Exploring the relevance of Mathematical Transforms using Image Analysis

Dhayanithi R, Rohit Narayanan, D.Sendil Vadivu, Dr. Narendran Rajagopalan
National Institute of Technology, Puducherry
dhayanithiramaraju@gmail.com,rohitnarayanan1729@gmailcom

ABSTRACT

This research project is in connection to computer science and mathematics. It focuses on learning and comparing various FFT algorithms and paving a path to analysing the theoretical scope of Ramanujan Fourier Transforms, along with examining the complexities of the same.

KEYWORDS

Fast Fourier Transforms, Cooley Tukey Algorithm, Rader Algorithm, Brunn's Algorithm, Winograd Algorithm, Order, Euler's totient function, Primitive roots, Mobius function , Ramanujan Sum, Ramanujan Matrix

1 INTRODUCTION

In this venture, different Fast Fourier Transform algorithms will be visited. The utilization of Fourier transforms is a numerical activity that converts from time domain to frequency domain. These procedures are generally utilized in electrical and electronics communication, however enjoy being a helpful benefit and can be applied here for image analysis. The benefit is that by being in the frequency domain, the picture can be separated into cosine and sine parts considering simply avoiding such parts, resulting in a compressed picture.

Fourier procedures permit us to disintegrate a picture in the frequency realm, eliminate or adjust specific frequencies and afterward recreate the picture. By eliminating specific frequencies, the compressed picture contains less data. Through the decision of which frequencies are taken out, the packed picture can cause very less distortion despite the fact that a large part of the basic data is disposed of.

A Fast Fourier Transform (FFT) is a group of calculations used to figure a DFT or IDFT of an array of numbers. This succession is generally a sign. Figuring DFT using its customary definition can consume a large chunk of the day. IF there are N elements in the sequence, the quantity of tasks performed through DFT will be N2. As N builds, the ideal opportunity for calculation increments separately. This is where FFT comes into play. By factorising the Discrete Fourier Transform framework into a result of sparse factors, an FFT quickly detects such changes. Because the dramatic term in DFT is intermittent, FFT computations take use of its balances. FFT reduced the N2 duties to NlogN activities, which can have a tremendous impact on computation speed as our N increases. FFT computations are based on a wide range of distributed hypotheses, ranging from fundamental complicated number-crunching to bunch hypothesis and number hypothesis.

National Institute of Technology-Puducherry,
2021.

2 BACKGROUND

The FFT compression implemented works on the basics of compression. The calculation is changing over a picture framework of 3 aspects (RGB) into the frequency side with FFT. To eliminate low signals, a limit is utilized in the calculation which is given by the client and in light of that, we are sifting Signs over the limit, making it a high pass channel. This is done because of the reality that the human vision is simply ready to see frequencies over a specific edge, and using this hypothesis, low frequencies can be eliminated from the picture while keeping the vast majority of the subtleties unblemished. In the wake of sifting through the low frequencies, the picture is changed once again into its unique area with IFFT.

3 VARIOUS FFT ALGORITHMS

There are several Fast Fourier Transforms(FFT) that we use in our day to day life.Here in this paper we will see few FFT algorithms and their computaions ,their disadvantages etc.

3.1 Cooley-Tukey Algorithm

Cooley-Tukey was first published in 1965. At the time, it was the most used FFT algorithm. The primary principle of the technique is to break an N-point DFT into M, N/M point DFTs [7, 12], thus if M=2, it will be split into two N/2 DFTs. The radix-2 is the name for this group. Radix-4, 8, 16, and so on are also available. Although the notion is recursive, most conventional implementations rearrange the technique in order to avoid explicit recurrence. The Cooley–Tukey algorithm divides the DFT into smaller components, allowing it to be used with any other DFT technique.

Read an image, and convert to grayscale. Resulting 2D matrix is flattened into a single dimensional array. Pad zeroes at the end of the flattened array, so as to make the length of the resulting array, a power of 2 (Zero-padding). Apply Fast Fourier transform on the array. Choose a suitable compression value C. A larger C value corresponds to a more compression, and hence more loss. Find maximum absolute value from the transformed array, and multiply C to get threshold value. Compare each value of the transformed array to the threshold value, and update values which don't meet the condition, to zero. Apply inverse Fourier transforms and convert the complex values to real values. Remove padding of zeroes. Resulting array is the inversed transform array. Reshape the 1D array, back into the 2D array, with the original dimensions, to get the final image.

Multiplications of complex numbers by solely real or imaginary numbers are used instead of complex multiplications. It's done by performing an N-point DFT with N=2t.

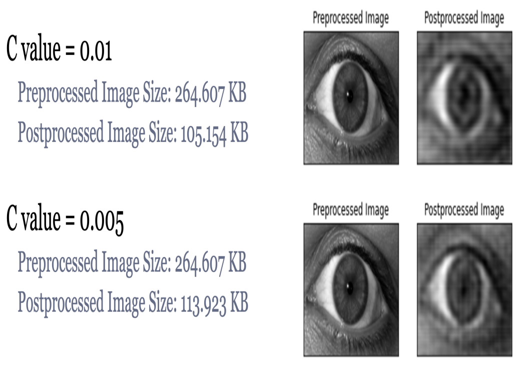

C value = 0.01
Preprocessed Image Size: 264.607 KB
Postprocessed Image Size: 105.154 KB
Preprocessed Image
Postprocessed Image
C value = 0.005
Preprocessed Image Size: 264.607 KB
Postprocessed Image Size: 113.923 KB
Preprocessed Image
Postprocessed Image

Figure 1: Cooley Tukey

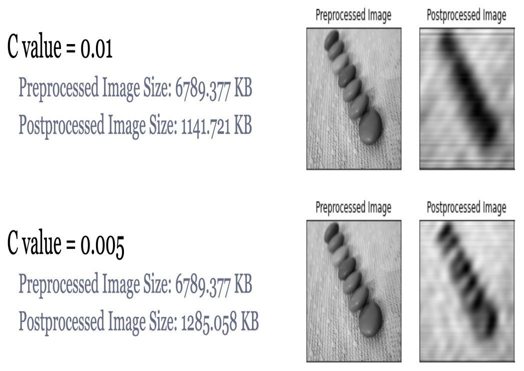

C value = 0.01
Preprocessed Image Size: 6789.377 KB
Postprocessed Image Size: 1141.721 KB
Preprocessed Image
Postprocessed Image
C value = 0.005
Preprocessed Image Size: 6789.377 KB
Postprocessed Image Size: 1285.058 KB
Preprocessed Image
Postprocessed Image

Figure 2: Winograd

$$X_k = \sum_{m=0}^{\frac{N}{2-1}} x_{2m} e^{\frac{-2\pi i}{N}(2m)k} + \sum_{m=0}^{\frac{N}{2-1}} x_{2m} + 1 e^{\frac{-2\pi i}{N}(2m+1)k}$$

$$X_k = \overbrace{\sum_{m=0}^{\frac{N}{2-1}} x_{2m} e^{\frac{-2\pi i}{N/2}(m)k}}^{\text{DFT of even-indexed part of } x_n} + e^{\frac{-2\pi i}{N}k} \overbrace{\sum_{m=0}^{\frac{N}{2-1}} x_{2m+1} e^{\frac{-2\pi i}{N/2}mk}}^{\text{DFT of odd-indexed part of } x_n}$$

$$X_k = E_k e^{\frac{-2\pi k}{N}} O_k$$

$$X_{k+\frac{N}{2}} = \sum_{m=0}^{\frac{N}{2}-1} x_{2m} e^{\frac{-2\pi i}{N/2}} m(k+\frac{N}{2}) + e^{\frac{-2\pi i}{N}(k+\frac{N}{2})} \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{\frac{-2\pi i}{n/2} m(k+\frac{N}{2})}$$

$$X_{k+\frac{N}{2}} = \sum_{m=0}^{\frac{N}{2}-1} x_{2m} e^{\frac{-2\pi i}{N/2}^{mk}} e^{-2\pi mi} + e^{\frac{-2\pi i}{N}k} e^{-\pi i} \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{\frac{-2\pi i}{N/2}mk} e^{-2\pi mi}$$

$$X_{k+\frac{N}{2}} = \sum_{m=0}^{\frac{N}{2}} x_{2m} e^{\frac{-2\pi i}{N/2}mk} - e^{\frac{-2\pi i}{N}k} \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{\frac{-2\pi i}{N/2}mk}$$

$$X_{k+\frac{N}{2}} = E_k - e^{\frac{-2\pi i}{N}k} O_k$$

$$X_k = E_k e^{\frac{-2\pi k}{N}} O_k$$

$$X_{k+\frac{N}{2}} = E_k - e^{\frac{-2\pi i}{N}k} O_k$$

3.2 Rader-Brenner Algorithm

Complex multiplications are substituted by multiplication of a complex number by a purely real or imaginary number in this case. It is achieved by doing an N-point DFT with N=2t.

$$X_0 = \sum_{n=0}^{N-1} x_n$$

$$X_{g^{-p}} = x_0 + \sum_{q=0}^{N-2} x_{g^q} e^{\frac{-2\pi i}{N} g^{-(p-q)}}$$

$$a_q = x_{g^q}$$

$$b_q = e^{\frac{-2\pi i}{N} g^{-q}}$$

3.3 Brunn's Algorithm

Bruun's algorithm is a fast Fourier transform (FFT) approach that employs a novel recursive polynomial-factorization method. In 1978, G. Bruun introduced it for powers of two. Until the final computing stage, it uses only real co-efficients in its operation. Because of this, it was first proposed as a method for computing the discrete Fourier transform (DFT) of actual data quickly..

$$X_k = \sum_{n=0}^{N-1} e^{\frac{-2\pi i}{N}} nk$$

$$\omega_N^n = e^{\frac{-2\pi i}{N} n}$$

$$x(z) = \sum_{n=0}^{N-1} x_n z^n$$

$$X_k = x(\omega_N^k) = x(z) mod(z - \omega_N^k)$$

3.4 Winograd Algorithm

The basic idea behind this method is that it factors ZN -1 into a variety of polynomials with coefficients of 1, 0, or 1, requiring only a few multiplications to operate, allowing Winograd to be used to build minimal-multiplication FFTs. It's commonly used in the quest for efficient minor factor algorithms[8]. Winograd proved that the DFT may be calculated using just irrational multiplications, reducing the number of multiplications while sacrificing precision.

Because current hardware architecture is made up of multiplier blocks, this is not regarded a vulnerability thus far. It is commonly used in conjunction with Rader's algorithm.

3.5 Some drawbacks of the existing system

- Cooley Tukey Algorithm seems meaningless if N is a prime number.
- FFTs work with arrays of size 2n for which an array had to be padded with zeroes so as to make it to a power of 2.
- Coefficients of all terms ranging from i=1 to n had to be computed. Dealing with complex numbers is a hefty task.
- It also increases the complexity of evaluation and calculation. 2D arrays need to be flattened to 1D arrays before applying transforms, and later on, needs to be converted back to 2D array.
- Winograd algorithm is advantageous for small values of N, but as N becomes larger its complexity increases.

3.6 How are Transforms used in Image Analysis?

- Read an image, and convert to grayscale. Resulting 2D matrix is flattened into a single dimensional array.
- Pad zeroes at the end of the flattened array, so as to make the length of the resulting array, a power of 2 (Zero-padding).
- Apply Fast Fourier transform on the array.
- Choose a suitable compression value C. A larger C value corresponds to a more compression, and hence more loss.
- Find maximum absolute value from the transformed array, and multiply C to get threshold value.
- Compare each value of the transformed array to the threshold value, and update values which don't meet the condition, to zero.
- Apply inverse Fourier transforms and convert the complex values to real values. Remove padding of zeroes. Resulting array is the inversed transform array.
- Reshape the 1D array, back into the 2D array, with original dimensions, to get the final image.

4 RAMANUJAN TRANSFORM

The discrete and rapid Fourier transformations are appropriate for analysing periodic and quasiperiodic sequences. However, they are ineffective for detecting aperiodic sequence characteristics such as low-frequency noise. The Ramanujan-Fourier transform was used to see whether there were any hidden patterns in such sequences. The fundamental concept is to compare the observed patterns and features in experimental data to the properties of the Ramanujan-Fourier transformations of well-known arithmetic functions in number theory and determine whether there is any correspondence between the two sets of attributes.

4.1 Driving Factors leading to Ramanujan Fourier Transform

4.1.1 ***Order****.*

4.1.2 Euler's totient function.

4.1.3 Primitive roots.

4.1.4 Mobius function.

4.1.5 Ramanujan Sum.

4.1.6 Ramanujan Transform.

4.1.1 Order

Let n be a positive integer, and let $GCD(a, n) = 1$. The order of a modulo n, denoted by ordn(a), is the smallest positive integer m such that a 1(mod n).

$$Order_7(2) = 3$$
$$2^1 \equiv 2(mod)7$$

4.1.2 Euler's totient function

If p1,p2,…,pk are the distinct primes dividing n, then

If n is even number

$$\varphi(n) = n(1 - \frac{1}{p_1})(1 - \frac{1}{p_2})....(1 - \frac{1}{p_k})$$

If n is prime number

$$\varphi(n) = n - 1$$

Some Examples

$$\varphi(7) = 6$$
$$\varphi(6) = 6(1 - \frac{1}{2})(1 - \frac{1}{3}) = 2$$

4.1.3 Primitive Roots

Suppose p is a prime number and a is an integer such that p does not divide a. We say that a is a primitive root modulo p if $ord_p(a) = \varphi(p) = p - 1$.

Example :

$$ord_7(3) = 6.$$
$$\varphi(7) = 7 - 1 = 6$$

Hence 3 is a primitive root modulo 7.

There are $\varphi(\varphi(n))$ primitive roots modulo n.

4.1.4 Mobius function

$mu(n)$ is the sum of the primitive nth roots of unity for any positive integer n. It has values in the range of 1, 0, 1 depending on how n is factored into primes.

$$\mu(n) = 1$$

if n is a positive integer with an even number of prime components that is square-free

4.1.5 Ramanujan Sum

$$c_d(n) = \sum_{k=1}^{d} e^{\frac{j2\pi kn}{d}}$$

Few examples: $c_1(n) = 1$
$c_2(n) = 1, -1$
$c_3(n) = 2, -1, -1$
$c_4(n) = 2, 0, -2, 0$
$c_5(n) = 4, -1, -1, -1, -1$
$c_6(n) = 2, 1, -1, -2, -1, 1$
$c_7(n) = 2, -1, -1, -1, -1, -1$
$c_8(n) = 4, 0, 0, 0, -4, 0, 0, 0$

Ramanujan established the relation shown aside, between RS and Mobius function to prove that the results of RS are integer valued. Hardy gave another relation connecting RS, Mobius function and Euler's totient function.

$$c_d(n) = \sum_{q|(d,n)} \mu(\frac{d}{q})q$$

$$c_d(n) = \frac{\mu(m)\varphi(d)}{\varphi(m)}, m = \frac{d}{(d,n)}$$

4.1.6 Ramanujan Matrix

We have an implementation for the Ramanujan transform matrix, which is given below. For a 1D array X, the forward 1D transform is given by $Y = AX$. The inverse transform is given by $X = A^1 Y$.

$$A(q, j) = \frac{1}{\varphi(q)M} c_q(mod(j-1, q) + 1)$$

Ramanujan matrix computation

For 2D arrays, we have $Y = AXAT$. The extended form of the above equation is given below.

$$Y(p, q) = \frac{1}{\varphi(p)\varphi(q)} \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} x(m, n)c_p(m), c_q(n)$$

Ramanujan Transformation

5 RESULTS

Cooley Tukey continues to be the best algorithm in terms of complexity. As far as the open problem regarding the minimum complexity of FFT algorithms is concerned, it is safe to say that Cooley Tukey still takes the lead even when compared to Ramanujan FFT, although the order of complexities might be a tad bit less in the case of Ramanujan FFT.

RFT's major disadvantages include the fact that it can work only on square matrices in the case of 2D transformations, and that it needs very large matrices for 1D transformation, which reduces its effectivity. Thus we might need to take the data in patches by sliding through the matrix, which only increases the overload of the whole process, and one which is not required in algorithms like Cooley Tukey FFT.

Ramanujan Fourier Transforms when implemented gave us a complexity of O(P log P). But there is a catch in it. As we approached the Open problem in Computer Science which asked about the lower bound of any FFT complexity, and whether it can be faster than O(NlogN). In one way it is, due to all the advantages of RFT we have listed above. And of course, P<N, so in a way, it is faster, but when we express them in terms of an order, Normal FFTs and RFT have the same order, so as far as the open problem is concerned, we may reach the conclusion that it is faster in terms of factors like CPU utilization, device utilization, but in terms of the order, they are one and the same. Future works for the same can be carried out in finding certain loopholes in the method to achieve a faster algorithm in terms of complexity and order of complexity.

6 OBSERVATION

FFT's major disadvantages include the fact that it can work only on square matrices in the case of 2D transformations, and that it needs very large matrices for 1D transformation, which reduces its effectivity, even though the number of computations required for that is very less. Thus we might need to take the data in patches by sliding through the matrix, which only increases the overload of the whole process, and one which is not required in algorithms like Cooley Tukey FFT
Cooley Tukey continues to be the best algorithm in terms of complexity. As far as the open problem regarding the minimum complexity of FFT algorithms is concerned, it is safe to say that Cooley Tukey still takes the lead even when compared to Ramanujan FFT, although the order of complexities might be a tad bit less in the case of Ramanujan

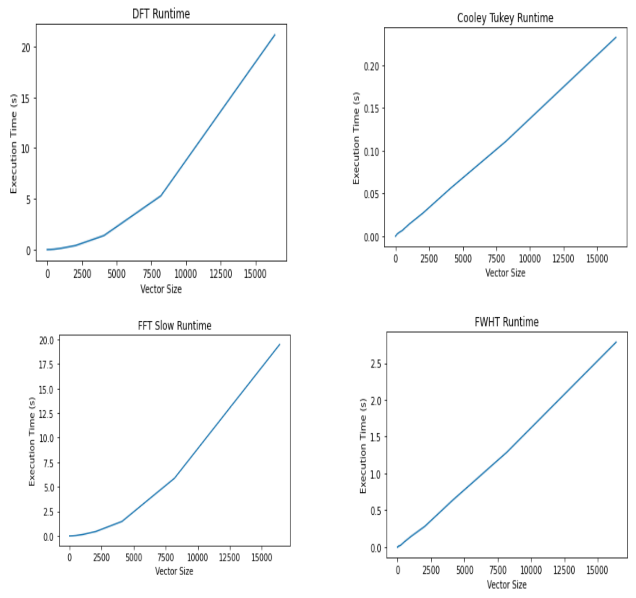
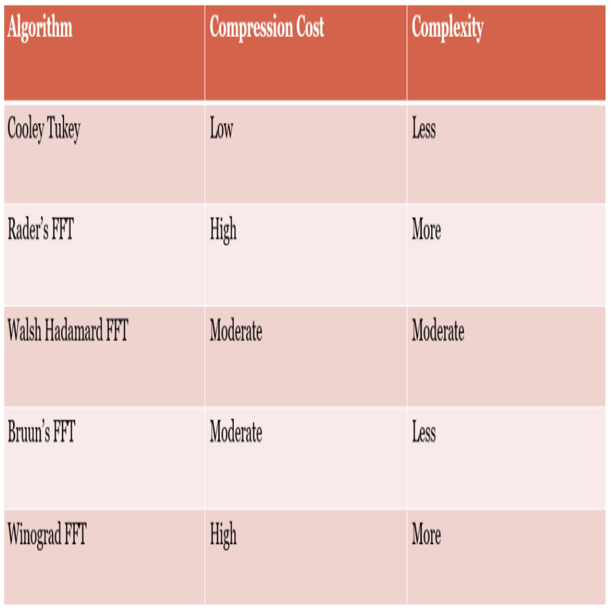


Figure 3: Runtime Assessments of various algorithms



Algorithm	Compression Cost	Complexity
Cooley Tukey	Low	Less
Rader's FFT	High	More
Walsh Hadamard FFT	Moderate	Moderate
Bruun's FFT	Moderate	Less
Winograd FFT	High	More

Figure 4: Comparison Summarization

REFERENCES

[1] En.wikipedia.org. . Fourier Transform. [online]. Available:https://en.wikipedia.org/wiki/Fast_Fourier_transform.

[2] Rasheed Mohammed H.etal.(2020).Image compression based on 2D Discrete Fourier Transform and matrix minimization algorithm.[Online]. Available: https://shura.shu.ac.uk/25961/11/Rodrigues_Image_Compression_Based%28VoR%29.pdf

[3] Pandey S.S. et.al.(2015).Image Transformation and Compression using Fourier Transformation.[Online]. Available:https://www.researchgate.net/publication/279979689_Image_Transformation_and_Compression_using_Fourier_Transformation

[4] Amente Bekele, (2016).Cooley-Tukey FFT Algorithm. [online]. Available: http://people.scs.carleton.ca/~maheshwa/courses/5703COMP/16Fall/FFT_Report.pdf

[5] dsp relaced.com. .Rader-Brenner FFT. [online]. Available:https://www.dsprelated.com/showthread/comp.dsp/40243-1.php

[6] Hari Krishna, (2017), (Digital Signal Processing Algorithms [Page 473-478]). [EBook]. Available:Kindle edition and Hardcover https://www.amazon.in/dp/B08R17D9WM?ref=KC_GS_GB_IN.

[7] cnx.org. Winograd's Short DFT Algorithms. [online]. Available:https://cnx.org/contents/Fujl6E8i@5.8:O4g83iRj@14/Winograd-s-Short-DFT-Algorithms

[8] Brad Osgaod, (2014). The Fourier Transforms and its Applications.[online]. Available:https://see.stanford.edu/course/ee261

[9] Javapoints. Digital Image Processing System. [Online]. Available:https://www.javatpoint.com/dip-image-transformations

[10] VN Krishnachandran, (2015). Ramanujan Computing Technology.[Online]. Available:https://www.researchgate.net/publication/350131841_Ramanujan_in_Computing_Technology

[11] MITopencourseware. Congruences mod Primes, Order, Primitive Roots.[online]. Available:https://ocw.mit.edu/courses/mathematics/18-781-theory-of-numbers-spring-2012/lecture-notes/MIT18_781S12_lec7.pdf

[12] RoyalSocietyPublishing. Srinivasa Ramanujan and signal proccessing problems. [online]. Availble:https://royalsocietypublishing.org/doi/10.1098/rsta.2018.0446

[13] IEEE. Matrix-Based Ramanujan-Sums Transforms. [Online]. Available:https://ieeexplore.ieee.org/document/6563143