Alma Mater Studiorum · Università di Bologna

SCUOLA DI INGEGNERIA E ARCHITETTURA
Corso di Laurea Magistrale in Ingegneria Informatica

SYNTHETIC DNA AS A NOVEL DATA STORAGE SOLUTION FOR DIGITAL IMAGES

Relatore:
Prof.ssa
SERENA MORIGI

Presentata da:
MATTIA PIRETTI

Correlatori:
Prof.
MARC ANTONINI
Dot.ssa
EVA GIL SAN ANTONIO

Anno Accademico 2019/2020

Contents

Abstract **3**

Introduction **4**

1 State of the Art **7**

1.1 Works on DNA data storage 7

1.2 Related works of image inpainting 9

2 Workflow for DNA data storage **12**

2.1 Compression 12

2.1.1 Computation of Damage Detection parameters 14

2.2 Encoding and Formatting 15

2.2.1 Preventing sequencing noise during Encoding 17

2.2.2 Barcoding process 18

2.2.3 Silhouette clustering for Double Representation 23

2.3 Biological procedures 25

2.3.1 Synthesis, Amplification and Sequencing 27

2.3.2 Types and effects of sequencing errors 28

2.3.3 Percentages and distributions of sequencing errors 29

2.4 Consensus establishment 31
2.4.1 Barcode correction 32
2.4.2 Outlier detection 33
2.5 Deformatting and Decoding 35
2.6 Post-processing 35
2.6.1 Automatic Damage Detection 37
2.6.2 Wavelet inpainting 41

3 Experimental results **45**
3.1 Analysis of the expected damage 47
3.2 Experiments with realistic Nanopore noise 50
3.3 Experiments with worst-case noise 53
3.4 Numerical experiment 57

4 Conclusions **60**

Thanks **62**

Bibliography **63**

Abstract

During the digital revolution there has been an explosion in the amount of data produced by humanity. The capacity of conventional storage devices has been struggling to keep up with this aggressive growth, highlighting the need for new means to store digital information, especially cold data. In this dissertations we will build upon the work done by the I3S MediaCoding team on utilizing DNA as a novel storage medium, thanks to its incredibly high information density and effectively infinite shelf life. We will expand on their previous works and adapt them to operate on the Nanopore MinION sequencer, by increasing the noise resistance during the decoding process, and adding a post-processing step to repair the damage caused by the sequencing noise.

Keywords: DNA, DNA data storage, long-term storage, Biologically constrained encoding solution, Image coding, Biological information theory, Robust encoding, Outlier detection, Damage detection, Image inpainting, Blind inpainting, Inpainting in the wavelet domain

Introduction

The amount of data produced and consumed by humanity has been steadily increasing for years, with an estimated 90% of all data available on the internet having been produced over the last two years. Most of this data is due to an explosion in multimedia content. However, much of this new content is also very rarely accessed, with close to 80% being effectively "cold" data. Nonetheless, this content must be stored and maintained for security and legal compliance. The sheer volume of data produced forces large companies to invest into entire data centers to house the conventional storage mediums, incurring in exorbitant costs to build all the security, power supply and environmental infrastructure surrounding one such endeavor. Additionally, this investment risks being just the tip of the iceberg. The mean time between failures of traditional storage solutions, like Hard Disk Drives and magnetic tapes, is measured in years, a couple of decades at most. This means that every few years the data must be migrated to a new set of devices to ensure reliability, adding a sizable recurring cost to the whole enterprise.

A novel alternative, that has become of great interest over the past few years [46], is to use DNA as a storage medium. DNA is a complex organic molecule, constituted by four building blocks, commonly referred to as nucleotides (nt): Adenine (A), Thymine (T), Cytosine (C), and Guanine (G). As the support for heredity used by living organisms, DNA possesses several properties that could make it a very effective medium to hold human generated data. Firstly, it can be an extremely dense medium, information wise, with a single gram of DNA being theoretically capable of storing up to 455 Exabytes of data under optimal conditions. This extreme data density is due to both its three-dimensional structure and quaternary encoding base. Secondly, it can retain information

for incredibly long periods of time, persisting for centuries and even millennia in a still readable state.

To use DNA as a storage medium is a very involved process, but it can be schematized as follows. Firstly, the digital data has to be encoded into a quaternary schema using the four DNA symbols A, T, C and G. Then, the DNA string needs to be synthesized. As the synthesis process is error free as long as the sequences produced are less than 300nt in length, the encoded data must first be split into smaller chunks. These are referred to as "oligos" and are formatted in a way that includes Headers to carry the meta-information needed to reconstruct the original encoded sequence, by aligning the oligos in the right order. The oligos are then finally synthesized and stored in a hermetically sealed capsule. This prevents contact with damaging agents and ensures their durability.

When the information needs to be retrieved, the capsule can be opened and the oligos read via specific machines called sequencers. This step is one of the sticking points for the whole process, as the sequencing tends to be a very error prone step. However, the sequencing error can be minimized (but for now not eliminated) by abiding to some biological constraints. These include limiting the total length of the synthesized DNA strands, as well as avoiding excessive repetitions of nucleotides and patterns in the oligos. The encoding algorithm must then be able to respect these restrictions, as is the case for the code previously proposed in [54]. Then, to complete the decoding, the sequenced oligos are amplified through a process called Polymerase Chain Reaction , filtered, and finally used to reconstruct and decode the originally encoded data.

In addition to the noisiness of the sequencing, the high cost of DNA synthesis and sequencing has remained one of the main drawbacks to biological data storage. However, over the last few years the nanopore sequencing technology [47], along with its portability, affordability and speed in data production has been revitalizing the idea of DNA storage. There are, of course, drawbacks. The faster sequencing times come with a much higher error rate when compared to other slower, more expensive, but more reliable sequencers.

To deal with the increased noise, abiding by the biological restrictions of the process is fundamental, but not necessarily sufficient. A host of works (see for example [50] and [56]) have focused on tackling this higher error rate by relying on error correction techniques

that increase the redundancy of the encoding. This however incurs into an increase of the encoding cost. The approach developed during the internship with the I3S MediaCoding [64] team marries a biologically constrained encoding process with noise resilience and post-processing techniques.

This work was originally developed during a year-long Erasmus mobility period, hosted by the Sophia Antipolis Polytechnique. The work was conducted as a six months internship at the CNRS's I3S Labs in Sophia Antipolis, as part of the MediaCoding team, and as continuation of a previous month-long project exploring the barcoding process. In this dissertation we will cover the work done during these six months.

We will provide the reader with a background on the current state of the art in chapter 1, regarding the current works on DNA data storage as well as most common inpainting techniques utilized for occlusion removal. In chapter 2 We will then proceed to outline a comprehensive workflow that allows for reliable encoding, storage, decoding, and repair of digital images onto synthetic DNA. This workflow, built upon the previous work done by the I3S MediaCoding team, will be geared toward operating on the Nanopore MinION[55], a much more portable and cheap sequencer than the classical Illumina[42] machine. During the exposition of this workflow we will highlight and discuss the novel additions brought by us during the internship, along with the effects of these contributions and the circumstances that necessitated them. For reference, these contributions will be covered in subsection 2.1.1, subsection 2.2.3, section 2.4 and section 2.6.

Lastly, in chapter 3 we will show the results of the encoding, decoding and post-processing workflow on two different images at different levels of sequencing noise. After that, in chapter 4, we will offer our view of the proposed work and possible future improvements and extensions.

Chapter 1

State of the Art

In this chapter we will provide a brief overview of the current literature. We will start by quickly analyzing the evolution of DNA data storage over the years in section 1.1, from its first implementations in the early 2010s [29] to the previous works done by the I3S MediaCoding team, that we will be expanding upon. This is done to provide the reader with a background into the current state of the art of DNA encoding, as well as the sticking points of common approaches. In section 1.2 we will then perform a more expansive analysis of the literature regarding image inpainting. We will focus mostly on the pros and cons of different inpainting approaches, rather than their evolution over time, as we want to provide a background on the various families of inpainting algorithms. We will refer back to this analysis when discussing how to repair the damage caused by the sequencing noise and our choice of algorithm for the Wavelet Inpaiting in section 2.6.

1.1 Works on DNA data storage

There have been several proposed implementations to encode data, especially media files, onto DNA. The first attempt was done in [29], which also provided an analysis of the main cause for biological errors. Later studies focused on guaranteeing noise resistance, as the error rate of the process became a sticking point for all works concerning the practicalities of DNA data storage. In [34] and [44] the problem was approached by

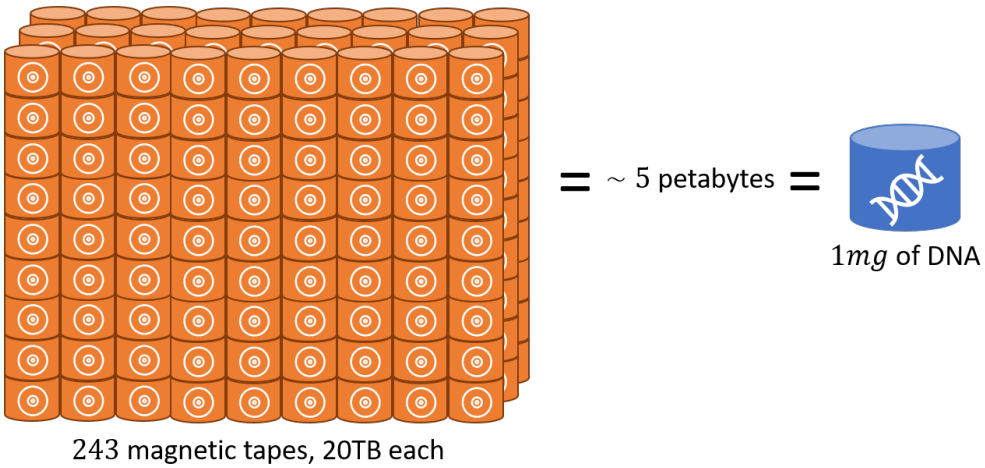


Figure 1: Storage capacity of modern magnetic tapes compared to DNA[29].

splitting the encoded data into overlapping segments, so that each fragment is represented multiple times. This technique, however, is not scalable and adds excessive redundancy. Other approaches include the use of Reed-Solomon code to treat erroneous sequences and guarantee the recovery of missing oligos [41], and performing forward error correction by utilizing more than one dictionary during encoding, so as to encode each symbol with more than one nucleotide sequence [45]. Lastly, [58] proposed a method that integrates Huffman coding in the encoding, in order to store quantized images onto DNA. This work also introduced the idea of utilizing post-processing techniques to correct the discoloration in the reconstructed image.

These approaches, however, are all based on introducing redundancy, which increase the global cost of the synthesis, rather than consider the input's characteristics. Other works have focused on introducing compression to DNA coding which, coupled with avoiding unnecessary redundancy, can help minimize the synthesis cost. The team's previous works [51],[54] focused on one such approach. Additionally, in [57], was proposed a novel DNA encoding algorithm for quantized images, capable of respecting the biological constraints, while providing a new way to map the Vector Quantization (VQ) indexes to DNA codewords so as to minimize the impact of errors in the decoded image. This adds resistance to noise without having to rely entirely on error correction.

Most of the presented works, including the previous ones from the MediaCoding team,

focused on the Illumina sequencer due to its higher accuracy. The work we are introducing here, instead, is geared toward Nanopore sequencing, in order to speed up the process and reduce its cost. To do so we will extend the work presented in [57] by applying the novel mapping algorithm to the DWT decomposition of an image, and adding an extra post-processing step, based on the inpainting algorithm described in [18], so as to reduce the visual artifacting in the reconstructed image.

1.2 Related works of image inpainting

Image inpainting is an approach aimed at repairing damage (or at removing unwanted objects) from images in a way that is seamless to the human eye. What sets image inpainting apart from other restoration techniques like haze-removal is that there is virtually no information to be gained from inside the damaged area. An inpainting algorithm must operate on information taken either from the undamaged parts of the image or, at most, from the contour between these and the target (damaged) area.

There are many families of inpainting algorithms and, similarly, a lot of different classifications are possible. A first macro division can be seen between structural inpainting techniques, textural inpainting methods and hybrid approaches [40]. The first type of approach focuses on the structure of the image, generally the isophote lines and the edges of shapes in the undamaged part of the image. These are then propagated into the damaged area via mathematical means. Texture-based approaches instead focus more on the replication, propagation, or synthesis of textures in the image. Hybrid approaches take aspects of both inpainting families.

A second, more detailed, classification could divide these macro families based on their algorithmic approaches. We would then have Partial Derivative Equations (PDE) based approaches, semi-automatic inpainting algorithms, texture synthesis methods, model/template based methods and a catchall family of hybrid algorithms that exhibit specific characteristics [38],[30]. We will focus more on a couple of these approaches, namely PDE, texture synthesis and exemplar-based approaches, and cover the others more briefly. The goal here is to highlight what are the strengths and weaknesses of

each approach, rather than present a comprehensive comparative review of the different inpainting techniques.

PDE approaches rely on purely mathematical inpainting algorithms, inspired by the partial differential equations that describe the flow of heat in the physical world. The idea can be traced back to [11], where it was proposed a method through which the information is propagated from the intact area into the damaged one, by following the isophote lines crossing the boundary between the two. Already from this first approach one can see the strength and weaknesses of PDE methods, as they have remained pretty much the same since then. While these techniques are very good and efficient at covering narrow damages, they cannot effectively reconstruct textures (see [25]). As such the blurring effect they create becomes very noticeable when trying to cover up large damages. Several other works have improved over PDE methods, with the introduction of second order derivatives in [12] and high order PDE in [21], but the limitations of these approaches remain the same.

Semi-automatic inpainting approaches require, as the name suggests, user interaction for the process to complete. For example in [20] the user is prompted to sketch the contour of the objects in the occluded area, before a texture synthesis algorithm applies the inpainting. This is in addition to the detection of the damaged area, that in virtually all the approaches mentioned in this chapter is assumed to be done outside of the algorithm itself, presumably by the user. As already stated, we require a way to automate the entire post-processing, and as such these methods are not what we are looking for.

Texture synthesis algorithms and exemplar-based approaches are often considered separately, but for our purposes they fall under the same umbrella. Both focus on the use of pixel color and texture to repair damaged areas without causing blur, and both have similar pros and cons. Texture synthesis approaches take a small sample of true texture and try to replicate it. This can be done pixel by pixels, starting from the edges of the occluded area [10], or more efficiently by considering whole blocks of pixels at a time [13]. Even with this optimization, the texture generation approach can be computationally expensive and still struggle with more complex textures. Exemplar-based methods side-step the texture generation problem by sampling and copying existing color

values from a specified source. This is commonly done from the undamaged parts of the image [15], but other solutions rely on entire databases of images similar to the damaged one [23]. Nonetheless, all these methods tend to present similar behaviors: they perform well when covering large, regular occlusions, but struggle with repairing complex or irregular textures. Additionally, they risk causing artifacting due to the order in which the textures are copied from the source to the damaged area. In this vein, Criminisi's algorithm [18] merits a special mention, as it is an exemplar-based approach that tries to account for and prevent this drawback.

The hybrid approaches are generally intended as methods that rely on both structure and texture information, usually gained by a decomposition of the image. From this point of view [18] could be classified as hybrid, due to the way its priority-based filling order is computed. Most of the works in this category try to split the image into a structure and a texture component [14],[16] and inpaint them separately before merging the results. Two sub-families merit mention here. The first are wavelet-based approaches. While they are an important background, they relate less to our problem than one might hope, as we will show in subsection 2.3.2. These inpainting algorithms [28],[32] rely on Wavelet Transform as a decomposition tool, to split the image into structure and texture information. While the approach is interesting, it does not match our situation. The second sub-family of hybrid algorithms is that of machine-learning based approaches. It is debatable whether this should be considered a family in and of itself, or whether the work of the convolution layers matches the definition of "decomposing" the image into structure and texture information. In any case, this is an approach that is becoming more prominent lately, relying on both convolution [48],[53] and deep [31] neural networks. These methods can be very good at handling blind-inpainting, where the mask of the damage is not given a priori. This would be an excellent match for our needs, and the approaches certainly merit further investigation in subsequent works. However, all machine learning approaches require a large amount of data that we did not have easy access to, as the pre-trained layers commonly used for these approaches were prepared over much different images and visual distortions than the ones we had to contend with.

Chapter 2

Workflow for DNA data storage

In this chapter we will first cover all the steps of the encoding and decoding process to familiarize the reader with the structure of the work. Additionally, we will investigate in more detail and provide some background for processes that appear often throughout the workflows, or that are especially relevant to the project as a whole.

Several steps are required to encode an image onto DNA, and to subsequently decode it correctly. The main steps are five, and can be seen exemplified in Figure 2. The innovations brought by this work add three additional main steps and can be seen exemplified in Figure 3. In the resulting workflow in addition to the five that are from the original implementation, an outlier detection step is added to the consensus establishment phase, and two new steps comprise a post-processing phase after the decoding is complete. We will now cover each step in detail.

2.1 Compression

Firstly, a Discrete Wavelet Transform (DWT) is performed [9]. This takes advantage of the spatial redundancies in the image to decompose it into subbands, by passing through a series of low and high-pass filters. These are sub-images with width and height equal to 2^{-N} times those of the original image, where N is referred to as the "level" of the

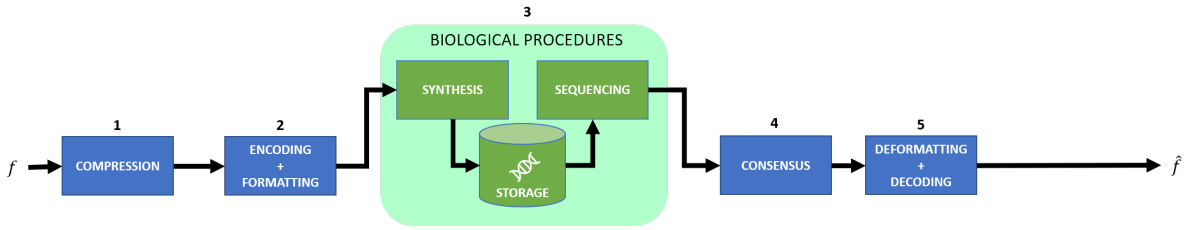


Figure 2: Previous workflow, presented and utilized in both [51] and [54].

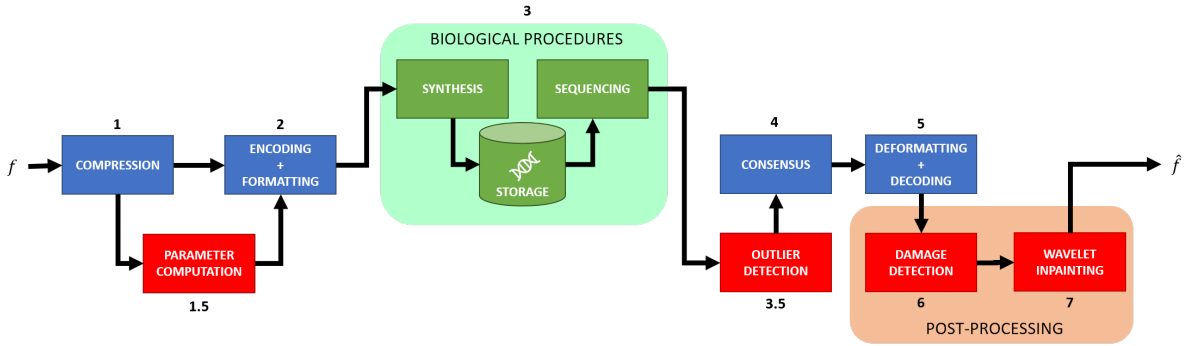


Figure 3: Novel proposed workflow. Additional steps shown in red, and presented in subsection 2.1.1, subsection 2.4.2, and section 2.6.

DWT. The HH, HL and LH subbands carry local variations in pixel color, meaning they generally represent the details of the image, while the bulk of the information is carried by the LL subband. An example of DWT applied to the image we will be utilizing during the rest of this work can be seen in Figure 4. From here until the end of the decoding, we will be working exclusively on this decomposition, rather than on the entire image.

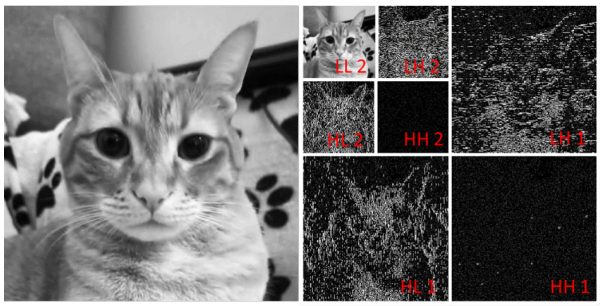


Figure 4: 2 level Discrete Wavelet Transform. Original image on the left, greyscale of the wavelet subbands on the right.

After the decomposition, each resulting subband is quantized independently from the others with a Vector Quantization (VQ) algorithm [4]. To further ensure optimal compression, a bit allocation [7] algorithm is employed to provide the best quantization codebooks for each subband, in order to maximize the encoded bits per nucleotide. These two processes (DWT and

VQ) comprise the compression step.

2.1.1 Computation of Damage Detection parameters

After the compression step we introduce the first contribution of this work that expands the workflows already built in [51] and [54].

The damage detection step, that will be described in details in subsection 2.6.1, depends on a series of parameters. The most obvious of these are the two thresholds that determine whether a pixel is considered anomalous or not, but the definition of "neighborhood" for a given pixel is also going to influence the behavior of the algorithm. There are essentially two ways of determining these values: either the user can input them by hand, or an algorithm can be used to try and determine them. As automation of the damage detection was one of our main goals, we opted for the second option.

Normally, this would mean analyzing the noisy subbands obtained after the DNA decoding. The obvious drawback would be that deriving the information from a corrupted image might give erroneous results, causing us to set the thresholds to values that would cause too many false positives or, more likely, too many false negatives. However, we are in the unique position of having access to the ground truth, the undamaged image as it was encoded before synthesis and sequencing, because we control the entire workflow.

We decided to exploit this by inserting a very small additional step after the compression. Here we define the neighborhoods for each subband and then analyze the image in the same way we would during the post-processing. At the end of this analysis we will have compiled a list of the values (difference from pixel's neighborhood and variance within neighborhood) that would be computed during steps one and two of the damage detection process. We can then compute what are the acceptable thresholds from these lists, either by setting them to the highest value or to a lower one and accepting a certain number of false positives.

This parameter computation can be customized to the single subband. For example, the easiest way to do this is to accept more false positives in larger and less meaningful subbands, were loss of details is less problematic than distracting visual distortion.

Another way would be to modulate the shape and size of the neighborhoods based on those of the quantization vectors used for each specific subbands.

2.2 Encoding and Formatting

After the compression step (and the parameter computation step added by this work), the quantized subbands are to be encoded into quaternary code. To do this we first use the encoding algorithm proposed in [51] to create codebooks. This is done by merging predefined symbols into codewords. The symbols are selected from two dictionaries:

$$D1 = AT, AC, AG, TA, TC, TG, CA, CT, GA, GT$$
$$D2 = A, T, C, G$$

Concretely, this means that we construct sequences of even length L by concatenating $\frac{L}{2}$ symbols from the first dictionary $D1$, and sequences of odd length by adding a single symbol from the second dictionary $D2$ at the end. So, for example, we can construct all the codewords of six nucleotides by computing all the permutations of three elements from $D1$ ($AT-AC-AG,\ AT-AC-TA,\ etc.$), and the ones of seven nucleotides by adding one element from $D2$ at the end ($AT-AC-AG-A,\ AT-AC-AG-T,\ etc.$).

The symbols that comprise these two dictionaries have been specifically chosen so that their concatenation always leads to "viable" words, meaning they abide by the restrictions indicated in subsection 2.2.1. Since the resulting codewords will be permutations of elements from the two dictionaries, this helps making the encoding more robust to the sequencing noise, without adding redundancy or overhead, as the words that would be more prone to misdecoding are discarded. Additionally, this algorithm is capable of encoding any type of input, rather than just binary.

After all the codewords have been constructed, the mapping binding these to the numerical values in the subbands is performed. This is done by the method originally presented in [5], that was expanded upon in [57]. The fundamental idea is to map the input vectors obtained from the Vector Quantization algorithm to the codewords generated by the coding algorithm. However, we also want to execute the binding in a

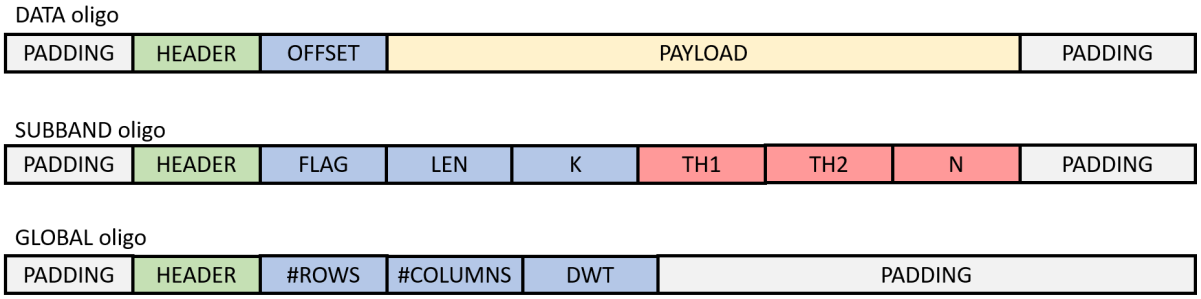


Figure 5: Formatting schema for the three oligo types. Oligo identifiers in green, metadata used during decoding in blue, metadata used during post-processing in red, and encoded information in yellow.

way that can minimize the impact of errors on the quaternary sequence. To do this, we map vectors that are close to each other, in terms of Euclidean distance, to codewords that are similarly close to one another, here in the context of Hamming distances. This means that, in case of a small distortion due to sequencing errors, erroneous words will still have a small Hamming distance from the correct ones, reducing the visual distortion that the errors cause in the reconstructed image. Again, this adds robustness against the sequencing noise without having to introduce costly redundancy in the encoding process.

As we will mention in section 2.3, the DNA synthesis process is effectively error free for short strands, in our case shorter than 250-300nt [65]. As such, the last process we need to undergo in this step is cutting into chunks of the appropriate size the encoded subbands that we just generated with the mapping and encoding algorithm. This also implies the necessity to add to each of these oligos some form of metadata, to indicate the position of the specific oligo in the input image. Not only that, we also need to similarly encode metadata related not just to the single oligo, but to the encoding process itself, as this information will be needed during the decoding. The end result is akin to the classical approach used for data packets that are to be transmitted over a network: a division of a large amount of information into smaller packets, each carrying one or more Header fields and replicated several times for robustness. The specific formatting schema we used can be seen in Figure 5.

As was the case in [51] and [54] we opted for a formatting schema revolving around three distinct types of oligos. The Global (G) oligo carries information concerning the

entire image, specifically its size and the levels of DWT used during encoding. A single Subband (S) oligo for each subband encodes the information concerning its encoding. In this case the VQ parameters: type (FLAG) and size (LEN) of the vectors used as well as the length of the codewords (K); and the information needed by the post-processing workflow: the threshold values (TH1 and TH2) and the size of the neighborhood (N) for the damage detection algorithm. Data (D) type oligos are carrying the actual information (PAYLOAD) as well as a field to identify the position of their Payload in the original data (OFFSET). Finally, all three oligo types have an identifying field (HEADER) that discriminates their type, and eventually the subband and level of DWT they belong to, to allow for their correct decoding. All oligo types also have some padding at the beginning and end as a concession to the distribution of the sequencing noise. This facet will be covered in more details in subsection 2.3.3 and subsection 2.3.2, but the necessity here is to try and preserve the Headers and metadata fields as much as possible, by protecting them with disposable DNA. Additionally, we protect the most important fields (especially the Header and Offsets fields) by relying on barcodes. These will be explored in more details in subsection 2.2.2, but they can be considered a type of self-repairing inline encoding. They are codewords specially chosen so that even in the presence of errors (up to a certain threshold) a correction algorithm is able to recover the original sequences from the noisy ones, without having to introduce any additional information other than the quaternary string itself.

2.2.1 Preventing sequencing noise during Encoding

We will cover the sequencing in more details in subsection 2.3.1, but for now we simply have to acknowledge that it is an extremely noisy process. While this noisiness cannot be completely avoided, it can be reduced if some biological restrictions are observed during the encoding process. Specifically, we must ensure that our DNA strands have the following features:

- No homopolymers over a certain length. A homopolymer is a repetition of the same nucleotide. The specific length varies between sequencer but can be assumed to be between a maximum of 3 and 6 nucleotides, meaning that ideally the same

nucleotide should not appear more than three times in a row in the final oligos.

- The percentage of CG content should be lower or at most equal to that of AT content.
- There should be no pattern repetition in the oligos.

The first two points are relatively easy to ensure, and the Single Representation (SR) encoding algorithm presented in [51] can guarantee that the oligos we create respect these restrictions. The third restriction is more complex to observe and has to be taken into account during the mapping.

To do this we can employ a technique called Double Representation (DR). This is an encoding method in which we build a codebook that is twice as large as the one that would be needed for traditional, SR, encoding. In SR, when a value needs to be encoded it is mapped to a single codeword, in DR it is instead mapped to one of two codewords chosen at random. This significantly reduces the repetitions of patterns in the Payloads of the oligos. This is especially true for the case of VQ-based compression, in which several adjacent indexes could be the same (for example, when encoding the background of an image, or large patches of solid color). In this work we decided to not test this approach and simply rely on SR, but the infrastructure needed to adapt the advanced encoding from [57] to use DR is already in place, including the codebook clustering algorithm described in subsection 2.2.3.

2.2.2 Barcoding process

As we already mention in subsection 2.2.1, it is possible to reduce the noisiness of the sequencing process by abiding to some biological restrictions. This, coupled with the redundancy added by the PCR amplification (covered further on is section 2.3), can be considered acceptable for the preservation of the data itself. It is not, however, sufficient protection for the metadata. Especially considering the findings of [43], we must protect the most important fields of the oligo further. While padding helps, we conducted several tests where crucial information (for example the Offset of the D oligos)

was simply encoded as we would any other data, and the resulting loss of oligos was unacceptable. As such we referred back to barcoding [33], a technique that the team had already relied on in previous works.

The use of barcodes to protect sensitive parts of the DNA encoding is not new and has been explored in several other studies over the years, see [24], [33], and [52] for some examples. The idea is to create a set of codewords with a high enough distance between them, so that in the case of an error we can correctly identify which codeword was the original one. In our case the codewords were constructed via the same algorithm we use for the encoding, guaranteeing that the barcodes themselves respect the biological constraints of the sequencing process.

For constructing the barcode sets from these codebooks We are using a greedy heuristic algorithm based on the work in [33] and shown in algorithm 1. Due to its greedy nature, this algorithm does not guarantee finding the maximal barcode set for a given codebook. It is much more efficient than an exhaustive search over the same set, as the generation of barcode sets is a very computationally expensive process. Given a codebook $C^* = c_1, c_2, \ldots, c_N$ and the expected error tolerance μ the set construction procedure is as follows:

- Set $S = 1$, create a first barcode set B_S and add the first codeword c_1 of the codebook C^* to it.
- Foreach codeword c_k with $k = 1, 2, \ldots, K$ do the following:
  - Check the distance between all the codewords in each barcode set B_i with $i = 1, 2, .., S$.
  - If the distance between all codewords in a barcode set B_i is bigger than $d_{min} = 2 \times \mu + 1$, add c_k to the set B_i and move to the next codeword $c_{(k+1)}$.
  - If this condition was not met for any existing barcode set then set $S = S + 1$, create a new barcode set B_S and add the codeword c_k to it.

Another way to see the problem is to consider the computation of the barcode set as the identification of the maximum independent set over a graph, where the vertices are

Algorithm 1: Barcode sets construction.

Input:
$C = (c_1, c_2, \ldots, c_N)$: codebook
μ : expected error tolerance.

Output:
$B = (B_1, B_2, \ldots, B_S)$: barcode sets with error tolerance μ.

$S \leftarrow 1$; $B(S) \leftarrow \{c_1\}$
$d_{min} = 2 \times \mu + 1$

for $i \leftarrow 1$ ***to*** N **do**
    $flag \leftarrow$ **false**
    for $k \leftarrow 1$ ***to*** S **do**
        $d_B \leftarrow$ min distance between c_i and all the codewords $c_B \in B_k$
        if $d_B \geq d_{min}$ **then**
            $B(k) \leftarrow B(k) + c_i$
            $flag \leftarrow$ **true**
            break
        end
    end
    if $!flag$ **then**
        $S \leftarrow S + 1$
        $B(S) \leftarrow \{c_i\}$
    end
end

the words in the codebook C^* and the edges connect the vertices that have distance lower than d_{min}. As this is a traditionally NP-hard problem, a non-greedy algorithm would incur into an unsustainable computational overhead very quickly.

A quick aside is necessary regarding the distance that is used when constructing the barcode sets. Until now we did not go into any specifics, but it is worth it to dig a little bit deeper into this detail. For the entire of the barcoding process, both the assignment and the reconstruction, we consider as metric the Sequence-Levenshtein distance that is presented in [33]. An example of the difference between this and the classical Levenshtein distance [1] can be seen in Figure 6. In short, both distance metrics measure the number of steps needed to go from one string to another. The main difference is that classical Levenshtein counts each added, removed or changed letter (or nucleotide, in our case) as a distance of one, and sums these elemental steps to obtain the minimum total distance.

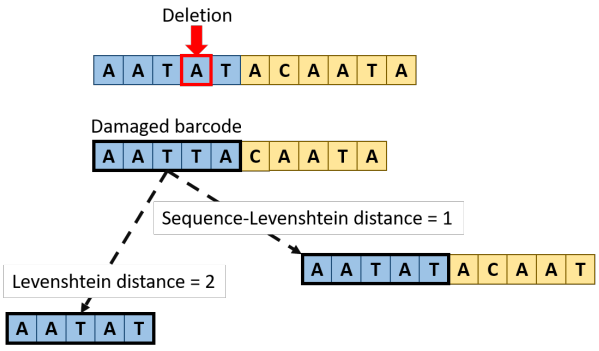


Figure 6: Example of difference between Levenshtein and Sequence-Levenshtein distance.

In the context of DNA however, this is insufficient. Specifically, in the case of deletions, the first nucleotide from the rest of the DNA strand will appear at the end of the barcode. This is the shifting we previously mentioned, that makes Indels so hard to deal with. This means that the now noisy barcode will have a Levenshtein distance of two from the original one (one for the deleted nucleotide and one for the nucleotide that was inserted at the end from the rest of the oligo).

To remedy this, in [33] was introduced a new distance metric called Sequence-Levenshtein, as an extension of the classical Levenshtein distance, in which adding nucleotides at the end of a word does not increase the distance. The effect is twofold. On one hand this allows us to utilize it in the construction of barcode sets that can ensure recovery from deletions, on the other it also means that the classical Levenshtein distance is an upper bound of the Sequence-Levenshtein distance, resulting in closer words and

barcode sets of reduced size. This last detail can become an obstacle and forces a sort of "ballooning" in the codeword length for the barcodes when either a high error tolerance is needed (as it is for the identifying Header fields), or when the data to protect can have many different values (as it happens with the Offset of D oligos, that can reach well into the hundreds).

In Figure 7 we can see an example of the correction process. Let us suppose that a certain information, for example the Offset of the D oligos, has been encoded with the barcode set on the right. If the field containing the information suffers an error, in this case a Deletion, we can recover the information by comparing the resulting (damaged) barcode with the intact ones in the set. We can see that the damaged barcode (AATTA) does not appear in the set. This is because the barcode set was built to be resistant to one error, meaning each word has a distance of at least three from one another. As such, during the decoding we can clearly see that an error has occurred and attempt to repair it. This is done by comparing the erroneous barcode to each of the original ones and picking as correct the one that is closest.

Of course, this does not guarantee that the value will be correctly recovered. If the number of errors is higher than the tolerance of the specific barcode set, the noisy barcode could be unrecoverable (meaning it has a distance higher than the tolerance from every barcode in the set) or, even worse, it could be recovered erroneously to the wrong barcode. In the first case the barcode and whatever information it was carrying is lost, but the assumption is that the added redundancy from the PCR is sufficient to compensate for the loss. The second case can be the source of potentially more impactful, and Byzantine, visual distortions in the reconstructed image. The erroneous correction of the Offsets in the D oligos is a big part of why we introduced an outlier detection step before the consensus,

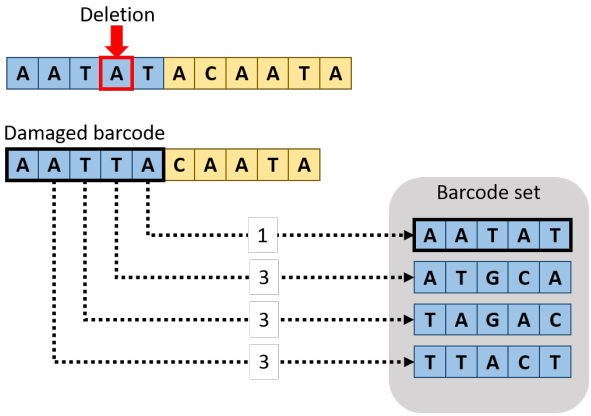


Figure 7: Example of how a barcode can be recovered.

in order to prune away oligos with Byzantine Headers that could poison the consensus for the oligo they were erroneously identified as.

2.2.3 Silhouette clustering for Double Representation

In subsection 2.2.1 we introduced the concept of Double Representation, an evolution of the classical encoding schema utilized in [54] that could reduce the amount of pattern repetitions in the formatted oligos. This method is very interesting, but cannot be directly applied to the advanced encoding presented in [57]. This is because the DR operates by selecting one of two random codewords to map to the numerical values during encoding. This stochastic selection obviously conflicts with the idea behind the DeMarca encoding, that tries to match close numerical values to close codewords to reduce the visual distortion caused by erroneous decoding. To try and reconcile these two encoding techniques, we developed in this work an ad-hoc clustering algorithm that given a codebook can split it into clusters based on the similarity between the codewords. This could allow us to map a value to one of two random codewords, while still guaranteeing that the values close to the number being encoded would be mapped to other codewords close to at least one of the randomly chosen ones.

Clustering as a whole is well treaded problem, that has been at the forefront of data-analysis for years if not decades [3],[49],[67]. The general idea is to classify a set of points, or objects, based on their position relative to each other. In most cases this is applied to unsupervised machine learning models, where the task is to group together unlabeled points to gleam some insight on the underlying structure of the data that is being analyzed. In our case, we have a slightly different problem. We know precisely how many clusters there must be in our data, the problem is finding which points belong to which cluster. This in and of itself is not uncommon, k-means clustering [2] is a very well-known method to separate a data-set into a given number of clusters. The problem with using a traditional k-means method, is that we also have a constraint on the exact number of points that must be in each cluster, as our mapping expects precise cluster sizes. While there is some work investigating versions of k-means clustering that account for underlying constraints of the data-set [68], adapting this approach to our needs was

deemed too costly.

As such we steered ourselves toward building a quick ad-hoc clustering algorithm based on the Silhouette Coefficient (SC), shown as pseudo-code in algorithm 2. Originally proposed in [6], Silhouettes are a metric used to validate and interpret clustering results. The SC (sometimes referred to as Silhouettes Index) is a tweak of this technique presented in [8], and still used in some modern applications [66]. The SC is a measure, varying between 1 and -1, that is calculated for every point, or object, in the data-set. The value of the coefficient indicates how appropriate is the assignment of an object to its current cluster, with 1 being a perfect assignment and -1 being the worst possible one. It is calculated by considering the average distance of each point from every other point in the cluster it is assigned to and comparing it to the average distance from all the other clusters. If the point is closer to the other points in its cluster it will have a positive coefficient. If it is closer to another cluster, the coefficient will be negative. Even from this quick definition, it can be inferred that the SC can be an expensive metric to employ, given that it requires the computation of the average distance with every point in the data-set, for all points. As our clustering needs were for very small sets (rarely above 1000 words) this overhead was considered negligible.

The way our algorithm works is as follows:

- First, compute the adjacency matrix for the entire codeword set. This is done to speed up the SC computation process.
- Assign to the set a random clustering, of course respecting the constraints on number and size of the clusters. This is done to introduce entropy in the data-set and eliminate any pattern that may be present in the original ordering of the codebook.
- Repeat the following steps until the Silhouettes of the clusters stops meaningfully improving, or a maximum number of steps has been reached:
  - Compute the SC for all codewords in the set. Additionally, keep a list P of the preferred clusters for each codeword. This can be done during the SC computation with no additional overhead.

- Select a codeword c_i at random from the ones with $SC \leq 0$. This stochastic approach, rather than choosing for example the codeword with the minimum SC, is done both to avoid tottering issues and hopefully help push past eventual local minimum.
- Select the codeword c_j with the lowest SC in cluster $P(c_i)$. Ideally the coefficient of this second word will be negative, but it is possible that it might be close to or above 0.
- Switch the two codewords.
- Compute the improvement and move to the next iteration.

In this way the total Silhouette Coefficient should generally be increasing, as at each step we are assigning a word to its preferred clusters, while simultaneously removing from that cluster a second word that either would fit better elsewhere, or was on the edge between two clusters. If it does not, it means we have reached a plateau, or we are having tottering between a handful of points. In these cases, the algorithm terminates. Testing of the algorithm showed it to be reasonably performant on the codebooks we employed it on, up to 4000-10000 words. Eventually, it was decided to move forward with the noise simulation and post-processing testing described in chapter 3 without the use of Double Representation, to avoid muddling the results of the experiments, meaning this algorithm does not appear in the final work. But should a future work wish to use both DR and DeMarca encoding, especially in the case of a wet experiment, this might be a good way to join them together.

2.3 Biological procedures

The biological procedures step covers the "wet" parts of the process, that revolve around actual physical DNA. We will be covering them very briefly in subsection 2.3.1, as DNA synthesis and PCR exude from the scope of our work. In subsection 2.3.1 we will cover the differences between two types of sequencers, the Illumina (used by most older works regarding DNA data storage) and the Nanopore (whose behaviour we will be

Algorithm 2: Silhouette Coefficient clustering.

Input:
$C = \{c_1, c_2, \ldots, c_N\}$: codebook to cluster
K : number of clusters
M : cardinality of clusters
Output:
$CL = \{CL_1, CL_2, \ldots, CL_K\}$: clustering of the codebook

$CL \leftarrow$ random clustering $\{CL_1, CL_2, \ldots, CL_K\}$
$SC = \{SC_{c_1}, SC_{c_2}, \ldots, SC_{c_N}\} \leftarrow \{-1, -1, \ldots, -1\}$
$(\ SC, SC_{improv}, P\ ) \leftarrow$ `computeSilhouetteCoefficients(`C,SC,CL`)`

while $SC_{improv} > 0$ **do**
  $c_i \leftarrow$ random $c_i \in C$, with $SC \leq 0$
  $CL_i = \{c_1, c_2, \ldots, c_i, \ldots, c_M\}$: cluster containing c_i
  $CL_j = \{c_1, c_2, \ldots, c_M\}$: preferred cluster $P(c_i)$
  $c_j \leftarrow$ codeword $\in CL_j$ with $\min SC(c_j)$
  $CL_i \leftarrow CL_i - \{c_i\} + \{c_j\}$
  $CL_j \leftarrow CL_j - \{c_j\} + \{c_i\}$
  $(\ SC, SC_{improv}, P\ ) \leftarrow$ `computeSilhouetteCoefficients(`C,SC,CL`)`

Function `computeSilhouetteCoefficients`**:**
  Input:
  $C = (c_1, c_2, \ldots, c_N)$: codebook
  $SC = (SC_{c_1}, SC_{c_2}, \ldots, SC_{c_N})$: Silhouette Coefficient for each codeword in C
  $CL = (CL_1, CL_2, \ldots, CL_K)$: current clustering
  Output:
  $SC' = (SC'_{c_1}, SC'_{c_2}, \ldots, SC'_{c_N})$: new Silhouette Coefficient for each codeword in C
  SC_{improv} : improvement of SC
  $P = (P_{c_1}, P_{c_2}, \ldots, P_{c_N})$: preferred clusters of each codeword in C

  $SC_{improv} \leftarrow 0$; **for** $i \leftarrow 1$ ***to*** $\#C$ **do**
    $c_i = C(i)$
    $CL_i = \{c_1, c_2, \ldots, c_i, \ldots, c_M\}$: cluster containing c_i
    $a(i) \leftarrow \frac{1}{\#CL_i - 1} \sum_{c_j \in CL_i, j \neq i} dist(c_i, c_j)$
    $b(i) \leftarrow \min_{k \neq i} \frac{1}{\#CL_k} \sum_{c'_j \in CL_k} dist(c_i, c'_j)$
    $P(c_i) \leftarrow$ cluster CL_k with min average distance to c_i
    $$SC'(c_i) \leftarrow \begin{cases} 1 - \frac{a_i}{b_i}, & \text{if } a_i < b_i \\ 0, & \text{if } a_i = b_i \\ \frac{b_i}{a_i} - 1, & \text{if } a_i > b_i \end{cases}$$
    $SC_{improv} \leftarrow SC_{improv} + SC'(c_i) - SC(c_i)$

simulating in this work). Lastly in subsection 2.3.2 and subsection 2.3.3 we will cover the incidence and types of sequencing errors that arise when using the Nanopore sequencer.

Once the encoding is complete, the digital oligos are synthesized into biological DNA. As stated before, we can ensure that this process remains error free as long as we only construct synthetic DNA strands that are relatively short, in the 250-300nt range. The synthesized DNA can then be stored in hermetically sealed capsule. This is done to minimize the damage that DNA can incur in and is effectively the last step in the encoding workflow.

The decoding workflow begins with the recovery of these hermetic capsules and the extraction of the encoded DNA. This is then subjected to a process called Polymerase Chain Reaction, or PCR amplification. This is a precursor to the sequencing process, and is used to add redundancy to the oligos, by creating many copies of each one. While the process is not error free, the incidence of this noise is minimal compared to the effect of the sequencing of the oligos. As such, it is a very effective way to add a lot of redundancy to the process, as the amplification can result in several hundreds or even thousands of copies of each oligo being produced, without incurring in the exorbitant cost that such a level of redundancy would imply if it was achieved before the synthesis.

The sequencing, often referred to simply as the "reading" of the oligos, is the process with which the biological DNA strands are scanned and translated into a digital quaternary representation. This is done by machines called sequencers. As stated in the introduction, most of the work on DNA encoding, including the teams own work in [54], has been done on the Illumina sequencer [42]. While the initial large presence in the literature was due to it being among the first of the Next Generation Sequencers (NGS) [26], its staying power is a testament to the accuracy and effectiveness of the machine. The drawbacks of the Illumina sequencer are the natural flip-side of its strengths. The machine is very large, and the sequencing process is cumbersome, slow, and very expensive.

2.3.1 Synthesis, Amplification and Sequencing

The Oxford Nanopore sequencer, especially its MinION version [47], are an up-and-comer of the sequencing scene. It is much cheaper, faster, and more portable than the Illumina and other classical NGS machines. Examples of both these sequencers in action can be seen in Figure 8b and Figure 8a. The main challenge keeping the Nanopore technology from more widespread adoption is the high error rate, especially when compared with the Illumina sequencer.

(a) Illumina sequencer.

(b) Nanopore MinION.

As the focus of this work was partly developing an encoding and decoding workflow that could be viable for use on the Nanopore MinION, understanding the types of errors that the sequencing could incur in was a top priority. Several studies have been conducted in this sense [43],[55].

2.3.2 Types and effects of sequencing errors

There are three types of errors that can be observed during the sequencing: Substitutions, Insertions and Deletions. In Substitutions a nucleotide is read erroneously, for example an Adenine (A) nucleotide is interpreted by the machine as being a Thymine (T). This obviously does not affect the order or values of the other nucleotides in the sequence. Insertions and Deletions, on the other hand, end up affecting all the nucleotides in the sequence that follow the error. The names are quite self-explanatory, a Deletion occurs when the sequencer "skips" a nucleotide during the reading, while an Insertion happens when a nucleotide that was not present in the original strand is erroneously added by the sequencer. Visual examples of all these errors can be found in Figure 9. As the figure illustrates, Substitutions are the least damaging type of errors we can incur in. While the nucleotide they affect is lost, the rest of the information encoded in the DNA strand remains intact. Errors like this are relatively simple to deal with, and both our mapping algorithm and barcoding process are designed to try and minimize the visual distortion

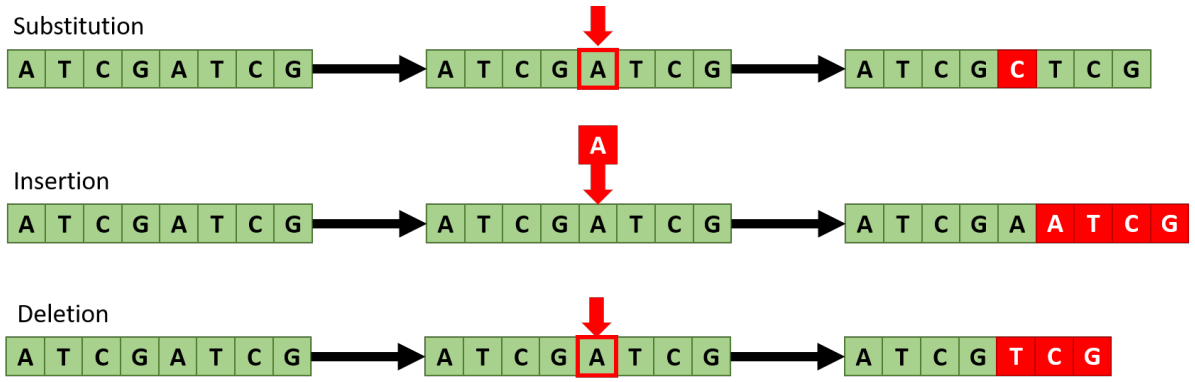


Figure 9: Visual example of Substitution, Insertion and Deletion on the same DNA strand.

they could cause in the final image.

Insertions and Deletions, often grouped together as "Indels", are much more difficult to handle. The effects of the error are not limited to only the misread nucleotide but end up affecting everything that comes afterwards in the strands, as the entire sequence ends up being shifted in either one direction or the other by one nucleotide. A clear example of the problem can be seen in Figure 10.

An Insertion in the Header of the oligo ends up affecting everything after it, including the Payload. While Header and Offset can be corrected via the Barcode Correction process, the Payload ends up being scrambled, with some codewords ending up as completely undecodable. Although it should be noted that in this example, to avoid additional complications, we are assuming a simple sequential mapping like the one originally used in [51],[54], rather than the more complex DeMarca mapping from [57]. While a more advanced encoding could have mitigated the damage showed in this example, the distortion would have not been eliminated entirely, and this mitigation might not always be possible. Very similar effects could be observed in the case of a Deletion.

2.3.3 Percentages and distributions of sequencing errors

Several studies have been compiled trying to determine the error percentage of the Nanopore sequencer, as well as the distribution of the errors. It has been found that while

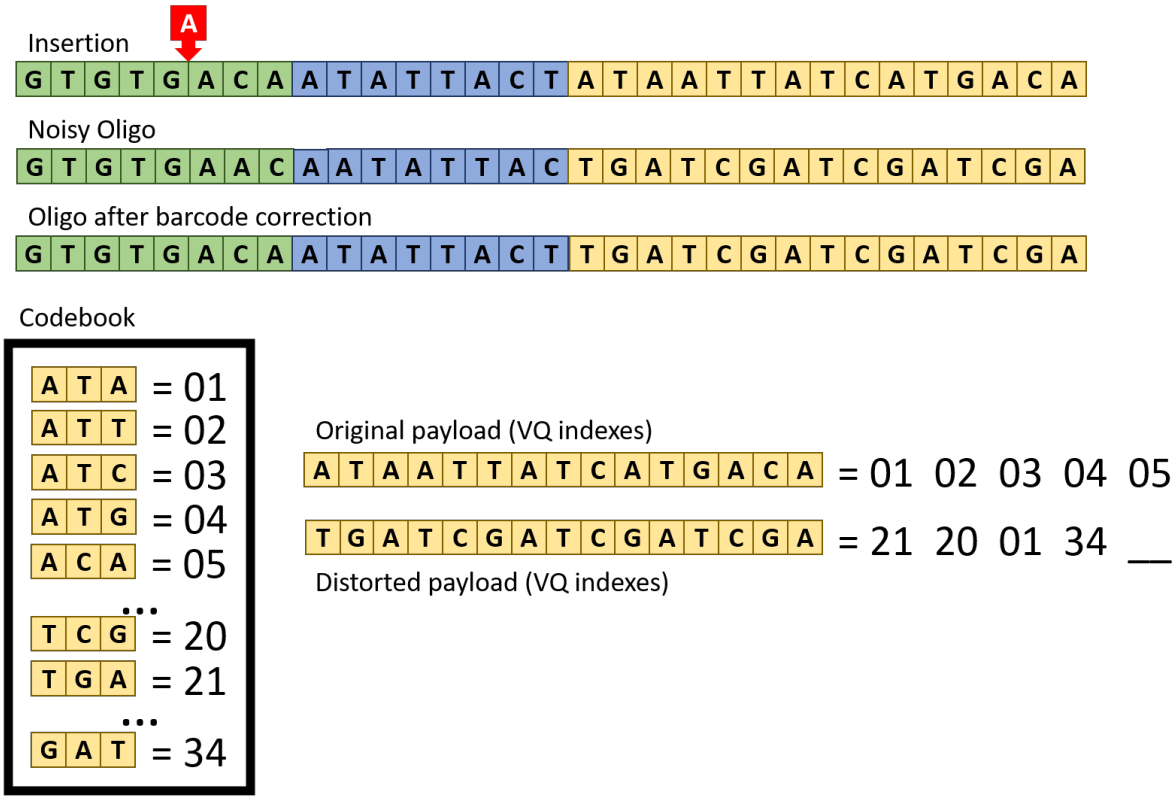


Figure 10: Distortion caused by an Insertion. Headers in green, Offset in blue, Payload in yellow. Codebook constructed with the algorithm from [54], mapping executed with the simple mapping algorithm from [51],[54].

the accuracy of the Nanopore technology was only around 85% when first introduced, it has since risen to 95% and above [59]. The incidence of each error type has similarly been measured [60]. For our work in this report we estimated 2.3% of Deletions, 1.01% of Insertions and 1.5% of substitutions, for a total error rate of 4.81%. Lastly, the distribution of errors has also been analyzed [43], with the finding that the vast majority of errors occur in the extremities of the oligo. In our simulations we estimated that up to 80% of all errors (Substitutions, Insertions and Deletions) occur in the first and last 20nt of the DNA strands. This last statistic is why we decided to add a padding at the start of the oligos. The advantage of having some disposable nucleotides in the areas most likely to be lost is two-fold.

Most obviously, errors that happen during sequencing in the padding do not directly

affect the Headers. While we are capable of repairing a limited number of errors, as we have show in subsection 2.2.2, more than a few errors at a time could overwhelm our barcode correction process. If this happens, and a Header becomes undecodable, we risk losing an entire oligo of data. In the worst case an entire subband could be lost if the Header identifying the S oligo for the subband is damaged beyond repair.

Additionally, while this helps minimizing the incidence of Substitutions and part of the effect of Indels, we can also use the padding to our advantage to try and reduce the damage caused by the shifting in the strand that follows Insertions and Deletions. Since we know the size of the original oligo, we can compare it to the size of the sequenced oligo before the barcode correction process shown in subsection 2.4.1 begins. This allows us to either add or shave a few nucleotides from the ends of the oligo, operating on the assumption that most of the Insertions or Deletions, if not all of them, happened in the padded area of the oligo. As a result, the padding can also help us "realign" the oligo to hopefully counteract the shifting caused by Indels.

2.4 Consensus establishment

After the PCR amplification and sequencing are complete, we find ourselves with several hundred to several thousands of times the number of oligos that were originally encoded. Before the start of the decoding we must take these potentially noisy oligos and reduce them. This is done in two separate processes: firstly, the barcode correction tries to repair the Headers of these oligos; then, the nucleotide sequences are sorted according to these Headers and a consensus is built over them.

The details of the barcode correction process will be covered in subsection 2.4.1 but for now we will focus on the need for a correction phase before the consensus computation. After the PCR and sequencing we find ourselves with several copies of each of the original oligos. As all these copies are potentially noisy, we can try to build a consensus from them, in order to reconstruct the original oligos. But to do so, we must first be able to correctly identify which of the nucleotide sequences are copies of the same oligos. Hence, we rely on the properties of the barcodes to try and repair the damage to the Headers of

these copies that identify their oligo type and position in the image.

Once this is done, we can cluster the nucleotide sequences based on these Headers. In order to overcome the increased noisiness of the Nanopore sequencer, in this work we introduce an additional filtering step. This is done by discarding all the copies that are too different from the rest of the cluster. Here the similarity is computed by considering the Levenshtein distance [1] between the sequences, given by the following formula:

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0 \\ \min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

The filtering is necessary to reduce the chance of "poisoning" the consensus with copies that are either very noisy or that were assigned to the wrong cluster as their Headers were corrupted to the point that the barcode reconstruction was erroneous. The details of this process will be covered in subsection 2.4.2.

Once these outliers have been eliminated, we can be reasonably sure that each cluster corresponds to one of the original oligos, and we can proceed to build a consensus from them. We do this with a classical majority voting algorithm that reduces the cluster of oligos to a single consensus sequence [63], which hopefully is going to resemble the original oligo as much as possible.

2.4.1 Barcode correction

After the sequencing and PCR steps described in subsection 2.3.1 are done, we find ourselves with a multitude of noisy oligos. In the case of the Illumina sequencing, the error rate is low enough that simply selecting the ones presenting intact Headers might be sufficient to build a good enough base for the consensus. When accounting for the increased Nanopore noise, this naive approach is insufficient. We needed to reliably select the oligos that best approximated the original ones, without of course having to rely on

the ground truth, to which we would not have access to in a real scenario. The algorithm we developed for this is divided in two parts. The first part will be quickly schematized here, while the second one will be covered in subsection 2.4.2.

First, a round of barcode correction is performed in order to identify and fix the Headers of the oligos. Thanks to the structure meta-modeling that we conduct before the encoding, the barcode correction can rely on information on the structure of the formatted oligos. With this data, our algorithm identifies the positions of the Header fields in the oligos and pads any oligos that have lost nucleotides (due to Deletions). Once the quaternary strings that comprise the noisy Header and Offset fields have been found, we rely on a K Nearest Neighbors (KNN) search algorithm [61] to compare these strings to the barcode sets used when encoding the original fields, utilizing the Sequence-Levenshtein distance extension proposed in [33]. If no barcode can be found that is close enough to the noisy string (under the max distance allowed by the barcode set) the Header is considered lost and the oligo discarded. The same process is repeated for the Offsets of the Data oligos.

At the end of this first step we are left with a large number of oligos, where the Headers have been corrected (when possible), but where the Payloads are still noisy. Among these oligos will be several dozens to hundreds of copies of each of the oligos originally encoded. The next step is then to collapse all of these redundant copies to a single value. However, simply building a consensus from these oligos might yield a very low quality of results, especially when operating with very noisy sequencers like the Nanopore. How to avoid this will be shown in subsection 2.4.2

2.4.2 Outlier detection

The barcode correction step alone is not sufficient to reduce the visual distortion. For this, we perform a second step, where we filter away the outlier oligos that sneaked through barcode correction but are too damaged to contribute to the consensus, or those whose headers were corrected erroneously in the previous step. There are a great many ways of performing outlier detection (see [19] for a survey on the topic), a lot of which could be applied to our problem with relatively little effort. Rather than employ some

of the more complex approaches, we steered ourselves toward a very simple statistical detection that could still give us acceptable results, to avoid increasing too much the complexity of the workflows. Nonetheless, a more advanced and nuanced outlier detection algorithm could reduce the visual distortion even further and would be a good subject for further study.

The algorithm we developed works as follows:

- group the oligos after the barcode correction by their Headers (and Offsets in the case of Data oligos).
- For each group G do the following:
  - Compute the adjacency matrix adj of the group G.
  - Compute the average distance of each oligo in G from the others.
  - Discard the oligos that have an average distance more than one scaled Median Absolute Deviation from the median [35].
  - Compute a consensus sequence [63] from the remaining oligos. This (sometimes called canonical sequence) is a standard form of consensus for nucleotide sequences in microbiology and bio-informatics. The computation consists of aligning the oligos, and then selecting for each nucleotide in the sequence the one that appears most frequently in that position in the aligned sequences.
  - If necessary truncate or pad the resulting oligo, so that the length is the one expected by the deformatting step.

At the end of the barcode correction and consensus processes, we should be left with a single copy of each oligo that was encoded. It is theoretically possible to be lacking one or more oligos entirely if none of the copies were recoverable. This is unlikely, but it happening could be a good indicator of a violation of the biological constraints, either by the Headers or by the Payload of that oligo, since it would mean that not a single copy of the oligo was left with a recoverable Header and/or an acceptable Payload. If this is the case, the oligo is effectively lost and a visual distortion will appear in the final image. Similarly, it is possible that a consensus sequence for an oligo never encoded might be

constructed. This can happen if the Offset of a Data oligo is decoded erroneously several times, as the consensus workflows cannot know which D oligos had been originally encoded. This case does not have any associated visual distortion, as the deformatting workflow is capable of detecting these oligos as "out of bounds" for the image and automatically discards them.

2.5 Deformatting and Decoding

Once the consensus has been achieved, we can proceed with the decoding and deformatting of the oligos. These are programmatically involved, but conceptually simple processes. The deformatting consists in first decoding the information carried by the metadata in the oligos, in order to extract the information encoded by the Payload of the Data oligos and recompose it in the correct order. The final result of it are the reconstructions of the originally encoded subbands, as long strings of quaternary code.

These encoded subbands are then decoded and reshaped to retrieve the original quantized subbands, ideally as they were just after the compression and before the encoding process. Of course, in the presence of errors the decoded subbands will present some form of visual distortion.

While a first attempt at error correction is done during decoding thanks to the mapping we introduced in [57], this can only account for minor errors. To correct more glaring visual distortions, this dissertation introduces in section 2.6 a new post-processing step, with an ad-hoc workflow operated after the decoding is done but before reversing the Discrete Wavelet Transform and reconstructing the image.

2.6 Post-processing

The addition of a post-processing workflow is one of the main innovations of this work when compared to our previous ones. The necessity for it is born from the additional noisiness of the Nanopore sequencer, as previous works had mostly focused on the

(a) PSNR=48.12, SSIM=0.99 (b) PSNR=36.20, SSIM=0.74 (c) PSNR=38.70, SSIM=0.94

Figure 11: (a): image after compression; (b): image after the Barcode Correction and Consensus; (c): image after post-processing.

Illumina machine. As can be seen in Figure 11, simply relying on Barcode Correction and Consensus is not completely sufficient in removing the sequencing noise from the image in Figure 11b, we must also perform post-processing to reduce the artifacting caused during the sequencing Figure 11c.

Because of this, we decided to add an additional form of error detection and correction as a post-processing step after the decoding of the synthesized DNA. This is not an entirely novel idea, in [58] the DNA decoding had been enhanced in a similar fashion by adding an inpainting step at the end of the process, with very convincing results. The main difference from our work will lay in the type of inpainting approach chosen, as our encoding and decoding workflows (and especially their reliance on the Discrete Wavelet Transform) dictate a somewhat peculiar approach to post-processing and de-noising.

A first necessity is the identification of the damage, that we will discuss in subsection 2.6.1. This step, and its necessary automation, is a very difficult problem that both us and the authors of [58] had to find our way around. While the resulting solutions are very different, as each is tailored to the specifics of the work it is embedded in, this converging speaks of an underlying problematic that any work approaching post-processing in this way will need to prepare for.

After having identified where the damage is located, we need to try and correct it. We chose to use a sample based inpainting, a technique that relies on covering damage

using existing information from the rest of the image. The details of the chosen algorithm are presented in subsection 2.6.2. Other approaches are possible, and might work better with different types of encoding, but we performed extensive testing before choosing the method that in our opinion best suited our existing workflows.

2.6.1 Automatic Damage Detection

Automating the detection of erroneous pixels is a very complex process. Several methods have been proposed over the years, as the problem is central to a lot of image quality issues. Most of these act at a very low level in cameras and sensors [17],[27],[39] and are intended to detect either single or small clusters of erroneous pixels caused by a defect in the device itself. Other algorithms focus more on detecting noisiness; the algorithm from [21] can be used in this way, as can others [22]. Detection of the type of damage our reconstructed images exhibit would normally require a more complex approach, as the ones used in [37] and [36], to be viable.

Luckily, our encoding process puts us into a relatively rare position. As our image was decomposed by the quantization process, each noisy subband carries less information than a full-sized image. Coupled with the way our advanced mapping works, this means that the visual distortion we incur in can be identified, in the subband, by a relatively simple algorithm partially inspired by the ones used in physical cameras [62]. While more advanced detection methods might have even more success, machine learning methods might again be investigated in future works if enough training data is available, the results we were obtaining with our own algorithm were deemed sufficient.

The damage detection process is composed of two separate steps. The first step is used to detect the damage caused by Substitutions during the sequencing. These only alter few pixels at a time, and generally result a type of damage resembling spots in the final image, that will be described in points 1 and 2 of section 3.1. Detecting these distortions is relatively straightforward since they only affect very small areas. As such, it is comparable to classical salt and pepper noise, which can be detected by simply checking the value of each pixel against that of its neighbors. The first step of the detection algorithm works in this same way, comparing the value of each pixel in the subband to

Algorithm 3: Damage detection.

Input:
$S = (S_1, S_2, \ldots, S_n)$: damaged subband
$NEIGHS = (\mathcal{N}(S_1), \mathcal{N}(S_2), \ldots, \mathcal{N}(S_n))$: neighborhoods of each value in S
$TH1$: step-1 threshold
$TH2$: step-2 threshold

Output:
M : damaged values mask

$M \leftarrow$[0]; $DEV \leftarrow$[0]; $MEAN \leftarrow$[0]

for $i \leftarrow 1$ ***to*** $\#S$ **do**
    $DEV(i) \leftarrow$ standard deviation$(NEIGHS(i))$
    $MEAN(i) \leftarrow$ mean$(NEIGHS(i))$

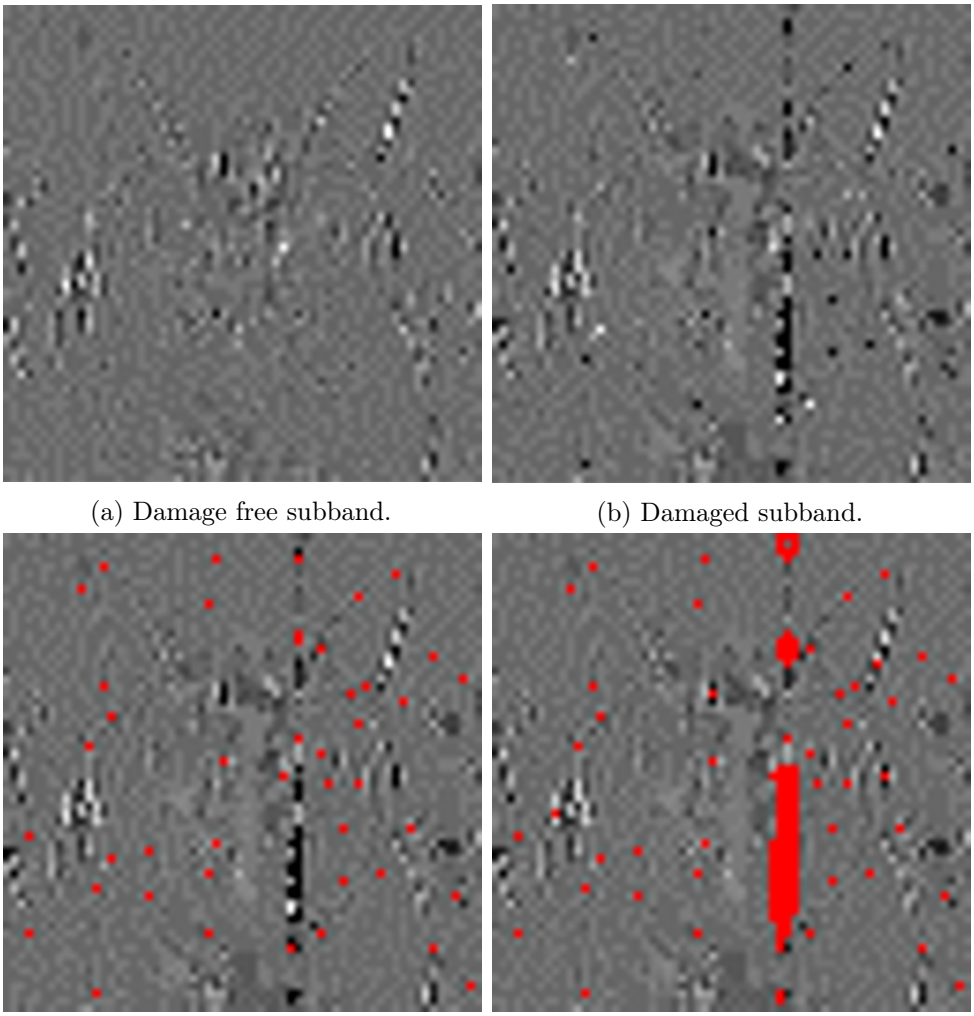
$MEAN\_DEV \leftarrow$ mean(DEV)

for $i \leftarrow 1$ ***to*** $\#S$ **do**
    if $\frac{\sqrt{(S(i)-MEAN(i))^2}}{DEV(i)} \geq TH1$ **then**
        $M(i) \leftarrow 1$ `// Indicate` $S(i)$ `as potentially damaged.`
    if $\frac{DEV(i)}{MEAN\_DEV} \geq TH2$ **then**
        $M(i) \leftarrow 1$ `// Indicate` $S(i)$ `as potentially damaged.`

that of its neighbors. Any pixel that has a variance from the rest of its neighborhood above a certain threshold is considered damaged. Again, this detection technique is only useful when working on the subbands rather than on the whole image, as the visual distortions in higher level subbands will increase in size during the reversing of the DWT. This first step, however, is not sufficient to detect more extensive damage. An example can be seen in Figure 12c. The first step detects effectively all the isolated erroneous pixels, but does not identify the large, black vertical line crossing the image.

This is in general true for all the distortions caused by lost Headers, or more rarely by several errors in a row that invalidate a lot of quantization vectors close to one another. In these cases where entire neighborhoods are affected, their average value might no longer be a good indicator of whether a pixel was damaged or not. It can be observed, however, that neighborhoods affected in this way tend to have a high internal variance. As such, we can perform a second step where we detect as potentially damaged the pixels whose neighborhoods have an internal variance above a certain threshold. The results of these two steps can be seen in Figure 12d.

A final optional step is to enlarge the binary mask built by the algorithm. This might not always be necessary, but it can help cover some of the imprecision of the previous algorithm. It consists of simply increasing the size of the features of the mask, potentially accounting for the shape and size of the quantization vectors. For example: a single pixel that was detected as anomalous might belong to a vector that was not repaired fully during the reversing of the advanced mapping. In this case it is possible that only the variance for that pixel was above the detection thresholds, but the rest of the vector could still be discolored, just within acceptable parameters. It might improve the visual distortion of the image to enlarge the binary mask, so that it if a quantization vector contains a damaged pixel, the whole vector is considered potentially damaged. This is possible since we know the shape and size of the vectors for each subbands from the S oligos and, since the VQ algorithm is deterministic, its behavior can be easily reproduced during the post processing. This option is not done by default, as it can also lead to loss of detail, but it is available in the workflow.



(a) Damage free subband. (b) Damaged subband.

(c) Binary mask (in red) after phase 1. (d) Binary mask (in red) after phase 2.

Figure 12: Creation of the binary mask from a damaged subband, without mask enlargement.

2.6.2 Wavelet inpainting

For the inpainting itself we chose to use the algorithm from [18], with some small modifications to adapt it to our case. Operating directly in the subbands allows us to use a simple inpainting algorithm. While other inpainting approaches could be possible, we opted for Criminisi's solution for a variety of reasons. Chief among them is that even when operating on the decomposition of the image we need to avoid blurring, which pushes us toward a texture synthesis or exemplar-based method. Among those, Criminisi's approach is unique for how it tries to prevent artifacting caused by employing a sub-optimal filling order, which could prove problematic for our approach. In this paragraph we will briefly describe the main characteristic of the inpainting algorithm that Criminisi et al. developed in [18], as it is an essential part of our post-processing. But, to reiterate, we did not develop this algorithm, we simply implemented it. See Criminisi et al.'s original paper for more information on its inner workings.

There are two main components to the inpainting proposed in the paper. The first one is the patch selection process, that is common to almost all exemplar-based inpainting approaches. A schematization of one step of this algorithm can be seen in Figure 13.

At the beginning of the algorithm a part of the image is designated as the target area, that will be inpainted, while the rest of the picture is considered as the source area. In our case this is done by the binary mask constructed during the damage detection phase, where the true values indicate pixels belonging to the target region and vice versa. An example situation can be seen in Figure 13a, with the source region ($\mathbf{\Phi}$), target region ($\mathbf{\Omega}$) and its contour ($\delta\mathbf{\Omega}$) being clearly indicated. To perform one iteration of the inpainting process, the algorithm designates a patch of pixels that will be inpainted. The patch (ψ_p) is centered on a point $p \in \delta\mathbf{\Omega}$, so that it contains both intact pixels belonging to $\mathbf{\Phi}$ and pixels that will be inpainted belonging to $\mathbf{\Omega}$. This first step can be seen in Figure 13b. After the patch has been selected, the algorithm searches over the image for the patch that best approximates the viable pixels of $\mathbf{\Phi}_p$. Once the best fitting one (in this example $\mathbf{\Phi}_{(q'')}$, centered on q'') has been found, it is copied over the original patch, effectively inpainting the pixels that belonged to the target region. This search and replace process can be seen in Figure 13c and Figure 13d. If the same inpainting had been performed with

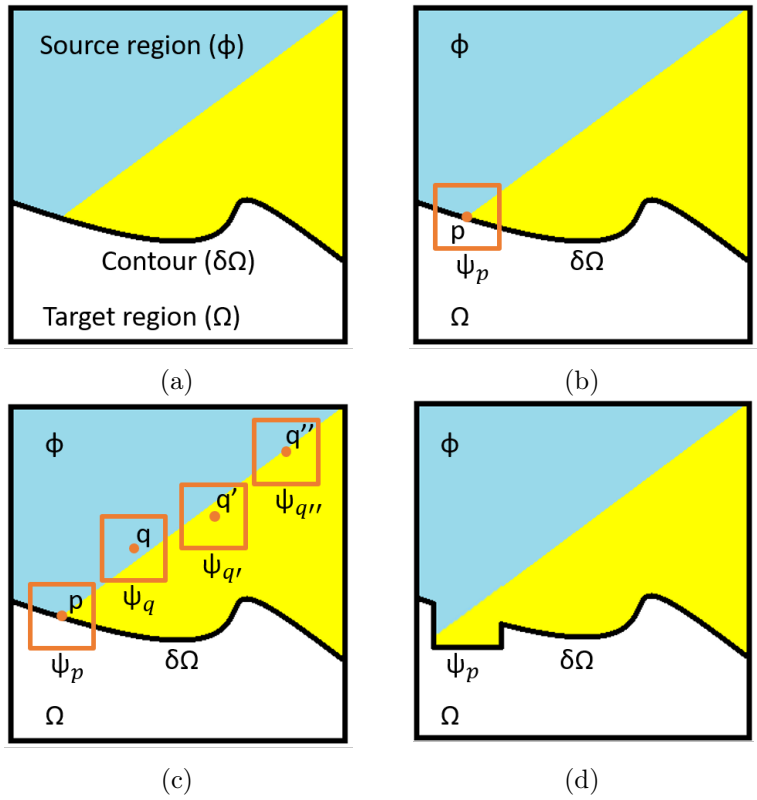


Figure 13: Patch selection process for the Criminisi algorithm. (a): image with the damaged area in white; (b): selection of patch ψ_p; (c): search over the image for the patch best matching ψ_p; (d): damaged area shrinking after the inpainting iteration is complete.

a PDE method the demarcation between yellow and azure areas would have been lost, as the blurring effect would have caused them to bleed into each other. An unintended but welcome upside of performing inpainting in the subbands, rather than on the whole image, is that this type of algorithms scales poorly with the size of the image. Thus, performing the inpainting over the decomposition of the image is much faster than doing the same over the intact image.

Of course, the patch selection process is only one part of the inpainting process, and it is performed similarly by most exemplar-based approaches. What sets Criminisi's inpainting method apart from the rest is the ordering in which the patch selection is performed. In Figure 13b we placed point p on the contour $\delta\Omega$. This makes sense, as picking a patch centered on the division between source and target regions strikes a

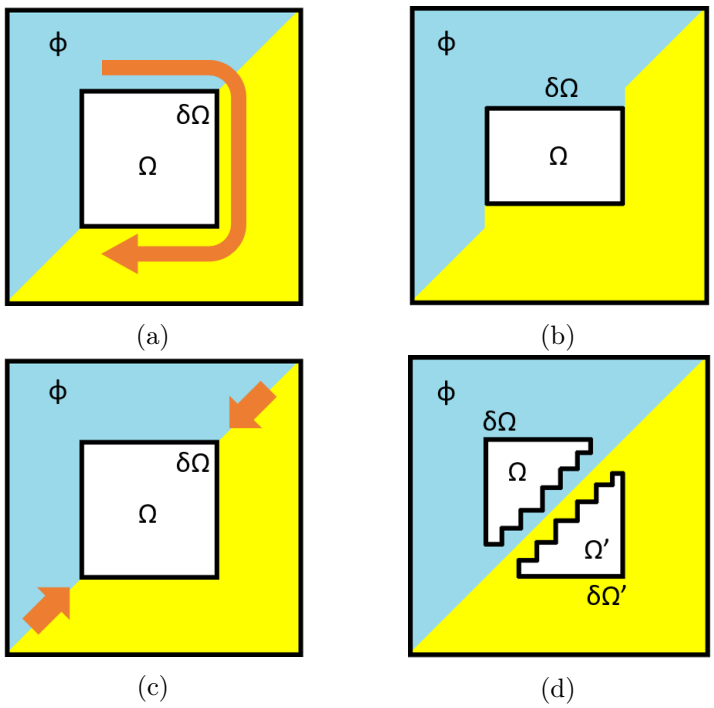


Figure 14: Effects of Criminisi's edge-driven ordering, compared with a classical onion peel ordering. (a),(b): schematization and effects of onion peel ordering. (c),(d): schematization and effects of priority-based ordering. Note the artifacting in (b).

good balance between pixels to inpaint and pixels to be used for the selection of the replacement patch. But we also, purposefully, picked a point on the intersection between the yellow and azure textures that the source area is composed of. This is because of the priority-based ordering used to determine which patch of pixels to inpaint. An example of this ordering algorithm can be seen in Figure 14.

The classical way to decide which patch to inpaint next is called "onion peel" ordering. Effectively, the algorithm simply goes around the contour $\delta\boldsymbol{\Omega}$, picking the patches in a spiral concentric fashion. This can cause artifacting, an example of which is shown in Figure 14b, somewhat exaggerated for visibility. This happens when the algorithm selects a patch that should contain more than one type of feature (in the example both yellow and azure texture) but, due to the ordering, the viable pixels from the original patch are not a correct representation of these features (in the example, the patch selected only had

one of the two textures, causing an overshoot of that texture past the diagonal). Similar issues can be seen when inpainting concave, irregular and/or complex surfaces.

Criminisi's priority-based, edge-driven ordering algorithm instead borrows from PDE methods to try and determine a filling order that minimizes artifacting in the image. This is done by balancing two terms. The first aims to prioritize patches where there is a lot of known information (encouraging the selection of protruding parts of the target region, that are surrounded by pixels belonging to the source region). The other term is the one that is most inspired by Partial Derivative Equations, and it tries to prioritize patches that lie on the edges of structures in the image. This is done by finding and propagating into the target area the isophote lines from the source region (in the example, the diagonal is an edge between two structures, one with a yellow texture and one with an azure one). The result is an inpainting algorithm that first tries to complete the edges of the structures that the damage is covering, prioritizing the patches it has a high confidence in. This is especially useful when dealing with occlusion removal, even more so when the original image is a photograph with complex textures and background structures.

While our situation is not exactly the one the algorithm was built to handle, its characteristics match our needs well enough. As already stated, the possibility to cover relatively large damage with no blurring was the thing that originally drew us to this, and others, exemplar-based approaches, but the edge-driven ordering proved very relevant in avoiding artifacting, once applied to the subbands rather than to the entire image. As such we decided to implement the algorithm and apply it to the damaged area detected in the first step of the post-processing workflow. The only modifications to the algorithm presented in [18] concerned the implementation details, as in the original paper the contour of the target region is defined by the user, while for us it is derived from the binary mask. Additionally, our implementation is only designed to operate on grey-scale images, but it could easily be expanded to operate on three channels, either separately or taken as a single three-dimensional matrix.

Chapter 3

Experimental results

In this chapter we will analyze the behaviour of the proposed workflow in simulations of realistic environments, over multiple of images, error profiles, and simulated sequencers with varying accuracy. Here we defined accuracy as how likely the sequencing machine is to read a nucleotide without error. It can be treated simply as 100 minus the error rate for that sequencer; so, for example, the Nanopore MinION simulated in section 3.2 would register as having a sequencing accuracy of 95.19%, as it has a total error rate of 4.81%. We will first show (section 3.1) some of the test done at the beginning of the work, to better understand and categorize the visual damages cause by sequencing noise.

In section 3.2 we will then simulate our encoding and decoding workflow on two different images, in order to observe its behaviour depending on the characteristics of the image. For this first series of experiments we simulated a Nanopore sequencing accuracy of 95.19%, with a rate of 1.5% for Substitutions, 1.01% for Insertions and 2.3% for Deletions, adjusted from [60]. The simulated noise was concentrated at the ends of the oligos, with 80% appearing in the first and last 20nt of the DNA strands, based on the observations from [43], as well as our owns. It should be noted that these experiments are meant as a proof of concept of the methods presented in this work. More in-depth tests are recommended and laid out in chapter 4 after our conclusions.

In section 3.3 we simulated a much stronger noise, with an increased number of Substitutions. The goal was to increase the effects of the noise on the image, to test the

(a) Image "Rico", uncompressed. (b) Image "May-Lily", uncompressed.

Figure 15: Images used in the experiments shown in this chapter.

performance of the algorithms when approaching the Substitutions rate of 5.1% measured originally in [43]. This is to be intended as a battery of experiments dealing with a worst-case scenario, as more recent Nanopore sequencers have shown total error rates closer to the ones utilized in section 3.2, with some even approaching 97-98% accuracy.

Lastly, in section 3.4 we compiled a table of numerical results, observing the behaviour of the algorithm at different levels of sequencing accuracy. This was done to show how the performance of the post-processing changes depending on the amount of noise introduced during the sequencing.

Figure 15 shows The images chosen for these experiments. Figure 15a ("Rico") was chosen as it presents fairly complex textures (especially in the fur). Additionally, the human brain is wired to recognize facial features. While the face in question is feline rather than human, it should still make damage concerning the eye and muzzle areas especially evident to visual inspection. Figure 15b ("May-Lily") was chosen to contrast the "Rico" image. The textures are simpler, and we can expect good performance on them, but there are sharp edges between darker and lighter textures; in cases like this blurring, artifacting, and texture bleed-out could be especially noticeable. When operating on this image the algorithm has to properly discern the damage caused by noise and the rapid switch from a light to a darker neighboring texture.

(a) PSNR=36.78dB, SSIM=0.79 (b) PSNR=35.04dB, SSIM=0.71

Figure 16: Damage caused by Substitutions only. (a): damage applied only to Payloads. (b): damage applied to both Headers and Payloads.

3.1 Analysis of the expected damage

After conducting a review of the literature to identify the strengths and weaknesses of the different inpainting techniques, we started focusing on simulating the type of visual distortion we could expect in the decoded images. The decision of which inpainting method to utilize during post-processing would obviously depend on the characteristics of the damage we would incur in this first battery of tests. Here we present two of these experiments, in order to further discuss the expected types of visual distortions.

The tests shown in Figure 16 were conducted on the "Rico" (Figure 15a) image at the start of the work and intended simply as a proof of concept for the inpainting process. As such, only Substitutions were simulated, no Insertions or Deletions, and the noise was uniformly distributed over the oligo, rather than concentrated in the ends as it is with the Nanopore. Nonetheless, they can help schematize the types visual distortion we can see at the end of the decoding process:

- Black and white spots, visible in both Figure 16a and Figure 16b. These are a distortion caused by errors in the decoding of the LL subband, that carries most of the information. Some of these errors cause only minor discolorations, thanks to

the advanced mapping proposed in [57], while others are effectively unrecoverable and appear as stark black and white stains. Crucially, these are several pixels in radius, as opposed to most cases of salt and pepper noise. The specific dimensions vary according to the size of the VQ vectors.

- Crisscross patterns, again visible in both Figure 16a and Figure 16b. These are similar in size to the spots, and are caused by similar erroneous decoding, on the other subbands (LH, HL and HH). As these subbands carry the details of the image, the variations in pixel color, the distortion is less defined. Again, similarly to the first type of distortion, the size and shape is depending on those of the vectors used to quantize the image.

- Long lines only found in Figure 16b. These are caused by errors on the Headers, and as such are only present in the image where noise was simulated on the Header and Offset field of the oligos. The cause here is the complete loss of an oligo, either because the Offset was decoded erroneously or because the identifying information was lost. The result can be either a long black line (for example the vertical one through the cat's left eye), or a line of crisscross (they are harder to see, there are two horizontal ones, one at the bottom of the image and one over the left ear). The type depends on the subband of the oligo that was lost but they are grouped together, as the problematic they present for the post-processing algorithm is similar.

After having identified the characteristics of the damage, we tried repairing it with traditional inpainting methods. We can see even just from a visual standpoint that the results from our algorithm are more satisfactory than the ones obtained with classical Criminisi inpainting. In the image where noise was applied only on the Payloads, we gained 2.65dB of PSNR over the noisy image, and 2.35dB over the traditionally inpainted image. Lower, but still comparable, results can be seen in the image with noise applied to both the Headers and the Payloads, where we can see a 1.47dB gain over the damaged image and 1.33dB over the image inpainted with Criminisi's algorithm. The Structure Similarity Index Measure is an even more telling indicator of the image fidelity than Peak Signal to Noise Ratio in this case, and we can see the same gains in this metric as well.

(a) PSNR=37.08dB, SSIM=0.84. (b) PSNR=39.43dB, SSIM=0.93.

Figure 17: Inpainting comparison, on Figure 16a. (a) repaired utilizing Criminisi's [18] algorithm; (b) inpainted as part of our workflows.

(a) PSNR=35.14dB, SSIM=0.77. (b) PSNR=36.47dB, SSIM=0.85.

Figure 18: Inpainting comparison on Figure 16b. (a) repaired utilizing Criminisi's [18] algorithm; (b) inpainted as part of our workflows.

Our approach gains 0.14 structure similarity compared to the damaged image, and 0.09 compared to the traditionally inpainted one, for handling Payload noise. Similarly, we gain 0.14 SSIM compared to the damaged image, and 0.08 compared to the inpainted one, when noise is applied to both Payloads and Headers.

From these results we can infer that our inpainting approach could be effective at repairing damage that might have slipped through the decoding process. While the damage simulated here is only caused by substitutions, the test shows that the damage detection and inpainting process can handle all the types of visual distortion that we can expect from the sequencing. The next step was then to simulate several full encoding and decoding workflows, complete of post-processing, to see how our algorithms behaved in the presence of more realistic Nanopore noise.

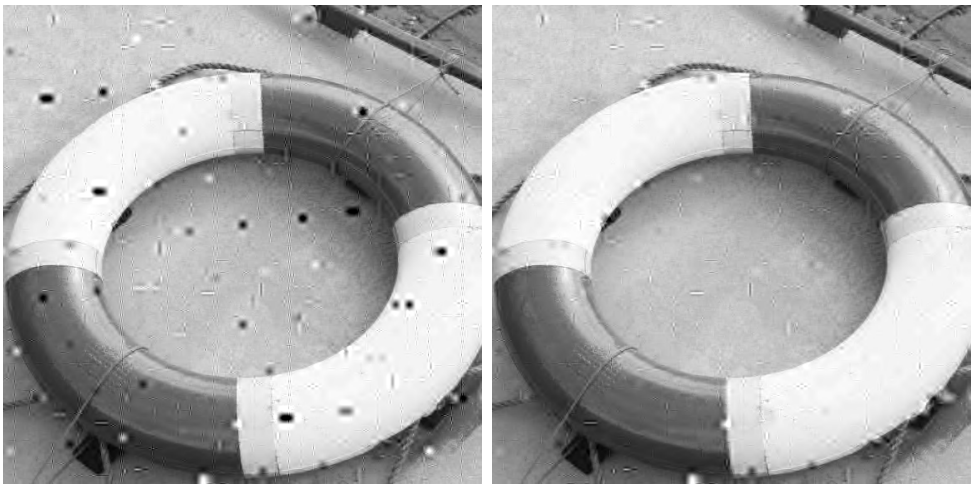
3.2 Experiments with realistic Nanopore noise

These experiments were conducted according to the workflow presented in chapter 2 and visualized in Figure 3. The "Rico" and "May-Lily" images shown in Figure 15 were compressed, quantized, encoded, and formatted following the steps comprising the encoding workflow, as explained in section 2.1 and section 2.2. Between the compression and encoding process we also applied the parameter computation step described in subsection 2.1.1, to best calculate the values that would be needed in the post-processing workflows. These values were then added to the formatting of the oligos, according to the oligo structure described in Figure 5. To simulate the Nanopore sequencing noise, we created 200 noisy copies of each formatted oligo. This was done to mimic the process of PCR amplification, followed by the introduction of noise during the sequencing, which produces multiple noisy reads. At the end of this we similarly had a set of multiple copies of the formatted oligos, each containing different errors in different positions, as it would happen in a real wet experiment.

To decode the noisy oligos we similarly followed the procedures laid out in section 2.4 and section 2.5, starting by applying the barcode correction process detailed in subsection 2.4.1 to distinguish the oligos of different types from each other, and in order to

(a) PSNR=37.92dB, SSIM=0.82. (b) PSNR=39.50dB, SSIM=0.90.

Figure 19: Realistic noise experiment from section 3.2, on "Rico". (a): picture after the consensus; (b): picture after the post-processing is complete.



(a) PSNR=36.52dB, SSIM=0.84. (b) PSNR=38.28dB, SSIM=0.93.

Figure 20: Realistic noise experiment from section 3.2, on "May-Lily". (a): picture after the consensus; (b): picture after the post-processing is complete.

locate their position in the image. Once the barcode correction was done, we proceeded to cluster the noisy oligos according to their Headers, and then pruned away the outliers to ensure the quality of the consensus. Once each cluster had been cleaned, we aligned the remaining oligos and used them to build a consensus sequence for each cluster, intended to be the most representative version of each oligo. These consensus sequences were then de-formatted and decoded, the result of which is shown in Figure 19a and Figure 20a. In these figures we calculated the Peak Signal to Noise Ratio and Structure Similarity Index Measure of both damaged and repaired images, with respect to the images in 15 after the compression.

Finally, we applied the post-processing algorithms described in section 2.6, starting with the damage detection shown in subsection 2.6.1, utilizing the parameters determined during the encoding of the image and recovered by the deformatting process. Once the damaged areas had been identified, we performed the inpainting directly in the subbands, utilizing the algorithm described in subsection 2.6.2. The results can be seen in Figure 19b and Figure 20b.

Analyzing the images, we observed a 1.5dB gain of Peak Signal to Noise Ratio in the case of "Rico", one of 1.76dB in the case of "May-Lily", when compared to the damaged image. The Structure Similarity Index has similarly increased, by 0.8 in the first case and 0.9 in the second. We can note that the gain in PSNR is lower than it was in the exploratory tests, while the SSIM increase is comparable. This is likely due to the addition of insertions and deletions causing more damages. One the one hand, the processes employed before the post-processing are capable of pruning the more impactful effects of Indels. For example, the significant damage lines indicated in point three of section 3.1 and visible in Figure 16 that are caused by missing or heavily damaged oligos can be avoided thanks to the padding and added protection for the Headers, as well as the outlier detection algorithm shown in subsection 2.4.2. On the other hand, the increased noise causes a lot of damage to the detail subbands (HH, HL and LH), where effectively identifying and repairing the erroneous pixels is a lot harder.

From a visual inspection we can see that most of the noise on the subbands has been cleaned by the post-processing workflow. This is especially true for the black spots in

both images, while lighter damages proved more difficult to identify due to the hue of the images themselves. In the "May-Lily" image the result is generally better than in the "Rico" image, as expected, and the post-processing caused little to no artifacting between the texture areas. In both images the hardest damage to identify and repair effectively came from the detail subbands; this is true especially in "Rico", where the textures are busier than in "May-Lily" and thus it is harder to separate the noise from the decomposition of the fur texture.

Overall the results of these tests are satisfactory. While they could be further improved, as we will briefly discuss in chapter 4, they represent a solid proof of concept for the DNA storage workflow as a whole and for the post-processing approach in particular.

3.3 Experiments with worst-case noise

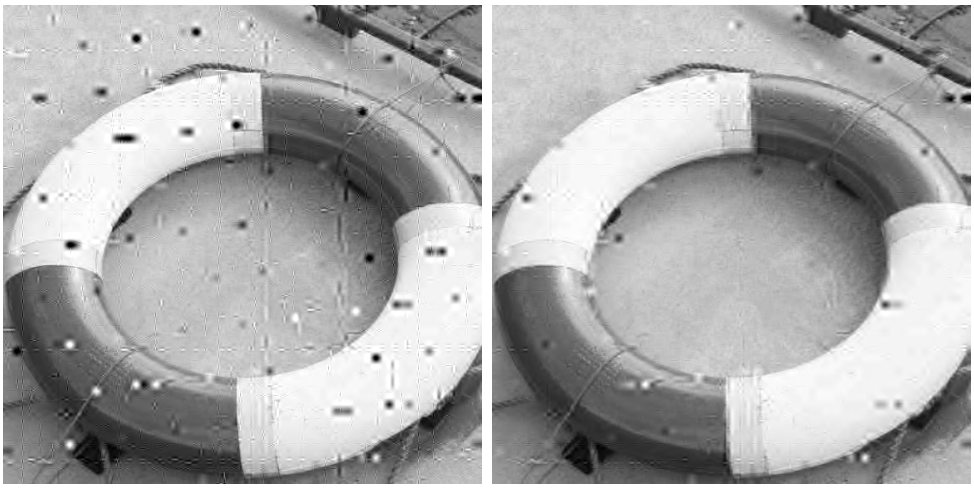
Similarly to what was done in section 3.2, in this experiment we encoded, formatted, decoded, and inpainted the images shown in Figure 15a and Figure 15b according to the workflow presented in chapter 2. This was done twice, once with a high — but still realistic — noise simulation, and once with very high and close to worst-case error percentages. After the decoding and post-processing we again calculated the Peak Signal to Noise Ratio and Structure Similarity Index of all the damaged and repaired images, compared with their compressed, but undamaged, versions.

The results of the first experiment can be seen in Figure 21 and Figure 22. The procedure and images utilized in the experiments shown here are the same as the ones in section 3.2. The difference is the accuracy of the simulated sequencer. In this first experiment we will simulate an accuracy of 93.69%, effectively doubling the expected number of Substitutions.

Despite this the, results of the post-processing are comparable to that of the experiment in section 3.2. The post-processing managed to gain 1.26dB of Peak Signal to Noise Ratio in the case of "Rico" and 1.93dB in the "May-Lily" image. Structure Similarity Index gains where similarly reminiscent of the results of the first experiments in section 3.2, with a gain of 0.11 and 0.12 SSIM respectively. In the case of Figure 22b the improvement

(a) PSNR=37.42dB, SSIM=0.79. (b) PSNR=38.68dB, SSIM=0.90.

Figure 21: First worst-case noise experiment from section 3.3, on "Rico". (a): picture after the consensus; (b): picture after the post-processing is complete.



(a) PSNR=35.53dB, SSIM=0.80. (b) PSNR=37.46dB, SSIM=0.92.

Figure 22: First worst-case noise experiment from section 3.3, on "May-Lily". (a): picture after the consensus; (b): picture after the post-processing is complete.

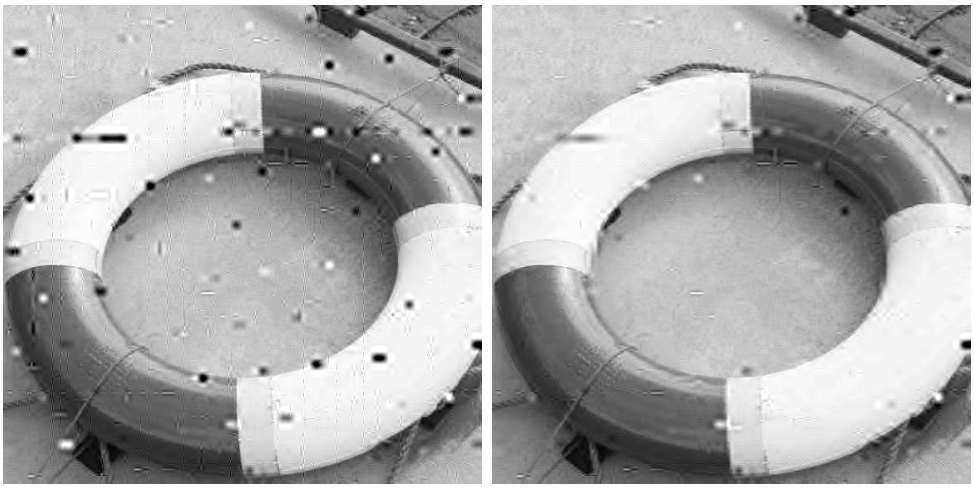
is actually more marked than before, although the final PSNR achieved is of course lower, at 37.46dB versus 38.28dB for Figure 20b. Nonetheless, the results are interesting. Even with an higher sequencing noise the post-processing can repair most of the obvious damage, as evidenced by the increase in SSIM as well as qualitative visual examination. The pitfalls remain the same ones underlined in the previous experiments: extensive damage to the HL, LH and HH subbands is hard to identify and correct. While the algorithm can still do so, at least for the bigger damages, it results into a loss detail. This can be seen as blurring and loss of definition in both the fur in Figure 21b and on the demarcation between light and dark textures in Figure 22b.

For the second experiment in this section we brought the Substitution rate to 5.1%, in line with the original findings from [43]. This in turn gives us a sequencing accuracy of 91.68%. The results of this experiment can be seen in Figure 23 and Figure 24. Beyond this threshold, the encoding starts breaking down, with the image becoming effectively unrecoverable at less than 85% accuracy. It should be noted that this effect can be avoided, or at least reduced, by strengthening the encoding scheme, protecting more the header data, increasing the simulated PCR redundancy, and/or decreasing the compression rate of the image (as it reduces the effect that a lost oligo or a damaged nucleotide has on the reconstructed image). However, these methods are outside the context of this experiment, as they revolve around the performance of the pre-existing workflow rather than the behaviour of the post-processing step.

Numerically, the expected gains are still present. A PSNR increase of 2.7dB and 2.49dB in Figure 23b and Figure 24b respectively is very significant, and it brings the final PSNR values for these images in line with the ones from the previous experiments. The same is true for the SSIM gain and final values, despite the significant increase in sequencing noise. From a visual inspection, however, we can infer that the algorithm is starting to struggle at keeping up with the amounts of noise introduced. The lines of missing and/or damaged oligos visible in both Figure 23a and Figure 24a are reminiscent of Figure 16b, and similarly indicative of either Header damage, or at least some almost undecodable Payloads. Damages like this are to be avoided rather than inpainted, whenever possible, which is the purview of the outlier detection and consensus steps. Evidently at this levels of noise the consensus sequences of some oligos are becoming

(a) PSNR=35.98dB, SSIM=0.75. (b) PSNR=38.68dB, SSIM=0.90.

Figure 23: Second worst-case noise experiment from section 3.3, on "Rico". (a): picture after the consensus; (b): picture after the post-processing is complete.



(a) PSNR=34.91dB, SSIM=0.77. (b) PSNR=37.40dB, SSIM=0.91.

Figure 24: Second worst-case noise experiment from section 3.3, on "May-Lily". (a): picture after the consensus; (b): picture after the post-processing is complete.

accuracy	**damaged image**		**inpainted image**		**gain**	
MayLily	PSNR	SSIM	PSNR	SSIM	I-PSNR	I-SSIM
99%	38.51	0.92	39.21	0.95	0.70	0.03
98%	38.48	0.90	39.09	0.95	0.61	0.05
97%	36.70	0.81	38.38	0.90	1.68	0.09
96%	35.90	0.81	37.67	0.91	1.77	0.10
95%	35.67	0.81	37.78	0.92	2.11	0.11
94%	35.64	0.80	37.21	0.90	1.57	0.10
93%	35.36	0.80	37.12	0.90	1.76	0.10
92%	34.97	0.78	36.65	0.89	1.68	0.11
91%	33.95	0.74	35.46	0.85	1.51	0.11
90%	29.36	0.40	29.78	0.53	0.42	0.13
Rico	PSNR	SSIM	PSNR	SSIM	I-PSNR	I-SSIM
99%	41.08	0.92	39.67	0.92	-1.41	0.00
98%	39.48	0.86	39.73	0.91	0.25	0.05
97%	37.30	0.80	38.21	0.87	0.91	0.07
96%	37.64	0.80	39.13	0.89	1.49	0.09
95%	37.49	0.79	38.98	0.88	1.49	0.09
94%	37.21	0.78	38.39	0.87	1.18	0.09
93%	36.47	0.77	37.55	0.85	1.08	0.08
92%	31.34	0.40	32.19	0.49	0.85	0.09
91%	29.02	0.18	29.32	0.21	0.30	0.03
90%	28.94	0.25	29.03	0.30	0.09	0.05

Table 1: PSNR and SSIM gains related to sequencing accuracy.

poisoned despite the presence of these steps, either due to a lower redundancy or less header protection that it would be needed. While the algorithm can still repair these types of damages in some cases (as can be seen on the cat right ear in Figure 23b) the inpainting is not always seamless, and largely dependant on the area around the damage.

3.4 Numerical experiment

To close this experimental chapter we want to present a quantitative analysis. We will be analyzing the performance of our proposed workflow in relation with varying levels of sequencing accuracy. We will be comparing the Peak Signal to Noise Ratio and Structure

Similarity Index of damaged and repaired images at various levels of sequencing accuracy, from the 99% accuracy expected of classical Illumina sequencers down to 90% accuracy, after which the encoding and decoding process starts breaking down. The tables relating PSNR and SSIM with sequencing accuracy can be seen in Table 1.

From the tables we can see that the gain in terms of PSNR and SSIM presents a peak at a sequencing accuracy of around 95%. This is not surprising. The work was developed to operate on the Nanopore MinION, that has an effective accuracy of 95% to 97%. It stands to reason that, given the default parameters, it would operate best around these margins. At lower accuracy percentages the workflow struggles to cover up the more extensive damage, with the image starting to become unreadable around 90% accuracy. At higher fidelity, we have the opposite problem. In these cases the gains reduce as, after having covered up what little damage was present in the image, the algorithm starts inpainting over details. This is especially visible in the case of Rico, that has complex textures. At 98% accuracy we have a minimal gain in the measured metrics, while at 99% we have an actual loss, as the algorithm causes blurring of details and especially fur texture, in the image.

It must be noted, however, that his "bump" in the algorithm efficacy can theoretically be moved. If the algorithm is supposed to operate on a sequencing machine with a lower error rate, and thus an higher accuracy, the parameters used in the compression, damage detection, and inpainting steps can be modified by the user before the start of the encoding. For example, a user might specify a tighter behaviour in the computation of damage detection parameters, tolerating fewer false positives. Indeed, it would be even possible to utilize the meta-data carried by the Headers of the Global, Subband and Data oligos to indicated if different inpainting or outlier detection algorithms should be used, as they might be better suited to the specific image or noise level. Or even to indicate the need to avoid post-processing entirely, in cases where it might be counterproductive. The packet-like structure of the formatted oligos lends itself well to such uses.

Overall, the numerical results are satisfying. The post-processing can increase the PSNR of most tested accuracy levels, even in case where the image is significantly damaged. While it performs better over a small range of noise levels, this range can be

moved by manipulating the configuration of the algorithms, rather than the algorithms themselves. It should be noted that the composition of the sequencing noise is likely to have as much of an impact , if not more, as the level of noise itself. As the tests in the previous sections have shown, the algorithm excels at covering Substitution noise, but can struggle with the larger damages caused by Indels. In these cases, it would be the responsibility of the outlier detection and consensus steps to prune away all the oligos that might be too damaged to contribute to an effective decoding. Again, this can be ensure by properly configuring these steps, as well as minimizing the presence of contiguous patterns in the DNA strands.

Chapter 4

Conclusions

In this work we have developed a general workflow to store digital images into DNA, utilizing the encoding algorithm proposed in [57] and expanding the previous works shown in [51] and [54]. The proposed workflow (outlined throughout 2) increases the noise resistance of the process by optimally assigning the vector indexes of an image quantized using Vector Quantization, to reduce the visual impact of sequencing errors. We also utilized an inline error correcting approach in the form of DNA barcodes [33] to encode the Headers of the formatted oligos, which coupled with a custom outlier detection step and a majority voting consensus algorithm allowed us to further reduce the visual distortion caused by sequencing errors. Finally, we introduced a post-processing step, utilizing an algorithm for detecting any remaining damaged areas and applying inpainting to improve the final image quality. This novel workflow can provide redundancy and inline error correction without increasing the cost of the synthesis and sequencing process, complemented by a post-processing approach to further improve the quality of the decoding.

Additionally, while most works up to date are focused on the Illumina sequencer thanks to its higher accuracy, we explicitly introduced the Nanopore MinION sequencer in our workflow in order to speed up the process and reduce the cost. We carried out an analysis on the performance of the DNA encoding and decoding process in the presence of simulated Nanopore sequencing noise, as well as higher error rates in order to test the

noise resilience of the proposed workflow.

We also introduced some possible further avenues of research, for example the possibility of improving the error detection and/or the inpainting performance in the post-processing step by relying on other more advanced algorithms. Again, machine learning based approaches could be of great help in both these steps, especially if treated as a blind inpainting problem, should a sufficiently large training set be available to retrain the convolution layers where needed.

Lastly, we proposed the possibility of further strengthening the error resistance of the encoding algorithm in [57] by employing a Double Representation approach. This would aim to reduce the amount of pattern repetitions in the oligos, reducing the sequencing noise by more closely abiding by the biological constraints of synthetic DNA encoding. To make this possible we developed an ad-hoc clustering algorithm, in order to maintain the advantages of the DeMarca encoding algorithm utilized in this work and [57] while still utilizing multiple codebooks during the encoding.

Nonetheless, this study was conducted by relying on a simulation of the Nanopore noise and intended mostly as a proof of concept. As such a proper wet-lab experiment is the most pressing priority to verify in practice the efficiency, as well as the efficacy, of the proposed encoding, decoding and post-processing workflows.

Thanks

I would like to thank Marc Antonini, both for his work as advisor and for his constant guidance during the project. I would also like to thank Eva and Melpo for being a joy to work with, especially in these trying times. This internship could not have happened without them, from their previous work on this subject to the exemplar work they did during this project. Being given the possibility to work on this part of their most recent research on DNA encoding was a pleasure and an honor. Additionally, I want to thank Prof. Serena Morigi for helping me shape this work into a proper thesis, in spite of the little time we had to do it.

Lastly, I would like to thank Stefano for being here with me. These twelve months would not have been the same without him.

Bibliography

[1] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals", in *Soviet physics doklady*, Issue: 8, vol. 10, 1966, pp. 707–710.

[2] L. M. L. Cam and J. Neyman, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press, 1967, 690 pp., Google-Books-ID: IC4Ku_7dBFUC.

[3] R. Dubes and A. K. Jain, "Clustering techniques: The user's dilemma", *Pattern Recognition*, vol. 8, no. 4, pp. 247–260, Oct. 1, 1976, ISSN: 0031-3203. DOI: 10.1016/0031-3203(76)90045-5. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0031320376900455.

[4] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design", *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84–95, Jan. 1980, Conference Name: IEEE Transactions on Communications, ISSN: 1558-0857. DOI: 10.1109/TCOM.1980.1094577.

[5] 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.

[6] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis", *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, Nov. 1987, ISSN: 03770427. DOI: 10.1016/0377-0427(87)90125-7. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0377042787901257.

[7] 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.

[8] Leonard Kaufman, *Finding groups in data*, in collab. with Internet Archive. Wiley, 1990, 380 pp., ISBN: 978-0-471-87876-6. [Online]. Available: `http://archive.org/details/findinggroupsind00kauf`.

[9] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform", *IEEE Transactions on Image Processing*, vol. 1, no. 2, pp. 205–220, Apr. 1992, ISSN: 10577149. DOI: `10.1109/83.136597`. [Online]. Available: `http://ieeexplore.ieee.org/document/136597/`.

[10] A. Efros and T. Leung, "Texture synthesis by non-parametric sampling", in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, Kerkyra, Greece: IEEE, 1999, 1033–1038 vol.2, ISBN: 978-0-7695-0164-2. DOI: `10.1109/ICCV.1999.790383`. [Online]. Available: `http://ieeexplore.ieee.org/document/790383/`.

[11] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting", in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '00, USA: ACM Press/Addison-Wesley Publishing Co., Jul. 1, 2000, pp. 417–424, ISBN: 978-1-58113-208-3. DOI: `10.1145/344779.344972`. [Online]. Available: `https://doi.org/10.1145/344779.344972`.

[12] T. F. Chan and J. Shen, "Nontexture inpainting by curvature-driven diffusions", *Journal of Visual Communication and Image Representation*, vol. 12, no. 4, pp. 436–449, Dec. 1, 2001, ISSN: 1047-3203. DOI: `10.1006/jvci.2001.0487`. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S1047320301904870`.

[13] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer", in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*, Not Known: ACM Press, 2001, pp. 341–346, ISBN: 978-1-58113-374-5. DOI: `10.1145/383259.383296`. [Online]. Available: `http://portal.acm.org/citation.cfm?doid=383259.383296`.

[14] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, "Simultaneous structure and texture image inpainting", *IEEE Transactions on Image Processing*, vol. 12, no. 8, pp. 882–889, Aug. 2003, ISSN: 1057-7149. DOI: 10.1109/TIP.2003.815261. [Online]. Available: http://ieeexplore.ieee.org/document/1217265/.

[15] I. Drori, D. Cohen-Or, and H. Yeshurun, "Fragment-based image completion", in *ACM SIGGRAPH 2003 Papers*, ser. SIGGRAPH '03, New York, NY, USA: Association for Computing Machinery, Jul. 1, 2003, pp. 303–312, ISBN: 978-1-58113-709-5. DOI: 10.1145/1201775.882267. [Online]. Available: https://doi.org/10.1145/1201775.882267.

[16] S. Rane, G. Sapiro, and M. Bertalmio, "Structure and texture filling-in of missing image blocks in wireless transmission and compression applications", *IEEE Transactions on Image Processing*, vol. 12, no. 3, pp. 296–303, Mar. 2003, Conference Name: IEEE Transactions on Image Processing, ISSN: 1941-0042. DOI: 10.1109/TIP.2002.804264.

[17] I. Baharav, R. Kakarala, X. Zhang, and D. W. Vook, "Bad pixel detection and correction in an image sensing device", U.S. Patent 6737625B2, May 18, 2004. [Online]. Available: https://patents.google.com/patent/US6737625B2/en.

[18] A. Criminisi, P. Perez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting", *IEEE Transactions on Image Processing*, vol. 13, no. 9, pp. 1200–1212, 2004.

[19] V. Hodge and J. Austin, "A survey of outlier detection methodologies", *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, Oct. 1, 2004, ISSN: 1573-7462. DOI: 10.1023/B:AIRE.0000045502.10941.a9. [Online]. Available: https://doi.org/10.1023/B:AIRE.0000045502.10941.a9.

[20] J. Sun, L. Yuan, J. Jia, and H.-Y. Shum, "Image completion with structure propagation", in *ACM SIGGRAPH 2005 Papers*, ser. SIGGRAPH '05, New York, NY, USA: Association for Computing Machinery, Jul. 1, 2005, pp. 861–868, ISBN: 978-1-4503-7825-3. DOI: 10.1145/1186822.1073274. [Online]. Available: https://doi.org/10.1145/1186822.1073274.

[21] D. Tschumperle and R. Deriche, "Vector-valued image regularization with PDEs: A common framework for different applications", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 4, pp. 506–517, Apr. 2005, Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2005.87.

[22] P.-E. Ng and K.-K. Ma, "A switching median filter with boundary discriminative noise detection for extremely corrupted images", *IEEE Transactions on Image Processing*, vol. 15, no. 6, pp. 1506–1516, Jun. 2006, Conference Name: IEEE Transactions on Image Processing, ISSN: 1941-0042. DOI: 10.1109/TIP.2005.871129.

[23] J. Hays and A. A. Efros, "Scene completion using millions of photographs", *ACM Transactions on Graphics*, vol. 26, no. 3, 4–es, Jul. 29, 2007, ISSN: 0730-0301. DOI: 10.1145/1276377.1276382. [Online]. Available: https://doi.org/10.1145/1276377.1276382.

[24] 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.

[25] A. Bugeau, M. Bertalmío, V. Caselles, and G. Sapiro, "A comprehensive framework for image inpainting", *IEEE Transactions on Image Processing*, vol. 19, no. 10, pp. 2634–2645, Oct. 2010, Conference Name: IEEE Transactions on Image Processing, ISSN: 1941-0042. DOI: 10.1109/TIP.2010.2049240.

[26] I. Inc, *Illumina Sequencing Technology: highest data accuracy, simple workflow, and a broad range of applications.* Technology Spotlight: Ilumina Sequencing. Illumina, Inc, 2010.

[27] V. Premachandran and R. Kakarala, "Measuring the effectiveness of bad pixel detection algorithms using the ROC curve", *IEEE Transactions on Consumer Electronics*, vol. 56, no. 4, pp. 2511–2519, Nov. 2010, Conference Name: IEEE Transactions on Consumer Electronics, ISSN: 1558-4127. DOI: 10.1109/TCE.2010.5681135.

[28] X. Zhang, T. F. Chan, ,UCLA Mathematics Department, Box 951555, Los Angeles, CA 90095-1555, and ,The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, "Wavelet inpainting by nonlocal total variation", *Inverse Problems & Imaging*, vol. 4, no. 1, pp. 191–210, 2010, ISSN: 1930-8345. DOI: 10.3934/ipi.2010.4.191. [Online]. Available: http://aimsciences.org//article/doi/10.3934/ipi.2010.4.191.

[29] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA", *Science*, p. 1 226 355, 2012, Publisher: American Association for the Advancement of Science.

[30] P. Patel, A. Prajapati, and S. Mishra, "Review of different inpainting algorithms", *International Journal of Computer Applications*, vol. 59, no. 18, pp. 30–34, Dec. 18, 2012, ISSN: 09758887. DOI: 10.5120/9650-4411. [Online]. Available: http://research.ijcaonline.org/volume59/number18/pxc3884411.pdf.

[31] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks", in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 341–349. [Online]. Available: http://papers.nips.cc/paper/4686-image-denoising-and-inpainting-with-deep-neural-networks.pdf.

[32] H. Zhang and S. Dai, "Image inpainting based on wavelet decomposition", *Procedia Engineering*, vol. 29, pp. 3674–3678, 2012, ISSN: 18777058. DOI: 10.1016/j.proeng.2012.01.551. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1877705812005619.

[33] T. Buschmann and L. V. Bystrykh, "Levenshtein error-correcting barcodes for multiplexed DNA sequencing", *BMC bioinformatics*, vol. 14, no. 1, p. 272, 2013, Publisher: BioMed Central.

[34] 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, Publisher: Nature Publishing Group.

[35] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median", *Journal of Experimental Social Psychology*, vol. 49, no. 4, pp. 764–766, Jul. 2013, ISSN: 00221031. DOI: 10.1016/j.jesp.2013.03.013. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0022103113000668.

[36] A. Mosleh, N. Bouguila, and A. B. Hamza, "Automatic inpainting scheme for video text detection and removal", *IEEE Transactions on Image Processing*, vol. 22, no. 11, pp. 4460–4472, Nov. 2013, Conference Name: IEEE Transactions on Image Processing, ISSN: 1941-0042. DOI: 10.1109/TIP.2013.2273672.

[37] N. Batool and R. Chellappa, "Detection and inpainting of facial wrinkles using texture orientation fields and markov random field modeling", *IEEE Transactions on Image Processing*, vol. 23, no. 9, pp. 3773–3788, Sep. 2014, Conference Name: IEEE Transactions on Image Processing, ISSN: 1941-0042. DOI: 10.1109/TIP.2014.2332401.

[38] C. Guillemot and O. Le Meur, "Image inpainting : Overview and recent advances", *IEEE Signal Processing Magazine*, vol. 31, no. 1, pp. 127–144, Jan. 2014, ISSN: 1053-5888. DOI: 10.1109/MSP.2013.2273004. [Online]. Available: http://ieeexplore.ieee.org/document/6678248/.

[39] D. Jerdev, "Correction of cluster defects in imagers", U.S. Patent 8817135B2, Aug. 26, 2014. [Online]. Available: https://patents.google.com/patent/US8817135B2/en.

[40] R. Vreja and R. Brad, "Image inpainting methods evaluation and improvement", *The Scientific World Journal*, vol. 2014, pp. 1–11, 2014, ISSN: 2356-6140, 1537-744X. DOI: 10.1155/2014/937845. [Online]. Available: http://www.hindawi.com/journals/tswj/2014/937845/.

[41] 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, Publisher: Wiley Online Library.

[42] I. Illumina, "An introduction to next-generation sequencing technology", 2015.

[43] 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, Publisher: Nature Publishing Group.

[44] 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, Publisher: ACM.

[45] Y. Erlich and D. Zielinski, "Capacity-approaching DNA storage", *bioRxiv*, p. 074 237, 2016, Publisher: Cold Spring Harbor Laboratory.

[46] A. Extance, "How DNA could store all the world's data", *Nature*, vol. 537, no. 7618, 2016, Publisher: Nature Publishing Group.

[47] M. Jain, H. E. Olsen, B. Paten, and M. Akeson, "The oxford nanopore MinION: Delivery of nanopore sequencing to the genomics community", *Genome biology*, vol. 17, no. 1, p. 239, 2016, Publisher: Springer.

[48] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting", presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2536–2544. [Online]. Available: `https://openaccess.thecvf.com/content_cvpr_2016/html/Pathak_Context_Encoders_Feature_CVPR_2016_paper.html`.

[49] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. P. Patel, A. Tiwari, M. J. Er, W. Ding, and C.-T. Lin, "A review of clustering techniques and developments", *Neurocomputing*, vol. 267, pp. 664–681, Dec. 6, 2017, ISSN: 0925-2312. DOI: `10.1016/j.neucom.2017.06.053`. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S0925231217311815`.

[50] S. M. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage", *Scientific Reports*, vol. 7, no. 1, p. 5011, Jul. 10, 2017, Number: 1 Publisher: Nature Publishing Group, ISSN: 2045-2322. DOI: `10.1038/s41598-017-05188-1`. [Online]. Available: `https://www.nature.com/articles/s41598-017-05188-1`.

[51] M. Dimopoulou, M. Antonini, P. Barbry, and R. Appuswamy, "DNA coding for image storage using image compression techniques", in *CORESA 2018*, 2018.

[52] J. A. Hawkins, S. K. Jones, I. J. Finkelstein, and W. H. Press, "Error-correcting DNA barcodes for high-throughput sequencing", *bioRxiv*, p. 315 002, 2018, Publisher: Cold Spring Harbor Laboratory.

[53] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions", *arXiv:1804.07723 [cs]*, Dec. 15, 2018. arXiv: 1804.07723. [Online]. Available: http://arxiv.org/abs/1804.07723.

[54] M. Dimopoulou, M. Antonini, P. Barbry, and R. Appuswamy, "A biologically constrained encoding solution for long-term storage of images onto synthetic DNA", in *EUSIPCO*, 2019.

[55] P. Spealman, J. Burrell, and D. Gresham, "Nanopore sequencing undergoes catastrophic sequence failure at inverted duplicated DNA sequences", Genomics, preprint, Nov. 23, 2019. DOI: 10.1101/852665. [Online]. Available: http://biorxiv.org/lookup/doi/10.1101/852665.

[56] 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, p. 4998, Mar. 21, 2019, Number: 1 Publisher: Nature Publishing Group, ISSN: 2045-2322. DOI: 10.1038/s41598-019-41228-8. [Online]. Available: https://www.nature.com/articles/s41598-019-41228-8.

[57] "A quaternary code mapping resistant to the sequencing noise for DNA image coding", in *MMSP (submitted)*, 2020.

[58] C. Pan, S. M. H. T. Yazdi, S. K. Tabatabaei, A. G. Hernandez, C. Schroeder, and O. Milenkovic, "Image processing in DNA", in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 8831–8835.

[59] O. N. Technologies, "New research algorithms yield accuracy gains for nanopore sequencing", 2020.

[60] 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, Publisher: Frontiers.

[61] (). An algorithm for finding best matches in logarithmic expected time | ACM transactions on mathematical software, [Online]. Available: https://dl.acm.org/doi/abs/10.1145/355744.355745.

[62] (). Bad pixel detection, [Online]. Available: https://www.photonstophotos.net/GeneralTopics/Sensors_&_Raw/Bad_Pixel_Detection.htm.

[63] (). Consensus sequence - an overview | ScienceDirect topics, [Online]. Available: https://www.sciencedirect.com/topics/biochemistry-genetics-and-molecular-biology/consensus-sequence.

[64] (). MediaCoding members, [Online]. Available: https://mediacoding.i3s.unice.fr/index.php/fr/membres.

[65] (). Oligo pools - custom CRISPR libraries | twist bioscience, [Online]. Available: https://www.twistbioscience.com/products/oligopools.

[66] S. Petrovic, "A comparison between the silhouette index and the davies-bouldin index in labelling IDS clusters", p. 12,

[67] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques", p. 20,

[68] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, "Constrained k-means clustering with background knowledge", p. 8,