

**MIGRAÇÃO REVERSA NO
TEMPO POR FILTROS
MÍNIMOS QUADRADOS DE
ÚNICA ITERAÇÃO COM USO
DE REDES NEURAIAS
CONVOLUCIONAIS NO
DOMÍNIO DA
TRANSFORMADA CURVELET**

ÁTILA SARAIVA QUINTELA SOARES

SALVADOR – BAHIA
JANEIRO – 2022

**Migração Reversa no Tempo por filtros mínimos quadrados de
única iteração com uso de redes neurais convolucionais no
domínio da transformada Curvelet**

por

ÁTILA SARAIVA QUINTELA SOARES

Geofísico (UFBA – 2019)

Orientador: Prof. Dr. Reynam da Cruz Pestana

DISSERTAÇÃO DE MESTRADO

Submetida em satisfação parcial dos requisitos ao grau de

MESTRE EM CIÊNCIAS

EM

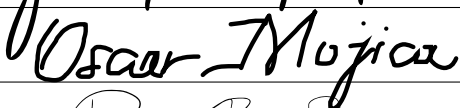
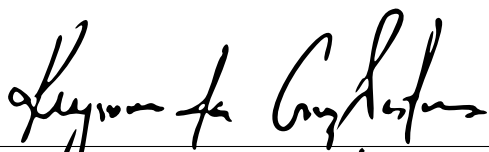
GEOFÍSICA

ao

Conselho Acadêmico de Ensino

da

Universidade Federal da Bahia



Comissão Examinadora

Dr. Reynam da Cruz Pestana

Dr. Oscar Fabian Mojica Ladino

Dr. Diego Fernando Barrera Pacheco

Aprovada em 18 de Janeiro de 2022

A presente pesquisa foi desenvolvida no Centro de Pesquisa em Geofísica e Geologia da UFBA, com recursos próprios, da CAPES, e da CNPq

Q999 Soares, Átila Saraiva Quintela,

Migração Reversa no Tempo por filtros mínimos quadrados de única iteração com uso de redes neurais convolucionais no domínio da transformada Curvelet / Átila Saraiva Quintela Soares. — Salvador, 2022.

48 f.: il., mapas, fotos.

Orientador: Prof. Dr. Reynam da Cruz Pestana

Dissertação (Mestrado) - de Pós-Graduação em Geofísica. Instituto de Geociências da Universidade Federal da Bahia, 2022.

1. Bahia - Geofísica. I. Título.

911.6(813.8)(043)

Dedico esse trabalho à três entidades.

Primeiramente a minha família, que teve tanta paciência com esse desafio que é o mestrado.

Ao meu orientador e professor Reynam Pestana, pelos ensinamentos de valor inestimável. E finalmente ao meu amigo Marcos Conceição, que tanto me ajudou nessa jornada.

Resumo

O conceito de filtros de focalização é uma nova abordagem para a migração de mínimos quadrados, tentando obter resultados semelhantes, mas com apenas uma iteração, sem recorrer a métodos iterativos computacionalmente caros. Nos últimos anos, vários estudos propuseram diferentes soluções para esse problema, como o uso de transformações baseadas em Fourier e redes neurais. No final, todos tentam alcançar algum tipo de reparametrização e focalização da imagem usando esses novos parâmetros, aproximando-se da inversa da Hessiana. Essa técnica não é nova, sendo empregada em câmeras digitais para filtrar o borrado inerente ao equipamento, usando um conceito similar. Neste trabalho estudamos o uso da topologia de rede neural U-Net, aplicada tanto no domínio de transformada de curvelet quanto para janelas no domínio espacial. A U-Net é um tipo específico de rede neural convolucional (CNN) que possui blocos de codificação e decodificação, o que aumenta sua capacidade de reconhecer características em diferentes escalas. Ao aplicá-lo ao domínio da transformada curvelet, que é separado por escala, ângulo e localização, tem a oportunidade de compreender melhor aspectos diferentes das mesmas feições espaciais. A ideia é treinar a rede como um filtro de correspondência entre o par de imagens remigradas e migradas e, em seguida, aplicá-lo à imagem migrada. Nossos estudos mostram que o treinamento do modelo de aprendizagem profunda (*deep learning*) U-Net no domínio curvelet pode melhorar as regiões mais profundas da seção sísmica migrada. Este é um dos poucos estudos recentes tentando usar redes neurais no domínio da transformada de curvelet, e muita pesquisa ainda é necessária para compreender plenamente quais são as possibilidades de usar tal técnica. O filtro baseado na rede U-Net foi testado em dois conjuntos de dados sintéticos (Marmousi e Sigsbee), apresentando resultados encorajadores, mostrando que a aplicação da rede após o processo de treinamento produziu uma imagem sísmica com melhor resolução do que o resultado da seção migrada convencionalmente.

Abstract

The concept of deblurring filters is a novel approach to the least-squares migration, trying to achieve similar results but with only one iteration, without resorting to computationally expensive iterative methods. There have been several studies proposing different solutions to this problem, such as using Fourier-based transforms and neural networks. In the end, all of them try to achieve some sort of reparametrization and deblurring of the image by using those new parameters, to approximate the inverse of the Hessian. This technique is not new, being employed in digital cameras to filter the blur inherent to the equipment by using a similar concept. This dissertation studies the use of the U-Net neural network topology, applied to the curvelet transform domain and to patches in the spatial domain. The U-Net is a specific type of convolutional neural network (CNN) that has encoding and decoding blocks, which enhances its ability to recognize features at different scales. By applying it to the curvelet domain, which is separated by scale, angle, and location, it has the opportunity to better grasp different aspects of the same features. The idea is to train the network to match the pair of remigrated and migrated images, and then apply it to the migrated image. Our research shows that training the U-Net deep learning model in the curvelet domain can improve resolution at the deep regions of the migrated seismic section. This is one of the very few recent studies attempting to use neural networks in the curvelet transform domain, and a lot of research is still needed to fully grasp what are the possibilities of using such a technique. The filter based in the U-Net network was tested on two synthetic data sets (Marmousi and Sigsbee), presenting encouraging results, on top of showing that the network application after the training process produced a seismic image with better resolution compared to the conventionally migrated section.

Índice

Resumo	4
Abstract	5
Índice	6
Índice de Tabelas	8
Índice de Figuras	9
Introdução	12
1 Teoria	15
1.1 Migração	15
1.2 O método de Migração Reversa no Tempo (RTM)	16
1.2.1 Caso pós-empilhamento	16
1.2.2 Caso pré-empilhamento	17
1.3 Modelagem Born	19
1.4 A transformada curvelet	20
1.4.1 Transformada curvelet Discreta Rápida	22
1.5 Redes Neurais Artificiais	25
1.5.1 Pesos	26
1.5.2 Função de ativação	27
1.5.3 O viés	28
1.5.4 Função custo	29
1.6 Redes Neurais Convolucionais	30
1.6.1 Arquitetura U-Net	31
2 Metodologia	34
2.1 Abordagem baseada na rede U-Net (Avila et al., 2021)	35
2.2 Domínio curvelet U-Net	38

3	Resultados e discussão	44
3.1	Marmousi 2D	44
3.2	Sigsbee2B	45
4	Conclusões	56
Agradecimentos		59
5	Referências Bibliográficas	61

Índice de Tabelas

2.1	Descrição da topologia U-Net utilizada, baseada no trabalho de Avila et al. (2021).	37
-----	---	----

Índice de Figuras

1.1	Esta Figura ilustra como o domínio da frequência é dividido em janelas trapezoidais. Perceba que as janelas tem seu comprimento e largura obedecendo o escalonamento parabólico. A parte sombreada representa uma janela de cisalhamento (Candès et al., 2006).	23
1.2	Ilustração do método <i>Wrapping</i> , que “embrulha” uma forma dentro de um paralelogramo ao assumir periodicidade (Candès et al., 2006).	24
1.3	Esquema representativo de um neurónio típico. (a) Dendrito (b) Soma (c) Núcleo (d) Axônio (e) Bainha de mielina (f) Célula de Schwann (g) Nódulo de Ranvier (h) Axônio terminal (Moreira, 2013).	25
1.4	Uma rede neural simples e o mapeamento da primeira camada oculta numa matrix 4x3 (Yasoubi, Hojabr e Modarressi, 2017).	26
1.5	Exemplo das operações matemáticas que ocorrem dentro de um neurônio. x_i são as entradas, ω_i são os pesos, n a quantidade de pesos e valores de entrada, b é o vies, e f é a função de ativação (Sharma, 2017).	27
1.6	Diferentes tipos de função de ativação e seus respectivos gráficos (Jadon, 2018).	28
1.7	Comparação de como funciona uma rede neural com dois neurônios com e sem bias.	29
1.8	Numa CNN no formato codificação-decodificação, os mapas de feições são espacialmente comprimidos por por uma rede de codificação, e então tem seu tamanho incrementado de volta para o tamanho da imagem de saída (Lucas et al., 2018).	32
2.1	A arquitetura U-Net utilizada como filtro de correspondência. Os mapas de característica convencionais resultantes das operações de convolução e mudança de escala são representados pela cor verde. Os de cor rosa representam aqueles que foram copiados a partir dos mapas de características de mesma escala, mas na fase de codificação. Estes são concatenados com aqueles presentes na fase de decodificação.	38

2.2	Representação de domínio curvelet de uma imagem de um círculo em 2 escalas com 8 fatias angulares na primeira escala fora do kernel. Os símbolos s e w representam os números de escala e fatias, respectivamente. Observe como as diferentes fatias captam diferentes ângulos do círculo. As amplitudes são o valor absoluto dos coeficientes complexos, e o resultado foi elevado ao quadrado e convolvido com um filtro gaussiano para garantir a legibilidade. .	39
2.3	Representação da imagem migrada a partir do modelo Sigsbee2B no domínio curvelet, com 2 escalas e 8 fatias angulares na primeira escala fora do kernel. Observe como diferentes fatias captam refletores de inclinações variadas. As amplitudes são o valor absoluto dos coeficientes complexos, e o resultado foi convolvido com um filtro gaussiano e transposto para garantir a legibilidade.	40
3.1	Campo de velocidade do modelo Marmousi, utilizado para a modelagem e migração.	46
3.2	(a) Representa a imagem migrada RTM a partir do dado observado no modelo Marmousi. (b) Resultado da aplicação da rede neural U-Net já treinada à imagem migrada. Ambas figuras são plotadas na mesma escala de amplitude para melhor comparação.	47
3.3	(a) Janela quadrada extraída a partir da imagem original, em profundidade, migrada a partir do dado observado no modelo Marmousi. (b) Janela quadrada extraída a partir da imagem filtrada com a rede U-Net na mesma posição. (c) Janela quadrada extraída a partir da imagem original, em meio as falhas no centro do modelo, migrada a partir do dado observado. (d) Janela quadrada extraída a partir da imagem filtrada com a rede U-Net na mesma posição.	48
3.4	(a) Representa a imagem migrada RTM a partir do dado observado no modelo Marmousi. (b) Resultado da aplicação da rede neural U-Net no domínio curvelet já treinada à imagem migrada. Ambas figuras são plotadas na mesma escala de amplitude para melhor comparação.	49
3.5	(a) Janela quadrada extraída a partir da imagem migrada RTM, em profundidade, a partir do dado observado no modelo Marmousi. (b) Janela quadrada extraída a partir da imagem filtrada com a rede U-Net no domínio curvelet na mesma posição.	50
3.6	Campo de velocidade do modelo Sigsbee 2B, utilizado para a modelagem e migração.	51

-
- 3.7 (a) Representa a imagem migrada RTM a partir do dado observado no modelo Sigsbee 2B. (b) Resultado da aplicação da rede neural U-Net já treinada à imagem migrada. Ambas figuras são plotadas na mesma escala de amplitude para melhor comparação. 52
- 3.8 (a) Janela quadrada extraída a partir da imagem original, logo abaixo do domo de sal, migrada a partir do dado observado no modelo Sigsbee 2B. (b) Janela quadrada extraída a partir da imagem filtrada com a rede U-Net na mesma posição. 53
- 3.9 (a) Representa a imagem migrada RTM a partir do dado observado no modelo Sigsbee 2B. (b) Resultado da aplicação da rede neural U-Net no domínio curvelet já treinada à imagem migrada. Ambas figuras são plotadas na mesma escala de amplitude para melhor comparação. 54
- 3.10 (a) Janela quadrada extraída a partir da imagem migrada RTM a partir do dado observado no modelo Sigsbee, logo abaixo do domo de sal. (b) Janela quadrada extraída a partir da imagem filtrada com a rede U-Net no domínio curvelet na mesma posição. 55

Introdução

Com alvos de exploração sísmica cada vez mais profundos, o imageamento desses refletores em grandes profundidades tem se tornado um grande desafio para o imageamento sísmico. A solução óbvia é ser criativo na fase de aquisição para ter afastamentos longo, receptores 3C, geometrias 3D e assim por diante; no entanto, isso apenas aumenta o custo. Por esse motivo, é muito mais barato aprimorar as técnicas de imageamento que já estão sendo empregadas. Assim, ser capaz de resolver refletores mais profundos do que normalmente seria possível, ao aprimorar os resultados dos métodos de migração já consolidados, como RTM¹, sem um custo computacional adicional elevado, tornando-se, então, muito mais atraentes.

A migração reversa no tempo mínimos quadrados (LSRTM²) é uma técnica de inversão comumente empregada, uma vez que resolve alguns dos problemas dos métodos tradicionais de migração sem ter que recorrer a métodos mais sofisticados, embora caros, como FWI³ multiparâmetro, para inversão do modelo da subsuperfície. O resultado disso, quando comparado à imagem migrada convencionalmente, é uma imagem com iluminação mais balanceada em relação à regiões mais profundas, refletores mais concisos e menos desfocados. No entanto, a maneira tradicional de fazer a migração mínimos quadrados (LSM⁴) tem sido usar métodos iterativos como *steepest descent* e conjugado do gradiente, levando várias iterações para atingir o objetivo final. Isso aumenta seu custo consideravelmente quando comparado ao custo da migração pré-empilhamento convencional. Como consequência, surgiram duas linhas principais de pesquisa para aliviar esse custo. A primeira consiste em encontrar pré-condicionadores para garantir uma convergência rápida (e.g. Dutta et al., 2015; Dutta et al., 2016; Dutta et al., 2017; M. Liu et al., 2017; Sun, Fomel e Zhu, 2015). A segunda, na qual esse trabalho se enquadra, é encontrar filtros de focalização que colhem alguns dos benefícios do LSM, embora sejam aplicados aos dados em uma única iteração, depois de serem previamente calculados.

Várias pesquisas têm proposto diferentes formas de se chegar a esse filtro. A ideia geral

-
1. Sigla em inglês para reverse time migration.
 2. Sigla em inglês para least-squares reverse time migration.
 3. Sigla em inglês para full waveform inversion.
 4. Sigla em inglês para least-squares migration.

proposta por Guitton (2004), e também sugerida por Claerbout (1992), é aproximar a inversa da Hessiana (\mathbf{H}^{-1}) calculando um filtro de correspondência entre a imagem remigrada⁵ e a imagem migrada, para aplicá-la na imagem migrada posteriormente. Hu, Schuster e Valasek (2001) propuseram basicamente a mesma metodologia, mas assumiram um modelo de velocidade horizontalmente invariante. Aoki e Schuster (2009) sugeriram o uso de modelos de referência para calcular o filtro. Wang et al. (2016) calcularam o filtro no domínio curvelet. Há um trabalho recente em particular de Sanavi, Moghaddam e Herrmann (2021) que usa o método de otimização BFGS de memória limitada⁶ (Nocedal e Wright, 2006) como o método de otimização de escolha para calcular um filtro semelhante, no entanto, a função de custo é calculada no domínio espacial aplicando a transformada de curvelet inversa antes de se fazer o cálculo da norma. Liu e Peter (2018) usaram um filtro de deconvolução de Wiener simples para aplicar aos dados observados, empregando a transformada de Fourier para calcular e aplicá-lo traço a traço. No ano seguinte, Q. Liu et al. (2019) usaram a deconvolução Gabor para obter um filtro semelhante, mas agora não estacionário. Mais recentemente, Avila et al. (2021) projetaram uma rede neural U-Net para atuar como um filtro de correspondência. U-Net é um tipo de rede neural convolucional (CNN⁷) com codificação e decodificação com conexões de salto integradas. É notável que a maioria dos trabalhos empregados, mais recentemente, utilizaram uma transformação da família Fourier ou uma rede neural.

Todos os métodos mencionados são muito semelhantes aos aplicados em outros problemas de imagem, como restauração de imagem, restauração de pintura, atenuação de ruído, e segmentação de imagem. A ideia geral é sempre modelar um sinal \mathbf{y} como a saída de um sistema \mathbf{T} , em que a entrada é denotada por \mathbf{x} (Lucas et al., 2018). A diferença é que o cálculo da inversa da Hessiana (\mathbf{H}^{-1}) foi substituído por um filtro que se infere da saída \mathbf{y} (imagem remigrada) e da entrada \mathbf{x} (imagem migrada). Por outro lado, a abordagem U-Net tem muitos parâmetros dentro de suas camadas ocultas, e sua natureza de codificação e decodificação deve significar que tem um nível semelhante de resolução local. No entanto, ainda há muito que não sabemos sobre redes neurais, por isso é difícil projetar um modelo matemático formal para expressar o que a CNN treinada faz. Portanto, muitos trabalhos têm o objetivo de entender melhor o que elas fazem, como Pappas, Romano e Elad (2016).

Neste trabalho, procuramos expandir a abordagem usada por Avila et al. (2021), implementando um filtro de correspondência no domínio curvelet. A ideia é usar uma rede neural U-Net semelhante como filtro, mas aplicada nos coeficientes do domínio curvelet da imagem

5. A imagem remigrada é o resultado de migrar um conjunto de tiros gerados a partir da demigração da imagem migrada original.

6. Também chamado de L-BFGS, algoritmo de otimização que aproxima o algoritmo Broyden–Fletcher–Goldfarb–Shanno (BFGS) utilizando uma quantidade limitada de memória RAM.

7. Sigla em inglês para convolutional neural networks.

remigrada e migrada. O resultado seria uma mescla dos métodos de Wang et al. (2016) e Avila et al. (2021); aproximando-se do limite do que o LSRTM baseado em filtro de focalização pode fornecer. Isso ocorre porque a transformada curvelet expressa os dados com base na escala, direção e localização de seus recursos, sendo uma técnica de transformação de dados atual de muita utilidade para identificar padrões de diferentes escalas e ângulo. Essa multiparametrização permitiria que a rede neural correspondesse aos mesmos aspectos das mesmas feições visuais. No entanto, parece que há um limite para o quanto você pode alcançar fazendo uso de transformações matemáticas como pré-processadores. Neste trabalho, testamos esse limite comparando este novo filtro de rede neural de domínio curvelet com o proposto por Avila et al. (2021).

A ideia de treinar uma rede neural em um domínio diferente não é nova, embora não seja tão comum. No trabalho de Pratt et al. (2017) que foi proposto treinar CNNs com dados de entrada no domínio de Fourier para diminuir a carga computacional das operações de convolução realizando-as no domínio da frequência. No entanto, para que o filtro de correspondência funcione, ele deve parametrizar a imagem de forma eficaz. Se o filtro usar apenas uma transformada de Fourier 2D simples, as variações locais na frequência não serão levadas em consideração, o que resultará na correspondência de coeficientes da mesma frequência, embora possam ser de partes diferentes da imagem. Não apenas isso, mas também os coeficientes não são separados por ângulo. A transformada curvelet resolve todos esses problemas, embora ainda tenha um modelo matemático explícito. No entanto, existem poucos trabalhos sobre o uso de redes neurais no domínio curvelet, enquanto nenhum está relacionado à pesquisa de processamento sísmico. Saxena, Sharma e Chaurasiya (2015) usam a transformada curvelet para extrair recursos de dados de impressão digital e os usa como entrada para um classificador de rede *feed-forward* simples. Gupta e Tiwari (2014) usam uma abordagem semelhante, embora para detectar imagens de tomografia computadorizada de câncer de pulmão. O objetivo deste trabalho é usar uma ideia parecida, mas para um problema de regressão, o LSRTM de estimativa de filtro de focalização.

Começamos descrevendo primeiro a teoria básica necessária para compreensão mínima da técnicas empregadas no capítulo 1. De posse deste conhecimento, a metodologia empregada será explicada no capítulo 2. Ao aplicar a metodologia aos modelos Marmousi e Sigsbee 2B, os resultados obtidos são exibidos e discutidos no capítulo 3. No final, a conclusão se encontra no capítulo 4, seguida das referências bibliográficas.

1

Teoria

1.1 Migração

A migração sísmica é um processo por qual um campo de ondas sísmicas, contendo ondas espalhadas e difratadas medidas como função do tempo, é então convertida em uma imagem focada da correspondente distribuição espacial dos espalhadores e difratores (George A. McMechan, 1989). Os espalhamentos e difrações são visíveis no campo de ondas sísmicas devido ao afastamento fonte-receptor. Devido a essa distância, o ponto do refletor sendo imageado está na posição mais profunda da primeira zona de Fresnel associada a um par fonte-receptor, porém na seção empilhada esse ponto é exibido na posição do receptor. Essa diferença entre posição real e do receptor deve ser corrigida para que os refletores sejam exibidos na posição correta. O processamento CMP¹ convencional tenta aproximar a posição do ponto refletor para a metade da distância entre fonte e receptor. A migração, por sua vez, utiliza princípios físicos como a equação da onda para corrigir a posição dos refletores.

Existem diversas abordagens para “migrar” uma seção em tempo, estas porém podem ser separadas por qual etapa do fluxo de processamento que atuam ou pela forma que resolvem a equação da onda e qual a equação específica que está sendo resolvida. Sob a primeira classificação, um método de migração pode ser aplicado aos sismogramas antes de serem empilhados (chamados de *pré-empilhamento*), ou no dado já empilhado (chamados de *pós-empilhamento*). Sob a segunda, um método pode ser classificado pela equação a ser resolvida: equação da onda no domínio da frequência (equação de Helmholtz), equação escalar acústica da onda, ou até mesmo equação elástica da onda, entre outros.

1. Common Middle Point.

1.2 O método de Migração Reversa no Tempo (RTM)

Um método de implementação do processo de migração é pela extrapolação reversa do campo de onda medido, utilizando o dado como condição de contorno na solução por diferenças finitas da equação escalar da onda, junto à aplicação de uma condição de imagem para extrair as amplitudes referentes aos refletores em cada passo temporal (George A. McMechan, 1989).

Tal método foi desenvolvido em meados do ano 1983; primeiramente por Whitmore (1983) e depois por G. A. McMechan (1982, 1983), bem como por Baysal, Kosloff e Sherwood (1983). Uma abordagem parecida também chamada pelo mesmo nome foi desenvolvida por Loewenthal e Mufti (1983). No entanto, a concepção de uma condição de imagem adequada para aplicação do RTM em dados pré-empilhamento veio a tona somente em meados de 1986 com os trabalhos de Chang e McMechan (1986) e Sun e McMechan (1986); no qual no primeiro propuseram uma condição de imagem em que se comparava a modelagem direta do dado ao dado reverso no tempo em cada passo temporal do algoritmo de diferenças finitas de segunda ordem, retirando-se as amplitudes nas posições imageadas naquele passo temporal, abordagem essa muito parecida com a que será detalhada nesse projeto; já no segundo propuseram uma condição de imagem parecida, porém resolvendo a equação elástica da onda. Desde então, as melhoras ao método de migração RTM orbitaram em torno de melhorar a condição de imagem, melhorar a iluminação da imagem de forma geral, ou de uma generalização ou otimização tanto da equação da onda, bem como na forma de resolvê-la.

1.2.1 Caso pós-empilhamento

Segundo Levin (1984), a ideia chave da migração reversa no tempo pós-empilhamento é que o dado sísmico pode ser alternativamente pensado como uma função de fonte. Assim, o dado age como uma condição de contorno em cada passo temporal, transformando cada receptor em uma fonte, emitindo a energia gravada de volta para a subsuperfície. A equação final que vai ser utilizada no algoritmo é obtida a partir da equação da onda:

$$\frac{\partial^2 P}{\partial t^2} - c^2 \nabla^2 P = c^2 f(t) \delta(z - z_s) \delta(x - x_s), \quad (1.1)$$

na qual $P(z, x, t)$ é o campo de pressão num tempo específico, $c(z, x)$ o campo de velocidade, e $f(t) \delta(x - x_s) \delta(z - z_s)$ é a fonte² injetada no tempo t , e na posição (z_s, x_s) . Esta é então discretizada, utilizando diferenças finitas (sendo o exemplo uma aproximação de segunda

2. O símbolo $\delta(z - z_s)$ na equação representa uma distribuição chamada delta de Dirac que é zero em todos os pontos do domínio exceto no ponto z_s , indicando que a fonte $f(t)$ é injetada neste ponto.

ondem) para resolver a derivada de segunda ordem no tempo, da seguinte forma:

$$\frac{P^{n+1} + P^{n-1} - 2P^n}{\Delta t^2} - c^2 \nabla^2 P^n = c^2 f^n \delta(z - z_s) \delta(x - x_s), \quad (1.2)$$

para então se isolar o P^{n-1} , de forma a se obter o campo de pressão passado a partir do presente e do futuro, e então regredir no tempo:

$$P^{n-1} = \Delta t^2 c^2 \nabla^2 P^n - P^{n+1} + 2P^n + \Delta t^2 c^2 f^n \delta(z - z_s) \delta(x - x_s). \quad (1.3)$$

No entanto, na RTM, o dado presta o papel de fonte na equação da onda, logo o termo $f(t) \delta(z - z_s) \delta(x - x_s)$ tem de ser substituído pelo dado da seguinte forma:

$$P^{n-1} = \Delta t^2 c^2 \nabla^2 P^n - P^{n+1} + 2P^n + \Delta t^2 c^2 d_{\text{obs}}^n(x) \delta(z - z_s), \quad (1.4)$$

onde $d_{\text{obs}}^n(x)$ consiste num corte do dado sísmico *zero-offset* no tempo de respectiva iteração n . O laplaciano pode ser calculado também por diferenças finitas, ou até mesmo por transformada de Fourier³. Por fim, a equação (1.4) é então resolvida com o índice n variando de Nt (número de amostras no tempo do dado) até $n = 1$, para então ser obtido o P^0 que representa a imagem no tempo zero.

A justificativa teórica para o método, assim como para todos os métodos de migração convencionais para dados empilhados, é baseada implicitamente no modelo do refletor explosivo (Loewenthal et al., 1976). Tal modelo assume que toda a energia presente num determinado tempo t no dado empilhado é consequência de uma reflexão no tempo $\frac{1}{2}t$, assim a amplitude do coeficiente de reflexão pode ser determinada por propagar a energia de volta à subsuperfície até metade do seu tempo de chegada, ou de forma equivalente, propagando a energia de volta para o tempo zero, com metade da velocidade (Levin, 1984). Sendo assim, a condição de imagem do método RTM pós-empilhamento se baseia neste modelo, ou seja, quando o dado é retropropagado até o tempo zero.

1.2.2 Caso pré-empilhamento

No caso da migração RTM pré-empilhamento, o dado é retropropagado e a cada passo temporal o campo de pressão é correlacionado com o passo temporal respectivo da modelagem direta.

A simplicidade elegante da condição de imagem de tempo zero trás certas limitações implícitas, sendo a mais séria delas o fato de que é somente válida para dados de afastamento nulo (*zero-offset*). Todavia, é possível generalizar tal condição de imagem para que

3. Este método de resolução de derivadas vem da teoria da transformada de Fourier. Apesar de não ser tratado neste trabalho, é um assunto amplamente abordado em livros de cálculo avançado.

possa ser aplicada para qualquer geometria fonte-receptor, ainda utilizando o modelo refletor explosivo. A ideia chave por trás dessa condição de imagem é que registro de fonte e de receptores podem ser extrapolados e que o refletor existe onde o ambos campos de onda estão em fase. Por um lado, no caso de afastamento nulo (*zero-offset*) ambos fontes e receptores são extrapolados porque ambos ocupam a mesma posição, logo as ondas propagadas percorrem a mesma trajetória. Por outro lado, no caso afastamento não nulo (*finite-offset*) a extrapolação da fonte e do receptor são executadas separadamente, e cada passo temporal é correlacionado. Esta condição de imagem é comumente chamada de *excitation-time imaging condition* (Chang e McMechan, 1986).

O algoritmo convencional então consiste de três etapas principais. Começando com a etapa de resolução do problema direto, utilizando algum método de extrapolação no tempo como diferenças finitas exemplificado na equação (1.3), injetando-se uma wavelet (que em geral representa a resposta impulsiva da fonte) no meio e então guardar todos os campos de pressão de cada passo temporal. A segunda etapa consiste na extrapolação do dado na posição dos receptores como solução da equação da onda por diferenças finitas. A terceira é executada em cada passo temporal do algoritmo de diferenças finitas da segunda etapa, e feita ao se correlacionar o campo de pressão retropropagado com o campo modelado naquele mesmo passo temporal respectivo. O algoritmo é então repetido para cara tiro, e as imagens resultantes para cada um são então somadas.

A migração RTM pré-empilhamento trás consigo desvantagem de um aumento considerável de custo computacional, porém aliado a uma série de vantagens. O aumento do custo computacional resulta do fato de que para cada tiro são necessárias duas modelagens, uma direta e uma reversa, para gerar a imagem. No entanto, a principal vantagem de se utilizar essa condição de imagem é a melhora da razão sinal-ruído conforme se soma ponto a ponto as imagens resultantes da migração RTM de diferentes tiros. Isso também trás uma iluminação do modelo mais abrangente, visto que o campos de ondas de cada tiro tem distribuições de energia diferentes dependendo das suas posições, e a soma da imagem resultante de cada um promove uma interferência construtiva nas posições de maior energia, ou seja, nos refletores. Outra vantagem implícita advém do fato de que diretamente do dado no domínio do tiro se obtém a imagem “empilhada e migrada” equivalente. Porém, o resultado é ainda melhor, pois os métodos de empilhamento CMP equivalentes como correção NMO⁴ e DMO⁵, apesar de serem efetivos, perdem tal eficácia quando os refletores estão excessivamente

4. Correção Normal Move-Out que tem como objetivo horizontalizar a hipérbole de reflexão visível no dado no domínio do CMP, para que seja possível somar as contribuições de diferentes afastamentos que representam a mesma reflexão.

5. Correção Dip Move-Out que tem como objetivo corrigir a o dado resultante da correção NMO, para que leve em conta a inclinação dos refletores.

inclinados, como é o caso de domos de sal, sistemas de falhas verticais, estruturas geológicas rotacionadas, entre outros.

Os conceitos básicos sobre migração RTM apresentados na presente seção servem somente para permitir o entendimento da dissertação. Existem várias considerações computacionais adicionais a serem feitas que fogem ao escopo do presente trabalho, como necessidade de se atenuar bordas para simular meios infinitos, otimizações para que se diminua a quantidade de armazenamento necessário para guardar os passos temporais do algoritmo de diferenças finitas, como evitar dispersão numérica, entre outros.

1.3 Modelagem Born

Na teoria de inversão do espalhamento de ondas (Tarantola, 1984), a solução da equação da onda (equação 1.1) se assume ser na forma de uma série de espalhamento desse modo:

$$P = \sum_{k=0}^{\infty} P_k \epsilon^k = P_0 + \epsilon P_1 + \epsilon^2 P_2 + O(\epsilon^3) \quad (\epsilon \rightarrow 0), \quad (1.5)$$

onde P_0 é o campo de onda de fundo, P_1 é o campo espalhado de primeira ordem e P_k são campos espalhados de ordem superiores para $k > 1$, e ϵ é um parâmetro adimensional (Moser, 2012). Esta teoria então assume que o modelo de velocidade pode ser decomposto em um modelo de fundo com variações brandas $c_0(\mathbf{r})$, sendo $\mathbf{r} = (x, z)$, e um modelo de variações abruptas que atua como espalhador das ondas incidentes $c_1(\mathbf{r})$. Em termos matemáticos, isso se entende como:

$$c(\mathbf{r}) = c_0(\mathbf{r}) + \epsilon c_1(\mathbf{r}). \quad (1.6)$$

Um exemplo de espalhadores são interseções entre camadas geológicas, domos de sal, falhas, e etc. Estes formam superfícies refletoras que espalham a onda. Ao se substituir $c = c_0 + \delta c$ e $P = P_0 + \delta P$ na equação 1.1, assumindo $\epsilon = 1$, temos:

$$\left(\frac{1}{[(c_0 + \delta c(\mathbf{r}))^2] \partial t^2} - \nabla^2 \right) (P_0 + \delta P) = f(t) \delta(\mathbf{r} - \mathbf{r}_s), \quad (1.7)$$

sendo $\delta c(\mathbf{r}) = c_1(\mathbf{r})$, e $\delta P = P_1$, sendo que P_0 satisfaz a equação 1.1 para $c = c_0$. O símbolo δ nesta ocasião representa a diferença entre o campo total, seja o de pressão ou de velocidade, e o campo de fundo. Observe que os campo de ondas P_k ($k > 1$) representam espalhamentos de diferentes ordens que tem como contribuição o campo de fundo $c_1(\mathbf{r})$, porém a modelagem Born tem como alvo somente P_1 . Usando da seguinte equação:

$$\frac{1}{[(c_0 + \delta c(\mathbf{r}))^2]} = \frac{1}{c_0^2} - \frac{2\delta c}{c_0^3} + O(\delta c^2), \quad (1.8)$$

na equação 1.7 temos:

$$\left(\frac{1}{c_0^2} \frac{\partial^2}{\partial t^2} - \nabla^2 \right) \delta P = \Delta s, \quad (1.9)$$

onde

$$\Delta s(\mathbf{r}, t; \mathbf{r}_s) = \frac{2\delta c}{c_0^3} \frac{\partial^2 (P_0 + \delta P)}{\partial t^2}(\mathbf{r}, t; \mathbf{r}_s) + O(\delta \mathbf{c}, \delta \mathbf{P})^2. \quad (1.10)$$

O campo espalhado δP pode ser interpretado como um campo secundário que se espalha num meio não perturbado (equação 1.9), submetido a fontes secundárias expressas na equação 1.10 (Tarantola, 1984). Como δc é considerado pequeno, bem como se assume $P_0 \gg \delta P$, o termo $O(\delta \mathbf{c}, \delta \mathbf{P})^2$ pode ser desprezado e $P_0 + \delta P \approx P_0$ pode ser usado, sendo que essa aproximação se chama aproximação de Born. Isso significa que efeitos de múltiplas reflexões são desprezados e isso também é o motivo de somente o campo espalhado de primeira ordem ser utilizado na equação 1.7.

Neste trabalho, a modelagem Born foi utilizada para demigração. Demigração tem como objetivo obter um conjunto de tiros associados a um modelo de refletividade, que neste caso seria a imagem migrada. No entanto a mesma técnica pode ser utilizada com uma refletividade de um modelo para gerar o dado espalhado associado a ele. Para empregá-la, as equações 1.7 e 1.1 são resolvidas num sistema de equações onde o termo $\frac{2\delta c}{c_0^3} \frac{\partial^2 P_0}{\partial t^2}$ é aproximado pelo produto da refletividade m_{ref} e um termo em função de P_0 , ou seja $\frac{2\delta c}{c_0^3} \frac{\partial^2 P_0}{\partial t^2} \approx m_{ref} f(P_0)$. A aproximação para a refletividade utilizada foi $m_{ref} \approx \frac{2\delta c}{c_0}$, o que implica em $f(P_0) = \frac{1}{c_0^2} \frac{\partial^2 P_0}{\partial t^2}$. Porém, neste trabalho utilizei de mais uma aproximação, que utiliza da relação da equação da acústica da onda, de modo que $f(P_0) = \frac{1}{c_0^2} \frac{\partial^2 P_0}{\partial t^2} \approx \nabla^2 P_0$. Outros trabalhos, em sua maioria, vão além e assumem $f(P_0) = P_0$ diretamente, ou utilizam $m_{ref} \approx \frac{2\delta c}{c_0}$. De um modo geral, o importante é que se tenha uma correlação do campo direto e a refletividade.

É importante salientar que muito esforço foi depositado em implementar tanto o algoritmo para modelagem Born, quanto o para migração RTM, ambos para placas de vídeo Nvidia utilizando Cuda C++. O código para ambos pode ser encontrado no repositório do github: <https://github.com/AtilaSaraiva/LSRTM-GPU-Experiments>.

1.4 A transformada curvelet

A transformada curvelet é uma transformação direcional multiescalar que é capaz de fazer representações esparsas eficientes de objetos com arestas. É usado em vários campos, como processamento de imagem, eliminação de ruído, simulação de equações diferenciais parciais, sensoriamento comprimido e filtragem (Ma e Plonka, 2010). Esta seção tem como objetivo explicar brevemente a transformada curvelet de segunda geração e suas variantes discretas.

Quando comparada com seus predecessores, acredita-se que a transformada curvelet vem de uma longa linha de melhorias na transformada de Fourier original para certas aplicações. Esta última falha em resolver distúrbios de frequência em pequena escala levou à transformada de Fourier de curto tempo (*Short-time Fourier transform*). Essa técnica usa uma janela retangular de forma fixa no domínio da frequência, que é então aplicada à função em todo o seu domínio. Isso solucionou o problema de falta de resolução em pequena escala, mas essa abordagem carecia de resolução multiescala, o que levou à transformada wavelet convencional que usava uma janela de forma mutável. No entanto, para que uma transformação atinja uma representação esparsa ótima das curvas da imagem, a frequência por si só não é suficiente, pois a direção da curva também é importante. Isso levou à criação de várias wavelets direcionais, como *contourlets*, *shearlets*, *beamlets* e *ridgelets*. Dentre elas uma das mais populares é a transformada curvelet. A transformada curvelet de primeira geração, proposta por Candes e Donoho (2000), é baseada na transformada ridgelet, que foi aplicada a diferentes escalas da transformada wavelet 2D. Uma transformada curvelet de segunda geração muito mais simples foi proposta pelos mesmos autores (ver Candès e Donoho, 2005a, 2005b), que será explicada no restante da seção.

A transformada curvelet contínua (CCT) é calculada pelo produto interno de curvelets e a função alvo $f \in L^2(\mathbf{R}^2)$. Isso pode ser representado por

$$c(j, l, k) := \langle f, \varphi_{j,l,k} \rangle = \int_{\mathbf{R}^2} f(x) \overline{\varphi_{j,l,k}(x)} dx, \quad (1.11)$$

em que $\varphi_{j,l,k}(x)$ é uma curvatura de escala j , fatia angular l e localização $k \in \mathbf{R}^2$. De acordo com Candès et al. (2006), o CCT satisfaz a propriedade que chamaremos de “moldura apertada” (*tight frame*). Isso significa que a função alvo pode ser representada por uma série de curvelets como

$$f = \sum_{j,l,k} \langle f, \varphi_{j,l,k} \rangle \varphi_{j,l,k}, \quad (1.12)$$

além de ter uma relação de Paserval

$$\sum_{j,l,k} |\langle f, \varphi_{j,l,k} \rangle|^2 = \|f\|_{L^2(\mathbf{R}^2)}^2, \quad \forall f \in L^2(\mathbf{R}^2), \quad (1.13)$$

similar à transformada de Fourier.

A família de curvelets $\varphi_{j,l,k}(x)$ é calculada por rotação e translação da curvelet mãe, ou elemento básico, φ_j . Esta operação pode ser expressa como

$$\varphi_{j,l,k}(x) = \varphi_j \left[R_{\theta_l}(x - x_k^{(j,l)}) \right], \quad (1.14)$$

dado que $x_k^{(j,l)} = R_{\theta_l}^{-1}(k_1 \cdot 2^{-j}, k_2 \cdot 2^{-j/2})$, onde R_θ e R_θ^{-1} são respectivamente o operador de

rotação pelo ângulo θ e seu inverso / transposto,

$$R_\theta := \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix}. \quad (1.15)$$

A curva mãe é calculada usando uma técnica de partição de frequência. Isso significa seccionar uma fatia no domínio da frequência definida por duas funções de janela parabólica de comprimento $\approx 2^j$ e largura $\approx 2^{j/2}$. A primeira é uma janela radial $W(r)$ com $r \in [1/2, 2]$ que definirá uma região do anel que contém a fatia. A segunda é uma janela angular $V(t)$ com $t \in [-1, 1]$ que define o intervalo angular equidistante para a cunha. De posse desses dois, o elemento básico pode ser encontrado por:

$$\hat{\varphi}_j(r, \theta) = 2^{-\frac{3j}{4}} W(2^{-j}r) V\left(2^{\lfloor j/2 \rfloor} \frac{\theta}{2\pi}\right), \quad (1.16)$$

onde o delimitador $\lfloor \cdot \rfloor$ representa a operação de aproximação para baixo de um valor real para um valor inteiro.

Existem muitas propriedades, condições e outras considerações a serem feitas para explicar completamente o CCT, mas isso foge o escopo desta dissertação. O que foi exposto acima serve ao propósito de mostrar uma visão ampla sobre o assunto. Uma explicação mais aprofundada pode ser encontrada em Candès e Donoho (2005a, 2005b), Candès et al. (2006) e Ma e Plonka (2010).

1.4.1 Transformada curvelet Discreta Rápida

Para implementar a transformada curvelet em um domínio discreto, algumas simplificações devem ser feitas. O domínio discretizado é comumente representado por uma matriz cartesiana, tornando a representação de fatias angulares por retângulos impraticável. Portanto, estes são substituídos por trapézios como ilustrado na Figura 1.1. Mesmo assim, uma equação semelhante à equação 1.16 é válida, mas as janelas radiais e angulares devem agora ser modificadas para a nova forma, e o operador de rotação terá que ser substituído por um de cisalhamento.

Esta janela de cisalhamento pode ser descrita por:

$$\tilde{U}(\omega_x, \omega_y) := \tilde{W}_j(\omega_x, \omega_y) V_j(\omega_x, \omega_y), \quad (1.17)$$

Em que a janela radial agora é definida em quadrados concêntricos definidos como:

$$\tilde{W}_j := \sqrt{\Phi_{j+1}^2(\omega) - \Phi_j^2(\omega)}, \quad j \geq 0, \quad (1.18)$$

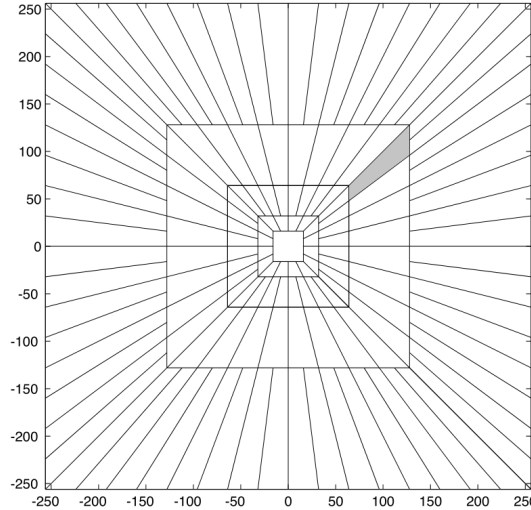


Figura 1.1: Esta Figura ilustra como o domínio da frequência é dividido em janelas trapezoidais. Perceba que as janelas tem seu comprimento e largura obedecendo o escalonamento parabólico. A parte sombreada representa uma janela de cisalhamento (Candès et al., 2006).

em que Φ consiste no produto de janelas unidimensionais passa-baixo dadas por $\Phi = \phi(2^{-j}\omega_x)\phi(2^{-j}\omega_y)$. A função ϕ está confinada entre 0 e 1 ($0 \leq \phi \leq 1$). A janela angular V é definida para cada $j \geq 0$ é definida como

$$V_j(\omega_x, \omega_y) := V \left(\begin{array}{c} 2^{\lfloor j/2 \rfloor} \frac{\omega_x}{\omega_y} \end{array} \right). \quad (1.19)$$

Em suma, a janela de cisalhamento na equação 1.17 isola frequências perto da cunha $\{(\omega_x, \omega_y) : 2^j \leq \omega_x \leq 2^{j+1}, -2^{-j/2} \leq \frac{\omega_y/\omega_x}{\leq} 2^{-j/2}\}$. Esta é a contraparte cartesiana da janela polar usada na transformação contínua. Como tal, o operador definido na equação 1.15 deve ser substituído por um equivalente de cisalhamento:

$$S_\theta := \begin{pmatrix} 1 & 0 \\ -\tan \theta & 1 \end{pmatrix}, \quad (1.20)$$

e os ângulos equidistantes por declives equidistantes $\tan \theta_l := l \cdot 2^{-\lfloor j/2 \rfloor}$, $l = -2^{-\lfloor j/2 \rfloor}, \dots, 2^{-\lfloor j/2 \rfloor} - 1$ (Candès et al., 2006). Como resultado, a família de janela de cisalhamento torna-se:

$$\tilde{U}_{j,l}(\omega) := \tilde{W}_j(\omega) V_j(S_{\theta_l} \omega), \quad (1.21)$$

que, comparado com o CCT é equivalente à família de curvelets na equação 1.14. Na mesma linha de pensamento, a equação 1.17 é equivalente ao elemento básico definido na equação 1.16.

Existem duas implementações principais da transformada rápida curvelet discreta (FDCT do inglês *Fast discrete curvelet transform*): USFFT e Wrapping. No USFFT, a inclinação da grade de translação está alinhada com a orientação da curva, sendo uma representação

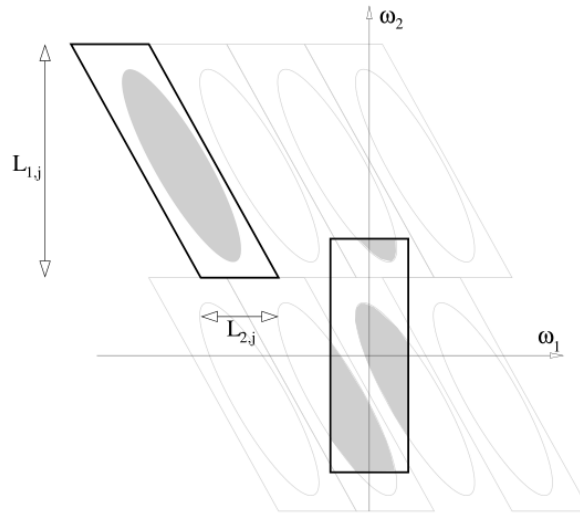


Figura 1.2: Ilustração do método *Wrapping*, que “embrulha” uma forma dentro de um paralelogramo ao assumir periodicidade (Candès et al., 2006).

mais precisa da transformada da curva em geral, embora um pouco mais lenta e complexa. Uma abordagem mais prática é a versão *Wrapping*, na qual a grade da função transformada de Fourier é multiplicada a cada j e l com a janela da equação 1.21. O resultado em janela é considerado periódico, envolvendo a origem, como ilustrado na Figura 1.2 (Candès et al., 2006). A transformada inversa de Fourier é aplicada à função envolvida em cada ângulo e escala, então os coeficientes da curvelet para aquela janela são obtidos por translação no domínio do espaço. Ambas as implementações para C++ e MATLAB podem ser encontradas na biblioteca *Curveletlab*, disponível em <http://curvelet.org>, sendo desenvolvida pelos criadores da transformada curvelet. A implementação usada foi o wrapper python chamado *curveletops* (Costa, 2020).

Assim, o procedimento básico da FDCT pode ser resumido alguns poucos passos. Como neste trabalho a versão *Wrapping* foi utilizada, segue o algoritmo básico deste método:

- Aplicar a transformada rápida de Fourier 2D (2D FFT) ao dado original, obtendo os coeficientes no domínio do número de onda,
- Para cada escala e ângulo, multiplicar ponto a ponto a janela de cisalhamento (Figura 1.1) com o dado no domínio de Fourier,
- “Embrulhar” (*wrap*) o resultado desta multiplicação em volta da origem,
- Aplicar a transformada rápida de Fourier inversa ao resultado, obtendo os coeficientes da transformada curvelet para esta escala e ângulo.

1.5 Redes Neurais Artificiais

As redes neurais artificiais tentam emular a forma como o cérebro humano funciona, ensinando o computador a resolver problemas que requerem um certo nível de cognição. Esses problemas são muito variados, indo desde identificação de padrões, interpretação de texto, criação artística, até mesmo previsão de dados históricos. Tais feitos são praticamente impossíveis de se modelar a partir de uma equação diferencial, ou de serem programados em uma lógica diretamente. A concepção da ideia da rede neural advém da biofísica matemática e da neurofisiologia, dentre as muitas tentativas de se entender e modelar matematicamente a forma como um neurônio funciona na esperança de entender o cérebro como um todo. Um dos trabalhos percursoros dessa linha de pesquisa foi o de Rashevsky (1938).

O neurônio humano, ilustrado na Figura 1.3, opera recebendo sinais elétricos de neurônios vizinhos através dos dendritos. De forma simplificada, o neurônio recebe um somatório de sinais elétricos de outros neurônios ligados aos dendritos, cada um com amplitudes diferentes dependendo da sua conexão sináptica, e o sinal resultante é avaliado: se for maior que um certo valor de corte o impulso é transmitido para o neurônio ligado ao axônio terminal, caso contrário nada acontece. Então o resultado é de fato binário, ou a sinapse acontece ou não, e sempre com a mesma amplitude de saída.

Observe então que existem que alguns parâmetros que regem a sinapse no neurônio: a amplitude de corte do sinal resultante bem como a força da conexão sináptica entre os dendritos com neurônios ligados eles. Os neurônios, ao se ligarem com outros, formam assim uma espécie de teia ou rede que formam nosso cérebro, o que justifica o nome “rede neural”.

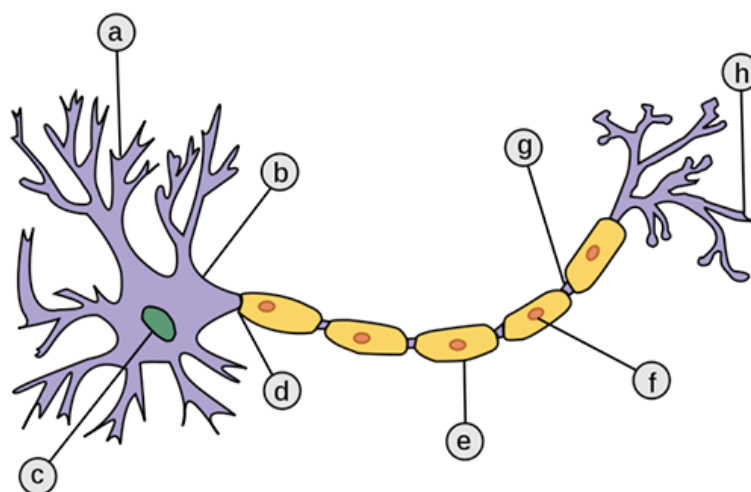


Figura 1.3: Esquema representativo de um neurônio típico. (a) Dendrito (b) Soma (c) Núcleo (d) Axônio (e) Bainha de mielina (f) Célula de Schwann (g) Nódulo de Ranvier (h) Axônio terminal (Moreira, 2013).

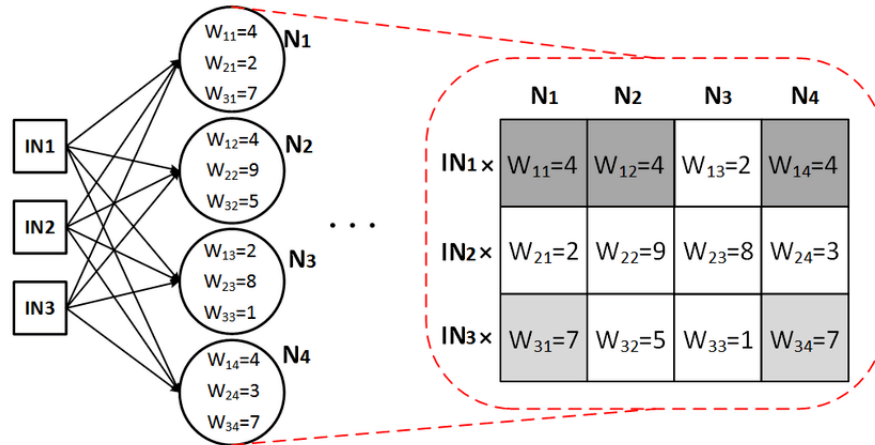


Figura 1.4: Uma rede neural simples e o mapeamento da primeira camada oculta numa matrix 4x3 (Yasoubi, Hojabr e Modarressi, 2017).

Os neurônios são de fato muito simples em sua estrutura, mas cada um dos parâmetros que eles controlam podem ser modelados matematicamente com algumas ressalvas. O restante da seção irá focar em explicar as diferentes etapas pertencentes à execução desse modelo.

1.5.1 Pesos

O peso é um dos parâmetros do neurônio que indica o nível de importância do sinal recebido pelo seu neurônio emissor. A Figura 1.4 mostra uma rede simples com uma camada de entrada com 3 neurônios, e a camada seguinte (camada oculta) com 4 neurônios. Cada neurônio da camada oculta recebe sinal de todos os neurônios de entrada, possuindo um peso associado a cada um deles (similar a conexão sináptica), indicando o nível de importância a ser dado por cada um. A matriz na Figura 1.4, que chamaremos de \mathbf{W}_1 , demonstra como esses pesos são aplicados aos valores de entrada. Observe que os pesos de cada neurônio da camada oculta estão dispostos nas colunas de \mathbf{W}_1 , e os valores da entrada no vetor sendo multiplicado por \mathbf{W}_1 . De um modo geral então temos até então

$$\mathbf{W}_1 \mathbf{x} = \mathbf{y}_1, \quad (1.22)$$

sendo x o vetor de entrada, e y o de saída da camada oculta. Essa operação, para cada neurônio nada mais é do que uma combinação linear entre pesos e os valores de entrada. Se tivéssemos outra camada oculta, teríamos outra matriz \mathbf{W}_2 e assim por diante, até que se chegasse na camada de saída.

1.5.2 Função de ativação

Somente a aplicação dos pesos não é o suficiente, pois o resultado tem que ser avaliado para que o neurônio decida se emite o impulso para o neurônio vizinho ou não. Como explicado anteriormente o sinal emitido é binário, e somente é transmitido se o valor resultante das aplicações dos pesos exceda um certo valor de corte. Esse comportamento é o de uma função degrau do tipo

$$H(x - \alpha) = \begin{cases} 0 & : x < \alpha, \\ \frac{1}{2} & : x = \alpha, \\ 1 & : x > \alpha, \end{cases} \quad (1.23)$$

onde α seria o valor de corte. Essa função usa como argumento o resultado da aplicação dos pesos, como ilustrado na Figura 1.5.

Observe porém que a função da equação 1.23 não é contínua, o que torna ela matematicamente indesejável, mesmo que seja mais coerente com a natureza. A solução encontrada é se utilizar outra função que tenha comportamento parecido e seja contínua, e uma das primeiras serem empregadas foi a função sigmoide

$$f(x) = \frac{1}{1 + e^{-x}}, \quad (1.24)$$

que tem comportamento parecido:

$$f(x) = \begin{cases} 0 & : x \rightarrow -\infty, \\ \frac{1}{2} & : x = 0, \\ 1 & : x \rightarrow +\infty. \end{cases} \quad (1.25)$$

Atualmente existem várias funções de ativação, e cabe ao responsável por projetar a rede escolher qual é a mais adequada. O exemplos mais comuns podem ser observados na Fi-

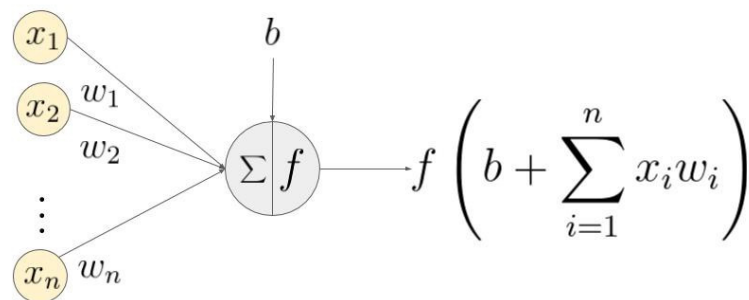


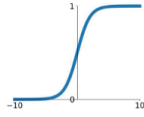
Figura 1.5: Exemplo das operações matemáticas que ocorrem dentro de um neurônio. x_i são as entradas, w_i são os pesos, n a quantidade de pesos e valores de entrada, b é o viés, e f é a função de ativação (Sharma, 2017).

gura 1.6. De um modo geral, a função de ativação serve para tornar a função que a rede aproxima não linear, o que evita noções preconcebidas sobre o modelo.

Funções de ativação

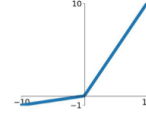
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



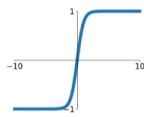
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$

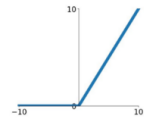


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ReLU

$$\max(0, x)$$



ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

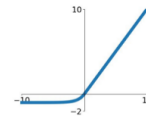


Figura 1.6: Diferentes tipos de função de ativação e seus respectivos gráficos (Jadon, 2018).

1.5.3 O viés

O viés, de forma simplificada, é um peso adicional somado à combinação linear dos pesos e valores de entrada, de forma a retirar o “viés” da função de ativação. Isso se traduz em duas implicações teóricas.

A primeira consiste na ideia que o viés funciona como o valor de corte do neurônio, transladando a função de ativação de acordo com a entrada. Observe que na subseção anterior o sigmoide, apresentado nas equações 1.24 e 1.25, bem como todas as funções de ativação na Figura 1.6, não possuem o valor de corte. Isso é proposital visto que ele deixa de ser necessário quando se usa o viés. Assim como $f(x - \alpha)$ é a translação de $f(x)$ por α , o viés b translada $\sigma(\sum_{i=0}^n w_i x_i + b)$, onde σ é a função de ativação. Isso permite aumentar o valor do viés de neurônios específicos que tenham como saída valores muito altos, mas que influenciam negativamente na acurácia da rede, servindo de fato como um equivalente de um valor de corte suavizado.

A segunda é que o viés permite se aumentar o grau de liberdade da rede. Como exemplo ilustrativo considere o caso de uma rede com somente dois neurônios exibido na Figura 1.7. Nesse caso, sem o viés a saída da rede é $y = \omega_0 x$ (desconsiderando a função de ativação), resultando em uma rede que consegue aproximar toda e qualquer função linear, se ela passar pela origem. Todavia, ao se adicionar o viés a saída da rede passa a ser $y = \omega_0 x + b$, sendo capaz de aproximar qualquer função linear no espaço \mathbb{R}^2 . O viés pode então também ser considerado como um peso adicional que aumenta a adaptabilidade da rede para além do espaço da combinação linear dos pesos e valores de entrada.

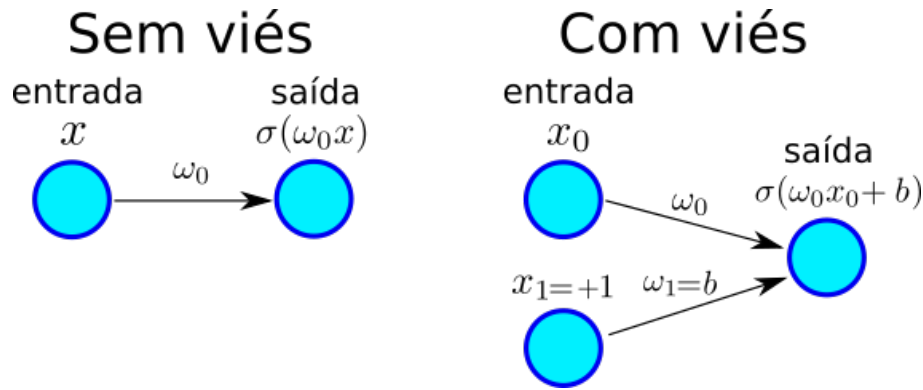


Figura 1.7: Comparação de como funciona uma rede neural com dois neurônios com e sem bias.

O viés e a função de ativação atuam em conjunto para aumentar o grau de liberdade e adaptabilidade da rede. A função de ativação torna a rede não linear, e o viés aumenta o seu grau de liberdade e também atua como um valor de corte.

1.5.4 Função custo

Com a aplicação dos pesos e da função de ativação para todas as camadas, a saída é o resultado da rede. Porém, para que este resultado se torne mais próximo do valor esperado, os pesos e os bias tem que ser modificados de tal forma a se cumprir esse objetivo. A função custo tem como princípio avaliar o quão longe a rede atual está da desejada, para que seja possível calcular quais devem ser os incrementos aos pesos atuais. Em geral isso é feito através de uma norma, sendo a norma L_2 a mais comum. Então a função custo pode ser expressa através de

$$E(\mathbf{f}) = \|\mathbf{f}(\mathbf{x}) - \mathbf{y}\|_2^2. \quad (1.26)$$

A contribuição para uma próxima iteração será calculada então com o cálculo do gradiente do erro com relação aos parâmetros, da mesma forma que o método do gradiente ou *steepest descent* é usado para resolver problemas de minimização. O motivo pelo qual a função de ativação tem que ser contínua é justamente para que ela possa ser derivada parcialmente com relação a cada parâmetro dentro do processo de otimização.

Ao se apresentar a rede vários pares de dados de entrada e saída, a mesma tende a escolher pesos ótimos para que se tenha a função custo mínima para todo o espaço amostral. Esse processo de otimização se chama de treinamento. Ao se obter os pesos da rede treinada, ela pode ser então aplicada a dados desconhecidos, e essa etapa se chama de inferência. O objetivo final do treinamento é se obter uma acurácia ótima na fase de inferência.

Existem várias particularidades para maximizar a acurácia. Uma delas é tomar medidas

para que erro da função custo não se torne excessivamente baixo, pois torna o modelo pouco adaptável à amostras futuras. O nome desse fenômeno é sobreajuste (em inglês, *overfit*). Levando isso em consideração, geralmente se separa parte das amostras para validação da acurácia da rede. Existem muitas outras técnicas comumente empregadas, mas falar sobre todas elas foge ao escopo dessa seção.

1.6 Redes Neurais Convolucionais

A rede neural convolucional (CNN) visa analisar dados espaciais de forma mais eficiente. Isso é especialmente útil quando os dados de entrada do problema são uma imagem ou matriz multidimensional, pois permite que as camadas da rede identifiquem feições espaciais na imagem, em muitas escalas e localizações diferentes. Na rede neural tradicional totalmente conectada, a imagem de entrada deve ser um vetor unidimensional, comprometendo a compreensão de informação espacial, uma vez que a imagem terá de ser achatada em uma dimensão. Além disso, a CNN está conectada apenas localmente, na qual um pequeno filtro de pesos diferentes é convolvido com a imagem de entrada, resultando no valor de um neurônio na camada seguinte (com a função de ativação devidamente aplicada). Consequentemente, um neurônio é conectado apenas a uma pequena porção de neurônios da entrada, reduzindo o número de parâmetros necessários significativamente (Goodfellow, Bengio e Courville, 2016). Embora seja possível simplesmente usar uma rede neural profunda totalmente conectada para resolver problemas de imagem, o que é conhecido a partir do teorema da aproximação universal (Hornik, Stinchcombe e White, 1990), isso trás consigo um alto preço tanto em desempenho quanto em eficácia.

O uso mais proeminente da CNN está no campo da visão computacional, justamente porque permite que a máquina identifique feições nas imagens, que podem ser desde números, formas, faces, arestas, objetos, bem como lista contínua. A primeira aplicação conhecida desta arquitetura foi feita por Cun et al. (1989), com o objetivo de projetar uma rede para identificar dígitos manuscritos. Isso só é possível porque esse tipo de rede permite que os filtros extraiam feições dos dados de entrada independentemente de sua posição, ou seja, por causa da invariância translacional.

Em cada camada da CNN, uma infinidade de filtros com pesos diferentes é gerada, e cada filtro é convolvido com a imagem de entrada e adicionado a um viés. Os resultados de tal convolução são chamados de mapas de feições. Assim, uma coleção de mapas de recursos associados a diferentes filtros é calculada. Uma função de ativação é então aplicada a eles para garantir a não linearidade, que pode ser sigmóide, tangente hiperbólica, ReLU

ou outros. Esta operação pode ser expressa por

$$Y_i = \sigma(W_i * X + b_i) \quad i \in \mathbf{N} : 0 < i < N_{\text{filtros}}, \quad (1.27)$$

em que Y_i é o mapa de feições de saída associado ao filtro W_i e o bias b_i , e σ é a função de ativação. Os parâmetros são ajustados usando o método *steepest descent* para minimizar a função de perda usada em um conjunto de dados. Uma CNN de aprendizado profundo é basicamente uma combinação de camadas convolucionais. Existem várias arquiteturas de rede neural que usam uma mistura de camadas convolucionais com outros tipos para obter um resultado melhor em alguns campos (por exemplo, U-Net).

O projeto das arquiteturas de CNN é fortemente baseado em evidências empíricas do que funciona no campo da visão computacional e em outros campos importantes. O projeto consiste no número de camadas convolucionais, número de filtros por camada, tamanho do filtro, tipo de função de ativação, se a camada atuará como um codificador ou decodificador, ou se haverá outros tipos de camadas no intervalo entre camadas convolucionais (Liu, Chen e Schuster, 2020). Uma comparação entre CNNs e codificação esparsa convolucional multicamadas foi recentemente estudada por Papyan, Romano e Elad (2016), dando um formalismo matemático a esse tipo de rede neural. Ainda assim, dependendo do problema em questão, algum nível de experimentação heurística será necessário. Como consequência, os parâmetros selecionados talvez funcionem para um conjunto de dados, mas muito menos para outros.

1.6.1 Arquitetura U-Net

Enquanto as camadas CNN mantêm as dimensões dos mapas de feições fixas como as mesmas da imagem de entrada e saída, pode-se também reduzir a resolução até a camada de gargalo e, em seguida, aumentar a resolução de volta à dimensão original (Lucas et al., 2018). Essa abordagem se tornou popular com problemas de segmentação. Um esquema visual de como redes convolucionais neste formato de codificação-decodificação pode ser observado na Figura 1.8. A parte de redução da resolução costuma ser chamada de caminho de codificação e é responsável por aprender conceitos abstratos sobre a imagem; enquanto a parte aumento da resolução é chamada de caminho de decodificação e usa a abstração do caminho de codificação e produz a imagem de saída. Essa abordagem é chamada de CNN codificador-decodificador. No entanto, a parte do codificador comprime as informações espaciais, o que pode resultar em uma perda significativa de detalhes (Lucas et al., 2018). Uma solução proposta por Ronneberger, Fischer e Brox (2015a, 2015b) é introduzir conexões de salto no caminho de decodificação, que se resume a concatenar mapas de feições das camadas convo-

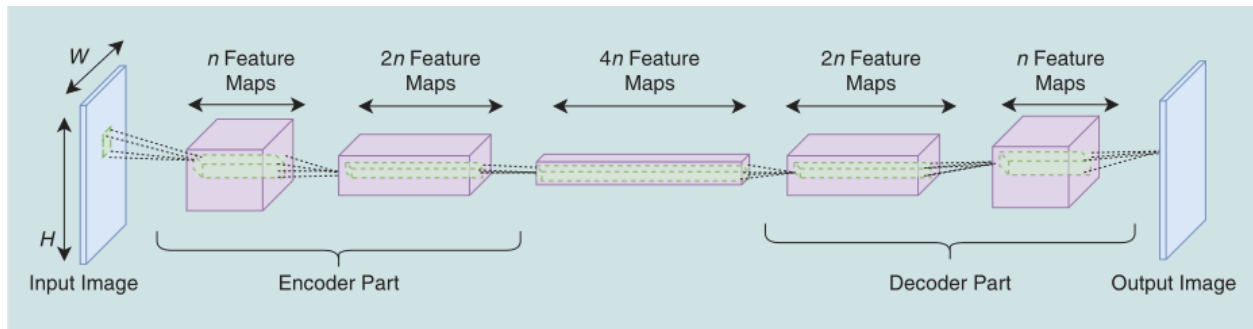


Figura 1.8: Numa CNN no formato codificação-decodificação, os mapas de feições são espacialmente comprimidos por uma rede de codificação, e então tem seu tamanho incrementado de volta para o tamanho da imagem de saída (Lucas et al., 2018).

lucionais do caminho de codificação e decodificação com as mesmas dimensões, preservando os detalhes da imagem. Essa arquitetura é chamada de U-Net.

No entanto, a topologia U-Net é originalmente ajustada para o propósito de segmentação de imagem, o que requer algumas mudanças para que possa ser usado para o problema de regressão enquanto ainda colhendo dos benefícios de sua abordagem multi-escala. O primeiro problema é que a função de ativação original usada foi ReLU⁶, a qual zera qualquer número negativo, o que é indesejável para a regressão, uma vez que os resultados são números reais com a amplitude da imagem. A função de ativação mais promissora é, na verdade, a tangente hiperbólica, que permite números positivos e negativos, mas não todo o eixo real. Ainda assim, usar apenas tangente hiperbólica fará com que as amplitudes de saída sejam muito diferentes das de entrada, uma vez que orbitarão em torno de zero, apresentando um resultado de função de alto custo. A solução é usar uma função de ativação linear na última camada convolucional, para fazer os resultados de saída se espalharem por todo o eixo real. O segundo problema é, uma vez que apenas um par de imagens será usado, mesmo com janelamento espacial, o número original de blocos e parâmetros na U-Net original é muito grande, fazendo com que o processo de treinamento demore muito e exija uma quantidade enorme de épocas para concluir. Para resolver isso, um número menor de núcleos é usado em cada camada convolucional. Ao fazer essas adaptações, a rede U-Net pode ser usada como uma arquitetura de rede neural de regressão. Este é o raciocínio por trás do filtro usado por Avila et al. (2021), conforme será explicado no capítulo 2.

O caminho de codificação é responsável por aprender a representação abstrata da imagem de entrada, contendo três blocos de três camadas. No início de cada bloco, existem duas camadas convolucionais convencionais com 3×3 núcleos de filtro. Este par de camadas convolucionais é responsável pela extração de feições na escala atual. A próxima camada

6. ReLU é especialmente bom para problemas de segmentação, uma vez que sua natureza de responder apenas a números positivos é na verdade muito semelhante à maneira como nosso cérebro funciona.

divide a resolução espacial pela metade, usando um algoritmo de *max pooling* (Nagi et al., 2011) com uma janela 2×2 . Esta diminuição na resolução permite uma diminuição no número de parâmetros e torna mais grossa a escala em que a camada convolucional do próximo bloco irá operar sobre. Ao final de cada bloco, existe uma camada de *dropout*, responsável por desativar alguns nós aleatoriamente durante o treinamento, como forma de evitar o *overfitting*⁷. Cada passagem de bloco tem seus parâmetros conectados ao próximo bloco com a mesma sequência de camadas. A diferença entre blocos consecutivos é que, a cada repetição, o número de mapas de feições nas camadas convolucionais é duplicado, para melhorar o reconhecimento de feições mesmo com o engrossamento da escala. O número de blocos no caminho de codificação do algoritmo original é quatro.

O caminho de decodificação é responsável por unificar informações de diferentes escalas, contendo quatro blocos de quatro camadas. No início de cada bloco, ocorre uma operação de convolução de aumento de escala, que consiste em uma camada que dobra a resolução da entrada seguida por uma camada de convolução. Como consequência, a escala que está sendo resolvida fica mais grosseira. Posteriormente, os mapas de feições na mesma escala do caminho de codificação são concatenados com a saída da convolução de aumento de escala, que é seguida por duas camadas de convolução, terminando o bloco. Como antes, o número de mapas de feições para as camadas de convolução em cada escala muda com cada bloco, embora agora ao contrário. Ele começa com o mesmo número do último bloco no caminho de codificação e divide seu número a cada bloco que passa até atingir o número da escala mais fina. No último bloco, há um par de camadas de convolução com 32 mapas de feições e outra com uma função de ativação linear, que fornece a imagem de saída.

Para resumir, a arquitetura U-Net empregada neste trabalho, explicado no capítulo 2, difere da versão original e da proposta por Avila et al. (2021). Isso mostra que a arquitetura U-Net não é imutável, apenas descrevendo um modelo convolucional com caminhos de codificação-decodificação e, geralmente, com camadas de concatenação.

7. Em teoria de inversão, *overfit* é a situação onde o modelo invertido se aproxima demais do real, tornando os seus parâmetros perderem generalidade. Ou seja, os parâmetros somente vão servir para aquele modelo.

2

Metodologia

A ideia por trás do LSRTM, usando-se uma linearização da modelagem, é que os dados podem ser obtidos a partir da refletividade por meio de:

$$\mathbf{d} = \mathbf{L}\mathbf{m}, \quad (2.1)$$

onde \mathbf{d} são os dados, \mathbf{m} é a refletividade e \mathbf{L} é um operador de modelagem linear. Considerando-se que $(\mathbf{L}^T\mathbf{L}) = I$, ou seja, $\mathbf{L}^T = \mathbf{L}^{-1}$, o operador de migração é o adjunto da modelagem, representado por \mathbf{L}^T . No entanto, operador adjunto não é o mesmo que o de inversão (ou pseudo-inversão neste caso), apesar da similaridade em muitos problemas de propagação de ondas. Em termos mais simples, $\mathbf{L}^T \neq \mathbf{L}^{-1}$. De acordo com a teoria do método dos mínimos quadrados, a pseudo-inversa seria $\mathbf{L}^{-1} = (\mathbf{L}^T\mathbf{L})^{-1}\mathbf{L}^T$ (Claerbout, 1992). Como consequência, a migração pode ser expressa pelas seguintes equações:

$$\mathbf{m}_{\text{mig}} = [\mathbf{L}_1^T \mathbf{L}_2^T \cdots \mathbf{L}_{N_s}^T] \begin{bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \\ \vdots \\ \mathbf{d}_{N_s} \end{bmatrix} = \sum_{i=1}^{N_s} \mathbf{L}_i^T \mathbf{d}_i, \quad (2.2)$$

em que a imagem migrada é o resultado do empilhamento de migrações de tiros individuais, sendo N_s o número de tiros (Dai e Schuster, 2013). Portanto, o problema de inversão seria minimizar a seguinte função de custo:

$$E(\mathbf{m}) = \frac{1}{2} \sum_{i=1}^{N_s} \|\mathbf{L}_i \mathbf{m} - \mathbf{d}_i\|^2. \quad (2.3)$$

Finalmente, a migração de mínimos quadrados pode ser expressa como:

$$\mathbf{m}_{MQ} = \sum_{i=1}^{N_s} (\mathbf{L}_i^T \mathbf{L}_i)^{-1} \mathbf{L}_i^T \mathbf{d}_i = \sum_{i=1}^{N_s} \mathbf{H}_i^{-1} \mathbf{m}_{\text{mig},i}, \quad (2.4)$$

no qual $\mathbf{m}_{\text{mig},i}$ são migrações de tiro individuais, e $\mathbf{H}_i = \mathbf{L}_i^T \mathbf{L}_i$ é a Hessiana da teoria da inversão. Da equação 2.4 pode ser pensado que a inversa da Hessiana (\mathbf{H}^{-1}) é um filtro de focalização da imagem migrada. A migração de mínimos quadrados visa aproximar a inversa da Hessiana usando métodos de otimização. A abordagem padrão consiste em métodos iterativos, como conjugado do gradiente (CG) ou *steepest descent*, uma vez que não é viável encontrar todos os coeficientes da matriz \mathbf{H}^{-1} , estando muito além de nossos recursos computacionais atuais. No entanto, tem trabalhos que mostram que são 4 modelagens e migrações por iteração, e quando aplicados a métodos como o LSRTM de onda plana tem custo computacional equivalente a 12 RTMs convencionais (Liu e Peter, 2018).

Este trabalho tenta usar filtros de correspondência convolucional para aproximar a inversa da matriz Hessiana (\mathbf{H}^{-1}). Assim, um tipo de filtro é aplicado da seguinte maneira:

$$\mathbf{m}_{MQ} = \mathbf{f} * \mathbf{m}_{\text{mig}}. \quad (2.5)$$

O filtro \mathbf{f} é então encontrado a partir da minimização da seguinte função de custo

$$E(\mathbf{f}) = \|\mathbf{f} * \mathbf{m}_2 - \mathbf{m}_1\|_2^2, \quad (2.6)$$

onde \mathbf{m}_1 é a imagem RTM ($\mathbf{m}_1 = \mathbf{L}^T \mathbf{d}$), e \mathbf{m}_2 é o resultado da imagem da migração dos dados modelados a partir de \mathbf{m}_1 ($\mathbf{m}_2 = \mathbf{L}^T \mathbf{L} \mathbf{m}_1$) (Guittou, 2004). Esse tipo é chamado de filtro no domínio de imagem. Outro tipo de filtro de iteração única proposto por Liu e Peter (2018) são filtros no domínio do dado em que a função de custo é

$$E(\mathbf{f}) = \|\mathbf{f} * \mathbf{d}_1 - \mathbf{d}\|_2^2, \quad (2.7)$$

no qual \mathbf{d}_1 é o resultado da modelagem da imagem migrada ($\mathbf{d}_1 = \mathbf{L}(\mathbf{L}^T \mathbf{d})$).

Este trabalho compara dois diferentes filtros convolucionais do tipo domínio da imagem (equação 2.5). A primeira abordagem será explicada na seção 2.1 e a segunda na seção 2.2.

O procedimento descrito foi aplicado em um par de dois modelos acústicos famosos, o Marmousi e o Sigsbee2B. Ambos modelos foram usados para criar agrupamentos de tiros sísmicos sintéticos com 80 e 200 tiros, respectivamente, utilizando a modelagem Born, explicada na seção 1.3. A partir dos tiros sintéticos, foram criados pares de imagens migradas e remigradas, que serviram de entrada para obtenção dos filtros a serem discutidos a seguir.

2.1 Abordagem baseada na rede U-Net (Avila et al., 2021)

Esta abordagem consiste simplesmente em usar uma rede neural convolucional para atuar como um filtro de focalização entre a imagem migrada e a imagem remigrada, aproximando-

se \mathbf{H}^{-1} . Em essência, os pesos da rede neural (U_{net}) são encontrados por regressão do par

$$\mathbf{m}_1 = U_{\text{net}}(\mathbf{m}_2), \quad (2.8)$$

A rede neural é treinada com um banco de dados formado por um conjunto de janelas extraídas aleatoriamente do par \mathbf{m}_1 e \mathbf{m}_2 de tamanho $N \times N$, com N variando para o tamanho da imagem de entrada. O modelo treinado é então aplicado à imagem migrada com a janela

$$\mathbf{m}_{MQ} = U_{\text{net}}(\mathbf{m}_1) \quad (2.9)$$

e reconstruído para encontrar uma aproximação da imagem de mínimos quadrados.

O modelo original pode ser visualizado na Figura 2.1. No entanto, algumas pequenas modificações ocorreram:

- Algumas camadas de *dropout* foram inseridas na fase de codificação, o que não é expresso na Figura 2.1,
- Na fase de pré-processamento, o algoritmo de normalização foi alterado de máximo absoluto¹ para um escalonador robusto²,
- Os tamanhos das janelas, o número delas e o algoritmo para obtê-las são provavelmente diferentes,
- Os algoritmos de migração e demigração (explicados nas seções 1.1 e 1.3 respectivamente) foram escritos pelo autor deste trabalho em sua forma básica, otimizados para GPU³ em CUDA C++(NVIDIA, Vingelmann e Fitzek, 2020), usando a API de Madagascar (Fomel et al., 2013), então provavelmente haverá uma diferença na implementação.
- Regularizadores de pesos de camadas⁴ foram usados nas últimas 3 camadas convolucionais, com um fator de regularização de 10^{-4} e norma L1, para garantir um certo nível de esparsidade nos dados de saída.

O layout exato das camadas pode ser encontrado na Tabela 2.1. A razão para alterar o algoritmo de escalonamento é simplesmente porque o escalonador máximo absoluto introduziu muito ruído na saída do modelo. O objetivo de janelar a imagem antes de aplicar a

1. Fonte do Scikit-learn: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MaxAbsScaler.html>.

2. Fonte do Scikit-learn: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html>.

3. Unidade de processamento gráfico.

4. Keras API source: <https://keras.io/api/layers/regularizers/>

rede neural é garantir algum nível de localidade para o filtro. Além disso, ajuda a prevenir o *overfit*, embora este seja inevitável com um número de épocas / iterações suficientes. As camadas de *dropout* foram inseridas no modelo a fim de resolver este problema.

Nome da camada	Tipo de camada	F. ativação	Kernel tamanho/salto	Tamanho do output
Conv1a	Convolutacional	tanh	$3 \times 3/1$	$N \times N \times 32$
Conv1b	Convolutacional	tanh	$3 \times 3/1$	$N \times N \times 32$
Maxpool1	Max pooling	–	$2 \times 2/2$	$N/2 \times N/2 \times 32$
Conv2a	Convolutacional	tanh	$3 \times 3/1$	$N/2 \times N/2 \times 64$
Conv2b	Convolutacional	tanh	$3 \times 3/1$	$N/2 \times N/2 \times 64$
Maxpool2	Max pooling	–	$2 \times 2/2$	$N/4 \times N/4 \times 64$
Conv3a	Convolutacional	tanh	$3 \times 3/1$	$N/4 \times N/4 \times 128$
Conv3b	Convolutacional	tanh	$3 \times 3/1$	$N/4 \times N/4 \times 128$
Maxpool3	Max pooling	–	$2 \times 2/2$	$N/8 \times N/8 \times 128$
Conv4a	Convolutacional	tanh	$3 \times 3/1$	$N/8 \times N/8 \times 128$
Conv4b	Convolutacional	tanh	$3 \times 3/1$	$N/8 \times N/8 \times 128$
Up-conv1	Upsampling + Conv	tanh	$2 \times 2/2$	$N/4 \times N/4 \times 128$
Drop1	Dropout(0.5)	–	–	$N/4 \times N/4 \times 128$
Concatenate1	Cat (Up-conv1,Conv3b)	–	–	$N/4 \times N/4 \times 256$
Conv5a	Convolutacional	tanh	$3 \times 3/1$	$N/4 \times N/4 \times 128$
Conv5b	Convolutacional	tanh	$3 \times 3/1$	$N/4 \times N/4 \times 128$
Up-conv2	Upsampling + Conv	tanh	$2 \times 2/2$	$N/2 \times N/2 \times 64$
Drop2	Dropout(0.5)	–	–	$N/4 \times N/4 \times 64$
Concatenate2	Cat (Up-conv2,Conv2b)	–	–	$N/2 \times N/2 \times 128$
Conv6a	Convolutacional	tanh	$3 \times 3/1$	$N/2 \times N/2 \times 64$
Conv6b	Convolutacional	tanh	$3 \times 3/1$	$N/2 \times N/2 \times 64$
Up-conv3	Upsampling + Conv	tanh	$2 \times 2/2$	$N \times N \times 32$
Concatenate3	Cat (Up-conv3,Conv1b)	–	–	$N \times N \times 64$
Conv7a	Convolutacional	tanh	$3 \times 3/1$	$N \times N \times 32$
Conv7a	Convolutacional	tanh	$3 \times 3/1$	$N \times N \times 32$
Conv8	Convolutacional	linear	$3 \times 3/1$	$N \times N \times 1$

Tabela 2.1: Descrição da topologia U-Net utilizada, baseada no trabalho de Avila et al. (2021).

Em suma, o procedimento consistiu em:

1. Entrada de m_1 e m_2 ,
2. Normalização com o algoritmo de escalonamento robusto,
3. Extração das janelas,
4. Validação e divisão de dados de treinamento,
5. Treinamento,
6. Inferência da imagem filtrada dos patches de m_1 ,
7. Mesclagem de patches resultantes na imagem,

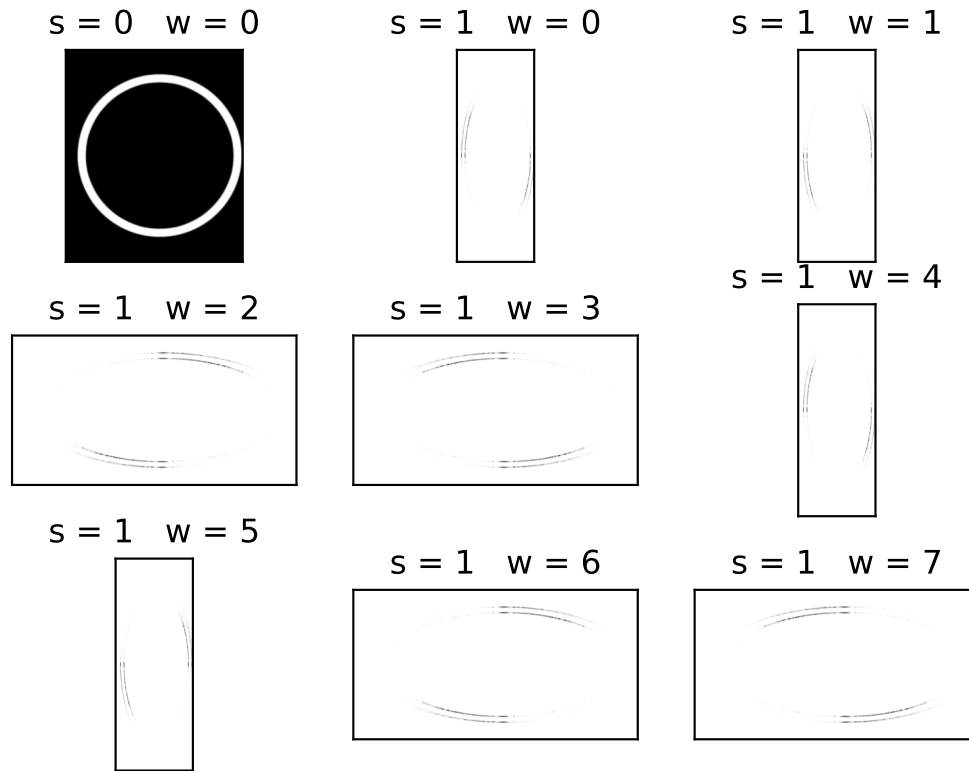


Figura 2.2: Representação de domínio curvelet de uma imagem de um círculo em 2 escalas com 8 fatias angulares na primeira escala fora do kernel. Os símbolos s e w representam os números de escala e fatias, respectivamente. Observe como as diferentes fatias captam diferentes ângulos do círculo. As amplitudes são o valor absoluto dos coeficientes complexos, e o resultado foi elevado ao quadrado e convolvido com um filtro gaussiano para garantir a legibilidade.

de espaço modificado contendo os coeficientes curvelet. Tome por exemplo a Figura 2.2, onde uma imagem de um círculo é representada no domínio curvelet. A escala 1 e a fatia 1 têm mais ou menos o mesmo aspecto da imagem original, mas na escala 1 vemos que a cada aumento do número da fatia, uma seção angular diferente é representada. Outro exemplo pode ser visto na Figura 2.3, onde a imagem migrada do modelo Sigsbee2B é representada no domínio curvelet, com refletores de diferentes inclinações sendo mostrados em diferentes feições curvelet. Se o número de escalas for incrementado, conforme o índice da escala aumenta, mais precisa se torna a resolução da escala.

Ainda assim, a mesma arquitetura U-Net não pode ser usada, pois precisa de uma entrada com ambos os eixos sendo números pares divisíveis 2^{n_m} , sendo n_m o número de camadas de *max-pooling* usadas. Isso porque esta camada, da forma como foi configurada,

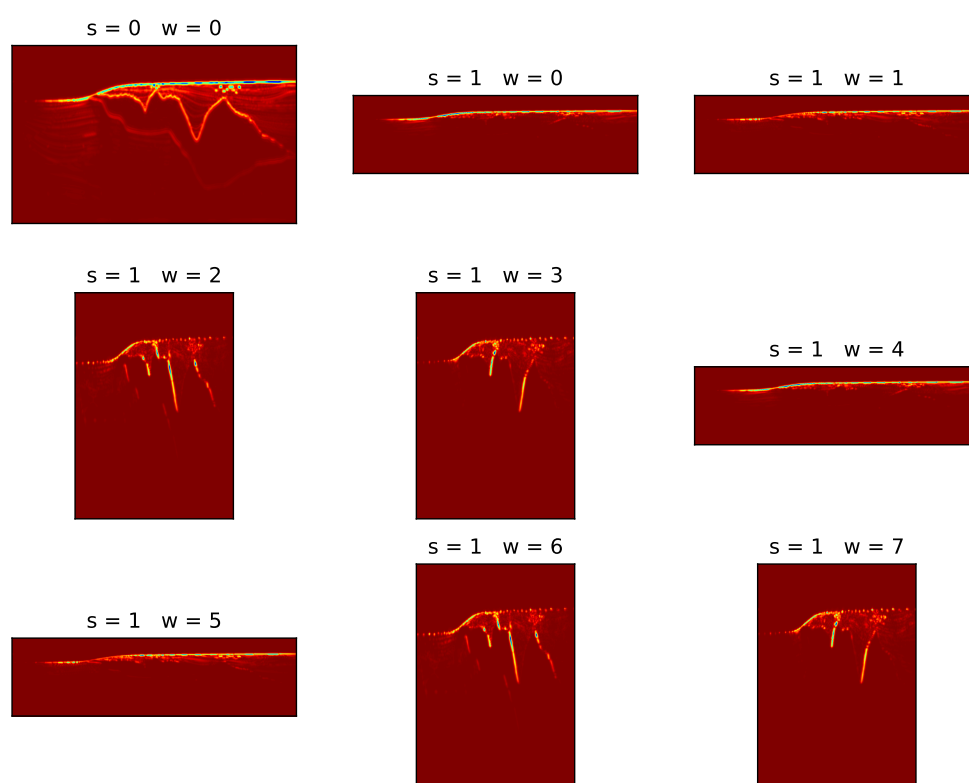


Figura 2.3: Representação da imagem migrada a partir do modelo Sigsbee2B no domínio curvelet, com 2 escalas e 8 fatias angulares na primeira escala fora do kernel. Observe como diferentes fatias captam refletores de inclinações variadas. As amplitudes são o valor absoluto dos coeficientes complexos, e o resultado foi convolvido com um filtro gaussiano e transposto para garantir a legibilidade.

assume o valor máximo em uma janela deslizante 2×2 , fazendo com que a saída da fase de codificação tenha o novo tamanho de cada dimensão como uma aproximação para baixo da relação da dimensão da entrada dividido pelo número de camadas de *max-pooling*, ou seja, para uma entrada com dimensões $N_1 \times N_2$ temos

$$N'_i = \left\lfloor \frac{N_i}{n_m} \right\rfloor. \quad (2.10)$$

Isso significa que se o formato da entrada for 9×9 , após um *max-pooling*, ele se tornará 4×4 e então 2×2 após mais um, ponto em que se tornará incapaz de prosseguir com o algoritmo. Além disso, na fase de decodificação, o algoritmo de aumento do número de amostras (*upsampling*) usado simplesmente repete as linhas e colunas da imagem de entrada, o que geralmente é seguido por uma camada convolucional. Em outras palavras, se a camada de entrada do modelo for 18×18 , após três camadas de *max-pooling* seguidas de *upsampling*, a saída terá 16×16 , o que tornará essas duas camadas incompatíveis para serem concatenadas. Isso é problemático, especialmente porque as formas das feições curvelet não têm garantia de obedecer a essas condições.

Para resolver isso, algumas adaptações foram feitas. A primeira é substituir a camada de *upsampling* por um de aumento de resolução por interpolação⁵. Essa camada foi configurada para redimensionar a matriz para a mesma forma que a saída da camada convolucional da fase de codificação usando interpolação bilinear. Ao fazer isso, a camada de concatenação tem a garantia de ter a mesma forma para operar. A segunda adaptação consiste em usar um número variável de camadas de *max-poolings* e *upscaling*, reduzindo a profundidade da fase de codificação. Dessa forma, mesmo se usarmos de muitas escalas na transformada curvelet de modo que o array das feições curvelet comecem a ficar muito pequenos, o modelo só diminuirá a escala se a saída resultante respeitar um certo limite. Mais precisamente, se uma convolução mais um *max-pooling* compõem um bloco de codificação, o número de blocos é dado por $\min(\lfloor \log_2 \minLen - 1 \rfloor, 3)$, enquanto *minLen* é o menor comprimento do eixo na geometria da imagem de entrada. Se o número de blocos for menor que um, esse recurso não é otimizado. A terceira adaptação é treinar a rede com o valor absoluto dos coeficientes curvelet, uma vez que são números complexos. A fase é preservada para reconstrução posterior. A quarta e última adaptação é que cada feição curvelet terá seu próprio modelo para otimizar. O modelo é descrito no algoritmo 1.

O otimizador Adam (Kingma e Ba, 2014) mencionado no algoritmo 1 é um método de gradiente estocástico. O Adam é o descendente de uma família de algoritmos de otimização baseados no *steepest descent* ou gradiente. No método do gradiente convencional, a convergência se dá calculando o gradiente da função custo para se determinar a direção ótima,

5. Fonte do Tensorflow: https://www.tensorflow.org/api_docs/python/tf/image/resize

Algorithm 1 Modelo U-Net no domínio curvelet

Require: $C_i \in \mathbb{C}$ feições curvelet com forma $M \times N$

for C_i in \mathbf{C} **do**

$minLen \leftarrow \min(M, N)$

$UNETBlocks \leftarrow \min(\lfloor \log_2 minLen - 1 \rfloor, 3)$

$f \leftarrow f_o := 16$ número dos mapas de características iniciais

$p \leftarrow C_i$

if $UNETBlocks \geq 1$ **then**

Encoding loop

for $j := 1, UNETBlocks$ **do**

$c_j \leftarrow \text{Conv}_{3 \times 3 \times f}(p, \sigma = \tanh)$

$p \leftarrow \text{MaxPool}_{2 \times 2}(c_j)$

$f \leftarrow f * 2$

end for

$m \leftarrow p$

$f \leftarrow \lfloor \frac{f}{2} \rfloor$

for $j := 1, UNETBlocks$ **do**

$c \leftarrow \text{Conv}_{3 \times 3 \times f}(m, \sigma = \tanh)$

$u \leftarrow \text{Upscale}(c, shape = shape(c_{-j}))$

$m \leftarrow \text{Concatenate}(c_{-j}, u)$

$f \leftarrow \lfloor \frac{f}{2} \rfloor$

end for

$c \leftarrow \text{Conv}_{3 \times 3 \times 1}(m, \sigma = \text{linear}, regularizer = L_1(10^{-4}))$

output $y_i \leftarrow c$

else

output $y_i \leftarrow C_i$

end if

end for

* c_{-j} representa a saída da camada convolucional contando ao contrário a partir da última iteração da fase de codificação, em direção a primeira iteração, sendo o mesmo que usar indexação reversa de uma sequência,

* $\text{Conv}_{3 \times 3 \times f}(p, \sigma = \tanh)$ representa uma camada convolucional com uma coleção de f janelas de kernel deslizantes com 3×3 , aplicado ao tensor p com função de ativação tanh,

* $\text{MaxPool}_{2 \times 2}$ representa uma camada de *max-pooling* com uma janela deslizando com 2×2 ,

* $\text{Upscale}(c, shape = shape(c_{-j}))$ representa uma camada de aumento de resolução que interpola o tensor c para as dimensões do parâmetro *shape*,

* $\text{Concatenate}(c_{-j}, u)$ representa uma camada de concatenação que emenda os tensores c_{-j} e u na sua terceira dimensão,

* $regularizer = L_1(10^{-4})$ é um parâmetro de regularização configurado para ser a norma L_1 com um fator de 10^{-4} ,

* Esse modelo usa o otimizador Adam.

porém conforme a função custo avança em direção ao mínimo global, o ruído de derivadas calculadas no passado interferem no cálculo da direção ótima nas últimas iterações. Como consequência, a convergência próximo ao mínimo global é lenta. Uma solução para esse problema é empregar o conceito de momento, que de forma similar a uma bola rolando em direção ao ponto mais baixo de um vale, que na parte mais inclinada desce com alta velocidade e momento, mas conforme se tem uma diminuição da inclinação a bola perde momento até descansar na parte mais profunda. Em suma, momento permite um peso maior para derivadas mais recentes no momento de se calcular a direção de convergência. O método Adam vai além e utiliza momentos adaptativos, com passos no espaço de parâmetros limitados por um hyper parâmetro que controla esses passos. De um modo geral, é um método robusto, com custo de memória relativamente menor, funciona bem com gradientes esparsos, e provê uma melhor velocidade de convergência.

Resumindo, o procedimento consiste dos seguintes passos:

1. Entradas de m_1 e m_2 ,
2. Transformada curvelet de ambos,
3. Separação dos coeficientes complexos em amplitude e fase,
4. Normalização das amplitudes usando escalonamento robusto,
5. Treinamento com as amplitudes,
6. Inferência com as amplitudes da imagem migrada,
7. Reconstrução de coeficientes complexos usando a saída da última etapa e a fase original,
8. Transformada inversa,
9. Escalonamento inverso na imagem final.

3

Resultados e discussão

Nesta seção, os resultados da aplicação dos métodos discutidos aos modelos 2D Marmousi e Sigsbee2B são mostrados e discutidos.

3.1 Marmousi 2D

O modelo de Marmousi não é tão profundo, tendo apenas em torno de 2.5km, porém, possui refletores muito inclinados, o que testará a direcionalidade dos filtros. O dado associado ao modelo utilizado tem espaçamentos $dx = 25$ m e $dz = 8$ m, dimensões de $nx = 368$ e $nz = 375$. O conjunto de tiros utilizados para a migração RTM foram criados empregando a modelagem Born a partir da refletividade do modelo. A condição de imagem foi a convencional explicada na seção 1.2.2. Foram criados 80 tiros, com 80 receptores em cada lado do tiro, numa geometria de aquisição *split-spread*. Esses tiros foram migrados e então remigrados, aplicados o filtro laplaciano, servindo então de input para os métodos de filtragem discutidos previamente.

O experimento teve os seguintes parâmetros. Por um lado, o modelo U-Net no domínio do espaço usou 300 janelas de tamanho 32×32 , com um salto de 10 unidades em ambas as direções. A validação por divisão de dados de treinamento foi de 0, 2 e a taxa de aprendizagem foi $lr = 0,0001$. Por outro lado, o modelo U-Net no domínio curvelet usou uma taxa de aprendizagem de $lr = 0,01$, $nbs = 7$ e $nbangles = 4$ escalas de transformada curvelet e número de fatias na escala mais grossa além do kernel ($s = 1$) respectivamente.

Os dois filtros apresentaram resultados distintos para o modelo Marmousi, como pode ser observado nas Figuras 3.2 e 3.4.

Por um lado, a imagem migrada filtrada com o modelo U-Net no domínio espacial apresenta uma melhor resolução dos refletores e balanceou a iluminação em profundidade, como pode ser observado na Figura 3.2b, quando comparada com a Figura 3.2a. No entanto, o filtro não foi capaz de melhorar muito a resolução de refletores fora da área iluminada pela geometria de modelagem. É possível observar também que algumas reflexões ao meio do modelo ficaram mais realçadas. Na Figura 3.3 observa-se um leve aumento de amplitude dos refletores em profundidade, o suficiente para ser visível sem o zoom. O mesmo comportamento pode ser observado tanto em profundidade (Figuras 3.3a e 3.3b) quanto mais próximo da superfície (Figuras 3.3c e 3.3d).

Por outro, a imagem migrada filtrada com o modelo U-Net no domínio curvelet apresenta também uma melhora na resolução dos refletores, porém com um aumento do ruído da imagem, como pode ser observado na Figura 3.4b, gerando artefatos em toda a imagem. Esse mesmo efeito pode ser observado com zoom na Figura 3.5, onde a continuidade dos refletores foi poluída com ruídos. Outras combinações de número de ângulos e escalas foram testadas, apesar de não exibidas neste trabalho, e observou-se que ao aumentar o número de ângulos na escala $s = 1$, a quantidade de ruído somente aumenta. Todavia, ao aumentar o número de escalas, o ruído diminuiu até um certo limite, porém ao custo de um aumento considerável de consumo de memória RAM do computador. Dito isso, este filtro foi capaz de melhorar pelo menos um pouco a continuidade de refletores mal iluminados.

De um modo geral, os filtros cumpriram o papel de melhorar a iluminação. Porém, no caso do modelo Marmousi, o modelo U-Net no domínio curvelet não foi capaz de desempenhar à altura do seu concorrente no domínio do espaço devido a inserção indesejada de ruído.

3.2 Sigsbee2B

O modelo Sigsbee representa uma domo de sal localizada no mar. O objetivo é representar melhor os refletores diretamente abaixo dela, uma vez que há pouca iluminação nesta região. O dado associado ao modelo utilizado tem espaçamentos $dx = 25,4$ m e $dz = 15,24$ m, dimensões de $nx = 960$ e $nz = 600$. O conjunto de tiros utilizados para a migração RTM foram criados empregando a modelagem Born a partir da refletividade do modelo. Foram criados 200 tiros, com 160 receptores para cada lado do tiro numa geometria de aquisição *split-spread*. Esses tiros foram migrados e então remigrados, aplicados o filtro laplaciano, servindo então de input para os métodos de filtragem discutidos previamente.

Como na última subseção, seguem os parâmetros do experimento. Para o modelo U-Net, foi usados 1000 patches, com patches de tamanho 64×64 , com uma passada de 20 em ambas

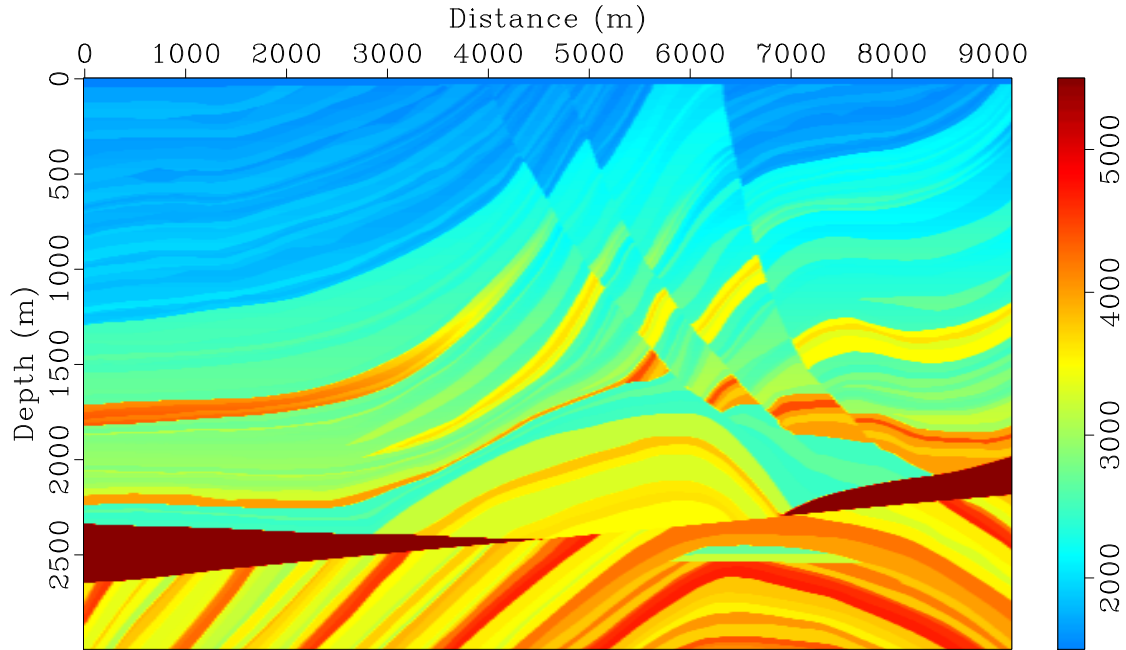


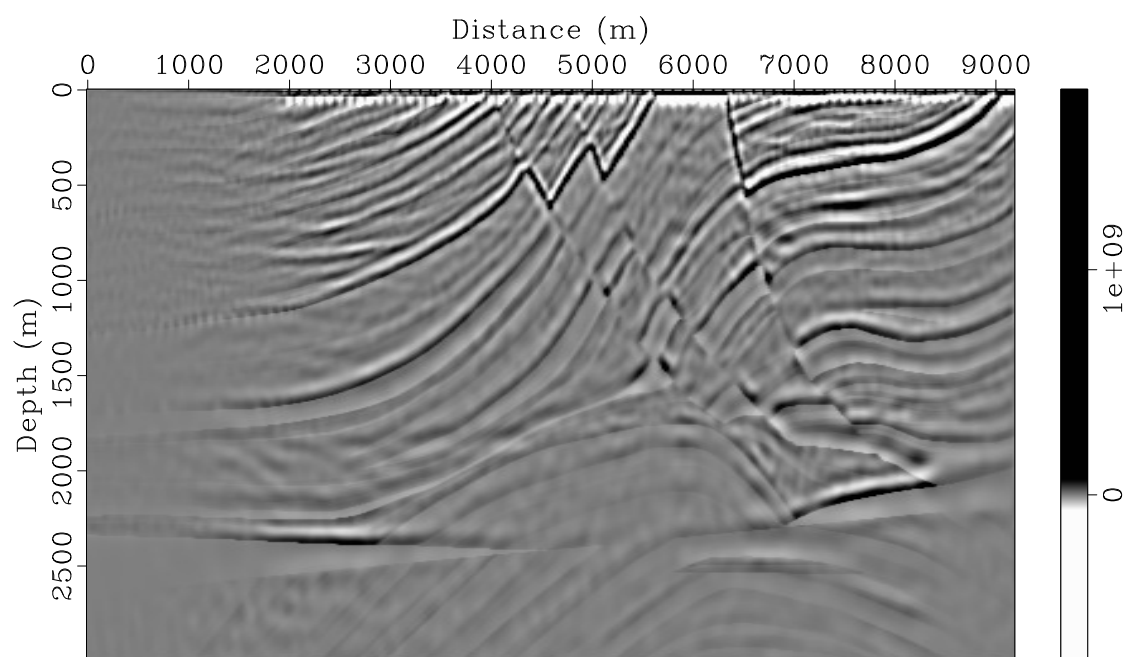
Figura 3.1: Campo de velocidade do modelo Marmousi, utilizado para a modelagem e migração.

as direções. A validação por divisão de dados de treinamento também foi 0,2, com a mesma taxa de aprendizado $lr = 0,0001$. Para o modelo U-Net do domínio curvelet, a taxa de aprendizagem foi $lr = 0,01$, o número de escalas de transformada curvelet foi $nbs = 4$ e o número de fatias angulares na escala mais grosseira foi $nangles = 4$. A imagem migrada foi criada a partir de 200 tiros igualmente espaçados, distribuídos ao longo do centro do modelo para atingir a domo de sal.

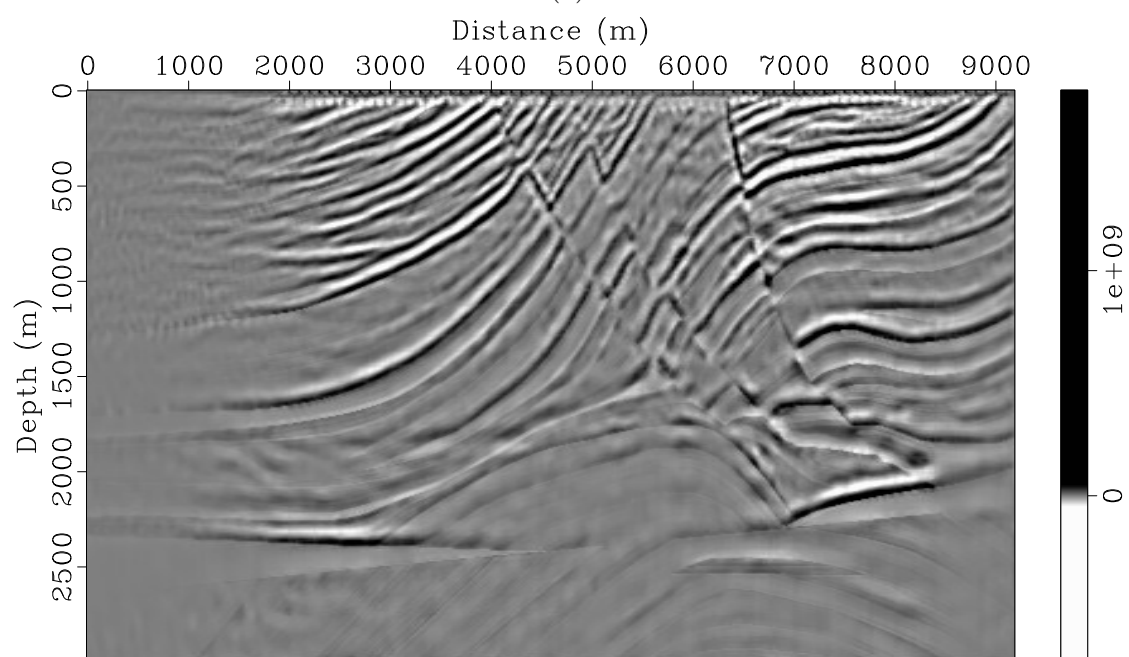
Em contraste com o modelo Marmousi, o Sigsbee respondeu melhor à ambos os filtros, como pode ser visto nas Figuras 3.7 e 3.9.

O filtro baseado no modelo U-Net no domínio espacial, assim como no caso do modelo Marmousi, apresenta uma melhor resolução quando comparada a imagem migrada tradicionalmente. No entanto, isso somente pode ser observado a olho nu nos refletores mais superficiais, como mostrado na Figura 3.7b. Ao se dar um zoom na região logo abaixo do domo de sal (Figura 3.8b), é possível observar uma melhora na amplitude e continuidade dos refletores, ao custo de artefatos da sobreposição dos janelamentos espaciais. Estes janelamentos espaciais são, no entanto imperceptíveis, devido a regularização com norma L1 das últimas camadas.

Já o filtro baseado no modelo U-Net no domínio curvelet, exibido na Figura 3.9b, apresenta um resultado muito melhor que a imagem migrada RTM, em contraste com sua performance no modelo Marmousi. A imagem ainda possui alguns artefatos em pontos difratores



(a)



(b)

Figura 3.2: (a) Representa a imagem migrada RTM a partir do dado observado no modelo Marmousi. (b) Resultado da aplicação da rede neural U-Net já treinada à imagem migrada. Ambas figuras são plotadas na mesma escala de amplitude para melhor comparação.

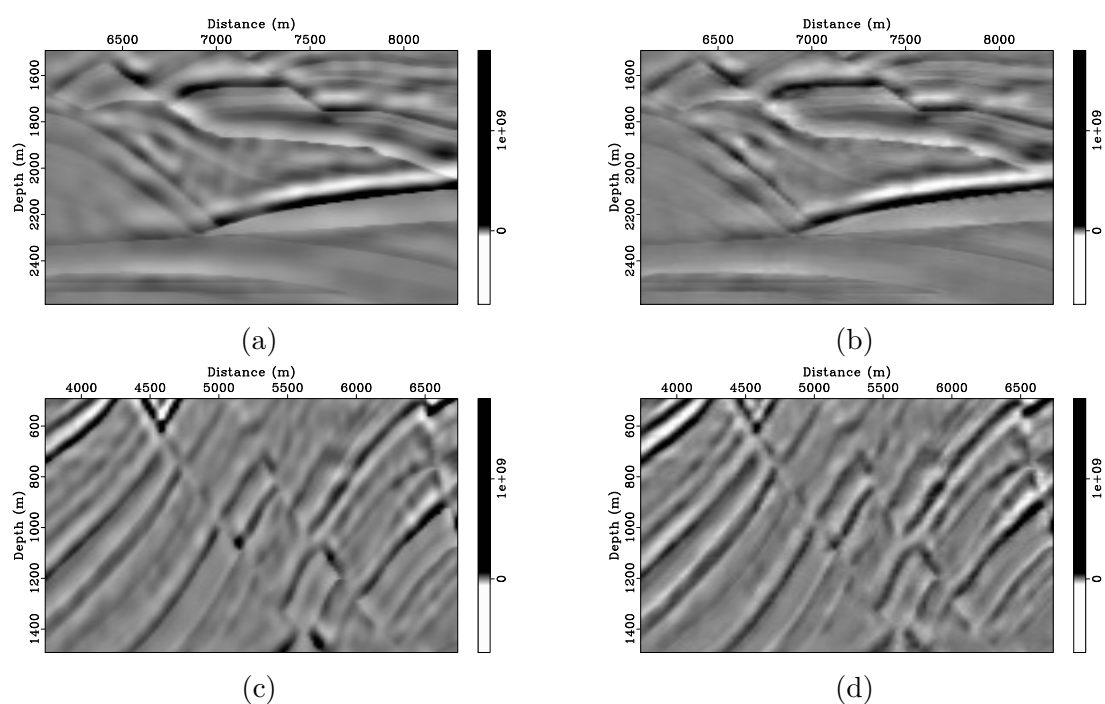
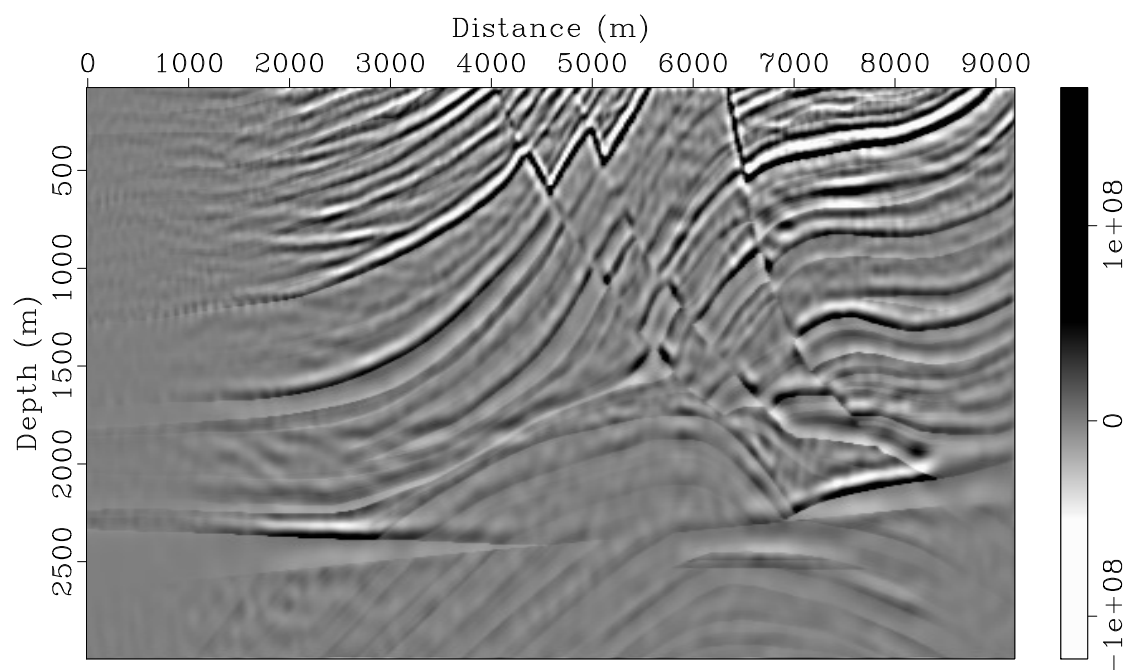
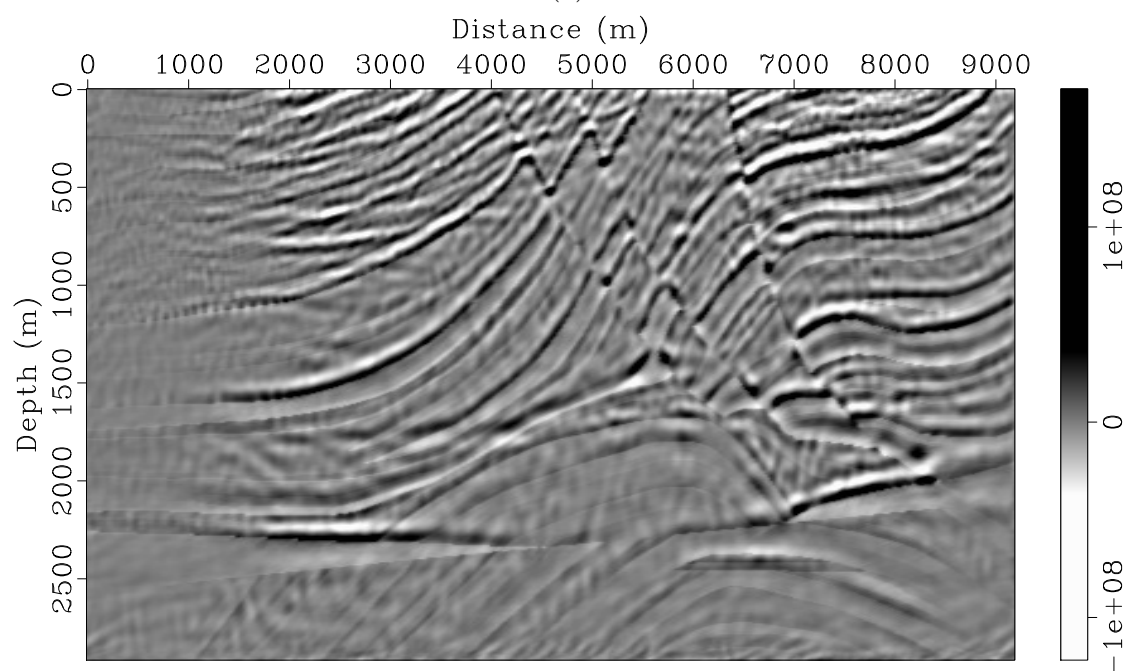


Figura 3.3: (a) Janela quadrada extraída a partir da imagem original, em profundidade, migrada a partir do dado observado no modelo Marmousi. (b) Janela quadrada extraída a partir da imagem filtrada com a rede U-Net na mesma posição. (c) Janela quadrada extraída a partir da imagem original, em meio as falhas no centro do modelo, migrada a partir do dado observado. (d) Janela quadrada extraída a partir da imagem filtrada com a rede U-Net na mesma posição.



(a)



(b)

Figura 3.4: (a) Representa a imagem migrada RTM a partir do dado observado no modelo Marmousi. (b) Resultado da aplicação da rede neural U-Net no domínio curvelet já treinada à imagem migrada. Ambas figuras são plotadas na mesma escala de amplitude para melhor comparação.

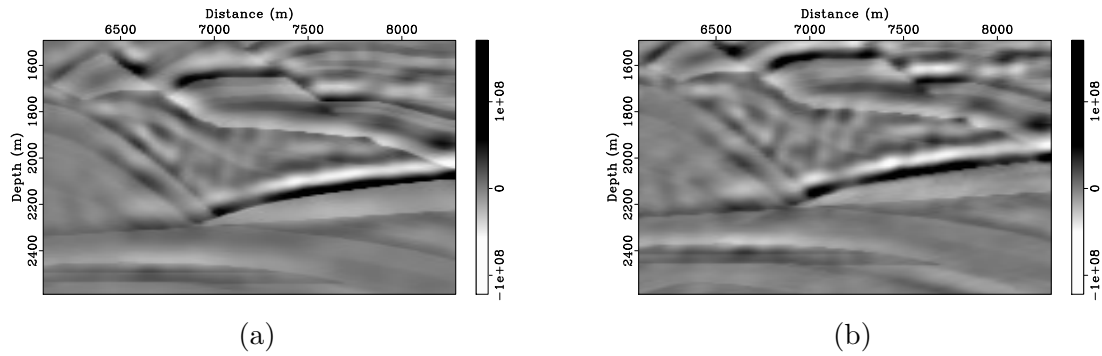


Figura 3.5: (a) Janela quadrada extraída a partir da imagem migrada RTM, em profundidade, a partir do dado observado no modelo Marmousi. (b) Janela quadrada extraída a partir da imagem filtrada com a rede U-Net no domínio curvelet na mesma posição.

isolados, como os posicionados a 7600 m de profundidade. No entanto, o resultado da filtragem gerou uma imagem que chega muito próximo da refletividade original, objetivo do método. Ao se observar na Figura 3.10b a região logo abaixo do domo novamente, é notável o realçamento de amplitudes sem custo de artefatos, somado a um considerável aumento de continuidade dos refletores à esquerda que sequer são visíveis na Figura 3.10a.

Mediante ao exposto, ambos os filtros apresentaram bons resultados no modelo Sigsbee2B. Ao contrário do caso do modelo Marmousi, o filtro no domínio curvelet ofereceu melhor balanceamento de iluminação, e sem injeção de ruídos indesejados.

Os dados de entrada (imagem migrada e remigrada) foram gerados utilizando o supercomputador do Centro de Pesquisa em Geofísica e Geologia (CPGG), mais especificamente a máquina chamada Marreca. As suas configurações de hardware na data de escrita consistem em aproximadamente 500 Gb de memória RAM, um processador Xeon de 24 threads, e uma placa de vídeo da NVIDIA modelo Tesla K40 de 12 Gb de VRAM. O seu sistema operacional é o CentOS 6. Já o treinamento da rede quanto a aplicação delas foi feito na minha máquina local, para utilizar de um sistema linux mais recente, e garantir compatibilidade com todas as dependencias utilizadas. O sistema operacional utilizado foi o Arch Linux, com configurações de hardware na data de escrita consistindo de 16 Gb de memória RAM, um processador Xeon com 20 threads e placa de vídeo da AMD modelo RX 580 com 8 Gb de VRAM.

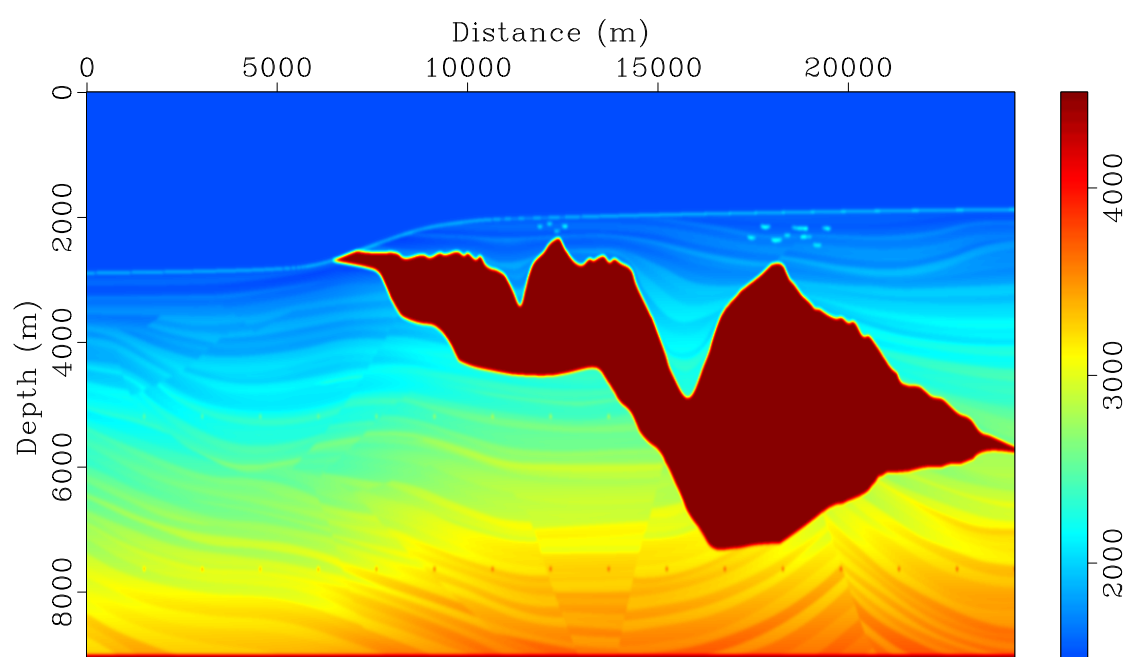


Figura 3.6: Campo de velocidade do modelo Sigbee 2B, utilizado para a modelagem e migração.

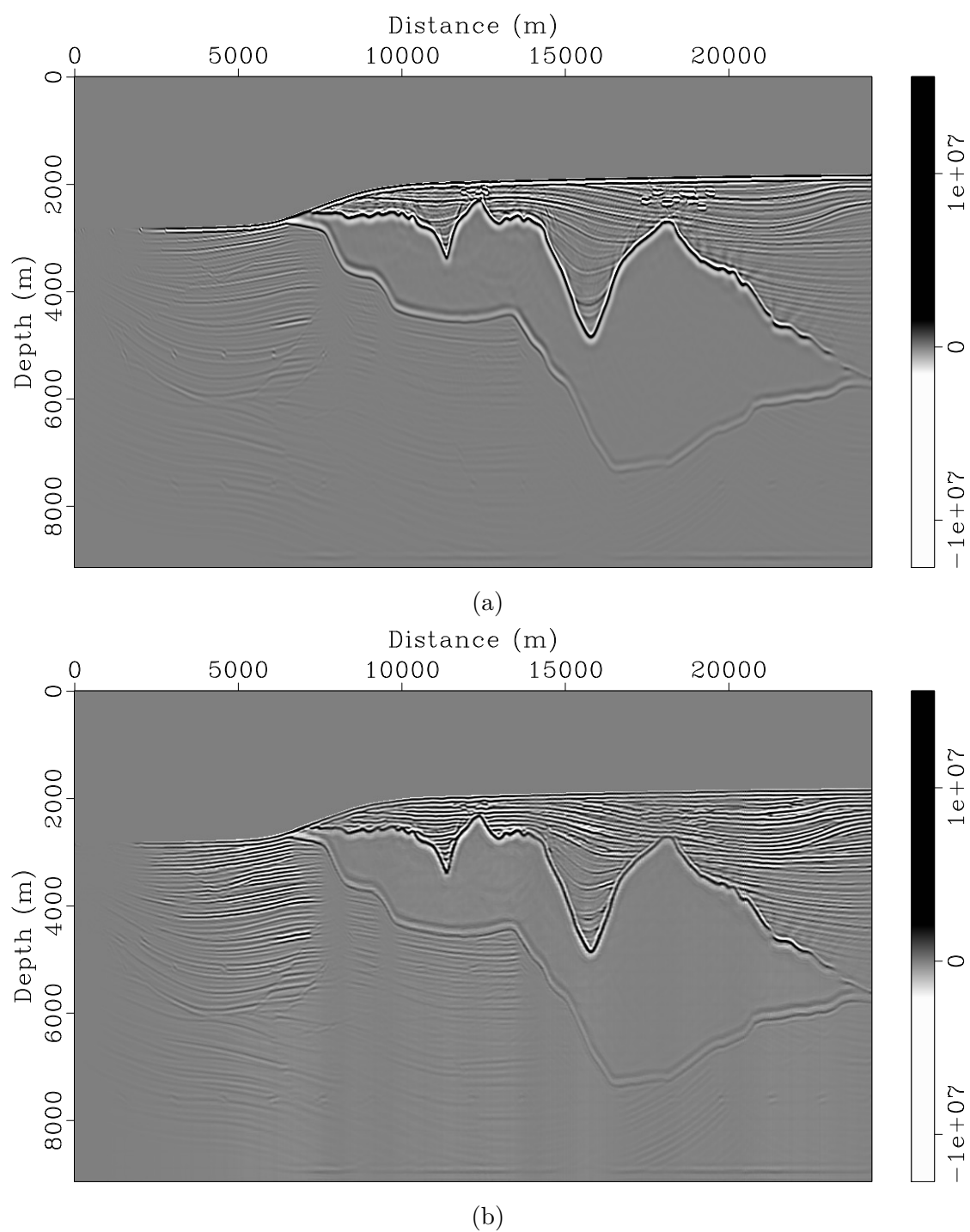


Figura 3.7: (a) Representa a imagem migrada RTM a partir do dado observado no modelo Sigsbee 2B. (b) Resultado da aplicação da rede neural U-Net já treinada à imagem migrada. Ambas figuras são plotadas na mesma escala de amplitude para melhor comparação.

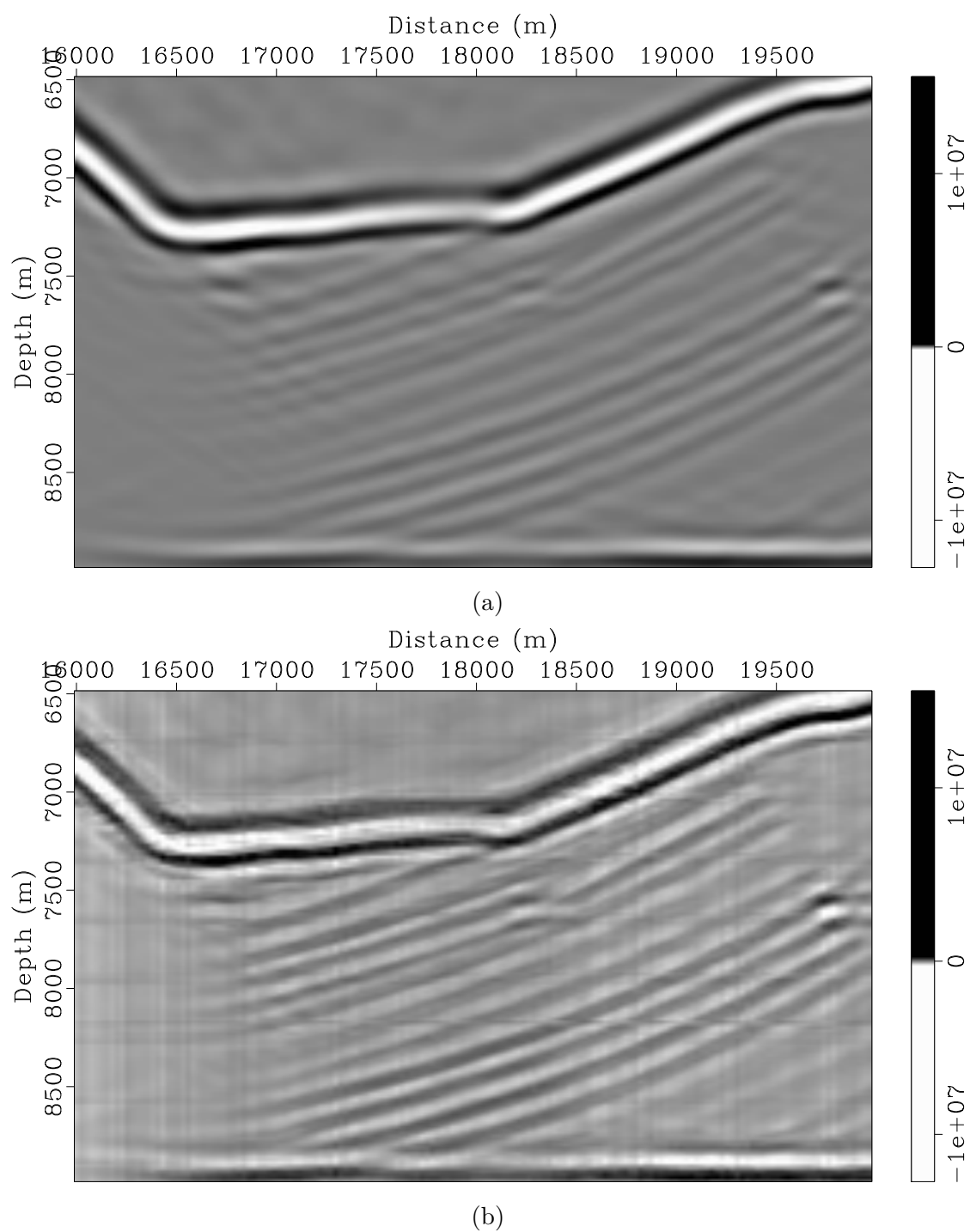


Figura 3.8: (a) Janela quadrada extraída a partir da imagem original, logo abaixo do domo de sal, migrada a partir do dado observado no modelo Sigsbee 2B. (b) Janela quadrada extraída a partir da imagem filtrada com a rede U-Net na mesma posição.

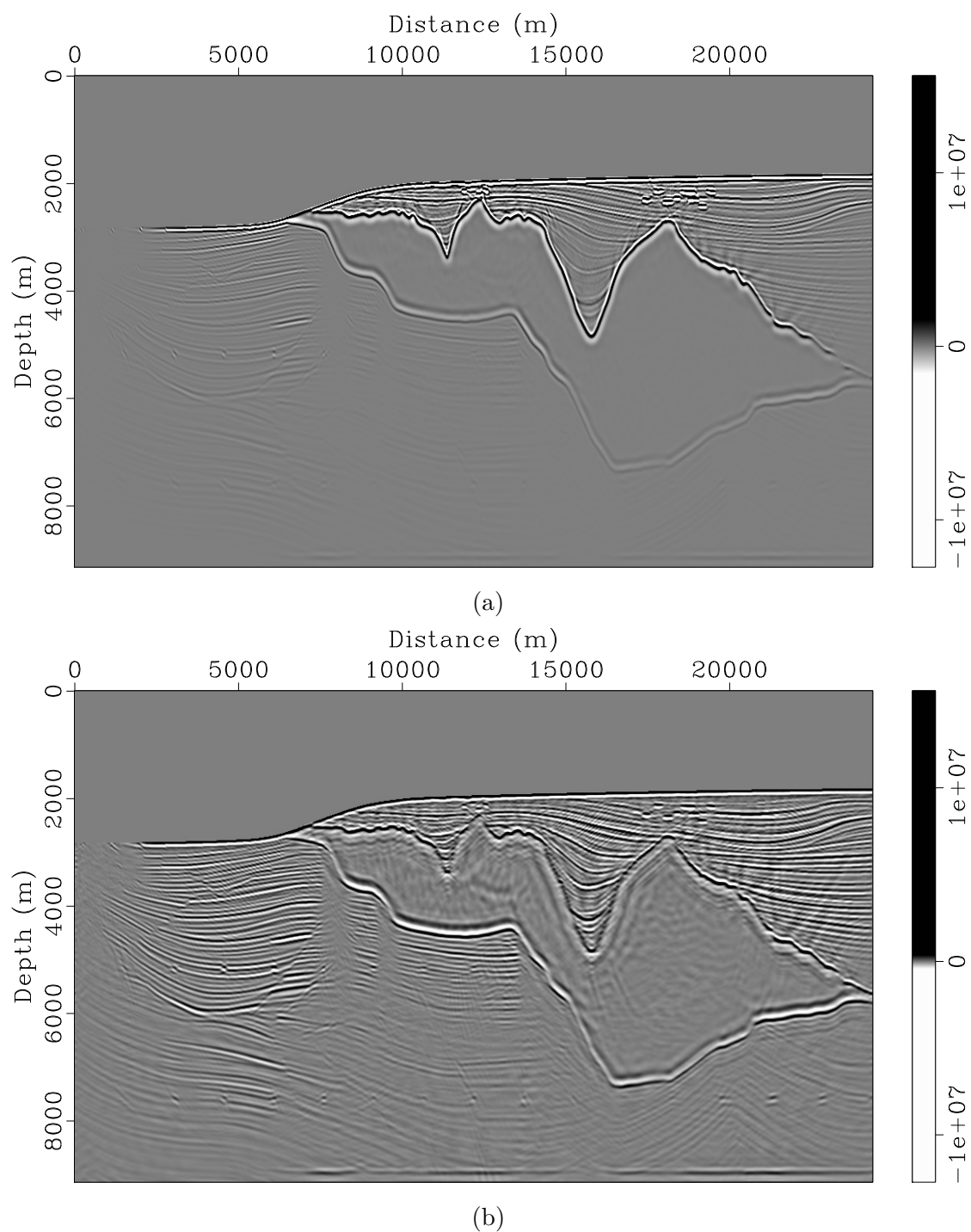
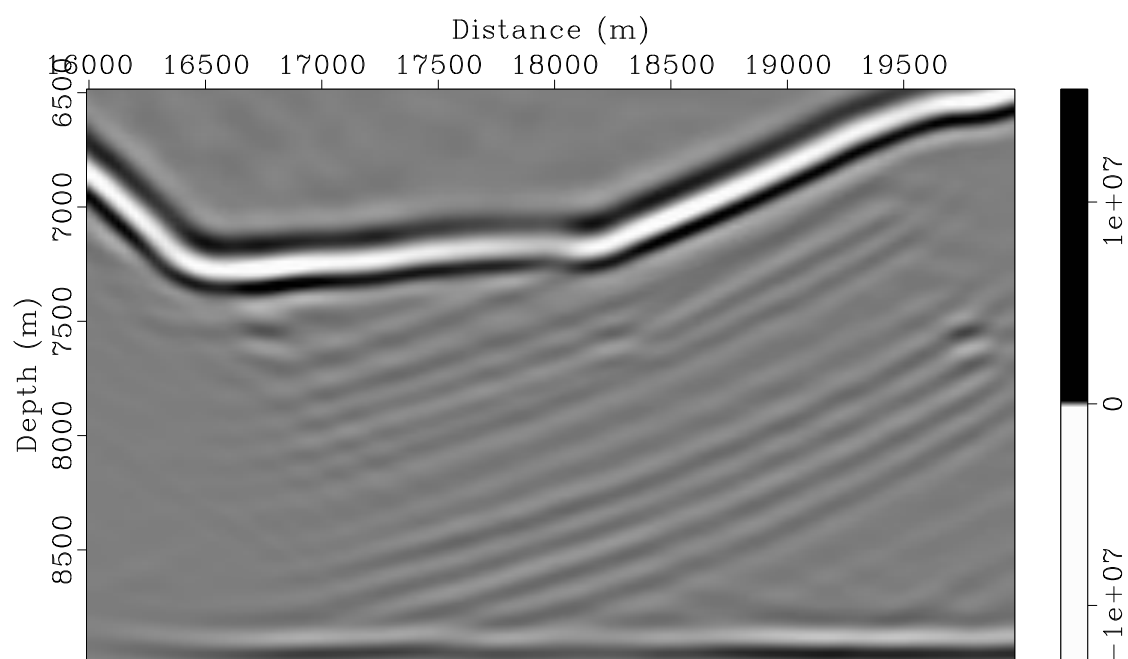
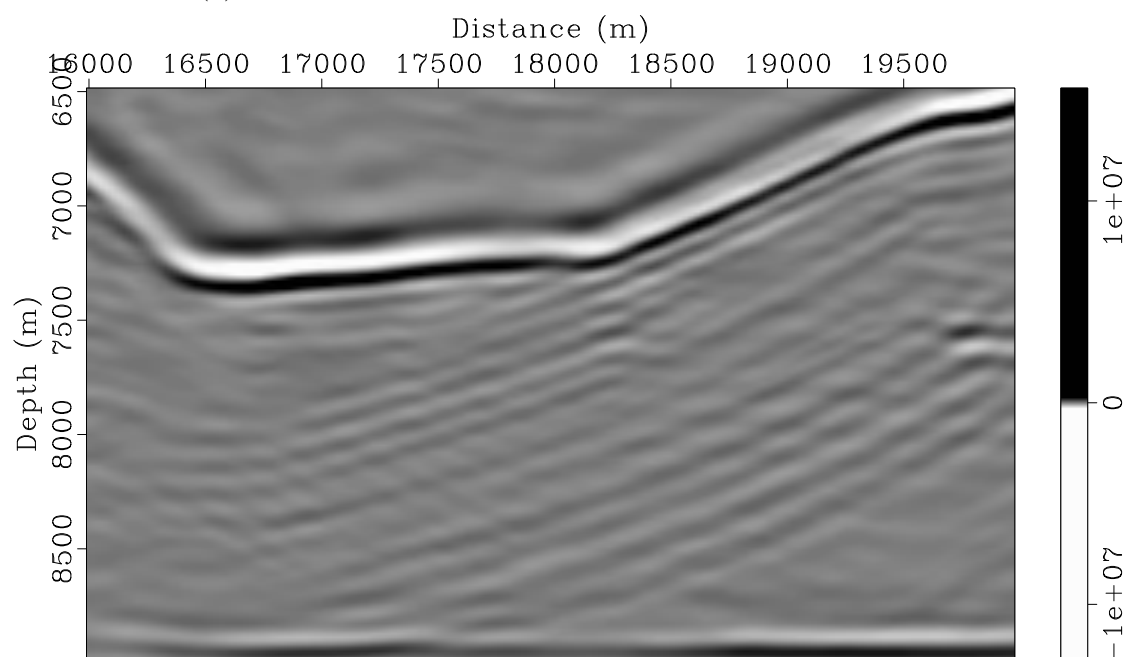


Figura 3.9: (a) Representa a imagem migrada RTM a partir do dado observado no modelo Sigsbee 2B. (b) Resultado da aplicação da rede neural U-Net no domínio curvelet já treinada à imagem migrada. Ambas figuras são plotadas na mesma escala de amplitude para melhor comparação.



(a) Janela quadrada extraída a partir da imagem migrada



(b) Janela quadrada extraída a partir da imagem filtrada

Figura 3.10: (a) Janela quadrada extraída a partir da imagem migrada RTM a partir do dado observado no modelo Sigsbee, logo abaixo do domo de sal. (b) Janela quadrada extraída a partir da imagem filtrada com a rede U-Net no domínio curvelet na mesma posição.

4

Conclusões

Em resumo, apresentamos uma comparação entre dois filtros baseados em redes neurais que visam alcançar um efeito de focalização quando aplicados à imagem migrada, um baseado no trabalho de Avila et al. (2021), e o outro proposto por nós como uma adaptação deste último ao um cenário de domínio de transformada curvelet. Ambos apresentaram resultados interessantes, claramente desempenhando bem no modelo Sigsbee2B, e não tão bem no modelo Marmousi.

Por um lado, o filtro U-Net no domínio espacial ofereceu uma performance mais robusta, demonstrando resultados similares tanto no modelo Marmousi quanto no Sigsbee2B. No entanto, o mesmo não foi muito capaz de melhorar a continuidade de refletores com pouquíssima iluminação, como os da lateral esquerda da Figura 3.2a, ou os logo abaixo do domo de sal do modelo Sigsbee2B na parte à esquerda na Figura 3.8a.

Por outro lado, o filtro no domínio curvelet ofereceu resultados ambíguos, sendo muito efetivo no modelo Sigsbee2B, ao custo de introduzir ruído no modelo Marmousi. Apesar de seus resultados não serem tão robustos quanto à sua contraparte no domínio do espaço, ele demonstrou ser capaz de além de melhorar a iluminação dos refletores em profundidade, de também melhorar a continuidade dos mesmos em regiões de baixa iluminação, como pode ser visto na Figura 3.10b.

Ambos filtros demonstraram vantagens e desvantagens a se considerar. Uma possível razão para a robustez do filtro no domínio espacial se dá pela capacidade do filtro de tomar decisões informadas sobre como distribuir as amplitudes a partir de uma região macro do tamanho da janela. Ou seja, o filtro tem acesso direto as feições espaciais do dado, sem depender de como elas são bem ou mal representadas num outro domínio reparametrizado

como o curvelet. No entanto, isso também significa que devido ao seu objetivo de somente reduzir o valor da função custo, talvez regiões que não influenciam tanto neste valor tenham suas amplitudes imutáveis. Essa seria uma hipótese que justifica a falta de continuidade dos refletores com o filtro no domínio espacial. No entanto, o seu filtro concorrente opera no domínio curvelet, o qual tem coeficientes altos para refletores com um certo nível de continuidade, independente da amplitude ou iluminação. Talvez essa seja a razão pela qual este filtro demonstrou bons resultado nesse ponto. O aumento de ruído no caso do Marmousi pode ser causado a uma intolerância do filtro com relação a artefatos e ruídos de fundo, visto que o modelo Sigsbee2B apresentou uma imagem migrada limpa de ruídos e por sua vez obteve bons resultados. De todo modo todas estas colocações são somente hipóteses que necessitam de estudos posteriores para confirmá-las. Uma desvantagem clara porém do filtro no domínio curvelet é seu consumo exagerado de memória RAM, visto que o número de fatias angulares quase que duplica a cada escala.

Uma possível razão para tal intolerância à ruído seja porque o filtro no domínio curvelet treina redes individuais para cada uma de suas feições curvelet, tornando-o incapaz de tomar decisões informadas com base na transformada inversa da saída. Isso faz com que diferentes cunhas na mesma escala introduzam artefatos de maneira construtiva no mesmo local nos cantos de refletores locais, uma vez que têm características de muitos ângulos diferentes. Esta foi a principal razão pela qual usamos um baixo número de ângulos em ambos os modelos. Se houvesse uma implementação da transformada compatível com a implementação do gradiente Tensorflow (ou em outras plataformas de *deep learning*), seria possível calcular a função de custo no domínio espacial aplicando a transformada curvelet inversa à saída da rede, ainda sendo capaz de realizar diferenciação automática. Inclusive, o trabalho de Sanavi, Moghaddam e Herrmann (2021), como mencionado na introdução, também executa um filtro de domínio curvelet, com uma função de custo no domínio espacial, e em vez de usar uma rede neural, ele usou um algoritmo de otimização da família quasi-Newton. Uma maneira que encontramos de chegar a um resultado semelhante e lidar com o ruído foi forçar a regularização L_1 , uma vez que o domínio curvelet é extremamente esparsa, o que por sua vez implica que a saída da rede neural também deve ser esparsa. Esta pequena mudança reduziu bastante o ruído do resultado final. De todo modo, se utilizada uma função de custo de domínio espacial, resultados muito melhores poderiam ser obtidos. Mesmo assim, treinar e aplicar o modelo U-Net no domínio curvelet conseguiu entregar resultados incríveis no modelo Sigsbee, com as devidas adaptações.

Vários parâmetros experimentais foram testados e os selecionados foram, por tentativa e erro, os melhores. Para o caso da rede U-Net no domínio do espaço, o tamanho das janelas em particular não importou muito ao aumentá-las após um certo limite. Já para o caso da

rede U-Net no domínio curvelet, aumentar o número de escalas a partir de um certo ponto não apresentou nenhuma melhora significativa, já aumentar o número de fatias angulares na primeira escala introduz uma quantidade considerável de ruído de baixa frequência na imagem. A metodologia empregada se concentrou em tentar testar os limites dos filtros de focalização, e não teve como objetivo fazer uma análise aprofundada dos parâmetros ótimos para cada método. À medida que mais trabalhos exploram o conceito de uso de um domínio diferente para treinar redes neurais, uma avaliação mais focada desses parâmetros faria muito mais sentido. De todo modo, é importante notar que a metodologia empregada não foi conceituada para garantir a amplitude de refletores verdadeira. A iluminação balanceada da aplicação do filtro pode melhorar visualmente a interpretação das características geológicas, mas mais pesquisa se faz necessária se alguém deseja aplicá-las a um estudo de amplitude versus afastamento (AVO).

Este trabalho tem como objetivo iniciar uma discussão sobre o uso de diferentes domínios para a entrada da rede neural em imagens sísmicas e áreas correlatas. O progresso feito aqui é apenas um passo na direção certa. Esses domínios não precisam necessariamente estar relacionados aos descendentes da transformada de Fourier, mas podem ser representados por outra rede neural feita para reparametrização. Resolver o problema de calcular a função de custo no domínio espacial é o mais óbvio, mas há muito mais melhorias que podem ser feitas. Primeiro, ao transformar os dados em um domínio mais esparsos, a menos que haja uma implementação de array esparsos para fazer o backup, ele consumirá apenas mais memória RAM. Uma melhor implementação da transformada de curvelet que leva isso em conta dado um determinado *threshold* para os coeficientes, melhoraria o desempenho do treinamento em geral. Em segundo lugar, testar abordagens semelhantes com dados reais lançaria uma luz sobre quais tipos de características geológicas seriam mais bem iluminadas. Em terceiro lugar, usando técnicas de migração de última geração, com condições de imagem mais sofisticadas (e.g. Moradpouri et al., 2017) e extrapolação de onda elástica. Em quarto, incorporar alguma rede neural informada por física (Raissi, Perdikaris e Karniadakis, 2019) para tornar o modelo robusto para muitos dados diferentes, o que resolveria a limitação de ter que treinar novamente a rede em todas as ocasiões. Por último, mas não menos importante, é de suma importância testar os filtros discutidos num esquema de LSRTM iterativo tradicional para atestar sua efetividade em diminuir o número de iterações até a convergência.

Agradecimentos

Esse trabalho não seria possível sozinho, de forma alguma. Em vários momentos eu me vi completamente cercado de obstáculos dos mais diversos tipos, relacionados direta ou indiretamente com o mestrado. Eu não consigo me imaginar superando eles sem ajuda das pessoas que mencionarei ao longo desse pequeno texto.

Primeiro, a pessoa que mais me ajudou sem sombra de dúvidas foi meu orientador Reynam Pestana por alguns motivos óbvios e outros não tanto. Eu cheguei a pós graduação em geofísica mudando de área, saindo da Petrofísica para o Imageamento sísmico, porque eu sonhava em trabalhar com computação de alta performance. Porém me faltava muito da bagagem necessária para atingir tal objetivo, mas Reynam me ensinou ou pelo menos me direcionou sobre os diversos assuntos que eu precisava dominar. Não obstante, ele demonstrou paciência e sobriedade enquanto eu lidava com vários acontecimentos na minha vida pessoal que querendo ou não acabaram se tornando obstáculos. Por esses motivos eu sou extremamente grato.

Não menos importante, eu agradeço enormemente a minha família que esteve ao meu lado em todos os momentos. Meu irmão, João Eduardo, meu pai, Clairton Quintela, e minha mãe Valéria Maria e meus gatos Dino, Thomas e Nina. Isso sem mencionar meus tios, tias e primos. Todos eles estiveram ao meu lado nos choros, sorrisos, tristezas e alegrias.

Também gostaria de agradecer enormemente aos meu colegas e amigos da Pós-graduação. Daniel Ravelo, que me ensinou literalmente o básico do básico de imageamento sísmico e se dispôs a tirar todas as minhas dúvidas, sem me julgar por quão bobas ou repetitivas elas eram. Agradeço também Betina, Quézia, Cristian, Marcelo Santana, Leonardo "Barril", Laila, Elienara, Laian, Adevilson e Anderson Silva, pelos muitos momentos de troca de ideias, explicações, conversas, festas, sorrisos, brincadeiras e piadas.

Preciso mencionar também Marcos Conceição, um amigo querido meu que também é meu orientando em seu trabalho de graduação. Eu, Marcos e Reynam formamos um mini grupo de pesquisa de forma a trocar figurinhas sobre os nossos avanços pessoais. Sem as ideias de Marcos, eu não sei como teria terminado esse mestrado.

Falando em amigos, gostaria de expressar minha gratidão a Felipe Duarte e sua namorada Allana, por estarem sempre preocupados comigo, e me apoiando das mais diversas formas durante todo o mestrado. Felipe em especial é meu amigo desde o ensino fundamental, e é pilar essencial do meu caráter hoje.

Gostaria de agradecer também aos meus amigos companheiros de RU (restaurante universitário), que me deram sorrisos diários enquanto trocávamos ideias almoçando juntos. Sabrina Sacramento que se tornou minha namorada e muito me apoiou durante essa jornada. Josafat, que sem suas piadas e histórias, meus almoços não seriam os mesmos. Rafael Paulo, com seu jeito engraçado de viver. Maria Gontijo que é minha amiga desde o ensino fundamental, e é meu ponto de apoio em todos os momentos. Iago Pinto e Gabriela Mylena, amigos queridos meu desde o ensino médio, companheiros de muitas fases da minha vida. Daniel Mascarenhas pela companhia tanto na sala da pós, nos almoços, nas aulas, nas dicas, e nas trocas de ideias.

Eu também quero agradecer a meus outros amigos da geofísica que não necessariamente iam para o RU comigo, mas sempre estavam ao meu lado. Estes seriam David Lenon, Rafael Inácio, Ana Paim e João Felipe. Estes são meus amigos que vivem comigo desde a graduação e tem me apoiado até hoje. David Lenon com seu companheirismo, dando sensação de que tudo vai dar certo. Rafael Inácio com sua sinceridade super engraçada e compreensão. Ana pelo seu carinho e presença nas fofoca. E João Felipe no companheirismo também, nos conselhos, ensinamentos e piadas, sendo a voz da razão durante toda a minha graduação e pós. Amâncio e Adilson pelas suas histórias engraçadas.

Gostaria de agradecer a banca, aos professores, ao CNPq, e todo o Centro de Pesquisa em Geofísica e Geologia. A banca por avaliar meu trabalho e dispor o tempo livre deles para tal propósito. O CNPq pela bolsa de pesquisa que garantiu meu sustento durante o seu período de vigência. E o CPGG por dispor das suas dependências, do seu supercomputador, e de toda a sua estrutura para me auxiliar nessa jornada.

Por último mas não menos importante gostaria de agradecer aos meus “conselheiros externos”, que me guiaram em áreas de conhecimento que eu acabei me especializando no mestrado. O primeiro seria Victor Koehne, que foi o que mais me incentivou na área de computação de alta performance. O segundo seria Alex Cerqueira, que me incentivou na área de inteligência artificial e redes neurais.

5

Referências Bibliográficas

- Aoki, Naoshi, e Gerard T. Schuster. 2009. “Fast least-squares migration with a deblurring filter”. *Geophysics* 74, n. 6 (novembro): WCA83–WCA93. <https://doi.org/10.1190/1.3155162>.
- Avila, Manuel Ramón Vargas, Luana Nobre Osorio, Júlio de Castro Vargas Fernandes, André Bulcão, Bruno Pereira-Dias, Bruno de Souza Silva, Pablo Machado Barros, Luiz Landau e Alexandre G. Evsukoff. 2021. “Migration Deconvolution via Deep Learning”. *Pure and Applied Geophysics* 178 (5): 1677–1695. ISSN: 1420-9136. <https://doi.org/10.1007/s00024-021-02707-0>.
- Baysal, Edip, Dan D. Kosloff e John W. C. Sherwood. 1983. “Reverse time migration”. *Geophysics* 48, n. 11 (novembro): 1514–1524. <https://doi.org/10.1190/1.1441434>.
- Candes, Emmanuel J, e David L Donoho. 2000. *Curvelets: A surprisingly effective nonadaptive representation for objects with edges*. Relatório técnico. Stanford Univ Ca Dept of Statistics.
- Candès, Emmanuel, Laurent Demanet, David Donoho e Lexing Ying. 2006. “Fast Discrete Curvelet Transforms”. *Multiscale Modeling & Simulation* 5, n. 3 (janeiro): 861–899. <https://doi.org/10.1137/05064182x>.
- Candès, Emmanuel J., e David L. Donoho. 2005a. “Continuous curvelet transform: I. Resolution of the wavefront set”. *Applied and Computational Harmonic Analysis* 19, n. 2 (setembro): 162–197. <https://doi.org/10.1016/j.acha.2005.02.003>.
- . 2005b. “Continuous curvelet transform: II. Discretization and frames”. *Applied and Computational Harmonic Analysis* 19, n. 2 (setembro): 198–222. <https://doi.org/10.1016/j.acha.2005.02.004>.
- Chang, Wen-Fong, e George A. McMechan. 1986. “Reverse-time migration of offset vertical seismic profiling data using the excitation-time imaging condition”. *Geophysics* 51, n. 1 (janeiro): 67–84. <https://doi.org/10.1190/1.1442041>.
- Chollet, Francois, et al. 2015. “Keras”. <https://github.com/fchollet/keras>.

- Claerbout, Jon F. 1992. *Earth soundings analysis: Processing versus inversion*. Vol. 6. Blackwell Scientific Publications London.
- Costa, Carlos da. 2020. “Curvelops”. <https://github.com/PyLops/curvelops>.
- Cun, Y. Le, L.D. Jackel, B. Boser, J.S. Denker, H.P. Graf, I. Guyon, D. Henderson, R.E. Howard e W. Hubbard. 1989. “Handwritten digit recognition: applications of neural network chips and automatic learning”. *IEEE Communications Magazine* 27, n. 11 (novembro): 41–46. <https://doi.org/10.1109/35.41400>.
- Dai, Wei, e Gerard T. Schuster. 2013. “Plane-wave least-squares reverse-time migration”. *Geophysics* 78, n. 4 (julho): S165–S177. <https://doi.org/10.1190/geo2012-0377.1>.
- Dutta, Gaurav, Cyril Agut, Matteo Giboli e Paul Williamson. 2016. “Least-squares reverse time migration with radon preconditioning”. Em *SEG Technical Program Expanded Abstracts 2016*. Society of Exploration Geophysicists, setembro. <https://doi.org/10.1190/segam2016-13943593.1>.
- Dutta, Gaurav, Matteo Giboli, Cyril Agut, Paul Williamson e Gerard T. Schuster. 2017. “Least-squares reverse time migration with local Radon-based preconditioning”. *GEOPHYSICS* 82, n. 2 (março): S75–S84. <https://doi.org/10.1190/geo2016-0117.1>.
- Dutta, Gaurav, Matteo Giboli, Paul Williamson e Gerard T. Schuster. 2015. “Least-squares reverse time migration with factorization-free priorconditioning”. Em *SEG Technical Program Expanded Abstracts 2015*. Society of Exploration Geophysicists, agosto. <https://doi.org/10.1190/segam2015-5912138.1>.
- Fomel, Sergey, Paul Sava, Ioan Vlad, Yang Liu e Vladimir Bashkardin. 2013. “Madagascar: Open-source software project for multidimensional data analysis and reproducible computational experiments”. *Journal of Open Research Software* 1 (1). <https://doi.org/10.5334/jors.ag>.
- Goodfellow, Ian, Yoshua Bengio e Aaron Courville. 2016. *Deep Learning*. [Http://www.deeplearningbook.org](http://www.deeplearningbook.org). MIT Press.
- Guitton, Antoine. 2004. “Amplitude and kinematic corrections of migrated images for non-unitary imaging operators”. *Geophysics* 69, n. 4 (julho): 1017–1024. <https://doi.org/10.1190/1.1778244>.
- Gupta, Bhawna, e Shamik Tiwari. 2014. “Lung cancer detection using curvelet transform and neural network”. *International Journal of Computer Applications* 86 (1).
- Hornik, Kurt, Maxwell Stinchcombe e Halbert White. 1990. “Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks”. *Neural Networks* 3 (5): 551–560. ISSN: 0893-6080. [https://doi.org/10.1016/0893-6080\(90\)90005-6](https://doi.org/10.1016/0893-6080(90)90005-6). <https://www.sciencedirect.com/science/article/pii/0893608090900056>.
- Hu, Jianxing, Gerard T. Schuster e Paul A. Valasek. 2001. “Poststack migration deconvolution”. *Geophysics* 66, n. 3 (maio): 939–952. <https://doi.org/10.1190/1.1444984>.
- Jadon, Shruti. 2018. “Introduction to different activation functions for deep learning”. *Medium, Augmenting Humanity* 16:140.

- Kingma, Diederik P., e Jimmy Ba. 2014. *Adam: A Method for Stochastic Optimization*. <https://doi.org/10.48550/ARXIV.1412.6980>.
- Levin, Stewart A. 1984. “Principle of reverse-time migration”. *Geophysics* 49, n. 5 (maio): 581–583. <https://doi.org/10.1190/1.1441693>.
- Liu, Mengli, Jianping Huang, Zhenchun Li, Chuang Li e Yingjun Ren. 2017. “Preconditioned Least-squares reverse-time migration with well-constrain regularization”. Em *International Geophysical Conference, Qingdao, China, 17-20 April 2017*. Society of Exploration Geophysicists / Chinese Petroleum Society, maio. <https://doi.org/10.1190/igc2017-262>.
- Liu, Qiancheng, Yongming Lu, Hui Sun e Hao Zhang. 2019. “Single-Step Data-Domain Least-Squares Reverse-Time Migration Using Gabor Deconvolution for Subsalt Imaging”. *IEEE Geoscience and Remote Sensing Letters*, 1–4. <https://doi.org/10.1109/lgrs.2019.2916847>.
- Liu, Qiancheng, e Daniel Peter. 2018. “One-step data-domain least-squares reverse time migration”. *Geophysics* 83, n. 4 (julho): R361–R368. <https://doi.org/10.1190/geo2017-0622.1>.
- Liu, Zhaolun, Yuqing Chen e Gerard Schuster. 2020. “Deep convolutional neural network and sparse least-squares migration”. *GEOPHYSICS* 85, n. 4 (junho): WA241–WA253. <https://doi.org/10.1190/geo2019-0412.1>.
- Loewenthal, D., L. Lu, R. Roberson e J. Sherwood. 1976. “The Wave equation applied to migration”. *Geophysical Prospecting* 24, n. 2 (junho): 380–399. <https://doi.org/10.1111/j.1365-2478.1976.tb00934.x>.
- Loewenthal, Dan, e Irshad R. Mufti. 1983. “Reversed time migration in spatial frequency domain”. *Geophysics* 48, n. 5 (maio): 627–635. <https://doi.org/10.1190/1.1441493>.
- Lucas, Alice, Michael Iliadis, Rafael Molina e Aggelos K. Katsaggelos. 2018. “Using Deep Neural Networks for Inverse Problems in Imaging: Beyond Analytical Methods”. *IEEE Signal Processing Magazine* 35, n. 1 (janeiro): 20–36. <https://doi.org/10.1109/msp.2017.2760358>.
- Ma, Jianwei, e Gerlind Plonka. 2010. “The Curvelet Transform”. *IEEE Signal Processing Magazine* 27, n. 2 (março): 118–133. <https://doi.org/10.1109/msp.2009.935453>.
- McMechan, G. A. 1982. “Determination of source parameters by wavefield extrapolation”. *Geophysical Journal International* 71, n. 3 (dezembro): 613–628. <https://doi.org/10.1111/j.1365-246x.1982.tb02788.x>.
- . 1983. “Migration by extrapolation of time-dependent boundary values”. *Geophysical Prospecting* 31, n. 3 (junho): 413–420. <https://doi.org/10.1111/j.1365-2478.1983.tb01060.x>.
- McMechan, George A. 1989. “A review of seismic acoustic imaging by reverse-time migration”. *International Journal of Imaging Systems and Technology* 1 (1): 18–21. <https://doi.org/10.1002/ima.1850010104>.

- Moradpouri, Farzad, Ali Moradzadeh, Reynam Pestana, Reza Ghaedrahmati e Mehrdad Soleimani Monfared. 2017. “An improvement in wavefield extrapolation and imaging condition to suppress reverse time migration artifacts”. 82, n. 6 (novembro): S403–S409. <https://doi.org/10.1190/geo2016-0475.1>.
- Moreira, Catarina. 2013. “Neurónio”. *Revista de Ciência Elementar* 1 (1).
- Moser, Tijmen Jan. 2012. “Review of ray-Born forward modeling for migration and diffraction analysis”. *Studia Geophysica et Geodaetica* 56 (2): 411–432.
- Nagi, Jawad, Frederick Ducatelle, Gianni A. Di Caro, Dan Ciresan, Ueli Meier, Alessandro Giusti, Farrukh Nagi, Jurgen Schmidhuber e Luca Maria Gambardella. 2011. “Max-pooling convolutional neural networks for vision-based hand gesture recognition”. Em *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*. IEEE, novembro. <https://doi.org/10.1109/icsipa.2011.6144164>.
- Nocedal, Jorge, e Stephen Wright. 2006. *Numerical optimization*. Springer Science & Business Media.
- NVIDIA, Péter Vingelmann e Frank H.P. Fitzek. 2020. *CUDA, release: 10.2.89*. <https://developer.nvidia.com/cuda-toolkit>.
- Papayan, Vardan, Yaniv Romano e Michael Elad. 2016. “Convolutional Neural Networks Analyzed via Convolutional Sparse Coding” (julho). arXiv: [1607.08194](https://arxiv.org/abs/1607.08194) [stat.ML].
- Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg et al. 2011. “Scikit-learn: Machine learning in Python”. *Journal of machine learning research* 12 (Oct): 2825–2830.
- Pratt, Harry, Bryan Williams, Frans Coenen e Yalin Zheng. 2017. “Fconv: Fourier convolutional neural networks”. Em *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 786–798. Springer.
- Raissi, M., P. Perdikaris e G.E. Karniadakis. 2019. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. 378 (fevereiro): 686–707. <https://doi.org/10.1016/j.jcp.2018.10.045>.
- Rashevsky, N. 1938. “Contribution to the mathematical biophysics of visual perception with special reference to the theory of aesthetic values of geometrical patterns”. *Psychometrika* 3, n. 4 (dezembro): 253–271. <https://doi.org/10.1007/bf02287932>.
- Ronneberger, Olaf, Philipp Fischer e Thomas Brox. 2015a. “Dental X-ray image segmentation using a U-shaped Deep Convolutional network”. Em *International Symposium on Biomedical Imaging*, 1–13.
- . 2015b. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. Em *Lecture Notes in Computer Science*, 234–241. Springer International Publishing. https://doi.org/10.1007/978-3-319-24574-4_28.
- Sanavi, Hamideh, Peyman P. Moghaddam e Felix J. Herrmann. 2021. “True amplitude depth migration using curvelets”. 86, n. 4 (julho): S299–S310. <https://doi.org/10.1190/geo2019-0307.1>.

- Saxena, Aditya K, Shweta Sharma e Vijay K Chaurasiya. 2015. “Neural network based human age-group estimation in curvelet domain”. *Procedia Computer Science* 54:781–789. <https://doi.org/https://doi.org/10.1016/j.procs.2015.06.092>.
- Sharma, Aditya. 2017. “Understanding activation functions in deep learning”. *Learn OpenCV Website*, <https://learnopencv.com/understanding-activation-functions-in-deep-learning/> (accessed 21 July 2021).
- Sun, Junzhe, Sergey Fomel e Tiejuan Zhu. 2015. “Preconditioning least-squares RTM in viscoacoustic media by *Q*-compensated RTM”, 3959–3965. SEG Technical Program Expanded Abstracts 0. Society of Exploration Geophysicists, agosto. <https://doi.org/10.1190/segam2015-5880944.1>. <https://doi.org/10.1190/segam2015-5880944.1>.
- Sun, R., e G.A. McMechan. 1986. “Pre-stack reverse-time migration for elastic waves with application to synthetic offset vertical seismic profiles”. *Proceedings of the IEEE* 74 (3): 457–465. <https://doi.org/10.1109/proc.1986.13486>.
- Tarantola, A. 1984. “Linearized inversion of seismic reflection data”. *Geophysical Prospecting* 32, n. 6 (dezembro): 998–1015. <https://doi.org/10.1111/j.1365-2478.1984.tb00751.x>.
- Wang, Ping, Adriano Gomes, Zhigang Zhang e Ming Wang. 2016. “Least-squares RTM: Reality and possibilities for subsalt imaging”. Em *SEG Technical Program Expanded Abstracts 2016*. Society of Exploration Geophysicists, setembro. <https://doi.org/10.1190/segam2016-13867926.1>.
- Whitmore, N. D. 1983. “Iterative depth migration by backward time propagation”. Em *SEG Technical Program Expanded Abstracts 1983*. Society of Exploration Geophysicists, janeiro. <https://doi.org/10.1190/1.1893867>.
- Yasoubi, Ali, Reza Hojabr e Mehdi Modarressi. 2017. “Power-Efficient Accelerator Design for Neural Networks Using Computation Reuse”. *IEEE Computer Architecture Letters* 16, n. 1 (janeiro): 72–75. <https://doi.org/10.1109/lca.2016.2521654>.