RUN TIME ANALYSIS OF FAST FOURIER TRANSFORM AGAINST SLOW IMPLEMENTATION

CS 412 - ALGORITHMS, DESIGN & ANALYSIS

AKEEL ATHER MEDINA • HABIB SHEHZAD • TASMIYA MALIK

INTRODUCTION

Fourier Transform is a data manipulation technique that converts a dataset from one domain to another. For example, a data can be represented either in the time domain where we have the values F as some function of time *t*, i.e. F(*t*) or we can represent the data in the frequency domain, with amplitude G as a function of frequency *f*, i.e. G(*f*). Fourier Transform is a medium through which we can convert the data from time to frequency domain, and form a transform pair. For our example, the transform pair will be,

F(*t*) ⟷ G(*f*)

Our aim was to implement both naive and D&C techniques to analyze their run time.

WHAT ARE DFT AND FFT?

Discrete Fourier Transform is defining the Fourier transform on the discrete finite sequence. It maps N complex numbers from one domain, say time, to N complex numbers in another domain, say frequency. The discrete sum gives,

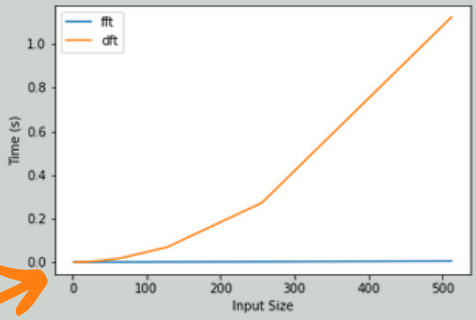
$$H_n = \sum_{k=0}^{N-1} h_k \, e^{2\pi i k n / N}$$

Formula application is called the slow or naïve approach, and it is expected to work in O(N²).

Fast Fourier Transform is a faster implementation of DFT using the divide-and-conquer technique in O(N.logN) time. The idea is to divide the sequence into two parts by even and odd indexes of the elements, and add them back with some twiddle factor W. The recurrence relation is identical to Mergesort: T(n) = 2T(n/2) + O(n)

RUN TIME ANALYSIS

By comparing both run time graphs, we can see that the D&C technique works way better than the naive implementation. We can only show the results for the sample size of 500 because the naive DFT takes more time and resources to compute which our machines could not provide.



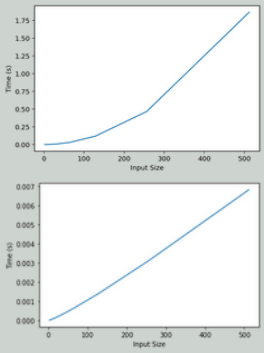
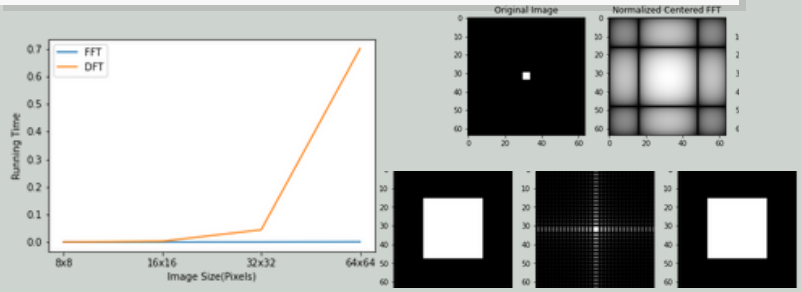
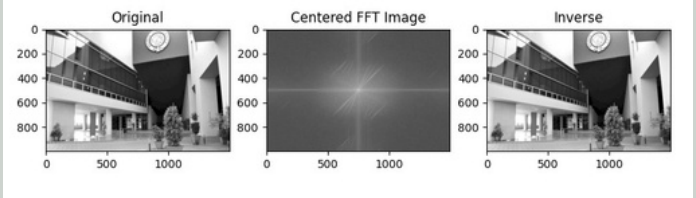


IMAGE COMPRESSION: DFT VS FFT

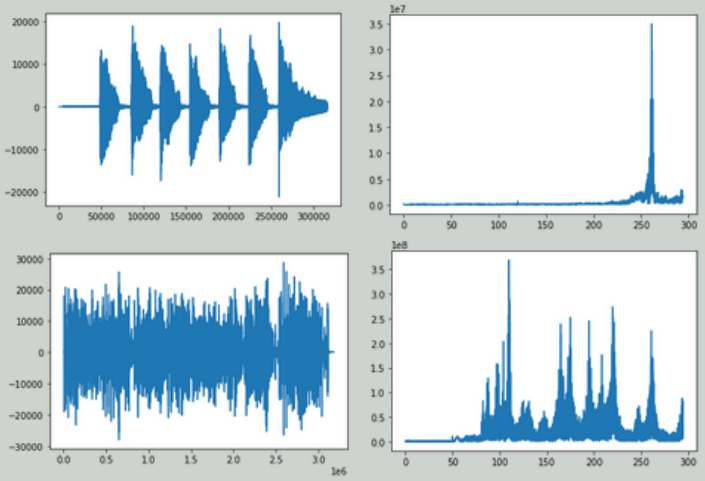
Our first application was based on image processing. We wanted to apply image compression using both naïve and D&C approaches. We compared the run time of DFT and FFT on various sizes of pixel arrays. We also applied FFT and inverse FFT to compress and recover the images. From the run time graphs we could see that FFT can compress larger images in lesser time than slow DFT. The run time graph goes steeper even for 16x16 images, while FFT could compress 128x128 pixel images in minimal time. The FFT image visualizations show the low frequencies in the center.





CONVERTING AUDIO FILES FROM TIME TO FREQUENCY DOMAIN

We also applied FFT on different audio files. The FFT took the audio signals in time domain, and converts it into frequency domain. We could not apply DFT on the files because of the immense amount of time it took to compute.



CHALLENGES/LIMITATIONS

A major issue that we faced was finding the dataset of the appropriate length. Our approach demanded a data set of size 2n, however, the audio files that we picked online were not guaranteed to be in the powers of 2, and could not be used in our implementation.

REFERENCES

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). Numerical recipes 3rd edition: The art of scientific computing. Cambridge university press.

Rao, K. R., Kim, D. N., & Hwang, J. J. (2010). Fast Fourier transform algorithms and applications (Vol. 32). Dordrecht: Springer.

Cochran, W. T., Cooley, J. W., Favin, D. L., Helms, H. D., Kaenel, R. A., Lang, W. W., ... & Welch, P. D. (1967). What is the fast Fourier transform?. Proceedings of the IEEE, 55(10), 1664-1674

tesfagabir, Digital Image Processing Implementations(2017), Github Repository. Digital-Image-Processing/02-Implementing-Fast-Fourier-Transform-Using-Python.ipynb at master · tesfagabir/Digital-Image-Processing (github.com)