JPEG-based image coding solution for data storage on DNA

Athul V
2018B1A30860H
Dr. Sudha Radhika
Digital Image Processing
29/04/2022

Abstract

I came across this project on DNA based storage, I thought it would be the ideal for long term storage needs. Today's harddisks have a storage life of no longer than 10-20 years.And recent studies have shown that DNA is a promising future for long term storage devices. I would like to look into making a simulator and looking into the efficiency of JPEG based DNA storage of Data. The project would involve applying conventional JPEG algorithms combined with DNA compression algorithms to achieve an efficient model.

Introduction

Generally speaking, an image of 1280x1024 Each pixel requires about 3 pixels locations to store RGB Colors.So 3 Blocks of 1280x1024 = 3932160 memory location. Which is redundant and excessive. So we have to look into the means of compression.

There are two types of compression - Lossy and Lossless, Lossless were inefficiently coded wherein we get the original image back when decompressed. Basically Lossless has a lot of processing time but less compression ratios.Lossy Compression on the other hand involves compression of the images exploiting the human strengths and weakness on understanding certain hues, shades and colors. For this exercise, I have chosen the tried and tested JPEG format.

JPEG

Jpeg is not really a single algorithm but rather a tool-kit to suit the needs of the user.
Jpeg uses lossy compression, where certain data are unrecoverable. It destroys them for the trade off of superior compression.
All this unwanted data is based on human nature’s inability to see and perceive color, hues and shades in its utmost precision.

Conventional use of Steps involved in JPEG

Convert RGB to YCbCr format (luminance & chroma)

Y′CbCr is used to separate out a luma signal (Y′) that can be stored with high resolution or transmitted at high bandwidth, and two chroma components (CB and CR) that can be bandwidth-reduced, subsampled, compressed, or otherwise treated separately for improved system efficiency.The luminance and chrominance components of the image are divided up into an array of 8x8 pixel blocks.Blocks on the right and bottom are full.

$$\begin{cases} Y = 16 + \frac{65.738R}{256} + \frac{129.057G}{256} + \frac{25.064B}{256} & (1) \\ Cb = 128 - \frac{37.945R}{256} - \frac{74.494G}{256} + \frac{112.439B}{256} & (2) \\ Cr = 128 + \frac{112.439R}{256} - \frac{94.154G}{256} - \frac{18.285B}{256} & (3) \end{cases}$$

Discrete Cosine Transform

This 8x8 block is put into process that performs forward DCT
The output of this is 64 values. DCT is only the real part of Fourier Transform.
Here I am using 1D Forward DCT. Here being u = intensities

$$F(u) = \sqrt{\frac{2}{n}} \quad C(u)\sum_{x=0}^{n-1} I(x)\cos\frac{(2x+1)\mu\pi}{2n}, u = 0...n-1$$

$$\text{where} \quad C(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for} \quad u = 0, \\ 1 & \text{otherwise} \end{cases}$$

Why DCT over DFT

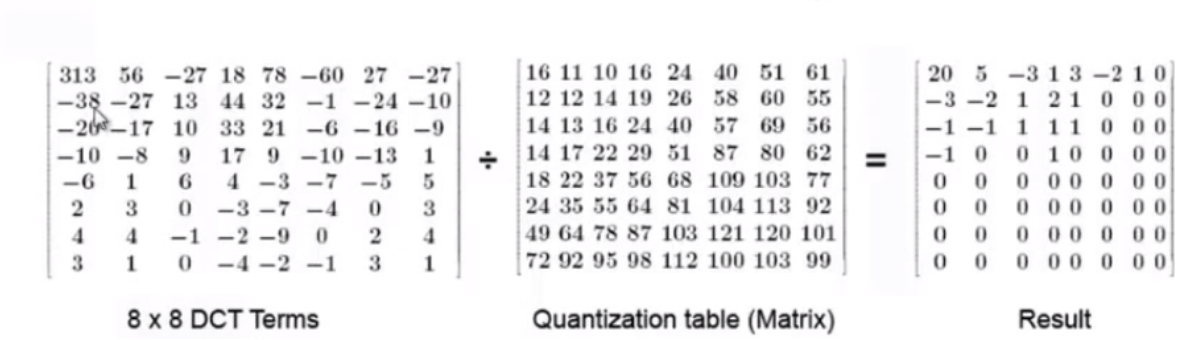
The property of the DCT that makes it quite suitable for compression is its high degree of "spectral compaction;" at a qualitative level, a signal's DCT representation tends to have more of its energy concentrated in a small number of coefficients when compared to other transforms like the DFT.

Quantization Table

Each DCT team is divided by the corresponding position in the Quantization table and then rounded to the nearest integer. High frequency components of an image are less important for the human perception of an image and thus can be discarded in order to save space. The mask Matrix

selects which DCT coefficients are kept and which ones get discarded in order to save space.

This looks like a variation of the quantization matrix. Low frequencies are in top left, high frequencies are in bottom right. Eye is more sensitive to low frequencies, so removal of high frequency coefficients will remove less important details of the image.



But we can use an approximation here, with lower freq being 1 and higher freq deleted off as zero.

```
mask = [1   1   1   1   0   0   0   0
        1   1   1   0   0   0   0   0
        1   1   0   0   0   0   0   0
        1   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0];
```

Quantisation

So, we don't want to compress the floating point data. This would bloat our stream, and not be effective. To that end, we'd like to find a way to convert the weights-matrix back to values in the space of [0,255]

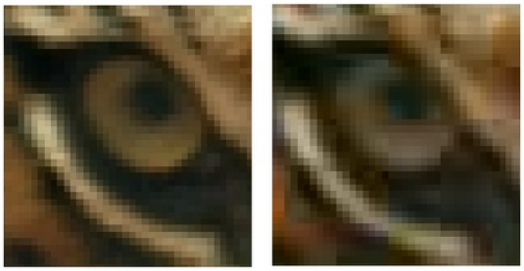
Compression

We are in the world of integers now Are read in a ZigZag Manner, i.e Jpeg Huffman.The result of our luma matrix.Once the data is in this format, the next steps are straightforward : execute RLE on the sequence, and then apply some statistical encoder (Huffman / Arithmetic / ANS) on the results.Your block is now JPEG encoded.

Conclusion on JPEG

Getting the quality value right, per image, is important to find the tradeoff between visual quality and file size.

Since this process is block-based, artifacts will tend to occur in blockiness, or “ringing”



original image

lossy JPEG format
with "artifacts"

Since processed blocks don’t intermingle with each other, JPG generally ignores the opportunity to compress large swaths of similar blocks together. Addressing that concern is something the WebP format is good at doing.

Converting Each JPEG Encoded Matrix into Nucleotide Sequences

Used the bioinformatics module in Matlab , int2nb to convert each array of JPEG Encoded matrix into a string of Nucleotide Sequences of adenine (A), thymine (T), guanine (G), and cytosine (C). Each A T G C, can be artificially synthesized by various biological methods and stored. This can be read by nanopore sequencing.
Given below is the table of converting each rounded up int to bases.

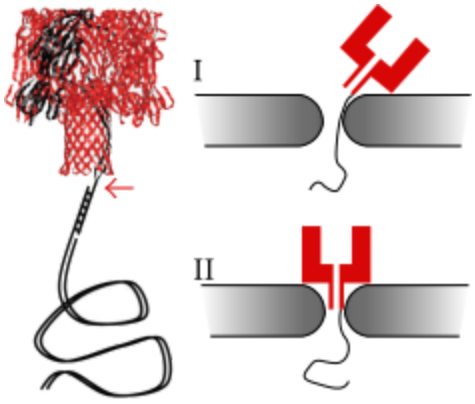
Mapping Nucleotide Integers to Letter Codes

Nucleotide	Integer	Code
Adenosine	1	A
Cytidine	2	C
Guanine	3	G
Thymidine	4	T
Uridine (if 'Alphabet' set to 'RNA')	4	U
Purine (A or G)	5	R
Pyrimidine (T or C)	6	Y
Keto (G or T)	7	K
Amino (A or C)	8	M
Strong interaction (3 H bonds) (G or C)	9	S
Weak interaction (2 H bonds) (A or T)	10	W
Not A (C or G or T)	11	B
Not C (A or G or T)	12	D
Not G (A or C or T)	13	H
Not T or U (A or C or G)	14	V
Any nucleotide (A or C or G or T or U)	15	N
Gap of indeterminate length	16	-
Unknown (any integer not in table)	0 or ≥ 17	* (default)

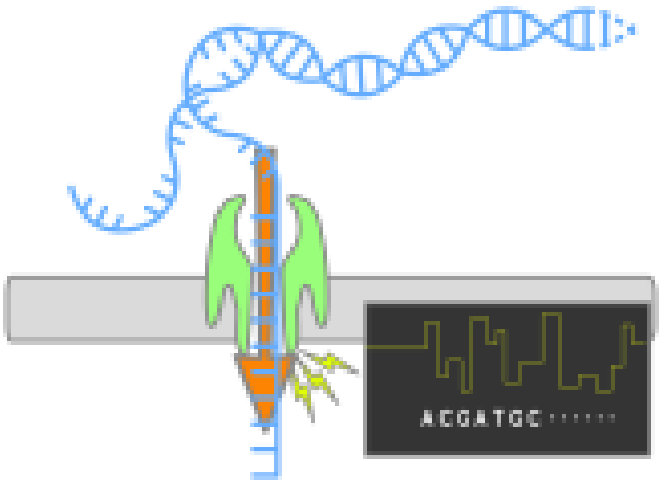
Nanopore Sequencing

Nanopore sequencing is a third generation approach used in the sequencing of biopolymers-specifically, polynucleotides in the form of DNA or RNA.

Using nanopore sequencing, a single molecule of DNA or RNA can be sequenced without the need for PCR amplification or chemical labeling of the sample. Nanopore sequencing has the potential to offer relatively low-cost genotyping, high mobility for testing, and rapid processing of samples with the ability to display results in real-time.



Each nanopore reads about 500 bases/second.

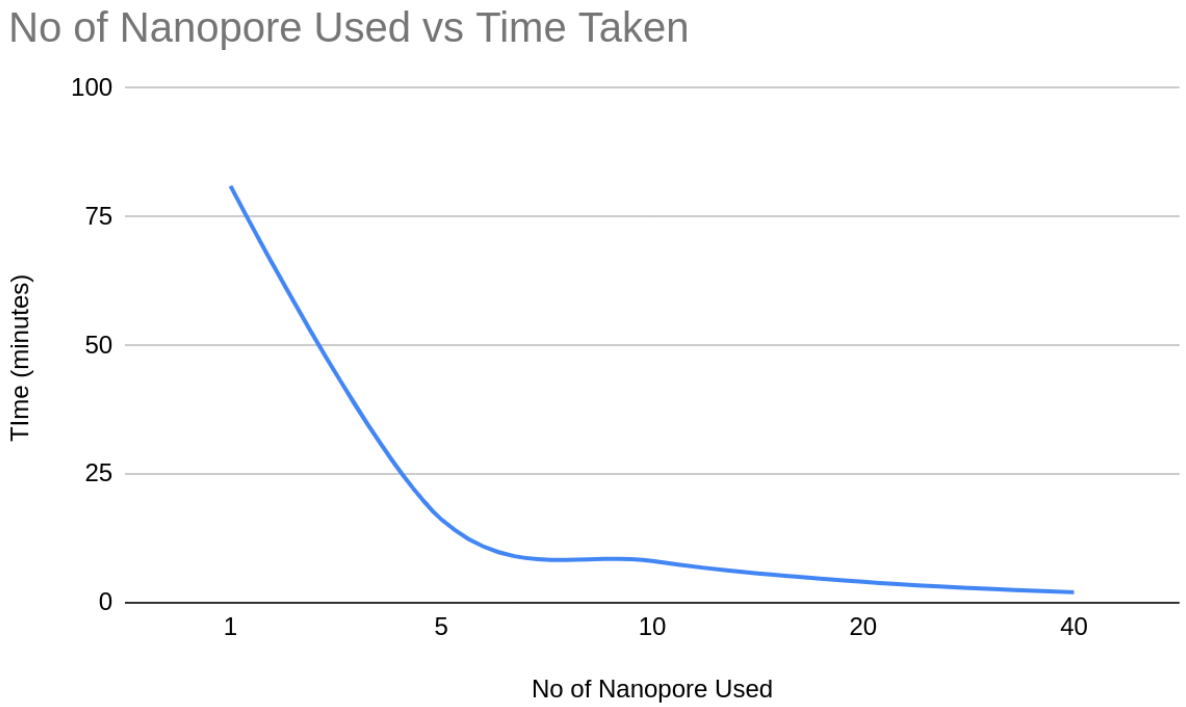


Storage Limits

	Hard Disk	Flash Memory	Bacterial DNA
Read-Write Speed us/bits	~3000-5000	100	<100
Data Retention years	>10	>10	>100
Power Usage gigabyte	~0.04	~0.01-0.04	<10^-10
Data Density bits/cm^3	~10^13	~10^16	~10^19

Conclusion

On varying the no of nanopore used we can vary the speed (decrease it), thus helping us find the optimal no of Nanopore to be used.



Code

The code done with Matlab, outputs three txt files of Nucleotide bases, which can be inserted into an external simulator when needed for analysis. It also outputs the compressed and original images and saves them to the directory. The command line output gives an approximate stimulation of the time taken vs the nanopore used along with the compression ratio as shown below.

```
The original is 527.426758 and compressed is 391.655273

The time taken for 1 Nanopore Sequencer to Reading at 500 bases/second is approx : 81.006567 minutes
The time taken for 5 Nanopore Sequencer to Reading at 500 bases/second is approx : 16.201313 minutes
The time taken for 10 Nanopore Sequencer to Reading at 500 bases/second is approx : 8.100657 minutes
The time taken for 20 Nanopore Sequencer to Reading at 500 bases/second is approx : 4.050328 minutes
The time taken for 40 Nanopore Sequencer to Reading at 500 bases/second is approx : 2.025164 minutes
```

Matlab Code

https://pastebin.com/AWKx5U2p

The images used for sample simulation and output txt files are attached in the folder.