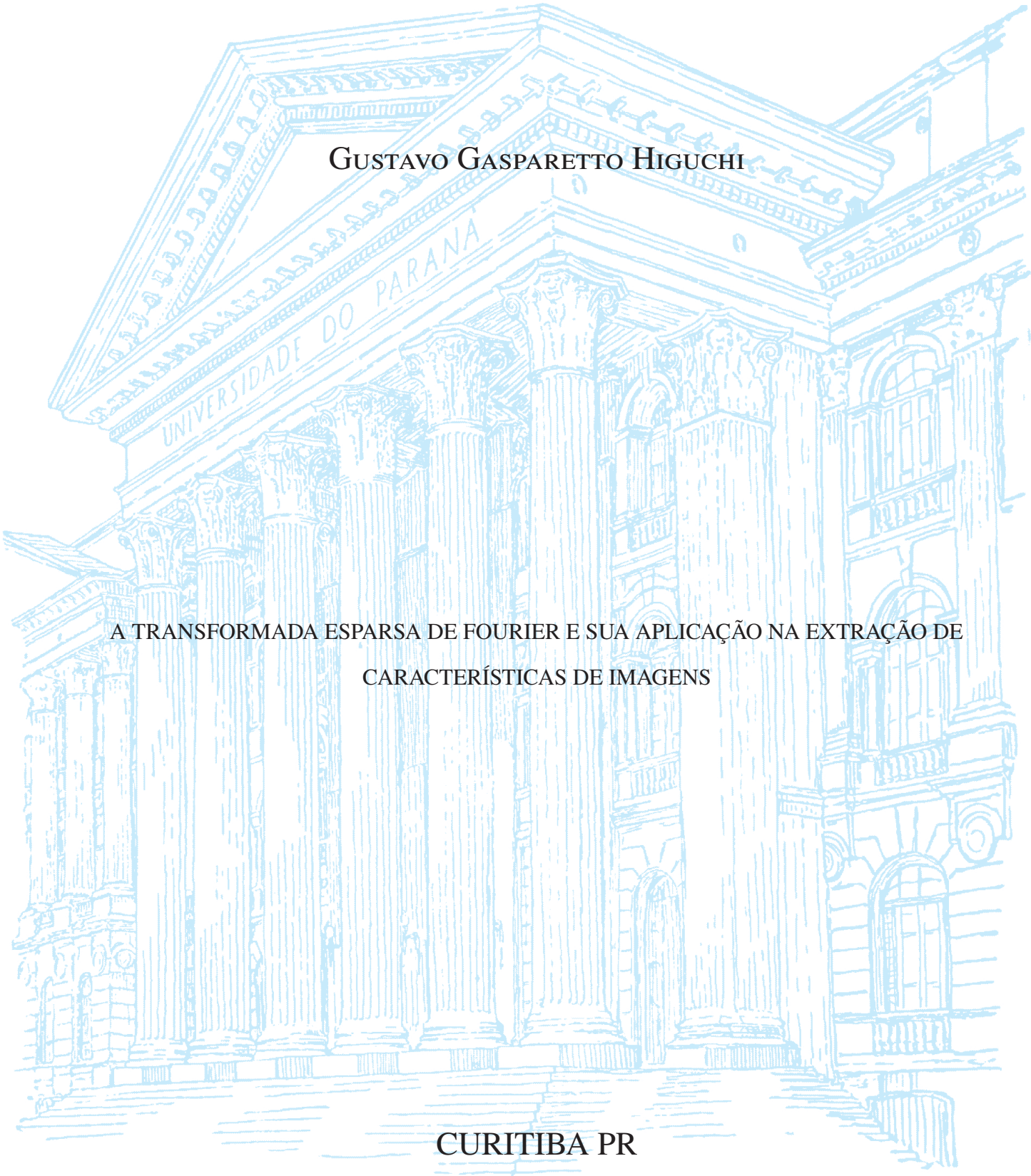
Universidade Federal do Paraná

Gustavo Gasparetto Higuchi



A TRANSFORMADA ESPARSA DE FOURIER E SUA APLICAÇÃO NA EXTRAÇÃO DE CARACTERÍSTICAS DE IMAGENS

CURITIBA PR
2018

Gustavo Gasparetto Higuchi

A TRANSFORMADA ESPARSA DE FOURIER E SUA APLICAÇÃO NA EXTRAÇÃO DE CARACTERÍSTICAS DE IMAGENS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Prof. André Vignatti.

CURITIBA PR
2018

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

H638t Higuchi, Gustavo Gasparetto

A transformada esparsa de Fourier e sua aplicação na extração de características de imagens [recurso eletrônico] / Gustavo Gasparetto Higuchi – Curitiba, 2018.

Dissertação (mestrado) - Universidade Federal do Paraná, Programa de Pós-Graduação em Informática, área de concentração em Ciência da Computação, setor de Ciências Exatas na Universidade Federal do Paraná.

Orientador: Prof. Dr. André Vignatti

1. Fourier, Transformações de. 2. Ciências Exatas. I. Universidade Federal do Paraná. II. Vignatti, André. III. Título.

CDD 515.2433

Bibliotecária: Vilma Machado CRB9/1563

MINISTÉRIO DA EDUCAÇÃO
SETOR SETOR DE CIENCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO INFORMÁTICA

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **GUSTAVO GASPARETTO HIGUCHI** intitulada: **A Transformada Esparsa de Fourier e sua Aplicação na Extração de Características de Imagens**, após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 28 de Novembro de 2018.

ANDRÉ LUÍS VIGNATTI
Presidente da Banca Examinadora (UFPR)

GUSTAVO ALBERTO GIMENEZ LUGO
Avaliador Externo (UTFPR)

EDUARDO TODT
Avaliador Interno (UFPR)

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
Setor de Ciências Exatas
Coordenação de Pós-Graduação em Informática

Aos meus amáveis pais...

Agradecimentos

Este trabalho é dedicado aos meus pais, que não só me apoiaram como também me incentivaram a continuar com meus estudos. Também não posso deixar de lado todas as pessoas do laboratório, inclusive meu orientador por conseguir concluir este trabalho.

RESUMO

Separar uma função complexa em uma série de funções mais simples simplificou o estudo de muitas área na ciência e na engenharia. Este processo ficou conhecido como *transformada de Fourier*, ou *transformada discreta de Fourier*. A transformada discreta de Fourier é computada, na forma mais direta, em $O(N^2)$ operações, onde N é o tamanho do sinal. Entretanto, desde a descoberta da *transformada rápida de Fourier*, um algoritmo que computa a transformada discreta de Fourier em $O(N \log N)$ operações, muitas aplicações surgiram e foi considerado um dos algoritmos mais importantes do século. A transformada rápida de Fourier é amplamente utilizada para processamento e análise de sinais digitais. Observou-se na prática que muitos sinais possuem um espectro com poucos coeficientes significativos. Estudos recentes apresentam algoritmos que fazem proveito dessa característica que sinais do mundo físico possuem para gerar um algoritmo de aproximação da tranformada de Fourier com complexidade de tempo sublinear ao tamanho do sinal. Estes algoritmos são chamados de *transformada esparsa de Fourier*. Este trabalho apresenta uma revisão de algoritmos dessa classe e, com mais detalhes, o primeiro algoritmo que superou, na prática, o tempo de execução da transformada rápida de Fourier para alguns sinais exatamente esparsos. Além disso, o trabalho também apresenta resultados de experimentos para determinar quão esparso o espectro de imagens são utilizando a definição de energia dada pelo Teorema de Parseval. Finalmente, o algoritmo da transformada esparsa é utilizado para recuperar os maiores coeficientes daquelas imagens diretamente.

Palavras-chave: transformada discreta de Fourier, transformada rápida de Fourier, transformada esparsa de Fourier, processamento de sinais digitais, sinais esparsos.

ABSTRACT

Separating a complex function into a series of simpler functions has improved studies in most areas in science and in engineering. This process became known as the *Fourier transform*, or *discrete Fourier transform*. The discrete Fourier transform can be computed, in the most direct way, with $O(N^2)$ operations, where N is the signal size. However, since the discovery of the *fast Fourier transform*, an algorithm that computes the discrete Fourier transform in $O(N \log N)$ operations, many applications became possible and the algorithm has been considered one of the most important algorithm of the century. The fast Fourier transform is widely used in digital signal processing and analysis. It was observed that signals in the physical world had spectra that only a few coefficients are significants. Recent studies presents algorithms that explores this characteristic of signals computes an approximation of the Fourier transform of these signal in a sublinear time complexity in the signals size. These algorithms are called *sparse Fourier transform*. This document presents an overview of algorithms of this class and with futher details, the first algorithm that outperforms the FFT in experimental results for some exactly sparse signals. Also, this work presents experiments for determining how sparse image spectra are by using the Parsevel's Theorem. Finaly, the sparse Fourier transform algorithm is used to recover the greatest coefficients of image spectra directly.

Keywords: discrete Fourier transform, fast Fourier transform, sparse Fourier transform, digital signal processing.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

f	Função ou sinal
$\hat{f}$	Transformada de Fourier da função f
N	Tamanho da função discreta
K	Quantidade de coeficientes significativos de um sinal
DFT	Transformada Discreta de Fourier
SFT	Transformada Esparsa de Fourier
T	Período de uma função periódica
i	Constante imaginária
v_f	Representação vetorial da função f
ω	Frequência de uma função trigonométrica
G	Filtro de janela
δ	Tolerancia do filtro
y	Sinal esparso mínimo

1 Introdução

Hoje, o aprendizado de máquina se mostra presente de várias formas. O objetivo deste trabalho é mostrar que o espectro de uma imagem possui grande parte de sua representação em apenas um subconjunto de seus valores. Portanto, este trabalho deseja responder à pergunta: o espectro de uma imagem pode ser utilizado como um vetor de características para um algoritmo de aprendizado de máquina?

A transformada discreta de Fourier é um processo que obtém o espectro de um sinal qualquer. Resultados apresentados neste trabalho mostram que podemos recuperar boa parte da imagem com apenas um subconjunto os valores do espectro de cada imagem. O tamanho desse subconjunto é muito menor que a própria imagem, geralmente, temos cerca de 10% do espectro que seja significativo em uma imagem 32x32, assim como para imagens maiores. Estas imagens são utilizadas para como conjunto de treinamento para algoritmos de aprendizado de máquina. Hoje, com imagens de alta definição, o processamento de milhares de imagens para treinamento é muito custoso. Portanto, sabendo que temos um subconjunto que representa boa parte da imagem, propomos utilizar um algoritmo sublinear no tamanho do sinal para obter diretamente esse subconjunto do espectro. Neste caso, basta dar uma representação ao espectro da imagem para se obter um vetor de característica dessa imagem e, assim, reduzir a complexidade de algoritmos de aprendizado de máquina.

Considerando um vetor de duas dimensões, podemos descrevê-lo com um par de números reais indicando que o vetor começa na origem e termina no par ordenado $[x, y]$. Quando nenhuma base é dada, consideramos que o vetor esteja na base chamada *canônica*. O par $[x, y]$ é a proporção dessas bases que, quando somadas, resulta no vetor que queremos. Na computação, uma função discreta $f(x)$ possui tamanho finito N. Por exemplo, um arquivo de áudio é uma função $f(t)$ em relação ao tempo t, e indica qual *voltagem* deve ser entregue às caixas de som no tempo t. É comum chamar o valor de $f(t)$ de *amostra* do áudio que, em algum momento, foi gravado. Sabendo disso, para comparar dois áudio, é necessário comparar a amostra de ambos os áudio para então dizer a diferenças e semelhanças. Sistemas de reconhecimento de áudio, como o *Shazam* [cWF03], conseguem obter um reconhecimento de até 15 segundos de amostras de uma música gravada em ambientes com bastante ruído além de usar um microfone de celular, sujeito a interrupção de conexão além de possível compressão do áudio pelo dispositivo. Ainda assim, é possível obter um reconhecimento rápido e com poucos *falso-positivos*. O reconhecimento do áudio é feito na *base de Fourier*, ou seja, utiliza um pré processamento aplicando *a transformada discreta de Fourier* nos dados obtidos.

A transformada de Fourier é, de forma simples, uma mudança de base. Dada uma função, obtém-se a mesma função na base de Fourier. Neste caso, estamos falando da transformada discreta de Fourier por se tratar de funções discretas, ou, como dito anteriormente, vetores. Colocando em formato de problema computacional, temos o Problema 1 abaixo.

Problema 1 *A transformada discreta de Fourier*

Entrada: Uma função discreta $f(x)$ de tamanho N, tal que $0 \leq x < N$

Retorno: Uma função discreta $\widehat{f}(k)$ de tamanho N, tal que $0 \leq k < N$ e seja a representação de $f(x)$ na base de Fourier

De um lado, temos uma função na base original com valores variados e diferentes de zeros. Mesmo que seja possível diferenciar uma função de outra, existe uma forma melhor para fazer essa diferenciação. Em alguns casos, quando obtemos a função na base de Fourier, a função passa a ter muitos valores nulos ou praticamente nulos. Por isso, dizemos que funções são *esparsas* na base de Fourier. Como temos a mesma função em bases diferentes, podemos tirar conclusões sobre a função em qualquer uma das bases. Em uma das bases temos uma função densa e na base de Fourier, temos uma função esparsa *i.e.*, com menos valores significativos. Portanto, computar análises de uma função na base de Fourier é vantajoso por se tratar de uma função com poucos valores significativos, *i.e.*, a função é *esparsa* na base de Fourier.

O resultado da tranformada de Fourier é um vetor esparso, ou seja, muitos valores insignificativos e nulos. Na última década, pesquisas voltadas para o desenvolvimento de algoritmos de aproximação que obtém apenas os valores significativos do vetor na base de Fourier tiveram grandes avanços [Man92, GGI+02, AGS03, ACG05, Aka10, Iwe10, HIKP12b, HIKP12a]. Conhecidos como *transformada esparsa de fourier*, estes algoritmos têm como objetivo atingir uma complexidade que seja proporcional à quantidade de valores significativos e não ao tamanho do vetor na base de Fourier, ou seja, devem possuir uma complexidade sublinear em relação ao tamanho da entrada. Adaptando o problema computacional, ficamos com o Problema 2.

Problema 2 *A transformada esparsa de Fourier*

Entrada: Uma função $f(x)$ discreta de tamanho N, tal que $0 \leq x < N$

Retorno: Uma função $\hat{f}'(k)$ de tamanho K, tal que $K = o(N)$, $0 \leq k < K$ e seja uma aproximação de $\hat{f}(x)$ na base de Fourier

Porém, muitos resultados não superam, na prática, o tempo de execução da *FFT* ou superam somente para vetores muito esparsos, $i.e.$, um K muito pequeno e um N muito grande ($K \leq 135$ e $N \approx 2^{22}$ [IGS07]). O primeiro algoritmo que superou, na prática, o tempo de execução da transformada rápida de Fourier foi apresentado por Hassanieh et al [HIKP12b]. O objetivo principal deste trabalho é mostrar como os algoritmos da transformada esparsa de Fourier funcionam e, com detalhes, o algoritmo de Hassanieh. Além disso, o presente trabalho pretende trazer resultados que podem indicar vantagens no uso da transformada esparsa para recuperar espectros aproximados de imagens.

1.1 Objetivo

Este trabalho tem como objetivo apresentar formas de computar a transformada discreta de Fourier, que é a principal ferramenta para análise de sinais digitais. Sabendo disso, também apresentamos uma classe de algoritmos para computar o espectro de um sinal que possui complexidade de tempo sublinear ao tamanho do sinal, a transformada esparsa de Fourier. A transformada esparsa de Fourier é uma classe de algoritmos aleatorizados que busca os coeficientes mais significativos do espectro de um sinal diretamente.

Além disso, apresentamos resultados de experimentos para determinar quão esparso é um espectro de uma imagem. Esses resultados podem indicar uma vantagem no uso de algoritmos de aprendizado de máquina, já que a abordagem padrão utiliza cada pixel da imagem como uma característica da imagem. Finalmente, considerar a imagem como um sinal de uma dimensão e

utilizar o algoritmo disponibilizado por Hassanieh et. al [**?**] para analisar o desempenho de uma recuperação aproximada dos maiores coeficientes de uma imagem. Portanto, neste trabalho, queremos mostrar se o espectro de uma imagem pode ser considerado *esparso* e se algoritmos que computem uma aproximação da DFT podem recuperar o espectro de uma imagem com um erro aceitável.

1.1.1 Objetivos específicos

- Apresentar algoritmos que computam a DFT.
- Descrever sobre algoritmos da classe da transformada esparsa de Fourier.
- Analisar com mais detalhes um algoritmo de aproximação da DFT.
- Executar e apresentar experimentos para determinar a esparsidade do espectro de imagens.
- Utilizar um algoritmo já implementado para recuperar o espectro de um sinal.

2 Notações e noções fundamentais

2.1 Sinais e domínios

Dizemos que uma função está no *domínio do tempo*, quando a variável está em relação ao tempo. Por exemplo, um áudio é uma função no domínio do tempo por ser uma função que determina qual voltagem deve ser entregue à uma saída de áudio em *um determinado instante no tempo*. De forma parecida, uma imagem está no *domínio do espaço* porque cada valor do vetor de imagem fazem referências espaciais. Ainda, um vídeo seria uma função no *domínio do tempo e do espaço*, por serem imagens que são apresentadas em um determinado instante no tempo. Por outro lado, quando analisamos estas funções na base de Fourier, é comum dizer que estamos analisando a função no *domínio da frequência*. A base de Fourier é formada de funções trigonométricas que diferem entre si pela frequência, por isso que é denominado domínio da frequência.

A análise de uma função no domínio da frequência é chamada de *análise de Fourier*. A análise de Fourier geralmente está ligada à área de processamento de sinais digitais e as funções discretas em questão são frequentemente chamadas de *sinais discretos*. Neste trabalho, faremos referência à uma função no domínio do tempo ou espaço como *sinais*.

Cada valor de uma função no domínio da frequência corresponde à magnitude de *coeficientes* que multiplicam cada função base e que, se somadas, resultam na função em sua base original. Na literatura, é comum fazer referência às funções no domínio da frequência como o *espectro* daquela função. Podemos assumir que a nomeclatura dada à função no domínio da frequência tenha origens no *latim* (*spectrum*), que significa aquilo que é incorpóreo ou invisível, então, como a função está em sua forma que não se vê. Portanto, o espectro de uma função é quando temos esta função no domínio da frequência.

2.2 Energia da função

Outro aspecto a ser definido é a energia de um sinal. Sinais no mundo físico geralmente estão relacionadas com movimento, como um sinal de áudio está relacionado com a propagação de energia entre moléculas do meio. Portanto, é comum haver comparações entre sinais utilizando a energia de cada um desses sinais.

O teorema de Parseval diz que a energia de um sinal é igual a energia do espectro deste sinal [PdC06]. De modo formal, o teorema de parseval é dado pelo Teorema 1.

Teorema 1. *Dado uma função $f(t)$ de período 2π tal que sua série de Fourier seja dada por*

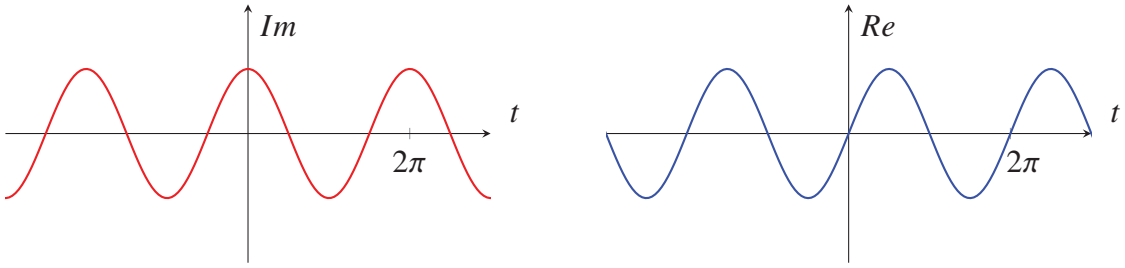
$$f(t) = \frac{1}{2}a_0 + \sum_{k=1}^{\infty} a_k cos(kt) + \sum_{k=1}^{\infty} b_k sen(kt) \tag{2.1}$$

Então vale a igualdade

$$\sum_{t=0}^{N-1} |f(t)|^2 = \sum_{k=0}^{N-1} |\widehat{f}(k)|^2 \tag{2.2}$$

2.3 Raízes complexas de unidade

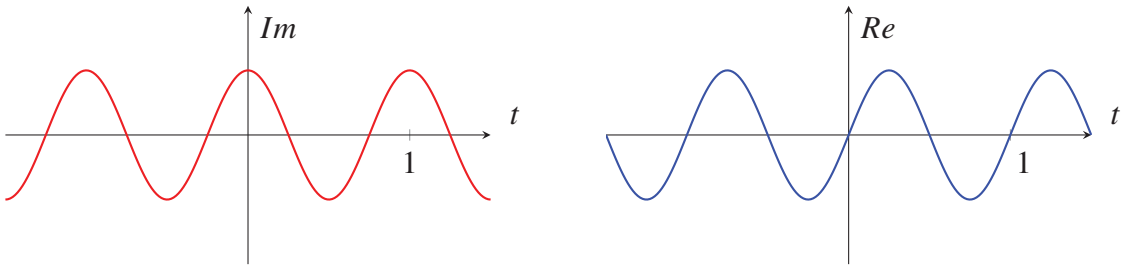
A *raíz complexa de unidade* é qualquer número complexo que elevado à um número inteiro positivo n seja igual a 1 [Lan93]. Considere o número complexo e^{it} que possui gráfico dado pela Figura 2.1. Considerando este número como uma função periódica de amplitude qualquer, temos a função periódica $f(t) = e^{it}$ com período $T = 2\pi$. Esta função é uma raiz de unidade para todo N múltiplo do período, *i.e.* $(e^{it})^{2\pi} = e^{i2\pi t} = 1$



(a) Parte real. (b) Parte imaginária.

Figura 2.1: O gráfico da função $f(x) = c \cdot e^{it}$.

Por questões de simplicidade, vamos considerar a mesma função periódica, porém com período $T = 1$. Assim, temos a função periódica $f(t) = e^{i2\pi t}$ com período $T = 1$ representado pela Figura 2.2. Nesta caso, a função é uma raiz de unidade para qualquer inteiro $N > 0$.



(a) Parte real. (b) Parte imaginária.

Figura 2.2: O gráfico da função $f(t) = c \cdot e^{i2\pi t}$.

Agora, considere a função $f(t) = e^{i2\pi t/N}$. Para um inteiro $N > 0$, dizemos que a função $f(t)$ é *N-ésima raíz de unidade*, porque será uma raiz de unidade apenas para múltiplos inteiros de N. Neste caso, temos uma função que possui N valores distintos uma vez que $f(1) = f(N+1)$. Para simplificar a notação, vamos denotar $\omega = e^{i2\pi/N}$. Por exemplo, para $N = 8$, temos uma função periódica com 8 valores distintos que podem ser distribuidos igualmente espaçados em um círculo, como está representado na Figura 2.3. Vale notar que $\omega^9 = \omega^1$. A N-ésima raiz de unidade também é conhecida como o *número de Moivre*.

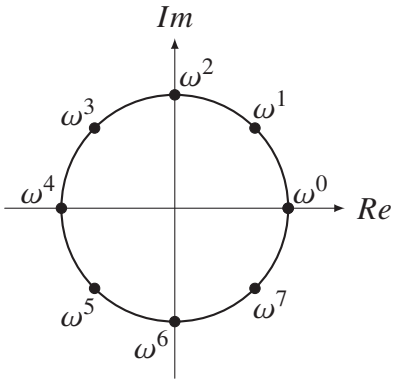


Figura 2.3: A oitava raiz de unidade no plano dos complexos.

Considerando que temos a igualdade $\omega^7 = \omega^{1-8} = \omega^{-1}$, podemos reescrever as raízes de unidade da forma que está representado na Figura 2.4. O valor ω^0 e o valor ω^4 são valores especiais, onde este é chamado de *frequência de Nyquist* e aquele é a *constante* [DdSN10].

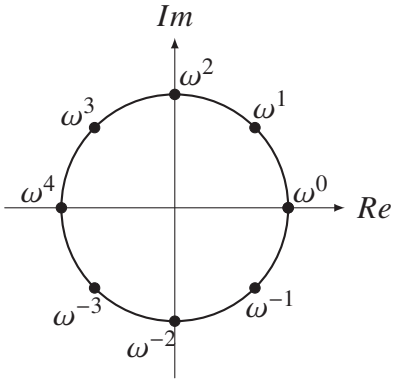


Figura 2.4: A oitava raiz de unidade no plano dos complexos, com frequências negativas.

Lembrando da fórmula de Euler $e^{it} = \cos(t) + i\text{sen}(t)$, podemos representar qualquer função trigonométrica por um exponencial complexo [Cha68]. Considerando a função $f(t) = \text{sen}(t)$ e sua representação em número complexo

$$\begin{aligned} e^{it} &= \cos(t) + i\text{sen}(t) \\ i\text{sen}(t) &= e^{it} - \cos(t) \end{aligned}$$

Sabendo que $e^{-it} = \cos(t) - i\text{sen}(t)$, temos $\cos(t) = e^{-it} + i\text{sen}(t)$, então

$$\begin{aligned} i\text{sen}(t) &= e^{it} - (e^{-it} + i\text{sen}(t)) \\ i\text{sen}(t) &= e^{it} - e^{-it} - i\text{sen}(t) \\ 2i\text{sen}(t) &= e^{it} - e^{-it} \\ \text{sen}(t) &= \frac{e^{it} - e^{-it}}{2i} \end{aligned}$$

De forma parecida, temos $\cos(t) = \dfrac{e^{it} + e^{-it}}{2}$. Portanto, na N-ésima raiz complexa de unidade, podemos representar qualquer função trigonométrica com frequência de até $N/2 - 1$.

2.3.1 Teorema da amostragem

Um sinal discreto possui um número finito de amostras que são medições igualmente espaçadas do sinal contínuo. Ao representar um sinal dado pela função $f(t) = \text{sen}(t)$, teremos um vetor discreto de tamanho $N = 4$, ou seja, o vetor possui 4 amostras igualmente espaçadas entre 0 e 2π do sinal contínuo. Portanto, um vetor v_f que representa o sinal discreto $f(t)$ possui os valores

$$v_f = [f(0), f(1/2\pi), f(2/2\pi), f(3/2\pi)]$$

Ao representar sinais discretos, pode surgir um fenômeno conhecido como *aliasing*, onde certos sinais se tornam iguais à outros sinais de frequência maior devido ao número de amostras do período. Considere as funções $g(t) = \text{sen}(5x)$ e $h(t) = \text{sen}(9x)$. Representando estas funções discretamente com $N = 4$ amostras, teremos sinais que possuem exatamente os mesmos valores, *i.e.*, $v_f = v_g = v_h$, onde v_g e v_h são vetores que representam o sinal discreto $g(t)$ e $h(t)$ respectivamente. Neste caso, dizemos que a função $g(t)$ ou $h(t)$ está *se passando pela* função $f(t)$. A Figura 2.5 mostra onde as amostras são medidas são exatamente iguais e isso ocorre para todos os sinais que tenham uma frequência $\mathcal{F}$ tal que $\mathcal{F} \mod N = 1$.

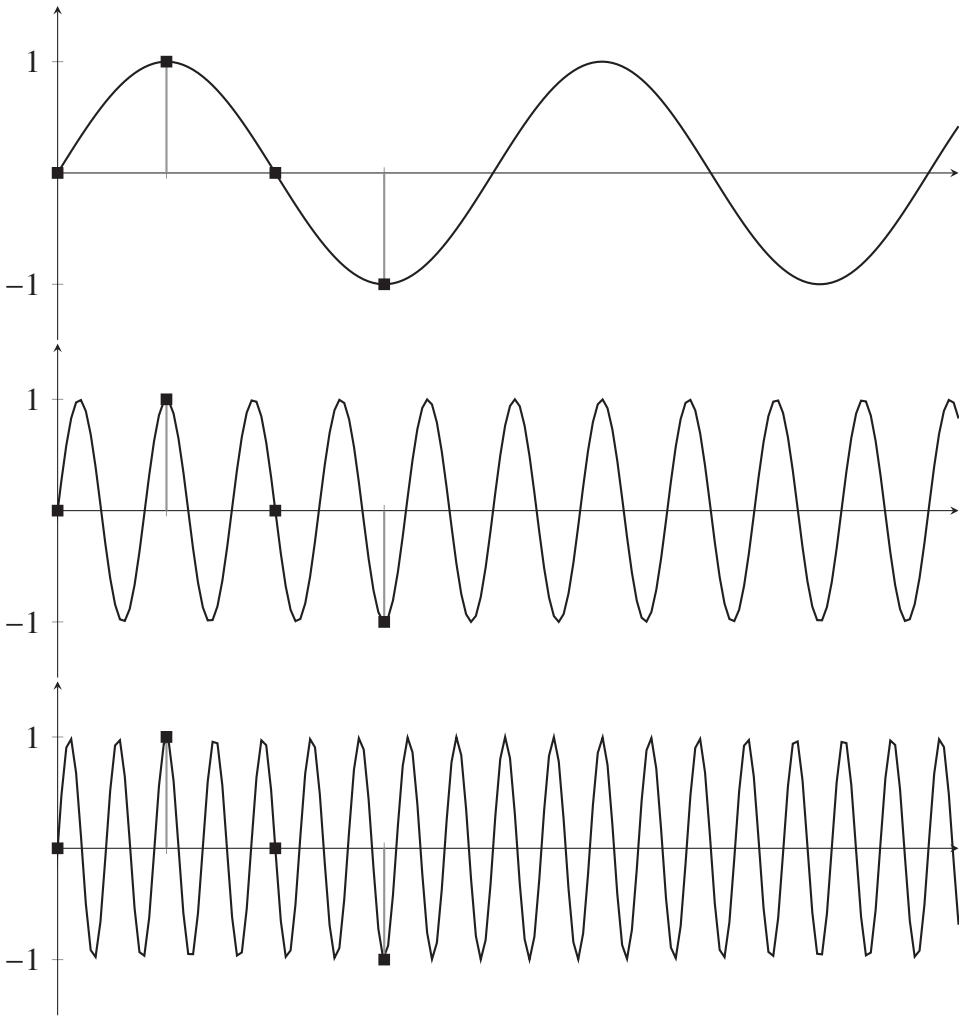


Figura 2.5: Com $N = 4$ amostras, o sinal $f(t)$ "se passa" pelos sinais $g(t)$ e $h(t)$.

O número de amostras necessárias para representar corretamente um sinal que possui uma frequência $\mathcal{F}$ é $N = 2\mathcal{F}$ amostras deste sinal. O *Teorema da amostragem* foi provado por

Shannon [Sha49] e ficou conhecido como o *Teorema de Shannon*, também como *Teorema de Nyquist*, devido à contribuição Harry Nyquist ao estudo. Formalmente, temos o Teorema 2.

Teorema 2. *Se uma função $f(x)$ não possui frequência maior ou igual que $\mathcal{F}$, então essa função pode ser representado corretamente por $N = 2\mathcal{F}$ amostras igualmente espaçadas.*

Neste caso, estamos falando de amostras múltiplas de $1/2\mathcal{F}$. A taxa de amostragem $N = 2\mathcal{F}$ e a frequência $\mathcal{F}$ são chamados de *taxa de Nyquist* e *frequência de Nyquist*, respectivamente. Ou seja, para representar todo espectro de um sinal, devemos usar uma taxa de amostragem que seja ao menos o dobro da maior frequência presente no espectro daquele sinal.

2.4 A transformada discreta de Fourier

Formalmente, a transformada de Fourier é uma transformação unitária entre dois espaços de Hilbert, sejam eles $L^2(\mathbb{R})$ e $L^2(\widehat{\mathbb{R}})$. Portanto não é possível afirmar que seja uma mudança de base para todos os casos. Porém, se considerarmos o caso discreto finito, *i.e.*, a transformada discreta de Fourier ou *DFT* (do inglês, *Discrete Fourier Transform*), podemos fixar o conjunto $\mathbb{R}$, tal que seja isomorfo ao seu dual, *i.e.*, $\mathbb{R} \cong \widehat{\mathbb{R}}$. Neste caso, teremos uma transformação de um espaço de Hilbert para ele mesmo, o que essencialmente é uma mudança de base. Portanto, no caso discreto, está certo afirmar que temos uma mudança de base para a transformada de Fourier.

2.4.1 Vetores e bases

Na álgebra linear, vetores são descritos como o resultado da soma com peso de dois vetores chamados de *vetor base*. É comum que seja considerada a base chamada *canônica* quando nenhuma base está explicita. Um vetor de duas dimensões pode ser descrito como um produto escalar entre os *vetores base* e um vetor de *coeficientes*. Para uma base de duas dimensões, temos um par de coeficientes $[x, y]$ tal que

$$v = x \cdot b_0 + y \cdot b_1$$

onde b_0 e b_1 são os vetores da base B. O vetor v definido na base B é denotado por v_B. No caso da base canônica de duas dimensões, temos $B = \{b_0, b_1\}$, onde $b_0 = [1, 0]$ e $b_1 = [0, 1]$. A Equação 2.3 representa a base B em forma de matriz, onde cada coluna representa um vetor de base.

$$B = \begin{bmatrix} b_0[0] & b_1[0] \\ b_0[1] & b_1[1] \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.3}$$

Generalizando, uma base B de N dimensões é representada pela Equação 2.4.

$$B = \begin{bmatrix} b_0[0] & b_1[0] & b_2[0] & \dots & b_{N-1}[0] \\ b_0[1] & b_1[1] & b_2[1] & \dots & b_{N-1}[1] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_0[N-1] & b_1[N-1] & b_2[N-1] & \dots & b_{N-1}[N-1] \end{bmatrix} \tag{2.4}$$

Uma mudança de base da base canônica para uma outra base $B' = \{b'_0, b'_1, \dots, b'_{N-1}\}$ é definida pela Equação 2.5 abaixo.

$$v_{B'} = B'^{-1} v_B \tag{2.5}$$

onde B é a base original e B' é a base que queremos representar o vetor v. A Equação 2.5 nos dá o algoritmo direto para computar essa mudança de base. Sabendo que o custo computacional

para computar a inversa de uma matriz qualquer é $O(N^3)$, onde N é o tamanho da matriz, mudar a base seria limitado pelo custo da inversão matricial.

Sabendo que a inversa de uma base *ortonormal* é igual a sua transposta, *i.e.*,

$$B^{-1} = B^t,$$

neste caso, o custo computacional de uma mudança de base é limitado pelo produto matricial, ou seja, $O(N^2)$. Assim, a k-ésima posição do vetor N-dimensional v na base B' é dada pela Equação 2.6.

$$v_{B'}[k] = \sum_{t=0}^{N-1} b'_k[t] v_B[t] \tag{2.6}$$

A Equação 2.7 representa uma mudança de base no caso de uma base ortonormal.

$$\begin{bmatrix} b_0[0] & b_0[1] & b_0[2] & \dots & b_0[N-1] \\ b_1[0] & b_1[1] & b_1[2] & \dots & b_1[N-1] \\ b_2[0] & b_2[1] & b_2[2] & \dots & b_2[N-1] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{N-1}[0] & b_{N-1}[1] & b_{N-1}[2] & \dots & b_{N-1}[N-1] \end{bmatrix} . \begin{bmatrix} v_B[0] \\ v_B[1] \\ v_B[2] \\ \vdots \\ v_B[N-1] \end{bmatrix} = \begin{bmatrix} v_{B'}[0] \\ v_{B'}[1] \\ v_{B'}[2] \\ \vdots \\ v_{B'}[N-1] \end{bmatrix} \tag{2.7}$$

2.4.2 Funções trigonométricas

Uma função discreta $f(t)$ possui finitos pontos que representam o valor da função para cada valor de t. Considerando funções periódicas, podemos representar a função corretamente considerando que temos um vetor com valores do período inteiro T. Supondo que temos uma função trigonométrica $f(t) = \text{sen}(t)$, sabemos que o período desta função é $T = 2\pi$. Então, para uma representação discreta de tamanho $N = 4$, teremos um vetor onde cada valor corresponde $t = \frac{i2\pi}{N} = \frac{i2\pi}{4}$ para $0 \geq i < N$. Ou seja, teremos um vetor como representado na Equação 2.8.

$$f = [0, 1, 0, -1] \tag{2.8}$$

Sabendo que funções trigonométricas formam um conjunto ortonormal, podemos definir uma base formada de funções trigonométricas para transformar uma função em uma base qualquer para uma base trigonométrica. Esta base trigonométrica é mais conhecida como a *base de Fourier*.

É possível descrever qualquer função [Tol62] em uma base trigonométrica. O cálculo para transformar um vetor de sua base para a base de Fourier permanece o mesmo da Equação 2.6. Os valores usados como base de Fourier são baseados nos valores de senos e cossenos, então utilizando a identidade de Euler, é possível representar os valores das funções trigonométricas na forma complexa. A mudança de base para uma base trigonométrica é conhecida como *transformada discreta de Fourier*. De forma geral, não é possível afirmar que a transformada de Fourier no caso contínuo é uma mudança de base.

Se temos uma função f, então esta mesma função na base de Fourier é denotado por $\widehat{f}$. O k-ésimo valor da função f na base de Fourier, *i.e.*, $\widehat{f}(k)$, é dado pelo produto escalar do k-ésimo vetor base pelo vetor f. De uma maneira informal, é possível interpretar a Equação 2.9 como a pergunta: Para cada instante de tempo t de um sinal f, qual a contribuição da função de

base $\omega_k(t)$ naquele instante? Formalmente, a equação da transformada discreta de Fourier, ou simplesmente *DFT* (do inglês, *Dicrete Fourier Transform*) é dada pela Equação 2.9.

$$\widehat{f}(k) = \sum_{t=0}^{N-1} \omega^{kt} f(t), \tag{2.9}$$

onde $\omega = e^{-2\pi i/N}$ e $\widehat{f}, f \mapsto \mathbb{C}$.

Neste caso, temos uma base N-dimensional. Então cada vetor da base possui um valor relacionado com a frequência da função trigonométrica. Por este motivo, é comum denominar uma função na base de Fourier como uma função no *domínio da frequência*. A Matriz 2.10 representa a base discreta de Fourier de N dimensões.

$$\begin{bmatrix} b_0[0] & b_0[1] & b_0[2] & \dots & b_0[N-1] \\ b_1[0] & b_1[1] & b_1[2] & \dots & b_1[N-1] \\ b_2[0] & b_2[1] & b_2[2] & \dots & b_2[N-1] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{N-1}[0] & b_{N-1}[1] & b_{N-1}[2] & \dots & b_{N-1}[N-1] \end{bmatrix} = \begin{bmatrix} \omega^{0\cdot 0} & \omega^{0\cdot 1} & \omega^{0\cdot 2} & \dots & \omega^{0\cdot N-1} \\ \omega^{1\cdot 0} & \omega^{1\cdot 1} & \omega^{1\cdot 2} & \dots & \omega^{1\cdot N-1} \\ \omega^{2\cdot 0} & \omega^{2\cdot 1} & \omega^{2\cdot 2} & \dots & \omega^{2\cdot N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega^{N-1\cdot 0} & \omega^{N-1\cdot 1} & \omega^{N-1\cdot 2} & \dots & \omega^{N-1\cdot N-1} \end{bmatrix} \tag{2.10}$$

Considere o caso de uma matriz de 8 dimensões dada pela Matriz 2.11.

$$\begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ \omega^0 & \omega^2 & \omega^4 & \omega^6 & \omega^8 & \omega^{10} & \omega^{12} & \omega^{14} \\ \omega^0 & \omega^3 & \omega^6 & \omega^9 & \omega^{12} & \omega^{15} & \omega^{18} & \omega^{21} \\ \omega^0 & \omega^4 & \omega^8 & \omega^{12} & \omega^{16} & \omega^{20} & \omega^{24} & \omega^{28} \\ \omega^0 & \omega^5 & \omega^{10} & \omega^{15} & \omega^{20} & \omega^{25} & \omega^{30} & \omega^{35} \\ \omega^0 & \omega^6 & \omega^{12} & \omega^{18} & \omega^{24} & \omega^{30} & \omega^{36} & \omega^{42} \\ \omega^0 & \omega^7 & \omega^{14} & \omega^{21} & \omega^{28} & \omega^{35} & \omega^{42} & \omega^{49} \end{bmatrix} \tag{2.11}$$

Sabendo que temos uma raiz de unidade de tamanho $N = 8$, temos apenas 8 valores distintos. Ou seja, podemos substituir todos os expoentes pelos respectivos módulos por 8. Neste caso, a Matriz 2.12 representa a mesma Matriz 2.11.

$$\begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ \omega^0 & \omega^2 & \omega^4 & \omega^6 & \omega^0 & \omega^2 & \omega^4 & \omega^6 \\ \omega^0 & \omega^3 & \omega^6 & \omega^1 & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ \omega^0 & \omega^4 & \omega^0 & \omega^4 & \omega^0 & \omega^4 & \omega^0 & \omega^4 \\ \omega^0 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega^1 & \omega^0 & \omega^3 \\ \omega^0 & \omega^6 & \omega^4 & \omega^2 & \omega^0 & \omega^6 & \omega^4 & \omega^2 \\ \omega^0 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega^1 \end{bmatrix} \tag{2.12}$$

Para realizar o processo inverso, *i.e.*, obter o sinal a partir do espectro, somam-se a base com o expoente negativo. A Equação 2.13 representa a transformada inversa de Fourier.

$$f(t) = \sum_{k=0}^{N-1} \omega^{-kt} \widehat{f}(k) \tag{2.13}$$

A Equação 2.9 e a Equação 2.13 nos dá um algoritmo para computar a transformada de Fourier e a transformada inversa de Fourier, respectivamente. Portanto, o algoritmo é limitado pelo produto de matriz por vetor. Assim, o custo computacional para computar a transformada de Fourier é quadrático em relação ao tamanho da matriz, ou seja, $O(N^2)$. Algoritmos mais eficientes para computar a *DFT* possuem uma complexidade de tempo de $O(N \log N)$ e são conhecidos na literatura como *transformada rápida de Fourier*, ou simplesmente como *FFT*, do inglês *Fast Fourier Transform*. Iremos descrever com mais detalhes o algoritmo *FFT* na Seção 2.8.

2.5 Convolução

Convolução é uma operação matemática formal, assim como a adição, subtração e multiplicação. Dados dois números, a adição produz um terceiro número, por outro lado dados dois sinais a e b, a convolução produz um terceiro sinal c. Cada ponto do sinal c é a integral da multiplicação entre o sinal a e o sinal b deslocado. A convolução é indicada pelo símbolo $*$, ou seja, dados dois sinais f e g, a convolução destes sinais é representada como $f * g$. Formalmente, temos a Definição 1.

Definição 1. *Dados dois sinais $f(n)$ e $g(m)$, uma convolução $h(k) = f(n) * g(m)$ é dado pela Equação 2.14*

$$h(k) = \sum_{m=0}^{2N-1} f(m)g(n-m) \tag{2.14}$$

para $n, m \in \{0, \ldots, N-1\}$ e $k \in \{0, \ldots, 2N-1\}$.

A convolução de dois sinais de tamanho N é um sinal de $2N-1$ amostras, onde a k-ésima amostra é dada pela Equação 2.14 tal que $0 \leq k < 2N$. Também é possível obter a convolução de dois sinais de tamanho diferentes, neste caso, para dois sinais de tamanho N e M, o sinal resultante terá tamanho $N + M - 1$. Por questões de simplicidade, vamos considerar sempre sinais de tamanho igual.

Considere o sinal quadrado $f(n)$ de tamanho $N = 32$ amostras tal que seu o gráfico é representado pela Figura 2.6.

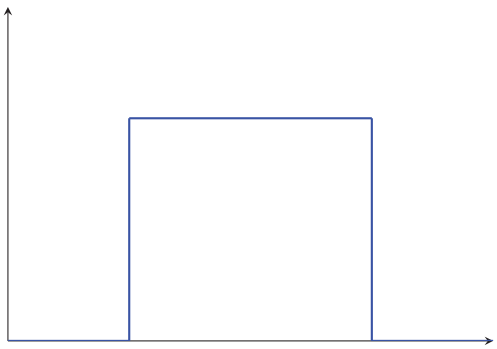


Figura 2.6: $f(n)$ é chamada de função quadrada

Também considere o sinal $g(m)$ de tamanho $N = 32$ que possui dois pulsos unitários nos pontos onde $m = 5$ e $m = 20$, tal que seu gráfico é dado pela Figura 2.7.

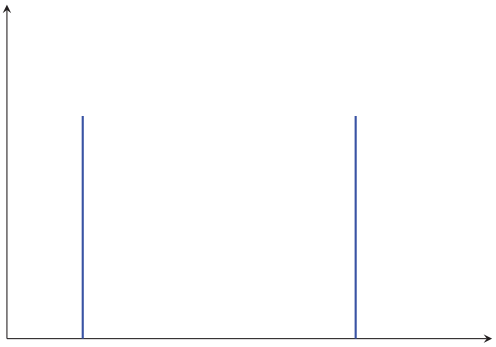


Figura 2.7: Uma função qualquer de exemplo para a operação de convolução

A convolução destes dois sinais resulta em um sinal $h(n)$ de tamanho $N = 32+32-1 = 63$ que possui dois pulsos quadrados localizados nos pontos $x = 20$ e $x = 40$ com largura igual ao pulso quadrado do sinal $f(x)$. O gráfico resultante está representado pela Figura 2.8

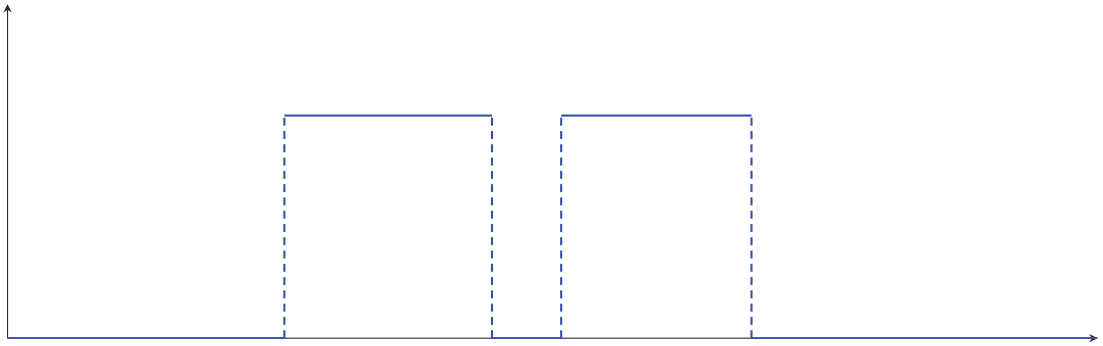


Figura 2.8: O sinal $h(x)$ resultante da convolutação de tamanho 63

A partir da Definição 1, temos o Teorema 4. O Teorema 3 abaixo é definido aqui porque é usado para provar o Teorema 4.

Teorema 3. *Seja $\hat{f}_k$ a k-ésima posição da transformada de f e um inteiro qualquer $\Delta \in [0 \ldots N-1]$,*

$$\hat{f}_k(n-\Delta) = \hat{f}(k) \cdot e^{-i2\pi\Delta k/N} \tag{2.15}$$

Demonstração.

$$\begin{aligned}
\hat{f}(n-\Delta)_k &= \sum_{n=0}^{N-1} f(n-\Delta) \cdot e^{-i2\pi nk/N} \\
&= \sum_{m=-\Delta}^{N-1-\Delta} f(m) \cdot e^{-i2\pi(m+\Delta)k/N} \quad (m = n-\Delta) \\
&= \sum_{m=0}^{N-1} f(m) \cdot e^{-i2\pi mk/N} \cdot e^{-i2\pi\Delta k/N} \\
&= e^{-i2\pi\Delta k/N} \sum_{m=0}^{N-1} f(m) \cdot e^{-i2\pi mk/N} \\
&= \hat{f}(k) \cdot e^{-i2\pi\Delta k/N}
\end{aligned}$$

□

Teorema 4. *Dados dois sinais $f(n), g(m) \in \mathbb{C}^N$ e suas respectivas transformadas $\hat{f}(n), \hat{g}(m) \in \mathbb{C}^N$, então*

$$f * g = \hat{f} \cdot \hat{g} \tag{2.16}$$

Demonstração. Aplicando a DFT na convolução de f e g, temos

$$\begin{aligned}\widehat{f * g}(k) &= \sum_{n=0}^{N-1} (f * g)(n) e^{-i2\pi kn/N} \\ &= \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} f(m) g(n-m) e^{-i2\pi kn/N}\end{aligned}$$

Alterando a ordem dos somatórios, ficamos com o seguinte

$$\begin{aligned}\widehat{f * g}(k) &= \sum_{m=0}^{N-1} f(m) \sum_{n=0}^{N-1} g(n-m) e^{-i2\pi kn/N} \quad \text{pelo Teorema 3} \\ &= \sum_{m=0}^{N-1} f(m) e^{-i2\pi km} \hat{g}(k) \\ &= \hat{f}(k)\hat{g}(k)\end{aligned}$$

□

O Teorema 4 também é válido no sentido contrário, ou seja $\widehat{f \cdot g} = \widehat{f} * \widehat{g}$. Computar a convolução diretamente como está na Definição 1 requer $O(N^2)$ operações. Porém, dado o Teorema 4, o produto escalar dos sinais f e g no domínio da frequência produzem o sinal h no domínio da frequência. Portanto, aplicando a transformada inversa de Fourier, teremos $h = f * g$. Utilizando o algoritmo de transformada rápida de Fourier, *FFT*, podemos executar uma convolução com $O(N \log N)$ operações.

Vale notar que executar um produto escalar de dois sinais f e g de tamanho N e M, respectivamente. Aplicando a transformada inversa de Fourier no resultado não dará um sinal de tamanho $N + M - 1$ porque a transformada de um sinal de tamanho N será também de tamanho N. Portanto, é necessário concatenar o sinal f com $M - 1$ zeros, do mesmo jeito, concatenar g com $N - 1$ zeros. O resultado será a convolução $h = f * g$ de tamanho $N + M - 1$ como esperado.

2.6 Vazamento espectral

Uma das regras que uma função deve satisfazer para que possa ser expandida em uma série trigonométrica é que ela seja periódica. Portanto, toda instância de uma transformada discreta de Fourier é um vetor onde seus valores representam um período completo de uma função periódica. De forma parecida, os coeficientes na série de Fourier também correspondem à funções contínuas. No caso discreto, temos *cestos* que correspondem à frequências específicas. Estes cestos são conhecidos como *cestos espectrais* e cada cesto espectral corresponde à energia de uma frequência.

Considere o sinal f dado pela função $f(x) = \text{sen}(2\pi x)$ com 32 amostras. Neste caso, o domínio da frequência possui apenas um coeficiente diferente de zero (dois se considerar as frequências negativas).

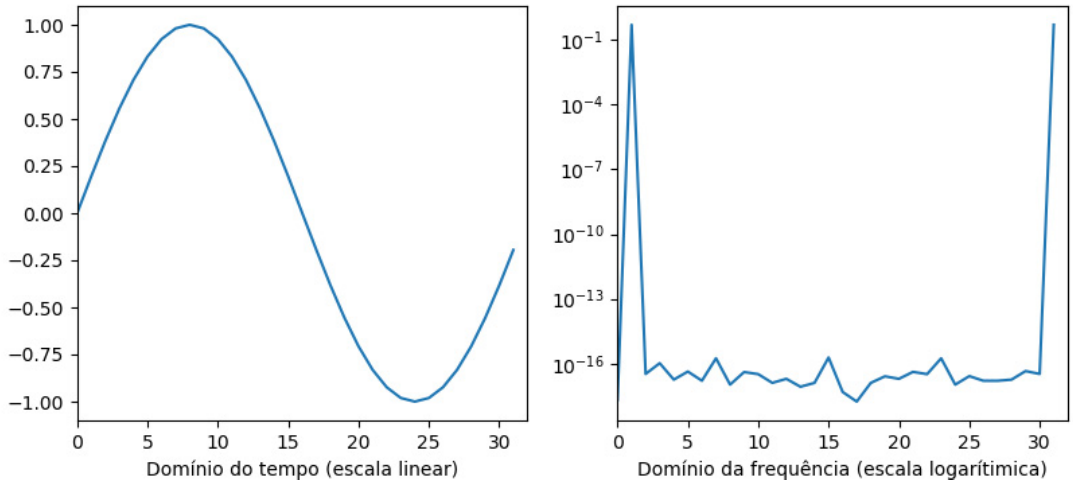


Figura 2.9: O sinal f e sua resposta espectral.

Vale notar que as posições diferentes de $\hat{f}[1]$ e $\hat{f}[N-1]$ não são exatamente zero, porém são insignificantes. Removendo apenas uma amostra do sinal, ocorre um fenômeno que é conhecido como *vazamento espectral*. Removendo o último elemento, o sinal com $N = 31$ amostras e tem seu espectro muito diferente do que temos na Figura 2.9.

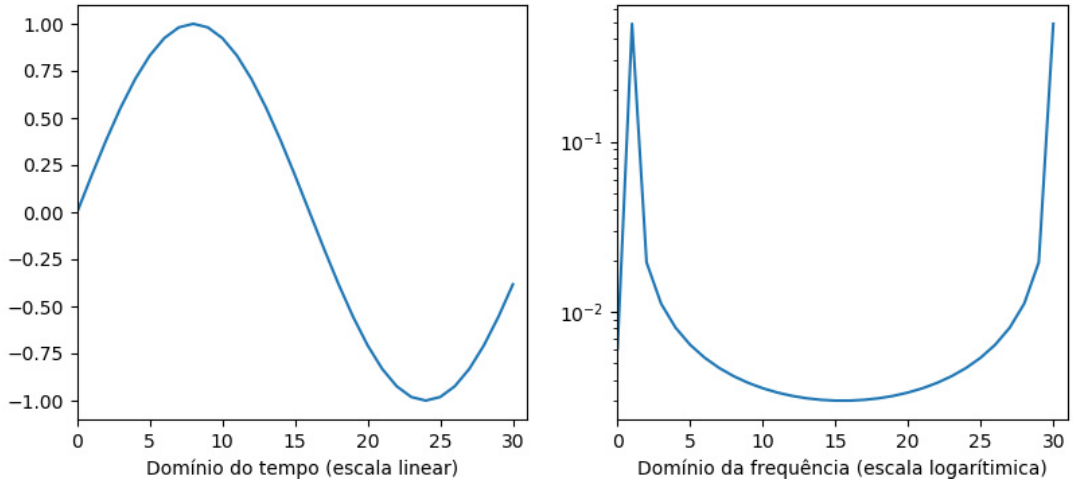


Figura 2.10: Processando o sinal f com uma amostra a menos nos dá um vazamento espectral.

O vazamento espectral ocorre quando existe alguma descontinuidade gerada através da expansão periódica. Ou seja, no primeiro caso, a frequência 1 do sinal é um múltiplo inteiro de $1/32$ e o sinal não possui descontinuidade. Porém não é verdade para o segundo caso, 1 não é múltiplo inteiro de $1/31$ e haverá uma descontinuidade na ponta desse sinal periódico. Essa descontinuidade faz surgir frequências que não existem no sinal. Dizemos que a energia da frequência "vaza" para os cestos espectrais vizinhos. Ao analisar sinais do mundo físico, estamos

assumindo que o sinal discreto é um período completo do sinal contínuo. Embora seja possível, um sinal discreto raramente será um período inteiro do sinal contínuo. Um método para mitigar os efeitos do vazamento espectral é utilizar as *funções de janelas*, mais conhecidas como *filtros de janela*.

2.7 Filtros de janela

Um filtro de janela, ou também chamado de função de janela, possui um gráfico semelhante ao gráfico de uma função Gaussiana, ou seja as pontas do gráfico de um filtro de janela se aproximam suavemente a zero. A resposta espectral de um filtro de janela é chamada de *kernel*. O kernel possui um pulso chamado de *lóbulo principal* e outros pulsos laterais chamados de *lóbulos laterais*. Aplicar um filtro de janela a um sinal é multiplicar ponto a ponto o filtro com o sinal e, portanto, aplicar uma convolução entre o espectro do sinal e o kernel. O lóbulo principal é onde a frequência "passa" e os lóbulos laterais são os efeitos do vazamento espectral. Quando nenhum filtro é usado, assume-se que tenha usado um filtro quadrado como mostra a Figura 2.11. O kernel do filtro quadrado é semelhante à uma função $\text{sinc}(x)$[1] e os lóbulos laterais se aproximam de zero linearmente. Observe que na Figura 2.10 temos um espectro com dois picos que decaem como o kernel da função quadrada. Isso é o resultado da convolução entre o kernel do filtro e o sinal.

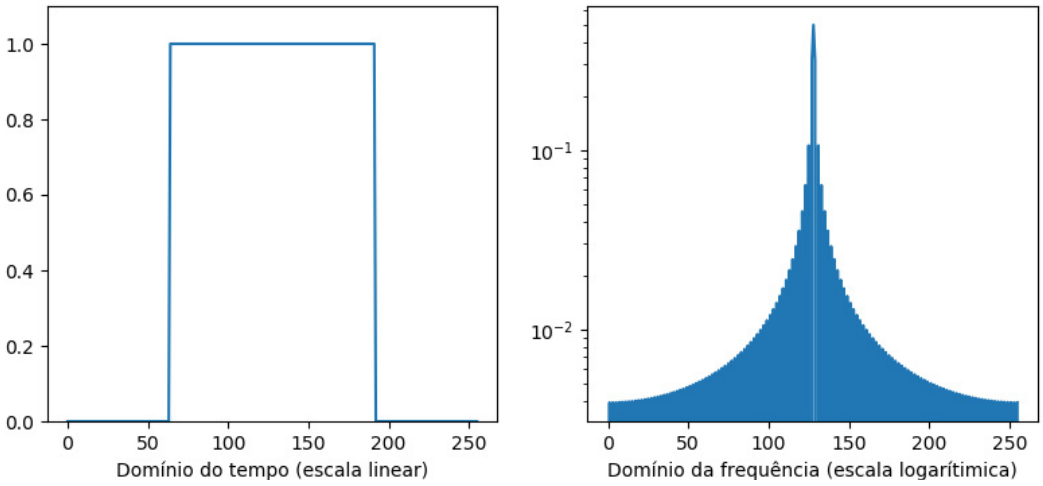


Figura 2.11: A função de janela quadrada é equivalente à usar nenhum filtro.

Existem diversos filtros de janela, um dos filtros mais comuns é dado pela Equação 2.17, chamada de função *Hann*. O gráfico do filtro Hann e o kernel desse filtro é representado pela Figura 2.12.

$$G(x) = 0,5 - 0,5 \cdot \cos\left(\frac{2\pi x}{N-1}\right) \tag{2.17}$$

[1] A função sinc é definida como: $sinc(x) = sen(x)/x$

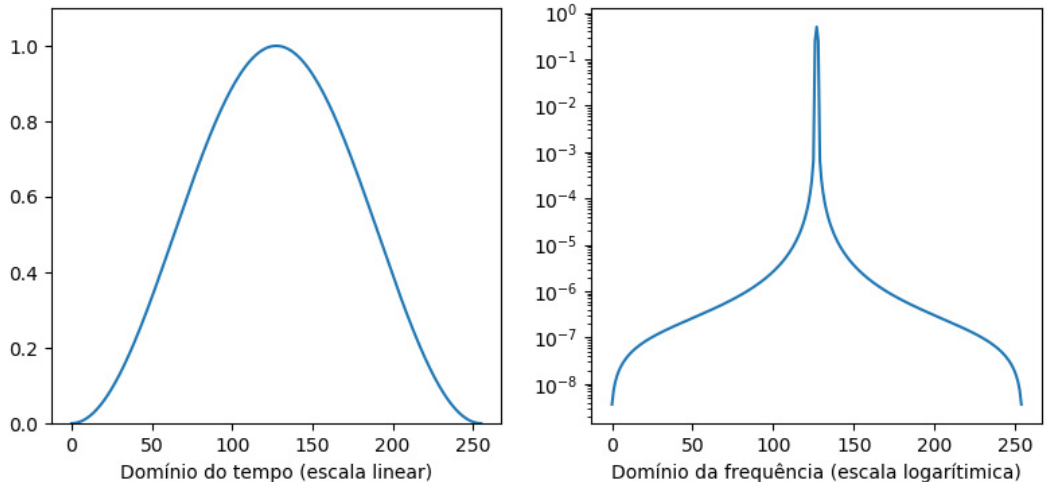


Figura 2.12: O filtro *Hann*, à esquerda o gráfico no domínio do tempo e à direita, a resposta espectral do filtro.

O espectro observado na Figura 2.10 é obtido sem o uso de um filtro de janela, teoricamente, deve-se ter um vazamento espectral mais acentuado que o espectro obtido utilizando um filtro de janela. Portanto, cada frequência diferente de zero do espectro terá um vazamento em que os lóbulos laterais decaem linearmente, como está mostrado na Figura 2.11. Então, aplicando o filtro Hann no exemplo anterior, cada frequência diferente de zero terá um vazamento em que os lóbulos laterais decaem exponencialmente. A Figura 2.13 mostra o sinal e o espectro do sinal após aplicar o filtro Hann. Observe que as frequências presentes são semelhantes porque a multiplicação ponto a ponto é uma convolução entre o sinal e o filtro no domínio da frequência, portanto o vazamento espectral é reduzido conforme o kernel do filtro utilizado.

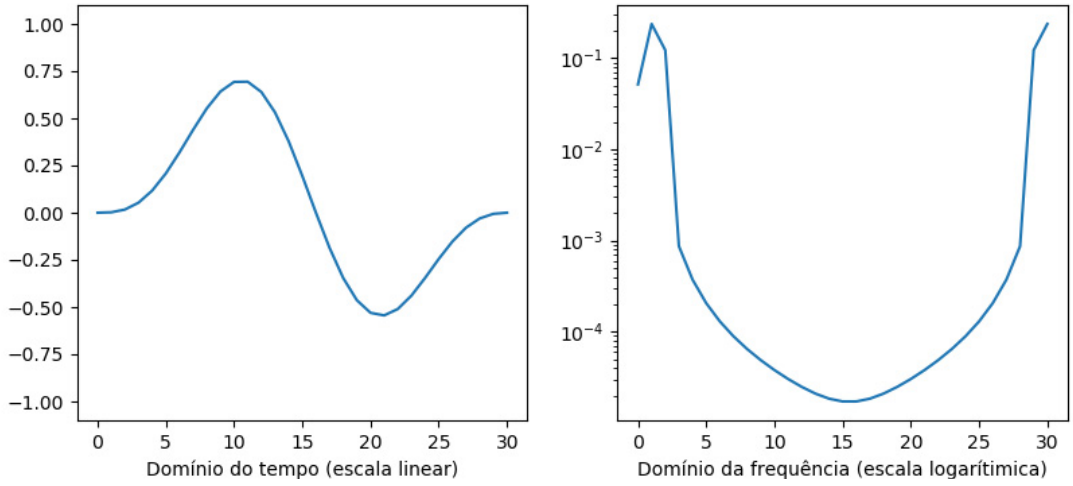


Figura 2.13: A resposta espectral do sinal após o filtro ser aplicado.

2.8 A transformada rápida de Fourier

Computar a transformada discreta de Fourier do modo mais direto terá sua complexidade de tempo limitada pela multiplicação de matriz por vetor, ou seja, $O(N^2)$. Porém, o algoritmo

mais conhecido para computar a DFT foi apresentado por Tukey e Cooley em 1967 [CT65]. Este algoritmo ficou conhecido como a *transformada rápida de Fourier*, ou *FFT*, do inglês *Fast Fourier Transform*. Considerado um importante algoritmo do último século [Cip00], a transformada rápida de Fourier (*FFT*) recursivamente divide a instância do problema da DFT em dois problemas da metade do tamanho, por isso é também conhecida como *Radix-2 DFT*. Essa abordagem entrega um algoritmo com complexidade de tempo de $O(N \log N)$.

2.8.1 História da transformada rápida de Fourier

Gauss foi o primeiro a desenvolver um método para computar a DFT dividindo recursivamente a instância de entrada e reduzindo a complexidade computacional. Inclusive, se fatos históricos estiverem corretos, Gauss teria descoberto a transformada rápida de Fourier antes que Joseph Fourier tivesse terminado o trabalho sobre a representação de funções como séries trigonométricas. A solução de Gauss teria sido publicado *post mortem* em 1866, porém foi escrito em 1805, antes mesmo do trabalho de Joseph Fourier sobre a representação de funções como séries trigonométricas em 1807, apesar de ter sido publicado apenas em 1822 [Fou22] por conta da má recepção por parte de Lagrange e então a publicação em *Memoirs of the Academy* foi negada [HJB84]. Apesar disso, Gauss não desenvolveu o método para um sinal de tamanho qualquer, nem analisou assintóticamente seu método.

Ao longo do século 19 e início do século 20, algoritmos para computar a transformada rápida de Fourier foram redescobertos. O algoritmo de Yates [Yat37] que surgiu em 1937 e ainda é usado na área de estatística. Danielson e Lanczos [DL42] também publicaram uma versão para computar a DFT utilizando a divisão recursiva da instância de entrada. A base de Danielson e Lanczos era dividir a entrada em pontos pares e ímpares. Formalmente, publicaram o *Lemma de Danielson-Lanczos*, mesmo que não tenha sido citado por Cooley e Tukey, é a base da transformada rápida de Fourier como se conhece hoje. Danielson e Lanczos não ficaram reconhecidos como os criadores da transformada rápida porque publicaram em 1944, antes do surgimento do computador, portanto os cálculos eram feitos à mão. Usando o algoritmo deles, executar manualmente uma *DFT* de 64 pontos levava em torno de 140 minutos.

A publicação que teve mais repercurssão foi de John Tukey e James Cooley em 1965 [CT65]. A única referência feita no artigo de Cooley e Tukey foi de *I. J. Good*[Goo58]. I. J. Good publicou uma solução para a DFT que consistia de uma divisão recursiva da entrada, porém exigia que sinais tivessem tamanho $N = N_1 N_2$ tal que N_1 e N_2 sejam relativamente primos. Mais tarde ficou sabendo que este seria um problema diferente da transformada de Fourier, conhecido como *prime-factor algorithm*. Porém, o algoritmo de Cooley e Tukey possibilitava para qualquer tamanho de sinal. Usando a mesma idéia que Danielson e Lanczos, o algoritmo de Cooley e Tukey computava a DFT de 2048 pontos em $0,02$ minutos em uma máquina IBM 7094 [CT65].

2.8.2 O algoritmo

O algoritmo da transformada rápida é simples, está praticamente todo baseado do Lema 1.

Lema 1. *A DFT de tamanho N, onde N é* par*, pode ser reescrita como a soma de duas DFT de tamanho* $N/2$ *cada*

$$\widehat{f}(k) = \widehat{f_p} + \omega^n \widehat{f_i} \tag{2.18}$$

onde $\widehat{f_p}$ *é a tranformada de Fourier para os pontos de índice* par*, e* $\widehat{f_i}$ *é para os índices* ímpares*.* $O\ \omega = e^{-2\pi i k t/N}$ *é chamado de* twiddle factor.

Demonstração.

$$\begin{aligned}\widehat{f}(k) &= \sum_{t=0}^{N-1} f(t) \cdot e^{i2\pi kt/N} \\ &= \sum_{t=0}^{N/2-1} f(2t) \cdot e^{i2\pi kt/(N/2)} + \omega^n \sum_{t=0}^{N/2-1} f(2t+1) \cdot e^{i2\pi kt/(N/2)} \\ &= \widehat{f_p} + \omega^n \widehat{f_i}\end{aligned} \tag{2.19}$$

□

O algoritmo inverte os bits dos índices para separar os pontos com índice par dos pontos com índice ímpar. E então chama recursivamente para cada um dos conjuntos. A condição de parada é quando temos uma DFT de tamanho $N = 1$.

Algoritmo 1: FFT(F, N)

Dados: Um vetor de valores complexos F e N sendo o tamanho
Saída: Um vetor C com os coeficientes harmônicos de F

```
1  se N = 1 então
2  |  return F[0]
3  senão
4  |  ω_N ← e^{-i2π/N}
5  |  ω ← 1
6  |  F_inv ← inverte_bit(F)
7  |  C_par ← FFT(F_inv[0 ... N/2 − 1], N/2)
8  |  C_impar ← FFT(F_inv[N/2 ... N − 1], N/2)
9  |  para k=0 até N/2-1 faça
10 |  |  C[k] ← C_par[k] + ω * C_impar[k + N/2]
11 |  |  C[k + N/2] ← C_par[k] - ω * C_impar[k + N/2]
12 |  |  ω ← ω * ω_N
13 |  fim
14 |  retorna C
15 fim
```

A função de `inverte_bit` inverte os bits de cada índice e copia para F_{inv}. Os primeiros $N/2$ elementos de F_{inv} são todos os índices pares de F, os $N/2$ seguintes são todos os índices ímpares. Esta etapa é simples e linear no tamanho de F. A relação de recorrência no Algoritmo 1 é a clássica recorrência de divisão e conquista. A relação é dada pela Equação 2.20. Ao resolver a relação, pode-se então concluir que a complexidade de tempo é $O(N \log N)$.

$$FFT(N) = \begin{cases} O(1), \text{ para } N = 1 \\ FFT(N/2) + FFT(N/2) + O(N), \text{ para } N > 1 \end{cases} \tag{2.20}$$

3 Transformada esparsa de Fourier

Quando temos um sinal f e gostaríamos de obter seu espectro $\hat{f}$, sabemos que pelo Teorema 2, devemos amostrar o sinal em uma frequência que seja o dobro da maior frequência que $\hat{f}$ possui. Porém, na maioria dos casos, sinais no mundo real são esparsos, *i.e.*, possuem muitos de seus coeficientes insignificantes, *i.e.*, zero ou próximos de zero, e apenas um pequeno conjunto com k coeficientes que são significativos. Com isso em mente, pesquisas recentes tem mostrado grandes avanços em algoritmos que fazem proveito dessa *esparsidade* destes sinais. Utilizando uma taxa de amostragem abaixo do que o Teorema 2 exige, estes algoritmos computam uma aproximação da DFT para estes sinais, e executando com complexidade sublinear no tamanho do sinal.

Por se tratar de sinais que são esparsos, cada aplicação possui sinais com características particulares, exigindo uma abordagem diferente para cada caso. Portanto estes algoritmos são um *framework* para algoritmos que computem a transformada de Fourier para sinais esparsos. Estes algoritmos receberam o nome de *sparse fast Fourier transform*, ou simplesmente *sFFT*. De uma forma geral, estes algoritmos são compostos de três etapas: subamostrar o sinal, localizar quais coeficientes são significativos e estimar a magnitude desses coeficientes.

Este capítulo é dedicado a definir o que é o problema da transformada esparsa de Fourier, fazendo uma revisão bibliográfica de algoritmos que computam a transformada esparsa de Fourier e analisar com mais detalhes o funcionamento de um algoritmo *simples* para computar a transformada esparsa de Fourier.

3.1 Definindo o problema

Existem dois tipos de transformada esparsa de Fourier, quando temos um sinal que possui exatamente k coeficientes diferentes de zero e quando temos um sinal *aproximadamente* esparso, com k coeficientes significativos.

Definição 2. *Considere um sinal f de tamanho N, cuja transformada de Fourier $\hat{f}$ é definida por*

$$\hat{f}(k) = \sum_{x=0}^{N-1} f(x) \cdot e^{-i2\pi kx/N}. \tag{3.1}$$

O espectro do sinal $\hat{f}$ é exatamente *k-esparso caso possua k coeficientes diferentes de zero e os restantes $N - k$ coeficientes são zero.*

O objetivo da transformada esparsa de Fourier é recuperar exatamente $\hat{f}$, achando a posição e o valor de cada um dos k coeficientes diferentes de zero. Para sinais *aproximadamente* esparsos, temos a Definição 3.

Definição 3. *Considere um sinal f de tamanho N, cuja transformada de Fourier $\hat{f}$ é definida por*

$$\hat{f}(k) = \sum_{x=0}^{N-1} f(x) \cdot e^{-i2\pi kx/N}. \tag{3.2}$$

O espectro do sinal $\hat{f}$ é aproximadamente *k-esparso caso possua k coeficientes diferentes de zero e o restante $N - k$ coeficientes são próximos de zero ou zero.*

A transformada esparsa de Fourier nos dois casos computa uma *aproximação* k-esparsa $\hat{f}'$ de $\hat{f}$. O objetivo é encontrar uma aproximação $\hat{f}'$ na qual o erro é dado pela diferença entre a *melhor* k-esparsa aproximação y e a DFT exata $\hat{f}$. A aproximação y de $\hat{f}$ é obtida definindo todos os $N - k$ coeficientes de $\hat{f}$ para zero. Desta forma, $\hat{f}'$ deve satisfazer a guarantia ℓ_2/ℓ_2 dada pela Equação 3.3.

$$||\hat{f} - \hat{f}'||_2 \leq C \min_{y \ k-\text{esparso}} ||\hat{f} - y||_2 \tag{3.3}$$

onde C é uma constante de aproximação e a minimização é dada pelo sinal exatamente k-esparso y.

3.2 Revisão bibliográfica

Nos últimos anos houve grande progresso no estudo de algoritmos que computam a transformada esparsa de Fourier. O primeiro algoritmo dessa classe surgiu para computar a transformada de Hadamard e foi publicada em 1991 por Mansour et al. [KM91, Man92]. Desde então, outros autores publicaram resultados semelhantes [Man92, GGI+02, AGS03, ACG05, Aka10, Iwe10]. Estes algoritmos possuem complexidade de tempo linear em relação à esparsidade do sinal, k, e sublinear ao tamanho do sinal, N. Os mais rápidos entre estes algoritmos possuem complexidade $O(k^2 \log^c N)$ [GGI+02, Iwe10], ou $O(k \log^c N)$ para $c > 2$ [ACG05]. Porém estes algoritmos superam o tempo de execução da FFT somente para sinais muito esparsos, *i.e.*, para $k \leq 135$ e $n = 2^{22}$ [IGS07].

Em [GGI+02, ACG05], os autores definem a função de janela G como sendo a função retangular comum (ou do inglês, *box-car*). A *DFT* $\widehat{G}$ dessa função de janela tem gráfico definido por uma função *sinc*, cujas pontas decaem linearmente. Desta forma, a energia dos coeficientes de Fourier mapeados para um determinado cesto "vaza" para outros cestos. Em [Man92], o algoritmo proposto estima uma convolução no domínio do tempo através de uma amostragem aleatória que leva a erros de estimação grandes. Então, para satisfazer a guarantia ℓ_2/ℓ_2, estes algoritmos estimam o maior coeficiente de Fourier e subtraem sua contribuição do sinal no domínio do tempo a cada iteração. Essas iterações são custosas e, na prática, não superam o tempo da FFT apenas para sinais muito esparsos. O algoritmo em [ACG05] ainda reduz o custo desse processo com o uso de "*amostras em grupo*", mas não resolve o problema porque haverá instâncias para *DFT* não-uniforme, o que também é custoso na prática.

Outro algoritmo que deve ser mencionado é proposto por [Iwe10]. É um algoritmo baseado em *interpolação*, que utiliza um filtro G tal que $G_i = 1$ se e somente se $i \mod N/p = 0$, e $G_i = 0$ caso contrário. Este filtro não causa vazamento espectral, portanto não é necessário múltiplas iterações como nos algoritmos anteriores. Por outro lado, é necessário que N seja divisível por p. Como não se pode assumir que N é divisível por p para qualquer p, o algoritmo trata o sinal como uma função contínua e realiza uma interpolação nos pontos necessários. Essa interpolação é custosa e acaba aumentando o tempo de execução.

Nenhum desses algoritmos conseguiu resultados empíricos melhores que a FFT para todo sinal esparso, *i.e.*, para $k = o(N)$. Além de possuirem estrutura complexa, possuem constantes assintóticas grandes que, na prática, acabam deixando o tempo de execução alto. O primeiro algoritmo que conseguiu resultados mais expressivos foi proposto por Hassanieh et al. [HIKP12b]. O algoritmo proposto utiliza um filtro que possui uma região de passe quase plana e pontas que decaem exponencialmente. Assim, o filtro gera um vazamento espectral desprezível e o algoritmo não necessita de muitas iterações como aqueles em [GGI+02, ACG05, Man92]. O algoritmo garante uma execução mais rápida que a implementação de alto desempenho da FFT para sinais de tamanho $N = 2^{22}$ e $k < 4000$ [SP14].

3.3 Princípios fundamentais

Esta seção introduz alguns conceitos básicos usados no algoritmo de [HIKP12a]. Definimos $\omega = e^{-i2\pi/n}$ como a n-ésima raiz de unidade. Todas as operações relacionadas com índices são feitas módulo N. Para um vetor $f \in \mathbb{C}^n$, o *vetor suporte* de f é o conjunto de índices i tal que $f(i) \neq 0$ é denotado por $sup(f)$. Finalmente, a norma $||f||_\infty$ é definida como $||f||_\infty = \max_i |f_i|$, para um vetor f.

3.3.1 Permutação de espectro

Definição 4. *Dado um número módulo inverso de n, σ^{-1}, um inteiro $\tau \in [N]$, definimos uma permutação de um sinal f como*

$$(P_{\sigma,\tau} f)_i = f_{i\sigma+\tau} \tag{3.3}$$

A permutação possui uma propriedade essencial para a etapa de divisão em cestos dos coeficientes. Podemos permutar o sinal f no domínio do tempo que o sinal no domínio da frequência $\hat{f}$ também será permutado, sem a necessidade de aplicar a DFT e então, permutar os valores de $\hat{f}$, conforme mostra o Teorema 5, adaptado de [ACG05].

Teorema 5. *Seja uma permutação $P_{\sigma,\tau}$ e f um vetor N-dimensional. Então*

$$\left(\widehat{P_{\sigma,\tau} f}\right)_{\sigma i} = \hat{f}_i \omega^{-\tau i}$$

Demonstração.

$$\begin{aligned}
\left(\widehat{P_{\sigma,\tau} f}_i\right) i &= \sum_{j\in[n]} \omega^{ij} f_{\sigma j+\tau} \\
&\quad \text{(definindo } a = \sigma i + \tau\text{)} \\
&= \sum_{a\in[N]} \omega^{i\sigma^{-1}(a-\tau)} f_a \\
&= \omega^{i\sigma^{-1}\tau} \sum_{a\in[N]} \omega^{\sigma^{-1}ai} f_a \\
&= \hat{f}_{\sigma^{-1}i} \omega^{-\tau\sigma^{-1}i}
\end{aligned} \tag{3.4}$$

Substituindo $i = \sigma i$, ficamos com o que queremos mostrar. Vale ressaltar que $\omega^{-\tau i}$ muda a fase em $f_i\omega^{-\tau i}$, mas não muda sua magnitude. □

3.3.2 Filtro de janela plano

Definição 5. *Seja $G(\epsilon, \delta, w)$ um filtro de janela padrão tal que G seja um vetor simétrico com vetor suporte $supp(G) \subseteq [-w/2, w/2]$, onde a primeira posição é unitária $\hat{G}_0 = 1$ e para todo $i \in [-\epsilon N, \epsilon N]$, temos $\hat{G}_i > 0$. Então, para todo $i \notin [-\epsilon N, \epsilon N]$, temos valores abaixo de um δ,* i.e., $|\hat{G}_i| < \delta$.

Um exemplo de um filtro de janela que satisfaças as condições dadas pela Definição 5 é representada pela Figura 3.1. Neste caso temos um filtro Dolph-Chebshev com os parâmetros $(1/22, 10^{-8}, 133)$. Isso nos dá um filtro com lóbulo principal de largura $|main\_lobe(\hat{G})| = 2 \cdot 256/22$ e ponto máximo no primeiro índice, $\hat{G}_0 = 1$, lóbulos laterais ao máximo $|side\_lobes(\hat{G})| = \delta = 10^{-8}$.

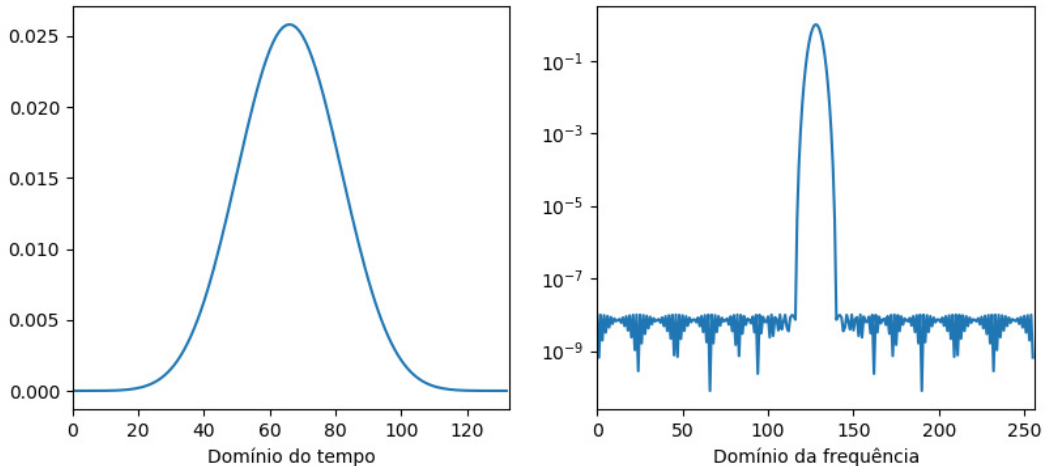
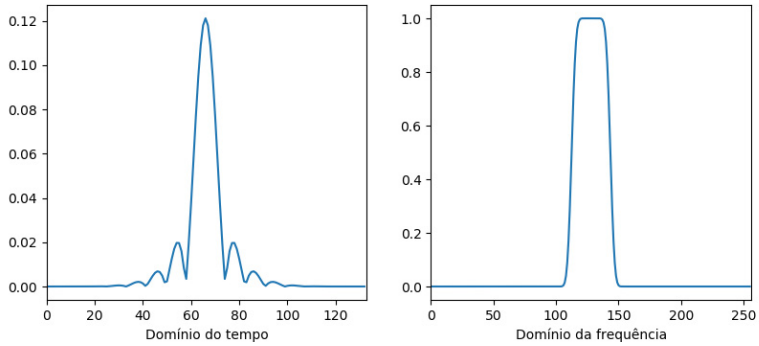


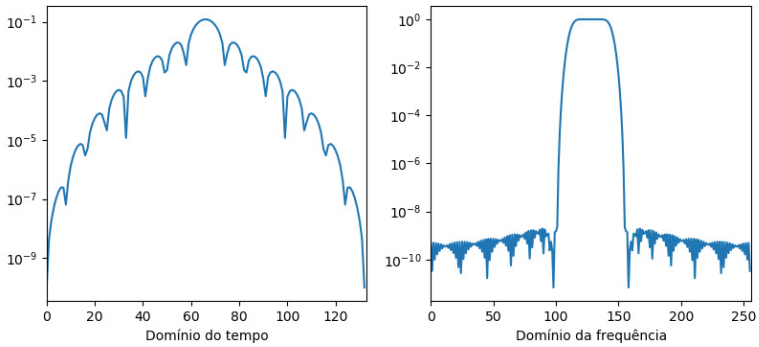
Figura 3.1: O filtro Dolph-Chebshev, com a posição $i = 0$ deslocado $w/2$

Definição 6. *Seja $G(\epsilon, \epsilon', \delta, w)$ um filtro de janela plano, onde G é um vetor simétrico com o vetor suporte $supp(G) \subseteq [-w/2, w/2]$ tal que para todo $i \in [-\epsilon N, \epsilon N]$, temos $\hat{G}_i = 1$*

Portanto, somando 31 filtros Dolph-Chebyshev $G(1/22, 10^{-8}, 133)$ temos um filtro de janela plano com parâmetros $(0.11, 0.0045, 1x10^{-9}, 133)$. Como está mostrado na Figura 3.2.



(a) Escala linear



(b) Escala logarítmica

Figura 3.2: Um filtro de janela plano produzido somando-se 31 filtros Dolph-Chebyshev

O filtro de janela definido acima é utilizado pelo algoritmo que foi estudado. Algoritmos para construir tal filtro possuem complexidade de tempo $O\left(\frac{B}{\alpha}\log(N/\delta)\right)$.

3.3.3 FFT subamostrado

Suponha que temos um vetor $f \in \mathbb{C}^N$ e um parâmetro B que divide N, e gostaríamos de computar $\hat{g}_i = \hat{f}_{i(n/B)}$ para $i \in [B]$.

Teorema 6. *Seja $B \in \mathbb{N}$ que divide N, f um vetor N-dimensional e g um vetor B-dimensional com $g[i] = \sum_{j=0}^{N/B-1} x[i+Bj]$ para $i = 1, \ldots, B$. Então*

$$\hat{g}[i] = \hat{f}[i(n/B)]. \tag{3.5}$$

Demonstração.

$$
\begin{aligned}
\hat{f}[i(n/B)] &= \sum_{j=0}^{N-1} f[j]\omega^{ij(N/B)} \\
&= \sum_{a=0}^{B-1} \sum_{j=0}^{(N/B)-1} f[Bj+a]\omega^{i(Bj+a)N/B} \\
&= \sum_{a=0}^{B-1} \sum_{j=0}^{(N/B)-1} f[Bj+a] \cdot \omega^{iBjN/B} \cdot \omega^{iaN/B} \\
&= \sum_{a=0}^{B-1} \sum_{j=0}^{(N/B)-1} f[Bj+a] \cdot \omega^{ijN} \cdot \omega^{iaN/B} \\
&= \sum_{a=0}^{B-1} \sum_{j=0}^{(N/B)-1} f[Bj+a] \cdot 1 \cdot \omega^{iaN/B} \\
&= \sum_{a=0}^{B-1} g[a]\omega^{iaN/B}
\end{aligned}
\tag{3.6}
$$

Lembrando que $e^{i2\pi x} = 1$, para qualquer $x \in \mathbb{N}$, portanto $\omega^{ijN} = 1$. □

3.4 sFFT versão 1

O algoritmo proposto por Hassanieh et al. [HIKP12b] distribui os coeficientes harmônicos em um pequeno número de *cestos*. A partir destes cestos é feito uma estimativa dos valores de cada coeficiente harmônico $\hat{f}(i)$, assim como sua posição i. Por ser um algoritmo de aproximação probabilístico, executamos algumas partes do algoritmo para que as chances de erro sejam diminuidas.

3.4.1 Divisão em Cestos

O primeiro passo é agrupar os coeficientes de $\hat{f}'$ em B cestos tal que o valor de cada cesto seja a soma de todos os coeficientes do domínio da frequência que foram mapeados para aquele cesto. Este processo é descrito pelo autor como *bucketization*, em uma tradução literal da palavra para o português seria algo parecido com "*cestificação*". Estes cestos são obtidos através da tranformada B-dimensional do sinal permutado, como mostrado no Teorema 6, e filtrado por um filtro G dado pela Definição 6. Em outras palavras, valores permutados do sinal no domínio do tempo são mapeado para um conjunto C de tamanho B. Os *cestos* são obtidos de sua transformada B-dimensional, $\widehat{C}$. Como o sinal sendo analisado é esparso, teremos vários cestos que estarão vazios e podem ser simplesmente descartados.

O processo de dividir em cestos é realizado com o uso de filtros e a escolha deste filtro afeta diretamente o tempo de execução do algoritmo. Idealmente, queremos um filtro que utilize poucas amotras do sinal de entrada no domínio do tempo e possua uma região de passe flat. O filtro que o algoritmo de Hassanieh usa é descrito pela Definição 6, e é baseada em uma multiplicação entre uma função Dolph-Chebyshev com uma função *sinc* no domínio do tempo. Isso nos dá um filtro que decai exponencialmente tanto no domínio do tempo como no domínio

da frequência e por isso temos um vazamento espectral de um cesto para outro desprezível e que pode ser ignorado. Além disso, o filtro é concentrado no domínio do tempo e portanto requer um pequeno número de amostras do sinal de entrada.

Este passo é repetido algumas vezes com diferentes parametros σ da função de permutação, para que coeficientes que podem ter caído no mesmo cesto em uma iteração sejam separados na próxima iteração. Então, a cada iteração $r \in \{1 \dots N\}$ desta divisão de cestos, é gerado um conjunto I_r com os dk maiores valores de $\widehat{C}_r$, onde d é uma constante e k é o número de coeficientes harmônicos significativos.

3.4.2 Localização dos coeficientes

Lembrando que o sinal de entrada f é esparso no domínio da frequência, a divisão em cestos fará com que cada cesto possua apenas um coeficiente grande isolado com uma boa probabilidade. Cada cesto é uma soma de todos os coeficientes do domínio da frequência que foram mapeados para esse cesto. Assim, por termos apenas um coeficiente diferente de zero em cada cesto com boa probabilidade, podemos estimar o valor do coeficiente $\hat{f}(i)$ e então o que temos que encontrar é sua posição i. Para se obter essa posição, o algoritmo utiliza uma abordagem de *pontuação* em que os cestos não-vazios pontuam para todo coeficiente que poderia ser mapeado para aquele cesto. Assim, cada coeficiente diferente de zero receberá um ponto, com *alta probabilidade*, para cada "*cestificação*" feita. Por outro lado, coeficientes que são zero ou desprezíveis, provavelmente terão poucos pontos. Portanto, após repetir o processo algumas vezes, os dk coeficientes que possuirem mais pontos serão os coeficientes que queremos. Na versão do algoritmo que iremos analisar, guardamos em um conjunto I' as posições que aparecerem mais que a metade das iterações feitas, e ficamos com um conjunto de candidatos de tamanho de $O(dkN/B)$.

3.4.3 Estimando valores

O último passo do algoritmo é estimar os valores de cada um dos candidatos encontrados na etapa anterior. Ou seja, para cada iteração realizada, teremos um parâmetro σ_r para a função hash $h_\sigma(i) : [N] \to [B]$ definida como $h_\sigma(i) = round(\sigma i B/N)$. Então, para cada posição $i \in I'$ é obtida a posição $h_{\sigma_r}(i) \in \widehat{C}_r$. Assim, o valor que temos no cesto $h_{\sigma_r}(i)$ é o valor multiplicado pelo filtro, para obter o valor original daquele coeficiente simplesmente divide pelo valor do filtro em uma posição dada por um *offset* $o_\sigma : [N] \to [-N/2B, N/2B]$ definido por $o_{\sigma_r}(i) = \sigma_r i - h_{\sigma_r}(i)(N/B)$. Finalmente, temos uma estimativa do valor do i-ésimo coeficiente dada pela mediana dos valores obtidos em cada iteração r. Vale notar que os valores da parte real e da parte imagiária são calculados separadamente.

3.4.4 O algoritmo

O algoritmo possui dois laços internos. O laço de localização das coordenadas dos coeficientes significativos e o laço de estimação da magnitude das coordenadas localizadas são os laços internos. Cada laço se repete $L = O(\log N)$ vezes. Ao final, ordena as respostas de cada iteração e retorna como resposta a mediana dos valores encontrados. Portanto, dados ϵ e δ, $B = O\left(\sqrt{\frac{Nk}{\epsilon \log(N/\delta)}}\right)$ cestos, $d = O(1/\epsilon)$ e um filtro de janela G dado pela Definição 6, os laços internos são definidos com segue.

1. Escolhe aleatoriamente $\sigma, \tau \in [N]$, onde σ é ímpar.

2. Seja $y = G \cdot (P_{\sigma,\tau}x)$, tal que $y_i = G_i x_{\sigma i+\tau}$. Então, $supp(y) \subseteq supp(G) = [w]$.

3. Computar $\hat{z} = \hat{y}_{i(N/B)}$ para $i \in [B]$. Pela Definição 6, essa é a DFT de $z_i = \sum_{j=0}^{\lceil w/B \rceil} y_{i+jB}$.

4. Seja a função hash $h_\sigma : [N] \rightarrow [B]$, onde $h_\sigma(i) = round(\sigma i B/N)$ e um "offset" $o_\sigma : [N] \rightarrow [-N/(2B), N/(2B)]$, onde $o_\sigma(i) = \sigma i - h_\sigma(i)(N/B)$.

5. Laços de localização: seja J um conjunto que contenha as dk coordenadas dos maiores valores de $\hat{z}$. Cada laço devolve um conjunto $I = \{i \in [N] | h_\sigma(i) \in J\}$, onde $|I| = dkN/B$.

6. Laços de estimação: para cada $i \in I$, estimar $\hat{f}_i$ como $\hat{f}'_i = \hat{z}_{h_\sigma(i)}\omega^{\tau i}/\hat{G}_{o_\sigma(i)}$.

Na primeira etapa, a permutação é dada por uma operação simples de índice e a filtragem e mapeamento do sinal é feito em tempo linear no tamanho do filtro G. Vale notar que o tamanho do filtro é menor que seu espectro, *i.e.*, $supp(G) < supp(\hat{G}) = [w]$. Disso, obtém-se o vetor B-dimensional C_r, que é um subconjunto das amostras do sinal f. E a partir da transformada desse vetor, $\widehat{C_r}$, é devolvido o índice dos dk maiores valores. Pelo Teorema 6, a complexidade de tempo dessa divisão em cestos é $O(|G| + B \log B)$. O algoritmo que executa essa *cestificação* é dado pelo pseudo código do Algoritmo 2.

Algoritmo 2: `cestos(`f`, `G`, `B`, `d`, `τ`, `σ`)`

Dados: Um sinal f, um filtro G, quatro inteiros B, d, τ_r, σ_r
Saída: As $d \cdot k$ coordenadas dos maiores coeficientes de $\widehat{z_r}$

```
1 indice ← τ_r;
2 para i = 0 até |G| faça
3 |   z_r[i mod B] ← z[i mod B] + f[indice] * G[i];
4 |   indice ← (indice + σ) mod N;
5 fim
6 ẑ_r ← fft(z_r);
7 I_r ← maiores_indices(d · k, ẑ_r);
8 retorna I_r, ẑ_r;
```

O Algoritmo 2 executa para todo $r \in \{1, \ldots, L\}$ e ao final teremos um conjunto de coordenadas $I = I_1 \cup I_2 \cup \ldots \cup I_L$ e um conjunto cestos, $\widehat{Z} = \widehat{z_1} \cup \widehat{z_2} \cup \ldots \cup \widehat{z_L}$. O Teorema 7, provado pelo autor em [HIKP12b], limita a probabilidade de cair dois coeficientes grandes no mesmo cesto.

Teorema 7. *Se $j \neq 0$, N é uma potência de 2, e σ é um número uniformemente aleatório ímpar em $[N]$, então,*

$$Pr[\sigma j \in [-x, x]] \leq 4x/N. \tag{3.7}$$

O próximo passo é estimar quais são as coordenadas dos coeficientes significativos. Portanto, dado um conjunto $I_r \in I$, obtemos um conjunto I' com as coordenadas candidatas dos coeficientes significativos. E para cada $i \in I_r$, `localiza()` executa $L = O(\log N)$ vezes e, ao

final, teremos um conjunto $I' = \{i \in I | s_i \geq L/2\}$ com $O(dkN/B)$ coordenadas candidatas, onde $s_i = |\{j | i \in I_r\}|$.

Algoritmo 3: `localiza(`I`, `N`, `d`, `k`, `σ`)`

Dados: Um conjunto I, o tamanho do sinal N, os parâmetros inteiros d, k e σ.
Saída: Um conjunto de coordenadas candidatas I'.

```
1  acertos ← 0
2  para i = 0 até d · k faça
3      comeco ← ⌈(I[i] − 0.5) ∗ N / B⌉ + N  mod N
4      fim ← ⌊(I[i] + 0.5) ∗ N / B⌋ + N  mod N
5      indice ← (comeco ∗ σ)  mod N
6      para j = 0 até fim − comeco faça
7          candidatos[indice]++
8          se candidatos[indice] = s_indice então
9              I'[acertos++] ← indice
10         fim
11         indice ← (indice +σ)  mod N
12     fim
13 fim
14 retorna I'
```

Executando uma função hash "*reversa*", o Algoritmo 3 estima quais coordenadas de $\hat{f}$ são de coeficientes significativos. Então, dada função hash $h_\sigma(i) : [N] \rightarrow [B]$, definida como $h_\sigma(i) = round(\sigma i B/N)$, queremos encontrar o conjunto $I' = \{i \in [N] | h_\sigma(i) \in I\}$. O Teorema 9 limita a probabilidade da coordenada de algum coeficiente significativo não estar em I'.

O último passo é descobrir o valor de cada coeficiente a partir das coordenadas em I'. Dado um offset $o_\sigma(i) : [N] \rightarrow [-N/2B, N/2B]$ definido como $o_\sigma(i) = \sigma i - h_\sigma(i)(N/B)$, obtemos uma estimativa do i-ésimo coeficiente tal que $\hat{f}'[i] = \widehat{z}[h_\sigma(i)]\omega^{\tau i}/\hat{G}[o_\sigma(i)]$, importante

ressaltar que a parte real e a parte imaginária são calculadas separadamente. O erro de estimativa do laço de estimação é limitado pelo Teorema 8.

Algoritmo 4: estima(G, N, Z, σ, τ)

Dados: O filtro G, o tamanho do sinal N, um vetor de candidatos I', o conjunto Z de cestos e os conjuntos σ e τ.
Saída: Os dkN/B maiores coeficientes em $resposta$.

```
1  real ← 0
2  imag ← 1
3  para i = 0 até |I'| faça
4      indice ← 0
5      para r = 0 até L faça
6          indice_perm ← (σ[r] * I'[i]) mod N
7          map_para ← indice_perm / (N/B)
8          offset ← indice_perm mod (N/B)
9          se offset > (N/B)/2 então
10             map_para ← (map_para +1) mod B
11             offset ← offset − N/B
12         fim
13         offset ← (N − offset) mod N
14         valor_filtro ← Ĝ[offset]
15         valores[real][indice] ← creal(Z[map_para] / valor_filtro)
16         valores[imag][indice] ← cimag(Z[map_para] / valor_filtro)
17         indice++
18     fim
19     ordena(valores)
20     mediana ← (loops − 1)/2
21     parte_real ← valores[real][mediana]
22     parte_imag ← valores[imag][mediana]
23     resposta[I'[i]] ← parte_real + j*parte_imag
24 fim
25 retorna resposta
```

O Algoritmo 4 recebe um conjunto $\sigma = \sigma_1 \cup \sigma_2 \ldots \sigma_L$ e um conjunto $\tau = \tau_1 \cup \tau_2 \ldots \tau_L$ dos parâmetros de permutação que definimos na Seção 3.3.

Com a resposta do Algoritmo 4, ordenamos e devolvemos o índice dos k maiores coeficientes achados. Por fim, teremos uma aproximação $\hat{f}'$ satisfazendo a guarantia dada pela Equação 3.8 e é provado no Teorema 11.

$$||\hat{f} - \hat{f}'||_\infty^2 \leq \frac{\epsilon}{k}||\hat{f} - y||_2^2 + \delta||f||_1^2 \tag{3.8}$$

O Algoritmo 5 nos dá o pseudo-código para o algoritmo completo, cuja complexidade de tempo é $O\left(\sqrt{\frac{Nk\log(N/\delta)}{\epsilon}}\log N\right)$, provada no Teorema 10. Assim, o algoritmo inteiro pode ser descrito como o laço externo como segue.

1. Para cada $r \in \{1, \ldots, L\}$, executa um laço de localização obtendo um conjunto de coordenadas I^r.

2. Para cada $i \in I$, onde $I = I_1 \cup I_2 \cup \ldots I_L$, seja $s_i = |\{r|i \in I_r\}|$, ou seja, número de vezes que a coordenada i ocorre nos laços de localização.

3. Considere o conjunto I' que contém as coordendadas que aparecem pelo menos na metade das iterações, *i.e.*,$I' = \{i \in I|s_i \geq L/2\}$.

4. Para cada $r \in \{1, \ldots, L\}$, executa um laço de estimação da magnitude das coordenadas de I' para obter $\hat{f}^r_{I'}$.

5. Para cada $i \in I'$, estimar cada valor como a mediana dos valores encontrados no laço de estimação, *i.e.*, $\hat{f}'_i = mediana(\{\hat{f}^r_i | i \in I'\})$. A mediana é feita para as partes reais e imaginárias separadamente.

3.5 Análise do algoritmo

Os teoremas dados por Hassanieh et. al.[HIKP12b] são apresentados nesta seção. O Teorema 9 determina o limitante de erro dos laços de localização dos coeficientes significativos. O Teorema 8 limita o erro nos laços de estimação da magnitude dos coeficientes localizados. Assim como a complexidade computacional do algoritmo e a garantia que a aproximação $\hat{f}'$ possui.

Teorema 8. *Seja S o vetor suporte dos k maiores coeficientes de $\hat{f}$, e $\hat{f}_{-S}$ contém o resto. Então para um $\epsilon \leq 1$,*

$$Pr\left[|\hat{f}'_i - \hat{f}_i|^2 \geq \frac{\epsilon}{k}||\hat{f}_{-S}||_2^2 + 3\delta^2||\hat{f}||_1^2\right] < O\left(\frac{k}{\epsilon B}\right) \tag{3.9}$$

Demonstração. Sabemos pelos Item 5 que,

$$\hat{f}' = \hat{z}_{h_\sigma(i)}\omega^{-\tau i}/\hat{G}_{o_\sigma(i)} = \hat{y}_{\sigma i - o_\sigma(i)}\omega^{\tau i}/\hat{G}_{o_\sigma(i)}$$

Considere que $\hat{x}$ é zero em todos os pontos exceto no i-ésimo ponto, então $supp(\widehat{P_{\sigma,\tau}x}) = \sigma i$. Então, $\hat{y}$ é a convolução de $\hat{G}$ e $\widehat{P_{\sigma,\tau}x}$ e $\hat{G}$ é simétrico, portanto

$$\begin{aligned}\hat{y}_{\sigma i - o_\sigma(i)} &= (\hat{G} * \widehat{P_{\sigma,\tau}x})_{\sigma i - o_\sigma(i)} \\ &= \hat{G}_{o_\sigma(i)}\widehat{P_{\sigma,\tau}x}_{\sigma i} \\ &= \hat{G}_{o_\sigma(i)}\hat{f}_i\omega^{-\tau i}\end{aligned}$$

neste caso, $\hat{f}'_i = \hat{f}_i$. Mas $\hat{f}'_i - \hat{f}_i$ é linear em $\hat{f}$, então podemos assumir que $\hat{f}_i = 0$ e o limitante $|\hat{f}'_i|$. Uma vez que $|\hat{f}'_i| = |\hat{z}_{h_\sigma(i)}/\hat{G}_{o_\sigma(i)}| < |\hat{z}_{h_\sigma(i)}/(1-\delta)| = \hat{y}_{\sigma i - o_\sigma(i)}/(1-\delta)|$ é suficiente o limitante $|\hat{y}_{\sigma i - o_\sigma(i)}|$. Seja $T = \{j \in [N] | \sigma(i-j) \in [-2N/B, 2N/B]\}$.

$$\begin{aligned}
|\hat{y}_{\sigma i - o_\sigma(i)}|^2 &= \left| \sum_{t=0}^{N-1} (\widehat{P_{\sigma,\tau} x})_t \hat{G}_{\sigma i - t - o_\sigma(i)} \right|^2 \\
&= \left| \sum_{j=0}^{N-1} (\widehat{P_{\sigma,\tau} x})_{\sigma j} \hat{G}_{\sigma(i-j) - t - o_\sigma(i)} \right|^2 \\
&\leq 2 \left| \sum_{j \in T} (\widehat{P_{\sigma,\tau} x})_{\sigma j} \hat{G}_{\sigma(i-j) - t - o_\sigma(i)} \right|^2 + 2 \left| \sum_{j \notin T} (\widehat{P_{\sigma,\tau} x})_{\sigma j} \hat{G}_{\sigma(i-j) - t - o_\sigma(i)} \right|^2 \\
&\leq 2(1+\delta)^2 \left| \sum_{j \in T} (\widehat{P_{\sigma,\tau} x})_{\sigma j} \right|^2 + 2\delta^2 \left| \sum_{j \notin T} (\widehat{P_{\sigma,\tau} x})_{\sigma j} \right|^2
\end{aligned}$$

Neste passo, o termo à esquerda vem de $|\hat{G}_a| \leq 1 + \delta$ para todo a. O termo à direita é por causa de $j \notin T$, $|\sigma(i-j) - o_\sigma(i)| \geq |\sigma(i-j)| - |o_\sigma(i)| > 2N/B - N/(2B) > N/B$, e $|\hat{G}_a| \leq \delta$ para $|a| > N/B$. Pela Definição 4, isso se torna

$$|\hat{y}_{\sigma i - o_\sigma(i)}|^2 \leq 2(1+\delta)^2 \left| \sum_{j \in T} \hat{f}_j \omega^{-\tau j} \right|^2 + 2\delta^2 ||\hat{f}||_1^2.$$

Seja $V = \left|\sum_{j \in T} \hat{f}_j \omega^{-\tau j}\right|^2$. Para uma escolha de τ, V é a energia de um coeficiente aleatório de Fourier do vetor $\hat{f}_T$, então podemos limitar a esperança sobre τ

$$\mathop{\mathbb{E}}_{\tau}[V] = ||\hat{f}_T||_2^2.$$

Agora, para cada coordenada $j \neq i$, $\Pr_\sigma[j \in T] \leq 8/B$ pelo Lemma 7. Portanto, $\Pr_\sigma[S \cap T \neq \emptyset] \leq 8k/B$, então, com probabilidade de $1 - 8k/B$ sobre σ, temos

$$||\hat{f}_T||_2^2 = ||\hat{f}_{T \setminus S}||_2^2.$$

Seja E o evento de que isso ocorra, então E é igual a 0 com probabilidade $8k/B$ e 1 caso contrário. Temos

$$\mathop{\mathbb{E}}_{\sigma,\tau}[E] = \mathop{\mathbb{E}}_{\sigma}[E||\hat{f}_T||_2^2] = \mathop{\mathbb{E}}_{\sigma}[E||\hat{f}_{T \setminus S}||_2^2] \leq \mathop{\mathbb{E}}_{\sigma}[||\hat{f}_{T \setminus S}||_2^2].$$

Além disso, sabemos que

$$\begin{aligned}
\mathop{\mathbb{E}}_{\sigma,\tau}[E] &\leq \mathbb{E}_\sigma[||\hat{f}_{T \setminus S}||_2^2] \\
&= \sum_{j \notin S} |\hat{f}_j|^2 \Pr_\sigma[\sigma(i-j) \in [-2N/B, 2N/B]] \\
&\leq \frac{8}{B} ||\hat{f}_{-S}||_2^2
\end{aligned}$$

pelo Lemma 7. Portanto, pela desigualdade de Markov e pelo limitante da união, temos para qualquer $\lambda > 0$ que

$$\begin{aligned}\Pr_{\sigma,\tau}\left[V > 8\frac{\lambda}{B}||\hat{f}_{-S}||_2^2\right] &\leq \Pr_{\sigma,\tau}[E = 0 \cup E > \lambda \mathbb{E}_{\sigma,\tau}[E]] \\ &\leq 8\frac{k}{B} + \frac{1}{\lambda},\end{aligned}$$

Concluindo

$$\Pr_{\sigma,\tau}\left[|\hat{y}_{\sigma i - o_\sigma(i)}|^2 \geq 16\frac{\lambda}{B}(1+\delta)^2||\hat{f}_{-S}||_2^2 + 2\delta^2||\hat{f}||_1^2\right] < 8\frac{k}{B} + \frac{1}{\lambda}.$$

Substituindo λ por $\epsilon B/(32k)$ e usando $|\hat{f}_i' - \hat{f}_i| < |\hat{y}_{\sigma i - o_\sigma(i)}|/(1+\delta)$ temos

$$\Pr_{\sigma,\tau}\left[|\hat{f}_i' - \hat{f}_i|^2 \geq \frac{\epsilon}{2k}\left(\frac{1+\delta}{1-\delta}\right)^2||\hat{f}_{-S}||_2^2 + \frac{2}{1-\delta}\delta^2||\hat{f}||_1^2\right] < (8 + 32/\epsilon)\frac{k}{B}.$$

que é onde queríamos chegar. □

Teorema 9. *Seja* $E = \sqrt{\frac{\epsilon}{k}||\hat{f}_{-S}||_2^2 + 3\delta^2||\hat{f}||_1^2}$ *como sendo o erro tolerado no Teorema 8. Então para qualquer* $i \in [N]$ *com* $|\hat{f}_i| \geq 4E$*, temos a probabilidade*

$$Pr[i \notin I'] \leq O\left(\frac{k}{\epsilon B} + \frac{1}{\epsilon d}\right) \tag{3.10}$$

Demonstração. Com probabilidade pelo menos $1 - O\left(\frac{k}{\epsilon B}\right)$, o valor da aproximação $|\hat{f}_i'|$ é $|\hat{f}_i'| \geq |\hat{f}_i| - E \geq 3E$. Neste caso, $|\hat{z}_{h_\sigma(i)}| \geq 3(1-\delta)E$. Portanto, é suficiente mostrar que, com probabilidade pelo menos $1 - O\left(\frac{1}{\epsilon d}\right)$, existe no máximo dk posições $j \in [B]$ com $|\hat{z}_j| \geq 3(1-\delta)E$. Uma vez que cada $\hat{z}_j$ corresponde a N/B posições em $[N]$, mostraremos que com probabilidade pelo menos $1 - O\left(\frac{1}{\epsilon d}\right)$, existe no máximo dkN/B posições $j \in [N]$ com $|\hat{f}_i'| \geq 3(1-\delta)^2E$.

Seja $U = \{j \in [N] || \hat{f}_j| \geq E\}$, e $V = \{j \in [N] || \hat{f}_j' - \hat{f}_j| \geq E\}$. Então, $\{j \in [N] || \hat{f}_j'| \geq 2E\} \subseteq U \cup V$. Já que $2 \leq 3(1-\delta)^2$, temos

$$|\{j || \hat{f}_j'| \geq 3(1-\delta)^2E\}| \leq |U \cup V|.$$

Também sabemos que

$$|U| \leq k + \frac{||f_{-S}||_2^2}{E^2} \leq k(1 + 1/\epsilon)$$

e pelo Lemma 8

$$\mathbb{E}[|V|] \leq O\left(\frac{kN}{\epsilon B}\right).$$

Portanto, pela desigualdade de Markov $\Pr[|V| \geq dkN/B] \leq O\left(\frac{1}{\epsilon d}\right)$, ou

$$\Pr[|U \cup V| \geq dkN/B + k(1 + 1/\epsilon)] \leq O\left(\frac{1}{\epsilon d}\right)$$

Já que o lado direito da equação anterior é apenas significativo para $d > 1/\epsilon$, temos $kdN/B > k(1 + 1/\epsilon)$. Portanto

$$\Pr[|U \cup V| \geq dkN/B] \leq O\left(\frac{1}{\epsilon d}\right)$$

e portanto

$$\Pr[|\{j \in [B]||\hat{z}_j| \geq 3(1 - \delta)E\}| > kd] \leq O\left(\frac{1}{\epsilon d}\right)$$

E assim, com o *limitante da união* sobre isso e a probabilidade que $|\hat{f}'_j - \hat{f}_j| < E$ nos dá

$$\Pr[i \notin I] \leq O\left(\frac{k}{\epsilon B} + \frac{1}{\epsilon d}\right)$$

como desejado. □

Teorema 10. *Executando o Algoritmo 5 com parâmetros $\epsilon, \delta < 1$, ficamos com um $\hat{f}'$ satisfazendo*

$$|\hat{f}'_i - \hat{f}_i|^2 \geq \frac{\epsilon}{k}||\hat{f}_{-S}||_2^2 + 3\delta^2||\hat{f}||_1^2 \tag{3.11}$$

com probabilidade $1 - 1/N$ e tempo de execução $O\left(\sqrt{\frac{Nk\log(N/\delta)}{\epsilon}}\log N\right)$.

Demonstração. Para analisar o algoritmo, note que

$$|I'|\frac{L}{2} \leq \sum_i s_i = \sum_r |I_r| = LdkN/B$$

ou também $|I'| \leq 2dkN/B$. Então o tempo de execução dos laços de localização e dos laços de estimação é $O\left(B\log\frac{N}{\delta} + dkN/B\right)$. Computar I' e as medianas leva tempo linear, ou seja $O(dkN/B)$. Portanto o tempo total de execução do algoritmo é $O(LB\log N\delta + LdkN/B)$. Considerando que

$$B = O\left(\sqrt{\frac{Nk}{\epsilon\log N/\delta}}\right)$$

e

$$d = O\left(\frac{1}{\epsilon}\right)$$

o tempo de execução do algoritmo é

$$O\left(\sqrt{\frac{Nk}{\epsilon\log N/\delta}}\log N\right)$$

□

Teorema 11. *Executando o algoritmo com parametros* $\epsilon, \delta < 1$ *temos* $\hat{f}'$ *satisfazendo*

$$||\hat{f}' - \hat{f}||_\infty^2 \leq \frac{\epsilon}{k}||\hat{f}_{-S}||_2^2 + \delta^2||\hat{f}||_1^2$$

com probabilidade $1 - 1/N$ *e tempo de execução* $O\left(\sqrt{\frac{Nk}{\epsilon \log N/\delta}} \log N\right)$

Demonstração. Seja o erro

$$E = \sqrt{\frac{\epsilon}{k}||\hat{f}_{-S}||_2^2 + 3\delta^2||\hat{f}||_1^2},$$

Considerando o Lemma 9, em cada iteração r, para qualquer i com $|\hat{f}_i| \geq 4E$, temos

$$\Pr[i \notin I^r] \leq O\left(\frac{k}{\epsilon B} + \frac{1}{\epsilon d}\right).$$

Portanto, $\mathbb{E}[s_i] \geq 3L/4$, e cada iteração é um evento independente. Então, pelo limitante de Chernoff a probabilidade de $s_i < L/2$ é no máximo $1/2^{\Omega(L)} < 1/N^3$. Então, pelo limitante da união, com probabilidade pelo menos $1 - 1/N^2$, $i \in I'$ para todo i com $|\hat{f}_i| \geq 4E$.

Em seguida, pelo Lemma 8, temos para cada iteração r de estimação e índice i,

$$\Pr[|\hat{f}_i^r - \hat{f}_i| \geq E] \leq O\left(\frac{k}{\epsilon B}\right) < \frac{1}{4}.$$

Portanto, com probabilidade $1 - 2^{-\Omega(L)} \geq 1 - 1/N^3$, $|\hat{f}_i^r - \hat{f}_i| \leq E$ em pelo menos $2L/3$ da iterações.

Uma vez que a parte real $real(\hat{f}_i')$ é a mediana da parte real $real(\hat{f}_i^r)$, deve existir duas iterações r tal que $|\hat{f}_i^r - \hat{f}_i| \leq E$, porém com um $real(\hat{f}_i^r)$ acima de $real(\hat{f}_i')$, e um abaixo. Com isso, em uma dessas iterações r temos $|real(\hat{f}_i' - \hat{f}_i)| \leq |real(\hat{f}_i^r - \hat{f}_i)| \leq E$, o mesmo ocorre para a parte imaginária. Então

$$|\hat{f}_i' - \hat{f}_i| \leq \sqrt{2}\max(|real(\hat{f}_i' - \hat{f}_i)|), |imag(\hat{f}_i' - \hat{f}_i)| \leq \sqrt{2}E.$$

Pelo limitante da união sobre I', com probabilidade pelo menos $1 - 1/N^2$, temos $|\hat{f}_i' - \hat{f}_i| \leq \sqrt{2}E$ para todo $i \in I'$. Já que para todo $i \notin I'$ temos $\hat{f}_i' = 0$ e $|\hat{f}_i| \leq 4E$ com probabilidade de $1 - 2/N^2$, ficamos com

$$||\hat{f}' - \hat{f}||_\infty^2 \leq 16E^2 = 16\frac{\epsilon}{k}||\hat{f}_{-S}||_2^2 + 48\delta^2||\hat{f}||_1^2.$$

Redimensionando ϵ e δ, chegamos onde queríamos. □

4 Experimentos sobre imagens

Nesta seção apresentamos os experimentos realizados em um conjunto de imagens pequenas. Neste trabalho foi utilizado a base de imagens *CIFAR* [Kri09]. A base possui 60.000 imagens coloridas de tamanho 32x32 igualmente divididas em 10 categorias, onde 50.000 são imagens de treinamento e as outras 10.000 são imagens de teste. Estas imagens estão dividas em 6 arquivos binários, onde cada imagem possui o primeiro byte para indicar a categoria, os 1024 bytes seguintes são valores de vermelho, os 1024 seguintes são valores de verde e analogamente, para os valores de azul. A Figura 4.1 representa uma imagem de cada categoria.

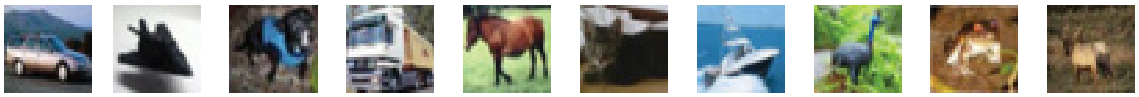


Figura 4.1: Uma imagem de cada categoria: automóvel, avião, cachorro, caminhão, cavalo, gato, navio, pássaro, sapo e veado.

As imagens foram processadas em uma máquina Linux Mint Cinnamon 19 com processador Intel® i7-8750H com 6 núcleos de 2,20GHz e 12 threads lógicas, 15, 3 Gb de RAM de frequência 2666MHz. Processar 5.000 imagens de cada categoria do conjunto de dados de treino levou em média 31 minutos. Para verificar se as imagens são esparsas ou não, foi utilizado o algoritmo *FFTW* [FJ05] que é uma implementação do algoritmo da FFT. O algoritmo de aproximação utilizado para a última etapa dos experimentos é a implementação do algoritmo apresentado na Seção 3.4. A implementação é disponibilizada pelo próprio autor.

4.1 Proposta dos experimentos

A transformada de Fourier já se provou eficiente para reconhecimento de padrões com o algoritmo publicado por Wang [cWF03]. O algoritmo consegue reconhecer uma música com apenas um pequeno trecho de áudio realizado em um microfone pequeno dentro de um ambiente com muito ruído. Não só o algoritmo consegue um resultado com poucos falsos-positivos como também consegue retornar uma resposta rápida, considerando que existe uma base de dados grande para comparar os resultados. O algoritmo utiliza os coeficientes mais significativos da música para determinar um *descritor* dela. Sabendo que podemos utilizar um algoritmo parecido com esse, basta saber se usar o espectro de Fourier é vantajoso, *i.e.*, se a imagem é realmente esparsa e se o processamento dessas imagens utilizando a *SFFT* é, na prática, mais eficiente.

Uma das propostas dos experimentos realizados neste trabalho é de analisar a possibilidade de obter classificação de uma imagem a partir do espectro dela. Sabendo que existem algoritmos que aproximam a resposta da transformada de Fourier em tempo sublinear, obter o espectro seria mais eficiente que analisar cada pixel da imagem. Portanto, o objetivo principal destes experimentos é comparar imagens em ambos formatos, *i.e.*, o domínio do espaço com o domínio da frequência. Utilizando a definição de energia do Teorema de Parseval, podemos

mostrar que boa parte da informação de uma imagem está contida em poucos coeficientes do domínio da frequência.

O espectro das imagens foram obtidos através de uma implementação da *FFT*, a *FFTW* [FJ05]. Portanto, o objetivo de processar essas imagens com a *FFT* é determinar *quão* esparso a imagem realmente é. Além disso, o algoritmo disponível para uso da *SFFT* é utilizado para sinais de uma dimensão, portanto foi realizado um terceiro experimento para considerar a possibilidade de um mapeamento entre o *1D* e o *2D*. E por fim, utilizamos as imagens mapeadas para o *1D* no algoritmo da *SFFT* para obter dados de performance da execução do algoritmo da transformada esparsa em comparação ao algoritmo da FFT, apresentando o tempo de execução além do erro obtido na aproximação feita pelo algoritmo.

Para determinar a esparsidade da imagem, não estaremos interessados na performance desses experimentos, apenas na relação de energia nos coeficientes do espectro. Então, foram realizados dois tipos de testes para determinar quão esparsas são as imagens utilizando a *FFTW*. Em um dos testes, comparamos a energia do espectro obtido sem o uso de um filtro de janela e no outro, com o uso de filtro de janela Dolph-Chebyshev. Além das imagens pequenas disponibilizadas pela base *CIFAR*, foram utilizadas imagens maiores de tamanho 1024x1024. Vale ressaltar que as imagens foram convertidas para tons de cinza em uma propoção de 30% de vermelho, 60% de verde e 10% de azul.

4.2 Esparsidade das imagens pequenas

Utilizando a definição de energia, podemos comparar a energia do espectro completo com a energia do espectro com um número arbitrário de coeficientes a fim de determinar quanto que o espectro é esparso. Para imagens CIFAR, temos imagens de tamanho $32x32$ e portanto, o espectro completo dessas imagens possuem um tamanho de $K = 1024$.

A Tabela 4.1 mostra o resultado da energia total do sinal comparada com a energia do sinal com apenas um subconjunto dos coeficientes. Neste caso, estamos analisando a porcentagem da energia contida para $k = o(N)$. A imagem resultante utilizando apenas esses coeficientes é mostrada na Figura 4.2.

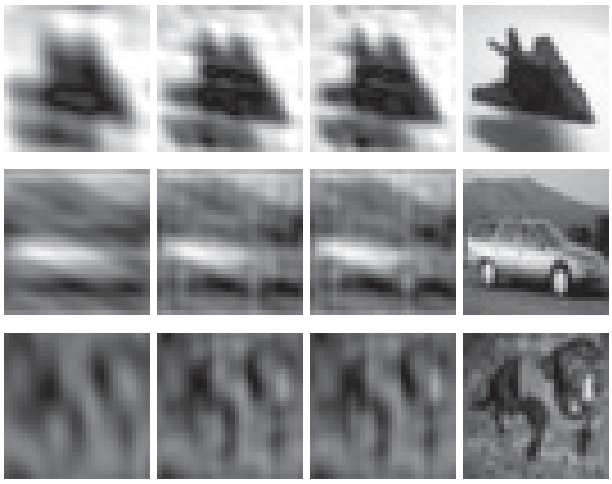


Figura 4.2: Imagens recuperadas pela transformada inversa com apenas uma parte dos coeficientes. Na ordem, foram usados $k = 32$, $k = 64$ e $k = 96$ coeficientes. Por último, a imagem completa.

Categoria	$k = \sqrt{N}$	$k = 2\sqrt{N}$	$k = 3\sqrt{N}$
Automóvel	94,9%	96,8%	98,3%
Avião	97,4%	98,5%	99,2%
Cachorro	96,1%	97,7%	98,9%
Caminhão	95,4%	97,1%	98,5%
Cavalo	95,7%	97,4%	98,6%
Gato	95,9%	97,6%	98,8%
Navio	97,3%	98,4%	99,2%
Pássaro	97,0%	98,1%	99,0%
Sapo	95,7%	97,1%	98,4%
Veado	96,9%	98,0%	98,9%

Tabela 4.1: A porcentagem da energia total para imagens com $k = o(N)$ coeficientes.

Dos resultados mostrados na Tabela 4.1, podemos concluir que aproximadamente 10% (ou seja, $k = 96$) dos coeficientes do espectro de qualquer imagem, corresponde a pelo menos 98% de toda energia do sinal. Na Figura 4.2, é possível observar que as imagens recuperadas pela transformada inversa usando apenas um subconjunto dos coeficientes possuem falhas, mas ainda é possível distinguir o objeto na imagem. A Tabela 4.2 mostra que 99% da energia total da imagem está contida, na média, em 14% dos coeficientes de Fourier daquela imagem. E mesmo para alcancar 99, 9% da energia da imagem, estamos usando, na média, 48% dos coeficientes, em alguns casos, como da categoria *avião*, cerca de 41%.

Categoria	**99,0%**	**99,5%**	**99,9%**
Automóvel	193	297	549
Avião	100	179	420
Cachorro	142	230	483
Caminhão	176	274	518
Cavalo	163	259	504
Gato	151	245	493
Navio	105	186	428
Pássaro	123	216	470
Sapo	199	314	582
Veado	132	228	487

Tabela 4.2: A maior parte da energia de uma imagem está contida apenas na metade dos coeficientes. Cada célula da tabela indica a quantidade de coeficientes necessários para alcançar a porcentagem (indicada na coluna) desejada.

Outra observação importante que deve ser feita é a energia contida no primeiro coeficiente, *i.e.*, o coeficiente que representa a constante do sinal. Mesmo que seja comum a constante possuir uma energia significativa, os resultados dos experimentos mostram que é possível que os valores dos coeficientes tenham sofrido influência de vazamento espectral. A Tabela 4.3 mostra a energia relativa do primeiro coeficiente. Podemos observar que grande parte da energia está dentro de um coeficiente apenas. Portanto, na próxima seção, será feita a mesma análise do espectro das imagens e sua esparsidade no domínio da frequência com o uso de filtros de janela.

Categoria	k=1
Automóvel	78,9%
Avião	88,2%
Cachorro	81,9%
Caminhão	79,7%
Cavalo	82,5%
Gato	81,2%
Navio	85,0%
Pássaro	87,4%
Sapo	84,8%
Veado	87,0%

Tabela 4.3: A energia média do primeiro coeficiente do espectro de cada categoria

4.3 Funções de janela 2D

Os experimentos realizados na seção anterior foram executados sem o uso de filtros de janela. Portanto, assume-se que os experimentos possam ter sido afetados pelo *vazamento espectral*. Então, outra etapa do experimento foi construir uma função de janela e adaptá-la para duas dimensões. Usando a técnica mais simples, a função de janela escolhida foi a *Dolph-Chebyshev* com parâmetros $(1/5.5, 10^{-8}, 32)$.

Para adaptar a função *Dolph-Chebyshev* para duas dimensões, basta computar o *produto diádico*. O produto diádico $(u \oplus v)$ de dois vetores u e v pode ser entendido como o produto matricial de u e a transposta de v (i.e. uv^T), resultando em uma matriz. A função de janela *Dolph-Chebyshev* de duas dimensões com tamanho 32x32 é representado graficamente na Figura 4.3.

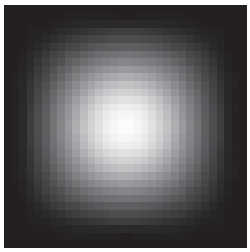


Figura 4.3: Uma imagem que representa graficamente a função *Dolph-Chebyshev* de duas dimensões 32x32.

4.4 Esparsidade com uso de filtro de janela

Na teoria, sabemos que a transformada de Fourier pode gerar o fenômeno do vazamento espectral. Portanto, os mesmos experimentos para obter uma estimativa da esparsidade das imagens de cada categoria foram executadas após o uso de filtros de janelas. Assim, teremos resultados com efeitos do vazamento espectral reduzidos.

Neste caso, podemos esperar que o número de coeficientes necessários para obter 99, 9% da energia total será maior. Isso se deve ao fato de que a energia dos maiores coeficientes terão uma amplitude menor, uma vez que as influências de frequências vizinhas são reduzidas com o uso do filtro de janela.

A Tabela 4.4 mostra o resultado da energia total do sinal comparada com a energia do sinal com apenas um subconjunto dos coeficientes. Como anteriormente, estamos analisando a porcentagem da energia contida em $k = o(N)$ coeficientes.

Categoria	$k = \sqrt{N}$	$k = 2\sqrt{N}$	$k = 3\sqrt{N}$
Automóvel	93,0%	96,2%	98,1%
Avião	93,4%	96,5%	98,3%
Cachorro	52,9%	65,1%	76,9%
Caminhão	92,8%	95,8%	97,8%
Cavalo	46,0%	60,4%	72,1%
Gato	94,1%	96,7%	98,4%
Navio	95,2%	97,5%	98,7%
Pássaro	94,3%	96,9%	98,5%
Sapo	95,4%	97,1%	98,4%
Veado	94,4%	96,9%	98,5%

Tabela 4.4: A porcentagem da energia total para a imagem com $k = o(N)$ coeficientes

A Tabela 4.4 mostra que em algumas categorias, tais como *automóvel* e *veado*, temos uma diferença bem pequena e para a categoria *sapo*, quase não há diferença. Lembrando que o vazamento espectral é quando a energia de uma frequência se *"espalha"* para as frequências vizinhas, então um dos motivos para que estas categorias não sejam afetadas pelo uso de filtro de janela, é que sinais dessas categorias são esparsos.

Ainda a partir da Tabela 4.4, podemos concluir que nas categorias *cachorro* e *cavalo* o vazamento espectral é maior, necessitando de praticamente todos os coeficientes para obter os mesmos $99,9\%$ da energia total do sinal. Isto pode significar que temos muito mais coeficientes que são grandes, e portanto temos muitas frequências *"vazando"* entre si. E isso é uma evidência de que temos sinais menos esparsos nestas categorias.

A Tabela 4.5 mostra que 99% da energia total da imagem está contida, na média, em 31% (18% se desconsiderar as categorias *cachorro* e *cavalo*) dos coeficientes. E para alcancar $99,9\%$ da energia da imagem, estamos usando, em média, 61% (52% se desconsiderar as categorias *cachorro* e *cavalo*).

Categoria	**99,0%**	**99,5%**	**99,9%**
Automóvel	207	312	567
Avião	187	284	534
Cachorro	899	947	999
Caminhão	227	331	573
Cavalo	918	959	1003
Gato	186	287	541
Navio	151	245	497
Pássaro	170	265	519
Sapo	195	310	582
Veado	178	277	528

Tabela 4.5: Utilizando filtro de janela, é necessário mais coeficientes para atingir $99,9\%$ da energia total. Cada célula da tabela indica a quantidade de coeficientes necessários para alcançar a porcentagem (indicada na coluna) desejada.

A Tabela 4.6 mostra a energia relativa do primeiro coeficiente após o uso de filtros. Neste caso, podemos observar que a energia da primeira frequência é consideravelmente menor com o uso de filtros. Isto ocorre para todas as categorias. Portanto, podemos reafirmar que o uso de filtros reduz o vazamento espectral, como gostaríamos de demonstrar.

Categoria	**k=1**
Automóvel	16,4%
Avião	17,7%
Cachorro	13,3%
Caminhão	16,9%
Cavalo	11,4%
Gato	17,4%
Navio	17,4%
Pássaro	18,2%
Sapo	17,3%
Veado	17,9%

Tabela 4.6: A energia média do primeiro coeficiente do espectro de cada categoria

Por fim, as coordenadas mais significativas utilizando o uso de filtros são diferente quando não se usa o filtro. Porém, esta diferença geralmente é pequena, *i.e.*, apesar de diferentes, a distância entre elas são pequenas. Isto é outra evidência que sem o uso de filtro de janela de duas dimensões temos coeficientes alterados pelo vazamento espectral.

4.5 Utilizando imagens maiores

No algoritmo da transformada esparsa de Fourier, os resultados teóricos são obtidos utilizado um sinal de tamanho $N = 2^{22}$, portanto, nesta seção será mostrado os resultados dos mesmos experimentos da seção anterior. O conjunto de imagens é consideravelmente menor que o conjunto de imagens pequenas. Obtidas através de um sítio na Internet de papel de parede, foi coletado 10 imagens de cada caategoria com tamanho de 1024x1024, totalizando um tamanho de sinal de $N = 2^{20}$. A Figura 4.4 representa a reconstrução de três imagens de três categorias quaisquer, lembrando que foi utilizado um número de coeficientes $k = o(N)$.

Figura 4.4: Imagens recuperadas pela transformada inversa com apenas uma parte dos coeficientes. Na ordem, com 1024, 2048 e 3072 coeficientes. Por último, a imagem completa.

Apesar das imagens maiores possuírem uma resolução melhor, o resultado das imagens esparsas são menos nítidas do que as imagens menores como podemos observar na Figura 4.4. A porcentagem de energia contida nos $k = o(N)$ coeficientes das imagens grandes é semelhante à energia contida nas imagens pequenas na mesma proporção como pode ser observado na Tabela 4.7. A Tabela 4.7 mostra o resultado da energia total do sinal comparada com a energia do sinal com apenas um subconjunto dos coeficientes, também o resultado com o uso do filtro de janela. A diferença do percentual de energia contida nos mesmo subconjuntos de coeficientes é pequena.

	Sem filtro			Com filtro		
Categoria	$\sqrt{N}$	$\mathbf{2}\sqrt{N}$	$\mathbf{3}\sqrt{N}$	$\sqrt{N}$	$\mathbf{2}\sqrt{N}$	$\mathbf{3}\sqrt{N}$
Automóvel	95,8%	96,8%	97,7%	94,7%	96,8%	97,1%
Avião	98,5%	98,9%	99,2%	96,3%	97,2	97,9%
Cachorro	98,5%	98,8%	99,1%	97,7%	98,1%	98,5%
Caminhão	94,7%	95,9%	96,9%	92,4%	94,2%	95,6%
Cavalo	97,9%	98,4%	98,8%	97,1%	97,9%	98,5%
Gato	98,6%	99,0%	99,2%	98,2%	98,5%	98,8%
Navio	95,7%	96,5%	97,2%	94,3%	95,5%	96,5%
Pássaro	98,0%	98,4%	98,8%	92,2%	93,7%	95,1%
Sapo	97,6%	98,3%	98,8%	97,3%	98,0%	98,6%
Veado	96,9%	97,5%	98,1%	95,9%	96,7%	97,5%

Tabela 4.7: A porcentagem da energia total para imagem com $k = o(N)$ coeficientes das imagens grandes.

A Tabela 4.8 mostra a energia relativa do primeiro coeficiente para as imagens grandes com e sem o uso de filtro. Novamente, os resultados são semelhantes aos dados das imagens pequenas.

	k=1	
Categoria	**Sem filtro**	**Com filtro**
Automóvel	68,5%	12,1%
Avião	86,1%	14,4%
Cachorro	82,5%	15,6%
Caminhão	69,6%	13,5%
Cavalo	79,2%	14,6%
Gato	76,6%	15,0%
Navio	70,5%	11,9%
Pássaro	83,6%	15,2%
Sapo	75,9%	14,7%
Veado	84,3%	15,4%

Tabela 4.8: A energia média do primeiro coeficiente do espectro de cada categoria com e sem o uso de filtro.

4.6 Esparsidade para imagens com mapeamento de Hilbert

Apesar de Giuseppe Peano ter sido o primeiro a descobrir as curvas de preenchimento de espaço, Hilbert foi o primeiro a apresentar um procedimento geométrico que permitisse a construção dessa classe de curva de preenchimento de espaço. A curva de Hilbert é um mapeamento entre espaços de duas dimensões para uma dimensão que preserva a propriedade

de localidade [MJFS01]. Portanto, outro experimento realizado foi executar os experimentos anteriores utilizando as imagens no formato de Hilbert, *i.e.*, imagens onde seus pixels estão conectados utilizando o mapeamento de Hilbert, a Figura 4.5 mostra como os pixels são conectados no formato de Hilbert. Como estamos fazendo um mapeamento entre os espaços de duas e uma dimensão, os espectro são obtidos utilizando a transformada de Fourier de uma dimensão.

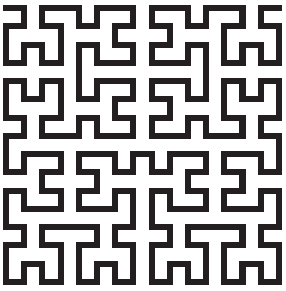


Figura 4.5: A ordem em que os pixels estariam conectados para obter um sinal de uma dimensão.

Sabendo que uma imagem é uma variação no espaço podemos utilizar isso para reduzir a dimensão da imagem e processar a imagem como um sinal de uma dimensão. Considerando que o objetivo de transformar a imagem em uma curva de Hilbert é para poder utilizar a imagem no algoritmo da transformada esparsa de Fourier que temos, então basta saber se é possível utilizar esse mapeamento sem perdas.

Utilizando a imagem no formato de Hilbert, obtemos seu espectro, e a partir desse espectro, selecionamos os $k = o(N)$ maiores coeficientes e zeramos o resto, como foi feito nos experimentos anteriores. Aplicando a transformada inversa de Fourier, teremos a imagem correspondente no formato de Hilbert. Então, revertendo mapeamento de Hilbert, temos as imagens mostradas na Figura 4.6. As imagens são as mesmas imagens na Figura 4.2, porém os maiores coeficientes foram obtidos do espectro da imagem no formato de Hilbert.

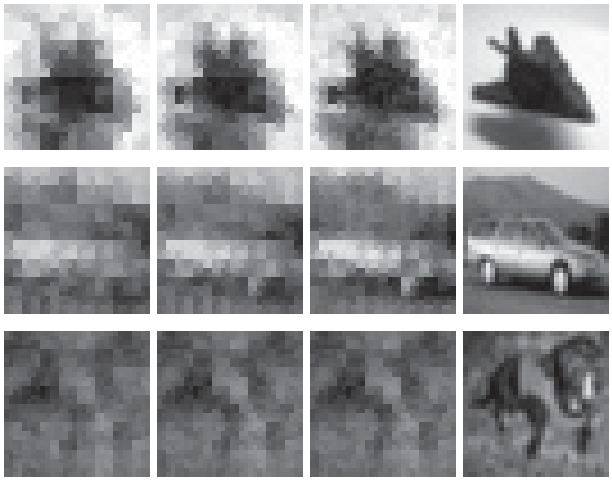


Figura 4.6: As imagens são semelhantes do processo anterior, porém um pouco menos borradas.

Nas imagens pequenas mostradas na Figura 4.6, podemos observar que temos imagens mais granuladas que anteriormente, mas ainda é possível distinguir o objeto na imagem. Na Figura 4.7, temos imagens esparsas com aspecto granulado como nas imagens menores, porém estão mais nítidas que as imagens da Figura 4.4.

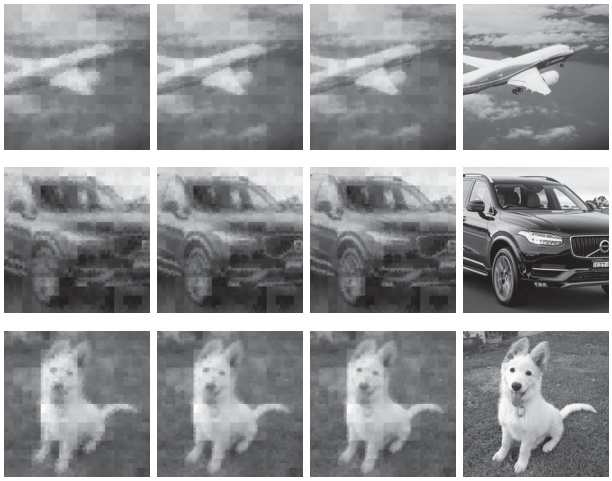


Figura 4.7: As imagens maiores possuem uma nitidez melhor quando se obtém os coeficientes no formato de Hilbert.

Considerando a energia dos coeficientes, temos dados parecidos com os levantados anteriormente. Com isso, é possível concluir que a energia relativa do sinal inteiro está principalmente em um subconjunto de coeficientes. A Tabela 4.9 mostra a energia relativa para as imagens pequenas, e na Tabela 4.10, temos a energia relativa para as imagens grandes.

	Sem filtro			**Com filtro**		
Categoria	$k = \sqrt{N}$	$k = 2\sqrt{N}$	$k = 3\sqrt{N}$	$k = \sqrt{N}$	$k = 2\sqrt{N}$	$k = 3\sqrt{N}$
Automóvel	93.1%	95.3%	97.0%	90.6%	93.5%	96.0%
Avião	96.4%	97.5%	98.4%	94.2%	96.0%	97.5%
Cachorro	94.7%	96.5%	97.9%	94.5%	96.3%	97.8%
Caminhão	94.0%	95.9%	97.4%	90.3%	93.3%	95.8%
Cavalo	94.4%	96.1%	97.6%	93.5%	95.4%	97.1%
Gato	94.7%	96.4%	97.8%	94.2%	96.1%	97.6%
Navio	96.2%	97.4%	98.4%	93.6%	95.6%	97.3%
Pássaro	95.9%	97.2%	98.3%	94.6%	96.3%	97.7%
Sapo	94.6%	96.1%	97.5%	93.4%	95.3%	97.0%
Veado	96.1%	97.3%	98.3%	95.1%	96.7%	97.9%

Tabela 4.9: A relação de energia para um subconjunto dos coeficientes das imagens pequenas no formato de Hilbert.

	Sem filtro			Com filtro		
Categoria	$k = \sqrt{N}$	$k = 2\sqrt{N}$	$k = 3\sqrt{N}$	$k = \sqrt{N}$	$k = 2\sqrt{N}$	$k = 3\sqrt{N}$
Automóvel	92.5%	93.9%	95.2%	96.4%	87.5%	89.7%
Avião	98.0%	98.5%	98.8%	96.3%	97.1%	97.8%
Cachorro	98.0%	98.5%	98.8%	97.6%	98.1%	98.5%
Caminhão	92.9%	94.2%	95.3%	88.4%	90.6%	92.6%
Cavalo	97.1%	97.8%	98.3%	95.2%	96.4%	97.3%
Gato	98.1%	98.6%	99.0%	98.0%	98.5%	98.9%
Navio	94.2%	95.2%	96.0%	92.0%	93.3%	94.5%
Pássaro	97.5%	97.9%	98.3%	95.6%	96.5%	97.3%
Sapo	96.4%	97.4%	98.1%	96.4%	97.4%	98.2%
Veado	96.3%	97.0%	97.6%	95.2%	96.0%	96.8%

Tabela 4.10: A relação de energia para imagens grandes no formato de Hilbert.

4.7 Utilizando o algoritmo da SFFT

O algoritmo baseado na transformada esparsa disponibilizado por Hassanieh et al. [HIKP12b] foi projetado para obter os coeficientes de sinais exatamente esparsos. Portanto, ao utilizar um sinal que seja aproximadamente esparso, espera-se que os erros sejam maiores que os provados. Sendo assim, aplicando uma redução de dimensão dos sinais de imagem para que fosse possível utilizar esse algoritmo, podemos executar o algoritmo com um sinal aproximadamente esparso.

Seja x o vetor exatamente k esparso do sinal, *i.e.*, com apenas os k maiores coeficientes mantidos e os restantes zerados, desejamos saber quão distante a resposta do algoritmo $\hat{f}'$ está de x. Inicialmente, utilizando $k = \log N$ como parametro, um número de laços $L = O(\log N) = 32$ e uma tolerancia $\delta = 10^{-3}$. Sabendo que $B = O\left(\sqrt{\frac{Nk}{\log N}}\right) = c\left(\sqrt{\frac{Nk}{\log N}}\right)$, teremos $B_{loc} = 128$ e $B_{est} = 64$, considerando a constante $c_{loc} = 4$ e $c_{est} = 2$, para os laços de localização e estimação, respectivamente. Assim, teremos um filtro plano para localização dado por $(1/128, 1/256, 10^{-3}, 827)$ e um filtro plano para estimação dado por $(1/64, 1/128, 10^{-3}, 551)$. Seja $d = 2$, *teoricamente* teremos um conjunto de candidatos I, tal que $|I| = dkN/B_{loc} = 2 \cdot 32 \cdot 1024/128 = 512$. A imagem recuperada pelo algoritmo usando os parametros descritos anteriormente é mostrada na Figura 4.8.

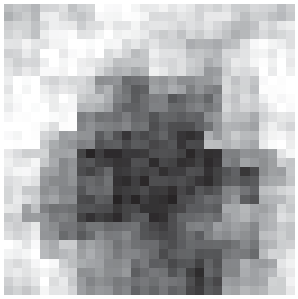


Figura 4.8: A recuperação da primeira imagem de avião buscando $k = 32$ coeficientes.

Considerando a comparação do espectro aproximado obtido $\hat{f}'$ com o espectro exato x, foi verificado que dos 32 coeficientes que estão em x, apenas 28 deles foram encontrados pelo algoritmo da *SFFT* a partir de 506 candidatos encontrados. Além disso, o erro calculado é

pequeno, porém não é tão pequeno como provado no Teorema 8, ou seja, temos $|x - \hat{f'}|_1 = 0,85$, um erro de aproximadamente $0,02$ por frequência. A Figura 4.9 mostra o espectro exato x e o espectro achado $\hat{f'}$. Podemos observar que os coeficientes encontrados não estão muito distantes do espectro exato.

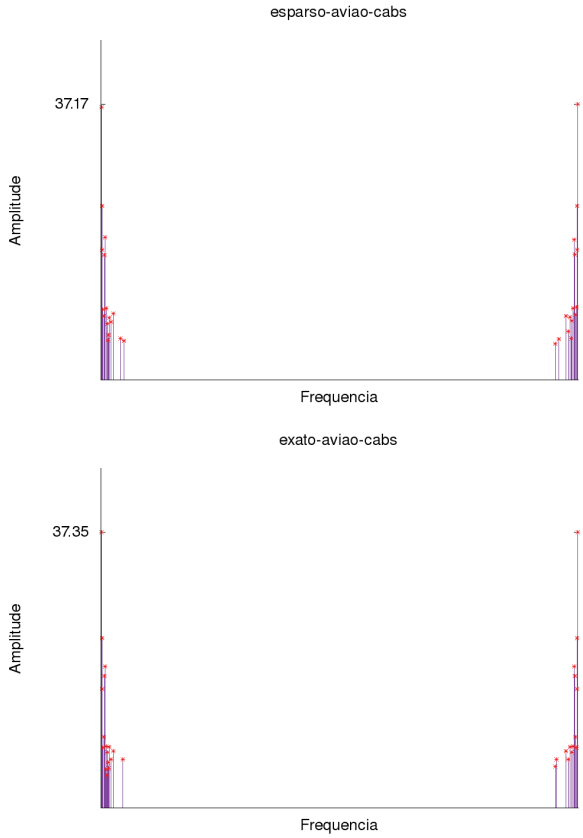


Figura 4.9: Em cima, o espectro $\hat{f'}$ recuperado pela *SFFT* e em baixo o expectro exato x.

O tempo de execução do algoritmo *SFFT* é muito maior neste caso. Para recuperar exatamente o espectro utilizando a *FFT*, gastamos $28x10{-}6$ segundo. A execução do algoritmo da *SFFT* foi de $10x10^{-3}$ segundo.

Considerando imagens maiores, o resultado do algoritmo não consegue encontrar na etapa de localização os coeficientes corretos, e portanto possui erros muito grandes de estimação. As imagens grandes de tamanho $N = 1048576$ queremos encontrar os $k = \log N = 1024$ maiores coeficientes. Utilizando parametros parecidos, com $L = 32$ e $\delta = 10^{-6}$, ficamos com $B_{loc} = 8192$ e um filtro de localização dado por $(1/8192, 1/16384, 10^{-6}, 70909)$ e $B_{est} = 2048$ com filtro de estimação dado por $(1/2048, 1/4096, 10^{-6}, 17723)$. Seja $d = 2$, *teoricamente* teremos um conjunto de candidatos I, tal que $|I| = dkN/B_{loc} = 2 \cdot 1024 \cdot 1048576/8192 = 262144$. A imagem recuperada pelo algoritmo usando os parametros descritos anteriormente é mostrada na Figura 4.10.

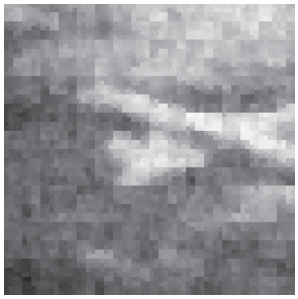


Figura 4.10: A recuperação da primeira imagem grande de avião buscando $k = 1024$ coeficientes. Porém a imagem é formada com apenas 283 coeficientes do espectro exato.

Na prática, foram encontrados 1718 candidatos, um número bem menor que deveria ter sido. Dos 1718 candidatos encontrados, apenas 283 deles fazem parte do sinal exato x correspondente. Apesar disso, o erro calculado $|x - \hat{f}'|_1 = 0,00794135$, ou seja um erro de $7,75\text{x}10^{-6}$ por coeficiente. A Figura 4.11 mostra o espectro exato x e o espectro achado $\hat{f}'$. Ao contrário do resultado anterior, podemos observar que os coeficientes encontrados estão muito distantes do espectro exato.

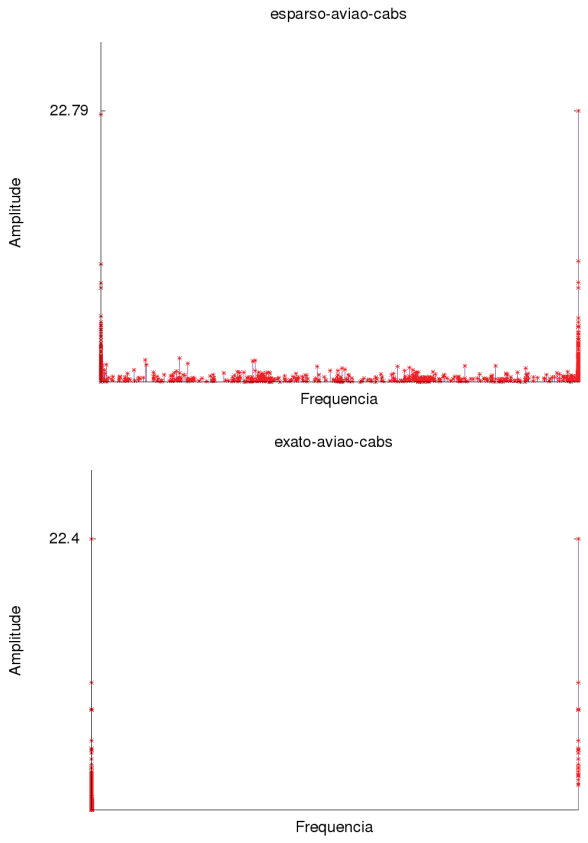


Figura 4.11: Em cima, o espectro $\hat{f}'$ recuperado pela *SFFT* e em baixo o expectro exato x.

5 Conclusão

A transformada rápida de Fourier é uma forma eficiente de se obter o espectro de um sinal, hoje, com avanços no desenvolvimento de algoritmos baseados na transformada esparsa de Fourier, é possível que obter o espectro de um sinal pode ser ainda mais eficiente. Este trabalho mostrou que um subconjunto dos coeficientes do espectro de uma imagem representa quase toda a informação daquela imagem. Portanto, isso é uma evidência de que imagens sejam sinais esparsos e que podemos utilizar os algoritmos de aproximação da *DFT* para obter o espectro de imagens. Sabendo que as imagens estão cada vez com tamanho maiores, *i.e.*, $N > 2^{22}$, obter o espectro dessas imagens utilizando algoritmos lineares já não são rápidos o suficiente, portanto algoritmos sublineares em relação ao tamanho do sinal podem ser uma alternativa para se ter um ganho no desempenho de sistemas que utilizam a *DFT* como sub-rotina.

O algoritmo utilizado neste trabalho está em sua primeira versão e além disso, o algoritmo considera que o sinal seja *exatamente* esparso para garantir a complexidade e uma resposta próxima do exato. Sabendo disso, os resultados apresentados na Seção 4.7 mostram que não temos uma resposta ideal, porém podemos utilizar um desenho de filtro específico para imagens e conseguir obter uma resposta ideal. Outra forma de otimizar o algoritmo para este tipo de aplicação é modificar a heurística da etapa de localização dos coeficientes. No algoritmo utilizado, temos a versão mais simples dessa heurística. Mesmo assim, se considerar que no caso das imagens maiores tivemos um erro baixo para cada coeficiente calculado mesmo com muitos coeficientes que não havia no espectro original, podemos assumir que um filtro mais adequado para esse tipo de aplicação e uma heurística mais robusta para localizar os coeficientes resultaria em um espectro obtido com menos esforço computacional que a *FFT* e um erro mais tolerável, viabilizando o uso da transformada esparsa de Fourier para obter o espectro de sinais de áudio de forma rápida.

Referências

[ACG05] M. Strauss A. C. Gilbert, S. Muthukrishnan. Improved time bounds for near-optimal sparse fourier representations, 2005.

[AGS03] Adi Akavia, Shafi Goldwasser, and Samuel Safra. Proving hard-core predicates using list decoding. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '03, pages 146–, Washington, DC, USA, 2003. IEEE Computer Society.

[Aka10] Adi Akavia. Deterministic sparse fourier approximation via fooling arithmetic progressions, 2010.

[Cha68] K. Chandrasekharan. *Introduction to Analytic Number Theory*. Springer-Verlag Berlin Heidelberg, 1968.

[Cip00] B. A. Cipra. The Best of the 20th Century: Editors Name Top 10 Algorithms. *SIAM News*, 33, 2000.

[CT65] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19:297–301, 1965. URL: `http://cr.yp.to/bib/entries.html#1965/cooley`.

[cWF03] Avery Li chun Wang and Th Floor Block F. An industrial-strength audio search algorithm. In *Proceedings of the 4 th International Conference on Music Information Retrieval*, 2003.

[DdSN10] Paulo S. R. Diniz, Eduardo A. B. da Silva, and Sergio L. Netto. *Digital Signal Processing: System Analysis and Design*. Cambridge University Press, 2 edition, 2010.

[DL42] G.C. Danielson and C. Lanczos. Some improvements in practical fourier analysis and their application to x-ray scattering from liquids. *Journal of the Franklin Institute*, 233(4):365 – 380, 1942.

[FJ05] Matteo Frigo and Steven G. Johnson. The design and implementation of fftw3. In *PROCEEDINGS OF THE IEEE*, pages 216–231, 2005.

[Fou22] J. B. J. Fourier. *Théorie analytique de la chaleur*. Didot, 1822.

[GGI+02] A. C. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss. Near-optimal sparse fourier representations via sampling. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 152–161, New York, NY, USA, 2002. ACM.

[Goo58] I. J. Good. The interaction algorithm and practical fourier series. *Journal of the Royal Statistical Society, Ser. B*, 20:361 – 375, 1958.

[HIKP12a] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Nearly optimal sparse fourier transform. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 563–578, New York, NY, USA, 2012. ACM.

[HIKP12b] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Simple and practical algorithm for sparse fourier transform. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 1183–1194, Philadelphia, PA, USA, 2012. Society for Industrial and Applied Mathematics.

[HJB84] M. Heideman, D. Johnson, and C. Burrus. Gauss and the history of the fast Fourier transform. *IEEE ASSP Magazine*, 1(4):14–21, October 1984.

[IGS07] M.A. Iwen, A. Gilbert, and M. Strauss. Empirical evaluation of a sub-linear time sparse dft algorithm. *Commun. Math. Sci.*, 5(4):981–998, 12 2007.

[Iwe10] M. A. Iwen. Combinatorial sublinear-time fourier algorithms. *Found. Comput. Math.*, 10(3):303–338, June 2010.

[KM91] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. In *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing*, STOC '91, pages 455–464, New York, NY, USA, 1991. ACM.

[Kri09] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[Lan93] Serge Lang. *Algebra (3. ed.)*. Addison-Wesley, 1993.

[Man92] Yishay Mansour. Randomized interpolation and approximation of sparse polynomials. In *Proceedings of the 19th International Colloquium on Automata, Languages and Programming*, ICALP '92, pages 261–272, London, UK, UK, 1992. Springer-Verlag.

[MJFS01] B. Moon, H. V. Jagadish, C. Faloutsos, and J. H. Saltz. Analysis of the clustering properties of the hilbert space-filling curve. *IEEE Transactions on Knowledge and Data Engineering*, 13(1):124–141, Jan 2001.

[PdC06] Marc-Antoine Parseval des Chênes. Mémoire sur les séries et sur l'intégration complète d'une équation aux différences partielles linéaire du second ordre, à coefficients constants. 1:638–648, 1806.

[Sha49] C. E. Shannon. Communication in the presence of noise. *Proc. Institute of Radio Engineers*, 37(1):10–21, 1949.

[SP14] Jarn Schumacher and Markus Püschel. High-performance sparse fast Fourier transforms. In *IEEE Workshop on Signal Processing Systems (SIPS)*, pages 1–6, 2014.

[Tol62] Georgi P. Tolstov. *Fourier Series*. 1962.

[Yat37] F. Yates. *The Design and Analysis of Factorial Experiments*. Imperial Bureau of Soil Science. Technical Communication. Imperial Bureau of Soil Science, 1937.