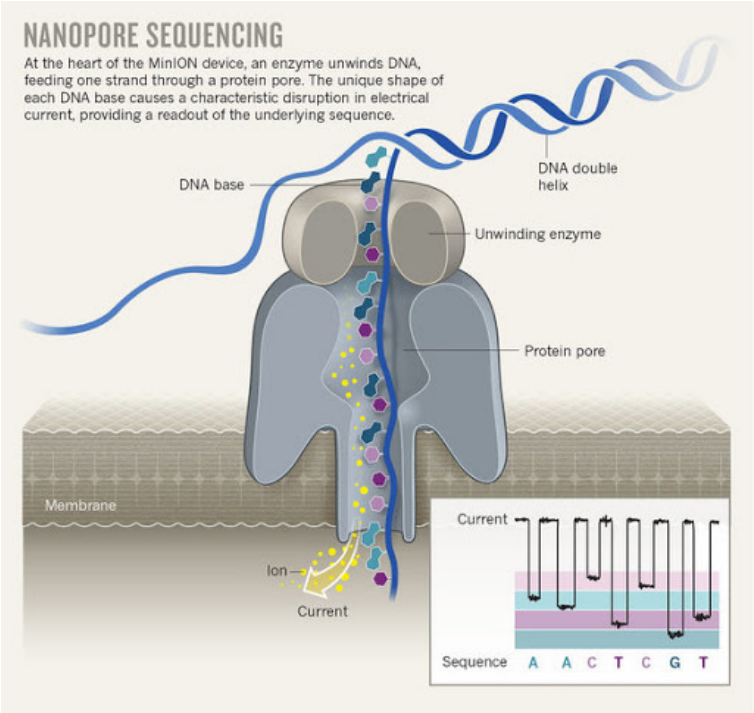


Figure 2.9: Sequencing with Nanopore explained [19].

Oxford Nanopore: As revealed by the name a Nanopore sequencer consists of a flowcell of nano-scale pores (holes) through which passes a single strand of DNA. An ionic current is constantly passing through the nanopores at the same time with the DNA strands. As different nucleotides differ in size, each one of them will create a different change in the flow of ions when passing through the pore. The information about the change in current can be used to

[1] https://www.illumina.com/systems/sequencing-platforms/nextseq.html

identify each molecule. More specifically, the changes in the ionic current as biological molecules pass through the nanopore or near it generate a signal which is then used for the decoding of each oligo.

The produced raw data (electrical signals) is processed using machine-learning techniques into basecalled data (the sequence of DNA bases). In other words, the electrical signals are translated into sequences of nucleotides. This procedure is carried out using kmer tables that translate sequentially fragments of the electrical signals into sets of nucleotides. A descriptive image of the function of the Nanopore sequencer is shown in figure 2.9.

It is the smallest sequencing device currently available. It can plug directly into a standard USB3 port on a computer with low hardware requirement and simple configuration. It also allows to sequence longer reads (up to few hundred thousand base pairs), improving the assembly quality [20]. Its portability, affordability, and speed in data production makes it suitable for real-time applications, enabling the sequencing of full human genomes quickly and at affordable prices. However, some analysis on the results of MinION nanopore sequencer like [21] show that although the sequencing coverage is generally consistent, between 2% and 3% of the positions are underrepresented. Among those, approximately 50% were located at the beginning and the end of the reference sequence which can lead to the loss of both ends when its coverage does not reach 20x since we are not able to correct and assemble those fragments. An image of the Illumina sequencer is depicted in figure 2.10.

Figure 2.10: The MinION nanopore sequencer by Oxbord Nanopore Technologies[2]

2.2 A constrained problem

DNA synthesis is a procedure with a very low error-probability as long as the DNA strands to be synthesised do not overpass the length of 150-300 nts. For longer sequences the synthesis error increases exponentially. Consequently to reduce this error to minimum, 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 correctly reconstructed in the decoding part. Detailed explanation for the formatting of the DNA sequence will be given in chapter 6.

On the contrary, the biological procedure of DNA sequencing introduces much error which can not be neglected and therefore there is a need for dealing with the erroneous oligos produced by the sequencer. Studies have

[2] https://nanoporetech.com/products

shown that the three main factors causing errors in the sequenced oligos are the following:

- **Homopolymers:** Consecutive occurrences of the same nucleotide should be avoided [22].
- **G, C content:** The percentage of G and C in the oligos should be lower or equal to the one of A and T [22].
- **Pattern repetitions:** The codewords used to encode the oligos should not be repeated forming the same pattern throughout the oligo length [23].

Taking into account all the above rules the sequencing error can be reduced. Consequently, to be efficient, any DNA coding algorithm should respect the above rules in order to reduce as much as possible the probabilities of sequencing error. To this end, in chapter 3 of this work we propose a novel efficient encoding algorithm which can be used for the encoding of digital information into DNA using codewords that respect all those biological constraints.

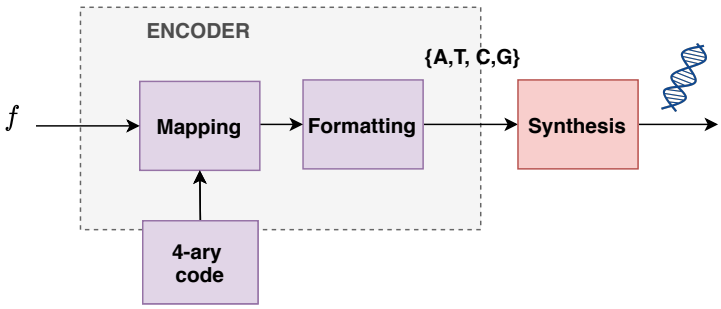


Figure 2.11: Encoding of digital data into DNA.

2.2.1 Encoding

Until this point, it is clear that the encoding of digital data into DNA is strongly constrained by the biological part of the process. More precisely, to sum up the main obstacles which have been discussed in the previous sections the encoding should provide a quaternary code which will respect the sequencing restrictions to ensure robustness and the length of the DNA oligos to be synthesised should not be higher than 150-300 nts. Consequently, the structure of a reliable encoder for DNA coding contains the following sub-parts (see figure 2.11).

The first step in the encoding workflow is the construction of a dictionary of codewords composed by the symbols A, T, C and G similarly to the nucleotides of the DNA molecule. Those codewords should provide a robust encoding when assembled at a long sequence. This means that the quaternary strands should not contain homopolymers, high G,C content compared to the content of A and T and finally it should not contain repeated patterns.

The next sub-process of a DNA workflow is a mapping function which assigns input symbols to codewords of the quaternary code. This function can be a simple one to one function or a more sophisticated one. In later

sections of this work we will extensively explain the mapping methods that have been used for our studies.

Finally, as the oligo length is restricted due to the synthesis limitations to avoid errors, it is necessary to adopt some formatting function for cutting the produced long encoding into shorter oligos and adding special headers for the reconstruction of the input at decoding. Those headers can contain information for the address of the data chunk in the original long sequence, information for any necessary encoding parameters as well as information about the input characteristics as for example the size. A general overview of the encoding of data into DNA is described by figure 2.11.

2.2.2 Decoding

Since DNA data storage is a process which is prone to both writing and reading errors, the decoding should include some techniques to predict, detect or even to correct the sequenced data. As explained in section 2.1.2, the addition of redundancy is necessary for the detection of errors and can be easily achieved using the method of PCR amplification which is applied during both DNA synthesis and sequencing. Consequently, in the output of the sequencer there will be multiple copies of each synthesised oligo. Each copy might contain different types of errors in various positions and this yields the need for selecting the most representative copy for each oligo. This selection can be based on computing a consensus sequence using all of the erroneous copies of each oligo or on finding the most frequent among all copies. This process can be followed by some error correction algorithm to treat any remaining errors for obtaining an error free decoding. It is important to mention that the efficiency of the error correction highly depends on the methods and machines that have been used during sequencing as some particular sequencers can cause higher error rates than others and can therefore create stronger distortion. Finally using the inverse mapping function one can retrieve the digital information which had been stored into DNA. An overview of the decoding process is described by figure 2.12.

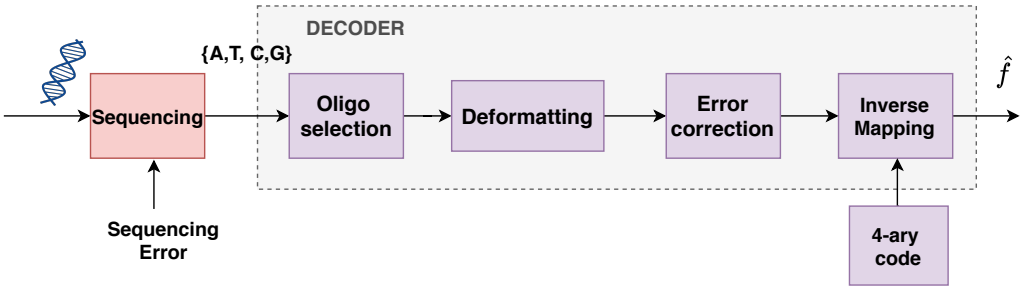


Figure 2.12: Decoding of digital data stored into DNA.

2.3 Existing works

DNA data storage is a relatively new field of research and thus the state of the art is limited to a few pioneering works which have, however, contributed widely to this emerging topic.

2.3.1 First references to the idea of DNA data storage

The idea for storing digital data using the DNA molecule ages back in the late 50's when soviet physicist Mikhail Samoilovich Neiman and cybernetician Norbert Wiener expressed ideas regarding the possibility of recording, storage, and retrieval of information on synthesized DNA and RNA molecules [24], [25]. However the first attempt of DNA data storage came in 1988 when the artist Joe Davis and researchers from Harvard collaborated for storing a 5 x 7 matrix in a DNA sequence in E.coli, which once decoded, formed a picture of an ancient Germanic rune representing life and the female Earth [26]. In the matrix, ones corresponded to dark pixels while zeros corresponded to light pixels. In 2007 at the University of Arizona scientists create a device which is using addressing molecules to encode mismatch sites within a DNA strand. These mismatches were then able to be read out by performing a restriction digest, thereby recovering the data. This was the starting point for various interesting works that followed, introducing multiple novel encoding algorithms that brought DNA data storage to practice and contributed widely to this emerging topic. In the following sections we will present the most widely used studies in the bibliography and briefly analyse the proposed solutions.

2.3.2 The first application of DNA data storage by Church *et al.*

In 2012, George Church et al. encode for the first time a 659-Kbyte book that was co-authored by Church into DNA. In their experiment the authors used a very simple encoding, by randomly translating zeros to A or C and ones to T or G [22]. The encoded sequence was then written onto a microchip as a series of DNA fragments using an ink-jet printer. The encoding resulted in 54,898 oligonucleotides, containing 96 bases of data along with a special 22-base sequence at each end to allow the fragments to be copied in parallel using the PCR amplification, and a unique, 19-base "address" sequence to denote the segment's position in the original document.

The resulting PCR amplified oligos were then read back using an Illumina sequencer to retrieve the original text. The storage density of the DNA fragments produced by this method was estimated to be more than 700 terabytes per cubic millimeter. This result represented the largest volume of data ever artificially encoded in DNA, and proved that data density for DNA is several orders of magnitude greater than that of state-of-the-art storage media as shown in their plot in 2.13.

Not only did this work make a pioneering step to prove the feasibility of using DNA as an alternative means of storage while demonstrating the extraordinary capacity compared to conventional storage devices but it also revealed that sequencing can be an error prone process. By analysing the different errors which occurred during sequencing this work provided a first study of the main constraints to be respected during the encoding.

After this important first step, several works followed to propose new encoding techniques, attempting to provide a robust encoding which would allow reducing the sequencing errors obtained in this study.

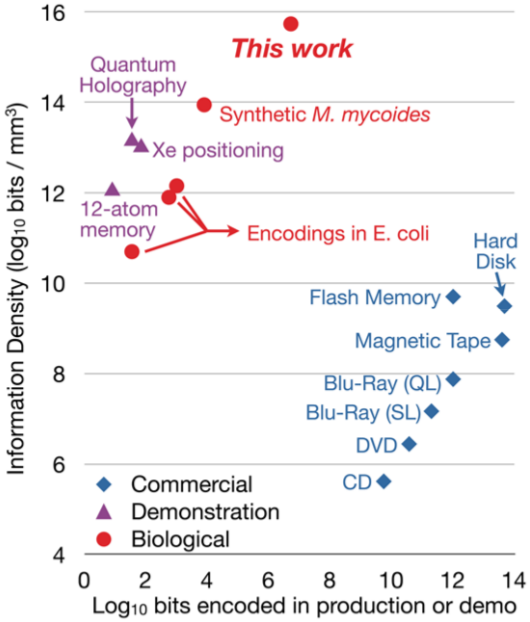


Figure 2.13: Results of the study carried out by Church *et. al.* [22]

2.3.3 First biologically constrained encoding by Goldman *et al.*

In 2013, Goldman et al. [1] proposed a novel algorithm for encoding digital data into binary so to respect the main sequencing constraints. The encoding proposed using ternary Huffman algorithm to encode each byte of a binary sequence into the digits 0,1 and 2. Those digits are then associated to three of the symbols A, T, C and G omitting the symbol that has been used for the encoding of the previous digit, so to ensure that no base is used twice in a row. This strategy avoided the creation of homopolymers while still making use of DNA's four-base potential. To enhance the reliability of the oligos and determine the data's position in the original file, Goldman's team synthesized oligonucleotides carrying 100 bases of data, with an overlap of 75 bases between adjacent fragments, so that each base was represented in four oligonucleotides creating a fourforld redundancy. Even so, the researchers lost two 25-base stretches during sequencing, which had to be manually corrected before decoding. The encoding followed in this study is explained in figure 2.14.

The code construction proposed by this work has been thereby used by Microsoft researchers in their later works.

2.3.4 Introduction of Reed-Solomon codes by Grass *et al.*

To deal with the remaining sequencing errors, in 2015, Grass and his team [27] have proposed for the first time the use of Reed Solomon codes to introduce error correction in the encoding. More precisely, in this work the authors proposed mapping the data to blocks which contain elements from Galois Field 47 (GF(47)). The column of each block is extended using a unique index consisting of elements in GF(47). The extended columns are then encoded to DNA by mapping each of the GF(47) elements to a triplet of nucleotides while ensuring that there is no repetition of the same base in the two last positions ensuring that homopolymers are avoided. Each encoded

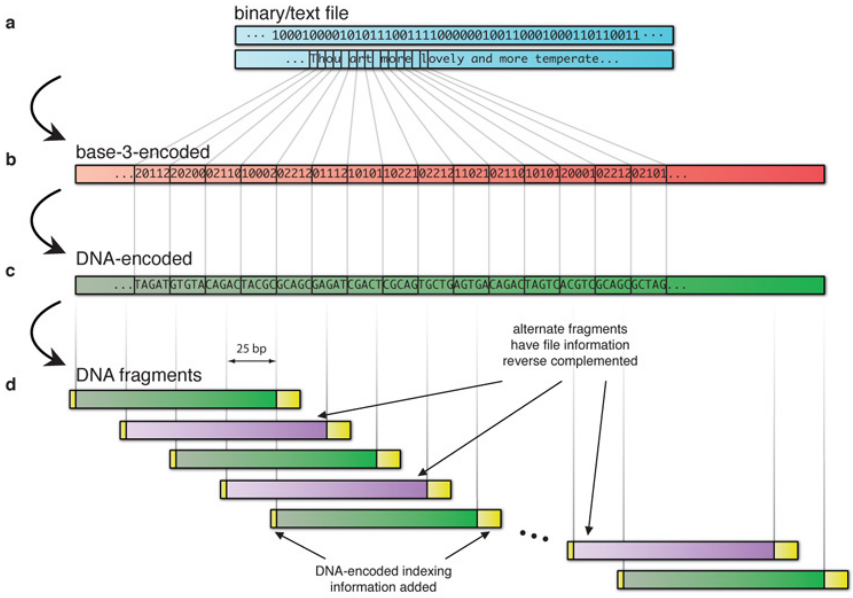


Figure 2.14: Goldman *et. al. encoding [1]*

column represented a DNA fragment to be synthesized and stored in silica to ensure long-term storage without corruption of the DNA. In their study the authors reported perfect retrieval of 83 kB of data encoded using a Reed-Solomon code, an error-correcting code used in CDs, DVDs, and some television broadcasting technologies. The storage workflow is shown in figure 2.15.

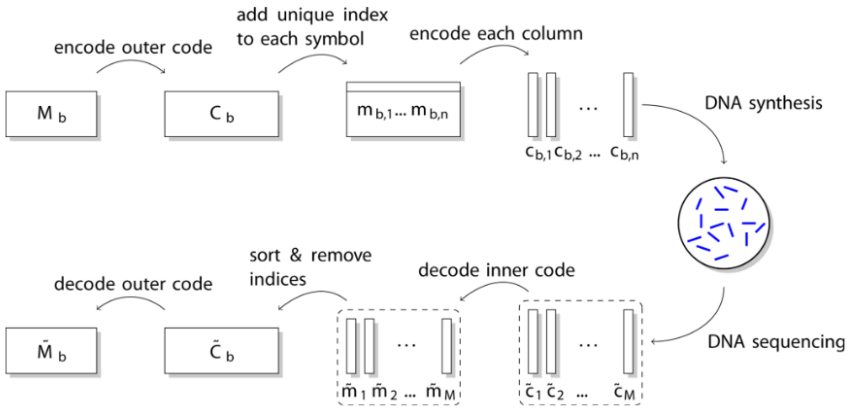


Figure 2.15: Grass *et. al.* encoding [27]

2.3.5 First random-access implementation by Yazdi *et al.*

At the same year (2015) Yazdi et al. [28] have introduced an important way for allowing random access using specific and robust addressing in the encoding! In their study, the authors proposed the addition of some specially designed primers in both ends of the encoded data to allow selective PCR amplification of particular oligos instead of amplifying the full oligo pool. The primers were specially designed to be robust to sequencing errors and the encoding DNA words for each oligo depend on the corresponding primer.

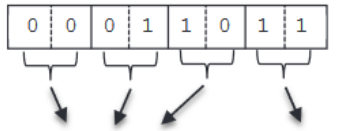
More precisely, for each oligo the DNA code is constructed by ensuring there is no correlation of the payload to the oligo's addressing primer as this would create secondary structures which can be catastrophic and can lead to losing the full oligo during sequencing.

In a later study published in 2017 [29], the authors provided an experiment testing the efficiency of their proposed encoding using the MinION — Oxford Nanopore's handheld sequencer for the reading of the DNA while also using JPEG compression to reduce the synthesis cost. This study has devised error-correcting algorithms specifically for the kinds of mistakes the MinION makes. The result is an error-free read-out, demonstrated earlier this year when the team stored and sequenced around 3.6 kB of binary data coding for two compressed images.

Finally in a co-authored paper of Chao Pan ([30]), the research group proposed the use of inpainting techniques for post-processing the image and correcting discolorations which occurred due to the synthesis and sequencing errors.

2.3.6 Reed Solomon codes on headers by Blawat *et. al*

In 2016 Blawat et al. [31] published another interesting method for constructing a robust quaternary code. In their work, the authors presented a new method for creating a quaternary code by encoding each byte of some digital data to 5 nucleotides using the following algorithm. To begin with the first three pairs of bits are encoded to the corresponding nucleotides from table 1 of figure 2.16 and represent the first, second and fourth nucleotide respectively in the encoded DNA word. Then the last pair of bits can be encoded to a pair of nucleotides among 4 different options as presented in table 2 of figure 2.16 and will be placed in the third and fifth position of the resulting DNA word. As a result, for each byte there are provided 4 different DNA words. To ensure that the limitation concerning the maximum run-length is respected, the 4 options are filtered so to not create homopolymers.



Value	Nucleotide
00	A
01	C
10	G
11	T

Table 1: Mapping of first 3 bit-tupel to 3 nucleotides

Value	Option 1	Option 2	Option 3	Option 4
00	AA	CC	GG	TT
01	AC	CG	GT	TA
10	AG	CT	GA	TC
11	AT	CA	GC	TG

Table 2: Mapping of last 2 bits to a pair of nucleotides

Figure 2.16: Blawat *et. al.* encoding [31]

To do so the authors propose keeping only the options that do not violate the following rules.

- The first three nucleotides shall not be the same.
- The two last nucleotides shall not be the same.

With the above described constraints at least 2 valid DNA symbols can be found for every data byte, thus introducing some redundancy which can be used for error detection. More precisely, the authors proposed separating the different codeword options into different predefined clusters and encode each input byte using the encoding of a specific cluster according to the byte's position. For example one option would be to use codewords from cluster A to represent even positions and cluster B for odd byte positions. Thus, in the case where an error alternates a codeword expected to be found in one cluster to another one that belongs to some other cluster, error detection is possible. Furthermore, in this work the authors proposed robustifying the addressing headers using Reed Solomon codes to allow a more reliable decoding.

2.3.7 DNA coding using Fountain codes by Erlich *et al.*

At the same year (2016), Columbia University researchers Yaniv Erlich and Dina Zielenski published a method based on a fountain code [32], an error-correcting code used in video streaming. As part of their method, they used the code to generate many possible oligos on the computer, and then screened them in vitro for desired properties. Focusing only on sequences free of homopolymers and high G content, the researchers encoded and read out, error-free, more than 2 MB of compressed data—stored in 72,000 oligonucleotides—including a computer operating system, a movie, and an Amazon gift card. Their encoding followed the following steps:

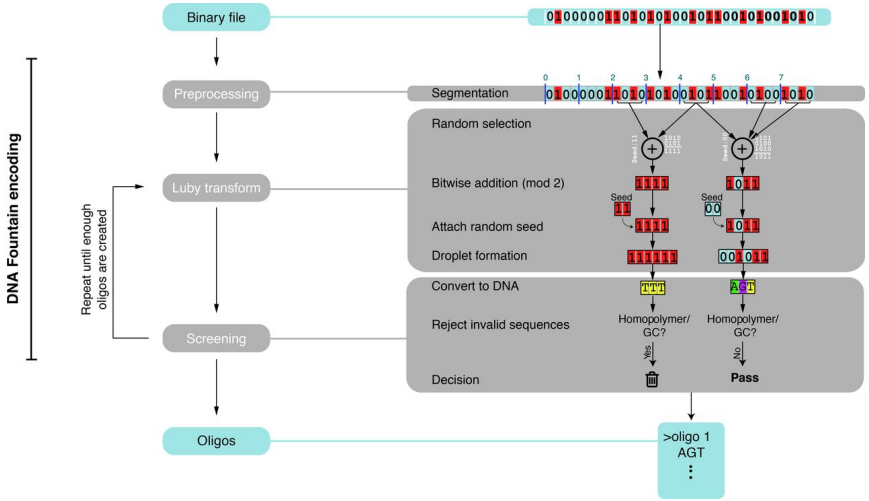


Figure 2.17: Erlich *et. al. encoding [32]*

First, the input binary file is segmented in partitions. Then using a luby transform droplets of bits are created by selecting randomly segments from the input sequence and bit-wise adding them attaching also the random seed used for the selection. The resulting bit droplets are then encoded into quaternary and scanned for satisfying the biological constraints of GC content and homopolymers. Encoded droplets which do not respect the above restrictions are discarded while the rest are used for creating the oligos. This process is repeated until enough oligos are produced resulting in a densely compressed encoding that reaches a capacity of 1.98 bits/nt.

2.3.8 Efficient end-to-end workflow by Microsoft researchers

In 2016 Borhholt et al. in a Microsoft research presented a DNA based archiving system using the quaternary code introduced by Goldman et al. In this study they improved the encoding by avoiding the fourfold redundancy using themselves addressing primers for allowing random access [33]. Researchers in Microsoft have then in 2017 presented some extra studies to improve their results using a clustering algorithm to cluster and correct the multiple reads provided by the sequencer allowing a better reconstruction quality [34], [35]. Finally, in 2019, a Microsoft team successfully encoded the word "hello" in snippets of fabricated DNA and converted it back to digital data using a fully automated end-to-end system, which is described in [36].

Those are the most well-known studies on DNA data storage until the starting date of the study presented in this manuscript! More details about the contributions of this work are given in the next section.

2.4 Contributions of this work

While DNA data storage is suited for the archiving of any type of digital data, the subject of this thesis focuses on the encoding of images. All the studies of the state of the art which have been described above, are providing some way for building a quaternary encoding of digital data by respecting the biological restrictions discussed in section 2.2. Each one of those encodings exhibits different advantages and weaknesses and since the subject is still very new it is necessary to provide new encoding ideas which can help enriching the existing studies and improve the quality of the stored data.

As the main drawback of DNA data storage is the high synthesis cost, the encoding methods proposed in the bibliography attempt to improve the storage capacity while also being robust to sequencing errors. To this end, some studies have proposed compressing images with JPEG before encoding. However, no study has proposed a method for controlling this compression such that it provides a closed loop solution which can allow selecting the best compression parameters for a given coding potential. In our study we included a source allocation algorithm which offers the possibility of not only reducing the synthesis cost, but also promising an optimal quality of the stored image for a predefined encoding rate and thus a given synthesis cost. As a low complexity source allocation requires a fixed length code we also propose a new efficient algorithm for the construction of a robust fixed length DNA code that facilitates the nucleotide allocation method. We also introduce two different mapping methods. The first one deals with pattern repetitions which might be the cause of error increase in the Illumina sequencers and has not been tackled by previous studies, and the second one aims in decreasing the visual impact of substitution errors which may remain after error correction. The reason for implementing a fixed-length encoder stems from the fact that variable-length coding is less robust to sequencing errors. In other words, in case of an error, variable-length coding is prone to losing important information about the structure of the encoded data which can result in wrong reconstruction of the input image.

To prove this last claim, we also implemented a variable-length encoder which is inspired by the classical binary JPEG encoder. This idea has been

created thanks to the JPEG Ad Hoc group which has recently shown interest in building a new JPEG standard for the purpose of image coding into DNA. Our proposed solution uses a modified workflow of the classical JPEG standard for binary coding which optimizes the compression of the input image according to a constrained quaternary code, producing a compressed nucleotide stream which is robust to sequencing error. Finally, we present a new formatting structure for cutting the encoded information into oligos and adding the needed headers which suits our proposed encoding.

Chapter 3

A novel constrained quaternary encoding

3.1 Introduction

One of the main challenges for providing an efficient solution for DNA data storage is the construction of a DNA code which is robust to sequencing noise. The process of sequencing can be compared to a digital noisy channel of wireless communications which can introduce bit errors, erasures, or even full packet loss to the transmitted data. To deal with such errors, in communication systems, the encoding process is composed by two main subprocesses as described by figure 3.1. The source coding aims in the encoding of the input information to a binary representation while the channel coding process adds any needed redundancy to allow error detection and correction. Examples of such redundancy would be the use of parity nucleotides to check correctness of the data received by the decoder as proposed by Appuswammy *et al.* in [37], or the use of Reed-Solomon codes as proposed by Grass *et al.* in [27] and Blawat *et al.* in [31] which are used for error correction. Hence, since sequencing is error-prone, it is necessary that both source and channel coding are applied when encoding digital data into DNA. In this section, we will present our proposed methods for encoding any source symbols into a quaternary stream while being robust to sequencing noise.

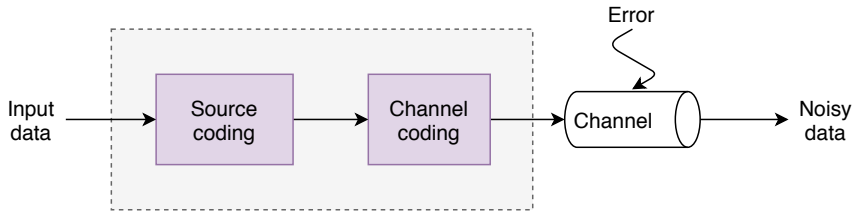


Figure 3.1: Encoding process of communication systems. The sequencing process resembles to a noisy channel of wireless communications as it introduces similar types of errors. Therefore the encoding of digital data into DNA should be encoded accordingly using both source coding and channel coding to deal with sequencing noise.

As extensively described in section 2.3, in the state of the art works, there have been proposed many different DNA code-construction algorithms. Nevertheless, most of the existing methods propose a variable length solution while also limiting application to binary input data. In this work we introduce a novel fixed length algorithm for building a robust fixed-length quaternary code that respects the biological restrictions imposed by sequencers

and can be applied to any representation of the input.

While this thesis focuses on the robust encoding of images into DNA, the proposed codebook creation is not restricted by the nature of the input data. Consequently, before discussing the application of our proposed code to image coding, we will first introduce in this chapter the method for creating a constrained quaternary code given an input set of symbols of any kind. We also propose a novel mapping function for assigning the constructed DNA codewords to the input indices so to avoid pattern repetitions.

3.2 Creating a constrained DNA code - Our solution

3.2.1 General Definitions

Before explaining the algorithms proposed by this work, it is necessary to define some notions related to the encoding. We first introduce the three following sets:

- The set $\mathcal{V} = \{v_1, v_2, \ldots, v_K\}$ of elements $v_k \in \mathbb{R}^n$ with $k = \{1, 2, \ldots, K\}$.
- The set $\mathcal{C}^*$ of finite sequences (words) c_j over the alphabet $\{A, T, C, G\}$ with $j = \{1, 2, \ldots, L\}$.
- A set $\Sigma = \{1, 2, \ldots, K\}, \Sigma \subset \mathbb{Z}^{*+}$

Let us define the function $\alpha : \mathcal{V} \mapsto \Sigma$ such that:

$$\alpha(v_k) = k, \forall v_k \in \mathcal{V} \tag{3.1}$$

that provides the index k of some element $v_k \in \mathcal{V}$. We also define $\Gamma : \Sigma \mapsto \mathcal{C}^*$ the mapping function which provides the following relation:

$$\Gamma(k) = c_j, \text{ with } c_j \in \mathcal{C}^* \tag{3.2}$$

The above function maps the index k of some element $v_k \in \mathcal{V}$ to some codeword $c_j \in \mathcal{C}^*$. The mapping can provide a one-to-one or one-to-many relation depending on the needs of the encoding. The composition of α and Γ, $\alpha_n = \Gamma \circ \alpha$, stands for the encoding function which assigns a codeword $c_j \in \mathcal{C}^*$ to an input symbol v_k.

Obviously, Γ is invertible and the inverse function can then be expressed as:

$$\Gamma^{-1}(c_j) = k \tag{3.3}$$

and will provide the index k which is mapped to some given codeword $c_j \in \mathcal{C}^*$. Thus to retrieve the element in $\mathcal{V}$ that corresponds to the index k one can apply the inverse function:

$$\alpha^{-1}(k) = v_k \tag{3.4}$$

When it comes to DNA coding for archival purposes, the code $\mathcal{C}^*$ should contain viable codewords which respect the main rules imposed by the sequencing as detailed in section 2.2. Having introduced the main terms of the encoding process, in the next paragraph we will explain our proposed algorithm for constructing a novel constrained fixed length code for encoding any the elements of any source input into DNA words.

3.2.2 Construction of the codewords (PAIRCODE)

Let's assume the source set $\mathcal{V}$, and let $\Sigma = \{1, 2, \ldots, K\}$ with $|\Sigma| = K$, be a set of indices of the elements $v_k \in \mathcal{V}$ to be encoded into a set $\mathcal{C}^* = \{c_1, c_2, \ldots, c_L\}$ of L quaternary codewords (with $L \geq K$) of length l. The goal of this encoding algorithm is to generate the code Γ where $\Gamma : \Sigma \rightarrow \mathcal{C}^\star$. We denote $\Gamma(k) = c_k$ the codeword associated with the index $k \in \Sigma$.

The algorithm for the construction of the code $\mathcal{C}^*$ is inspired by the restrictions imposed by the biological procedures included in the process of DNA data coding. The main idea is the creation of codewords by selecting elements from a set of duplets (pairs of symbols) which, when assembled in a longer strand, create an acceptable sequence. 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:

- $\mathcal{C}_1 = \{AT, AC, AG, TA, TC, TG, CA, CT, GA, GT\}$
- $\mathcal{C}_2 = \{A, T, C, G\}$

As observed, dictionary $\mathcal{C}_1$ is composed by pairs of symbols that when selected and concatenated to create a longer strand, the resulting quaternary stream will respect the biological restrictions. More precisely, to ensure that the code does not create homopolymers, dictionary $\mathcal{C}_1$ does not contain pairs of the same symbol. This means that the pairs AA, TT, CC and GG are omitted from $\mathcal{C}_1$. Furthermore, to keep the CG percentage lower or equal to the one of A and T, the pairs GC and CG are also excluded.

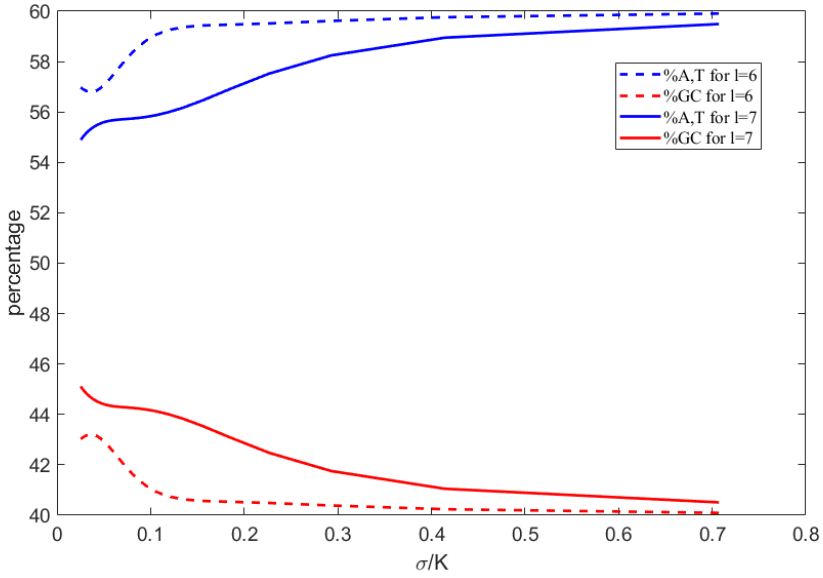


Figure 3.2: Evolution of the percentages of A,T (blue curves) and G,C (red curves) for a centered Gaussian source of variance σ^2 and mean $\mu = \frac{K}{2}$ (K denoting the number of different source symbols) in function of the source dynamic normalized by K. The dashed and continuous-line curves represent two different cases of codeword length l of 6 and 7 nucleotides respectively.

Codewords of an even length l, are constructed by selecting $\frac{l}{2}$ pairs from dictionary $\mathcal{C}_1$. Codewords of an odd length, are constructed by selecting $\frac{l-1}{2}$ pairs from $\mathcal{C}_1$ also adding a symbol from $\mathcal{C}_2$ at the end of the codeword. 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 $\mathcal{C}_1$. There are $10^{\lfloor \frac{l-1}{2} \rfloor}$ different choices for filling those first nucleotide pairs. Then, to fill in the last nucleotide or equivalently the last pair of nucleotides in the case of even codewords, one should choose a symbol from $\mathcal{C}_2$ or a last pair of symbols from $\mathcal{C}_1$ respectively.

It is obvious to prove that such an encoding respects the first restriction among the ones described in 2.2 for avoiding homopolymers. Furthermore, by construction, the proposed algorithm is expected to produce a percentage of A's and T's which is around 60% and a percentage of C's AND G's which is around 40%. This is consistent with the recommended percentages for reducing the sequencing error. To verify the GC percentage of a stream produced by our proposed method, we have computed the evolution of the A,T and G,C percentages of a DNA sequence which has been created using our code and given an input source of symbols that follows a Gaussian distribution of the variance σ^2. Figure 3.2 illustrates the result of the different percentages in function of the source dynamic. As expected, we can see that the amount of content of G's and C's tends towards 40% while the one of A's and T's tends to 60% as the source becomes more uniform (big values of σ). Interestingly enough, the plot shows that the dashed line (even codeword length) reaches the asymptote more rapidly than the continuous line (odd codeword legth). This is due to the fact that in the case of an odd codeword length, the codewords are constructed using an extra nucleotide from the dictionary C_2, which obviously modifies the two percentages. Therefore, it is clear that our encoding process deals with the first two restrictions described in 2.2

At this point, it is necessary to mention that if the number K of different indices k to be encoded is lower or equal to 4 we avoid an encoding which uses exclusively symbols from dictionary $\mathcal{C}_2$. This is due to the fact that in the case where the same index k is repeated many times in a row in the input sequence, the encoding can create either homopolymer runs or pattern repetitions. Hence, even though it is feasible, an encoding rate $R = 1$ nt/index is avoided in order to ensure robustness of the code. As a result the codebook size L is given by the following relation.

$$L = \begin{cases} 10^{\frac{l}{2}}, & \text{if } l \text{ is even} \\ 10^{\lfloor \frac{l}{2} \rfloor} * 4, & l \text{ is odd} \end{cases} \tag{3.5}$$

If the code was not constrained by the biological restrictions of DNA sequencing, the code would contain all 4^l possible arrangements of the symbols A, T, C and G. However, since some codewords are considered as non-viable due to the fact that they might contain homopolymer runs or increase the G,C content, the length L of the constrained code $\mathcal{C}^*$ will be restricted to specific values. More precisely, assuming $L \in \{L_1, L_2, \dots\}$, then:

$$L_1 = 10 \text{ and } L_{i+1} = \begin{cases} 4L_i, & i\text{: odd} \\ 10L_{i-1}, & i\text{: even} \end{cases} \tag{3.6}$$

Consequently, 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 increase by 10. This codebook extension can be relatively big compared to the encoding needs ($L >> K$) and can possibly leave a big part of codewords unused. Such an example is the case where $K = 12$ in which a code of $L = 40$ constrained codewords would be required, thus leaving 28 codewords unused. In other words, the number of unused codewords is more than 2 times bigger than the number K of input indices to encode. 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 index 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 $\mathcal{C}^*$ 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 section 3.3.

3.2.3 Discussion

The algorithm described in the above section provides a constrained code $\mathcal{C}^*$ of a size L which contains only viable DNA codewords which when assembled at a longer DNA strand, respect the restrictions described in section 2.2. The proposed algorithm is efficient and low complexity but due to the way of constructing the DNA words using specific pairs of symbols, some viable codewords are omitted. More precisely, the selected code omits the following categories of viable codewords:

- Codewords with a correct GC percentage which contain a consecutive C and/or G in positions i and $i+1$ of a codeword with i an even non-zero integer.
- Codewords with no homopolymers that contain the same nucleotide in positions i and $i+1$ of a codeword with i an even non-zero integer. To this last case we should precise that since codewords are concatenated for the creation of an encoded DNA strand, it is important that the concatenation avoids homopolymers. Therefore, codewords which begin or end with repetition of the same nucleotide 3 times should be avoided. In addition to this, codewords can either begin or end with a pair of the same symbol but this should not be allowed in both the beginning and the end of codewords. Once a selection of allowing a pair of the same symbol in the beginning or the end of codewords has been chosen, it should be respected through-out the whole encoding.

The construction of a code which includes the above types of viable codewords, which are omitted when using our codebook construction algorithm (PAIRCODE), can be achieved using the following method.

- Construct a code $\mathcal{C}$ containing all 4^l different quaternary words of A, T, C and G of length l.
- Read every word in $\mathcal{C}$ discarding the ones which:
  - Have a high G,C percentage
  - Contain an homopolymer run
  - Have a repetition of the same symbol twice in the end of the codewords. This last restriction can be also applied by allowing a pair of the same symbols in the beginning but avoiding it in the end of codewords.

The above method results in a constrained code $\mathcal{D}$ that contains all possible viable codewords that respect the DNA sequencing constraints. This is an asset over the proposed PAIRCODE algorithm. However, this filtering of codewords from the set $\mathcal{C}$ requires $O(n^2)$ extra computations. This additional complexity is increasing as the size of the code to construct gets higher. Furthermore, the creation of the constrained code using PAIRCODE is trivial and instantly creates a code containing only viable codewords even if omitting some of them. On the contrary the creation of the code $\mathcal{D}$ requires extra computations and if the code will need to be computed both during encoding and decoding the computational time can increase. A solution to this problem would be the storage of all possible codes of any possible size so to avoid recomputing it with every use but this would require allocating much memory. Consequently, while PAIRCODE is omitting some viable codewords, it is more efficient in computational cost and does not need storage of the created codes. Table 3.1 shows the comparison in terms of computational time and number of generated codewords for both methods of DNA code construction. Even though the difference in computational time might seem negligible for creating a code of shorter DNA codewords, we must point out the fact that this code might need to be created multiple times iteratively when applied to algorithms which search for a convergence point. Such an algorithm is the one of source allocation for the optimization of the encoding which will be extensively discussed in latter sections. Therefore, in such a scenario, the difference in computation time between the two algorithms adds-up and the total difference might reach much higher values.

	PAIRCODE		EXHAUSTIVE CODE	
# nts	Time (s)	# words	Time (s)	# words
2 nts	10^{-3} s	10	10^{-3} s	10
3 nts	$2*10^{-3}$ s	40	$5*10^{-3}$ s	60
4 nts	$9*10^{-3}$ s	100	$2*10^{-2}$ s	130
5 nts	$3*10^{-2}$ s	400	$4*10^{-2}$ s	612
6 nts	$2*10^{-2}$ s	1,000	10^{-1} s	1,944
7 nts	10^{-1} s	4,000	1 s	9,618
8 nts	$2*10^{-1}$ s	10,000	1, 8 s	29,856
9 nts	1.2 s	40,000	8.4 s	140,352
10 nts	2.5 s	100,000	35 s	465,024

Table 3.1: Comparison between the PAIRCODE algorithm and the Exhaustive code generation (for word length 2nt to 10nt).

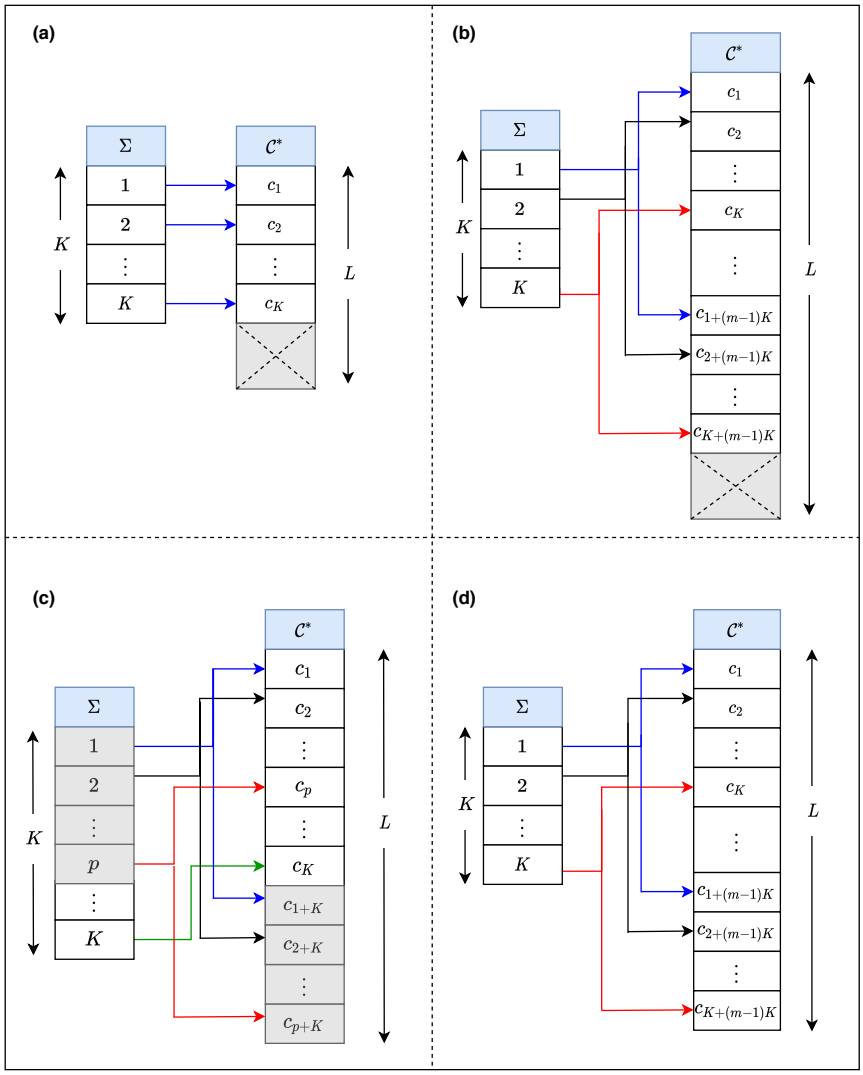


Figure 3.3: Mapping of quantized values into quaternary code

3.3 The proposed mapping algorithm for avoiding pattern repetition

3.3.1 The algorithm

The mapping step of the encoding algorithm is an application $\Gamma : \Sigma \mapsto \mathcal{C}^*$ associating an index in Σ to one or more possible codewords in $\mathcal{C}^*$. 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 indices Σ can fit more than one times into the codebook $\mathcal{C}^*$. 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 ($m = 1$),

the mapping algorithm is trivial (see figure 3.3a.) and is described as:

$$\Gamma(k) = c_k \tag{3.7}$$

If the replication step is needed for the purposes of the encoding, the mapping algorithm works as illustrated in figure 2b. In this case, the code $\mathcal{C}^*$ is constructed so that each index in Σ is mapped to $m \geq 2$ different non-empty quaternary codewords in $\mathcal{C}^\star$ following a one-to-many relation in such a way that it is uniquely decodable. By ensuring $L \geq 2K$, the pseudo-random mapping can at least provide two possible codewords for one input symbol. The value of m can be selected by the user and the mapping is performed using the following algorithmic steps:

1. Decide the number of replications m needed for the encoding.
2. Build the corresponding code $\mathcal{C}^*$ of size $L \geq mK$ using all possible codewords of length l which can be built following the rules described in section 3.2.2.
3. The mapping of the input indice k to a codeword in $\mathcal{C}^*$ is given by:

$$\Gamma(k) = c_{rK+k}, \quad \text{with } r \in \mathbb{Z}^+ \text{ and } 0 \leq r < m \tag{3.8}$$

The choice of the value r can be either random or can follow some predefined selection pattern.

In the case where the corresponding code size L is very large compared to the number K of symbols to encode such that ($L \geq 2K$) the encoding can profit of a replication step either-way. For example, in the case where one needs to encode 12 symbols ($K = 12$), the corresponding code will have a size of $L = 40$. Therefore there can be $m = 3$ replications of the set of symbols Σ into $\mathcal{C}^*$ by default. Thus it is important that before selecting the number of replications needed for the encoding the user is aware of the dynamic of the corresponding code. The algorithm for the encoding of a known set of symbols Σ to quaternary words in $\mathcal{C}^*$ is described in algorithm 1 (see Appendix A).

Another important point would be the fact that when a fixed set of symbol indices Σ should be encoded into a quaternary code $\mathcal{C}^*$, it is probable that a part of the codewords is left unused (grey partition in figures 3.3a and 3.3b). An interesting approach for taking advantage of those unused words would be their assignment to the most probable codewords from Σ, assuming that the information of symbol frequency is also transmitted to the decoder. This mapping scenario is illustrated by figure 3.3c. In later sections we will discuss the possibility that the size K of the set Σ is not predefined but is rather computed according to some desired code size L. Such an example is the case of using the proposed code construction algorithm for image coding where the input has to be quantized and encoded such that the encoding rate does not exceed a predefined value R_{target}. In such applications the size of the code L is given as an argument and the encoding should be performed by quantizing the input in such a way so that the code is fully used (i.e., L is a multiple of K) providing the optimal compression quality. In other words, the produced code $\mathcal{C}^*$ has no codewords which are left unused and consequently the mapping can be described by figure 3.3d. The encoding procedure in such a case is explained by algorithm 2 (see Appendix A).

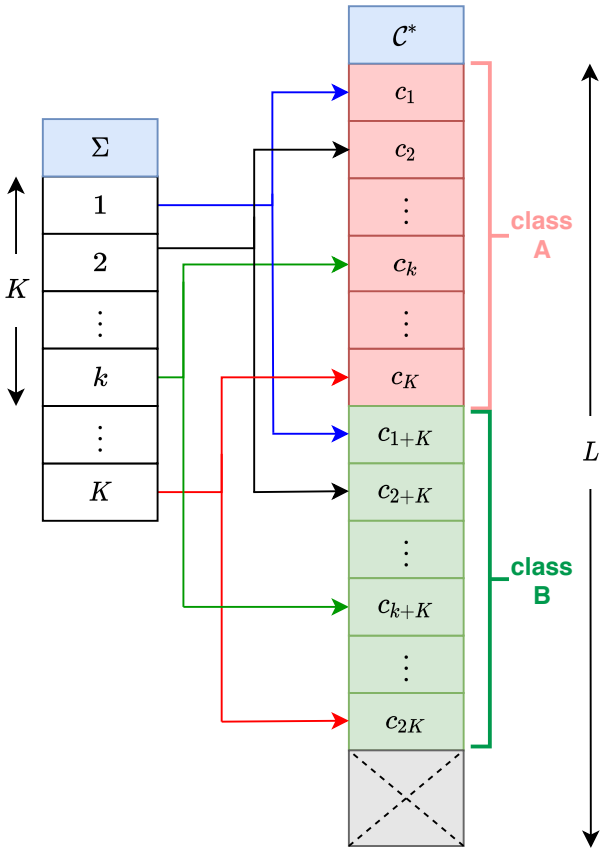


Figure 3.4: Mapping of quantized values into quaternary codewords using a controlled selection of one of the m possible codewords for a given index.

3.3.2 Discussion

The number of times m that the dictionary Σ can fit into the code $\mathcal{C}^*$, reveals the number of different codewords of $\mathcal{C}^*$ that represent any symbol in Σ. Hence, during encoding one needs to select one of the different possible codewords for coding each input symbol. This selection can be performed using a random generator, as described in the algorithm above, or another interesting option would be using a deterministic function which will be known to the decoder. Such an example would be the following one. Assuming a set of symbols Σ of length K which fits $m = 2$ times in the code $\mathcal{C}^*$. Then, let's imagine that the codewords of $\mathcal{C}^*$ are divided into two different classes A and B containing the first and the second possible codewords assigned to each symbol from Σ accordingly (see figure 3.4). A possible deterministic encoding scenario would be coding an index k using the first corresponding codeword representation c_k if the index k is found in an odd position of the input stream, or using the second representation c_{k+K} if the index is found at an even position of the input stream. An interesting advantage of using such a deterministic function for selecting one of the possible codewords assigned to some index can be its potential use for correcting errors that might occur during the sequencing of the data. 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.

3.4 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 described in section 2.3, 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 [22] 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 [1], have proposed an algorithm that respects the constraint from section 2.2 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, 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 [23]. Furthermore, since the works of Goldman *et. al.* use Huffman codes which rely on the frequency distribution of the input in such an encoding and unlike our proposed encoding algorithm, it is necessary to transmit the distribution to the decoder.

A later study in [28], 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. More precisely 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 in sequencing. According to a later publication [29], this encoding can reach a coding potential of 1.57 bits/nt. 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 it is not fixed throughout the encoding process.

Another interesting work has been proposed by Blawat *et al.* [31]. In this study, the authors have proposed using 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

Codec	JPEG 2000				JPEG			
PSNR (dB)	57	48.1	40.2	35.9	61.5	49.2	40.6	35.6
Our rate (bits/nt) $K \leq L \leq 2K$	3.8	6.4	16	32	2.46	3.86	9.4	23.6
Our rate (bits/nt) $L \geq 2K$	2.7	5.3	13.3	26.6	2.05	3.21	7.7	19.72
Goldman et al. rate (bits/nt)	2.9	5.7	14.4	28.8	2.25	3.53	8.6	22.4

Table 3.2: Comparison of Our encoder with $L \geq 2K$ and our encoder with $K \leq L \leq 2K$ to Goldman et. al. [1] Here the rate is expressed in bits per nucleotide (bits/nt) to highlight the coding potential of the different solutions.

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. Our proposed algorithm also needs 5 nucleotides to encode 8 bits (255 different symbols). Nevertheless, a strong advantage of our algorithm is the fact that when applied for transcoding, it can be extended to the encoding of more than 8 bits of information. In addition to this, it can be applied to any type of input data (binary or not). In the works of Grass *et al.* [27], the encoding is performed using Reed Solomon codes. This encoding achieves a coding potential of 1.187 bits/nt introducing some extra redundancy in order to allow error correction. Nevertheless, similarly to [31], it is only applicable to binary stream. Bornholt *et al.* in [33] have applied the same encoding as in [1] 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 [1].

Parameter	Church et al. [22]	Goldman et al. [1]	Yadzi et al. [28]	Grass et al. [27]	Bornholt et al. [33]	Blawat et al. [31]	Erlich et al. [38]	Our work (raw data)
Input data (Mbytes)	0.65	0.75	0.017	0.08	0.15	22	2.15	0.26
Coding potential (bits/nt)	1	1.58	1.57	1.78	1.58	1.6	1.98	1.6
Redundancy	1	4	1	1	1.5	1.13	1.07	1
Error correction	No	Yes	Yes	Yes	No	Yes	Yes	No

Table 3.3: 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).

Finally, Erlich *et* al [38] 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 [1]. The choice

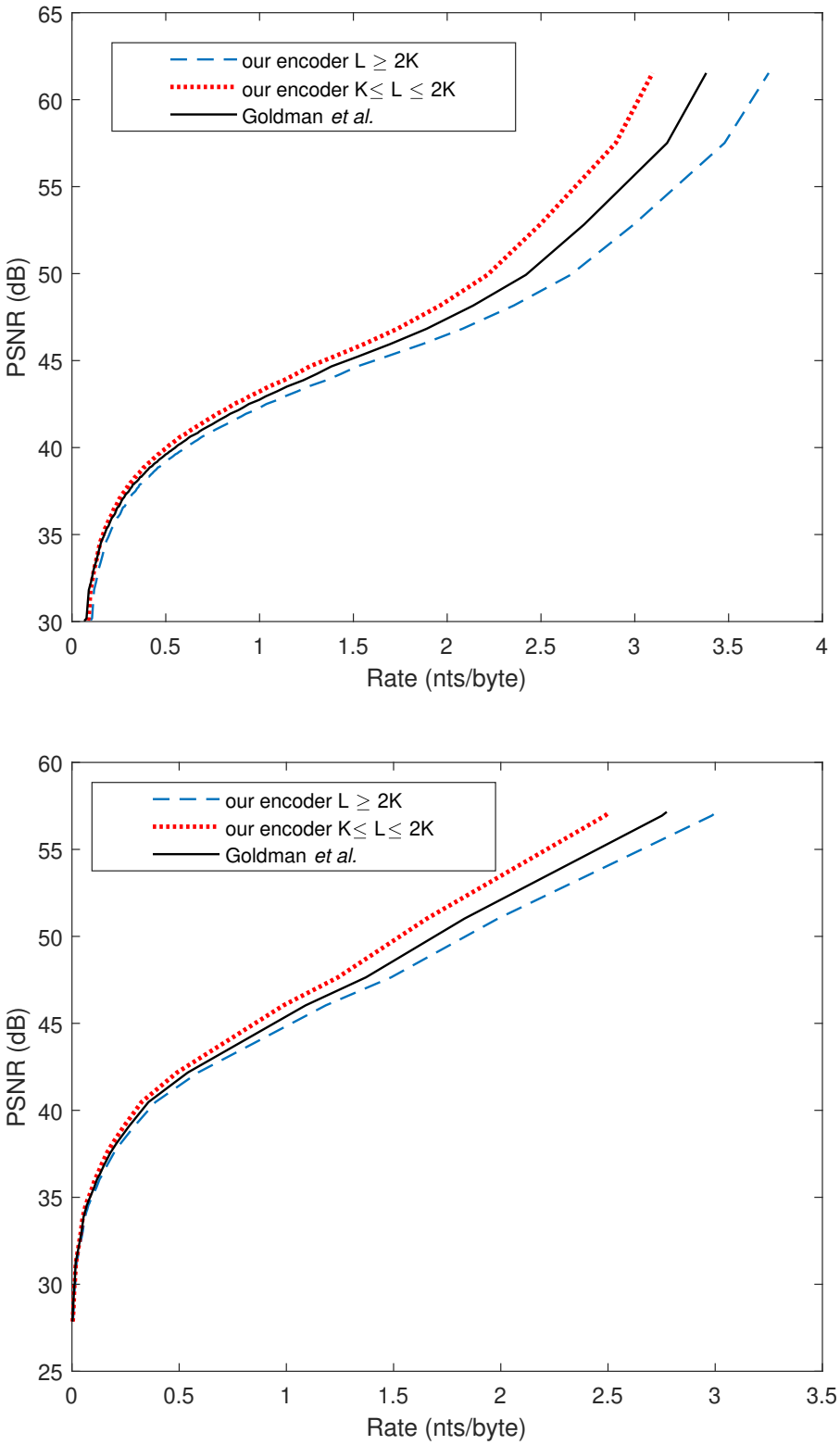


Figure 3.5: Comparison to the encoding algorithm of Goldman *et.* al. For the encoding we consider each byte (8 bits, K=256) as a symbol to be encoded. The figures show the evolution of the PSNR (dB) in function of the rate (nts/byte) for different compression qualities using the JPEG codec (top figure) and the JPEG2000 (bottom figure).

of this work for the comparison is for two main reasons. Firstly, the work proposed by [1] 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 a set of 10 different images [1] of size 1510×5120 pixels to different compression rates. For the experiments we used the classical transcoding method which is used in bibliography for encoding digital images into DNA. More precisely, we assume each byte of the binary stream produced by JPEG and JP2000 codecs represents a different symbol and thus $K = 2^8 = 256$. We then compared our algorithm to the one proposed in [1] to encode the binary stream into a quaternary sequence of A, T, C and G and have built the curves of coding potential (expressed in nts/pixel) in function of the PSNR. The results of this comparison are illustrated in figure 3.5. In table 3.2 we also show the coding rates expressed in bits/nt for different values of PSNR. More precisely, in our results we compare the encoder of [1] to the one proposed in section 3.2.2 for two different cases of mapping. The first case ($L \geq 2K$), is the mapping of one symbol index to at least two different codewords as proposed in section 3.3, and the second case ($K \leq L \leq 2K$) corresponds to the mapping of the most frequent symbol indices to the codewords that are left unused. 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 thanks to mapping repetition our encoder 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, the encoding solution proposed in [1] embeds a ternary Huffman tree to transform the output bitstream of some codecs (like JPEG and JPEG2000) into a ternary sequence of 0,1 and 2 and then encodes it into a sequence of A, T, C and G. Because of the use of Huffman, the probabilities of the different input symbols should be known or transmitted to the decoder as well. To the contrary, our proposed quaternary coder produces a simple fixed length code that doesn't require the transmission of any side information, 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. A comparison of the coding potential that has been reached using the different encoding approaches proposed by the state of the art is presented in table 3.3.

3.5 A controlled code-mapping resistant to sequencing noise

As extensively described in section 2.2 the sequencing of DNA strands is an error-prone process and can cause insertions, deletions and substitutions of nucleotides introducing important noise in the decoded data. Illumina and

[1] https://people.xiph.org/~tdaede/pcs2015_vp9_vs_x264/png/

Nanopore are the two most widely used types of sequencers. While Illumina is the most reliable solution providing a low error-rate in the decoding, scientists are turning towards the more recent Nanopore machine due to its low cost, fast throughput and user-friendly small size. Unfortunately this type of sequencer introduces more noise in the decoded sequence compared to the Illumina, creating the need for finding new ways to deal with errors in the decoded sequence. Existing studies have proposed some interesting methods for error correction by adding redundancy in the encoded sequence, while always using robust encoding to reduce the possibility of sequencing noise. However, the high error-rate introduced by sequencers can't easily be completely eliminated. Thus, in this study we take a very first step in proposing an algorithm which provides a solution resistant to sequencing noise. In other words, this algorithm aims to reduce the impact of the remaining sequencing errors. In this first attempt, we will assume that the input set of symbols Σ contains the indices of vectors produced by a Vector Quantizer(VQ) and we will only focus on the noise of substitutions.

3.5.1 Introduction to the proposed resistant to noise mapping

In [39] DeMarca *et al.* have proposed an algorithm for assigning binary words to codevectors of a multi-dimensional vector quantizer (VQ) in such a way so to be resistant to single-word errors which are inserted by a binary symmetric channel. Inspired by this idea and under the assumption that the substitutions created by sequencers can be considered as bit error noise introduced by digital channels, we extend the algorithm proposed in [39] which is applied to binary codewords to propose a new mapping function which is using a constrained quaternary code. This new mapping algorithm is very interesting in the case that the set Σ contains the indices of vectors which are produced by a VQ. It is also necessary to mention that while in reality sequencing can also cause errors of insertions and deletions for this first study we will only treat substitution errors.

Before describing the proposed mapping algorithm it is important to introduce some basic notions and definitions. Let $\mathcal{V} = \{v_1, v_2, \ldots, v_K\}$ with $\mathcal{V} \in \mathbb{R}^n$ be a set of n-dimensional vectors v_k with $1 \leq k \leq K$ and let's assume that Σ contains the indices k of the vectors in $\mathcal{V}$. For the encoding of such vectors, it is necessary to define a code of at least K quaternary codewords to be assigned to the vectors' indices. Since this study focuses on DNA data storage this specific code should be created using some constrained encoding that respects the sequencing restrictions discussed in 2.2 in order to reduce the noise of the reading process. Hence, in this study we will use as a code the set $\mathcal{C}^*$ of codewords c_k which is constructed using the dictionaries C_1 and C_2 providing a constrained encoding. We recall that the size K of the code $\mathcal{C}^*$ is restricted to specific values as the words should be created according to the constraints imposed by the process of DNA coding.

Assuming a substitution error in the encoded quaternary stream, some correct codeword $c_c \in \mathcal{C}^*$ will be transformed to an erroneous codeword c_e with either $c_e \notin \mathcal{C}^*$ or $c_e \in \mathcal{C}^*$. Under the hypothesis that the error rate produced by the sequencer will be reasonably small, such that any codeword can be affected by only one error, the Hamming distance d_H between the correct and the erroneous codeword will be $d_H(c_c, c_e) = 1$. Therefore, for each correct codeword c_c of length l there are different erroneous codewords

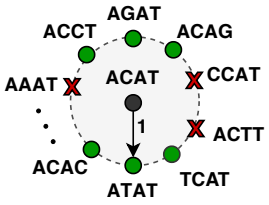
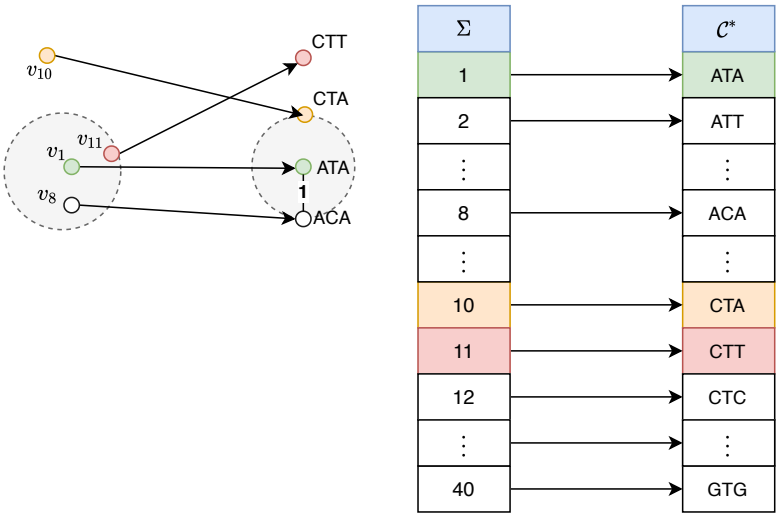


Figure 3.6: Example of a Hamming sphere. The cross-elements denote non viable words that would belong in the Hamming sphere but are omitted due to the constrained quaternary code.

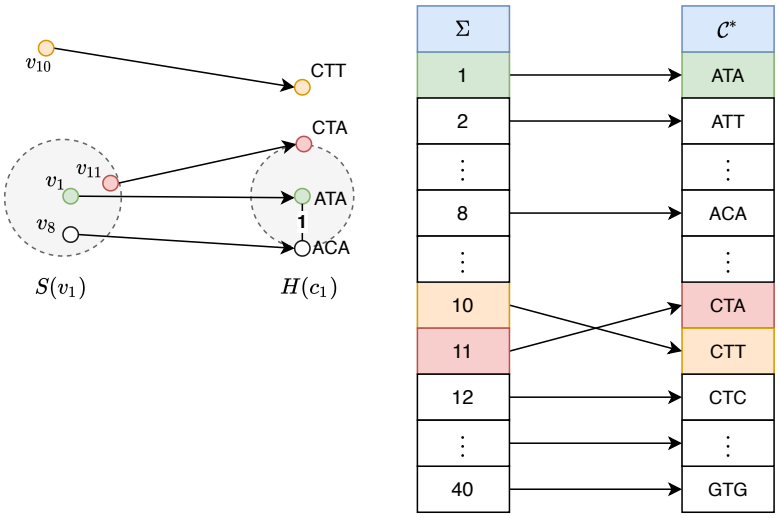
of distance 1. The set of all possible erroneous codewords can be, according to coding theory, represented in the Hamming Space as a sphere $H(c_c)$ of radius 1 the center of which is the correct codeword c_c. An example of such a sphere is depicted in figure 3.6. Let us define $\mathcal{C}$ the set of codewords of length l constructed by using all 4^l possible arrangements of A, T, C and G such that $\mathcal{C}^* \subset \mathcal{C}$. We then define as $\overline{\mathcal{C}^*}$ the set of all codewords w_u that belong to $\mathcal{C}$ but not to $\mathcal{C}^*$. Hence, $\mathcal{C} = \overline{\mathcal{C}^*} \cup \mathcal{C}^*$. In the case where $\mathcal{C}^* = \mathcal{C}$ there would be 4^l different spheres the cardinality of which would be $|H(c_c)| = 3l$. However, as explained in section 3.2.2, since the code used for DNA coding excludes some words which can't be viable due to the encoding constraints, in this work we consider K different spheres with varying cardinality. In other words, some codewords c_e that would normally belong to some sphere of center c_c might be omitted due to the fact that they do not respect the rules of DNA coding (i.e., $c_e \notin \mathcal{C}^*$). As a result, a substitution can cause two different possible types of error:

- **Decodable error:** The substitution transforms a correct codeword c_c to an erroneous codeword c_e which exists in the constrained code $\mathcal{C}^*$ proposed in section 3.2.2 and therefore $c_e \in H(c_c)$. Decoding will then provide an erroneous vector v_e instead of the correctly decoded vector v_c. In the case that the Euclidean distance $d_E(v_c, v_e)$ is small, the produced error will not significantly affect the visual quality of the decoded image. To this end, we propose a special mapping algorithm which allows the assignment of codeword indices to vectors such that the possible errors will lead to a minimum distortion. This algorithm is analytically described in section 3.5.2.

- **Undecodable error:** The substitution transforms a correct codeword c_c to an erroneous codeword c_e which does not exist in the constrained code $\mathcal{C}^*$. In this case, decoding is not possible and thus the application of some error correction is necessary to allow decoding. To distinguish the erroneous codewords c_e from the undecodable erroneous words we will define the later as w_u. The applied correction techniques are further described in section 3.6.



a) Example of mapping vector indices using a simple one-to-one mapping.



b) Example of a controlled mapping where close codewords in terms of Hamming distance are mapped to vectors that are close in Euclidean distance.

Figure 3.7: An example of using a simple mapping vs. using a controlled noise resistant mapping for $L = 40$.

3.5.2 The proposed controlled mapping algorithm

Before explaining the proposed mapping algorithm it is important to understand its main purpose. The idea behind this new controlled mapping is the assignment of indices of vectors which are close in Euclidean distance d_E to codewords which are close in terms of Hamming distance d_H. The reason for performing such an assignment stems from the initial requirement that a single substitution error should produce a minimum visual distortion. Therefore, in case of an error during sequencing and assuming that the sequencing noise is small enough, a correct codeword c_c will be transformed to an erroneous one c_e one which will have a small Hamming distance with the correct one ($d_H(c_c, c_e) = 1$). Thus, if c_c and c_e are assigned to input vectors which are close in Euclidean distance, the error will not significantly affect the image quality.

For better understanding of the algorithm's main purpose let's take the following example. Let us assume a set Σ of vector indices k with $1 \leq k \leq K$ corresponding to vectors v_k and a code $\mathcal{C}^*$ of words of length $l = 3$ nts as depicted in figure 3.7 Let's also assume a simple one-to-one mapping function (figure 3.7a) such that:

$$\Gamma(k) = c_k$$

Since in our study the relation between the input vectors $v_k \in \mathcal{V}$ and their corresponding indices $k \in \Sigma$ is trivial with $\alpha(v_k) = k$, it is obvious that the assignment of an indice k in some codeword $c_j \in \mathcal{C}^*$ with $j = \{1, 2, \ldots, K\}$ yields the assignment of the vector v_k to c_j. Thus, to simplify notations, in this chapter we will from now on be mapping directly the vectors from $\mathcal{V}$ to the codewords in $\mathcal{C}^*$ Then, vector v_1 is represented by codeword c_1='ATA'. Let's then suppose that v_1 belongs to a neighborhood of vectors which are close in Euclidean distance. Assuming a substitution error in the first position of c_1 which transforms the codeword 'ATA' to the codeword 'CTA' which is decoded to the vector v_{10}. Since v_{10} does not belong in the neighborhood of the correct vector v_1 (the two vectors are far in terms of Euclidean distance), the visual distortion caused by this substitution is expected to be high. On the contrary, in the case where the word 'CTA' was mapped to a vector such as v_{11} (figure 3.7b), which belongs to the neighborhood of v_1, the substitution error will cause an erroneous decoding to a vector which is close in Euclidean distance to the correct one and hence the MSE will be low. Therefore, a mapping such as the one proposed by figure 3.7b would be preferable. The proposed resistant to noise mapping aim is creating such an optimal assignment to minimize the distortion caused by substitution errors. To further define which vectors can be considered as close vectors we introduce a set $S(v)$ which contains the closest vectors to the vector v in terms of Euclidean distance d_E. The central idea of the proposed mapping is to assign codewords of the same sphere to vectors which belong to the same neighborhood as shown in figure 3.8. However, such an assignment is not possible for every neighborhood S and thus it is necessary to perform this assignment according to some priority. To this end, we define the following empirical function $F(v)$ for a vector v as proposed by DeMarca *et al.* in [39]:

$$F(v) = \frac{p(v)}{\phi^{\beta}(v)} \tag{3.9}$$

where $p(v)$ is the probability of v in the input sequence, and

$$\phi(v) = \sum_{j|v_j \in S(v)} d_E(v_j, v) \tag{3.10}$$

with $\beta \geq 0$ a trade-off parameter. Therefore, vectors with a higher value for F are considered to belong in a denser neighborhood and should consequently get higher priority to be assigned to the same sphere of words. The algorithm can be very roughly described by the following parts:

- For each codeword c_q: Create a sphere containing the B_q codewords which have a Hamming distance of 1 with c_q. As the code $\mathcal{C}^*$ is constrained, the different spheres $H(c_q)$ created for each codeword c_q will have a different cardinality B_q due to the fact that some codewords are

omitted as non-viable. Thus, we define $B = \max_q(B_q)$, $q \in \{1, \ldots, L\}$.

- For each input vector v_k with $k \in \{1, 2, \ldots, K\}$:
  - Find a set $S(v_k)$ of B neighboring vectors v_{k_b} ($b \in \{1, 2, \ldots, B\}$) which are the closest to v_k in terms of Euclidean distance $d_E(v_k, v_{k_b})$.
  - Compute the following empirical function $F(v_k)$.
- Use algorithm 3 to progressively perform assignment of vectors v_k to codewords c_q such that vectors with a bigger $F(v_k)$ as well as their neighboring vectors $v_n \in S(v_k)$ will be assigned to the same sphere of codewords whenever possible as depicted in figure 3.8. If this is not possible assignment is performed such that vectors are assigned to codewords that do not belong to the current sphere $H(c_q)$ and have a small Hamming distance from the codewords already assigned to their neighboring vectors. This progressive assignment of vectors $v_k \in \mathcal{V}$ to codewords $c_q \in \mathcal{C}^*$ results in a set of vectors $\mathcal{V}_s$ which is sorted according to the codeword assignment.
- Optimization of the first assignment:
  - Exchange the previously mapped codewords between each possible pair of vectors from $\mathcal{V}_s$.
  - For each exchange check if the average distortion has decreased. If true keep this change, else keep the initial state of mapping.

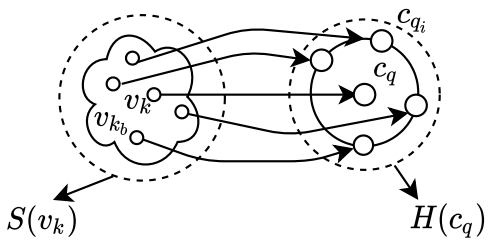


Figure 3.8: Assuming a vector v_k, the set $S(v_k)$ contains the B closest to v_k vectors v_{k_b} in terms of Euclidean distance. Then given a codeword c_q, the Hamming sphere $H(c_q)$ of radius 1 contains all possible codewords c_{q_i} with $i \in \{1, \ldots, B_q\}$ for which $d_H(c_q, c_{q_i}) = 1$. An optimal case of mapping would be the one where all vectors that belong to the same neighborhood $S(v_k)$ are assigned to the same sphere $H(c_q)$. However as this mapping is not possible for all the words $c_q \in \mathcal{C}^*$ we search for a solution that globally optimizes the assignment such that close codevectors are mapped to close codewords.

3.6 Proposed decoding of undecodable words

As discussed in section 3.5.1 in the case in which a substitution error creates an undecodable word it is necessary to employ some error correction to allow decoding. In this section we will propose two possible methods for correcting

undecodable words. To begin with, it is important to denote that the study of the proposed decoding algorithms relies on three main assumptions.

- A substitution error will produce an erroneous codeword which will not differ from the original one in more than one nucleotide.
- Using the mapping algorithm proposed in the previous section it is probable that codewords that are close in terms of Hamming distance are assigned to codevectors which are close in terms of Euclidean distance.
- Since the input data is an image there can be correlations between neighboring elements.

The proposed error-resistant mapping can only be applied to fixed-length encoding workflows. It is this fixed-length code which allows the application of some error-correction methods for correcting any undecodable words which have occurred after a substitution error. The proposed error-correction methods are explained in the next two subsections.

3.6.1 Simple Correction Decoding (SCD)

The first method which we will call Simple Correction Decoding (SCD) is making use of the first two assumptions described in the above section (3.6). Let us assume a correct word c_c which is corrupted by a substitution error producing an undecodable erroneous codeword w_u. Given the first assumption, $d_H(c_c, w_u) = 1$. To allow decoding, w_u should be first corrected to a decodable word c_d. The idea behind SCD is to correct w_u to some word c_d we chose as most appropriate the codeword $c_d \in \mathcal{C}^*$ which is assigned to the median vector-index of the input image. This choice has two possible outcomes. Either c_d will be equal to c_c creating no visual distortion in the decoded image, or $c_d \neq c_c$ and $d_H(c_c, c_d) = \{1, 2\}$. To better understand this last relation let's consider the following example. Assuming a correct codeword c_c = ACA which due to a substitution error in the first position, is transformed to the undecodable word w_u = GCA. Since this erroneous word is not found in the constrained codebook $\mathcal{C}^*$, the codeword should be corrected to some existing decodable one. Supposing that the median vector is assigned to the codeword c_d = GTA we correct w_u to c_d and d_H(ACA,GTA) = 2. However, if the median vector was assigned to the codeword c_d = TCA we would have d_H(ACA,TCA) = 1. In other words, if the correction is performed at the same nucleotide where the error was made then the distance between the correct codeword and the corrected one will be either 0 or 1, else it will be 2.

3.6.2 Advanced Correction Decoding (ACD)

By making use of the third assumption indicated in section 3.6 which states that there can be correlations between neighboring elements in an image, we propose a second more sophisticated decoding, which we will call Advanced Correction Decoding (ACD) and takes advantage of possible correlations between neighboring elements of an image. ACD works in two consecutive decoding cycles. In the first cycle, it performs a first decoding by simply

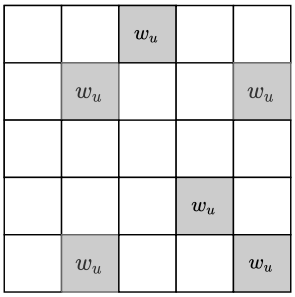


Figure 3.9: The decoding grid after the first decoding step. Gray tiles represent undecodable codewords w_u which still remain in their quaternary codeword representation while white tiles represent already decoded codewords which have been already assigned some index of input vector during the first decoding.

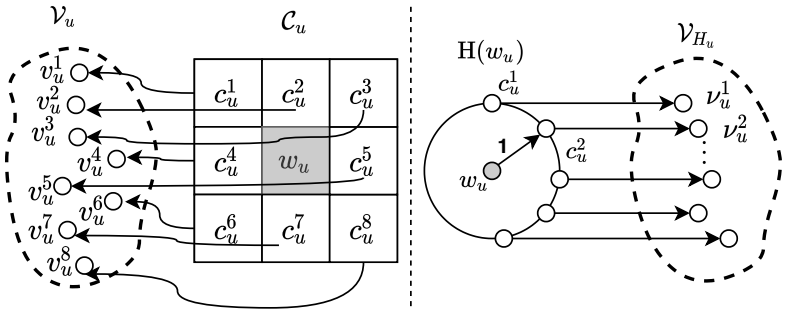


Figure 3.10: The different sets used for ACD decoding. Each undecodable word w_u constitutes the center element of a neighborhood $\mathcal{C}_u$ of neighboring codewords c_u^j. The set $\mathcal{V}_u$ contains the vectors to which are mapped the codewords $c_u^j \in \mathcal{C}_u$. Each undecodable codeword w_u is the center of a sphere $H(w_u)$ containing decodable codewords c_u^i which have a Hamming distance of 1 to w_u. The vectors to which are mapped the codewords that belong to the sphere $H(w_u)$, form the set of vectors V_{H_u}.

omitting all the undecodable words. More precisely, this step decodes the words which exist in the code $\mathcal{C}^*$ to the corresponding VQ indices of Σ while in parallel reconstructing the image in the VQ block-space to retrieve the spatial information. After this decoding cycle, the undecodable words are still expressed in quaternary representation but are spatially placed to the corresponding positions in the image resulting in a decoding state which can be described by figure 3.9.

Before continuing to the second part of the decoding it is important to define the following notions:

- Assuming a set $\mathcal{W} \subset \overline{\mathcal{C}^*}$ containing all the undecodable words w_u with $u = \{1, 2, \ldots, U\}$.

- We define as $\mathcal{C}_u \subset \mathcal{C}^*$ the set of decodable neighboring codewords c_u^j with $j \in \{1, \ldots, |\mathcal{C}_u|\}$ where $0 \leq |\mathcal{C}_u| \leq 8$, which are found around an undecodable word w_u. The size of the neighborhood depends on the needs of the encoding process and can vary. In our study, for simplicity,

we are going to test the proposed algorithms for the 8 neighboring elements around w_u (1-ring neighborhood). As depicted in figure 3.9 after the first decoding cycle the neighborhood might contain vector indices which are already decoded but it is also possible to include some other remaining undecodable codewords. Furthermore it is obvious that the same element might be present more than once in some neighborhood.

- We define $\mathcal{V}_u \subset \mathcal{V}$ as the set of vectors v_u^j with $0 \leq |\mathcal{V}_u| \leq 8$ to which the codewords in $\mathcal{C}_u$ have been assigned.
- We define as $\mathcal{V}_{H_u}$ the set of vectors v_u^i, $1 \leq i \leq |H(w_u)|$ to which have been assigned the codewords in the Hamming sphere $H(w_u)$.
- Define the set $S = \mathcal{C}_j \cap H(w_u) = \{c_1, c_2, \ldots, c_Z\}$ containing codewords $c_z \in \mathcal{C}^*$ with $z = \{1, 2, \ldots, Z\}$ which belong to the intersection of the neighborhood of words of $\mathcal{C}_u$ and the Hamming sphere $H(w_u)$ of the undecodable word w_u.

The second round of the decoding algorithm aims to decode the remaining undecodable words. This step is more complicated and is working according to the following steps:

- If $|S| = 1$ and $S = \{c_k\}$ then $c_u \leftarrow c_k$
- If $|S| > 1$
  - Define $f(c_z)$ as the frequency of a word c_z, $c_z \in S$
  - $c_u \leftarrow c_z \in S$ such that $c_z = \arg\max_z f(c_z)$
- if $|S| = \varnothing$
  - Compute: $D(v_u^i) = \frac{\sum_{j=1}^{|\mathcal{V}_j|} d_E(v_u^i, v_u^j) f(\Gamma^{-1}(v_u^j))}{|\mathcal{V}_{H_u}|}$, $\forall v_u^i \in \mathcal{V}_{H_u}$
  - $c_u \leftarrow c_d$ where $c_d = \Gamma(v_d)$ with $v_d := \arg\min_{v_u^i}(D(v_u^i))$

3.7 Experimental results

In our study we performed multiple comparisons in order to prove the efficiency of the proposed mapping algorithm. For each comparison we ran 10 different realisations of random substitution errors added on the same input image and have plot the improvement in PSNR in function of the error rate. As explained in 3.5.2, the main assumption behind the proposed mapping algorithm is the restriction of the errors to one substitution per quaternary word. Thus, for the noise addition, we used a uniform random distribution to select a percentage of correct codewords from the encoded sequence and we introduced an error of one nucleotide in each selected codeword. The points in each of the plots presented in this section correspond to the mean value of the 10 different realisations of noise.

To begin with, to set an upper bound in the performance of the controlled mapping we tested the improvement of PSNR compared to the non-controlled case by adding only decodable errors. More specifically, in this experiment we added random substitutions while ensuring that the created

erroneous words exist in the constrained codebook. Figure 3.11a, shows the improvement of PSNR (ΔPSNR) for different parameters of K and n for VQ in function of the introduced error rate. As observed in this figure, the controlled mapping can improve the PSNR by at least 3 and at most 7.5 dB in the optimal case where no corrections are needed. The addition of only decodable errors might not be a realistic case in practice but this plot reveals the true potential of the proposed mapping algorithm. This is because in the case of an undecodable error the correct codeword will be transformed to an erroneous undecodable one w_u with $d_H(c_c, w_u) = 1$. Then, at the decoding, the erroneous undecodable word w_u will need to be first corrected to a predicted decodable codeword c_d with $d_H(w_u, c_d) = 1$. This yields that $d_H(c_c, c_d) \leq 2$. Consequently, since the mapping is optimized for only one substitution per word and the Hamming spheres are created to have a radius of 1, this will reduce the performance of the mapping algorithm.

However, the creation of only decodable errors is not a realistic scenario and thus we performed a second more realistic experiment in which we also created undecodable errors, always by creating a random substitution to a percentage of selected codewords. This new experiment is presented in figure 3.11b and depicts the improvement of PSNR in function of different error rates comparing the controlled and non-controlled mapping using SCD for correcting the undecodable words. As expected the undecodable errors decreases the improvement of PSNR providing a best case of ΔPSNR= 3dB.

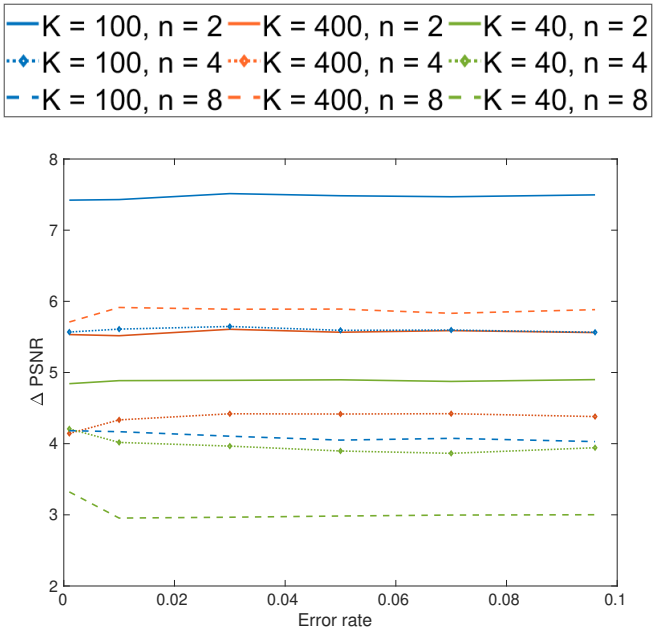
To improve those results and since SCD is a simple decoding where the undecodable word is corrected to the codeword mapped to the median vector-index of the quantized input image, we also tested the case of ACD. The result is shown in figure 3.11c which depicts the improvement in PSNR comparing the controlled mapping using ACD to the non-controlled mapping using SCD. This case reveals the total improvement which can be achieved providing a ΔPSNR between 2.5 and 5dB. On the right column of figure 3.11 we also present the visual improvement for each of the previous cases on the input image which was used in our experiments. The visual results are presented for an error rate of 3%, which is the percentage of substitutions introduced by the sequencer of the Nanopore (according to a study in [40]), and for the values of K and n which provided the best PSNR improvement in each case. The resistant to noise controlled mapping algorithm proposed in section 3.5 as well-as the proposed decoding of undecodable words described in section 3.6 have been published in [41].

To further test the efficiency of the proposed controlled mapping we have also performed an experiment using a discrete wavelet transform (DWT) for the decomposition of an image to 3 levels and quantizing each of the subband independently using a VQ. This particular encoding workflow using wavelets will be further studied and analysed in later sections. We nevertheless present here the results of the efficiency of the proposed noise resistant mapping. In our experiment, we encoded an image using a simple mapping without repetition and the proposed controlled mapping. For both mapping cases we used the encoding workflow briefly described above, reaching a compression rate of $R = 14.81$ bits/nt. The encoding produces two different representations according to the mapping which has been use in each case. We then added substitution errors to the same positions of the encoded strands. For the decoding of any undecodable codewords which might have occurred after the introduction of noise, the ACD decoding method has been

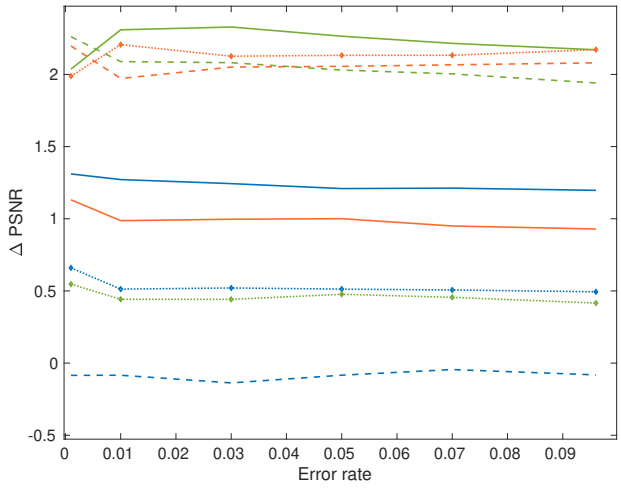
applied in both mapping cases. The visual results of this experiment are shown in figure 3.13 where it can be seen that the improvement of the PSNR when using the proposed controlled noise-resistant mapping is 2dB. Those results are very promising and encourage us to extend this work in our future studies to the use of the Levenshtein distance, in order to also allow the algorithm to deal with insertions and deletions.

3.8 Conclusion

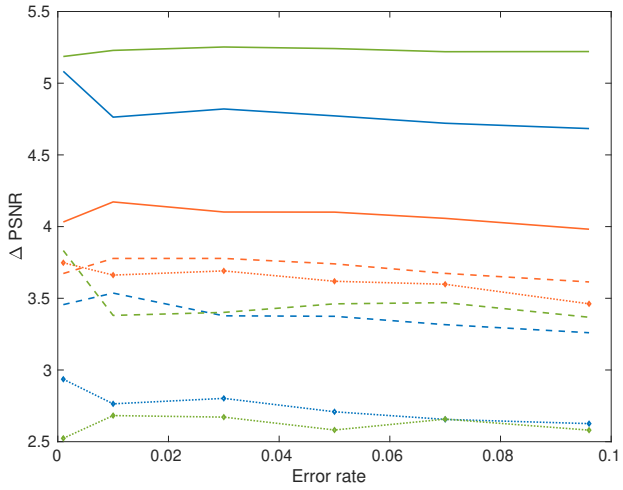
In this chapter we have introduced our proposed solution for generating a constrained codebook of DNA codewords which respect the biological restrictions of DNA sequencing. The proposed code-construction algorithm (PAIRCODE) can be used for implementing fixed length encoding solutions and achieves a coding potential which is comparable to the most well-known DNA coding methods among the state of the art. To exploit any redundant codewords of the code which is generated using PAIRCODE we have introduced a mapping function which performs a one-to-many assignment of source symbols to quaternary codewords of the code. This assignment allows dealing with the problem of pattern repetitions which can occur in the case that a source symbol is repeated many times in a row. Consequently, even though PAIRCODE does not outperform all existing encoding methods, it has the great advantage of being applied to any type of data without being constrained to binary inputs while also ensuring that there are no repeated patterns in the encoded strand. Summing up all the assets of our proposed encoding, it is interesting to denote that such an algorithm would be appropriate to be used in image coding: unlike previous works which use transcoding, it can be directly applied to encode sparse input data such as the quantized coefficients of some transforms. In addition to this, it can also avoid the pattern repetitions that can occur due to the consecutive occurrence of the most frequent symbols as in the case of quantized data. Consequently, this algorithm is a good candidate to be embedded to a workflow for image coding for DNA data storage. In the next chapter we present our proposed solution for implementing such a workflow which is useful for the efficient compression of digital images so to control and reduce the high DNA synthesis cost. Finally, since DNA sequencing is an error-prone process we propose an alternative mapping function which is resistant to the noise of substitutions which can be caused by sequencers and can be applied in the case that the source belongs to the space of vectors of length greater than 2. The proposed solution significantly reduces the impact of substitution noise in the visual quality of the decoded images.



a) Controlled mapping vs.non-controlled mapping (decodable errors only)

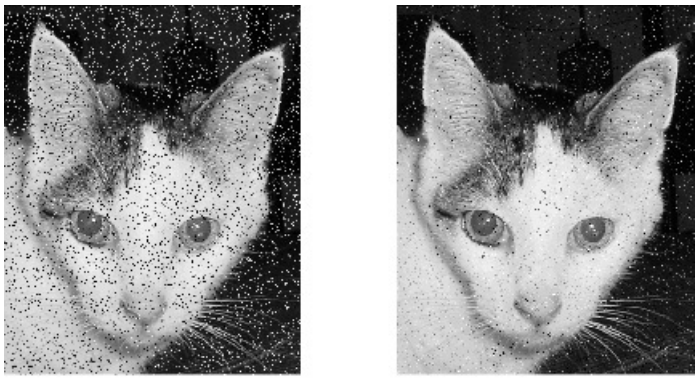


b) Controlled mapping SCD vs. non-controlled mapping SCD.



c) Controlled mapping ACD vs. non-controlled mapping SCD

Figure 3.11: PSNR improvement in function of the error rate for different cases of VQ parameters K and n. The three different plots correspond to different experiments as explained in the legends a, b and c.



Non-controlled mapping
PSNR = 15.63 dB

Controlled mapping
PSNR = 23.07 dB

a) Controlled mapping vs.non-controlled mapping (decodable errors only)

Non-controlled mapping + SCD
PSNR = 16.72 dB

Controlled mapping + SCD
PSNR = 19.12 dB

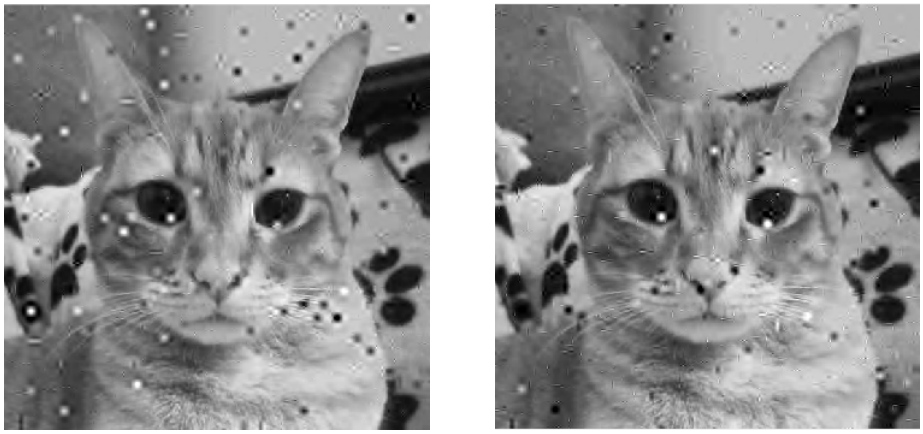
b) Controlled mapping SCD vs. non-controlled mapping SCD

Non-controlled mapping + SCD
PSNR = 16.61 dB

Controlled mapping + ACD
PSNR = 21.61 dB

c) Controlled mapping ACD vs. non-controlled mapping SCD

Figure 3.12: Visual results for each of the cases a, b and c for the values of K and n that provided the best performance. The selected error rate for those images is 3% accordingly to the percentage of substitutions caused by the Nanopore sequencer [40].



Simple Mapping (ACD)
PSNR=25.524 dB

Controlled Mapping (ACD)
PSNR=27.281 dB

Figure 3.13: Results of the use of controlled mapping algorithm vs. the use of simple mapping when applied to a workflow containing DWT and Vector Quantization. For the encoding we apply a 3-level DWT and quantize each subband independently using Vector Quantization. We then map the quantized codevectors to codewords of a code produced by our PAIRCODE algorithm using simple or controlled mapping. The encoded strands have been then corrupted by substitution noise inserted at the same positions of the strand. The figure depicts the impact of the error for the two different mapping cases.

Chapter 4

Design of a closed-loop DNA-based image coder

4.1 Introduction

One of the main goals of this thesis is to propose a full encoding workflow for the robust and efficient encoding of digital images into synthetic DNA. While multiple existing methods in the bibliography have been tested on image coding there is still room for further study and improvement. One of the greatest limitations of DNA data storage being the high cost of DNA synthesis, it is necessary to efficiently compress the input data before the encoding so to reduce the expenses.

Previous works until today have been transcoding binary information from many types of data into DNA, without taking into consideration the data characteristics. Thus, when encoding an image into DNA, the state of the art works have been using the protocol of JPEG which optimally compresses an image given some arbitrary target rate of compression expressed in bits/pixel. Then, the output bitstream of JPEG is transcoded to a quaternary stream leading to an "open loop" coding solution. In other words the compression is not optimized according to some quaternary encoding but is instead compressed according to a binary encoding. The implementation of a closed-loop transcoding algorithm for JPEG would be an option for further improving compression but such an encoding would be computationally expensive. As the DNA synthesis process is an expensive process requiring several dollars per synthesized oligo, it is fundamental to control this cost by including the compression into the process of encoding. Therefore, in our approach we don't use some fixed black box of compression. We rather designed a new codec (fully controllable) embedding a quaternary encoder leading to a "closed loop" coding solution so to be able to perform an optimal source allocation algorithm that can optimally compress the image given a target rate expressed in nucleotides/pixel (or bits/nucleotide) and not in bits/pixel. Indeed, our approach doesn't perform "transcoding" from binary to quaternary (as in the case of the state of the art) but rather we are directly encoding using a quaternary code the quantized coefficients allowing optimal rate/distortion control. Therefore, the encoding process in DNA is considered as a main factor for the optimal compression, something which is not possible when using the standard JPEG to compress.

In the next sections we present our proposed encoding workflow for the

efficient encoding/decoding of digital images into synthetic DNA while controlling the synthesis cost by optimizing the parameters used for the compression to provide the best possible visual quality.

4.2 A novel image coding/decoding solution into synthetic DNA

In this section we present our first proposed encoding for storing digital data into DNA. As explained in section 2.4, due to the high cost of DNA synthesis it is necessary to compress the data before encoding it into a quaternary representation of A, T, C and G. Previous works have used the JPEG protocol for compressing images before storing them into DNA. However, JPEG provides a compression quality which has been optimized for a given bit rate and therefore for the storage of binary bits rather than nucleotides.

In our study we propose including the compression process in the general workflow and applying a source allocation algorithm, which we will thereby call nucleotide allocation, to provide a closed loop solution by computing the optimal compression parameters given a predefined nucleotide rate. In other words, instead of using an existing compression protocol which is built for source coding, we introduce a simple step-by-step compression process which is constructed to deal with DNA coding. Allocation is an iterative algorithm which can be time consuming and computationally complex. To this end, we introduce a new efficient algorithm which produces a fixed-length quaternary code and facilitates the job of the nucleotide allocation. In contrast to existing coding methods in the bibliography the proposed encoding can be applied to any type of input data, binary or not. Furthermore, since pattern repetition is one factor which can cause errors during sequencing we also introduce for the first time a novel mapping algorithm for assigning to the input values the quaternary words of a fixed length code in such a way so to avoid creating patterns during the encoding.

The proposed compression workflow uses the following steps. To begin with a DWT is applied for reducing spatio-temporal redundancies on the input image providing a set of DWT subbands. Those subbands are then independently quantized using some quantization method (scalar or other) and separately encoded into a quaternary representation using a novel efficient algorithm for building the quaternary code. A nucleotide allocation algorithm for computing the optimal quantization parameters to provide the best possible image quality for a given compression rate expressed in bits/nt or nt/pixel. This predefined compression rate is equivalent to controlling the DNA synthesis cost and can therefore allow an estimation of the best quality/cost trade-off. This optimization is fundamental given the high cost of DNA synthesis which remains one of the main drawbacks of DNA data storage.

Once the quatization parameters have been selected by the allocation , they are then used for compressing the DWT subbands and encoding them independently into long strands of A, T, C, and G. As explained in section 2.1.2 the biological process of DNA synthesis limits the oligo size to 100-300 nts to promise reliability. Thus, after encoding each of the encoded subband strands is cut into smaller oligos of predefined size and formatted by adding any necessary headers to form the set of final oligos, ready to be synthesized!

The set is then sent to some company or laboratory of biology for synthesis. The produced DNA is then stored in safe capsule-sized containers that guarantee to keep it safe from corruption for centuries or even longer.

4.3 Source Allocation

The purpose of source allocation is to select the optimal compression parameters for the allocation of a given number of units of source information so to achieve a minimum compression distortion. When applied for encoding input data in binary representation this algorithm is called bit allocation. In this thesis we apply a source allocation for the encoding into a chain of nucleotides and will therefore call it nucleotide allocation.

The most well-known protocols used for image compression is JPEG and JPEG 2000 which use transform coding to enable better (more targeted) quantization. Those protocols are using bit-allocation to optimally quantize the DCT or DWT subbands accordingly of an input image into a compressed binary stream. As previously mentioned, inspired by standard of JPEG 2000, in this work we select the DWT transform for the compression but any other similar transform (such as DCT) could also be applied. DWT decomposes the input image into subbands of different resolution to capture spatio-temporal information. Thus, the main purpose of nucleotide allocation is to compute the optimal quantization parameters of each DWT subband to obtain the maximum quality in the reconstructed image.

In general the allocation of coding resources has been early recognized as a key issue in transform coding and in subband coding problems. A first analytical approach came by Huang and Schultheiss, who stated the theoretical optimal bit-rate allocation for generic transform coding in the high-resolution hypothesis [42]. They proposed allocating the optimal bit-rate to each set of data depending on their variances. While simple and elegant this interesting solution only holds when a high rate is available for encoding.

Shoham and Gersho have then proposed in [43] an optimal algorithm with no such restriction on the target bit-rate; however, it requires the computation of the RD characteristics for each possible quantization step, and thus its computational complexity is high. Ramchandran and Vetterli presented in [44] an RD approach to encode adaptive trees using generalized multiresolution wavelet packets. For the general case, an analytical expression of optimal rates has not been found, and different approaches have been applied. One of the most successful theories often used for solving the problem consist in modelling the relationship between the RD characteristics of the data sets and the global RD characteristics. The theory aims at minimising the distortion for a given maximal rate of the reconstructed data, by optimally allocating source units such as bits or as in this case nucleotides, to the various classes of data, which in our case will be the different subbands. For the case of orthogonal subband coding, Gersho and Gray showed in [45] that the global distortion D_T can be expressed as a sum of the sb subband distortions D_{sb}:

$$D_T(R_T) = \sum_{sb=1}^{SB} D_{sb}(R_{sb})$$

Usevitch has then extended this approach to the case of biorthogonal wavelet transform [46], modifying the above relation using some weights ω_{sb} which account for the non-orthogonality of the filters:

$$D_T(R_T) = \sum_{sb=1}^{SB} \omega_{sb} D_{sb}(R_{sb}) \tag{4.1}$$

The problem of source allocation can then be expressed as follows:

$$\min_{R_{sb}} D_T = \sum_{sb=1}^{SB} \omega_{sb} D_{sb}(R_{sb}), \text{ w.r.t. } R_T = \sum_{sb=1}^{SB} a_{sb} R_{sb} \leq R_{target} \tag{4.2}$$

where a_{sb} is the fraction of total pixels of subband sb. Namely, if P_{sb} is the number of pixels in the subband sb:

$$a_{sb} = \frac{P_{sb}}{\sum_{k=1}^{SB} P_k} \tag{4.3}$$

In other words, the goal is to minimize the total distortion with respect to some defined total target rate R_{target}. In the above equation the rate R_{sb} of each subband sb is parametrized by a set of quantization parameters $\mathcal{Q}_{sb}$ of the quantizer used for the compression. Hence, the problem can be expressed as:

$$\min_{\mathcal{Q}_{sb}} D_T = \sum_{sb=1}^{SB} \omega_{sb} D_{sb}(R_{sb}(\mathcal{Q}_{sb})), \text{ w.r.t. } R_T = \sum_{sb=1}^{SB} a_{sb} R_{sb}(\mathcal{Q}_{sb}) \leq R_{target} \tag{4.4}$$

It is therefore clear that one fundamental step for solving the problem of allocation is the correct computation of the R-D curves of the different subbands. One simple but computationally expensive approach for this computation is the use of a "brute force approach" which computes the R-D curves point by point. To simplify the computations later works in [47] propose representing the scalar-quantized coefficients using generalized gaussian models. An analytical solution to both rate allocation and distortion allocation problems can then be given by defining a model for subbands R-D curves, in order to lower the computational complexity of the algorithm while improving its accuracy. This approach tries solving equation 4.2 by introducing the lagrangian functional as presented by C. Parisot in [47], resulting in the following relation:

$$J(R, \lambda) = \sum_{sb=1}^{SB} \omega_{sb} D_{sb}(R_{sb}(\mathcal{Q}_{sb})) + \lambda \left(\sum_{sb=1}^{SB} a_{sb} R_{sb}(\mathcal{Q}_{sb}) - R_{target} \right) \tag{4.5}$$

By imposing the zero-gradient condition, it is found that the resulting optimal rate allocation R_{opt} verifies the following equation:

$$\frac{\omega_{sb}}{a_{sb}} \frac{\partial D_{sb}(R_{sb}(\mathcal{Q}_{sb}))}{\partial R_{sb}(\mathcal{Q}_{sb})} \Big|_{R_{sb}(\mathcal{Q}_{sb}) = R_{opt}} = -\lambda \tag{4.6}$$

Thus, the optimal rates correspond to points having the same slope on the "weighted" R-D subband curves. To compute the slopes T.André proposed in [48] the modeling of the R-D curves using splines. This solution is one of the solutions used in our studies and will explain it in more detail in the next sections.

4.4 Quantization

As described in the above sections the efficient compression of the input image is necessary in order to reduce the synthesis cost. Compression is performed using a quantization step after a DWT decomposition which quantizes the image coefficients of each subband independently, using the desired parameters to determine the quantization quality. The number and the type of parameters vary according to the type of Quantizer selected. In our study we carried out experiments using two different types of quatizer, Uniform Scalar Quantizer and VQ. The detailed study will be explained in the next sections. While quantization serves to the reduction of the synthesis cost, compression can lead to the creation of uniformity in blocks of neighboring coefficients and can therefore create pattern repetitions in the encoding. Consequently, even though in section 3.5.2 we have presented all the possible applications of the mapping algorithm according to the encoding needs, when applied to image coding the mapping should be selected so to treat pattern repetitions. The proposed value for the number of different codewords to represent each coefficient value is therefore $m \geq 2$. Furthermore since the compression should be optimal to provide the best possible quality it is clear that the encoding should take advantage of the full length of the code. Thus, the mapping algorithm should respect the following constraints:

- The number L of codewords in the code $\mathcal{C}^*$ and the number K of indices of quantized subband coefficients to encode in Σ should be related as $L \geq 2K$.
- To achieve the best possible quality for a given coding potential defined by L there should be no codewords in $\mathcal{C}^*$ left unused. Consequently the mapping of symbols to codewords should be done according to figure 4.1.

In the following sections we will present the encoding and the selection of the preferred quantization parameters according to each type of quantizer used in our experiments. It is important to denote that other types of quantization could also provide interesting results. Such an example would be the use of Lattice VQ as presented in [49] and [50] in which the approximating vectors are centroids of Voronoi polyhedrons of same size and shape which are forming a lattice. Interesting applications of Lattice quantizers on image coding with wavelet transform and rate-distortion models have been presented in [51] and [52].

4.4.1 Nucleotide allocation using Uniform Scalar Quantization

The simplest and most low-cost quantizer for testing the performance of our proposed encoding is the Uniform Scalar Quantizer with no dead-zone. The

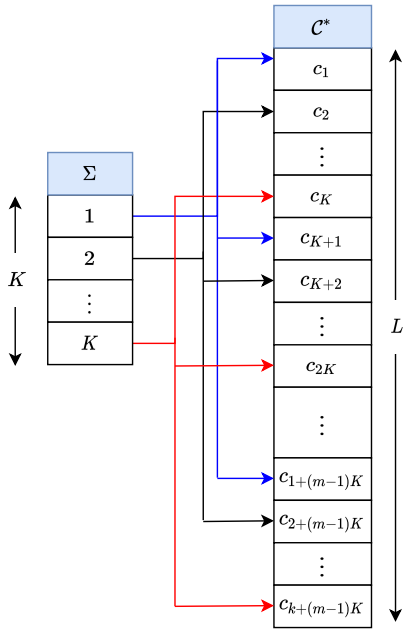


Figure 4.1: Optimized Mapping of quantized values into quaternary code for image coding application

scalar quantizer has only one parameter to define the quantization quality and that would be the quantization step-size q. Consequently, in this case, optimization of the compression using a nucleotide allocation algorithm can be achieved by selecting an optimal quantization step-size q_{sb} for each of the different subbands sb with $4 \leq b \leq SB$ and $SB = 3 * (\#\ell) + 1$ of an $\#\ell$-level DWT, which minimizes the compressed image's distortion D_T for a given target compression rate R_{target}. In other words, when used along with a uniform scalar quantizer, the problem addressed by nucleotide allocation can be described by equation 4.4 where $\mathcal{Q}_s = \{q_{sb}\}$. More precisely, the source allocation will estimate the optimal combination of subband rates R_{sb} which give a final total rate $R_T = R_{target}$ while providing a minimum subband distortion D_{sb} and therefore a minimum total distortion D_T. The encoder and decoder in the case where DNA coding is applied on image compression that uses a scalar quantizer is described in figure 4.2.

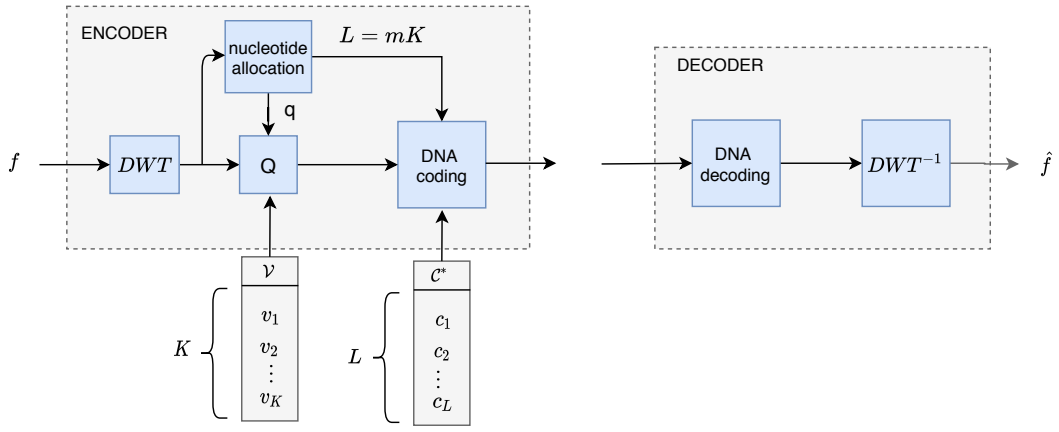


Figure 4.2: The general DNA encoding schema for image coding

Nucleotide allocation using splines approximation

As introduced in section 4.3, to solve the problem of optimal source allocation he main principle of this solution is to build the Rate-Distortion (R-D) curves for each of the subbands of DWT by varying the value of the quantization parameters at a sufficient range and computing this way the global R-D curve for the reconstructed image. One of the main goals of this solution is to correctly build the R-D curves. An interesting and efficient low-complexity solution is the one proposed by André in [48] which suggests approximating the R-D curves using splines to avoid computing many points in the R-D curve. We recall that the allocation problem can be solved using equation 4.6. In this work we used the DWT 9/7 which is biorthogonal with weights $\omega_{sb} \approx 1$. Furthermore since in this experiment we use a Uniform Scalar Quantizer, the allocation is only parametrized by the quantization step size q.

As explained in the previous section the allocation can be performed by solving the following equation:

$$\frac{\partial D_{sb}(R(\mathcal{Q}_{sb}))}{\partial R(\mathcal{Q}_{sb})} \Big|_{\mathcal{Q}_{sb}=(\mathcal{Q})^*_{sb}} = -\lambda a_{sb} \tag{4.7}$$

The solution of the above equation is given by computing the slope on a continuous R-D curve. However, since the rate R_{sb} of a subband sb is taking discrete values, the corresponding curve is not continuous. Thus, for solving the above equation, we need to approximate the continuous curve using smoothing splines and then compute an approximation of the desired slope λ for two close consecutive points $D(R_1)$ and $D(R_2)$ on the smoothed R-D curve as:

$$\lambda = \frac{D(R_2) - D(R_1)}{R_2 - R_1} \tag{4.8}$$

We compute the smoothed R-D curve and solve the allocation problem for some given value of λ for each subband of the DWT as follows:

- Select r different decreasing values for the quantization step-size q, quantize and encode the subband coefficients to obtain r different points on the R-D curve. As explained in section 3.3, the code $\mathcal{C}^*$ has a defined dynamic. Thus, given the relation between the number K of indices to encode in Σ and the size of the code L, the number of codewords that are left unused will vary. Consequently, different values of quantization step q_{sb} may require the same code size L for the encoding, thus producing the same encoding rate R_{sb}. This means, that there might be multiple values of MSE achieved for a given rate as depicted in figure 4.3. Since the main goal of nucleotide allocation is the minimization of MSE for a given target rate, we select for each rate the point that provides the lowest MSE value lying on the convex hull of the R-D curve (black line in figure 4.3)

- Create a smoothed approximation of the rough R-D curve obtained before using smoothing b-splines approximation (red curve in figure 4.3).

- Select the point on the smoothed curve which has the closest slope to the value of λ. To accelerate the computations, the method of dichotomy can be used for locating the desired point.
- The selected point provides an approximation of the desired rate R^*_{sb}. The desired subband rate R^*_{sb} is expressed in nucleotides/coefficient and is equal to the desired codeword length l^* of the code. Then, it is easy to compute the corresponding desired size L^* of the code C^* as:

$$L^* = \begin{cases} 10^{\frac{l^*}{2}}, & \text{if } l^* \text{ is even} \\ 10^{\lfloor \frac{l^*}{2} \rfloor} * 4, & l^* \text{ is odd} \end{cases} \tag{4.9}$$

- Using L^* one can derive the number of quantization levels and using the maximum and minimum values of the subband coefficients the desired quantization step-size q^*_{sb}.
- Performing the encoding using this estimated value of q^*_{sb} the real values of R_{sb} and D_{sb} are obtained and constitute one point on the global R-D curve of the reconstructed image.
- Repeating the same procedure for many different values of lambda if needed a complete global R-D curve can be obtained which contains the optimal parameters for many different target rates R_{target}.

The performance of the above nucleotide allocation algorithm has been tested on 512x512 pixel image of Lena providing the R-D curve depicted on the left side of figure 4.4.

Nucleotide allocation using permutations

The method described in the previous section is using splines to approximate the R-D curve of each subband and thus the parameters derived from the allocation will provide a total rate R_T which is close to the desired target rate R_{target} but not exactly equal. In this section we will present a new algorithm which computes the quantization parameters so to provide the real R_T values which are predefined by the user as an input.

Let $R_{sb} = \{0, 2, 3, \ldots, N\}$, be the set of all the possible values of subband rates where R_{sb} is computed in nucleotides per subband-coefficient (nt/coef) and $|R_{sb}| = N$. It is clear that a subband rate $R_{sb} = 1$ nt/coef is omitted because our proposed code construction algorithm (explained in section 3.2.2) respects the sequencing constraints by using specific pairs of symbols to form viable quaternary words. Therefore, the length l of the produced words will be at least 2 nt. This means that any subband can either not be encoded at all and thus $R_{sb} = 0$ nt/coef or $R_{sb} \geq 2$ nt/ coef. The problem of allocation can be more simply expressed as finding the best possible distribution of rate-values R_{sb} to the SB different subbands sb such that:

- The total rate R_T of the reconstructed image is equal to the target rate R_{target}.
- The total distortion D_T is minimized.

As explained in section 4.3, one way for solving the problem of allocation is the brute force approach by finding all possible points of the R-D curve for each subband to optimally distribute resources to the different subbands. Apart from the method of building the R-D curves and computing the target points using the slopes of the curves, in this section we will propose a new exhaustive method for performing the allocation without building the R-D curves. The proposed method is computationally expensive but has the advantage of computing the real optimal values of R_{sb} with no approximations and thus the exact target rate R_{target} defined in the input.

The above problem can be solved using a problem of ordered arrangements (permutations) by the following steps:

- Find all the possible ways for filling SB different positions (different subbands), with one of the elements $R_{sb}^{p} \in R_{sb}$ (possible subband rates) where $sb = 1, 2, \ldots, SB$ and $p = 1, \ldots, N^{SB}$. The solution is given by the following matrix P_t which is composed by the N^{SB} different permutations.

$$P_T = \begin{bmatrix} R_1^1 & R_2^1 & \ldots & R_{SB}^1 \\ R_1^2 & R_2^2 & \ldots & R_{SB}^2 \\ \vdots & \vdots & \ddots & \vdots \\ R_1^{N^{SB}} & R_2^{N^{SB}} & \ldots & R_{SB}^{N^{SB}} \end{bmatrix} \tag{4.10}$$

  where the columns represent the different subbands and the lines the different permutation cases. In other words, this matrix contains all the possible allocations which provide any possible total rate R_T. By encoding an input image using all the possible permutations existing in P_T to encode the different DWT subbands we get a cloud of points as depicted in figure 4.5.

- The solution of the allocation problem is found among the lines of P_T which give a total rate of $R_T = R_{target}$. The next step to solve the allocation problem is therefore to reduce the matrix P_T to a new matrix P_R where all lines of which provide a total rate equal to R_{target}.

$$P_R = \begin{bmatrix} R_1^1 & R_2^1 & \ldots & R_{SB}^1 \\ R_1^2 & R_2^2 & \ldots & R_{SB}^2 \\ \vdots & \vdots & \ddots & \vdots \\ R_1^G & R_2^G & \ldots & R_{SB}^G \end{bmatrix} \text{ with } G < N^{SB} \tag{4.11}$$

More precisely, the solution is given by the line of P_R which provides the minimal MSE value. Hence, to detect the optimal line of permutations it is necessary to encode the image using the encoding provided by the different lines of P_R. More precisely the quantization step size for the encoding is given by:

$$q_{sb} = \frac{m(v_{sbmax} - v_{sbmin})}{L_{sb}} \quad \text{with} \quad L_{sb} = \begin{cases} 10^{\frac{R_{sb}}{2}}, & \text{if } R_{sb} \text{ is even} \\ 10^{\lfloor \frac{R_{sb}}{2} \rfloor}, & \text{if } R_{sb} \text{ is odd} \end{cases}$$

where, v_{sbmin} and v_{sbmax} are the minimum and maximum respectively values of the coefficients of subband sb.

Encoding the SB subbands using the different permutations of rates (lines) of P_R we end up with G different cases of encoding. The solution of the nucleotide allocation is the line of P_R which provides the minimum MSE and thus the best visual quality in the reconstructed image.

4.4.2 Nucleotide allocation for Vector Quantization

In the previous sections we have presented the proposed techniques for optimally compressing an image to be efficiently encoded into DNA using a Uniform Scalar Quantizer. To further improve the performance of the compression in this section we propose the use of a VQ for the quantization. Similarly to the SQ case, we will perform a VQ on the coefficients of the subbands $sb \in \{1, 2, \ldots, SB\}$, with $SB = 3i + 1$ produced by an $\#\ell-$level DWT decomposition. More precisely, the purpose of VQ is to map n-dimensional vectors in the vector space $\mathbb{R}^n$ into a finite set of vectors $\mathcal{V}$ (the codebook) defined here as:

$$\mathcal{V} = \{\hat{v}_k \in \mathbb{R}^n | k = 1, 2, \ldots, K\}$$

The quantization algorithm is described by the following relation:

$$Q : \mathbb{R}^n \rightarrow \mathcal{V} \subset \mathbb{R}^n$$

The codebook $\mathcal{V}$ contains the centroid vectors that have been computed after clustering a set of training vectors according to the generalized Lloyd algorithm presented in [53]. For an efficient quantization, the training vectors used for creating the codebook should come from a set of images which have a similar content to the one of the image to be encoded.

As explained in section 4.3, the nucleotide allocation aims in the selection of the optimal set $\mathcal{Q}^*_{sb}$ of quantization parameters for each subband sb. In the case of VQ, the $\mathcal{Q}^*_{sb}$ contains the parameters K^*_{sb} and n^*_{sb} for the quantization of each subband sb. Hence, the quantization requires the use of a codebook of K^*_{sb} vectors of length n^*_{sb} which should be constructed using a training set of vectors coming from a large set of subbands of similar characteristics. To reduce the computational cost, it is preferable to build all possible codebooks of vectors for all different values of K_{sb} and n_{sb} in advance rather than computing it during the allocation. The codebook can be stored to be used for images of similar characteristics. It is important to denote that an efficient VQ requires a well-trained codebook. For the codebook construction we select a large-enough set of images with similar content to the input image. The images should then be decomposed into $\#\ell$ DWT levels to produce SB subbands. Each subband sb of the training images is then divided into a large set of training vectors of length n_{sb}. Then, by performing a k-means clustering, the set of training vectors is divided into K^*_{sb} clusters and a set of K_{sb} centroid vectors is computed to constitute the final codebook to be used for the quantization of the corresponding subband of the input image. The output of the VQ will be a matrix of vector indices that represent the different vectors in the codebook. Using the above method, one can construct the codebooks of any desired value of K_{sb} and n_{sb} for every level of DWT and every subband type of the training set. Consequently, when performing the nucleotide allocation, the codebook for each subband and any selected optimal value K^*_{sb} and n^*_{sb} can be directly retrieved. The proposed nucleotide allocation for

the encoding of an image into DNA using VQ is described by the following steps:

- The input image is decomposed into SB subbands using a DWT. Each DWT subband sb with $sb = \{1, 2, \ldots, SB\}$ is then quantized and encoded independently.
- A source allocation algorithm is used to find the set $\mathcal{Q}_{sb}$ of optimal parameters of VQ that solve equation 4.4 and minimize the distortion for a given target rate. However, similarly to the scalar case, since the subband rate R_{sb} is taking discrete values, equation 4.6 is not derivable so to allow computation of the desired slope λ on the R-D curve. To this end, we will need to first create a continuous approximation of the R-D curve for each subband sb. To build the approximation of the R-D curves for each subband sb we exhaustively compute the rate R_{sb} and the distortion D_{sb} produced by the vector quantizer for each set of parameters (K_{sb}, n_{sb}). Then, from the set of produced points on the R-D curve, we select the points lying on the convex hull. As the convex hull may not be smooth enough to allow correct computation of the slopes λ we use an exponential decay approximation which can provide a fair estimation of the slope at each computed point. Figure 4.8 shows a typical example of the R-D curve for one subband of a 9-7 biorthogonal wavelet decomposition along with the convex hull and its exponential approximation. Finally, by selecting the points that correspond to the desired slope for every subband one can select the optimal tuning, i.e. the values of K^*_{sb} and n^*_{sb} for $sb \in \{1, 2, \ldots, SB\}$, which provides the lowest distortion D_{sb} for a given subband rate R_{sb}. The global R-D curve for encoding an 512x512 pixel image of Lena is depicted in figure 4.9.
- Each subband is quantized according to the selected parameters. For the quantization, we select the codebook which has been already computed and corresponds to the same subband type and the selected optimal values of K^*_{sb} and n^*_{sb}. We then decompose the subband into K^*_{sb} vectors of length n^*_{sb} and quantize each one of them to the indice of the closest vector from the ones in the selected codebook. This last step results in a matrix of quantized vector indices.
- The matrix of indices is encoded using our proposed code construction algorithm described in section 3.2.2 to produce a quaternary stream of A, T, C and G.

A schematic representation of the VQ encoding/decoding workflow is depicted in figure 4.6. At this point it is necessary to discuss some important points concerning the application of VQ to DNA image coding. While VQ allows a very efficient compression in terms of coding potential, it is clear that the decoding of some image that has been encoded using the method of VQ requires knowledge of the codebook that has been used for the quantization. This yields that the codebook should be also stored into DNA to allow correct decoding of the stored image in the long-term. The storage of the codebook can be expensive if used for encoding only one image. However, in the case that the same codebook is used for encoding many different images the cost of the codebook storage is compensated.

Pattern repetition problem: VQ is an efficient way for compressing images as blocks of information are being packed into quantization vectors which are then encoded to a codeword of $\mathcal{C}^*$. It is also a solution which is adapted to the needs of the encoding since we are able to capture the source distribution while using a fixed length code. While this quantization is very efficient in terms of coding potential, the fact that more than one elements can be represented by the same codeword can easily cause the problem of pattern repetitions which is an ill-case for DNA coding as it can produce errors in the sequencing process. Figure 4.7 depicts an example of the same information encoded in an oligo using VQ with (bottom oligo) and without (top oligo) treating the case of pattern repetitions. Therefore, by applying a double representation to the most frequent vectors, as discussed in section 3.3, the pattern repetition can be avoided. This is due to the fact that in the case of repetition of the same vector one can alternate the two associated codewords increasing this way the robustness of the encoded sequence. More precisely, we propose two different cases of mapping to deal with patterns. In the first case which is depicted in figure 3.3c all the vector indices in Σ are mapped once to the K first codewords of $\mathcal{C}^*$. Then the p most probable vector indices from Σ are also mapped to a second codeword from the $L-K$ remaining vectors of $\mathcal{C}^*$. In the second case we ensure that all vector indices in Σ can be encoded by at least m different codevectors in $\mathcal{C}^*$ ($L = 2K$). In our experiments we used m=2. This second case of mapping is described in figure 3.3d. For further details in the encoding algorithm and the mapping process the reader can refer to section 3.3.

The global R-D curve for encoding an 512x512 image of Lena is depicted in figure 4.9.

This last curve depicts the results of comparison of our compression scheme which is using a scalar quantizer (section 4.4.1) and 4 different cases of VQ:

- Without treating patterns: Lena inside or outside the training set.
- Treating patterns (for Lena outside the training set):
  - Using only 80% of the code $\mathcal{C}^*$ for the encoding ($K = \frac{80}{100}L$) and using the 20% remaining codewords ($K - L$) to allow double representation of the most frequent quantized vectors.
  - Mapping 100% of the code ($L = 2K$).

The results show significant improvement in comparison to the results obtained by the scalar case. The above results on the storage of digital images into DNA using Vector Quantization have been published as an abstract for the DCC 2020 [54] conference and as a full conference paper in EUSIPCO 2020 [55].

4.5 Conclusions

In this chapter we have introduced a fixed-length encoder for the efficient compression of images into synthetic DNA. Our proposed solution allows controlling the synthesis cost by including the PAIRCODE algorithm presented in the previous chapter in a "closed-loop" source allocation which provides the optimal compression quality for a predefined coding potential.

For the compression of the images, in our experiments we tested two different alternatives of quantization. The first option uses a simple uniform scalar quantizer and to our knowledge constitutes the first attempt among the existing methods in bibliography which is allowing the code to be optimized with respect to the quaternary representation of the 4 DNA bases while also respecting the biological restrictions imposed by DNA sequencing. Even though this encoding produced interesting results in terms of providing an optimal code for a given compression rate, due to simplicity of the compression the coding potential which is achieved is not high enough. Thus, to enhance the performance of the compression we also tested the use of a Vector Quantizer for the process of quantization. This solution provided much more promising results as unlike scalar quantization, VQ captures the source thanks to the training of the codebook. However since VQ depends on the construction of a good codebook which has been trained using a good data set of images. Furthermore, this codebook has to be known to the decoder to allow correct reconstruction of the input image and therefore this increases the cost of the encoding. To this end, we have proposed a solution for reducing the cost of the codebook storage by using the same codebook to encode multiple images of the same content while storing it to some known codebook DNA database.

Finally, we discussed about the potential study of replacing VQ with a Lattice Vector Quantizer which is expected to provide even more promising results since it does not require transmission of the codebook to the decoder while exploiting at the same time the advantage of increased compression efficiency that can be achieved by VQ.

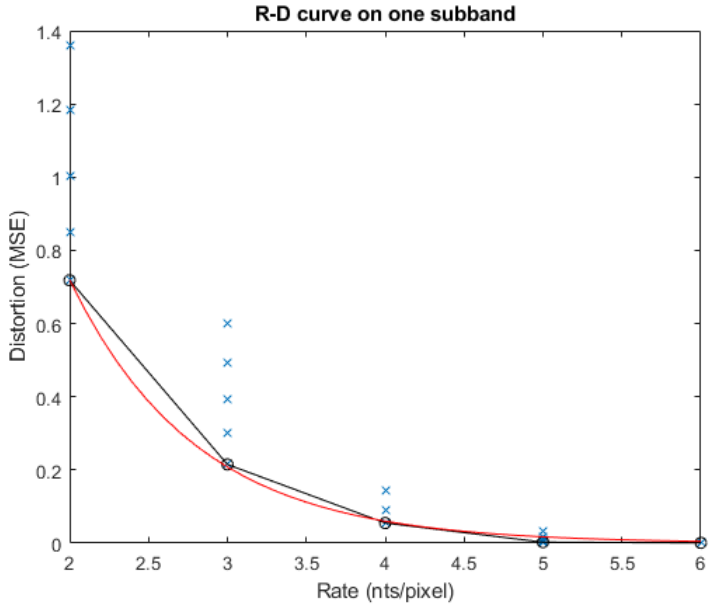
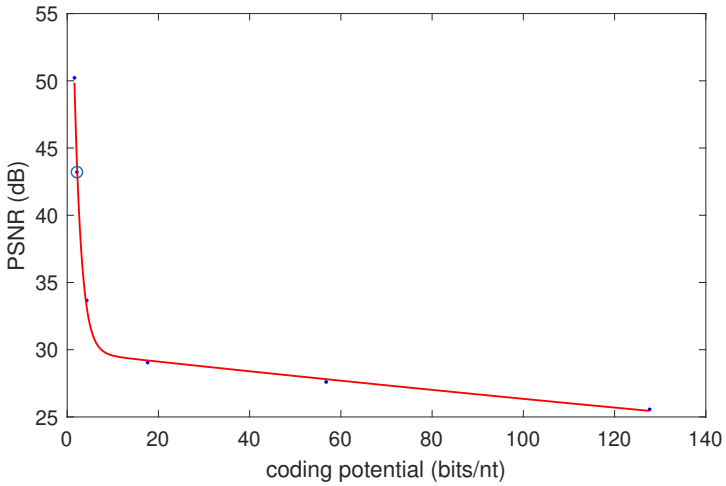


Figure 4.3: RD-curve of one subband sb as provided by the nucleotide allocation using Scalar Quantization. The 'x' points represent the different RD points obtained by the different quantization step-size values q_{sb}. The 'o' points represent the selected points for building the rough representation of the RD curve (in black). The red curve represents the smoothed RD-curve obtained using b-splines.



Visual quality:
PSNR = 43.21 dB
Rate = 2.14 bits/nt

Figure 4.4: Left: R-D curve provided by the nucleotide allocation using spline approximation for a 512x512 pixel image of Lena. The selected point corresponds to the visual quality of the image provided on the right side. Right: Visual quality of the selected point in the R-D curve.

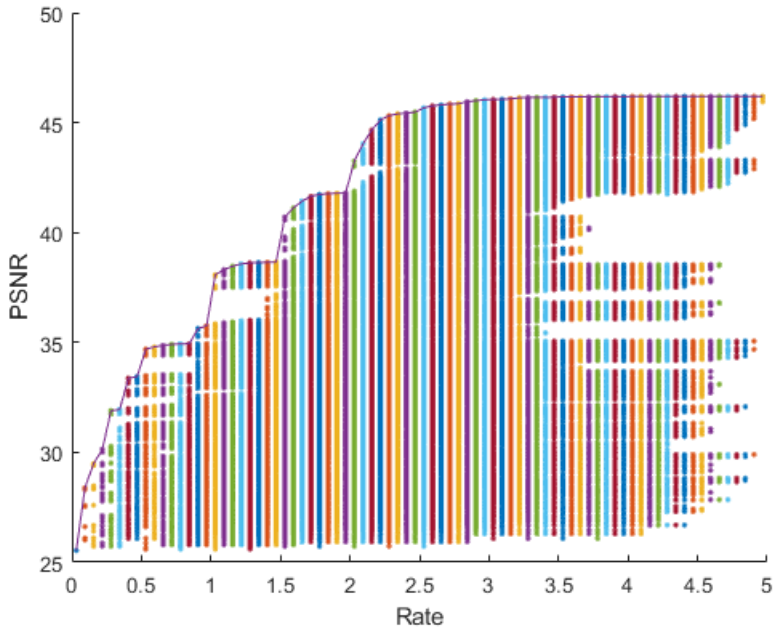


Figure 4.5: Example of a point cloud of values of Rate vs. PSNR when using all existing permutations of subband rates to encode some input image. The best values of PSNR for each encoding Rate are kept as the optimal encodings of the allocation method.

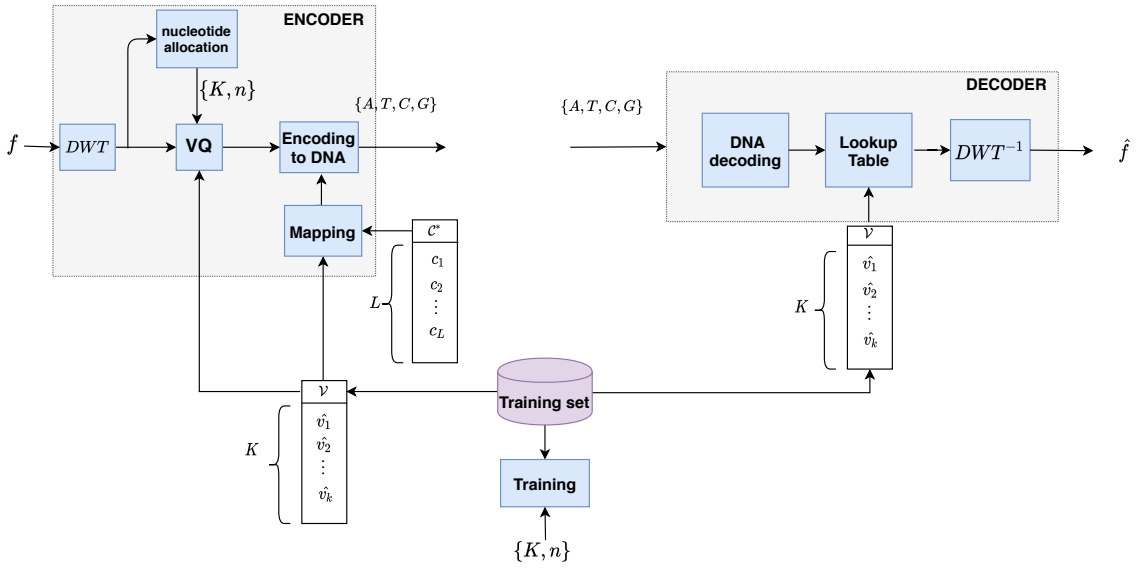


Figure 4.6: Coding/decoding workflow using Vector Quantization.

5' end of payload - ATGT ATAT ATAT ATAT ATGT ATGT GTGT GTAT ATAT ATAT GTGT ATGT ATGT ATAT ACAT ATGT GTAT ATGT - 3' end of payload
5' end of payload - TGAAG TTGAA GCATA TGATG ACTCT GATCG AGCTC GTCGG TGCTT TGACT CTGAA TAAGC CTTCT TATAG - 3' end of payload

Figure 4.7: Example of an oligo encoded using VQ with pattern repetitions (top oligo) as well as avoiding pattern repetitions (bottom oligo)

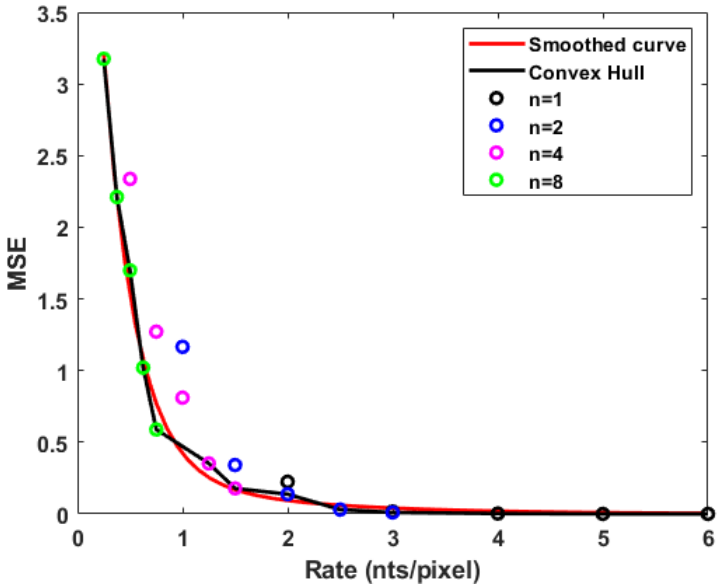


Figure 4.8: Behaviour of the rate-distortion curve in one wavelet subband of Lena image. Each point corresponds to different value of K_{sb} and n_{sb}. The convex hull is plotted in black and its continuous exponential approximation in red.

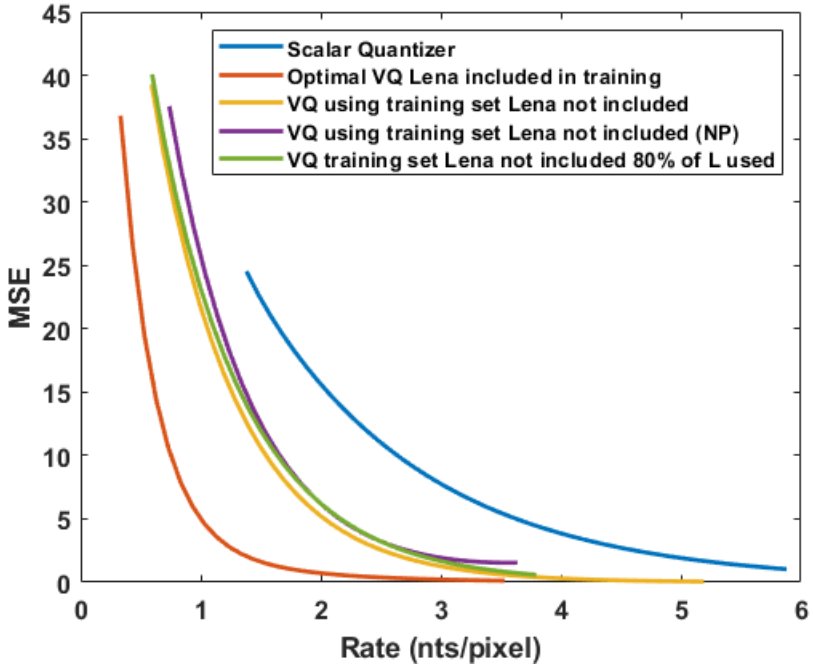


Figure 4.9: Comparison of the global rate-distortion curves for 5 different cases of Vector Quantization of Lena image. Blue: scalar quantizer, Orange: VQ, Lena inside the training set with pattern repetition ($K_{sb} = L_{sb}$ for all subbands sb). Yellow: VQ, Lena outside the training set with pattern repetition. Purple: VQ, Lena outside the training set no patterns (NP) with L=2K. Green: VQ, Lena outside the training set using only 80% of the code (K_{sb} is 80% of L_{sb} for all subbands sb).

Chapter 5

A JPEG-based coding algorithm for DNA image storage

5.1 Introduction

In the previous chapter we presented an encoding workflow for the efficient compression of an input image to be stored into DNA. Compression is controlled and optimised thanks to a nucleotide allocation algorithm which is applied to a fixed length encoding that uses the constrained code presented in section 3.2.2. As explained earlier, in our main study we have selected to implement and use a fixed-length code instead of a variable-length one as the latter is more prone to errors. Nevertheless, variable-length coding can provide better results in terms of compression ratio. This thesis focuses on the encoding of digital images into DNA. Inspired by the JPEG standard used for the compression of images in classical source coding, we propose a new variable-length encoding workflow for producing a constrained DNA code which is optimized using entropy-coding. Many state of the art works have used the JPEG algorithm to first compress an input image into a binary stream, then encoding it into a quaternary representation of A, T, C and G using some constrained DNA coding algorithm. While this practice can provide satisfying results it does not guarantee an optimized solution in terms of compression efficiency. This is due to the fact that the compression is performed using binary Huffmann codes and is thus optimized for a binary output. In this chapter, we make a very first attempt to implement a modified JPEG coding algorithm adapted to the needs of DNA data storage. The proposed algorithm uses the same general workflow as the classical JPEG one with the difference that the encoding is performed using constrained codes to be more robust to sequencing errors. In the next sections we will describe the algorithm and will compare its efficiency to our proposed fixed length workflow.

5.2 Theoretical background: The JPEG algorithm

The classical JPEG standard is using the following encoding steps. First, an input image is decomposed into blocks of size 8 by 8 as shown in figure 5.1. The blocks which are at the edges of the image are padded. Then a DCT is performed to each block and the produced coefficients are quantized using predifined quantization tables which have been computed based on psychovisual threshold experiments. The quantization table which is used

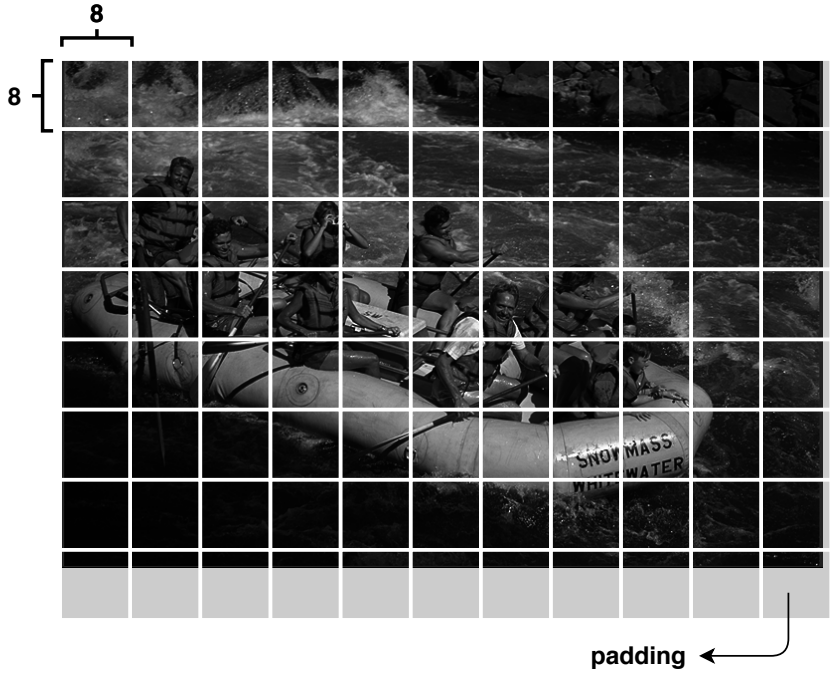


Figure 5.1: Block decomposition for JPEG

for luminance is given by the following matrix:

$$G = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 36 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \tag{5.1}$$

After quantization of the blocks two different processes follow to compute the AC and DC quantization indices. The DC quantization values are obtained using Differential coding with DPCM which computes the difference between the first element of each block and the first element of the previous block to obtain a vector of differences. The obtained sequence is then encoded using a variable length coding (VLC) that encodes each difference value according to the category to which the corresponding value belongs to. In more detail, the category of an element specifies the number of bits needed for one's encoding. Thus each difference value can be expressed as a couple [category, value] in which the category field is later encoded using Huffman codes and the value is transformed to the corresponding binary representation using the number of bits denoted by the category.

The AC quantization indices are computed by parsing each block using a zig-zag scanning to create a sequence of quantized coefficients from the input block. Each sequence is then encoded using a Run-level coding that counts the number of consecutive zero values and the category of the first

Range	Category (#bits used for encoding)
0	0
-1, 1	1
-5, . . . , −2, 2, . . . , 5	2
-7, . . . , −4, 4, . . . , 7	3
-31, . . . , −16, 16, . . . , 31	4
-63, . . . , −32, 32, . . . , 63	5
-127, . . . , −64, 64 . . . , 127	6
-255, . . . , −128, 128, . . . , 255	7
-511, . . . , −256, 256, . . . , 511	8
-1023, . . . , −512, 512, . . . , 1023	9
-2047, . . . , −1024, 1024, . . . , 2047	10
-4095, . . . , −2048, 2048, . . . , 4095	11
-8191, . . . , −4096, 4096, . . . , 8191	12
-16383, . . . , −8192, 8192, . . . , 16382	13
-32767, . . . , −16384, 16384, . . . , 32767	14

Table 5.1: Category range for encoding values using JPEG

non-zero element which follows the run of zeros. The category of a non-zero element depends on the value of the element itself and corresponds to the number of bits which are used for its encoding as shown in table 5.1. This process produces a sequence of couples [run/category, value] which are then encoded using VLC. More precisely, the run/category field is encoded using Huffman coding while the value is encoded by transforming it to its corresponding binary representation. The encoding of the values in binary is performed using a number of bits which is indicated by the category. The range of values which can be encoded by each category are predefined and known to both the encoder and the decoder. By concatenating the AC and the DC binary sequences the final output of JPEG is obtained. The general workflow of the classical JPEG standard is depicted in figure 5.2.

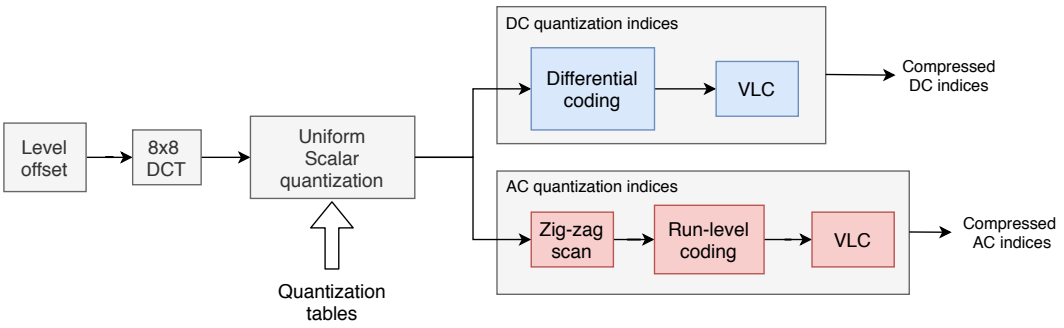


Figure 5.2: Workflow of the JPEG standard

5.3 The proposed variable-length DNA coding algorithm

Inspired by the main workflow of the classical JPEG standard in this section we present a modified version of this algorithm for the encoding of an image into a constrained quaternary representation of A, T, C and G. As explained in the previous section JPEG is using two main coding techniques to a binary representation. The first technique is the Huffman coding used to encode the run/category of the AC indices as well as the category of the DC indices. Huffman codes take into consideration the frequency of each run/category

AC element or category DC element throughout the full image and assigns binary words to the different values while ensuring that each binary representation is not a prefix of another. The main asset of Huffman coding is that the algorithm assigns the shortest words to the most frequent elements so to enhance the performance of the encoding in terms of compression. Then, the second method used in JPEG is a simple binary coding of the values of AC and DC indices. This encoding is simply transforming each value into its binary representation using a number of bits which is predefined by the category field that precedes. It is therefore clear, that in order to modify the existing JPEG algorithm so to provide a quaternary representation one would need to replace those two binary encodings by some appropriate quaternary ones while also respecting the encoding constraints of DNA coding (see section 2.2).

As described in section 2, in the works of Goldman et al. [1] the authors have presented an interesting algorithm for encoding an input sequence of symbols to a constrained DNA sequence. One of the main assets of this algorithm compared to other state of the art methods is that similarly to our proposed constrained codebook it can be applied to any type of input without being restricted to binary inputs. We recall that the algorithm proposed by Goldman works according to the following steps:

- The input sequence is encoded using a ternary Huffman into a stream of the trits 0, 1 and 2.
- Reading the ternary sequence, each trit is replaced with one of the three nucleotides different from the previous one used, ensuring no homopolymers were generated.

Hence, since the above algorithm is using Huffman codes it seems to be an interesting candidate to replace the binary Huffman encoding of the classical JPEG standard.

Then, for replacing the binary encoding of the non-zero values of the AC and DC indices of JPEG we propose using our constrained code presented in 3.2.2. This code provides a simple fixed length encoding of any input sequence of elements. It is a promising method that works similarly to the classical fixed-length binary encoding and provides a quaternary representation while respecting the sequencing constraints. The use of this code will change the range of the categories which will now determine the number of nucleotides rather than the number of bits that will be used for the encoding of an element. Since in our encoding a representation of 1nt per element is not permitted, category 1 is omitted from the list of possible categories as shown in table 5.2.

In other words, in the proposed variable length algorithm we have used the main workflow of JPEG by replacing the binary VLC with a quaternary VLC algorithm which is using the algorithm proposed in Goldman et al. instead of the binary Huffman and our constrained code instead of the classical binary encoding. The modified workflow is presented in figure 5.3. In the following section we provide experimental results on the performance of this modified JPEG code compared to the fixed length encoding which has been proposed in the previous chapter 4. We also provide the results of the transcoding method which encodes the binary output of a classical JPEG using a fixed length coding for encoding each byte of the JPEG binary stream.

Range	Category (#nt used for encoding)
0	0
-5,...,−1,1,...,5	2
-25,...,−6,6,...,25	3
-75,...,−26,26,...,75	4
-275,...,−76,76,...,275	5
-775,...,−276,276,...,775	6
-2775,...,−776,776,...,2775	7
-7775,...,−2776,2776,...,7775	8

Table 5.2: Category range for encoding values using DNA. Category 1 is omitted due to the biological constraints imposed by DNA sequencing.

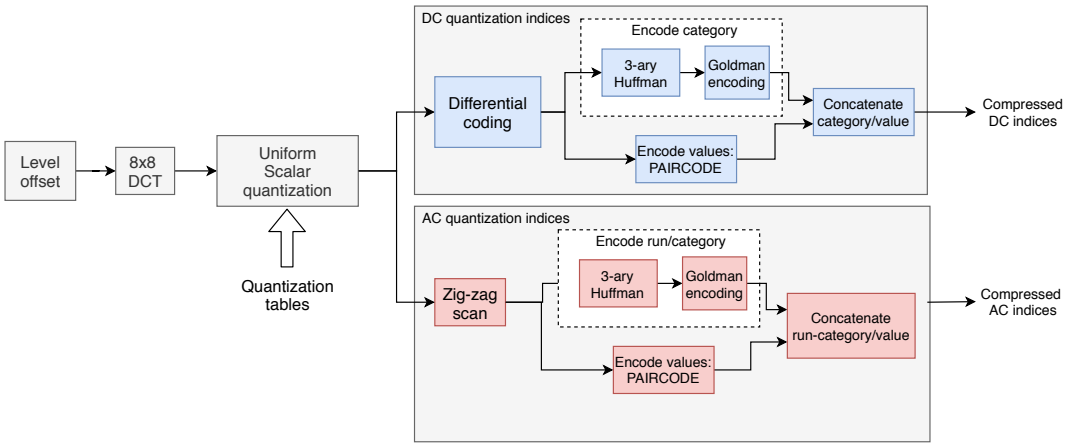


Figure 5.3: Workflow of the modified JPEG workflow to suit the needs of DNA coding

5.4 Comparison of the different encoding solutions

In this section we compare all the encoding methods proposed in this thesis. Namely, we will compare the two fixed-length encoding solutions using a Uniform Scalar Quantizer as proposed in section 4.4.1 and a VQ as proposed in section 4.4.2 to the variable-length encoding solutions based on the JPEG standard and more precisely our proposed "closed-loop" solution of inserting our code in JPEG and the simpler transcoding method. For the comparison we applied all the encoding cases on the image of a cat (see figure 5.4) and built the curve of PSNR in function of the coding potential in bits/nt. The training set of cat images used for the construction of the codebook of vectors for the case of VQ encoding is given in B. The test cat-image which has been encoded does not belong in this training set. The result of the comparison is given in figure 5.5. As observed, the fixed length encoding with the Scalar Quntizer has the less efficient performance. This result is justified by the fact that, as expected, fixed length solutions are less performant than variable length encodings and in the case of a Scalar quantization the encoder is not taking into consideration the distribution of the source and thus the compression is less adapted to the characteristics of the input image. However, this has been only our very first attempt to build a simple fixed-length encoder which allows controlling the compression rate to test the performance

Figure 5.4: Original image of cat which has been used in our experiment of comparing the efficiency of the different encoding methods

of our proposed PAIRCODE algorithm.

Interestingly enough, the fixed-length solution which is using a VQ for the quantization, provided results which are comparable to the ones of the variable-length solutions. This improvement in the performance is explained by the fact that in the case of a VQ the encoder uses some knowledge of the characteristics of the input. This information stems from the use of a well-selected training set of images, with similar characteristics to the input image, for the creation of a good codebook of quantization vectors. Furthermore, as explained in section 4.4.2, since the use of VQ can create strong patterns in the encoded strands due to the repetition of the most frequent coefficients, we have also included in this comparison a case in which 20% of the produced constrained code is dedicated to the double mapping of the most frequent input elements. In other words, the most frequent coefficient indices in Σ will be represented by two different codewords in $\mathcal{C}^*$ instead of only one. As expected, since this solution uses some extra redundancy to avoid patterns, this encoding scenario is slightly less efficient than the one of simple VQ which is not avoiding patterns.

We also observe that among the variable length solutions, simple transcoding shows the best results. In other words, even if the "closed-loop" solution of inserting the quaternary code into the code of JPEG to allow controlling the compression was expected to perform better, the result is different than predicted. The difference in the performance between the two variable length encoding methods is explained by the constraints imposed on the quaternary encoding. More specifically, in the proposed "closed-loop" solution the category 1 that is using 1nt to encode a value is omitted in order to avoid the creation of homopolymers. However, in most cases the values of category 1 are the most frequent ones, and thus in the transcoding case those values will be encoded by 1 bit while in the quaternary case they are encoded by

2 nts. This explanation becomes even more apparent when checking how the difference in both cases' performance evolves with the increase of the encoding rate. When the more the encoding rate increases, the more frequent values which are found around zero will be quantized to category 1 and thus transcoding will take the lead in terms of performance. Nevertheless, since this has been only a first proposal for implementing an efficient variable-length quaternary encoding for DNA data storage, the proposed algorithm can be further improved to deal with this issue and provide better results.

Even though the above experiment is testing the strength of the encoding scenarios in terms of compression quality, one must not neglect the main obstacle of DNA coding which is the error-prone DNA sequencing. It is clear, that even though a fixed length encoding might be less efficient in compression, it is more robust to sequencing noise. This is due to the fact that in the case of an error during sequencing, only a part of the decoding will be affected. To facilitate understanding, let's take the following example. Assuming the case of a deletion error the full structure of an oligo is affected by shifting all the nucleotides following the deleted-one, by one position. However, since the encoding is fixed-length, the decoding of one oligo does not depend on the decoded information of the previous oligos. Thus, even is one oligo is lost, the following oligo can be correctly decoded. on the other hand, in a variable-length encoding, a deletion error in some oligo will also affect the rest of the decoding and the structure of the image can be lost. To prove this claim we have tested the decoding of an image in the presence of one single deletion at a random position for the cases of a simple fixed-length encoding using Vector Quantization and the two variable length solutions of transcoding and modified JPEG for DNA. The selection of testing the error impact for the case of VQ relies on the fact that it provided the most interesting results in terms of coding potential. Furthermore to ensure that the addition of noise is not favorable for the case of VQ we have selected adding the deletion noise in the LL subband of DWT so to create the strongest possible distortion. The impact of such an error in the visual quality of the decoded image is presented in figures 5.7 and 5.6 for an encoding which produces an image of the same PSNR and rate respectively. As observed, the impact of a single deletion error in the case of a fixed-length encoding using VQ affects less the visual quality of the input image and causes the loss of a single vector. However, in the case of a variable length encoding, one deletion will completely change the decoding and reconstruction of the image causing a much bigger visual distortion. More precisely, in the case of transcoding, the error has a much worse impact on the decoding of the image. To give an explanation to this fact, let's consider a pool of oligos which have been selected as the most representative ones after computing the consensus sequences. Then let's assume that all the selected oligos are correct except for one, which has suffered a deletion error. In the case of the modified JPEG, one deletion will cost a shift in all the nucleotides that follow the deletion. Therefore, the total impact on the long reconstructed strand will be a shift of all nucleotides following the deleted nucleotide by one position to the left. Let us now think of the same type of error in the case of transcoding. In this case, the DNA strand has to be first decoded into a binary representation by decoding each 5 nts of the DNA sequence to one byte and then the produces binary stream is decoded using the JPEG standard. Consequently, in the case of transcoding, one single deletion will create a shift on the DNA strand after

the deleted nucleotide resulting to a wrong decoding of all the bytes that follow the deletion. This creates much more distortion that leads in losing big part of the input image.

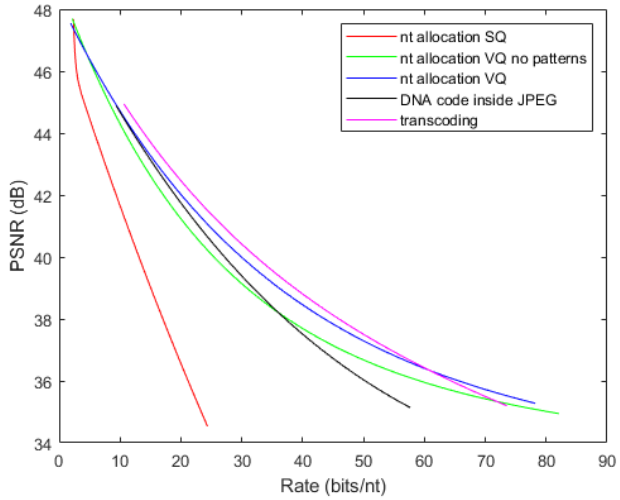


Figure 5.5: Comparison of the different proposed encoding solutions. The methods compared are the following: nucleotide allocation using SQ and PAIRCODE for the encoding (red), nucleotide allocation using VQ and PAIRCODE for the encoding without treating pattern repetition(blue) and treating pattern repetition(green), JPEG transcoding (magenta) and JPEG with the quaternary code included in the optimization (black).

It is obvious that, as predicted, the fixed length encoding solutions are much more robust to noise as in the case of variable-length one error can result in losing the structure of the image. It is therefore clear that in DNA coding the selection of the best encoding solution does not only rely upon the performance, but highly depends on the robustness to sequencing error. Hence, as observed, VQ is a fixed-length solution which is robust to noise and comparable in performance with the variable-length encoding methods and seems to be a promising candidate to be selected for the needs of DNA data storage! However, as explained in section 4.4.2, the proposed VQ solution requires knowledge of the codebook that has been used for the quantization. In this manuscript, there have been proposed several solutions to this issue, such as the use of the same codebook to encode multiple images and the creation of a database of codebooks to be used for the purpose of DNA coding. Nevertheless, since the results of VQ have been so encouraging, it is in our first priorities in future works to test the application of Lattice VQ ([49], [50], [52], [51], [56], [57]) for the encoding as it does not require knowledge of the codebook for the decoding.

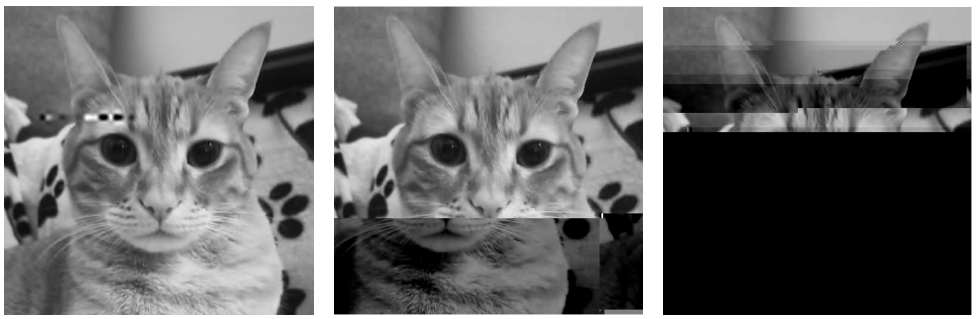
5.5 Conclusions

In this chapter, we extend our study to the implementation of a variable-length encoding method which is based on the JPEG standard of binary image coding. The proposed solution constitutes an extension of the existing

standard to the generation of a quaternary DNA representation which is optimized using entropy coding with respect to the biological constraints imposed by the sequencing process. To test the efficiency of our proposed solution we compared it to the fixed-length solutions presented in the previous chapter as well as to the method of transcoding a JPEG compressed binary stream which has been widely used in the state of the art. In addition to this, we tested the robustness of each encoding method to one single deletion error.

The comparison proved the performance of the fixed-length encoding with VQ to be comparable to the variable-length methods while being the most robust solution. Our proposed JPEG-based encoding method provided very interesting results but did not out-perform the method of JPEG transcoding. This is justified by the fact that in order to respect the sequencing constraints our proposed algorithm avoids an encoding which is using 1 nt for the encoding of a source symbol. Such a restriction does not apply to the case of JPEG transcoding which is using a binary optimization for the encoding. It is however important to denote that our JPEG-based solution remains more robust to sequencing noise than transcoding.

Summing up all the above points, one can conclude that fixed-length solutions are much more robust to sequencing noise than variable-length solutions for which the decoding of an element depends on the correct decoding of the elements that precede it and risks losing information of the image structure. Thus, the choice of the optimal encoding solution depends on the needs of the encoding as well as the robustness of the sequencing machine which is used for reading back the DNA sequences. Furthermore, since the results of a variable-length encoding are more interesting in terms of coding potential, we strongly believe that further improvement in the encoding of our proposed JPEG-based solution can provide much more promising results. Finally the addition of some error-correction techniques can allow the encoding to be more robust to sequencing noise for ensuring that the information about the image structure is maintained throughout the decoding process.



nt allocation using VQ
1 deletion in LL subband

JPEG "closed loop"
1 deletion

transcoding
1 deletion

Figure 5.6: The impact of one deletion error on a 512x512 pixel image of a cat for the different encoding solutions.The original images have been compressed to obtain the same Rate of 25.5 bits/nt.

nt allocation using VQ
1 deletion in LL subband

JPEG "closed loop"
1 deletion

transcoding
1 deletion

Figure 5.7: The impact of one deletion error on a 512x512 pixel image of a cat for the different encoding solutions. The original images have been compressed to obtain the same PSNR value of 38.5 dB.

Chapter 6

Formatting the encoded data for oligo synthesis

6.1 Introduction

In the previous chapters we have explained the different encoding options for storing an efficiently compressed image into DNA. Namely, we have presented both a fixed length and a variable-length encoding solution. The proposed fixed-length encoding consists of a compression part which uses a 9/7 DWT to decompose the input image into different subbands, a subband quantization step which can be performed either by a Uniform Scalar quantizer or a VQ, an allocation to determine the optimal quantization parameters to efficiently compress the input image and reduce the synthesis cost by building the appropriate quaternary code and finally a mapping function which is adaptable to the needs of the user and can be either avoiding patterns or resistant to sequencing noise. The proposed variable length encoding solution consists of two different encoding scenarios. The first one is using the classical JPEG standard for the efficient compression of images in a binary representation which is later encoded in a quaternary representation using the PAIRCODE codec proposed in section 3.2.2. However, since this encoding is open-loop and provides an encoding which is optimized according to the binary representation of the input we then proposed a second encoding scenario. In this second variable-length solution we proposed modifying the main structure of the classical JPEG standard to provide a constrained quaternary representation of A, T, C and G. Consequently, the above encoding methods provide in the output a long sequence of nucleotides. However, as discussed in section 2.1.2, the synthesis of DNA inserts one final restriction to be respected in the DNA coding workflow and requires cutting the encoded strands into shorter chunks of information and formatting them by adding particular headers related to the position and the encoding parameters used for each data-chunk.

The proposed fixed-length encoder provides in the output a long sequence of nucleotides for each of the DWT subbands which is designed according to the needs of DNA data storage. Similarly in the case of using the proposed variable length encoder (or else JPEG DNA), the output will be a sequence of encoded data as well as a set of frequencies for the different categories of the DC indices as well as the different run/category pairs of the AC indices of DCT. Thus, for any of the encoding scenarios, the process of formatting should be designed according to the workflow used for the encoding and should contain all the necessary headers that should be stored along with

the data for correctly decoding and reconstructing the image that has been stored into DNA. Consequently, the headers contain important information which should be correctly decoded and retrieved without errors so to ensure reliability. To protect those important headers we propose protecting them using the method of barcoding which we will further explain and describe in the next sections along with the proposed formatting for each of the different encoding cases which have been used in our experiments.

6.2 The proposed formatting for the fixed length encoding

As explained in chapter 2 the efficiency of DNA data storage highly depends on the quality and reliability of the decoding and thus on the quality of PCR amplification and sequencing. We recall that during decoding the synthesized oligos are amplified into multiple copies with the process of PCR introducing the necessary redundancy to facilitate the decoding process. PCR amplification requires specific primer sequences of a length of around 20 nts to allow correct binding of the polymerase enzyme so to create enough copies of each synthesized oligo.

The amplified oligos are then passed through the sequencers to read the content of the DNA strands. The sequencing can be performed by either the Illumina or the MINION Nanopore machine. Both of the sequencers require special primers to start the reading process. Consequently, each oligo needs to contain special primer fields which are related to the bioloical processes of PCR amplification and sequencing.

During sequencing, the DNA strands can be read in both senses starting from either the 3′ or the 5′ end of the oligos. This means that after sequencing some of the reads will need to be reversed and complemented to be used for the decoding. To this end one special nucleotide is placed in each oligo-end to indicate the sense of the reading.

Another important field to be added to each oligo, is a header of ID which will contain the identification number of each stored image. This ID is required in the case where multiple data is stored at the same DNA pool. This particular field can be very useful for random-access in the DNA pool of oligos. More precisely, many works such as the ones presented in [28] and [37] are exploiting the possibility of retrieving and reading particular sections of information from a DNA memory by using the field of ID as a key.

Finally, a parity field (P) can verify the correctness of an oligo. For checking the parity one can use a field of 4 nts for counting the parity of each of the 4 different nucleotides of DNA independently. More precisely, we define that the first nucleotide of the parity field is used for checking the parity of A's, the second for the parity of T's, the third one for the parity of C's and finally the last nucleotide for the parity of G's in the encoded payload. To make sure that no homopolymers are created, we define that an even parity is represented by the symbols A or C and an odd parity by the symbols T or G. This way if all 4 nucleotides have the same parity in the payload, one can alternate the two symbol-options and avoid repeating the same symbol more than 2 times.

All the above fields are required for an efficient encoding and should be included to all the produced oligos. The rest of the oligo length is used for storing the payload or in other words the encoded image as well as all the

necessary headers containing information about the specific encoding of the image to be stored. Consequently, the content of the payload depends on the encoding methods used for producing the DNA sequence containing the input image. Thus, it is obvious that the type and the number of headers to be stored in the payload along with the encoded data varies according to the different encoding techniques. More precisely, while the general formatting of the oligos is common for all the proposed compression workflows, there are slight changes in the content of headers according to the type of quantization and mapping used for the encoding.

Our proposed general compression workflow decomposes the input image using an #ℓ-level DWT decomposition into SB different subbands. Each subband is then independently quantized and encoded into a quaternary stream. The quantization parameters are optimally selected for each subband thanks to the nucleotide allocation algorithm described in section 4.3. Consequently, since the different DWT subbands are independently treated, for the formatting we propose creating SB different Subband Information Oligos (SIO) which will contain information about their specific encoding as for example the quantization parameters selected for each subband, the length of the encoding codewords etc. Furthermore, it is necessary to introduce a Global Information Oligo (GIO) which will contain all the global information for the encoding such as the image size and the number of DWT levels that were used in the encoding. Since the SIO and GIO only store headers for the decoding while the length of the oligo is relatively big, in those two types of oligos there is an empty field left which can be filled with any needed extra information. An example would be to replicate many times the same headers so to introduce some extra redundancy which can improve the decoding. This extra redundancy does not affect a lot the total encoding cost as this type of padding fields will occur only in the GIO and SIO oligos. Finally, the data will be cut and formatted into Data Oligos (DO) including an additional offset header which encodes the position of the data in the input image. The length of each field of the oligo formatting is predefined according to the encoding needs. Let us call g the length of a data field in a DO. Then for each encoded image there will be produced one GIO and SB different SIO. The number of DO d is then given by the following relation:

$$d = \sum_{sb=1}^{SB} \frac{\frac{MN}{4^{(\#\ell)}} l_{sb}}{g} \tag{6.1}$$

where M and N represent the number of rows and columns of the input image, l_{sb} denotes the length of the quaternary words in the code $\mathcal{C}^*_{sb}$ of each subband sb. The proposed formatting is illustrated in figure 6.1.

6.2.1 Formatting when using DWT and Uniform Scalar Quantizer

As explained in the previous paragraph in any case of encoding all the produced oligos will follow a same general format containing the payload field surrounded by the headers of Primers, Sense, ID and parity. The payload can be filled either with global information on the encoding and will follow the format of a GIO, with a SIO containing specific information on the encoding of some subband sb with $1 \leq sb \leq SB$ or with a DO containing the data of some subband sb.

Figure 6.1: General formatting of an oligo. Any synthesized oligo should contain a primer sequence in both ends (Primer 1, Primer2) to allow both PCR amplification of the synthesized oligos and to enable reading by the sequencer. Oligos should also contain a sense nucleotide (S) in each end and after the primer sequences to denote the way that the oligo has been sequenced. The field of ID contains the identifier of the stored information and is unique for each different type of information stored in the form of oligos. The data is encoded into a constrained quaternary sequence of nucleotides and stored in the Payload field and a parity field (P) can control the parity of the payload to verify its reliability.

In the case of a Uniform Scalar Quantizer the formatting of a GIO contains one header in the beginning to denote the type of payload (G), a header containing the number of levels used for the DWT (#lvls of DWT), then the number of image rows (#image rows) and finally the number of image columns (#image columns). The rest of the oligo (empty field) can be filed with any needed redundant information to facilitate error correction or with random padding (see figure 6.2).

The formatting of an SIO should contain all quantization parameters which are necessary for decoding and recovering the stored quantized image. More precisely, the SIO will contain one header in the beginning to denote the payload type (S), a header field to denote the type of subband which can be either HH, HL, LH or LL, a header field to mark the level of the subband $\#\ell$ (Lvl of DWT), a field containing the quantization step-size q selected for this particular subband, a field containing the size L of the quaternary code $\mathcal{C}$ for the given subband, a field to denote the sign of the minimum value existing in the quantized subband and a last field containing the minimum value (min val) of the subband (see figure 6.2). As the minimum value of the subband can be a float and not necessarily an integer, this last field can be separated if needed in 3 sub-fields containing the integer part of the minimum subband value, the decimal part rounded to some digit and expressed as integer as well as the number of decimal digits used for the encoding expressed as integer. As in the GIO the remaining length can be used for any needed information or it can be filled with random padding.

Finally, the data oligo payload contains a long field of encoded data which is preceded by the necessary headers starting with a header to denote the type of payload (D) which is followed by a field to denote the type of subband from which the corresponding data come from as well as the level of the subband and the offset of the data in the subband. This proposed formatting is depicted in figure 6.2.

6.2.2 Formatting when using DWT and Vector Quantization

Similarly to the case of using a Uniform Scalar Quantization, in the case of VQ the formatting will follow the same general format with a payload enclosed between the global header fields of Primers, Sense, Parity and ID of the encoded image. The payload will follow one of the three possible formats

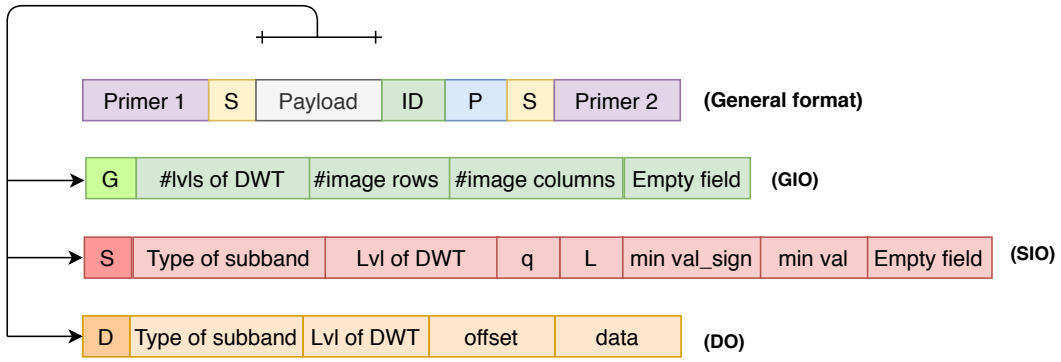


Figure 6.2: Proposed oligo formatting for a compression workflow that uses a DWT and a Uniform Scalar Quantizer

of GIO, SO or DO so to store all the needed information about the encoding parameters in order to allow correct decoding of the stored data.

As explained in section 4.4.2 VQ is more efficient in terms of the achieved compression rate but the main drawback is the fact that the codebook of vectors needs to be known to the decoder to correctly retrieve the stored image. To this end, in this section, we propose two different alternatives for formatting the encoded data when VQ is used for the quantization. In our experiments we have shown that in the case where the codebook has been built using a good training set of images which share similar characteristics and depict similar content, one codebook can efficiently encode many different images of the same kind. Consequently, the more images stored using the same codebook the more the cost for storing the codebook is compensated. Codebooks can potentially be stored in some database which will contain many codebooks of different content, for example human faces, cats, landscapes, buildings etc. and be uniquely used for the purposes of DNA data storage. In other words, in order to ensure the knowledge of the encoding codebook to the decoder, any image which is stored into DNA using VQ will need to be encoded according to some codebook that already exist in the codebook database. The codebook database can be either stored online to allow decoding of the corresponding stored images in the short term, or in a most realistic scenario of long term preservation it could also be stored into DNA. A DNA encoded database can be a pool of codebook oligos which will be identified thanks to a particular header of codebook ID and can be retrieved among different codebooks anytime using the methods proposed by Appuswamy et. al in [37]. In this last work, the authors proposed an in-vitro query processing allowing to access data directly in the DNA storage by realizing SQL operations using molecular biology techniques. More precisely, with the help of PCR, DNA nucleases [58] and overlap-directed DNA assembly methods, we can detect single DNA sequence in a background of 100 nanograms of irrelevant DNA sequences.

A proposed formatting when encoding an image using VQ is depicted in figure 6.3. It is obvious that in any encoding scenario the general format of the oligos containing the primers, sense, parity check and image ID remains the same. Furthermore, similarly to the Scalar Quantization case, when VQ is used for the encoding there can exist 3 different types of payload containing global information on the encoding (GIO), information regarding the specific

encoding of each subband (SIO) and the encoded data (DO). In a VQ formatting scenario the SIO should contain the quantization parameters regarding the number of vectors K used for the encoding as well as the vector length n. In addition to this, the GIO should contain an extra header field to denote the codebook ID (cb ID) to allow identification of the encoding codebook by the decoder. In the lowest part of figure 6.3 we provide a proposed formatting for the case in which the codebook should be stored into DNA.

A complete codebook should contain the encoding vectors for all the different subbands and for any possible value of K and n. Therefore each codebook oligo must contain the following header fields:

- Codebook ID (cb ID): It corresponds to the unique identifier of a codebook.
- Vector type (vec type): To denote the type of vectors used in VQ. It corresponds to the "shape" of the vectors and can be either row, column or block vectors.
- Subband (sb): It corresponds to the subband for which the encoding codebook is required. This field contains the level of DWT and the type of subband (HH, HL, LH or LL).
- Offset: It corresponds to the offset of the vector data which are included in the oligo.

The vector values follow the above header fields. Each vector value is expressed using the following information:

- sign: Denoting if the value is positive or negative
- int: The integer part of the vector value encoded using our proposed codebook construction
- dec: The decimal part of the vector value expressed as integer. It can contain a predefined number of digits.

6.2.3 Formatting when using VQ and controlled mapping

In sections 3.5.1 and 3.5.2 we have proposed a sequencing noise resistant mapping for reducing the visual impact of substitution errors on an image that is encoded in DNA using VQ. This controlled mapping creates an optimal mapping by appropriately sorting the codewords in the quaternary code according to the probabilities of each quantization vector in the codebook. It is therefore obvious that since the optimal mapping depends on the input characteristics it will be different for each input image and thus it is necessary to also store the sorting of the codewords produced by the mapping.

The proposed formatting for including a controlled mapping in the encoding method is the same as the one used for VQ with the only difference that in the case that controlled mapping is used we introduce one extra payload type which is dedicated to the storage of information about the mapping. The most efficient way to store the required information is by providing a set of sorted vector indices which correspond to each codeword in the

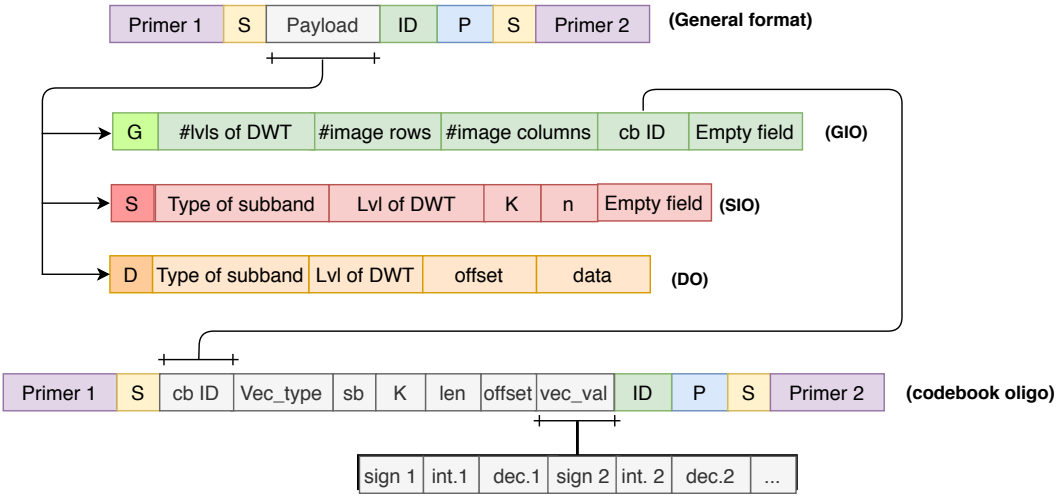


Figure 6.3: Proposed oligo formatting for a compression workflow that uses a DWT and a Vector Quantizer

quaternary code as selected by the mapping function. The proposed formatting of a VQ which is using controlled mapping is depicted in figure 6.4. The mapping payload (MO) which is depicted as a last possible case of payload type is composed by a nucleotide to denote the payload type 'M', a field to denote the subband type (HL,HH,LH or LL), the level of DWT, an offset to denote the order of the information to follow and finally the set of the mapping indices.

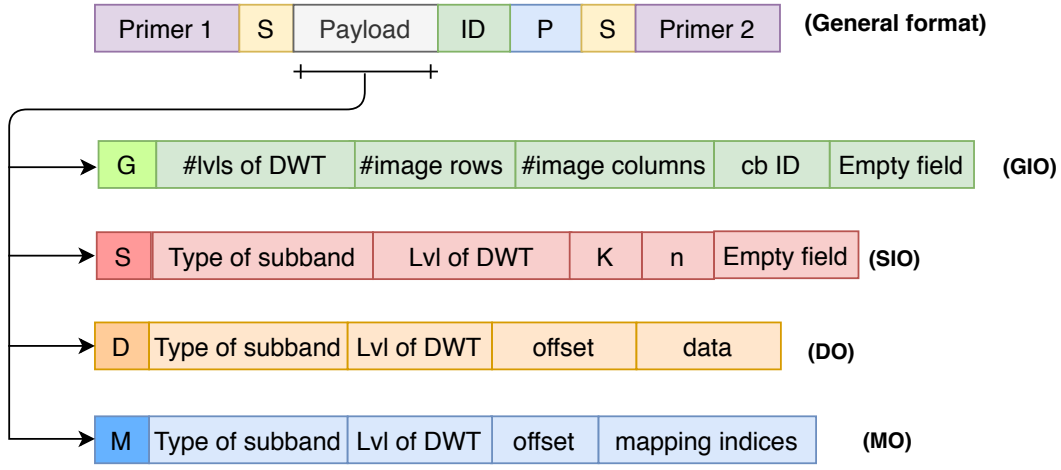


Figure 6.4: Proposed oligo formatting for a compression workflow that uses a Vector Quantizer with controlled mapping

6.3 The proposed formatting for the variable-length encoding

When it comes to variable length coding the general format of the oligo remains the same as the one proposed in section for the case of fixed length encoding. Thus, each oligo needs to contain, the sequencing primers, the

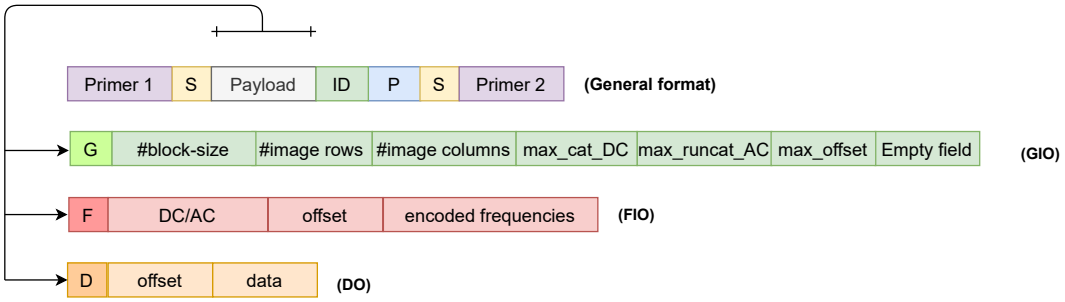


Figure 6.5: Proposed oligo formatting for a compression workflow that uses closed-loop JPEG for DNA coding

sense nucleotides, the identifier field and the parity check field. As in the case of fixed length encoding, in this encoding scenario, there will also be several different types of payload. A payload containing general information about the encoding which will form a General Information Oligo (GIO), a payload containing the frequencies of 3-ary Huffman used for the encoding of AC and DC indices of the encoding algorithm which has been analytically described in 5.3 that will form a Frequency Information Oligo (FIO) and a payload containing the encoded data that will form a Data Oligo (DO). The content of the different payload types is depicted in figure 6.5 and is analytically described in the following paragraph.

The GIO payload contains the following fields:

- A nucleotide that denotes the payload type (G) for distinguishing the global information oligos from the other types.
- A field to determine the block size of the DCT of JPEG DNA (# block-size).
- A field for the number of image rows (# image rows).
- A field for the number of image columns (# image columns).
- a field for the maximum category of DC indices available for this encoding (max_cat_DC).
- A field for the maximum index of run/category for the AC indices of the encoding (max_runcat_AC).
- A field for the maximum offset of data oligos to determine the end of the encoded data (max_offset). This field is necessary as all oligos have the same oligo size and therefore the oligo containing the last part of encoded data might be padded with random nucleotides to reach the required oligo length.
- A field that will be randomly padded with random sequence of nucleotides to reach the required oligo length (empty_field).

The FIO payload contains the header fields that is presented below:

- A nucleotide that denotes the payload type (F) for distinguishing the Frequency Information Oligos from the other types.

- A field to distinguish whether it is a DC frequency oligo which contains the frequency of the category indices or an AC frequency oligo which contains the frequency of the run/category indices of the AC coefficients (DC/AC).
- A field to denote the offset of the encoded frequencies or in other words the order of the encoded frequencies (offset).
- The encoded frequencies (encoded frequencies).

Finally the DO payload contains the following information:

- A nucleotide that denotes the payload type (D) for distinguishing the data oligos from the other types.
- A field to denote the order of the encoded data or else the offset of the data (offset).
- The encoded data (data).

6.4 Barcodes

One of the main challenges when storing data into DNA is to ensure decodability of the data. Since the high throughput DNA sequencing is a procedure which introduces much noise in the oligos the full retrieval of the stored information can be at stake. One of the main problems when encoding images into DNA is the fact that if an error occurs in some important headers, the decoding becomes challenging if not impossible. This problem is also faced in biological applications where biologists are using specific barcodes (short sections of DNA) to tag DNA gene fragments in order to identify the species to which those fragments belong to. Similarly to the formatting headers used in DNA data storage, those barcodes hold important information and should be correctly recovered during sequencing. To robustify this information several works have proposed different methods to create error correcting DNA barcodes which can be more robust to sequencing error. Inspired by this idea we propose using such a method for protecting the most important header fields of our proposed formatting to increase the reliability of the decoding. In the next sections we will present the existing methods for creating robust barcodes, we will explain the main assets of each method and will present the method that we have proposed for our experiments.

6.4.1 Background and existing methods

As discussed in section 2.1.3, Next Generation Sequencing can create substitutions, insertions or deletions of nucleotides. Therefore, the sequencing process can be associated to a noisy channel for data transmission. Many works on classical binary source-coding theory propose the use of binary error correcting codes to robustify the transmitted information. Such an example is the use of Hamming binary codes as presented in [59]. Inspired by this idea, later works in [60] and [61] have proposed using binary linear error correcting codes to create barcode sets for DNA coding applications but since linear codes are only capable of correcting substitution errors, Buchmann et al.

have proposed in [2] the use of the so called "Sequence-Levenshtein codes" to also allow correction of insertion and deletion errors. Sequence Levenshtein code works very similarly to the classical Levenshtein codes [62] but is more adapted to the DNA context. In order to better understand the barcode-set creation using sequence Levenshtein distance let us first introduce the notion of an error-correcting DNA code.

Let's assume a code $\mathcal{C}$ containing all possible codewords that respect the biological restrictions discussed in section 2.2. We define as error correcting DNA barcode set $\mathcal{B}$ as a subset of $\mathcal{C}$ containing codewords $w = \{w_1, w_2, \dots\}$ such that:

$$d(w_i, w_j) \geq 2\mu + 1, \forall i \neq j \tag{6.2}$$

where $d(.)$ is the distance metric which can be either Hamming distance denoted as d_H, Levenshtein distance denoted as d_L or Sequence-Levenshtein distance denoted as d_{SL} and μ is the number of errors that can be corrected. In the case of a Hamming distance the produced barcode set can correct only substitution errors while in the rest two cases it also includes insertions and deletions.

As proven in [2] by Buchmann et al., the best option for constructing a robust to sequencing error DNA barcode set is using the sequence Levenshtein distance metric. The Sequence-Levenshtein distance between two arbitrary words A and B is the minimum number of substitutions or/and deletions or/and insertions which results in a word $\hat{A}$, finalized by one of the following operations exactly once:

- Truncating $\hat{A}$ to match the length of B
- Elongating $\hat{A}$ to match the length and bases of B

The latter two operations do not increase the Sequence-Levenshtein distance between A and B. Thus, $d_{SL}(A, B) = 0$ if A is a prefix of B or vice versa.

The reason for which sequence-Levenshtein distance is more efficient in error correction than the classical Levenshtein codes is because the latter fails when used in real DNA applications where there is a DNA sequence that follows the barcode prefix. To better understand this let us study a simple example. Let's assume a barcode set $\mathcal{B}$ containing the 3 following barcodes:

$$\mathcal{B} = \{\text{"CAGG", "CGTC", "CTAT"}\}$$

All three codewords in the set have a Levenshtein distance of 3 compared to any other word in the set. Now let's suppose that the original DNA barcode is "CAGG" and the occurence of a deletion error during sequencing in the second position of the barcode. Then, "CAGG" is transformed to "CGG". The deletion error will then create a word which does not exist in the barcode set $\mathcal{B}$ revealing that an error has occurred. By checking the Levenshtein distances between the produced codeword "CGG" and all the barcodes in the set $\mathcal{B}$ we easily find that the minimum distance occurs when comparing it to the codeword "CAGG" and therefore the erroneous word will be corrected to the original one. But let's see what happens in a more realistic scenario where the barcode is followed by some long DNA strand.

Let's assume the same barcode set $\mathcal{B}$ as before and let's suppose that the original barcode "CAGG" was inserted in the beginning of some DNA sequence producing the DNA strand "CAGG|CA..." as depicted in figure 6.6.

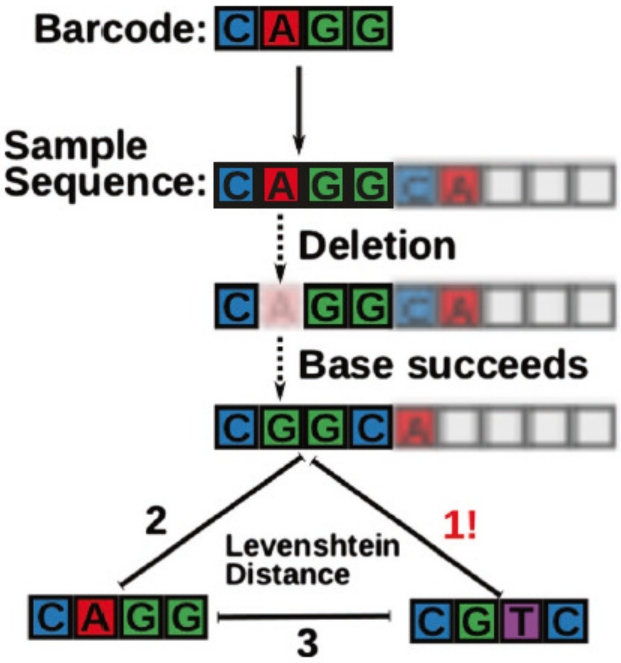


Figure 6.6: Figure taken from [2]. An example where Levenshtein distance fails to capture a deletion error.

If the base 'A' at the second position becomes deleted, the base 'C' (previously on position 5) would now succeed the base at position 4 so that the sequenced DNA received would read: "CGGC | A...". Using the simple Levenshtein distance metric will fail to detect the original barcode in this case as the produced erroneous word "CGGC" will be closer in terms of Levenshtein distance to the word "CGTC" rather than the correct one.

However, according to the Sequence-Levenshtein distance $d_{SL}(CAGG, CGTC) = 2$. More precisely, the codeword "CAGG" can be transformed to the word "CGTC" using the following steps:

1. Deletion of the second base A: CAGG becomes CGG
2. Substitution of the last base G with a T: CGG becomes CGT
3. Elongation of the word by adding a C in the end of the codeword: CGT becomes CGTC

Since the last step of elongation does not increase the distance metric, d_{SL}("CAGG","CGTC")= 2. Thus even though those two words could co-exist in the same barcode set according to the classical Levenshtein distance (as in that case d_L("CAGG","CGTC")= 3), according to this new metric of Sequence-Levenshtein this is no longer possible. Hence using the Sequence-Levenshtein metric one can build more robust barcodes.

By testing different error scenarios the authors in [2] concluded that sequence Levenshtein codes are more efficient that classical Levenshtein codes and thus this is the metric that we will also adopt for our experiments

6.4.2 The algorithm for building error correcting barcode sets: discussion

The algorithm for building an error-correcting barcode set is composed by 3 main parts:

1. Selection of the length of barcodes l according to the number of errors μ to be corrected. More precisely:

$$l \geq 2\mu + 1 \tag{6.3}$$

2. Construction of a code $\mathcal{C}$ of codewords of length l which respects the biological constraints for DNA coding (see section 2.2).

3. Build a set $\mathcal{B} \subset \mathcal{C}$ of barcodes such that any words $w_i, w_j \in \mathcal{B}$ satisfy the following equation:

$$d_{SL}(w_i, w_j) \geq 2\mu + 1, \quad \forall i \neq j \tag{6.4}$$

The main challenge of an error-correcting barcode building algorithm lies in the computational complexity of step 3. More specifically since the computation of an optimal barcode set with maximal cardinality can only be found using an exhaustive search the distance between any two codewords should be calculated at least once making $\frac{L^2}{2} - L$ calculations necessary, with L denoting the size of the code containing all viable codewords of length l. To lower the computational complexity, in [2], the authors propose the use of a greedy closure evolutionary algorithm as it has been first proposed by [63].

This algorithm allows finding a local optimal solution to the given problem. The authors initialized the code set $\mathcal{C}$ with a small number (2-4) of random barcodes that fulfill the distance requirement (the so-called seed). Then, by checking all eligible barcodes in lexicographical order they added the tested barcode to the code set if its distance was at least $2\mu + 1$ to every other barcode that was already in the code set. Using an evolutionary approach (in the computational sense), they tried a large number of different seeds or altered very successful seeds to find the seed giving the best, i.e. largest code set. Among other heuristic algorithms for the generation of classic Levenshtein codes, this particular method has shown the best results [64]. Consequently, the authors re-run the same algorithm with multiple seeds to find the seed that generated the largest barcode set.

In our works for this thesis, we extended this algorithm for the simultaneous creation of more than one barcode sets and the selection of the best case for a given seed. Thus, there is no need for running multiple times the barcode construction algorithm for different seeds but instead we create many barcode sets at once and select the set with the maximum cardinality. Furthermore, the filtering of the code that contains all 4^l possible quaternary words of length l for excluding the non-viable DNA codewords can be complex in computation time. Since the algorithm for constructing barcodes is already a complex process that requires long execution time we tried to lower the complexity of this filtering step by replacing it with the algorithm PAIRCODE that has been proposed in 3.2.2.

6.4.3 Our proposed barcoding algorithm

In this section we describe our proposed version of the algorithm for constructing error correcting DNA barcodes. This algorithm uses as an input some code containing only viable codewords which respect the biological restrictions of homopolymers and GC content and provides as an output multiple sets of barcodes. Each barcode set has a different cardinality and therefore the largest set can be selected as the best option. This selection is due to the fact that the more barcodes one set contains, the more different oligos can be protected from noise. The proposed construction of all the possible barcode sets using a constrained codebook is achieved by the following procedure:

- We first define a set $\mathcal{C}$ containing all possible quaternary words c_k which respect the sequencing constraints, with $k = \{1, 2, \ldots, K\}$. We also define as $\mathcal{B}$, a set containing S different barcode sets $\mathcal{B}_s$ with $s = 1, 2, \ldots, S$, $S \leq K$.
- Initialization: Add one codeword $c_k \in \mathcal{C}$ of the codebook $\mathcal{C}$ in a first codeword set $\mathcal{B}_1$, set $S = 1$. This first word is considered as a seed of the algorithm and for simplicity it can be selected to be the first codewords $c_1 \in \mathcal{C}$.
- For each next codeword c_k:

  - Check the distance $d_{SL}(c_k, c_j)$ between c_k and all codewords c_j in each of the existing barcode sets $\mathcal{B}_s$ with $s = 1, 2, \ldots, S$.

  - If $d_{SL}(c_k, c_j) \geq 2\mu + 1, \forall c_j \in B_s$, then add c_j to B_s. - If the previous condition wasn't met for any existing barcode set, create a new barcode set $\mathcal{B}_{S+1}$ containing this codeword c_k.

  - Once all the possible barcode sets have been created and all codewords in the input constrained code $\mathcal{C}$ have been used, we select the set $\mathcal{B}_s$ with the biggest cardinality.

The above algorithm is more complex in terms of computational cost than the one proposed in [2] but has the asset of creating more than one barcode sets in one single run using as a seed only the selection of the first codeword to be inserted in the first barcode set $\mathcal{B}_1$. By changing the first word which is inserted in the first barcode set we change the seed of the algorithm and therefore the different barcode sets that will be created will differ according to this seed. As described in the previous section the barcoding algorithm proposed in [2], the authors are using a step of filtering out the non-viable codewords from a codebook which contains all possible 4^l quaternary codewords of length l. This filtering introduces some complexity to the barcoding algorithm as it increases exponentially with the increase of the codeword length l. Therefore, in this work we also propose the direct use of the code constructed using PAIRCODE which has been described in section 3.2.2 to reduce the complexity of the code construction step. As explained in section 3.2.3, when using PAIRCODE the constrained codebook is built using particular pair symbols to construct codewords which respect the biological restrictions. Hence, some viable codewords are omitted. However, this code construction is very simple in complexity and provides a set of suitable codewords for DNA coding. It is thus expected that since some viable codewords are missing, the number of barcodes found in the code will be lower than in

# errors	codeword length	Buschmann et. al. [2]	Exhaustive Code	PAIRCODE
1	4	4	4	4
1	5	12	10	10
1	6	28	25	19
1	7	77	76	59
1	8	186	202	132
1	9	615	615	426
2	6	3	2	2
2	7	5	5	4
2	8	8	9	7
2	9	17	18	14

Table 6.1: Comparison of the number of barcodes produced by: The barcoding algorithm proposed in [2], our proposed barcoding algorithm applied to an exhaustive code which contains all possible viable DNA codewords (similarly to [2]), and our proposed barcoding algorithm applied to a code constructed by PAIRCODE (see section 3.2.2).

the case where the code is constructed using the filtering of non-viable words proposed by [2].

To test the efficiency of our proposed barcode construction algorithm, in our studies we performed the following experiment. We tested the performance of the barcoding algorithm for the following scenarios:

- Using the codebook filtering which is checking all 4^l possible codewords of length l and filtering out all the non-viable codewords that don't respect the encoding constraints. This exhaustive code as well as its assets over the proposed PAIRCODE algorithm have been discussed in section 3.2.3.
- Using directly the code produced by our proposed PAIRCODE algorithm.

The result is depicted in table 6.1. From these results it is clear that our proposed barcode construction algorithm in the case of using the exhaustive code, can produce the same amount of barcodes with only one single run. In the case where the proposed barcode construction is applied to a code created using PAIRCODE, the number of barcodes produced is lower for longer codewords when correcting one error but performs well enough for the cases of shorter codewords or in the case of longer barcodes which can correct two errors.

6.5 Conclusions

In this chapter, we have proposed multiple formatting scenarios for the construction of DNA oligos that contain all the necessary information for the correct decoding of the stored content. The more complicated the encoding workflow, the more headers might be needed for the decoding, Since those headers contain fundamental information for the reconstruction of the encoded image, such as details regarding the image structure, it is extremely important to protect these fields by using some robust encoding. To this end,

we proposed an algorithm for the creation of a set of DNA codewords, to be used as barcodes for the correct identification of some important oligo headers. Those barcodes are robust to sequencing noise as they can be easily corrected in case of an error of insertion, deletion or substitution which can be caused by the sequencing process.

Chapter 7

Wet lab experiment

7.1 Description of the experiment

In order to verify the efficiency and robustness of the proposed encoding methods we carried out a real wet lab experiment by synthesizing, storing two small images in the form of DNA and retrieving the stored data with the method of DNA sequencing. As this experiment has been realized in 2018, it is important to denote that the encoding workflow which has been used for this implementation corresponds to our first studies which have been further improved over the last two years. However, due to multiple limitations we haven't been able to test our latest encoding models using a new wet-lab experiment before the writing of this thesis. In the next paragraphs we will describe the encoding, the methods used for the biological processes of synthesis, storage and sequencing, as well as the decoding workflow adopted for this experiment.

7.2 Encoding

7.2.1 The general workflow

In our experiment, we used two small gray-scale images which we efficiently compressed encoded and stored into DNA using the following workflow. To begin with, the input image of size MxN has been decomposed using a 3-level DWT into 10 different sub-bands. Each subband has been then independently quantized using a Uniform Scalar Quantizer. The quantization step-size has been optimized thanks to a nucleotide allocation method, as described in section 4.4.1, to provide the maximum quality (computed using the PSNR) or equivalently the minimum distortion (computed using MSE) for a given target compression rate R_{target}. More precisely, the nucleotide allocation is using the method of spline approximation, which has been extensively explained in section 4.4.1 to provide an optimal value for q_{sb}, with $1 \leq sb \leq 10$ for each of the 10 subbands produced by the DWT. The values of q_{sb} are used for quantizing each subband of level $\#\ell$ with $1 \leq \#\ell \leq 3$, producing a matrix Q_{sb} of quantized subband coefficients of size $\frac{M}{2^{(\#\ell)}} x \frac{N}{2^{(\#\ell)}}$, which are then restructured in a one dimensional vector $v_{Q_{sb}}$ by reading Q_{sb} using a raster scan. Each of the elements in $v_{Q_{sb}}$ is then encoded into a quaternary representation using the encoding algorithm proposed in section 3.2.2, resulting in a long sequence of A, T, C and G. At the next step of the encoding the long encoded subband strands need to be formatted into short oligos to ensure that the DNA synthesis will provide very low error-rates. In this

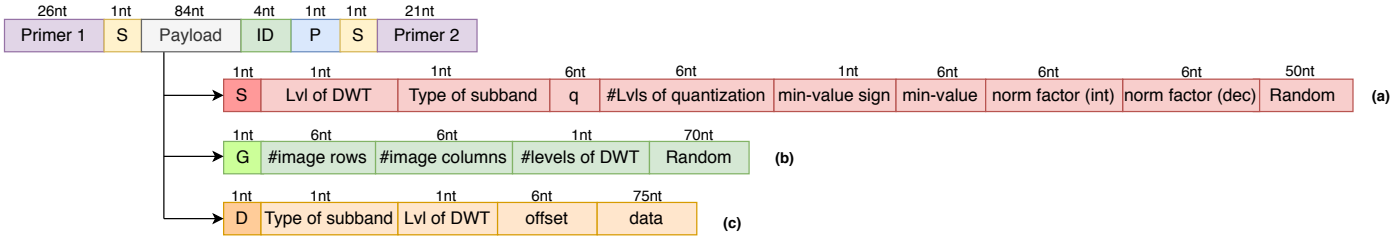


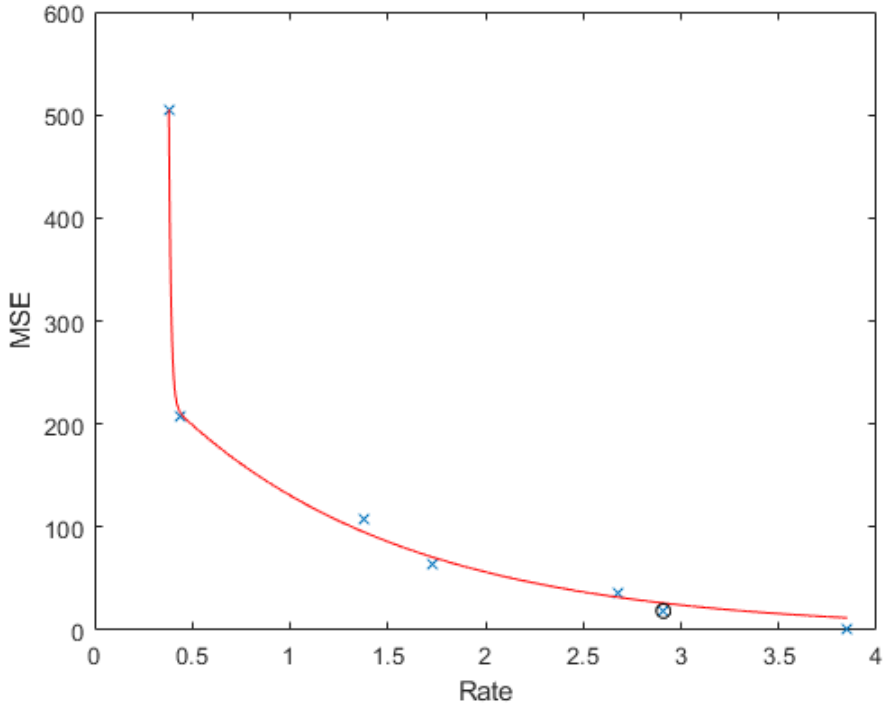
Figure 7.1: The formatting used in the wet-lab experiment.

experiment, the formatted oligos were 138 nts long. The reason for using this length is linked to the fact that the quality of DNA synthesis was not guaranteed for oligos of a length higher than 150-200 nts at the time when the experiment has been carried out. Today, companies which expertise in DNA synthesis can manage to provide good quality oligos (with very low error probability) of up to 300 nts. In this experiment, we adopted the formatting strategy shown in figure 7.1.

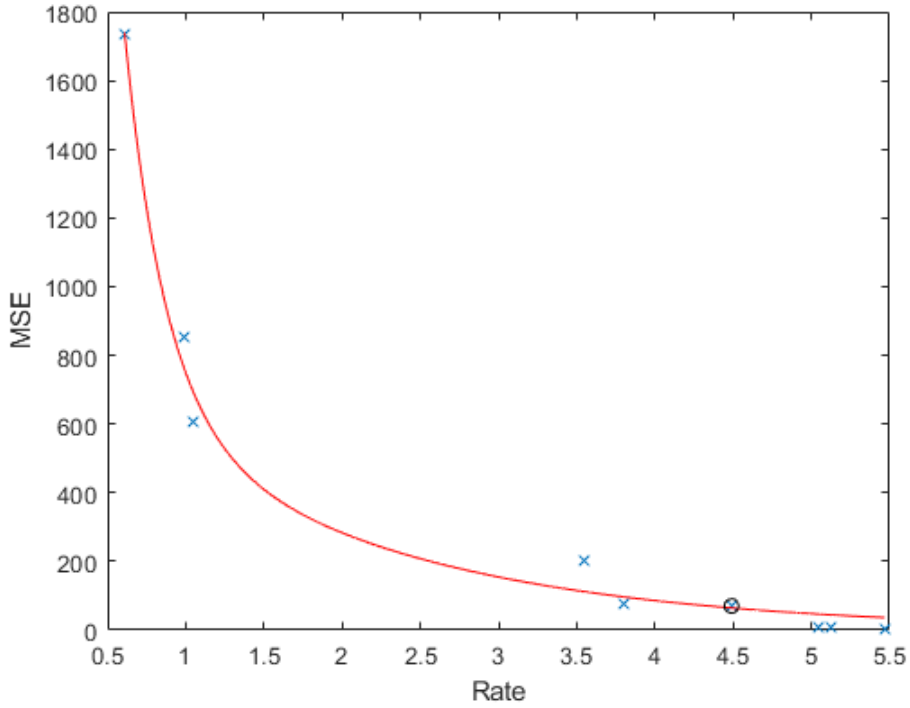
We start the formatting by creating the payload fields of the oligos. We first construct a global information oligo (GIO) payload which we denote using 1nt of 'G' attached to the left end. This oligo is holding the global information of the encoding such as the dimensions of the image M and N as also the number of levels used in the DWT. The remaining nucleotides can be either randomly padded respecting the biological restrictions or can contain some extra copies of the above fields to introduce redundancy which is helpful for robustifying important headers. We then proceed to create 10 subband information oligo (SIO) payload fields which we denote by attaching the nucleotide 'T' at the left end and contains information on the specific encoding of each subband including the optimal step-size q_{sb}, the minimum and maximum value of the quantized subband coefficients, the sign of the minimum value, the number of quantization levels. Similarly to the GIO, the unused length can be either randomly padded or filled with redundant information. Then we cut the long encoded subband sequences into smaller chunks of length 75 nts to be formatted into data oligo (DO) payloads. The DO are marked by adding the nucleotide 'A' in the left end which is followed by the necessary headers to denote the subband type and level in which the corresponding chunk originates as well as the offset denoting its position in the initial strand. The remaining part of the DO is filled with the data chunk. As the last data chunk of each encoded subband strand might be shorter than the rest, we fill in the remaining payload length of the oligo with random padding. After creating the different payload types we finalise the formatting by adding the fields of primers, sense (S), identifier (ID) and Parity (P) as shown in figure 7.1.

7.2.2 Details on the encoding used for the experiment

In our experiment we selected to store two different images into DNA. A 128x128 pixel image of Lena and a 120x120 pixel image of the cover of the album Mezzanine from the Massive Attack music band. The original images selected for this experiment are shown in figures 7.3.a and 7.3.b while figures 7.3.c and 7.3.d depict the compressed versions which have been stored into DNA.



(a) nucleotide allocation R-D curve for Lena. The rate is expressed in nt/coeff.



(b) nucleotide allocation R-D curve for Mezzanine. The rate is expressed in nt/coeff.

Figure 7.2: Nucleotide allocation curves used for the wet lab experiment for the image of Lena (top curve) and Mezzanine (bottom curve). The 'x'-points mark the real computed points on the R-D curve while the red curve represents the smoothed approximation of the global allocation curve. The Rate is expressed in nts/coeff and the point marked with an 'o' represents the selected encoding rates and distortion values of the images that have been stored into DNA.

(a) Lena original image

(b) Mezzanine original image

(c) Stored Lena image
PSNR = 32.5 dB
Rate = 2.68 bits/nt

(d) Stored Mezzanine image
PSNR = 29.67 dB
Rate = 1.78 bits/nt

Figure 7.3: The two images selected for our wet-lab experiment. Figures a and b show the original images whereas figures c and d depict the compressed versions which have been selected using the source allocation and have been stored into DNA. The PSNR values are computed with respect to the original images.

For our wet-lab experiment we used the Rate-Distortion curves provided by the allocation method using spline approximation as depicted in figure 7.2. More precisely we stored the image of Lena at a rate of $R = 2.98$ nt/pixel or equivalently $R = 2.68$ bits/nt for an $MSE = 36.43$ and a $PSNR = 32.5$ dB. The image of Mezzanine has been stored at a rate of $R = 4.49$ nt/ pixel or equivalently $R = 1.78$ bits/nt for an $MSE = 70.15$ and a $PSNR = 29.67$ dB. The reason for selecting these values of quality relies only on the budget (in terms of euros) which was available for the experiment.

For the primer fields we used the following sequences which have been provided by the IPMC laboratory in Sophia Antipolis:

- 3′ end primer: 'GTTCAGAGTTCTACAGTCCGACGATC'
- 5′ end primer: 'TGGAATTCTCGGGTGCCAAGG'

The above primers have been selected as they had reported a good performance for the Illumina sequencer in previous sequencing experiments carried-out by the CNRS/IPMC laboratory[1] For the sense fields we have used the nucleotides 'A' and 'T' to denote the 3′ end of the synthesized oligos and a 'C' or a 'G' to mark the 5′ end. We recall that during sequencing

[1]Institut de Pharmacologie Moléculaire et Cellulaire à Sophia Antipolis, France.

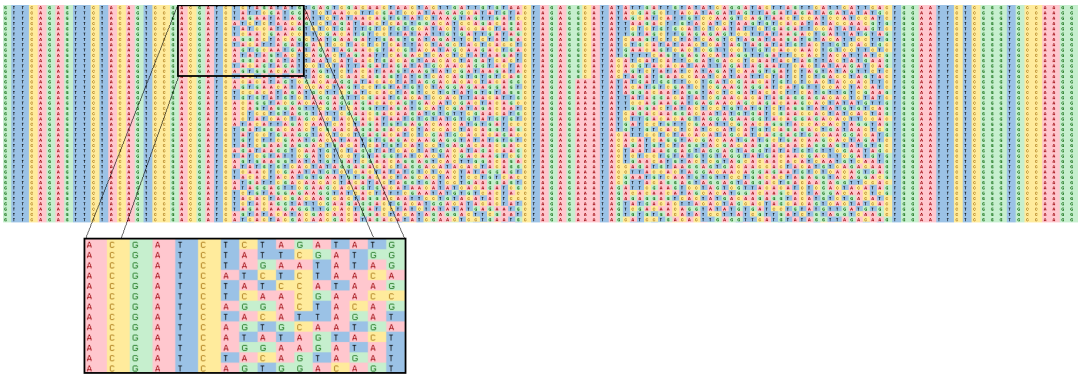


Figure 7.4: Part of the real formatted oligos that have been synthesized during our wet-lab experiment.

the produced reads will correspond either to an estimation of the synthesised oligo strand as is (following the correct sense) or to an estimation of the complementary strand in the reverse sense. Consequently, some of the reads received after sequencing will need to be reversed and complemented before decoding. Therefore, receiving a C or a G nucleotide in the beginning of some read denotes such a case.

As both images have been stored in the same pool of oligos we marked each image using a different identifier. More precisely, for the ID fields we used the following identifiers:

- Lena image: 'CTAG'
- Mezzanine: 'CTAC'

In this particular wet-lab experiment the parity field has not been used and thus the parity field is only one nucleotide long instead of 4 nts (as proposed in section 6.2) and has been filled-in randomly.

After the formatting we produced in total 662 oligos of length 138 nts for the image of Lena and 875 oligos of length 138 nts for the image of Mezzanine. The encoded oligos have been then sent for synthesis. Figure 7.4 depicts some of the encoded oligos that have been used in our wet-lab experiment.

7.3 DNA Synthesis and storage

For the synthesis of the produced oligos we colaborated with a well-known public company based in San Francisco that manufactures synthetic DNA for clients in the biotechnology industry. Twist was founded in 2013 by Emily Leproust and Bill Peck, who each had worked on DNA synthesis technology at Agilent Technologies, and Bill Banyai. The company sells the genes, gene fragments, and oligonucleotides to customers who use them in fundamental research such as the research of DNA data storage presented in this thesis.

On the 23rd of June 2018 the order of oligos has been placed and the DNA pool of synthesized DNA oligos depicted on the left image of figure has been delivered the 3rd of July 2018.

The product delivered by Twist Bioscience had to be stored at -20 °C to guarantee storage up to one year or at -80°C for longer preservation. It is important to denote that the oligos that have been ordered should then be amplified so to have enough oligos for performing further experiments. Thus,

Figure 7.5: Left image: Pool of synthesized oligos produced by Twist Bioscience. Right image: Capsules containing the synthesized DNA libraries. The capsules have been provided by Imagene company which is located in Evry in France.

reading a sample of the same synthesized pool of oligos in the future will not require a new DNA synthesis. To produce enough copies of all the synthesized oligos, our collaborators in IPMC performed simple PCR starting with 10ng of DNA product and amplifying it to 100ng creating many samples of the initial DNA pool, the so-called libraries.

All libraries have been then stored into some specially designed capsules provided by the company Imagene, based in Evry in France which allow long-term preservation in room temperature. Imagene is a company which proposes unique and disruptive solutions allowing the preservation of biospecimens at ambient temperature, thus to be free from the cold, its technical constraints, its risks, its costs in equipment, energy and maintenance. This preservation technology guarantees the preservation of the samples in conditions never reached before on the market. This encapsulation is the final part of the DNA data storage process and guarantees reliable preservation in room temperature for many years. We therefore safely stored our pool of oligos hoping that some day in the long future those two images will still be available for next generations to decode. The encapsulated pools of oligos form our wet-lab experiment are depicted in the rightt image figure .

7.4 DNA sequencing and oligo selection

Since one of the most challenging parts of this study is the errors inserted in the sequencing process, in our wet-lab experiment we tested the efficiency of the proposed workflow by reading and decoding our synthesized oligos. Thus, after amplification, one of the amplified pools has been sequenced using the Illumina Next-Seq 500 technology. As explained in previous sections the Illumina machine performs sequencing by synthesis. In other words the machine can read the oligos by creating clusters of copies of each oligo bound on the flow-cell. Thus, the result of the reading process is a set of reads that constitute estimations of the original synthesized oligos. For each original oligo, the sequencer provides a set of reads which might contain substitutions, insertions, deletions or ambiguous base-calls marked with the symbol 'N'. Therefore, decoding is a challenging process and is possible only after pre-processing the set of reads provided by the Illumina machine. In more detail, during our experiment the Illumina machine produced 625,075,716

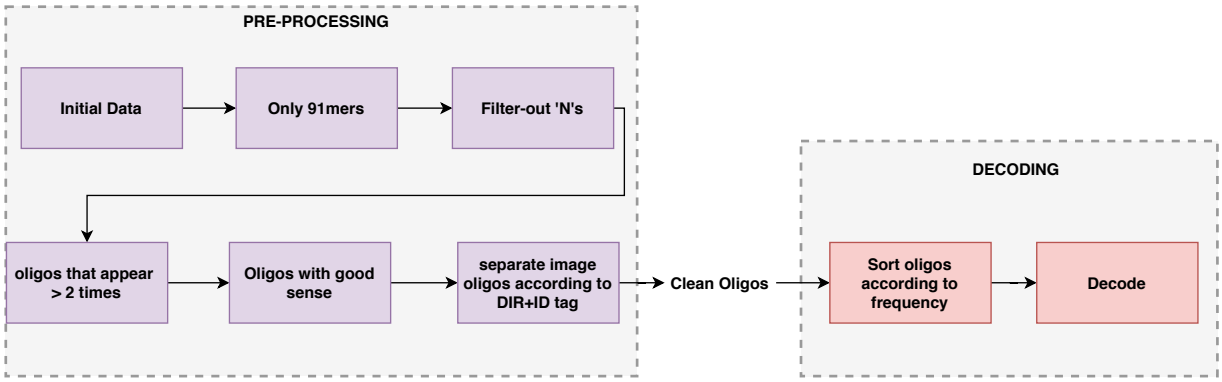


Figure 7.6: Oligo pre-processing workflow

reads in total In this particular experiment part of the oligos were used for another experiment, managed by EURECOM in Sophia Antipolis and thus a small fragment of the above oligos contains data of SQL tables that have been used in their experiments. The pre-processing workflow for cleaning the set of oligos to initiate decoding is described by figure 7.6.

The raw-data (initial data) as produced by the sequencer are first filtered to keep only the oligos which have the length of the synthesized DNA strands which are 91nts long (91-mers). This step is reducing the number of oligos that have been corrupted with insertion or deletion errors (indels) which are the most difficult to treat since they can shift big part of the oligo affecting the decoding. However, in the pool of 91-mers which are kept after this first filtering, there might remain some oligos which have been affected by indels as in the case where the number of insertions is equal to the number of deletions the oligo length is not affected. The remaining oligos are then checked for containing nucleotides marked as 'N's keeping only the oligos which do not contain such ambiguous base-calls. In the next stage of filtering we impose some threshold according to the number of copies of each read. More precisely, to reduce the number of errors and under the assumption that if an oligo read has a low number of copies it is more probable to have been created after some error during reading, we have kept only the reads that were found to have at least two copies in the remaining set of oligos. Finally, as the sense and ID fields are crucial for the correct decoding of the stored information we scan the set of oligos for strands in which the above fields have been affected. We therefore reject any reads that contain 'A'/'T' or 'C'/'G' in both the beginning and in the end and we separate the rest of the oligos according to the ID fields to decode each of the stored images. The oligos which are kept after the above pre-processing workflow can be passed in the decoder.

7.5 Decoding

The output of the pre-processing workflow provides a set of reads that occur in multiple copies. We will thereby refer to this number as the oligo's frequency. As explained in previous sections this redundancy is necessary for a more reliable decoding and occurs due to the amplification of the synthesized oligos during sequencing. One can imagine the procedure of sequencing like the classical repetition coding used for transmission over a noisy channel that may corrupt the transmission in various positions. As in repetition coding,

the main idea of amplification is to repeat the oligos several times hoping that the sequencing corrupts only a minority of these repetitions. Under this simple hypothesis, we assume that after all the copies of oligos are sequenced, one can apply the method of majority vote to distinguish the most representative oligos. We therefore create a matrix Y_o by sorting the remaining oligos according to their frequency. More precisely, each line of Y_o will represent one read and the lines are sorted such that the most frequent oligos appear in the lowest lines. The produced matrix will not contain multiple entries of the same reads. The frequency of each read is denoted using a vector f containing the number of copies of each line of the matrix. We then start the decoding process. For the decoding we first create a matrix Y_d of zero values of size M by N. The idea is to read each line of Y_o and fill the corresponding cells of Y_d with the decoded data of each oligo according to the decoded address fields. In the case that some cells of Y_d are already filled with some decoded value which might have occurred by decoding some read of lower frequency, the existing information is overwritten with the new data that appear lower in Y_o and therefore have a higher frequency.

To prove our claim that the most frequent oligos will be the more reliable ones we performed two different scenarios of decoding. The first scenario is the decoding method discussed above, which is sorting the oligos according to frequency and thus is overwriting the data existing in the most frequent oligos above the previous ones found in the oligo pool, and a second scenario where we decode using the not sorted matrix of oligos and the overwriting is random. The result of those two decoding scenarios are presented in figure 7.7.

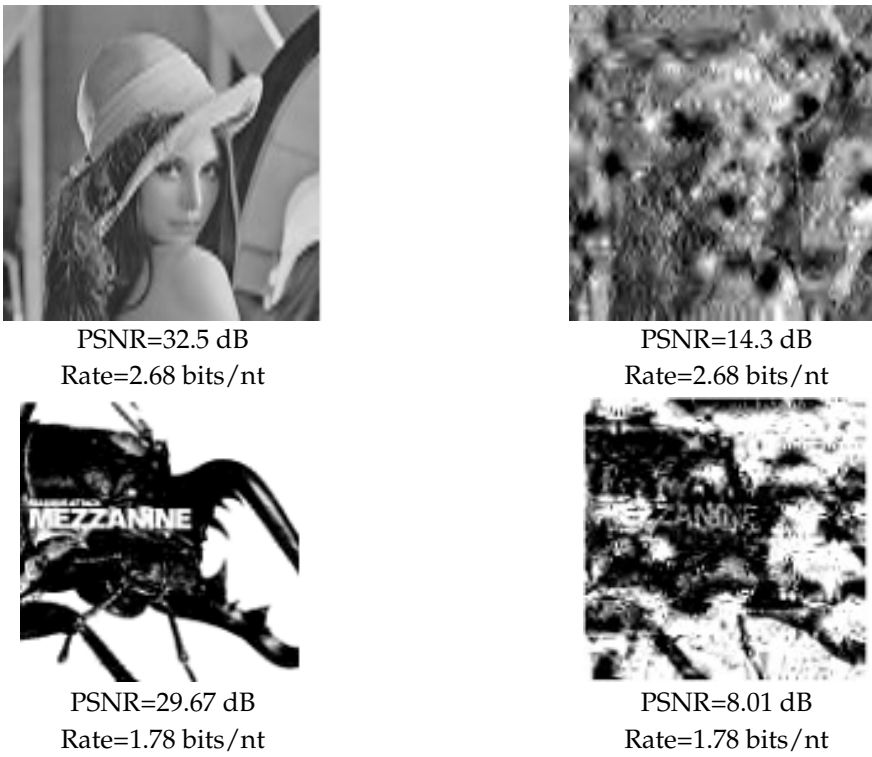


Figure 7.7: Visual results for two different cases of reconstruction: using the most frequent oligos (left column) and random selection (right column). For the left images the PSNR value is only due to the error inserted by the quantization process as we have managed to get a reconstruction without any sequencing noise, For the right images (random selection), both quantization and sequencing error appear.

7.6 On the reduction of the DNA sequencing cost

As explained, DNA synthesis and sequencing are expensive and can cost several thousands of dollars depending on the size of the encoded data. On one hand, in order to reduce the synthesis cost, we proposed compressing the input image and control the coding rate thanks to a nucleotide-allocation algorithm. By doing so, one can select an optimal rate which provides the minimum distortion in the visual result. On the other hand, the sequencing cost can be decreased by reducing PCR and BA cycles which have been explained in sections 2.1.1 and 2.1.3 respectively, and create multiple copies of the synthesized strands to introduce the desired redundancy for robustness to sequencing errors. We remind that PCR and BA take place before and during sequencing respectively and can provide correct or erroneous clones of the synthesized data. The reduction of the PCR and BA cycles can be simulated by subsampling the data set of sequenced oligos which have been provided by our wet-lab experiment. In the initial experiment we discarded the oligos exceeding 91 nts as those oligos for the moment can not be decodable. This resulted to a total number of 28,876,259 sequenced oligos, from which the initial encoded image should be reconstructed. In this experiment we have subsampled this initial number of oligos using different sampling sizes. For each case, the most frequent oligos from the subsampled data were

selected as the most representative sequences and were used to decode and reconstruct the image.

Consequently, for the decoding, we have subsampled the sequenced data and reconstructed an image for each sampling size. The procedure was repeated twenty times for each sampling size and we computed the average of the Peak Signal to Noise Ratio (PSNR) as well as the percentage of exact matches when comparing the most frequent sequenced oligos with the encoded correct ones. Those results are presented in figure 7.9 and provide information about the quality of the reconstruction. In figure 7.8 we present the visual results for different sampling rates as also the corresponding values of PSNR and SSIM.

Interestingly enough we observe that we can achieve a perfect reconstruction by using only a small percentage of the sequenced oligos provided by our experiment. More precisely we can see in figure 7.9 that only using 5000 samples of the sequenced oligos we get 100% correctness when comparing to the 662 original synthesised oligos. Taking into account the initial number of amplified data (28,876,259 oligos) we conclude that only 0.0173% of those oligos are needed for perfect decoding. This can be confirmed by the evolution of the visual results in figure 7.8. The above study has been published as a conference paper in GRETSI 2019 [65].

#samples = 1500 $PSNR = 13.05dB$ $SSIM = 0.59$ | #samples = 2250 $PSNR = 28.12dB$ $SSIM = 0.95$ | #samples = 2500 $PSNR = 40.79dB$ $SSIM = 0.97$ | #samples = 3500 $PSNR = 47.53dB$ $SSIM = 0.99$ | #samples = 5000 $PSNR = inf$ $SSIM = 1$

Figure 7.8: Visual results after decoding at different subsampling rates. Lena 128x128 pixels

7.7 Two years later...

Since the wet-lab experiment which has been described in this chapter has been carried-out in 2018 we had the opportunity to test the decodability of the data 2 years after the storage of the DNA oligos in the capsules. To this end, our collaborators in the IPMC laboratory in Sophia Antipolis in France have run a new round of sequencing using one of the stored libraries of oligos and the same sequencing machine (Illumina NextSeq 500) that had been used in the previous experiment. To test the efficiency of the sequencer with this 2 years old library, they have plot the number of times that each correct oligo was read by the sequencer in this new sequencing round in function of the one of two years ago. The result of this comparison is depicted in figure 7.10.

It is important to mention that the frequency of some oligo in the new experiment depends on multiple factors. First of all in this new experiment, a new library has been used. Since the libraries are built by taking a sample of the PCR amplified pool of oligos, it is normal to select different amount of copies for each correct oligo existing in the pool. In addition to this, in the

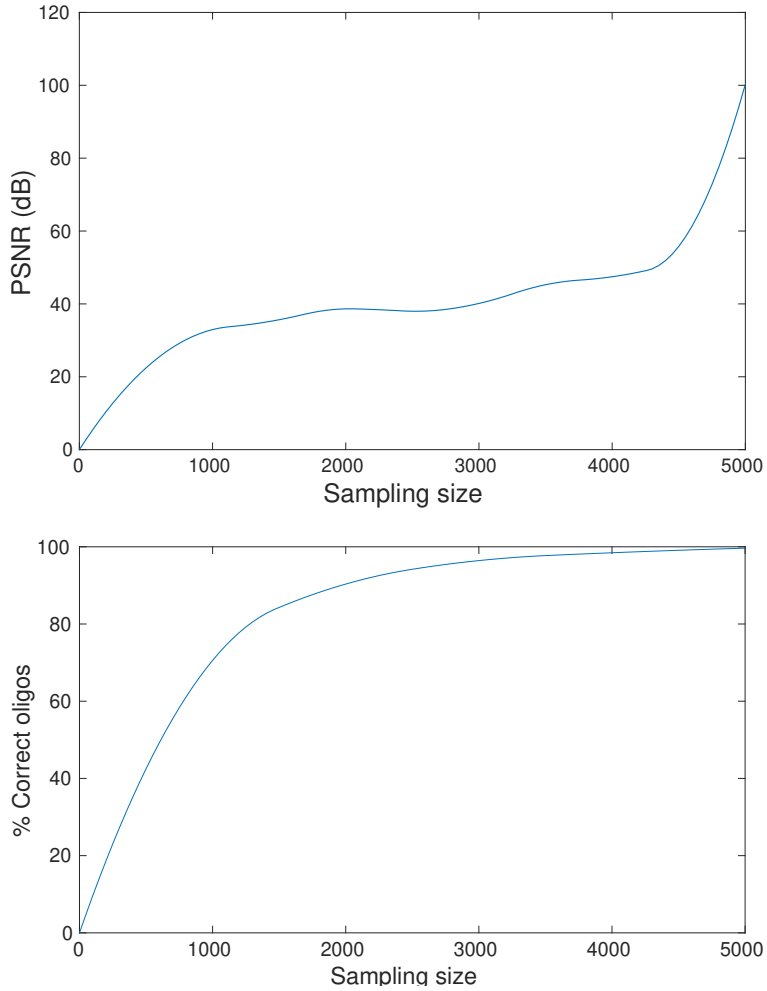


Figure 7.9: The evolution PSNR and percentage of correct oligos for different sampling sizes

experiments of IPMC they have noticed that PCR amplification was much less performant for some oligos rather than others. Therefore, in this new experiment the frequency of an oligo also depends on the efficiency of PCR as well as the efficiency of BA (Bridge Amplification) during sequencing. We recall that BA is a process used during Illumina sequencing which creates copies of the oligos while reading them. Thus the number of times that a correct oligo appears in the reads of this new experiment does not necessarily only depend on the degradation of time. It is however proven, that even if some oligos exist in few copies, we have still been able to recover all the correct oligos which correspond to the stored data.

In our first experiment we have assumed that the correct oligos appear in more copies than their erroneous representations. To check if this assumption stil stands in the new sequencing, we decoded again the two stored images of Lena and Mezzanine using the most frequent oligos for the decoding. The result was a perfect reconstruction of the two encoded images. Consequently we can conclude that after two years of storage correct decoding is still possible when using the most frequent oligos for the reconstruction. To verify our results we also performed the experiment of decoding using random copies of the oligos rather than the most frequent ones. Similarly to the experiment

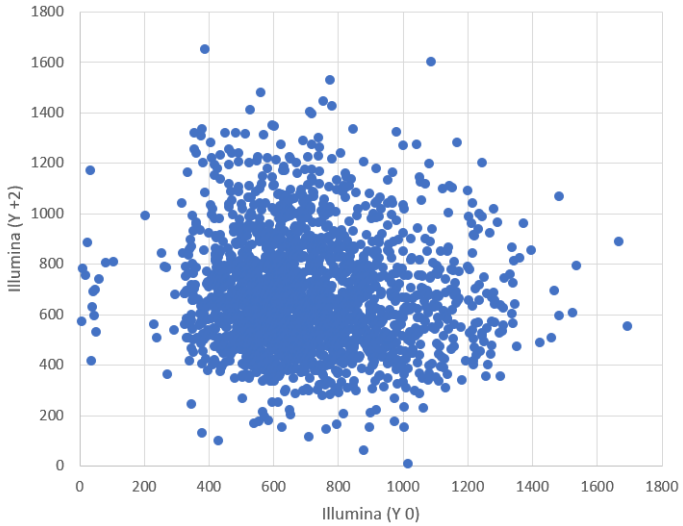


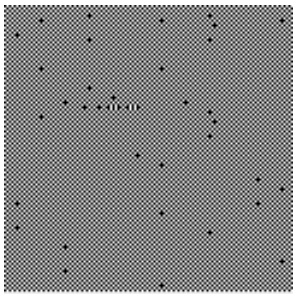
Figure 7.10: The number of copies of correct oligos in the second sequencing experiment (after 2 years of storage) in function of the number of correct oligo copies in the first sequencing experiment. Each point in the point cloud corresponds to a different correct oligo. The axis represent the number of identical correct copies of each oligo. The x-axis corresponds to the sequencing experiment in 2018 and the y-axis corresponds to the experiment of 2020.

performed two year earlier, this last attempt provided an erroneous reconstruction. The visual results for both decoding cases is depicted in figure 7.11. Our future works, include the testing of the rest of the proposed encoding scenarios, by a new wet-lab experiment.

PSNR=32.5 dB
Rate=2.68 bits/nt

PSNR=21.76 dB
Rate=2.68 bits/nt

PSNR=29.67 dB
Rate=1.78 bits/nt



PSNR=3.48 dB
Rate=1.78 bits/nt

Figure 7.11: Visual results of the decoding of the stored oligos two years after storage. The figures correspond to 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.

7.8 Conclusions

In this chapter we presented the details of the wet-lab experiment which has been carried-out to verify feasibility of our theoretical studies. Due to limitations of budget and time due to the ongoing pandemic of COVID-19 our study has been restricted to only testing our first proposed fixed-length encoding workflow which is using a Scalar Quantizer for the compression. This single experiment has nevertheless allowed us to study the noise caused by the Illumina sequencer, which is the most reliable and most widely used sequencing machine which exists until today. It is however in our first priorities to perform more wet-lab experiments for testing the rest of the encoding solutions proposed in this thesis while we have already prepared our next oligos to be sent for synthesis in the following weeks. This new experiment will also include testing the Nanopore sequencing and the robustness of our solutions to the Nanopore's increased sequencing noise.

Chapter 8

General conclusions and future steps

8.1 Conclusions

In conclusion, in this thesis we have presented a study on the efficient storage of digital images into DNA. Motivated by the need to reduce the high DNA synthesis cost, we proposed for the first time an encoding solution which allows controlling the compression of the input to produce an efficient encoding and reduce the cost of DNA synthesis. More precisely, in our encoding we used a workflow which efficiently compresses an input image using a DWT and quantizing each of the produced subbands independently to be then encoded in a quaternary representation of A, T, C and G. To this end, we introduced a novel algorithm for creating a fixed length quaternary code which can provide an encoding that respects the necessary sequencing constraints to reduce the probability of sequencing errors. Our proposed algorithm for constructing this code differs from the existing ones in the state of the art as it is fixed length and can therefore be easily used to perform a closed-loop nucleotide allocation to optimize the compression of some input image without requiring high computational cost. Furthermore, in contrast to the state of the art encoding methods, our proposed solution can be applied for encoding any type of data and is not restricted to binary inputs.

We also proposed a new mapping method which makes use of any unused codewords in the produced constrained code to allow double representation of the most frequent coefficients and deal with pattern repetitions which can lead to sequencing errors during reading. We then further improved our compression workflow by using a Vector Quantizer instead of a Uniform Scalar one. This last case allowed us to introduce a new sequencing-noise resistant mapping algorithm to assign the vector indices produced by VQ to the quaternary codewords of our code. This mapping aims in finding the optimal assignment to reduce the visual impact of substitution errors in the decoded image.

In our studies we mainly focused in providing a robust fixed-length encoding solution which allows controlled compression for reducing the synthesis cost. The selection of a fixed-length encoding instead of a variable-length one stems from the fact that since sequencing is prone to errors, in the case of a variable length encoding an error can lead to losing the structure of the data and lead to very poor decoding. To prove this claim, we have thus implemented two different variable length solutions which are based on the JPEG standard. The first solution is the one that has been mostly used by

the state of the art and uses the classical JPEG protocol to compress the input image into an optimized binary representation and then encodes the binary stream into a quaternary representation. As this solution is not optimized for a quaternary representation and requires passing from binary before encoding the input into DNA we also proposed a new modified version of the JPEG algorithm by including the quaternary code inside the existing standard. In more detail, we used the same encoding workflow as the one of classical JPEG for binary but we replaced the variable-length coding by combining our proposed fixed length PAIRCODE algorithm for encoding the values and the algorithm proposed by Goldman *et. al.* [1] for the encoding of the runs of zeros and categories. We then compared all the proposed encoding methods in terms of compression efficiency and robustness. This comparison has proven the fixed-length encoding methods to be much more robust to sequencing noise. Interestingly enough the results of the fixed-length method of nt allocation using VQ are comparable to the results of variable length encoding. We therefore concluded that there is an important trade-off between compression efficiency and robustness to error which has to be considered for the selection of the most preferable encoding solution.

As DNA synthesis requires cutting and formatting the encoded information into smaller oligos and since the proposed encoding workflow is more complex than the ones proposed in bibliography, we have proposed different formatting scenarios to suit each encoding method. Furthermore, since the header information used for the formatting is highly important for a reliable decoding we have proposed robustifying some fields using the method of Error Correcting Barcodes.

Finally, during this thesis we have performed a wet lab experiment on one of the proposed workflows testing the efficiency of our quaternary code construction when used along with an Illumina Sequencer. It is important to denote that as the topic of this thesis is relatively new and some of the techniques used in the encoding (such as the nucleotide allocation and the creation of the training sets for VQ) are complex and require long execution time, it has been very challenging to provide further comparisons and results on different images. However, we recognise the importance of performing further experiments after the end of this thesis to provide a more complete study on the efficiency of the different workflows.

8.2 Discussion

This thesis presents the first attempt of introducing controlled compression in DNA data storage by proposing an end-to-end solution which is adaptable to the needs of the encoding and can reduce the high synthesis cost. The produced results are very promising and set the ground work for further study and improvement. As DNA data storage is a very challenging multidisciplinary field of study that highly depends on biological manipulations, it is expected to evolve along with the changes in the methods and the machines used for DNA synthesis and sequencing. Thus the encoding methods might change in the following years so to respect different encoding constraints. In addition to this, since this is a relatively new topic of research with great potential in future applications, it is sure to attract much interest in the next few years, hoping that the more it will be studied, the more the

cost of biological processes such as DNA synthesis and sequencing will be reduced.

It is very encouraging to notice that there is already a great interest on the topic by many different research groups around the world. Namely, during this thesis, we had the chance to collaborate with some of those teams through the OligoArchive project Horizon 2020 [1] which is founded by the European Union. This collaboration includes the I3S/CNRS laboratory, the IPMC/CNRS and EURECOM which are located in Sophia Antipolis in France, as well as the Imperial College of London in the UK and the startup of Helixworks which is synthesizing DNA and is located in Ireland. This project aims in the creation of a prototype system will allow the research of the whole cycle from encoding to the sequencing of data to DNA. Furthermore, recently, the JPEG community has launched an Ad Hoc Group on Digital Media Storage using DNA [66], in which we have the honor to participate as invited experts on the topic of DNA data storage. We therefore hope that these collaborations will create some wonderful ideas that will help the field advance rapidly so to be soon used in practice.

Another important point to be discussed is the fact that DNA data storage is intended for the archiving of digital data to be decoded in the very long term. The reading of DNA will always be guaranteed, since the molecule of DNA exists in every living organism and thus there will always be some machine for its reading. However, it is fundamental to find a way for ensuring that the decoder as well as the information for the decoding will be available when needed to allow correct reconstruction after reading. Some interesting works on digital preservation propose some solutions for creating durable ways for storing the information for decoding while also expressing it in a way that will be understandable by anyone in the future that might have no previous knowledge on the encoding. Such solutions include the storage of the decoding information in microfilms, as proposed by the company of EUPALIA in France [12] and in their latest collaboration with EURECOM in [67] or in silica glass. Some interesting works for storing data in silica glass have been proposed in [13].

8.3 Future work and perspectives

DNA data storage is a very promising new field of research which is expected to play a significant role in the solution of fundamental challenges of digital data storage. However, since it constitutes a multidisciplinary subject which is highly constrained by some limitations of the biological manipulations, there are multiple challenges to be addressed in the encoding of digital data into DNA. This thesis sets the groundwork for further improvement for the creation of an optimal image coding workflow for the storage of digital images into DNA. By summing up all the points discussed and studied by this study as well as the state of the art works one can conclude to the following challenges to be addressed:

- Biological constraints - The encoding workflow must consider the relevant biological constraints on the coding process to avoid affecting the stability of the sequence against sequencing errors.

[1] https://oligoarchive.eu

- Compression efficiency - Due to the high cost of DNA synthesis, the codec shall offer significantly increased compression efficiency regarding simple solutions in the literature, e.g. based on binary coding.
- Random access - To reduce the sequencing cost, the encoding should allow the access to specific parts of the information without having to decode/sequence the full coded information.
- Error resilience - The encoding workflow must offer error robustness and error correction reading/sequencing errors.
- Scalability - The encoding shall allow scalable representations of the information where reading only part of the full information offers a lower quality or resolution of the full represented information.
- Ambiguity - The workflow shall allow decoding without any ambiguity, i.e. a decoded bit may not be both '0' and'1'.
- Artificial recognition - The encoding shall allow the encoding output to be unambiguously recognized as artificial; this may be relevant if the artificial DNA stream should not be confused with natural DNA streams.

While this thesis took a very first step in the introduction of a closed-loop codec for the efficient compression of digital images into DNA, there is still room for improvement on both the fixed-length encoding by using for example Lattice VQ instead of a classical VQ, and the proposed variable-length codec by improving the encoding workflow and introducing error-correction techniques. Furthermore, another interesting point which would help the improvement of the encoding workflows is the study of the noise of different sequencers in order to robustify the encoding. Such a study would provide extra information about the sequencing constraints and would allow the generation of DNA strands which are more adapted to the biological experiments of synthesis and sequencing. At the same time, since a big percentage of the end-to-end workflow is dedicated to biological experiments, the advances in the methods used during the biological manipulations can significantly improve the quality of the encoding. For example, since the nanopore sequencer is a user-friendly, low-cost sequencing method which allows the reading of longer oligos, many companies which perform DNA synthesis (such as Helixworks and DNA Script) are currently studying the possibility of synthesizing longer DNA strands which are free from synthesis error. It is therefore clear that there are many interesting paths to explore from both the computer science and the biological perspective so that DNA data storage will be soon a viable option towards a greener solution for long-term archival.

Appendix A

Algorithms

A.1 Section 1

Algorithm 1 Encoding Algorithm for a known set of symbols Σ (K is given)

Initialise the number m of representations needed for each symbol to encode
Compute length l of codewords needed for encoding $m * K$ symbols $s_i \in \Sigma$ as:
if $\log_{10}(mK)$ not an integer **then**
  if $10^{\lfloor \log_{10}(mK) \rfloor} * 4 \leq K$ **then**
    $l = \lfloor \log_{10}(mK) \rfloor * 2 + 1$
  else $l = \lceil \log_{10}(mK) \rceil * 2$
  end if
else$l = \log_{10}(mK) * 2$
end if
Build code $\mathcal{C}^*$ of L different codewords c_i:
if l is even **then**
  Construct all possible codewords of length l using $\frac{l}{2}$ choices from $\mathcal{C}_1$
else if l is odd **then**
  Construct all possible codewords of length l by using $\frac{l}{2}$ choices from $\mathcal{C}_1$ adding one symbol from $\mathcal{C}_2$
end if
Mapping of index values of quantization to codewords from $\mathcal{C}$
Compute: $\Gamma(s_i) = \mathcal{C}^*(i + rand(1, m-1) * K)$

Algorithm 2 Encoding Algorithm when optimizing K given a target compression rate R_{target}

$R_{target} = l$
Compute size of the codebook L:
if l is even **then**
  Construct codebook of length $L = 10^{\frac{l}{2}}$
else Construct codebook of length $L = 10^{\frac{l-1}{2}} * 4$
end if
Compute $K = \frac{L}{m}$ (optimal solution for $m = 2$)
Mapping of symbols to codewords from $\mathcal{C}^*$
Compute: $\Gamma(s_i) = \mathcal{C}^*(i + rand(1, m-1) * K)$

Algorithm 3 Mapping Algorithm

Definitions:

Set $\mathcal{V} = \{v_1, v_2, \ldots, v_K\}$ of codevectors v_k , $|\mathcal{V}| = K$
Set $\mathcal{C}^* = \{c_1, c_2, \ldots, c_L\}$ of quaternary words c_l, $|\mathcal{C}^*| = L$
Define a mapping function $\Gamma : \mathcal{V} \mapsto \mathcal{C}^*$
Set $C \subseteq \mathcal{V}$ of vectors $v_i : \Gamma(v_i) = \varnothing$
$H(c_l)$: Set containing codeword c_l and the B_i codewords c_n that differ from c_l in one nucleotide (Hamming distance of 1)
Define $B = \max_i(B_i)$ with $i \in \{1, 2, \ldots, L\}$
$S(v_k)$: Set containing vector v_k and its B closest neigboring vectors v_n
Function $\phi(v_q) = \sum_{j|v_j \in S(v_q)} d(v_j, v_q)$ with $\beta \geq 0$ a trade-off parameter.
Empirical function: $F(v_q) = p(v_q)/\phi^{\beta}(v_q)$ where $p(v_q)$ is the probability of v_q in the input sequence

Phase 0: For the first (B+1) indices

Initialise: $C = \mathcal{V}$
$v_q := \{v_i \in C : v_i = \arg\max_v F(v)\}$
$\Gamma(v_q) = c_q$
$\Gamma(v_n) = c_n$, $\forall v_n \in S(v_q)$, $c_n \in H(c_q)$

Phase 1: For the remaining indices

while $C \neq \varnothing$ **do**
  $C - \{v_q, v_n\}$
  $v_q := \{v_i \in C : v_i = \arg\max_v F(v)\}$
  if $\Gamma(v_n) = \varnothing, \forall v_n \in S(v_q)$ **then**
    if $\exists c_q : \forall c_n \in H(c_q), \Gamma^{-1}(c_n) = \varnothing$ **then**
      $\Gamma(v_q) = c_q$
      $\Gamma(v_n) = c_n$,$\forall v_n \in S(v_q)$, $\forall c_n \in H(c_q)$
    else
      $H := \arg\max_s(|H_s|)$, $\forall s \in \{1, 2, \ldots, K\}$,
      $\Gamma^{-1}(H_s) = \varnothing$
      $\Gamma(v_q) = c_q$
      $\Gamma(v_n) = c_n, \forall c_n \in H(c_q) : \Gamma^{-1}(c_n) \neq \varnothing$
      and $v_n = \arg\min_v(d(v_q, v))$ with $v \in S(v_q)$
    end if
  else
    if $\Gamma(v_q) = \varnothing$ but $\Gamma(A_q) \neq \varnothing$ with $A_q \subseteq S(v_q)$ **then**
      if $\exists c_q$: $\Gamma(A_q) \subseteq H(c_q)$ **then**
        $\Gamma(v_q) = c_q$
      else
        $H := \arg\max_s(|H_s \cap \Gamma(A_q)|)$ $\forall s \in \{1, 2, \ldots, K\}$
        if $|\arg\max_s(|H|)| \geq 2$ **then**
          Let $c_n^j, j = 1, 2, \ldots,$ be the tied indices
          Define $a = v_n | \Gamma(v_n) \in \Gamma(A_q) \cap H(c_q)$
          Define $b = v_n | \Gamma(v_n) \in \Gamma(A_q) \cap H(c_n^j)$
          Assign to vector v_q a word c_q such that:

$$\sum_a d(v_n, v_q) = \min_q \sum_b d(v_n, v_q)$$

        end if
      end if
    end if
  end if
end while

Appendix B

Training set

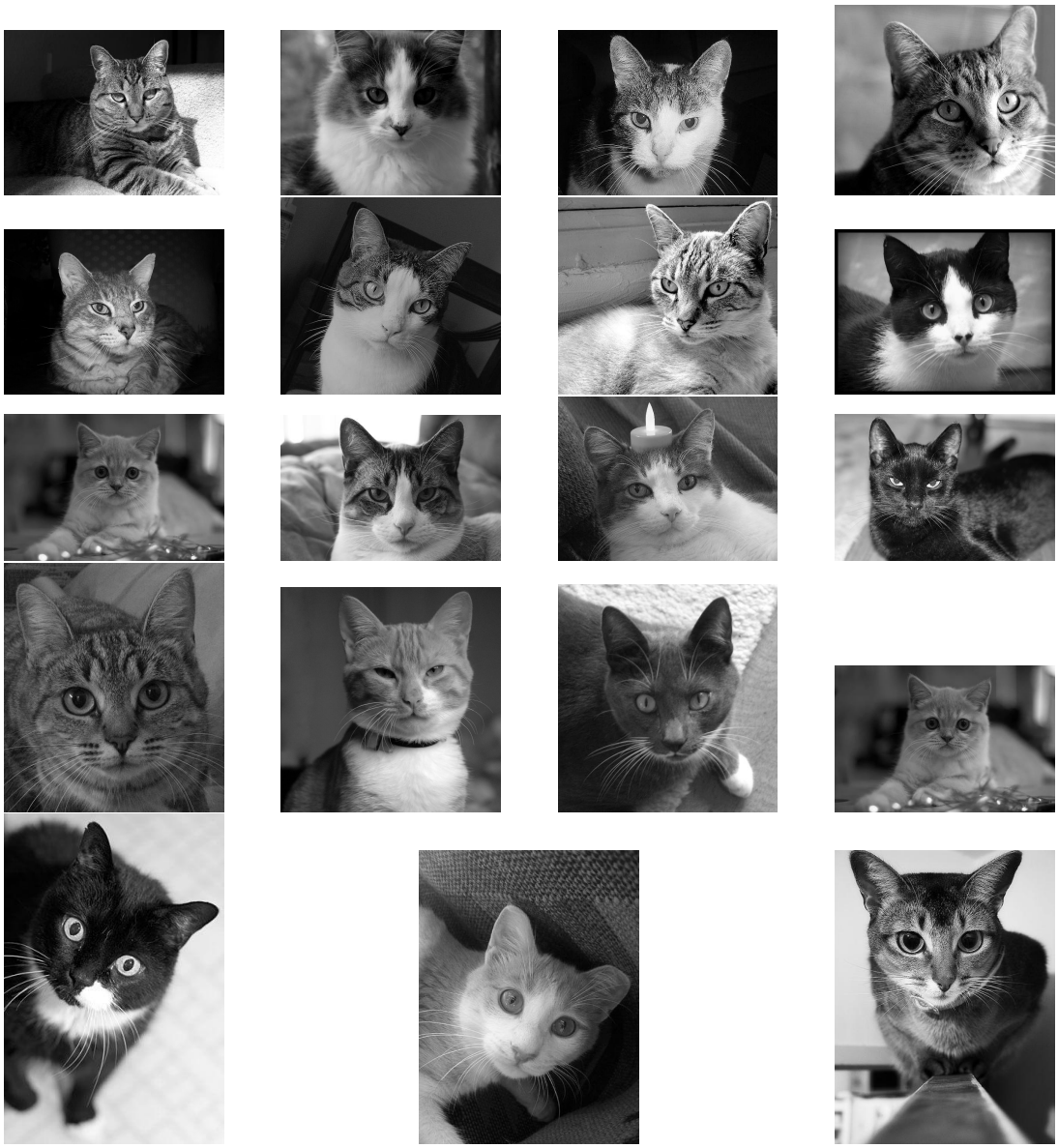


Figure B.1: Training set of cat images used for training the codevectors in our VQ experiments.

Appendix C

Publications

C.1 Journals

1. Melpomeni Dimopoulou, Marc Antonini, Pascal Barbry, Raja Appuswamy, *"Image Storage onto Synthetic DNA"*, submitted to Signal Processing: Image Communication journal, ELSEVIER, Oct. 2019.
2. Melpomeni Dimopoulou, Marc Antonini, *"Data and image storage on synthetic DNA - Existing solutions and challenges"*, submitted to Journal on Image and Video Processing, EURASIP, Oct. 2020.

C.2 Conference papers

1. Melpomeni Dimopoulou, Marc Antonini, *"Signal Quantization using the Leaky Integrate-and-Fire neuron"*, Groupe de Recherche et d'Etudes du Traitement du Signal et des Images 2017, Juan-Les-Pins, France
2. Melpomeni Dimopoulou, Effrosyni Doutsi, Marc Antonini, "A Retina-Inspired Encoder: An Innovative Step on Image Coding Using Leaky Integrate-and-Fire Neurons", International Conference on Image Processing, 2018, Athens, Greece
3. Melpomeni Dimopoulou, Marc Antonini, Pascal Barbry, Raja Appuswamy, *"DNA coding for image storage using image compression techniques"*, COmpression et REprésentation des Signaux Audiovisuels, 2018, Poitiers, France (Best Paper Award)
4. Melpomeni Dimopoulou, Marc Antonini, Pascal Barbry, Raja Appuswamy, *"A biologically constrained encoding solution for long-term storage of images onto synthetic DNA"*, European Signal Processing Conference 2019, A Coruña, Spain (Best Student Paper Award)
5. Melpomeni Dimopoulou, Eva Gil San Antonio, Marc Antonini, Pascal Barbry, Raja Appuswamy, *"On the reduction of the cost for encoding/decoding digital images stored on synthetic DNA."*, Groupe de Recherche et d'Etudes du Traitement du Signal et des Images 2019, Lille, France.
6. Melpomeni Dimopoulou, Marc Antonini, Pascal Barbry, Raja Appuswamy, *"Storing Digital Data into DNA: A Comparative Study of Quaternary Code Construction"*, International Conference on Acoustics, Speech, and Signal Processing, 2020, Barcelona, Spain.

7. Melpomeni Dimopoulou, Marc Antonini, "Efficient storage of images onto DNA using vector quantization", Data Compression Conference, 2020, Utah, USA.

8. Melpomeni Dimopoulou, Marc Antonini, *"Image storage in DNA using Vector Quantization"* European Signal Processing Conference 2019, 2020, Amsterdam, Netherlands.

9. Melpomeni Dimopoulou, Eva Gil San Antonio, Marc Antonini, *"A quaternary code mapping resistant to the sequencing noise for DNA image coding"*, International Workshop on Multimedia Signal Processing, 2020, Tampere, Finland.

10. Eva Gil San Antonio, Mattia Piretti, Melpomeni Dimopoulou, Marc Antonini, *"Robust image coding on synthetic DNA: Reducing sequencing noise with inpainting"*, International Conference on Pattern Recognition, 2020, Milan, Italy.

11. Melpomeni Dimopoulou, Marc Antonini, *"A JPEG-based image coding solution for data storage on DNA"*, submitted to European Signal Processing Conference (EUSIPCO), 2021, Dublin, Ireland.

C.3 Patents

1. Melpomeni Dimopoulou, Marc Antonini. *METHODS FOR STORING DIGITAL DATA AS, AND FOR TRANSFORMING DIGITAL DATA INTO, SYNTHETIC DNA.* United States, Patent n : 16/811,985. 2020 (Granted).

C.4 Awards

1. Best paper award at CORESA (2018) for "DNA coding for image storage using image compression techniques".

2. First prize in the regional contest of "Three Minute Thesis" of Côte d'Azur in 2019.

3. Best student paper award with honorable mention at EUSIPCO 2019 conference for the paper "A biologically constrained encoding solution for long-term storage of images onto synthetic DNA".

4. Prize of excellence from the Université Côte d'Azur for the year 2019 (Prix d' excellence UCA 2019).

Bibliography

[1] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," Nature, vol. 494, no. 7435, p. 77, 2013.

[2] T. Buschmann and L. V. Bystrykh, "Levenshtein error-correcting barcodes for multiplexed dna sequencing," BMC bioinformatics, vol. 14, no. 1, p. 272, 2013.

[3] T. Coughlin, "175 zettabytes by 2025," Forbes [https://www.forbes. com/sites/tomcoughlin/2018/11/27/175-zettabytes-by-2025/#3ad581754597], 2018.

[4] M. McNerney, "The data center dilemma: Is our data destroying the environment?" The DataCenter Knowledge, 2019.

[5] "Is your data hot, warm, or cold?" Systems Integration Asia https://www.systemsintegrationasia.com/is-your-data-hot-warm-or-cold/, 2019.

[6] R. Miller, "Facebook builds exabyte data centers for cold storage," Retrieved June, vol. 8, p. 2014, 2013.

[7] Y. Lei, M. Sakakura, L. Wang, Y. Yu, R. Drevinskas, and 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.

[8] Y. Shimotsuma, K. Miura, and H. Kazuyuki, "Nanomodification of glass using fs laser," International Journal of Applied Glass Science, vol. 4, no. 3, pp. 182–191, 2013.

[9] E. Chargaff, R. Lipshitz, and C. Green, "Composition of the desoxypentose nucleic acids of four genera of sea-urchin," Journal of Biological Chemistry, vol. 195, no. 1, pp. 155–160, 1952.

[10] E. M. Prager, A. C. Wilson, J. M. Lowenstein, and V. M. Sarich, "Mammoth albumin," Science, vol. 209, no. 4453, pp. 287–289, 1980.

[11] A. Extance, "How DNA could store all the world's data," Nature, vol. 537, no. 7618, 2016.

[12] V. Joguin, "Passive digital preservation now & later," 2019.

[13] A. Chatzieleftheriou, I. Stefanovici, D. Narayanan, B. Thomsen, and A. Rowstron, "Could cloud storage be disrupted in the next decade?" in 12th {USENIX} Workshop on Hot Topics in Storage and File Systems (HotStorage 20), 2020.

[14] A. Ghosh and M. Bansal, "A glossary of dna structures from a to z," Acta Crystallographica Section D: Biological Crystallography, vol. 59, no. 4, pp. 620–626, 2003.

[15] S. Palluk, D. H. Arlow, T. De Rond, S. Barthel, J. S. Kang, R. Bector, H. M. Baghdassarian, A. N. Truong, P. W. Kim, A. K. Singh et al., "De novo dna synthesis using polymerase-nucleotide conjugates," Nature biotechnology, vol. 36, no. 7, p. 645, 2018.

[16] R. A. Hughes and A. D. Ellington, "Synthetic DNA synthesis and assembly: putting the synthetic in synthetic biology," Cold Spring Harbor perspectives in biology, vol. 9, no. 1, p. a023812, 2017.

[17] I. Inc., "Illumina sequencing technology: highest data accuracy, simple workflow, and a broad range of applications." 2010.

[18] D. Bentley, S. Balasubramanian, H. Swerdlow, G. Smith, J. Milton, C. Brown, K. Hall, D. Evers, C. Barnes, H. Bignell et al., "S. vandevondele, y," Verhovsky, SM Virk, S. Wakelin, GC Walcott, J. Wang, GJ Worsley, J. Yan, L. Yau, M. Zuerlein, J. Rogers, JC Mullikin, ME Hurles, NJ McCooke, JS West, FL Oaks, PL Lundberg, D. Klenerman, R. Durbin, AJ Smith, Accurate whole human genome sequencing using reversible terminator chemistry. Nature, vol. 456, pp. 53–59, 2008.

[19] M. Eisenstein, "An ace in the hole for dna sequencing," 2017.

[20] H. Lu, F. Giordano, and Z. Ning, "Oxford nanopore minion sequencing and genome assembly," Genomics, proteomics & bioinformatics, vol. 14, no. 5, pp. 265–279, 2016.

[21] M. Jain, I. T. Fiddes, K. H. Miga, H. E. Olsen, B. Paten, and M. Akeson, "Improved data analysis for the minion nanopore sequencer," Nature methods, vol. 12, no. 4, pp. 351–356, 2015.

[22] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," Science, p. 1226355, 2012.

[23] T. J. Treangen and S. L. Salzberg, "Repetitive DNA and next-generation sequencing: computational challenges and solutions," Nature Reviews Genetics, vol. 13, no. 1, p. 36, 2012.

[24] M. Neiman, "On the molecular memory systems and the directed mutations," Radiotekhnika, vol. 6, pp. 1–8, 1965.

[25] N. Wiener, "Machines smarter than men? interview with dr. norbert wiener. noted scientist," US News & World Report, pp. 84–86, 1964.

[26] G. M. Skinner, K. Visscher, and M. Mansuripur, "Biocompatible writing of data into DNA," Journal of Bionanoscience, vol. 1, no. 1, pp. 17–21, 2007.

[27] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," Angewandte Chemie International Edition, vol. 54, no. 8, pp. 2552–2555, 2015.

[28] S. H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," Scientific reports, vol. 5, p. 14138, 2015.

[29] S. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free dna-based data storage," Scientific reports, vol. 7, no. 1, pp. 1–6, 2017.

[30] C. Pan, S. Yazdi, S. K. Tabatabaei, A. G. Hernandez, C. Schroeder, and O. Milenkovic, "Image processing in dna," arXiv preprint arXiv:1910.10095, 2019.

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

[32] Y. Erlich and D. Zielinski, "Capacity-approaching DNA storage," bioRxiv, p. 074237, 2016.

[33] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "A DNA-based archival storage system," ACM SIGOPS Operating Systems Review, vol. 50, no. 2, pp. 637–649, 2016.

[34] 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," bioRxiv, p. 114553, 2017.

[35] C. Rashtchian, K. Makarychev, M. Racz, S. Ang, D. Jevdjic, S. Yekhanin, L. Ceze, and K. Strauss, "Clustering billions of reads for DNA data storage," in Advances in Neural Information Processing Systems, 2017, pp. 3362–3373.

[36] C. N. Takahashi, B. H. Nguyen, K. Strauss, and L. Ceze, "Demonstration of end-to-end automation of dna data storage," Scientific reports, vol. 9, no. 1, pp. 1–5, 2019.

[37] R. Appuswamy, K. Le Brigand, P. Barbry, M. Antonini, O. Madderson, P. Freemont, J. McDonald, and T. Heinis, "Oligoarchive: Using dna in the dbms storage hierarchy." in CIDR, 2019.

[38] Y. Erlich and D. Zielinski, "DNA fountain enables a robust and efficient storage architecture," Science, vol. 355, no. 6328, pp. 950–954, 2017.

[39] J. B. De Marca, N. Jayant et al., "An algorithm for assigning binary indices to the codevectors of a multi-dimensional quantizer," in 1987 IEEE International Conference on Communications (ICC'87). , 1987, pp. 1128–1132.

[40] J. Zeng, H. Cai, H. Peng, H. Wang, Y. Zhang, and T. Akutsu, "Causalcall: Nanopore basecalling using a temporal convolutional network," Frontiers in Genetics, vol. 10, p. 1332, 2020.

[41] M. Dimopoulou, E. Gil, and M. Antonini, "A quaternary code mapping resistant to the sequencing noise for DNA image coding," in MMSP 2020, Tampere, Finland, Sep. 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02959884

[42] J. Huang and P. Schultheiss, "Block quantization of correlated gaussian random variables," IEEE Transactions on Communications Systems, vol. 11, no. 3, pp. 289–296, 1963.

[43] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers (speech coding)," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 36, no. 9, pp. 1445–1453, 1988.

[44] K. Ramchandran and M. Vetterli, "Line based, reduced memory, wavelet image compression," IEEE Transactions on Image Processing, vol. 2, no. 2, pp. 160–175, 1993.

[45] A. Gersho and R. M. Gray, Vector quantization and signal compression. Springer Science & Business Media, 2012, vol. 159.

[46] B. Usevitch, "Optimal bit allocation for biorthogonal wavelet coding," in Proceedings of Data Compression Conference-DCC'96. IEEE, 1996, pp. 387–395.

[47] C. Parisot, M. Antonini, and M. Barlaud, "3d scan-based wavelet transform and quality control for video coding," EURASIP Journal on Advances in Signal Processing, vol. 2003, no. 1, p. 670972, 2003.

[48] T. André, M. Cagnazzo, M. Antonini, and M. Barlaud, "A scalable video coder with scan-based lifted mcwt and model-based bit-rate allocation," IEEE Transactions on Image Processing, 2004.

[49] P. Rault and C. M. Guillemot, "Lattice vector quantization with reduced or without look-up table," in Visual Communications and Image Processing'98, vol. 3309. International Society for Optics and Photonics, 1998, pp. 851–862.

[50] P. Onno and C. Guillemot, "Data-rate constrained lattice vector quantization: a new quantizing algorithm in a rate-distortion sense," in Proceedings., International Conference on Image Processing, vol. 1. IEEE, 1995, pp. 117–120.

[51] P. Raffy, M. Antonini, and M. Barlaud, "Distortion-rate models for entropy-coded lattice vector quantization," IEEE Transactions on Image Processing, vol. 9, no. 12, pp. 2006–2017, 2000.

[52] J. Vaisey, M. Barlaud, and M. Antonini, "Multispectral image coding using lattice vq and the wavelet transform," in Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No. 98CB36269), vol. 2. IEEE, 1998, pp. 307–311.

[53] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," IEEE Transactions on communications, vol. 28, no. 1, pp. 84–95, 1980.

[54] M. Dimopoulou and M. Antonini, "Efficient storage of images onto dna using vector quantization," in 2020 Data Compression Conference (DCC), 2020, pp. 363–363.

[55] M. Dimopoulou and M. Antonini, "Image storage in DNA using Vector Quantization," in EUSIPCO 2020, Amsterdam, Netherlands, Jan. 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02959330

[56] P. Loyer, J.-M. Moureaux, and M. Antonini, "Lattice codebook enumeration for generalized gaussian source," IEEE Transactions on Information Theory, vol. 49, no. 2, pp. 521–527, 2003.

[57] M. Barlaud, P. Solé, T. Gaidon, M. Antonini, and P. Mathieu, "Pyramidal lattice vector quantization for multiscale image coding," IEEE Transactions on Image Processing, vol. 3, no. 4, pp. 367–381, 1994.

[58] R. J. Roberts and K. Murray, "Restriction endonuclease," CRC critical reviews in biochemistry, vol. 4, no. 2, pp. 123–164, 1976.

[59] R. W. Hamming, "Error detecting and error correcting codes," The Bell system technical journal, vol. 29, no. 2, pp. 147–160, 1950.

[60] A. Krishnan, M. Sweeney, J. Vasic, D. W. Galbraith, and B. Vasic, "Barcodes for dna sequencing with guaranteed error correction capability," Electronics letters, vol. 47, no. 4, pp. 236–237, 2011.

[61] L. V. Bystrykh, "Generalized dna barcode design based on hamming codes," PloS one, vol. 7, no. 5, p. e36852, 2012.

[62] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in Soviet physics doklady, vol. 10, no. 8, 1966, pp. 707–710.

[63] D. Ashlock and S. K. Houghten, "Dna error correcting codes: No crossover." in 2009 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology. IEEE, 2009, pp. 38–45.

[64] S. K. Houghten, D. Ashlock, and J. Lenarz, "Construction of optimal edit metric codes," in 2006 IEEE Information Theory Workshop-ITW'06 Chengdu. IEEE, 2006, pp. 259–263.

[65] M. Dimopoulou, E. Gil San Antonio, M. Antonini, P. Barbry, and R. Appuswamy, "On the reduction of the cost for encoding/decoding digital images stored on synthetic DNA," in GRETSI 2019, ser. GRETSI 2019, Lille, France, Aug. 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02400380

[66] "Dna-based media storage: State-of-the-art, challenges, use cases and requirements version 2.0," JPEG Ad Hoc group, ISO/IEC JTC 1/SC29/WG1M89031, 2020.

[67] R. Appuswamy and V. Joguin, "Universal layout emulation for long-term database archival," in Submitted on ArXiV, 8 September 2020, 09 2020. [Online]. Available: http://www.eurecom.fr/publication/6335