UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Ilha Solteira

Natalia Caroline Lopes Travassos

Armazenamento e reconstrução de imagens comprimidas via codificação e decodificação

Ilha Solteira
2016

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Ilha Solteira

Natalia Caroline Lopes Travassos

Armazenamento e reconstrução de imagens comprimidas via codificação e decodificação

Dissertação apresentada à Faculdade de Engenharia do Campus de Ilha Solteira - UNESP, como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica.
Área de Conhecimento: Automação.

Orientador: Dr. Francisco Villarreal Alvarado

Ilha Solteira
2016

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

T779a

Travassos, Natalia Caroline Lopes.
Armazenamento e reconstrução de imagens comprimidas via codificação e decodificação / Natalia Caroline Lopes Travassos. -- Ilha Solteira: [s.n.], 2016
72 f. : il.

Dissertação (mestrado) - Universidade Estadual Paulista. Faculdade de Engenharia . Área de conhecimento: Automação, 2016

Orientador: Francisco Villarreal Alvarado
Inclui bibliografia

1. Compressão. 2. Codificação. 3. Decodificação. 4. Armazenamento de Imagens.

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Ilha Solteira

CERTIFICADO DE APROVAÇÃO

TÍTULO DA DISSERTAÇÃO: Armazenamento e reconstrução de imagens comprimidas via codificação e decodificação

AUTORA: NATALIA CAROLINE LOPES TRAVASSOS
ORIENTADOR: FRANCISCO VILLARREAL ALVARADO

Aprovada como parte das exigências para obtenção do Título de Mestra em ENGENHARIA ELÉTRICA, área: AUTOMAÇÃO pela Comissão Examinadora:

Prof. Dr. FRANCISCO VILLARREAL ALVARADO
Departamento de Matemática / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. RICARDO TOKIO HIGUTI
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Profa. Dra. TATIANA BERTOLDI CARLOS
Departamento de Matemática / Universidade Federal de Mato Grosso do Sul

Ilha Solteira, 19 de dezembro de 2016

Dedico a Deus e à minha família.

Agradecimentos

Agradeço a Deus pelas bençãos, sabedoria e conhecimentos adquiridos durante esta jornada.

Ao professor Dr. Francisco Villarreal Alvarado pelo apoio, confiança e orientação deste trabalho. Ao professor Dr. Marco Aparecido Queiroz Duarte pela dedicação, paciência, prontidão em oferecer seu conhecimento e pelas sugestões. A todos funcionários da UNESP, pela solicitude.

Ao meu querido esposo Marco, por ser tão importante em minha vida, contribuindo para este trabalho com seus conhecimentos, apoio emocional e amor incondicional.

À minha avó Izolina e minha mãe Santina, que me sustentam com suas orações e que foram as maiores incentivadoras para que eu chegasse até aqui.

Aos professores da UFMS/CPTL, que fizeram parte da minha formação pessoal e intelectual. Em especial aos professores Dr. Antônio Carlos Tamarozzi e Fernando Pereira de Souza, por acreditarem e investirem em mim.

Aos amigos que já tinha, Patrícia e Hugo. Aos que fiz no mestrado, Monara, Mário e Fabrício. Aos amigos em geral, pela amizade e apoio.

A Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo suporte financeiro.

"If I have seen further
it is by standing on the shoulders of giants."
(Isaac Newton)

Resumo

Este trabalho apresenta um algoritmo de codificação para imagens comprimidas que representa cada pixel de uma imagem e suas coordenadas por um único valor. Para cada pixel e suas coordenadas, esse valor único é armazenado em um vetor que é usado na reconstrução da imagem sem que sua qualidade seja comprometida. O método proposto apresenta melhorias em relação a dois outros algoritmos propostos anteriormente, sendo que um deles já é uma melhoria do primeiro. O algoritmo apresentado neste trabalho difere dos outros dois algoritmos estudados na diminuição significativa do espaço necessário para armazenamento das imagens, na determinação de uma taxa de compressão exata e na redução do tempo de processamento de decodificação. Um outro avanço apresentado foi a compressão de imagens coloridas utilizando a ferramenta `wavemenu` em conjunto com o algoritmo que determina a taxa de compressão.

Palavras-chave: Compressão. Codificação. Decodificação. Armazenamento de imagens.

Abstract

This work presents an encoding algorithm for compressed images that represents each pixel of an image and its coordinates by a single value. For each pixel and its coordinates, this unique value is stored in a vector that is used in the reconstruction of the image without its quality being compromised. The proposed method presents improvements in relation to two other algorithms previously proposed, one of which is already an improvement for the first one. The algorithm presented in this work differs from the other ones in the foollowing characteristcs: by the significant reduction of the space required for image storage, the determination of an exact compression rate and the reduction of the decoding processing time. Another advancement was the compression of colored images using the tool `wavemenu` in improvement with the algorithm that determines the compression ratio.

Keywords: Compression. Encoding. Decoding. Image storage.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AMR	Análise de Multirresolução
bit	Binary Digit
Byte	Binary Term
dB	Decibel
`double(x)`	Função do Matlab que retorna o valor de dupla-precisão para x
`db6`	Função Wavelet de Daubechies Nível 6
ECG	Eletrocardiograma
JPEG	Join Picture Expert Group
kByte	Kilobyte (kB)
LZW	Lempel-Ziv-Welch
MATLAB	Matrix Laboratory
MLP	Multilayer Perceptron
MSE	Mean Squared Error
PIXEL	Picture Element
PSNR	Peak Signal-to-noise Ratio
QMF	Quadrature Mirror Filter
REZW	Robust Embeeded Zerotree Wavelet
RGB	Red-Green-Blue
RLE	Run-length Encoding
RLEA	Run-length Encoding Adaptativo
TW	Transformada Wavelet
TWD	Transformada Wavelet Discreta
TWDI	Transformada Wavelet Discreta Inversa

Lista de símbolos

$\mathbb{R}$	Conjunto dos números reais
$\mathbb{N}$	Conjunto dos numeros naturais
$\mathbb{Z}$	Conjunto dos números inteiros
$L^2(\mathbb{R})$	Espaço das funções mensuráveis de Lebesgue de quadrado integrável
N	Função que resume os passos de Silva, Duarte e Villarreal (2011)
M	Inversa da função N
m, n	Linhas e colunas de uma matriz $m \times n$
V_j	Espaço de Escala 2^j
W_j	Complemento ortogonal de V_j
φ	Função Wavelet
ϕ	Função Escala
$\hat{\varphi}$	Transformada de Fourier de φ
V_cod	Vetor codificado de Silva (2008)
V_cod_mat	Vetor codificado de Silva, Duarte e Villarreal (2011)
V_cod_prop	Vetor codificado do método proposto
σ	Estimativa do ruído
λ	Limiar usado para a filtragem dos sinais
$\hat{y}$	Sinal filtrado pelo limiar λ
$\|.\|$	Valor absoluto
$\langle , \rangle$	Produto interno
$\|\|.\|\|$	Norma euclidiana
$\text{proj}_\text{u}(\text{v})$	Projeção de v em u
$\downarrow 2$	Operador de subamostragem
sgn	Função sinal

`sign`	Função sinal do Matlab
`mod`(a, b)	Função do Matlab que retorna o resto da divisão de a por b.
`round`(x)	Função do Matlab que arredonda x para o inteiro mais próximo
`floor`(x)	Função do Matlab que arredonda x na direção de menos infinito.
$\oplus$	Soma direta de conjuntos
j	Nível de resolução em Análise de Multirresolução

Sumário

1 INTRODUÇÃO

A grande quantidade de informações trocadas nos dias atuais faz com que a compactação, ou compressão, de dados seja algo altamente desejado em diversos ambientes. Dados econômicos, de previsão de tempo, textos, informações biomédicas, gravações de áudios, vídeos e imagens das mais diversas naturezas são algumas das informações que são constantemente transmitidas pelos meios de comunicação, sendo que o principal deles é a internet. Tais informações, quando preparadas para transmissão, são consideradas sinais. E estes sinais necessitam ser analisados e, na maioria das vezes, são comprimidos para um armazenamento eficaz.

Assim, podemos dizer que vários tipos de sinais fazem parte do nosso dia-a-dia, a saber; uma imagem é um sinal e, mais precisamente, uma função é um sinal. Deste modo, o processamento de sinais tem papel fundamental na melhoria da qualidade da informação.

A compressão de sinais se faz necessária nos dias atuais, pois grande é a demanda por dados digitais que necessitam ser compartilhados ou armazenados diariamente. Na compressão, uma imagem processada e a imagem original podem ter diferenças mínimas, e isto se deve ao fato de que coeficientes que não comprometem a reconstrução da mesma são suprimidos. Após a compressão, são armazenados apenas os coeficientes que não foram eliminados na imagem transformada e as informações que possibilitam sua reconstrução através desses coeficientes (SILVA, 2008).

Uma ferramenta matemática que se destaca no processamento de sinais, mais ainda quando se trata de imagens, é a transformada wavelet (DAUBECHIES, 1992). Devido as propriedades da análise de multirresolução, a transformada wavelet possibilita que uma imagem seja decomposta em vários níveis de resolução de forma que suas características sejam totalmente explicitadas (DAUBECHIES, 1992; STOLLNITZ; DEROSE; SALESIN, 1995; MALLAT, 1989b). Isto possibilita uma análise detalhada da mesma e uma compressão muito eficaz (COIFMAN, 1990).

Uma das aplicações do processamento de sinais é a compressão de dados, cujo objetivo fundamental é a transmissão ou o armazenamento eficientes, preservando ao mesmo tempo as características significativas (SINGH; KAUR; SINGH, 2015). No caso deste trabalho, utilizamos a transformada wavelet para a compressão, entretanto a mesma possui diversas áreas de aplicações, por exemplo: na mecânica de fluidos (simulação de vórtice coerente), computação numérica (resolução numérica de equações diferenciais), análise de imagens (decomposição hierárquica de funções), processamento de sinais (compressão), sistemas de controle (controle adaptativo de sistemas dinâmicos), medicina (análise de sinais de eletrocardiograma) e psicologia (análise de séries temporais) (BIANCHI, 2006).

A análise do sinal pela transformada wavelet permite a obtenção de informações tanto no domínio do tempo quanto no domínio da frequência. Wavelets mais simples são capazes de processar alguns tipos de sinais. Entretanto não são capazes de processar todos os tipos existentes, porém existe um grande número de wavelets que processam os mais variados tipos de sinais, e se caso não acharmos a wavelet ideal dentro desse conjunto de wavelets podemos construir a nossa própria função (SILVA, 2008).

Neste trabalho, propõe-se um algoritmo de compressão de imagens implementado especialmente para posterior codificação de imagens digitais que passam a ser representadas por um único vetor de codificação, sem que a reconstrução das mesmas seja comprometida. O método busca eficiência, facilidade de implementação e bons resultados na área de processamento de imagens. O algoritmo proposto é uma versão modificada dos algoritmos descritos em (SILVA, 2008; SILVA; DUARTE; VILLARREAL, 2011). O algoritmo de Silva, Duarte e Villarreal (2011), por sua vez, é baseado no algoritmo de Silva (2008), e é em média 33,3% mais econômico que o anterior. O método proposto ainda faz com que seja possível a escolha a priori da taxa de compressão. Esta característica é importante em termos de largura de banda, espaço de armazenamento, complexidade computacional e custo. O algoritmo apresentado neste trabalho difere dos algoritmos estudados por Silva (2008) e Silva, Duarte e Villarreal (2011), na diminuição do espaço necessário para armazenamento das imagens, determinação a priori da taxa de compressão, redução do tempo necessário para decodificar o vetor proposto e reconstruir a imagem.

O restante do texto é organizado da seguinte forma: No Capítulo 2 há um breve resumo sobre imagens digitais e como são armazenadas. No Capítulo 3 é apresentado uma introdução à teoria wavelet utilizada, onde se enfatizam os aspectos da transformada wavelet julgados mais importantes para a compressão de imagens. O Capítulo 4 trata sobre a compressão de sinais e métodos propostos em trabalhos realizados recentemente, além de uma introdução sobre os limiares mais utilizados, rígido e suave. No Capítulo 5 são apresentados os algoritmos de Silva (2008) e Silva, Duarte e Villarreal (2011), bem como os resultados por eles obtidos. No Capítulo 6 é apresentado o algoritmo proposto e os resultados obtidos; a comparação com os resultados de Silva (2008) e Silva, Duarte e Villarreal (2011); as melhorias obtidas em relação à taxa de compressão, tempo de processamento, processamento de imagens coloridas e principalmente a redução de espaço computacional. Por fim, no Capítulo 7 são dadas as considerações finais.

2 PROCESSAMENTO DE IMAGENS

O processamento de imagem digital pode ser considerado como a subárea do processamento digital. Como tal, todos os métodos para falar e analisar medições e seus erros também podem ser aplicados ao processamento de imagem (JAHNE, 1997). Ele surge com o objetivo de suprir as necessidades de análise e compactação de informações, tornando-se assim uma ferramenta essencial no mundo moderno, cuja demanda de processamento é relativamente crescente (GONZALEZ; WOODS, 2000).

2.1 AQUISIÇÃO DE IMAGENS

O processo de aquisição de imagens é a fase em que obtém-se a representação da informação visual, que deve ser a mais fiel possível e ao mesmo tempo ser processável por um computador. Normalmente, tal representação é construída a partir de alguma fonte de radiação, por exemplo calor, raios X, luz visível, etc..., captada por dispositivos sensíveis a tais radiações. A Figura 1 ilustra o processo de aquisição de uma imagem, através de um dispositivo de captura, permitindo que a imagem obtida seja convertida em um sinal digital.

Figura 1 – Processo de aquisição de imagem: (a) Imagem adquirida da flor (b) Dispositivo de captura (c) Sinal digital.

Fonte: Adaptado de Silva (2008).

Durante o processo de aquisição de imagem, os sensores fotográficos que medem as informações armazenadas posteriormente em cada pixel, primeiro, restauram um sinal analógico que deve ser convertido em valores numéricos. Durante esta conversão analógico-digital, há discretização, que é quando a imagem é convertida em valores numéricos, e só após esse processo o computador poderá analisá-la (BAGHDADI; ZRIBI, 2016). A imagem então será composta por pixels. Os pixels determinam a resolução da imagem, ou seja, quanto mais pixels melhor será sua qualidade.

2.2 ARMAZENAMENTO DE IMAGENS

Um aspecto de grande interesse é o armazenamento da imagem na memória do computador. Um exemplo que apresenta essa necessidade é o problema com imagens médicas, já que elas são grandes em tamanho, e assim, laboratórios e hospitais necessitam de uma grande quantidade de largura de banda para enviar e receber imagens. Considerando um hospital que faça ressonância magnética, tomografia computadorizada, radiografia computorizada, etc., pode-se estimar que serão produzidos de 5 a 15 GByte de dados por dia (BAIRAGI, 2015). Neste caso, e em tantos outros, é necessário comprimir estes dados utilizando uma técnica de compressão.

O número de bits necessários para o armazenamento de uma imagem na memória do computador é dado pela equação (1) (GONZALEZ; WOODS, 2000):

$$bits = M \times N \times n \tag{1}$$

onde $n = log_2(M)$, M é o número de linhas e N, o número de colunas da imagem.

2.3 A NECESSIDADE DA COMPRESSÃO DE IMAGENS

Uma imagem, depois de discretizada e codificada, carrega consigo uma quantidade muito grande de informação, e eventualmente ocupará muito espaço para o seu armazenamento (GOMES; VELHO; GOLDENSTEIN, 1997).

Tomando como exemplo uma imagem cuja dimensão é 256×256 *pixels*, haverá 65536 pontos a serem codificados. Caso o armazenamento seja feito por uma matriz, e supondo que a quantização da cor usa 256 níveis (1 byte) para cada um dos três canais RGB (vermelho, verde e azul), haverá a necessidade de três bytes para cada ponto, logo, existirão 196608 pontos a serem codificados. O exemplo dado serve para ilustrar a necessidade da codificação, para posterior armazenamento.

Os métodos de compressão se dividem em: com perdas e sem perdas. Tais métodos serão abordados com um pouco mais de detalhes mais à frente.

A seguir daremos mais detalhes sobre a transformada wavelet, que é usada em vários métodos de compressão. Sua grande vantagem é concentrar a informação relevante em um número pequeno de coeficientes.

3 TEORIA WAVELET: CONCEITOS FUNDAMENTAIS

O termo wavelet tem como origem a palavra francesa "Ondelette", cujo significado é "onda pequena". Um histórico sobre as wavelets é apresentado por (JAWERTH; SWELDENS, 1994; GRAPS, 1995). A Transformada de Haar é o mais antigo dos métodos de transformada wavelet, a primeira menção às wavelets apareceu em um apêndice na tese de Haar (1910 apud GRAPS, 1995). A análise wavelet mostra muitas origens diferentes, parte do trabalho se desenvolveu muito na década de 1930 e, na época, os esforços realizados separadamente não pareciam ser parte de uma teoria coesa. Ainda nesta década, vários grupos trabalhando de forma independente pesquisaram a representação de funções usando uma base variando com a escala. Usando wavelets de Haar como base, Paul Levy, um físico dos anos 30, investigou o movimento browniano, um tipo de sinal aleatório. Outro esforço na pesquisa em 1930 veio pelos trabalhos realizados por Littlewood, Paley e Stein. O trabalho deles forneceu a David Marr um algoritmo eficaz para processamento de imagem numérica usando wavelets, no início dos anos 1980. Entre 1960 e 1980, os matemáticos Guido Weiss e Ronald R. Coifman também deram sua parcela de contribuição, permitindo que em 1980, Grossman e Morlet, físico e engenheiro, definissem amplamente wavelets no contexto da física quântica. Em 1985, Mallat deu às wavelets um grande impulso através de seu trabalho em processamento digital de imagens. Yves Meyer, inspirado nos resultados de Mallat, construiu a primeira wavelet não-trivial. Ao contrário das wavelets de Haar, as wavelets de Meyer são continuamente diferenciáveis, mas não têm suportes compactos (GRAPS, 1995). Em 1988, Mallat desenvolveu uma teoria denominada análise de multirresolução. No ano seguinte ele mostrou que a análise de multirresolução pode ser vista simplesmente como uma forma de algoritmos de pirâmide que é usado para calcular a transformada wavelet (MALLAT, 1989b). Em 1990, Ingrid Daubechies usou os trabalhos de Mallat para construir um conjunto de bases ortonormais de wavelets suaves com suportes compactos. Os trabalhos de Daubechies são os alicerces das aplicações atuais das wavelets.

3.1 TRANSFORMADA WAVELET (TW)

As notações a seguir, bem como a teoria, baseiam-se em Kaiser (1994), e caso não haja familiaridade com a linguagem usada, os livros de Guidorizzi (2001) e Coelho e Lourenço (2001) trazem conceitos básicos de cálculo e álgebra linear que podem auxiliar.

Dada uma função $\varphi(x)$, e tomando uma família de funções variando a escala: $p \geq 0$ é fixo, e para todo $s \in \mathbb{R}$, $s \neq 0$ definimos:

$$\varphi_s(u) = |s|^{-p} \varphi\left(\frac{u}{s}\right).$$

Se φ possui largura T (calculada de acordo com o princípio da incerteza, encontrado em Gomes, Velho e Goldenstein (1997)), então a largura de φ_s é sT. Cada função φ deve ser localizada no tempo, para isso, definimos para cada $t \in \mathbb{R}$ a função (2):

$$\varphi_{s,t}(u) = \varphi_s(u-t) = |s|^{-p}\varphi\left(\frac{u-t}{s}\right). \tag{2}$$

Pode-se definir uma transformada utilizando as funções $\varphi_{s,t}$ como função moduladora de f. Mais precisamente,

$$\tilde{f}(s,t) = \int_{-\infty}^{\infty} f(u)\varphi_{s,t}(u)du = \langle \varphi_{s,t}, f\rangle. \tag{3}$$

A transformada definida em (3) é conhecida como *transformada wavelet* (TW). Para que φ seja chamada de wavelet, é necessário satisfazer a condição (4):

$$C_\varphi = \int_{-\infty}^{\infty} \frac{|\hat{\varphi}(u)|^2}{|u|}du < \infty \tag{4}$$

onde $\hat{\varphi}(u)$ é a transformada de Fourier de $\varphi(u)$. A equação (5) segue da *condição de admissibilidade*, (4):

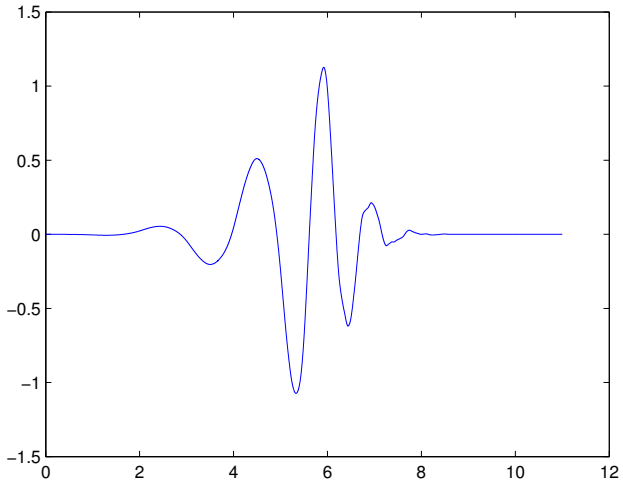
$$\lim_{u\to\infty} \hat{\varphi}(u) = 0. \tag{5}$$

Assim, se $\hat{\varphi}(u)$ é contínua então, $\hat{\varphi}(0) = 0$, ou seja:

$$\int_{-\infty}^{\infty} \varphi(u)du = 0. \tag{6}$$

O gráfico de uma wavelet φ tem forma de onda de curta duração, pois decorre de (6) que o gráfico da função oscila de modo a cancelar áreas positivas e negativas. Um exemplo do gráfico de uma wavelet é dado na Figura 2.

Figura 2 – Exemplo do gráfico de uma wavelet (Wavelet Daubechies 6).



Fonte: Elaboração da própria autora.

3.2 TRANSFORMADA WAVELET DISCRETA (TWD)

A discretização da transformada wavelet $\tilde{f}(s,t) = \langle f, \varphi_{s,t}(u)\rangle$ é dada por

$$\tilde{f}_{m,n}\langle f, \varphi_{m,n}(u)\rangle$$

onde, $\varphi_{m,n}(u)$ é como na equação (7).

$$\begin{aligned} \varphi_{m,n}(u) &= \varphi_{s_0^m, nt_0 s_0^m}(u) \\ &= s_0^{-m/2}\varphi\left(\frac{u - nt_0 s_0^m}{s_0^m}\right) \\ &= s_0^{-m/2}\varphi\left(s_0^{-m}u - nt_0\right), \end{aligned} \tag{7}$$

sendo t_0 o tempo, $s_0^m t_0$ a duração dos intervalos de amostragem do tempo e s_0^m a escala.

3.3 ANÁLISE DE MULTIRRESOLUÇÃO (AMR)

3.3.1 Espaços de escala

Para cada $j \in \mathbb{Z}$, pode-se criar um subespaço $V_j \subset L^2(\mathbb{R})$ formado pelas funções cujos detalhes estão na escala 2^j.

Existe uma função $\phi \in L^2(\mathbb{R})$ tal que a família de funções

$$\phi_{j,k}(u) = 2^{-j/2}\phi(2^{-j}u - k), \quad j,k \in \mathbb{Z}, \tag{8}$$

é base ortonormal de V_j, provado por Stromberg (1981) e Meyer (1985-1986 apud MALLAT, 1989b). A representação de uma função $f \in L^2(\mathbb{R})$ por projeção ortogonal em V_j é dada por

$$Proj_{V_j}(f) = \sum_k \langle f, \phi_{j,k}\rangle \phi_{j,k}.$$

O espaço V_j é chamado de *espaço de escala*.

A largura da função $\phi_{j,k}$ e ϕ são relacionadas pela equação

$$largura(\phi_{j,k}) = 2^j largura(\phi).$$

Portanto, quando j decresce, a largura de $\phi_{j,k}$ diminui, refinando a escala. Como os detalhes do sinal que aparecem na escala 2^j certamente devem estar presentes na escala 2^{j+1} (GOMES; VELHO; GOLDENSTEIN, 1997), temos a inclusão em (9):

$$V_j \subset V_{j-1}. \tag{9}$$

Dada uma função $f \in L^2(\mathbb{R})$, tem-se:

$$f \in V_j \Leftrightarrow f(2u) \in V_{j-1}$$

Tomando $j = 0$ na equação (8), tem-se

$$\phi_{0,k}(u) = \phi(u - k),$$

que é base ortonormal do espaço de escala V_0. O espaço $L^2(\mathbb{R})$ contém todas as possíveis escalas, o que pode ser escrito como

$$\overline{\bigcup_{j\in\mathbb{Z}} V_j} = L^2(\mathbb{R}).$$

Por outro lado, a interseção dos V_j resulta na função nula, a única que pode ser representada em qualquer escala:

$$\bigcap_{j\in\mathbb{Z}} V_j = \{0\}.$$

3.3.2 Analise de Multirresolução

Adaptando a definição dada por Mallat (1989a), temos a seguinte definição:

Definição 1. U*ma análise de multirresolução (AMR) em* $L^2(\mathbb{R})$ *é uma sequência de subespaços fechados* V_j, $j \in \mathbb{Z}$, *de* $L^2(\mathbb{R})$, *satisfazendo as seguintes propriedades:*

(M1) $V_j \subset V_{j-1}$

(M2) $f \in V_j \Leftrightarrow f(2u) \in V_{j-1}$

(M3) $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$

(M4) $\overline{\bigcup_{j\in\mathbb{Z}} V_j} = L^2(\mathbb{R})$

(M5) *Existe uma função* $\phi \in V_0$ *tal que* $\{\phi(u-k);\, k \in \mathbb{Z}\}$ *é uma base ortonormal de* V_0

(M3) é consequência de **(M1)**,**(M2)** e **(M5)**. A demonstração para tal fato e outros, oriundos das consequências acima, são encontradas em (HERNÁNDEZ; WEISS, 1996; DAUBECHIES, 1992).

3.4 MULTIRRESOLUÇÃO E WAVELETS

Para todo $j \in \mathbb{Z}$, definimos W_j como sendo o complemento ortogonal de V_j em V_{j-1}, vale a igualdade (10).

$$V_{j-1} = V_j \oplus W_j \tag{10}$$

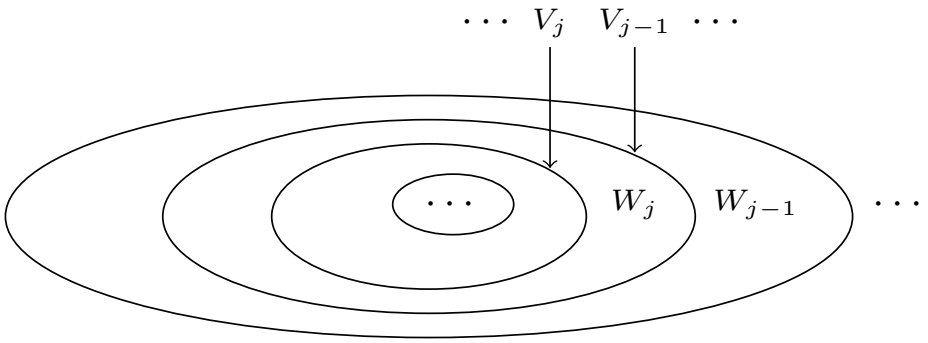
O espaço W_j contém os detalhes do sinal na escala V_j, e pode ser obtido usando uma filtragem de passa-banda(as wavelets constituem filtros de passa-banda). A Figura 3 mostra como é a distribuição dos espaços V e W.

Como $\phi \in V_0 \subset V_{-1}$ e, além disso, $\{\phi_{-1,k}\}_{k\in\mathbb{N}}$ é base ortonormal de V_{-1}, podemos escrever

$$\phi = \sum_k h_k \phi_{-1,k}, \tag{11}$$

onde $h_k = \langle \phi, \phi_{-1,k} \rangle$ e $\sum_{k\in\mathbb{Z}} \|h_k\|^2 = 1$.

Figura 3 – Distribuição dos subespaços W e V.



Fonte: Adaptado de Ghanbari (2003).

Substituindo $\phi_{-1,k}(u) = \sqrt{2}\phi(2u - k)$ em (11), tem-se a equação (12), que é conhecida por vários nomes diferentes: equação de refinamento, equação de dilatação ou equação de diferença de escala-dois.

$$\phi(t) = \sqrt{2}\sum_k h_k \phi(2t - k). \tag{12}$$

Também podemos expressar a wavelet φ em termos da função de escala ϕ, segundo a equação (13), para um conjunto finito de coeficientes g_k. Desta forma ϕ e φ ficam completamente determinadas de forma recursiva, respectivamente, pelas sequências $\{h\}_k$ e $\{g\}_k$.

$$\varphi(t) = \sqrt{2}\sum_k g_k \phi(2t - k). \tag{13}$$

3.5 TRANSFORMADA WAVELET BIDIMENSIONAL

A transformada wavelet assemelha-se a um banco de filtros, constituído de dois filtros: H (passa-baixa) e G (passa-alta), chamados de *quadrature mirror filters* (QMF). A aplicação dos QMF sobre a imagem nas direções vertical e horizontal gera um nível de decomposição e produz quatro sub-bandas, LL (aproximação), LH (detalhes horizontais), HL (detalhes verticais) e HH (detalhes diagonais), e este processo pode ser realizado recursivamente na sub-banda LL .

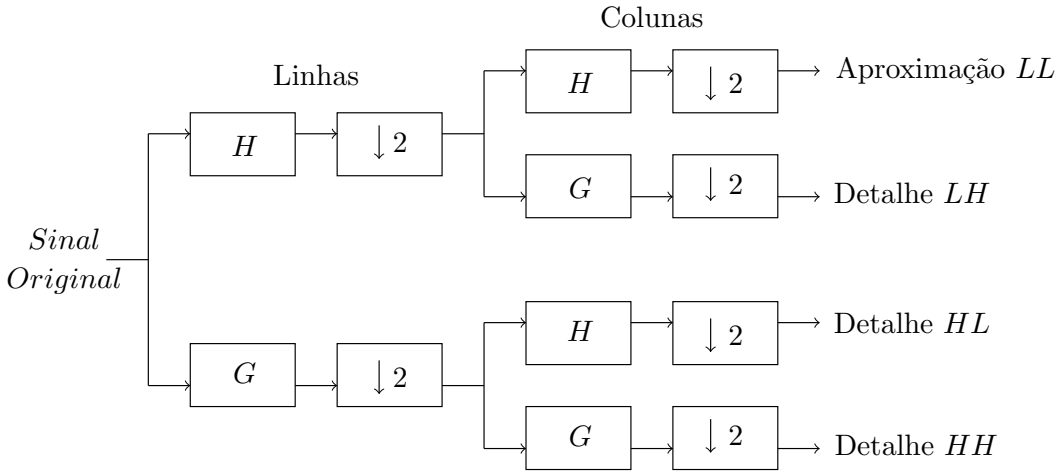
A análise wavelet pode ser feita em qualquer número de dimensões. As ideias essenciais, no entanto, são reveladas em duas dimensões. Muitas das ideias básicas são análogas para o caso 1D.

Existem duas maneiras pelas quais podemos usar wavelets para decompor uma

imagem bidimensional (STOLLNITZ; DEROSE; SALESIN, 1996): decomposição padrão e decomposição não padrão.

Na decomposição padrão, aplica-se a TW unidimensional em cada linha da imagem. Na sequência, consideramos as linhas transformadas como uma nova imagem e a TW unidimensional é aplicada em cada coluna. Aplicando a TW em todos os níveis de resolução, o resultado será todos os coeficientes de detalhes, exceto o primeiro pixel, que corresponde ao único coeficiente de aproximação. As Figuras 4 e 5 ilustram o processo da decomposição padrão. Na Figura 4 tem-se o processo de filtragem da imagem e na Figura 5, as suas faixas de decomposição.

Figura 4 – Transformada discreta wavelet bidimensional (Padrão).

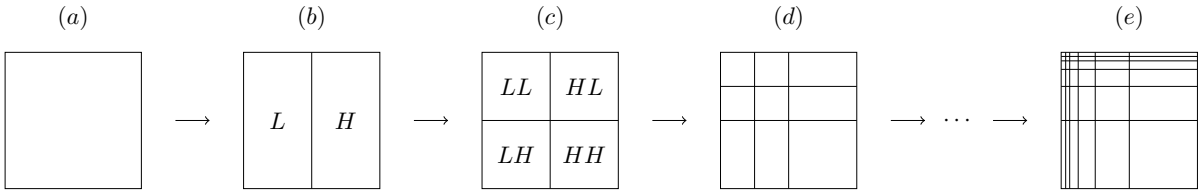


Fonte: Adaptado de Sanches (2001).

Dada uma imagem $x(m, n)$, ela é inicialmente filtrada na direção m (linhas da imagem), resultando numa imagem passa-baixa L e uma imagem passa-alta H. Após a subamostragem de ordem 2 ($\downarrow 2$), teremos ambas as imagens reduzidas pela metade em relação a imagem original. Em seguida realiza-se a filtragem na direção n (colunas da imagem) resultando em quatro subimagens: LL, LH, HL e HH (SANCHES, 2001).

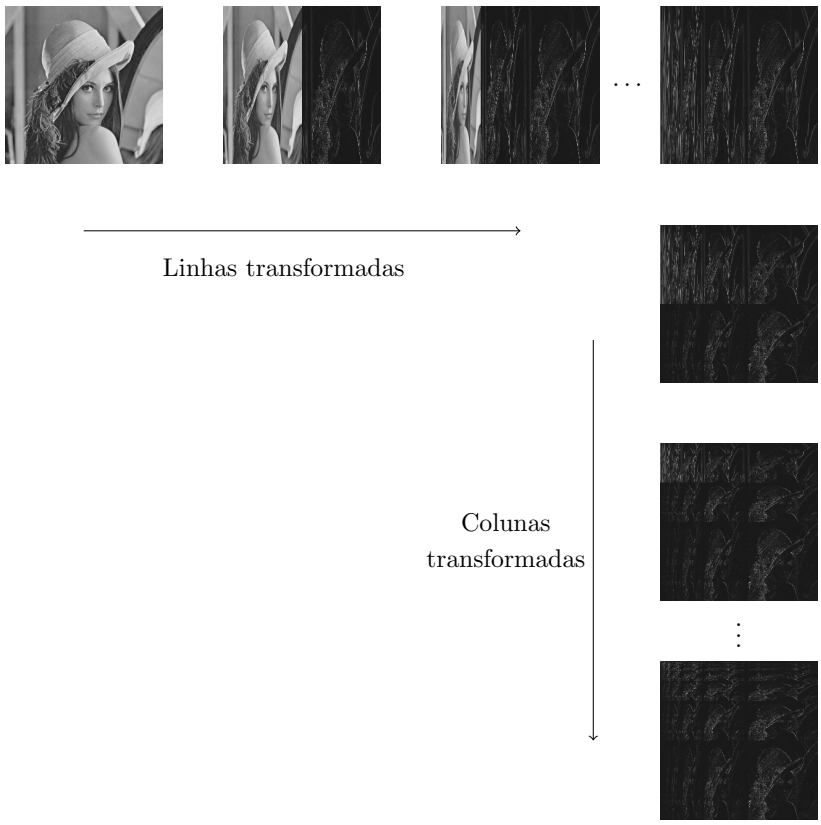
Na Figura 6 mostramos a aplicação da transformada wavelet padrão, usando a imagem Lena. Inicialmente a transformada é aplicada apenas nas linhas da imagem, e em seguida apenas nas colunas.

Figura 5 – Estágios de decomposição wavelet bidimensional padrão com 5 níveis de resolução.



Fonte: Adaptado de Sanches (2001).

Figura 6 – Decomposição padrão da imagem Lena.



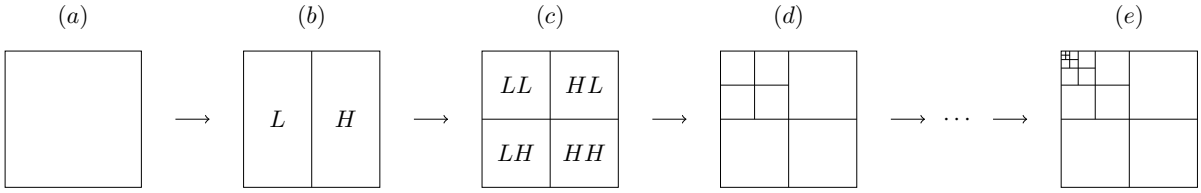
Fonte: Adaptado de Stollnitz, DeRose e Salesin (1995) com imagens de Sanches (2001).

Na decomposição não padrão primeiro, aplica-se a TW em cada linha da imagem. Em seguida, ela é aplicada em cada coluna. Após estas operações obteremos a imagem da Figura 7-(c), composta por quatro imagens menores. A imagem do canto superior esquerdo contém os coeficientes de baixa resolução, enquanto que as três demais imagens contêm os coeficientes de alta resolução. Por fim, repetimos este processo recursivamente,

somente nos quadrantes contendo os coeficientes de baixa resolução, em ambas as direções.

Se uma imagem for decomposta em N níveis de resolução ($N > 0$ e $N \in \mathbb{Z}$), isso implicará na obtençao de $3N + 1$ subimagens.

Figura 7 – Estágios de decomposição wavelet bidimensional não padrão.



Fonte: Adaptado de Sanches (2001).

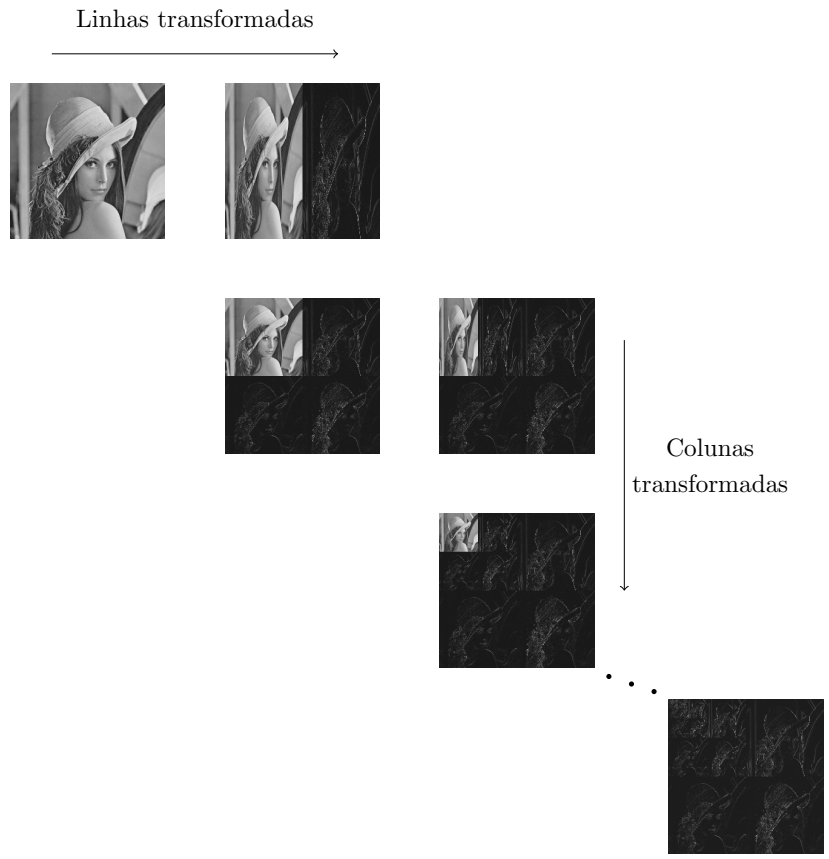
A Figura 7 ilustra os cinco primeiros níveis de resolução de uma imagem bidimensional após a aplicação da transformada wavelet não padrão. Em 7-(a) temos a imagem original, 7-(b) a imagem após a aplicação da transformada em suas linhas, 7-(c) aplicação da transformada nas linhas e nas colunas, um nível de resolução, 7-(d) a imagem com dois níveis de resolução e finalmente em 7-(e) a imagem após cinco níveis de resolução.

Na Figura 8 mostramos a aplicação da decomposição wavelet não padrão, usando a imagem Lena. A decomposição padrão de uma imagem é atraente porque simplesmente requer a realização de transformadas unidimensionais em todas as linhas e, em seguida, em todas as colunas. Por outro lado, é um pouco mais eficiente calcular a decomposição não padrão. Para uma imagem $m \times m$, a decomposição padrão requer o cálculo de $4(m^2 - m)$ operações, enquanto que a decomposição não padrão requer somente $\frac{8}{3}(m^2 - 1)$ operações (STOLLNITZ; DEROSE; SALESIN, 1996).

Podemos verificar que a transformada wavelet permite armazenar uma imagem em diversas resoluções, ilustrado também na Figura 9. Dessa maneira, podemos transmitir inicialmente os coeficientes da imagem com menor resolução, permitindo assim a visualização de uma aproximação da imagem (SANCHES, 2001).

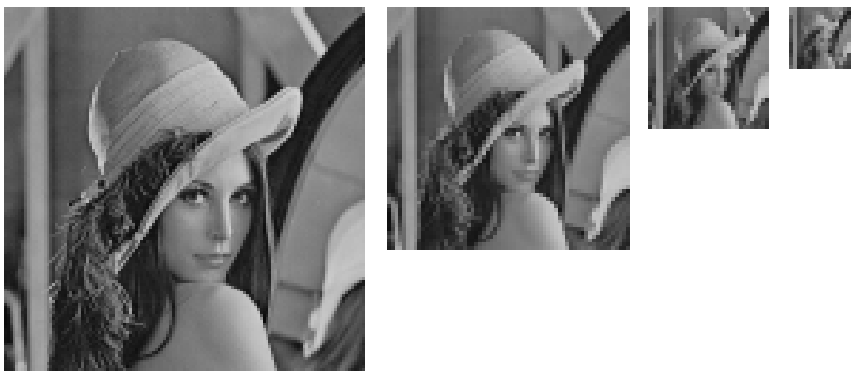
A TW concentra as informações da imagem em um número pequeno de coeficientes. Se fixarmos um limiar λ e substituir por zero todos os coeficientes transformados, que tenham valor absoluto menor ou igual a λ, teremos uma matriz esparsa. Quanto maior for o limiar maior será o fator de compressão, pois uma maior quantidade de valores serão zerados, porém com uma menor qualidade da imagem reconstruída. Este processo é chamado de compressão , e será abordado no próximo capítulo.

Figura 8 – Decomposição não padrão da imagem Lena.



Fonte: Adaptado de Stollnitz, DeRose e Salesin (1995) com imagens de Sanches (2001).

Figura 9 – Lena em várias resoluções.



Fonte: Adaptado de Sanches (2001).

4 WAVELETS E PROCESSAMENTO DE IMAGENS

Há um interesse e uma necessidade de se obter métodos de codificação de imagens que apresentem altas taxas de compressão, o que faz com que as pesquisas sobre esse tema cresçam cada vez mais. O principal objetivo da compressão de imagens é representá-las com a menor quantidade de bits possível, preservando a qualidade exigida para uma dada aplicação. A codificação por transformada é um dos métodos mais utilizados na compressão de imagens. No domínio da transformada procura-se concentrar a maior quantidade de energia possível em um menor número de coeficientes.

Como já citado anteriormente, a transformada wavelet se destaca na compressão de imagens, além do mais, entre essa classe de transformadas e o sistema visual humano, há uma estreita relação: Olshausen e Field (2004) apresentam alguns paralelos entre o sistema visual humano e wavelet, suas análises baseiam-se no fato que o cérebro representa imagens de uma forma esparsa em um número relativamente pequeno de neurônios ativos, enquanto a wavelet tem a habilidade de representar dados de forma esparsa usando um pequeno conjunto de coeficientes. Isso faz com que possamos obter altas taxas de compressão e ainda assim perceber pouca degradação. Das diversas propriedades da TW, uma que se destaca é a sua flexibilidade com relação à duração e à posição das funções de base da representação, o que permite que sinais com conteúdos de frequência variáveis, como imagens, possam ser bem representados com um número reduzido de funções.

Neste capítulo serão apresentados os métodos de compressão propostos nos últimos anos, além de dois métodos de limiar muito utilizados.

4.1 COMPRESSÃO DE SINAIS

A transformada wavelet apresenta-se como uma ferramenta alternativa para o processamento de sinais, mudando o paradigma de representação do mesmo, ao utilizar funções base de suporte compacto (em vez das senoidais de Fourier) para transformar um sinal para o espaço de tempo-escala.

No caso do processamento de imagens, um dos pontos fortes da transformada wavelet vem do fato que com uma quantidade reduzida de coeficientes (em uma escala menor da imagem) é possível representar a imagem toda sem perdas relativas na maioria dos casos. Esses coeficientes devem ser aproveitados na montagem do vetor de características para representação de uma imagem.

Existem duas categorias básicas de técnicas de compressão. A primeira é a compressão sem perdas (*lossless*). Os métodos típicos *lossless* são: compressão Huffman , compressão LZW, compressão aritmética ou compressão *run-length* (WU; OTOO; SHOSHANI,

2006). Combinações destas técnicas são utilizadas em programas de compressão *lossless* populares, por exemplo o tipo que produz arquivos "*.zip*". Infelizmente, as taxas de compressão que podem ser obtidas através de métodos *lossless* raramente são mais de 2:1 para arquivos de áudio, que consistem em música ou fala. A segunda categoria é a compressão com perdas (*lossy*). Um método de compressão *lossy* é aquele que produz imprecisões no sinal descomprimido. Técnicas *lossy* são utilizadas quando estas imprecisões são tão pequenas que se tornam imperceptíveis. A vantagem da técnica *lossy* sobre a *lossless* é que proporções de compressão muito mais elevadas podem ser atingidas. Com o método de compressão wavelet, que é *lossy*, se estamos dispostos a aceitar as pequenas imprecisões no sinal descomprimido, então podemos obter taxas de compressão de $10:1$, ou $20:1$, ou muito elevada como $50:1$ ou até mesmo $100:1$ (WALKER, 2008).

4.1.1 Método de compressão usando a transformada wavelet

Passo 1: Decomponha o sinal com a transformada wavelet.

Passo 2: Defina igual a 0 todos os coeficientes wavelet que são insignificantes, ou seja, aqueles cujos valores absolutos se encontram abaixo de algum valor limiar.

Passo 3: Preserve apenas os coeficientes significativos, coeficientes diferentes de zero a partir do Passo 2. Este deve ser um conjunto de dados muito menor do que o sinal original.

Passo 4: Na extremidade receptora, realizar a transformada wavelet inversa dos dados transmitidos no passo 3, atribuindo zero para os coeficientes não significantes que não foram transmitidos. Este passo de descompressão produz uma aproximação do sinal original.

4.1.2 Trabalhos desenvolvidos

Dentre os métodos de compressão existentes, muitos foram adaptados, trazendo assim novas técnicas de compressão. A seguir, uma revisão na literatura traz métodos estudados recentemente que trabalham com compressão de sinais utilizando a transformada wavelet.

Leite (2011), diz em seu trabalho que imagens codificadas com os padrões atualmente em estado-da-arte apresentam artefatos visuais característicos, como efeito de bloco e *ringing* (falsas bordas). Para contornar a inabilidade das transformadas ortogonais em lidar com a geometria, é proposto na literatura o uso de dicionários *wedgelet* e da decomposição *cartoon*-textura. O autor propõe um método de codificação híbrido *wedgelet*-wavelet inédito que preserva componentes de *cartoon* e textura, superando em qualidade visual ao

uso de dicionários isolados e se aproximando do desempenho de sistemas de codificação completos, tais como o padrão JPEG 2000.

Gusmão (2002), propõe um algoritmo de compressão de imagens implementado especialmente para aplicações onde há uma maior necessidade de proteção contra erros durante a transmissão. O algoritmo proposto apresenta certa robustez a erro de bits de transmissão. Ele é uma versão modificada do algoritmo REZW (*Robust Embedded Zerotrees Wavelet*), em que a codificação aritmética é substituída por codificação de Huffman, com a vantagem de redução de complexidade. Sua robustez é alcançada pela divisão dos coeficientes wavelets em diferentes sequências que são quantizadas e codificadas independentemente antes da transmissão. Deste modo, um erro de bit em uma das sequências não afeta as outras sequências, permitindo que mais informação correta chegue ao decodificador.

Figueredo (2008), em seu trabalho levantou a hipótese de que redes neurais artificiais podem ser utilizadas na compressão de eletrocardiograma (ECG) em conjunto com as transformadas wavelet. Em sua abordagem, as transformadas foram responsáveis por ressaltar os padrões do ECG no domínio tempo-frequência. Esses padrões, por sua vez, foram comprimidos por uma rede neural MLP treinada com algoritmo de retro-propagação. O processo obteve bons resultados no quesito de qualidade.

Agulhari (2009), também apresenta um método de compressão de ECG. O método, chamado *Run Length Encoding Adaptativo* (RLEA), é baseado nas transformadas wavelet e consiste basicamente em utilizar uma função wavelet otimizada que se ajuste ao sinal a ser comprimido. Após a resolução do problema de otimização é aplicado o procedimento de decomposição wavelet no sinal e os coeficientes mais significativos são retidos, sendo que o número de coeficientes retidos é determinado de forma a satisfazer uma medida de distorção pré-especificada. Os coeficientes retidos são então quantizados e compactados. Tanto os valores dos coeficientes retidos quanto o bitmap são codificados utilizando uma variante do método *Run Length Encoding* (RLE).

Chowdhury e Khatun (2012), propuseram um sistema de compressão de imagem novo e muito eficiente, com base na TWD que resulta em menor complexidade computacional sem comprometer a qualidade da imagem. Com a técnica proposta, o primeiro passo é escolher a TW e um nível N, em seguida calcular a wavelet e decompor os sinais ao nível N. No segundo passo, para cada nível de 1 a N um limar é selecionado e o limiar rígido é aplicado aos coeficientes de detalhe. Por fim, calcula-se a reconstrução wavelet usando os coeficientes de aproximação original do nível N e os coeficientes de detalhe modificados dos níveis de 1 a N.

Kekre, Sarode e Natu (2013), propõem a compressão de imagens usando transformadas wavelet ortogonais de Walsh, Cosseno, Haar, Kekre, Slant e Seno. Uma transformada wavelet de tamanho $N^2 \times N^2$ é gerada usando a sua transformada ortogonal

correspondente de tamanho $N \times N$. Estas transformadas wavelet são aplicadas sobre canais R, G e B de imagens coloridas (de tamanho $256 \times 256 \times 3$) separadamente. Em cada transformada, linhas/colunas são classificadas em sua ordem decrescente de energia, e menores coeficientes de energia são eliminados para comprimir a imagem. Este procedimento é repetido para diferentes taxas de compressão e em cada caso é reconstruída a imagem.

4.2 MÉTODOS DE LIMIAR WAVELET

Em Donoho e Johnstone (1994) são apresentados dois métodos de compressão por limiar. Esses métodos são os mais utilizados na literatura quando se trata de compressão de imagens, e são também os mais antigos. São eles o limiar rígido (*hard thresholding*) e o limiar suave (*soft thresholding*).

Nos dois métodos, o limiar usado é o limiar universal proposto por Donoho e Johnstone (1994), definido na equação (14):

$$\lambda = \sigma\sqrt{2log_{10}(N)}, \tag{14}$$

sendo σ a estimativa ou perfil do ruído presente no sinal, conforme a equação (15).

$$\sigma = \frac{mediana(|\hat{y}|)}{0,6745}. \tag{15}$$

onde $\hat{y}$ é o sinal transformado.

O limiar rígido é aplicado nos coeficientes do sinal ou imagem transformada, $\hat{y}$, conforme a equação (16), obtendo a saída y_{thr} , ou seja, assume-se que os coeficientes wavelet com valores absolutos menores que o limiar λ são componentes ruidosos, ficando o sinal bem descrito pelos coeficientes wavelets maiores que o limiar. Assim, os coeficientes menores que o limiar são eliminados (SOARES et al., 2008).

$$y_{thr} = \begin{cases} \hat{y} & \text{se } |\hat{y}| > \lambda \\ 0 & \text{se } |\hat{y}| \leq \lambda \end{cases}. \tag{16}$$

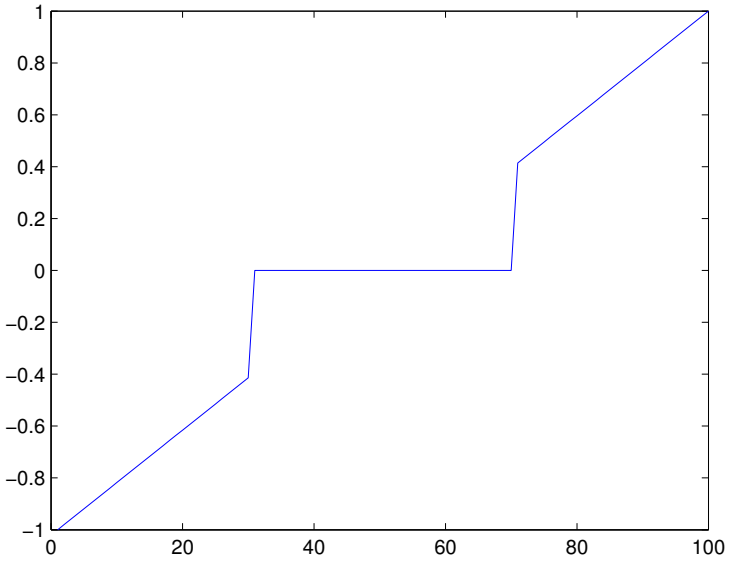
No limiar suave assume-se que os componentes ruidosos são distribuídos igualmente em todos os coeficientes wavelets, assim todos são reduzidos pelo limiar (SOARES et al., 2008). Os coeficientes que têm valores absolutos menores que um dado $\lambda > 0$ são eliminados da mesma forma que na equação (16). Coeficientes que estão acima de λ são diminuídos em sua magnitude, procurando evitar a descontinuidade apresentada pelo limiar rígido(SOARES, 2009).

A equação (17) apresenta a função de transferência do limiar suave:

$$y_{thr} = \begin{cases} sgn(\hat{y})(|\hat{y} - \lambda|) & \text{se } |\hat{y}| > \lambda \\ 0 & \text{se } |\hat{y}| \leq \lambda \end{cases}. \tag{17}$$

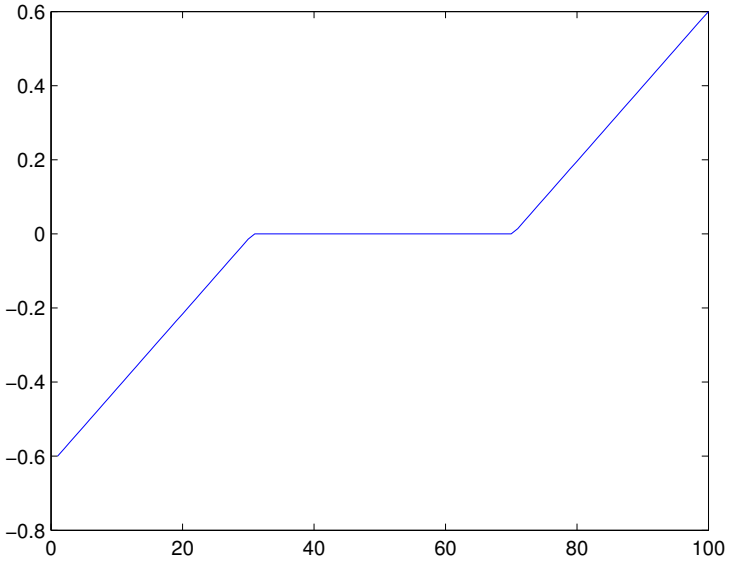
Os gráficos das funções (16) e (17) são apresentados nas Figuras 10 e 11, respectivamente. O gráfico do sinal original é dado na Figura 12, e o pseudocódigo para geração dos gráficos é dado na Figura 13.

Figura 10 – Limiar rígido.



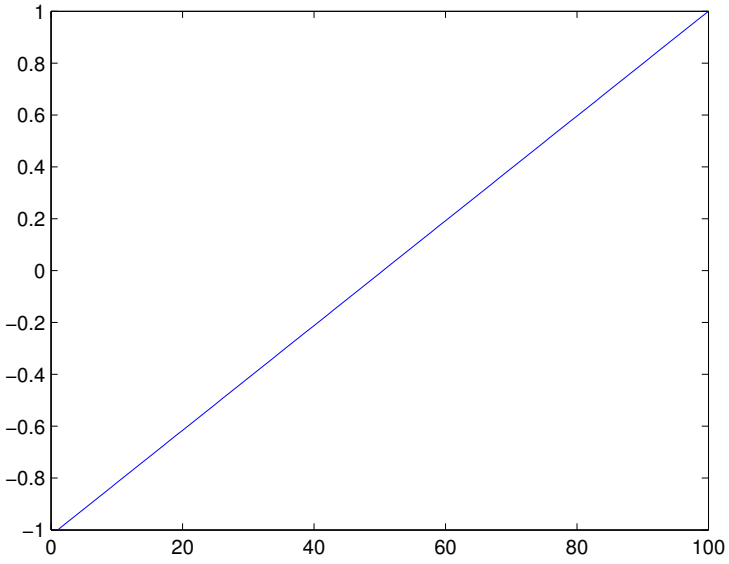
Fonte: Adaptado de Soares (2009).

Figura 11 – Limiar suave.



Fonte: Adaptado de Soares (2009).

Figura 12 – Gráfico do sinal de entrada.



Fonte: Elaboração da própria autora.

Figura 13 – Pseudocódigo do Matlab para gerar as Figuras 10 a 12.

```
% Sinal de entrada
y = linspace(-1,1,100);
% λ
thr = 0.4;
% Limiar hard
ythard = wthresh(y,'h',thr);
% Limiar soft
ytsoft = wthresh(y,'s',thr);
% Plota o grafico do limiar hard
plot(ythard)
% Plota o grafico do limiar soft
plot(ytsoft)
```

Fonte: Elaboração da própria autora.

Após ser decomposta pela transformada wavelet bidimensional, qualquer operação que se faça numa imagem deve considerar suas linhas ou suas colunas, ou seja, sendo a imagem representada por uma matriz do tipo $m \times n$ existirão $m \times n$ sinais unidimensionais (STOLLNITZ; DEROSE; SALESIN, 1995).

Assim, um método de limiar deve ser aplicado nas linhas e nas colunas da matriz que representam a imagem, de forma separada. Para cada linha, independente do método aplicado, um valor de limiar deve ser calculado.

Os métodos que "zeram" os coeficientes com valores absolutos menores que um determinado limiar são os mais indicados para compressão de imagens, ou de sinais em geral, enquanto que os métodos que apenas atenuam esses coeficientes são mais indicados para redução de ruído (DUARTE, 2005).

5 ARMAZENAMENTO DE IMAGENS COMPRIMIDAS

Descrevemos agora dois métodos de compressão de imagens, o primeiro proposto por Silva (2008) e o segundo por (SILVA; DUARTE; VILLARREAL, 2011). Tais métodos servirão de base para a construção do algoritmo proposto neste trabalho. Nos dois sistemas de armazenamento de imagens comprimidas acima mencionados, a transformada wavelet bidimensional foi implementada usando a função wavelet de Daubechies 6 (`db6`), processada através do MATLAB®. O limiar usado foi o limiar rígido.

5.1 MÉTODO DE SILVA (2008)

A função wavelet `db6` foi usada porque apresenta um número de momentos nulos suficiente para evidenciar detalhes da imagem e para propiciar uma boa reconstrução da mesma (DAUBECHIES, 1992). A escolha em utilizar o limiar rígido, foi feita devido ao comportamento de tal limiar, que quando aplicado em uma imagem, os valores de seus coeficientes não sofrem variações irreversíveis, ou seja, os valores dos coeficientes são zerados ou mantidos.

Para os testes de ambos os métodos, foi usado um conjunto de cinco imagens, sendo três de 256×256 pixels (Flor, Meninos e Usina), e duas de 128×128 pixels (Mão e Ultra), todas de propriedade dos autores.

O objetivo do trabalho desenvolvido por Silva (2008) foi a implementação de um algoritmo que fizesse a compressão da imagem decomposta pela transformada wavelet e que identificasse os coeficientes que não foram eliminados pelo método de limiar rígido, e os armazenasse, com informações suficientes para que um sistema de descompressão fosse capaz de reconstruir a imagem usando apenas estes coeficientes.

Silva (2008) propôs a criação de um vetor que armazena os coeficientes significativos da imagem, e isso permite que a mesma seja reconstruída. O vetor criado é denominado V_cod e contém o valor de cada coeficiente não eliminado pelo método de limiar, além da dimensão da imagem e sua posição na imagem original.

As imagens processadas por este método são imagens de dimensão quadrada de potência de 2, ou seja, imagens do tipo $m \times m$, sendo $m = 2^j$, onde j o número máximo de níveis de resolução no qual a transformada wavelet decomporá a imagem.

O primeiro componente de V_cod é a dimensão da matriz que representa a imagem. A partir do segundo componente, os coeficientes já podem ser armazenados, sendo que cada coeficiente deverá trazer consigo a informação de qual linha e qual coluna ele está localizado. A seguir, é feita uma ilustração do funcionamento desse sistema de armazenamento para uma matriz quadrada de ordem 4 (SILVA, 2008).

Seja A, a matriz original, $\hat{A}$ sua versão decomposta pela transformada wavelet discreta (TWD) como em (18) e A_L a matriz após aplicação do limiar escolhido, mostrado em (19).

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \overset{TWD}{\rightarrow} \hat{A} = \begin{bmatrix} b_{11} & c_{12} & b_{13} & c_{14} \\ b_{21} & c_{22} & b_{23} & c_{24} \\ b_{31} & c_{32} & c_{33} & c_{34} \\ b_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix} \tag{18}$$

Em $\hat{A}$, os coeficientes representados por b_{ij} são coeficientes significantes, enquanto que os que estão representados por c_{ij} podem ser eliminados. Aplicando o limiar rígido, a matriz A_L resultante é a seguinte:

$$A_L = \begin{bmatrix} b_{11} & 0 & b_{13} & 0 \\ b_{21} & 0 & b_{23} & 0 \\ b_{31} & 0 & 0 & 0 \\ b_{41} & 0 & 0 & 0 \end{bmatrix}. \tag{19}$$

A matriz A_L é uma matriz esparsa, com poucos coeficientes representativos. Por isso, só há a necessidade de armazenar tais coeficientes. Assim, o vetor que representa a matriz comprimida, observando as linhas da matriz A_L é dado em (20):

$$V\_cod = [4 \quad b_{11} \quad 1 \quad 1 \quad b_{13} \quad 1 \quad 3 \quad b_{21} \quad 2 \quad 1 \quad b_{23} \quad 2 \quad 3 \quad b_{31} \quad 3 \quad 1 \quad b_{41} \quad 4 \quad 1], \tag{20}$$

sendo que o número 4 na primeira posição indica a dimensão da matriz. Após o valor do primeiro elemento, b_{11}, os números 1 e 1 indicam que este coeficiente se encontra na linha 1 e na coluna 1 e, assim por diante. Assim, para uma matriz transformada e comprimida de ordem m, o vetor V_cod é escrito, de modo geral, como em (21):

$$V\_cod = [m \quad b_{i_1 j_1} \quad i_1 \quad j_1 \quad b_{i_2 j_2} \quad i_2 \quad j_2 \quad \ldots \quad b_{i_r j_r} \quad i_r \quad j_r] \tag{21}$$

onde i_r e j_r representam linha e coluna, respectivamente, em que se encontra o elemento $b_{i_r j_r}$, com $r \in \mathbb{N}$.

Portanto, não será necessária uma matriz quadrada para representar a matriz comprimida, apenas um vetor.

O vetor V_cod pode ser gerado na mesma rotina que faz a eliminação dos coeficientes com valores absolutos abaixo do limiar, pois nesta rotina há a identificação dos coeficientes que estão acima do limiar. O vetor V_cod é a versão compactada da matriz no domínio wavelet. O pseudocódigo da rotina para sua construção é apresentado na Figura 14.

Figura 14 – Pseudocódigo para a criação do vetor *V_cod*.

```
%Procedimento compactacao de matriz (m: matriz mxm, λ, V_cod)

DWT(matriz)
V_cod ← []
V_cod(1) ← m
Para i de 1 ate m
Para j de 1 ate m
Se |xt(i,j)|> λ entao
V_cod ← [V_cod xt(i,j) i j]
Senao
Limiar(xt(i,j))
Fim se
Fim para
Fim para
Fim do procedimento
```

Fonte: Silva (2008).

O procedimento de reconstrução da matriz, a partir do vetor *V_cod*, também é simples. A ordem da matriz é o valor do primeiro elemento do vetor. Portanto, o primeiro passo é a criação de uma matriz quadrada nula cuja ordem seja igual ao primeiro elemento do vetor *V_cod*. A partir do segundo elemento, cada três elementos representam, respectivamente, o valor do coeficiente preservado na matriz transformada, sua linha e sua coluna de localização, então basta usar esses três valores para localizar e posicionar o coeficiente no seu lugar na matriz nula criada inicialmente. Após a alocação de todos os elementos na matriz, a transformada wavelet discreta inversa (TWDI) é aplicada nesta matriz (SILVA, 2008).

O pseudocódigo para a reconstrução da matriz transformada a partir do vetor *V_cod* é apresentado na Figura 15.

Figura 15 – Pseudocódigo para a reconstrução do vetor *V_cod*.

```
%Procedimento decodificao (V_cod - vetor [1 x k])

m = V_cod(1)
T = matriz_nula(mxm)
X = V_cod[2 ... K]
l = comprimento(X)
Para i de 1, passo 3, ate l
T(X(i+1), X(i+2)) = X(i)
Fim para
mr = idwt(T)
Fim do procedimento
```

Fonte: Silva (2008).

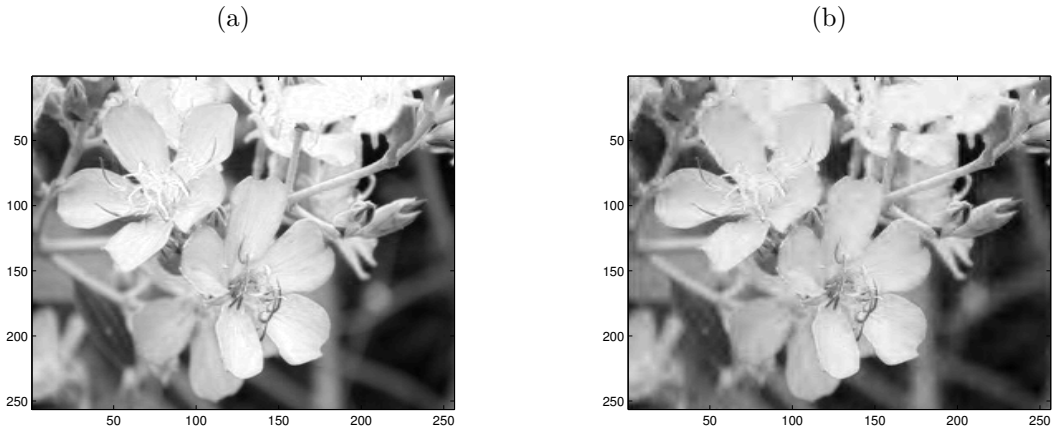
Para verificar a eficiência do método de armazenamento, os únicos valores que devem ser observados são: o tamanho original da imagem, em *kBytes*, o tamanho do vetor de codificação da mesma, em *kBytes*, e a redução em porcentagem, quando se passa da matriz original para o seu vetor de codificação. Resultados para esse método são apresentados na Tabela 1 e nas Figuras 16 a 20.

Tabela 1 – Imagens comprimidas e decodificadas usando o sistema de armazenamento de Silva (2008).

Imagens	**Taxa de Compressão (%)**	**Tamanho original (kBytes)**	**Tamanho codificado (kBytes)**	**Redução (%)**
Flor	86,34	524,29	214,81	59,03
Mão	86,85	131,07	51,73	60,53
Meninos	92,24	524,29	122,00	76,73
Ultra	86,69	131,07	52,35	60,05
Usina	89,38	524,29	166,95	68,15

Fonte: Silva (2008).

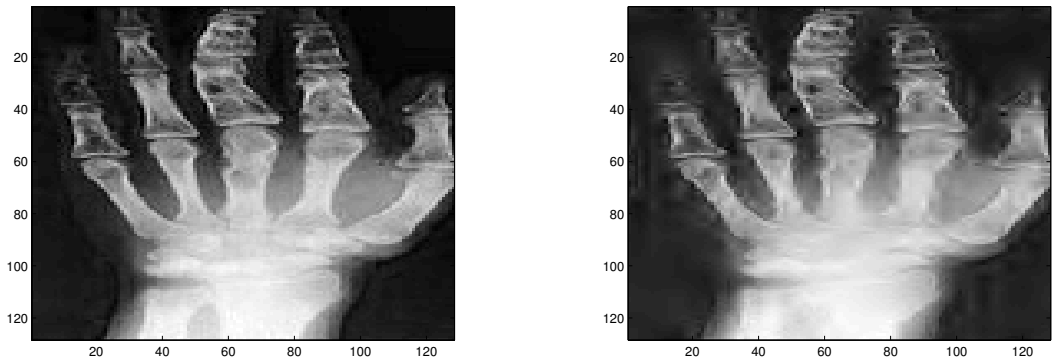
Figura 16 – (a) Flor original (b) Flor decodificada pelo método de Silva (2008).



Fonte: Silva (2008).

Figura 17 – (a) Mão original (b) Mão decodificada pelo método de Silva (2008).

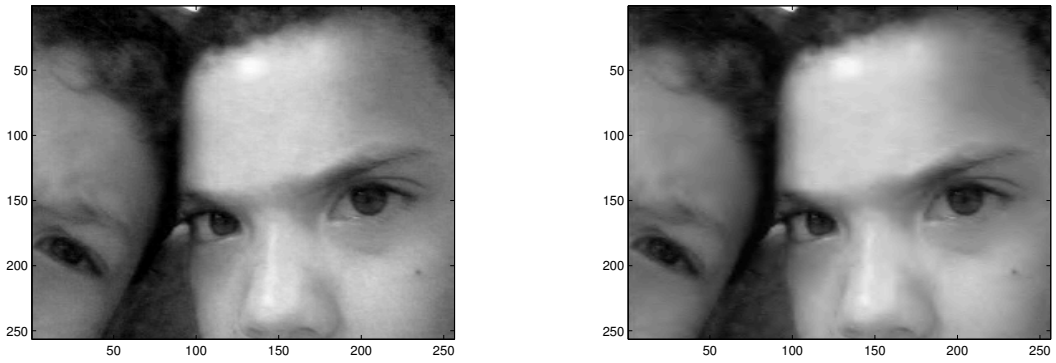
(a) (b)



Fonte: Silva (2008).

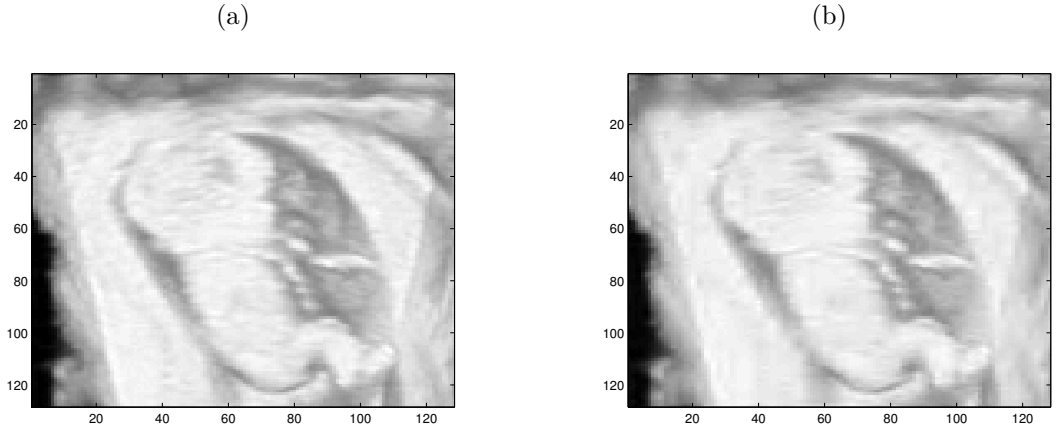
Figura 18 – (a) Meninos original (b) Meninos decodificada pelo método de Silva (2008).

(a) (b)



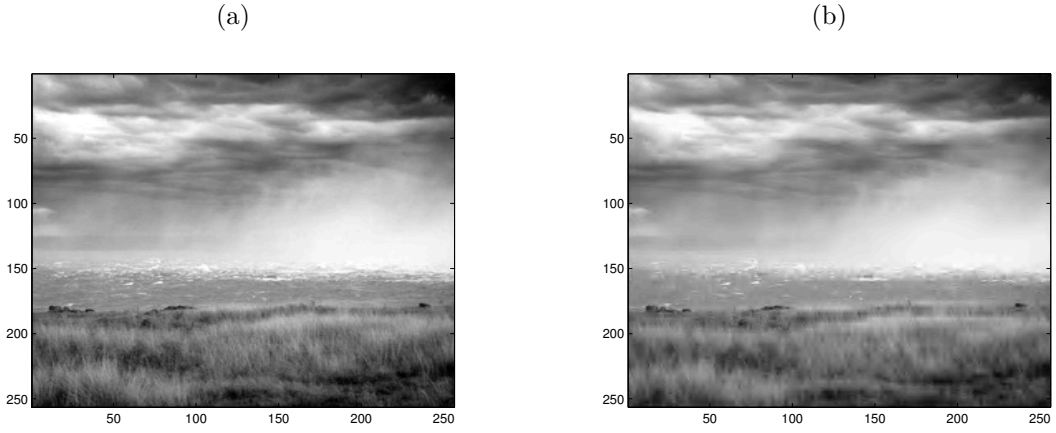
Fonte: Silva (2008).

Figura 19 – (a) Ultra original (b) Ultra decodificada pelo método de Silva (2008).



Fonte: Silva (2008).

Figura 20 – (a) Usina original (b) Usina decodificada pelo método de Silva (2008).



Fonte: Silva (2008).

Conforme se pode observar na Tabela 1, o conjunto de imagens originais tem um total de 1835,01 *kBytes*. Após o processamento ele passa a ter um total de 607,84 *kBytes*, representando uma redução total de 64,89% do espaço necessário para o seu armazenamento. Note que, o conjunto de imagens obteve uma taxa de compressão superior a 86%.

Esse sistema de armazenamento foi implementado, usando a linguagem de programação do MATLAB®, num conjunto de cinco imagens e apresentou uma redução superior

a 50% do espaço de armazenamento necessário para qualquer uma das imagens. Sendo que a reconstrução dessas imagens não foi comprometida. Portanto, pode-se concluir que o sistema de armazenamento de imagens comprimidas de Silva (2008) é eficiente e possui como pontos fortes a facilidade de implementação e a reconstrução das imagens com alta taxa de compressão e boa resolução.

5.2 MÉTODO DE SILVA, DUARTE E VILLARREAL (2011)

O segundo método de compressão é apresentado por Silva, Duarte e Villarreal (2011), e também tem por objetivo a economia do espaço computacional necessário para o armazenamento, fazendo uma codificação com o mínimo possível de informações, porém suficientes para a decodificação da imagem. O algoritmo consiste em uma evolução do método de Silva (2008) apresentado anteriormente.

O segundo algoritmo propõe um método de codificação de imagens digitais que também passam a ser representadas em um vetor de codificação, porém com menos informações que o anterior. Nessa proposta, a composição do vetor de codificação é muito parecida com a primeira, porém os valores de i e j que representam, respectivamente, linha e coluna de localização de um determinado elemento, passam a ser representados por um único valor. Em relação a Silva (2008), isto causa uma redução de praticamente um terço do espaço computacional necessário para o armazenamento da imagem comprimida. Os passos dessa proposta são dados a seguir:

Passo 1: Criar uma matriz A_1 onde cada elemento é igual a soma da linha com a coluna em que ele se encontra. Como exemplo, apresenta-se também uma matriz 4×4 definida em (22).

$$A_1 = (a_{ij})_m, \quad a_{ij} = i + j \tag{22}$$

$$A_1 = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \quad \Rightarrow \quad A_1 = \begin{bmatrix} 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

Percebe-se que A_1 é uma matriz simétrica, isto é, $a_{ij} = a_{ji}$, causando um problema de similaridade entre seus elementos. Este problema é contornado no Passo 2.

Passo 2: Adicionar 1 aos elementos da primeira linha da matriz $A_1 = (a_{ij})_m$ através da equação (23):

$$a_{ij}^2 = (1 + j) + 1, \quad i = j = 1, 2, \ldots, m \tag{23}$$

$$A_2 = \begin{bmatrix} 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \\ 5 & 6 & 7 & 8 \end{bmatrix}.$$

Passo 3: A partir da segunda linha, cada linha será igual a anterior somada com a ordem m de A em cada um de seus elementos, conforme equação (24).

$$a^3_{(i+1)j} = a^3_{ij} + m, \quad i = j = 1, 2, \ldots, m \tag{24}$$

$$A_3 = \begin{bmatrix} 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 \\ 15 & 16 & 17 & 18 \end{bmatrix}.$$

A matriz ordenada criada com números naturais distintos entre si obedece a duas situações: o último elemento de A_3 deverá obedecer a equação (25):

$$a^3_{mm} = m^2 + 2, \tag{25}$$

e o primeiro elemento, independente da dimensão da imagem, será sempre igual a 3.

$$a^3_{11} = 3$$

Cada elemento da matriz A_3 é chamado de código matricial dos *pixels* da imagem.

Passo 4: Representar o vetor de codificação com o código matricial da imagem.

$$V\_cod\_mat = \begin{bmatrix} m & b_{i_1 j_1} & a^3_{i_1 j_1} & b_{i_2 j_2} & a^3_{i_2 j_2} & \ldots & b_{i_r j_r} & a^3_{i_r j_r} \end{bmatrix}. \tag{26}$$

Desta forma, na equação (26), tem-se como primeiro elemento do vetor V_cod_mat a ordem m da imagem e, a partir do segundo elemento, a cada dois elementos, o primeiro representa o valor do pixel da imagem e o segundo seu código matricial.

Para a decodificação temos os seguintes passos:

Passo 1: Criar uma matriz nula T de ordem m;

$$T = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_m$$

Passo 2: Criar uma matriz $A' = (a'_{ij})_m$ igual a matriz A_3 criada no processo de codificação;

$$A' = \begin{bmatrix} 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 \\ 15 & 16 & 17 & 18 \end{bmatrix}. \tag{27}$$

Passo 3: Para cada elemento a^3_{ij} do vetor (26), identificar o seu respectivo valor a'_{ij}. O elemento b_{ij} será alocado na matriz T na mesma posição em que a'_{ij} se encontra em A'.

Passo 4: Aplicar a transformada wavelet inversa na matriz T obtendo a imagem reconstruída.

Para verificar a eficiência do método de Silva, Duarte e Villarreal (2011), e também para fins de comparação com o método de Silva (2008), os únicos valores observados foram: o tamanho, em *kBytes*, das imagens original e codificada e a redução, em porcentagem, quando se passa da matriz original para a codificada em ambos métodos. As imagens usadas para os testes e suas versões processadas após a decodificação são apresentadas nas Figuras 21 a 25, e nas Tabelas 2 e 3 estão os valores numéricos citados anteriormente.

Tabela 2 – Imagens comprimidas e decodificadas usando o sistema de armazenamento de Silva, Duarte e Villarreal (2011).

Imagens	**Taxa de Compressão (%)**	**Tamanho original (kBytes)**	**Tamanho codificado (kBytes)**	**Redução (%)**
Flor	86,34	524,29	143,21	72,68
Mão	86,85	131,07	34,49	73,68
Meninos	92,24	524,29	81,34	84,48
Ultra	86,69	131,07	34,90	73,37
Usina	89,38	524,29	111,30	78,77

Fonte: Silva, Duarte e Villarreal (2011).

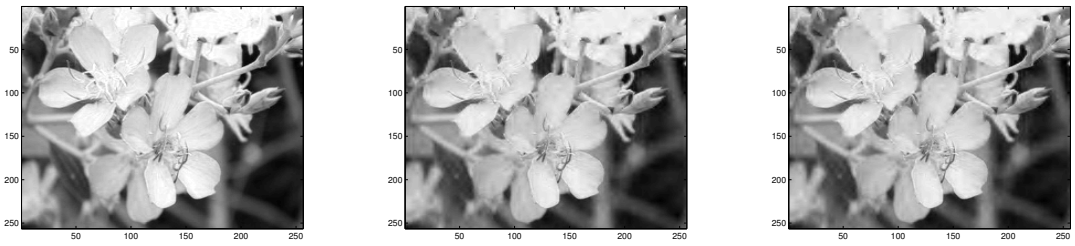
Tabela 3 – Comparação entre os espaços requeridos para armazenamento das imagens codificadas por Silva (2008) e por Silva, Duarte e Villarreal (2011).

Imagens	**Silva, 2008 (kBytes)**	**Silva; Duarte; Villarreal, 2011 (kBytes)**	**Redução (%)**
Flor	214,81	143,21	33,27
Mão	51,73	34,49	33,33
Meninos	122,00	81,34	33,33
Ultra	52,35	34,90	33,34
Usina	166,95	111,30	33,34

Fonte: Silva, Duarte e Villarreal (2011).

Figura 21 – (a) Flor original (b) Flor decodificada pelo método de Silva (2008) (c) Flor decodificada pelo método de Silva, Duarte e Villarreal (2011).

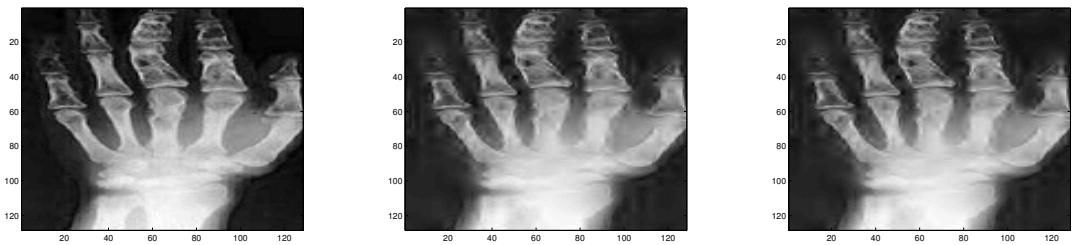
(a) (b) (c)



Fonte: Silva, Duarte e Villarreal (2011).

Figura 22 – (a) Mão original (b) Mão decodificada pelo método de Silva (2008) (c) Mão decodificada pelo método de Silva, Duarte e Villarreal (2011).

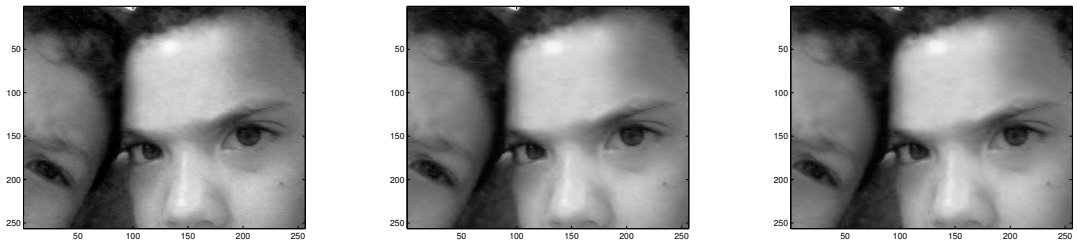
(a) (b) (c)



Fonte: Silva, Duarte e Villarreal (2011).

Figura 23 – (a) Meninos original (b) Meninos decodificada pelo método de Silva (2008) (c) Meninos decodificada pelo método de Silva, Duarte e Villarreal (2011).

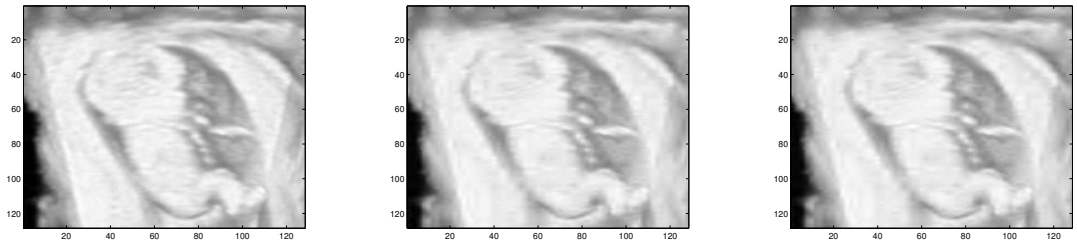
(a) (b) (c)



Fonte: Silva, Duarte e Villarreal (2011).

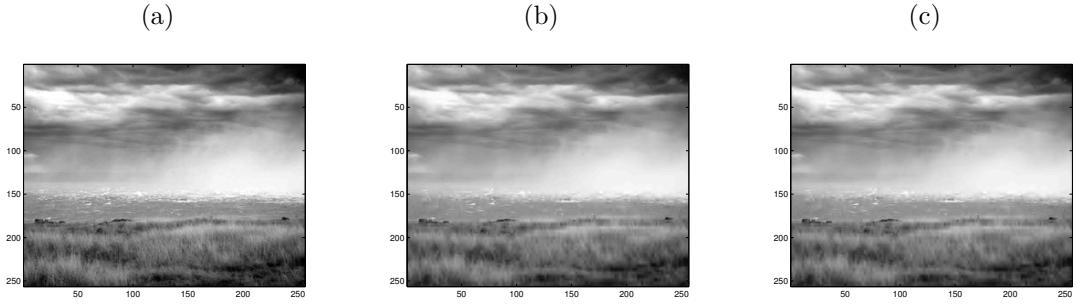
Figura 24 – (a) Ultra original (b) Ultra decodificada pelo método de Silva (2008) (c) Ultra decodificada pelo método de Silva, Duarte e Villarreal (2011).

(a) (b) (c)



Fonte: Silva, Duarte e Villarreal (2011).

Figura 25 – (a) Usina original (b) Usina decodificada pelo método de Silva (2008) (c) Usina decodificada pelo método de Silva, Duarte e Villarreal (2011).



Fonte: Silva, Duarte e Villarreal (2011).

De acordo com a Tabela 2, utilizando o método de Silva, Duarte e Villarreal (2011), o conjunto de imagens originais que tem um total de 1835,01 *kBytes*, após a codificação, passa a ter um total de 405,24 *kBytes*. A redução do espaço utilizado para armazenamento é de 77,9%, se comparado com o total ocupado pelas imagens originais.

Em resumo, o sistema de Silva, Duarte e Villarreal (2011) reduz em mais de 70% o espaço de armazenamento da imagem comprimida e reduz em média 33% o número de elementos necessários para a codificação de uma imagem comprimida por meio da transformada wavelet. Esses valores referem-se ao conjunto de imagens utilizadas para os testes.

6 MÉTODO PROPOSTO

6.1 INTRODUÇÃO

O método proposto neste trabalho é uma evolução dos algoritmos propostos por Silva (2008) e Silva, Duarte e Villarreal (2011), apresentados no capítulo anterior. Para a sua implementação, também serão usados a função `db6`, o limiar rígido, e para facilitar a programação foi utilizado um pacote desenvolvido pela Universidade Rice, que é disponibilizado através do site <http://dsp.rice.edu/software/rice-wavelet-toolbox>.

Os objetivos pretendidos, e conseguidos, do método proposto estão elencados abaixo. Destes, o mais importante consiste em reduzir ainda mais o espaço de armazenamento da imagem comprimida, além de reduzir o número de elementos necessários para a sua codificação, fazendo com que esses elementos ainda que poucos, sejam suficientes para a decodificação da imagem.

1. Redução dos vetores V_cod e V_cod_mat, com o objetivo de armazenar o mínimo de informação possível sobre uma imagem.
2. Determinação da taxa de compressão a priori.
3. Redução do tempo de processamento.
4. Processamento de imagens coloridas, usando a ferramenta `wavemenu` do MATLAB®.

O método de Silva (2008), apresentado no capítulo anterior, propôs a construção de $V\_cod = [m \quad b_{i_1 j_1} \quad i_1 \quad j_1 \quad b_{i_2 j_2} \quad i_2 \quad j_2 \quad \ldots \quad b_{i_r j_r} \quad i_r \quad j_r]$, que em seguida teve seu tamanho diminuído, tornando-se o vetor apresentado por Silva, Duarte e Villarreal (2011), a saber, $V\_cod\_mat = \begin{bmatrix} m & b_{i_1 j_1} & a^3_{i_1 j_1} & b_{i_2 j_2} & a^3_{i_2 j_2} & \ldots & b_{i_r j_r} & a^3_{i_r j_r} \end{bmatrix}$.

Uma grande contribuição do método que será proposto neste trabalho, consiste na descoberta de uma função que resume os passos para a criação da matriz A. A regra de correspondência dessa função é dada em (28).

$$\begin{aligned} N : D_m = \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\} &\to I_m = \{3, 4, \ldots, m^2 + 2\} \\ (i, j) &\mapsto N(i, j) = a^3_{ij} = -(m - 2) + mi + j \end{aligned} \tag{28}$$

Note que N é uma função que associa, a cada par $(i, j) \in D_m$, um único $a^3_{ij} \in I_m$. Reciprocamente, cada $a^3_{ij} \in I_m$ é o correspondente de um único $(i, j) \in D_m$. Desse modo, a função N é invertível de D_m sobre I_m. Sua regra de correspondência é dada em (29) e

sua inversa é a função dada pela equação (30). A demonstração desse fato e o cálculo da inversa da função N são apresentados no Apêndice A.

$$\begin{array}{rl} M : \{3, 4, \ldots, m^2 + 2\} & \to \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\} \\ a_{ij}^3 & \mapsto M(a_{ij}^3) \end{array}, \tag{29}$$

onde

$$M(a_{ij}^3) = \left(\frac{a_{ij}^3 - 3 - \texttt{mod}(a_{ij}^3 - 3, m)}{m} + 1, \texttt{mod}(a_{ij}^3 - 3, m) + 1 \right), \tag{30}$$

e para $a, b \in \mathbb{R}$, $\texttt{mod}(a, b)$ retorna o resto da divisão de a por b.

Descobrir a lei que define N, e consequentemente sua inversa M, possibilitou que o tempo de decodificação fosse encurtado significativamente, como será mostrado mais adiante.

O método proposto consiste na criação de um novo vetor, denominado *V_cod_prop* que reduzirá os outros dois vetores já criados, da seguinte forma:

Passo 1: A primeira posição do vetor será ocupada pela dimensão m da matriz original, como feito anteriormente.

Passo 2: As posições que se seguem serão ocupadas por um único número que será calculado em função do valor do pixel transformado da imagem (a saber, b_{ij}), de i e de j. A função que relaciona i e j é dada por N, logo, precisamos de uma nova função f capaz de relacionar b_{ij}, i e j de tal modo que $f(b_{ij}, i, j) = f(b_{ij}, a_{ij}^3)$. Tal função é dada pela equação (31), cujo processo de construção está descrito no Apêndice B:

$$f(b_{ij}, a_{ij}^3) = \texttt{sign}(b_{ij})(\log_{m^5} |b_{ij}| + a_{ij}^3) = k_r \quad r = 1, \ldots, R. \tag{31}$$

onde R é o número de elementos conservados na compressão.

Note que m é a dimensão da imagem, o que torna m^5 um número grande, fazendo com que $\log_{m^5} b_{ij}$ seja um número pequeno, com valor absoluto entre zero e um. $\texttt{sign}(x)$ é a função sinal do MATLAB® que retorna 1, -1 ou 0, se o argumento for positivo, negativo ou nulo, respectivamente. Como no caso deste método $b_{ij} \neq 0$, $\texttt{sign}(b_{ij})$ sempre retornará 1 ou -1. Sendo assim, o vetor proposto é descrito em (32):

$$V\_cod\_prop = [m \quad k_1 \quad k_2 \quad k_3 \quad \ldots \quad k_R]\,. \tag{32}$$

Exemplo 1:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \overset{TWD}{\rightarrow} \hat{A} = \begin{bmatrix} 34,0000 & 0 & 0 & 0 \\ -8,6491 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \overset{Limiar}{\rightarrow} A_L = \begin{bmatrix} 34,0000 & 0 & 0 & 0 \\ -8,6491 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\Rightarrow V\_cod = [4 \quad 34,0000 \quad 1 \quad 1 \quad -8,6491 \quad 2 \quad 1]$$

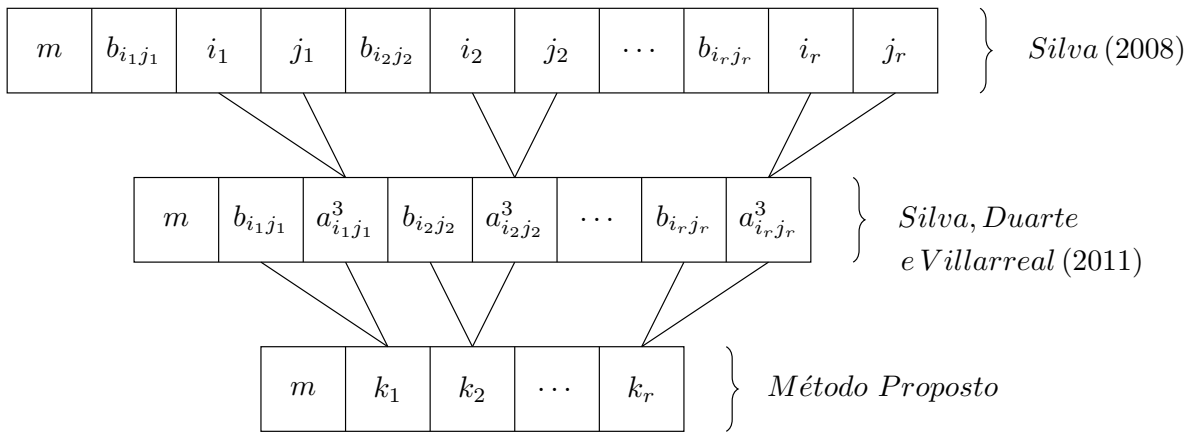
$$\Rightarrow V\_cod\_mat = [4 \quad 34,0000 \quad 3 \quad -8,6491 \quad 7]$$

$$\Rightarrow V\_cod\_prop = [4 \quad 3,5087 \quad -7,3112]$$

Usando a função N foi possível obter $a^3_{11} = 3$ e $a^3_{21} = 7$, e com f foi possível encontrar os valores $f(34,3) = 3,5087$ e $f(-8,6491,\ 7) = -7,3112$.

De forma geral pode-se resumir a evolução do primeiro método até o método proposto, de acordo com a Figura 26:

Figura 26 – Representação dos métodos estudados e do método proposto.



Fonte: Elaboração da própria autora.

O vetor V_cod_prop também pode ser gerado na rotina que faz a eliminação dos coeficientes com valores absolutos abaixo do limiar, pois nesta rotina há a identificação dos coeficientes que estão acima do limiar. Logo, o custo computacional para gerar este vetor será desprezível.

O procedimento de reconstrução da matriz, a partir do vetor V_cod_prop, acontece da seguinte maneira: A ordem da matriz é o valor do primeiro elemento do vetor. Portanto, o primeiro passo é a criação de uma matriz quadrada nula cuja ordem seja igual ao primeiro elemento de V_cod_prop.

A partir do segundo elemento, k_r representa o valor do coeficiente preservado na matriz transformada, sua linha e sua coluna de localização, então, basta usar k_r para localizar e posicionar o coeficiente no seu lugar na nova matriz nula criada inicialmente.

Através de k_r achamos a_{ij}^3 segundo a equação (33):

$$a_{ij}^3 = \texttt{round}(|k_r|). \tag{33}$$

A função $\texttt{round}(x)$ do MATLAB® arredonda x para o inteiro mais próximo. Como o logaritmo calculado na equação (31) é um número entre 0 e 1, então na equação (33) teremos exatamente o valor do código do pixel que a_{ij}^3.

O elemento b_{ij} pode ser calculado através da equação (34):

$$b_{ij} = \texttt{sign}(k_r)(m^5)^{||k_r|-\texttt{round}(|k_r|)|}. \tag{34}$$

Exemplo 2: Com o vetor obtido no Exemplo 1, e usando os resultados de (33) e (34), é possível achar o valor de b_{ij} e sua posição na matriz.

$V\_cod\_prop = [4 \quad 3,5087 \quad -7,3112]$

$\Rightarrow a_{ij}^3 = \texttt{round}(|3,5087|) = 3 \Rightarrow i = 1, \quad j = 1$

$\Rightarrow b_{ij} = \texttt{sign}(3,5087) * (4^5)^{||3,5087|-\texttt{round}(|3,5087|)|}$

$\Rightarrow b_{ij} = 1 * 1024^{|3,5087-3|} = 34,0000$

Após a localização de todos os elementos da matriz, a transformada wavelet discreta inversa (TWDI) é aplicada nesta matriz.

6.2 COMPARAÇÃO ENTRE MÉTODOS

As Tabelas 4, 5 e 6 contém os resultados obtidos pelo método proposto atuando no conjunto das cinco imagens usadas nos dois métodos discutidos anteriormente, e utilizando a mesma taxa de compressão.

Tabela 4 – Redução do espaço computacional obtida pelo Método Proposto.

Imagens	**Original (kBytes)**	**Método Proposto (kBytes)**	**Redução (%)**
Flor	524,29	71,61	86,34
Mão	131,07	17,25	86,84
Meninos	524,29	40,67	92,24
Ultra	131,07	17,46	86,68
Usina	524,29	55,66	89,38

Fonte: Elaboração da própria autora.

Tabela 5 – Comparação entre os espaços requeridos para armazenamento de imagens codificadas por Silva (2008) e pelo Método Proposto.

Imagens	**Silva, 2008 (kBytes)**	**Método Proposto (kBytes)**	**Redução (%)**
Flor	214,81	71,61	66,66
Mão	51,73	17,25	66,65
Meninos	122,00	40,67	66,66
Ultra	52,35	17,46	66,65
Usina	166,95	55,66	66,66

Fonte: Elaboração da própria autora.

Tabela 6 – Comparação entre os espaços requeridos para armazenamento de imagens codificadas por Silva, Duarte e Villarreal (2011) e pelo Método Proposto.

Imagens	**Silva; Duarte; Villarreal, 2011 (kBytes)**	**Método Proposto (kBytes)**	**Redução (%)**
Flor	143,21	71,61	50,00
Mão	34,49	17,25	49,99
Meninos	81,34	40,67	50,00
Ultra	34,90	17,46	49,97
Usina	111,30	55,66	49,99

Fonte: Elaboração da própria autora.

Comparando os espaços para armazenamento requeridos pelo método proposto e os outros dois métodos anteriores, o proposto é em média $66,66\%$ mais econômico que o de Silva, 2008, ao passo que é em média $49,99\%$ mais econômico que o de Silva; Duarte; Villarreal, 2011.

De acordo com o conjunto de imagens usado para os testes, o sistema proposto reduz em mais de 88% o espaço de armazenamento da imagem comprimida e reduz em média $66,66\%$ o número de elementos necessários para a codificação de uma imagem comprimida por meio da transformada wavelet.

6.3 RESULTADOS

6.3.1 Taxa de compressão

A forma como foi programado o método de Silva (2008) e Silva, Duarte e Villarreal (2011) não permite que a taxa de compressão seja escolhida. Linha por linha um limiar

é definido e os coeficientes são eliminados. Ao final a taxa de compressão é dada. Com o intuito de melhor tal método, Travassos, Duarte e Villarreal (2016) apresentaram um algoritmo que possibilita escolher a taxa de compressão mínima. Entretanto o método proposto permite que a taxa de compressão exata seja escolhida.

O raciocínio que possibilita a escolha da taxa de compressão é o seguinte: Um vetor recebe os coeficientes da matriz transformada, os ordena, e de acordo com a taxa de compressão determina quantos elementos devem ser eliminados para que a mesma seja atingida. Dentre os menores elementos a serem eliminados, o maior é escolhido para ser o limiar. A taxa de compressão pode ser determinada usando o pseudocódigo da Figura 27.

Figura 27 – Pseudocódigo do Matlab para determinação da taxa de compressão.

```
% O vetor coefs recebe os coeficientes da matriz contendo os pixels
coefs=reshape(xt',1,l^2);
% Comprimento do vetor coefs
dim=length(coefs);
taxa = 0.85;
% Reordenar os valores absolutos transformados (ordem crescente)
C_dec = sort(abs(coefs));
% Quantidade de numeros que serao eliminados
num_el=round(length(coefs)*taxa);
% Achar o maior numero dentre os que serao elimnados
maior = C_dec(num_el);
% Aplicar o limiar hard
coefs=wthresh(coefs,'h',maior);
% Transforma vetor em matriz novamente
xt=reshape(coefs',l,l)';
```

Fonte: Elaboração da própria autora.

6.3.2 Tempo de processamento

Os algoritmos estudados no Capítulo 5 processam imagens do tipo $m \times m$, com $m = 2^j$. Para $j = 9$ o processo de decodificação começa a tornar-se lento. Para imagens com $j \geq 10$, a decodificação pode levar horas. Isso porque é necessário trabalhar com a matriz A' criada em (27).

O método proposto reduz o tempo de processamento de decodificação da imagem, substituindo a matriz A' pela inversa da função N, a saber, a função M dada pela equação (30). A redução é bastante significativa, como mostra a Tabela 7. O pseudocódigo da Figura 28 é parte da programação da decodificação, caso não fosse usada a função inversa. Com esse código a decodificação levaria em alguns casos, até dias para rodar, variando de acordo com o processador do computador.

Figura 28 – Pseudocódigo da decodificação no Matlab sem o uso da função inversa.

```
% A := matriz cujo primeiro elemento e 3 e o ultimo, m^{2}+2
% dc := matriz de zeros de tamanho mxm
for i=3:2:length(V_cod_prop)
for j=1:m
for k=1:m
if V_cod_prop(i)== A(j,k)
dc(j,k)=V_cod_prop(i-1);
end
end
end
end
```

Fonte: Elaboração da própria autora.

Usando a função inversa, a programação acima pode ser resumida, resultando no pseudocódigo da Figura 29:

Figura 29 – Pseudocódigo da decodificação no Matlab com o uso da função inversa.

```
% dc := matriz de zeros de tamanho mxm
% d := round(abs(x(i)))
for i=3:2:length(V_cod_prop)
d=V_cod_prop(i);
dc((d-3-mod(d-3,n))/n+1,mod(d-3,n)+1)=V_cod_prop(i-1);
end
```

Fonte: Elaboração da própria autora.

A Tabela 7 apresenta a redução do tempo de decodificação para um conjunto de imagens com tamanhos variados. De acordo com as imagens utilizadas, a redução foi em média de 92,65%.

Tabela 7 – Redução em porcentagem do tempo (em segundos) de processamento da decodificação.

		Tempo de decodificação		
Imagens	**Tamanho (m)**	**Silva; Duarte; Villarreal, 2011 (segundos)**	**Método Proposto com função inversa (segundos)**	**Redução (%)**
Mão	128	0,9860	0,1870	81,03
Ultra	128	0,9560	0,2350	75,42
Meninos	256	15,0360	0,3750	97,50
Usina	256	15,2930	0,4610	96,98
Baboon	512	271,5010	5,7200	97,89
Borboleta	1024	5,4924e+03	139,1030	97,47
Peppers	1024	5,4473e+03	136,1820	97,50
Cass	2048	1,0427e+05	2,6590e+03	97,45

Fonte: Elaboração da própria autora.

A descoberta da função N não teve contribuição relevante no tempo usado para a codificação.

6.3.3 Métricas

Existem diferentes maneiras de se calcular a distorção por uma função predefinida, entre a imagem original $f(x, y)$, e a imagem reconstruída $\hat{f}(x, y)$. A medida mais utilizada é o MSE (*Mean Squared Error*), definido em (35):

$$MSE = \frac{1}{mn} \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} |f(x, y) - \hat{f}(x, y)|^2, \tag{35}$$

m é o número de linhas e n o número de colunas da imagem original e, consequentemente, da imagem reconstruída.

Na área de processamento de imagens, o MSE é normalmente utilizado no cálculo da $PSNR$, do inglês, *Peak Signal to Noise Ratio*, que é usada para avaliar a diferença global entre duas imagens, dada como na equação (36) (PEDRINI; SCHWARTZ, 2008):

$$PSNR = 10 \log_{10} \left(\frac{L^2}{MSE} \right), \tag{36}$$

sendo L o valor máximo possível de um pixel, ou seja $L = max(\hat{f}(x, y))$

A métrica $PSNR$ é expressa em decibels (dB), unidade originalmente definida para medir a intensidade sonora em escala logarítmica, quanto maior o valor da $PSNR$, melhor será a qualidade da imagem reconstruída. Na compressão com perdas, os valores típicos

para a PSNR das imagens estão entre 30 e 50 dB. Valores acima de 40 dB normalmente são considerados muito bons, e aqueles abaixo de 20 dB normalmente são inaceitáveis (BULL, 2014).

6.4 PROCESSAMENTO DE IMAGENS

As implementações computacionais realizadas para testes do método proposto e também dos outros métodos foram feitas no software MATLAB®, que possui apenas funções logarítmicas nas bases 2, 10 e natural (e). Por isso, para o cálculo do logaritmo na equação (31) é usada a propriedade da mudança de base do logaritmo, fazendo,

$$\log_{m^5} b_{ij} = \frac{\log(b_{ij})}{\log(m^5)}, \tag{37}$$

onde o $\log(x)$ representa o logaritmo natural de x.

A mudança de base proporcionada pela equação (37) faz com que o MATLAB® cometa um erro de aproximação. Porém, esse erro é muito pequeno, da ordem de 10^{-11}, e não afetará a reconstrução da imagem processada.

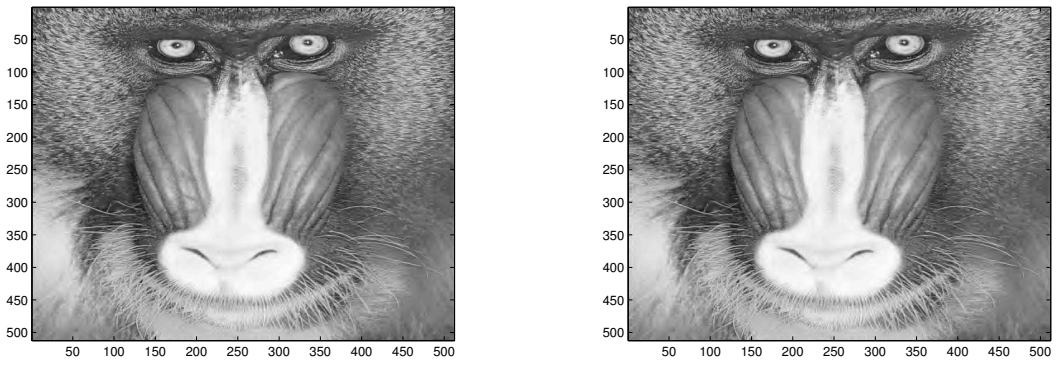
O erro de aproximação que ocorre quando usa-se a propriedade de mudança de base do logaritmo não compromete a qualidade da imagem reconstruída. Para verificação de tal fato, vamos aplicar o método proposto para três imagens diferentes e de tamanhos distintos, a saber: Baboon, Barbara e Lena, de tamanhos 512×512, 256×256, 128×128, respectivamente. Estas imagens podem ser encontradas em <https://homepages.cae.wisc.edu/~ece533/images/>.

O processamento dessas imagens será comparado em termos de $PSNR$ com os métodos (SILVA, 2008; SILVA; DUARTE; VILLARREAL, 2011). Na reconstrução de uma imagem, os métodos Silva (2008) e Silva, Duarte e Villarreal (2011) têm a mesma qualidade, pois na geração de seus vetores de armazenamento, os valores dos *pixels* das imagens não são envolvidos em nenhuma operação, eles apenas são alocados nos respectivos vetores. Enquanto que no método proposto, os valores dos *pixels* são envolvidos nas aplicações das equações (31) e (34).

As imagens processadas pelos métodos de Silva, Duarte e Villarreal (2011) e pelo método proposto são apresentadas nas Figuras 30 a 32. A taxa de compressão escolhida foi 85%.

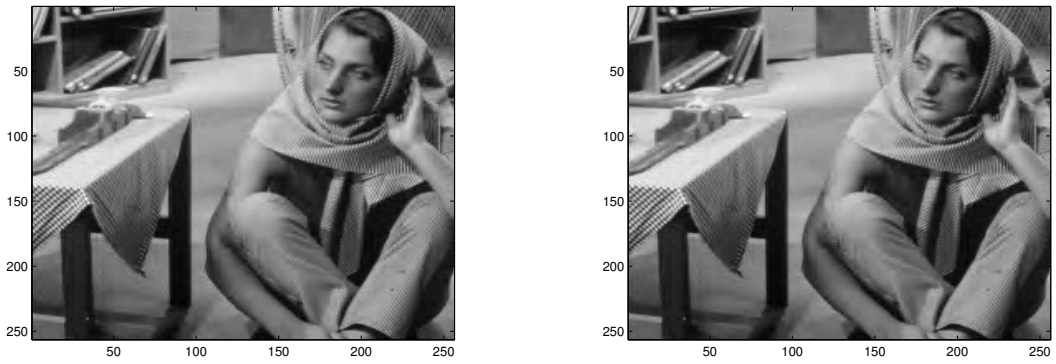
Figura 30 – (a) Baboon reconstruída pelo método de Silva, Duarte e Villarreal (2011) (b) Baboon reconstruída pelo método proposto.

(a) (b)



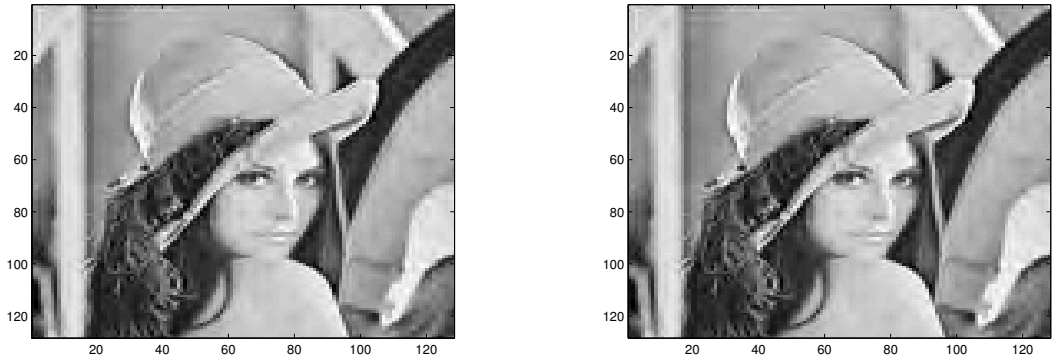
Fonte: Elaboração da própria autora.

Figura 31 – (a) Barbara reconstruída pelo método de Silva, Duarte e Villarreal (2011) (b) Barbara reconstruída pelo método proposto.



Fonte: Elaboração da própria autora.

Figura 32 – (a) Lena reconstruída pelo método de Silva, Duarte e Villarreal (2011) (b) Lena reconstruída pelo método proposto.



Fonte: Elaboração da própria autora.

Os valores da $PSNR$ em ambos os métodos são praticamente iguais, isso porque a diferença entre os valores dos *pixels* é muito pequena. Os valores obtidos são listados abaixo, onde o primeiro é o obtido pelo método de Silva, Duarte e Villarreal (2011) e o segundo pelo método proposto:

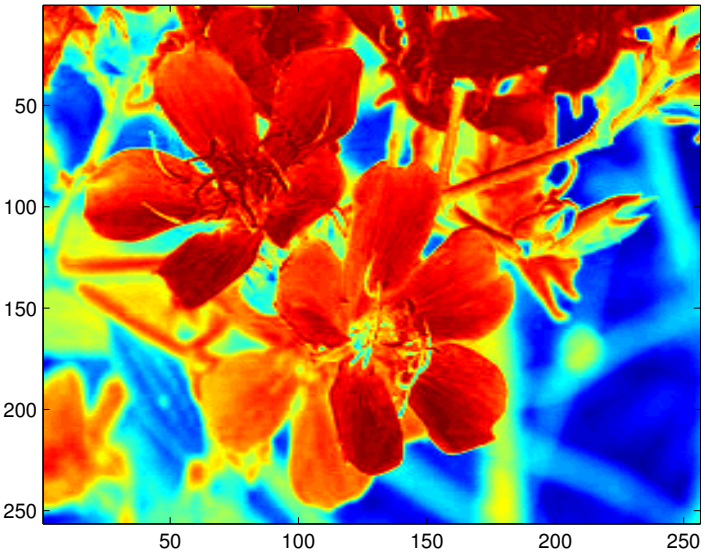
1. Baboon: 28,210658270002313 e 28,210658270084306, logo, $PSNR = 28{,}21$.
2. Barbara: 33,707098735827210 e 33,707098735818200, logo, $PSNR = 33{,}7$.
3. Lena: 32,278994027769410 e 32,2789940277664830, logo, $PSNR = 32{,}28$.

Note que as $PSNRs$ são iguais até a décima casa decimal, e a partir desta são distintos, ou seja, isso representa um erro insignificante. Para todas as imagens a taxa de compressão foi a mesma, 85%, e de acordo com cada $PSNR$, a qualidade das imagens reconstruída está aceitável.

6.4.1 Processamento de imagens no *Command Window*

Para processar uma imagem no *Command Window* é necessário remover seu mapa de cores, e trabalhar apenas com uma matriz que contenha o valor dos *pixels*. Removendo o mapa de cores, a variável ainda é do tipo `uint8`, mas será como na Figura 33.

Figura 33 – Imagem exibida sem o mapa de cores



Fonte: Elaboração da própria autora.

O próximo passo é transformar a variável `uint8` em formato `double`. Se uma imagem de dimensões $m \times m$ é carregada no *Command Window*, a variável `uint8` que a representa terá $m \times m \times 3$ *Bytes*. Transformando essa variável em `double`, o número de *Bytes* necessários para armazenamento passa a ser $m \times m \times \log_2(m)$.

Levando em conta as considerações feitas até aqui, concluímos ser necessário um outro ambiente onde possamos processar imagens coloridas. Tal ambiente é a ferramenta `wavemenu` do MATLAB®.

6.4.2 Processamento de imagens coloridas

Um dos objetivos deste trabalho consiste em processar imagens coloridas, e para isso será utilizada a caixa de ferramentas `wavemenu` do MATLAB®. A pretensão desta proposta é fazer a compressão de imagens, de modo análogo ao feito para as imagens em escala de cinza, porém com algumas mudanças: As aplicações da TWD e da TWDI são feitas no `wavemenu`, a limiarização é feita no *Command Window*.

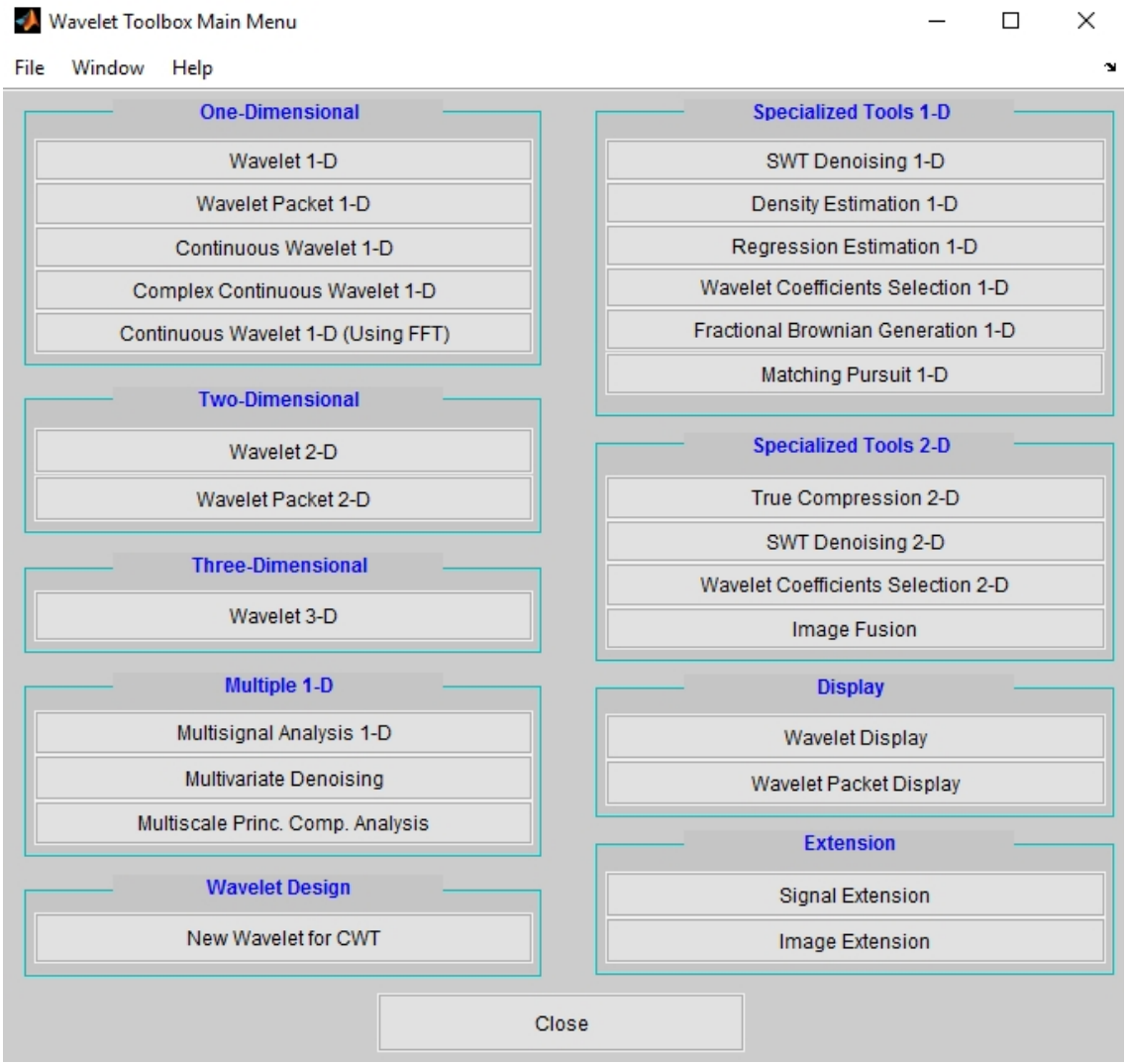
Após a aplicação da TWD o `wavemenu` fornecerá um vetor, cujo tamanho é $1 \times l, l \neq 2^n$, $n \in \mathbb{N}$. Por esse motivo, não faremos a codificação dos elementos do modo como é feito para matrizes, pois a função N está definida apenas para o domínio dado em (28). Entretanto, desde que se queira é possível codificá-lo de outra forma.

Para a realização de todos os testes, foram utilizadas imagens coloridas pertencen-

tes a classe `uint8`, isto quer dizer que a imagem possui valores variando de 0 a 255. Para trabalhar com essas imagens no método proposto, é necessário transformá-las em matrizes, e como consequência deste fato as imagens perdem seu mapa de cores e são exibidas em escala de cinza. Surge aí a necessidade de uma ferramenta que faça a compressão em parceria com o método proposto. Tal ferramenta é o `wavemenu`, e o passo-a-passo para compressão e reconstrução é dado a seguir:

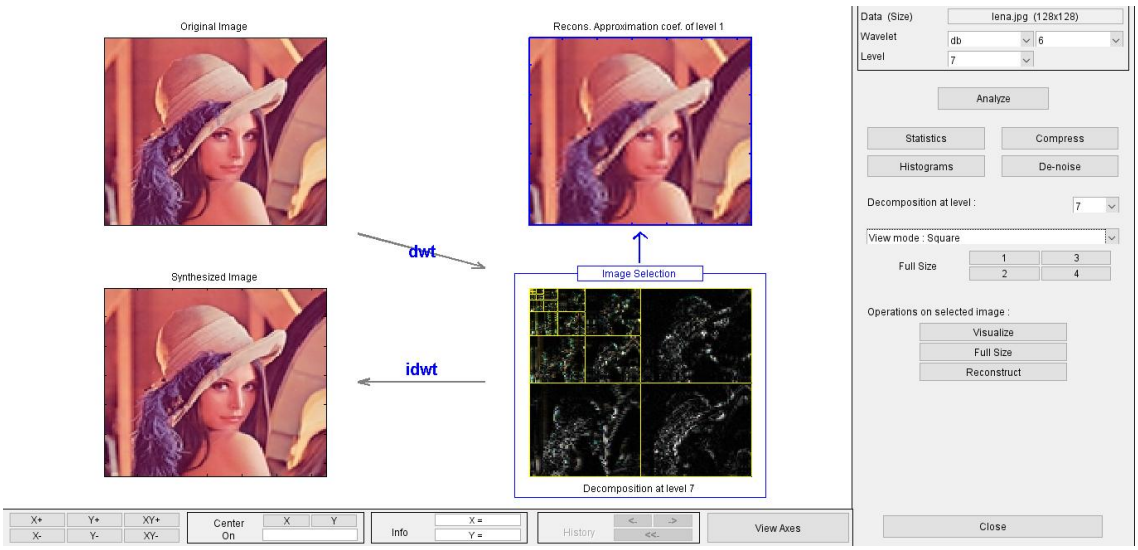
Passo 1: Digite `wavemenu` no *Command Window*. Em seguida escolha a opção Wavelet 2-D na interface mostrada na Figura 34. O ambiente usado para processamento de imagens pode ser visto na Figura 35.

Figura 34 – Interface do `wavemenu`.



Fonte: Elaboração da própria autora.

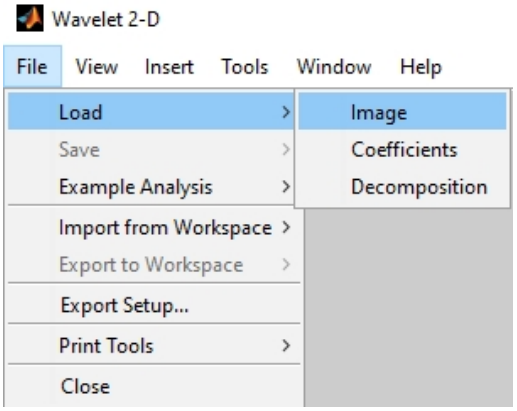
Figura 35 – Ambiente do `wavemenu` para processamento de imagens



Fonte: Elaboração da própria autora.

Passo 2: Carregue uma imagem colorida, como mostra a Figura 36 :

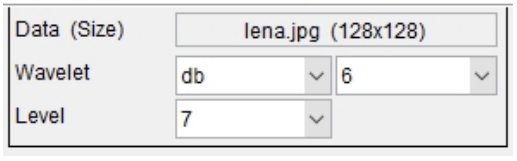
Figura 36 – Passo-a-passo de como carregar uma imagem no `wavemenu`.



Fonte: Elaboração da própria autora.

Passo 3: A função escolhida é a `db6`. Processe a imagem em todos os níveis de resolução como feito na Figura 37.

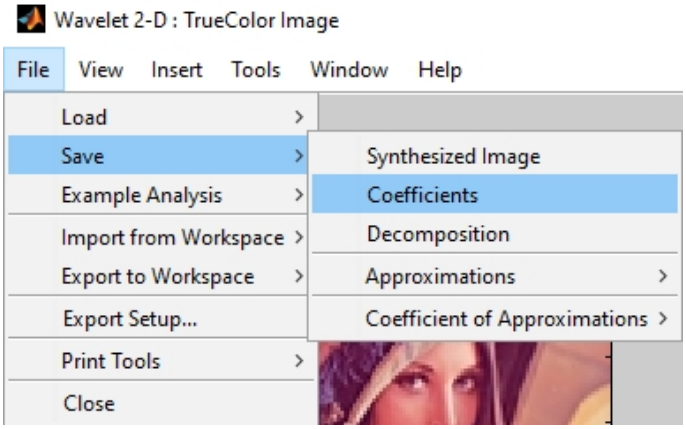
Figura 37 – Ambiente para a escolha da função wavelet e o número de níveis.



Fonte: Elaboração da própria autora.

Passo 4: Salve apenas os coeficientes seguindo os passos da Figura 38.

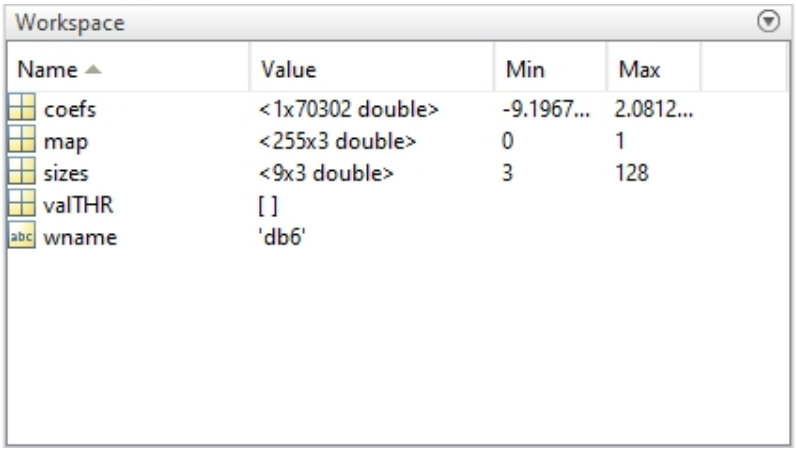
Figura 38 – Passo-a-passo para salvar os coeficientes da imagem transformada.



Fonte: Elaboração da própria autora.

Passo 5: Volte ao *Command Window* e abra o arquivo salvo no formato `.mat`. Ele será como na Figura 39:

Figura 39 – Interface da *workspace*.



Fonte: Elaboração da própria autora.

O vetor `coefs` contém os elementos transformados, e a partir de agora é nele que será aplicado o limiar. Primeiramente, a taxa de compressão é escolhida, em seguida um vetor auxiliar é criado para receber todos os coeficientes ordenados em módulo. O programa verifica a quantidade de elementos que deve ser eliminada para atingir a taxa de compressão e dentre esses escolhe o maior, logo, "`maior`" será o limiar. A função `wthresh` do MATLAB® é usada para eliminar os números abaixo do limiar. O vetor `coefs` é comprimido de acordo com o pseudocódigo da Figura 40.

Figura 40 – Pseudocódigo do Matlab para compressão do vetor `coefs`.

```
l=length(coefs);
taxa = 0.85;
% Reordena os valores absolutos transformados (ordem crescente)
aux = sort(abs(coefs));
% Quantidade de numeros a serem eliminados
maior = aux(num_el);
% Achar o maior numero dentre os que serao elimnados
num_el=round(length(coefs)*taxa);
% Aplcar o limiar hard
coefs=wthresh(coefs,'h',maior);
```

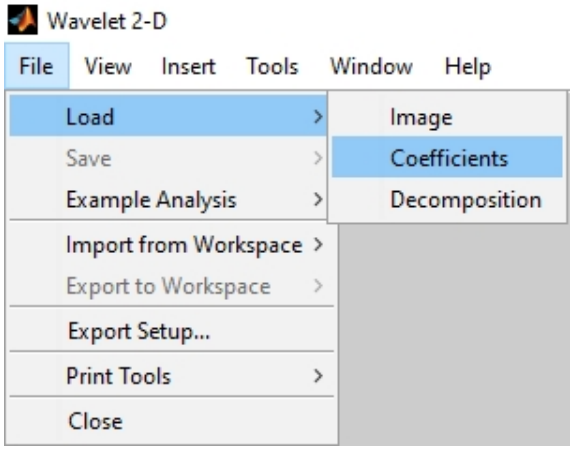
Fonte: Elaboração da própria autora.

Passo 6: O novo vetor `coefs`, após a limiarização, deve ser salvo. Basta usar o comando:

```
save('nome_do_arquivo','coefs','map','sizes','valTHR','wname');
```

Passo 7: Abra novamente o `wavemenu`, a opção Wavelet-2D e carregue o arquivo `.mat` como mostra a Figura 41. Escolha a função `db6`, o número máximo de níveis e reconstrua a imagem.

Figura 41 – Passo-a-passo para carregar os coeficientes limiarizados.

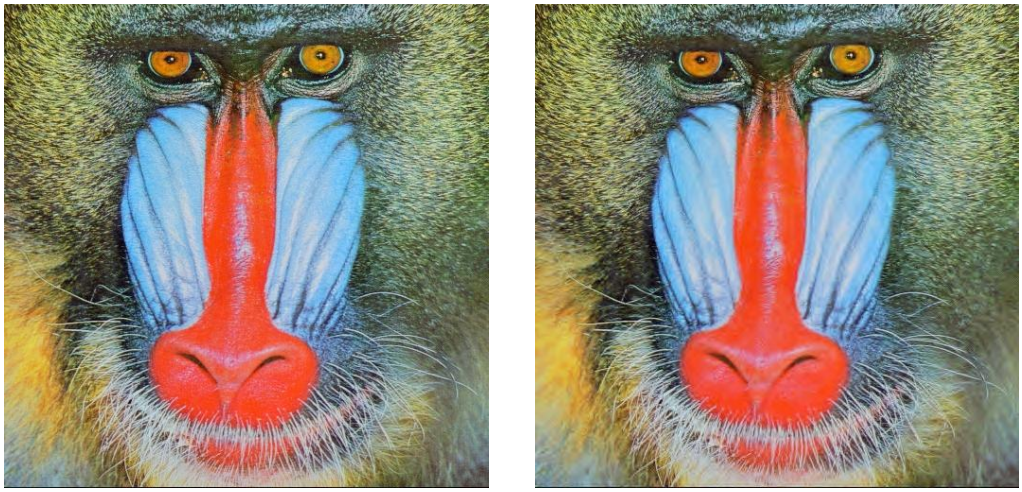


Fonte: Elaboração da própria autora.

As imagens Baboon e Lena foram comprimidas com uma taxa de 80%, seguindo os passos descritos anteriormente. Os resultados da compressão são mostrados nas Figuras 42 e 43:

Figura 42 – (a) Baboon original (b) Baboon comprimida com taxa de 80%.

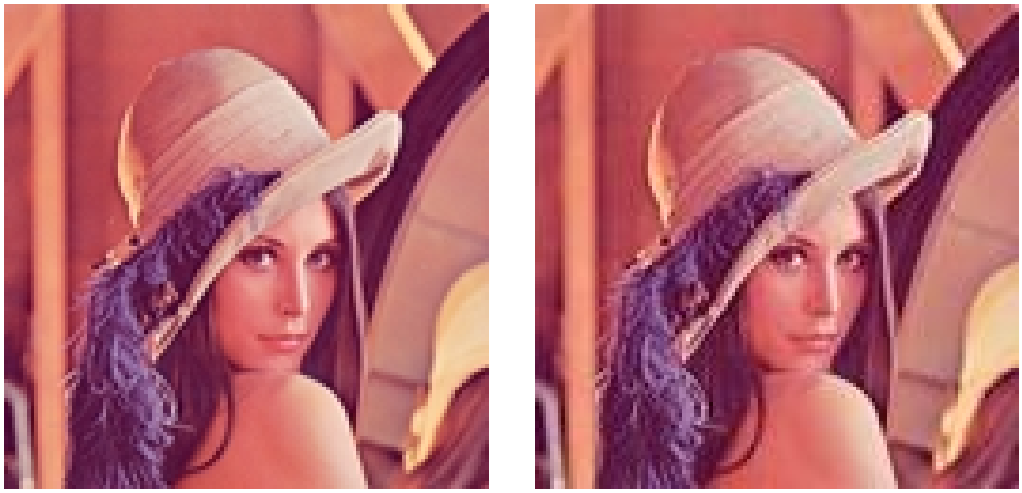
(a) (b)



Fonte: Elaboração da própria autora.

Figura 43 – (a) Lena original (b) Lena comprimida com taxa de 80%.

(a) (b)



Fonte: Elaboração da própria autora.

Os valores da $PSNR$ foram 27,38 dB e 32,32 dB para as imagens Baboon e Lena, respectivamente.

7 CONSIDERAÇÕES FINAIS

Neste trabalho, foi proposto um sistema de armazenamento de imagens, que são comprimidas usando a transformada wavelet com o objetivo de reduzir o espaço de memória computacional necessário. Este sistema armazena apenas a dimensão da imagem e os valores que a representam, através de uma função, os coeficientes que não são eliminados na compressão e suas respectivas posições. Desta maneira, torna-se bem menor o espaço utilizado para armazenamento da imagem e posterior transmissão.

Este sistema de armazenamento foi criado com base em sistemas já propostos anteriormente nos trabalhos de (SILVA, 2008; SILVA; DUARTE; VILLARREAL, 2011). Além da contribuição relativa à redução de espaço de memória computacional, houve uma grande contribuição na redução do tempo de processamento da decodificação das imagens. A proposta também permite que a taxa de compressão seja escolhida a priori. Um dos diferenciais deste trabalho é o processamento de imagens coloridas, podendo ser possível neste processo, também determinar a taxa de compressão.

Determinando a mesma taxa de compressão que os algoritmos de Silva (2008) e Silva, Duarte e Villarreal (2011) obtiveram, e usando o mesmo conjunto de imagens, o sistema proposto reduz em mais de 86% o espaço de armazenamento da imagem, em relação ao trabalho de Silva (2008). Embora o sistema armazenamento tenha sido implementado no MATLAB®, o mesmo pode ser feito em qualquer linguagem de programação e para compressão de imagens usando qualquer tipo de técnica, não apenas as técnicas baseadas na transformada wavelet.

Referências

AGULHARI, C. M. *Compressão de eletrocardiogramas usando wavelets*. 2009. 66 f. Dissertação (Mestrado em Engenharia Elétrica) — Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas- UNICAMP, Campinas, 2009.

BAGHDADI, N.; ZRIBI, M. *Optical remote sensing of land surface:* techniques and methods. [S.l.]: Elsevier, 2016. 388 p.

BAIRAGI, V. K. Symmetry-based biomedical image compression. *Journal of digital imaging*, Springer, v. 28, n. 6, p. 718–726, 2015.

BIANCHI, M. F. *Extração de características de imagens de faces humanas através de wavelets, PCA e IMPCA*. 2006. 148 f. Dissertação (Mestrado em Engenharia Elétrica) — Escola de Engenharia de São Carlos, Universidade de São Paulo- USP, São Carlos, 2006.

BULL, D. R. *Communicating pictures:* a course in image and video coding. [S.l.]: Academic Press, 2014. 560 p.

CHOWDHURY, M. M. H.; KHATUN, A. Image compression using discrete wavelet transform. *International Journal of Computer Science Issues- IJCSI*, Mahebourg, v. 9, n. 1, p. 327–330, 2012.

COELHO, F. U.; LOURENÇO, M. L. *Curso de álgebra linear*. [S.l.]: Edusp, 2001. 245 p.

COIFMAN, R. Adapted multiresolution analysis, computation, signal processing and operator theory. In: INTERNATIONAL CONGRESS OF MATHEMATICIANS, 21., 1990, Kyoto. *Proceedings...*. Kyoto: Springer-Verlag, 1990. p. 879–887.

DAUBECHIES, I. *Ten lectures on wavelets*. [S.l.]: SIAM, 1992. 377 p.

DONOHO, D. L.; JOHNSTONE, J. M. Ideal spatial adaptation by wavelet shrinkage. *Biometrika, Biometrika Trust*, London, v. 81, n. 3, p. 425–455, 1994.

DUARTE, M. A. Q. *Redução de ruído em sinais de voz no domínio wavelet*. 2005. 120 f. Tese (Doutorado em Engenharia Elétrica) — Faculdade de Engenharia, Universidade Estadual Paulista- UNESP, Ilha Solteira, 2005.

FIGUEREDO, M. V. M. *Uso de transformadas wavelet e redes neurais artificiais na compressão do eletrocardiograma*. 2008. 101 f. Dissertação (Mestrado em Informática) — Escola Politécnica, Pontifícia Universidade Católica do Paraná- PUC, Curitiba, 2008.

GHANBARI, M. *Standard codecs:* image compression to advanced video coding. [S.l.]: IET, 2003. 450 p.

GOMES, J.; VELHO, L.; GOLDENSTEIN, S. *Wavelets:* teoria, software e aplicações. Rio de Janeiro: Instituto de Matemática Pura e Aplicada- IMPA, 1997. 216 p. ISBN 85-244-0135-4.

GONZALEZ, R. C.; WOODS, R. E. *Processamento de imagens digitais*. [S.l.]: Edgard Blucher, 2000. 528 p.

GRAPS, A. An introduction to wavelets. *IEEE computational science and engineering*, Los Alamitos, v. 2, n. 2, p. 50–61, 1995.

GUIDORIZZI, H. L. *Um curso de cálculo*. 5. ed. Rio de Janeiro: LTC, 2001. 636 p.

GUSMÃO, A. A. *Método robusto e simples de compressão de imagens baseado no algoritmo EZW*. 2002. 89 f. Dissertação (Mestrado em Engenharia Elétrica) — Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas- UNICAMP, Campinas, 2002.

HAAR, A. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, Springer, v. 69, n. 3, p. 331–371, 1910.

HERNÁNDEZ, E.; WEISS, G. *A first course on wavelets*. [S.l.]: CRC, 1996. 489 p.

JAHNE, B. *Digital image processing:* concepts, algorithms, and scientific aplications. Heildelberg: Springer-Verlag, 1997. 402 p.

JAWERTH, B.; SWELDENS, W. An overview of wavelet based multiresolution analyses. *SIAM Review*, Philadelphia, v. 36, n. 3, p. 377–412, 1994.

KAISER, G. *A friendly guide to wavelets*. Boston: Birkhäuser, 1994. 300 p.

KEKRE, H. B.; SARODE, T.; NATU, P. Image compression using column, row and full wavelet transforms of walsh, cosine, haar, kekre, slant and sine and their comparison with corresponding orthogonal transforms. *International Journal of Engineering Research and Development- IJERD*, Mumbai, v. 6, n. 4, p. 102–113, 2013.

LEITE, R. B. *Compressão de imagens digitais combinando técnicas wavelet e wedgelet no ambiente de comunicações móveis*. 2011. 156 f. Dissertação (Mestrado em Engenharia Elétrica) — Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas- UNICAMP, Campinas, 2011.

MALLAT, S. G. Multiresolution approximations and wavelet orthonormal bases of $L^2(\mathbb{R})$. *Transactions of the American mathematical society*, Providence, v. 315, n. 1, p. 69–87, 1989.

MALLAT, S. G. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Piscataway, v. 11, n. 7, p. 674–693, 1989.

MEYER, Y. Principe d'incertitude, bases hilbertiennes et algèbres d'opérateurs. *Séminaire Bourbaki, Société Mathématique de France*, v. 28, n. 662, p. 209–223, 1985–1986.

OLSHAUSEN, B. A.; FIELD, D. J. Sparse coding of sensory inputs. *Current opinion in neurobiology*, Amsterdam, v. 14, n. 4, p. 481–487, 2004.

PEDRINI, H.; SCHWARTZ, W. R. *Análise de imagens digitais:* princípios, algoritmos e aplicações. [S.l.]: Thomson Learning, 2008. 528 p.

SANCHES, I. J. *Compressão sem perdas de projeções de tomografia computadorizada usando a transformada wavelet*. 2001. 136 f. Dissertação (Mestrado em Informática) — Universidade Federal do Paraná- UFPR, Curitiba, 2001.

SILVA, J. F. *Sistema de armazenamento de imagens comprimidas através da transformada wavelet*. 2008. 99 f. Dissertação (Mestrado em Engenharia Elétrica) — Faculdade de Engenharia, Universidade Estadual Paulista- UNESP, Ilha Solteira, 2008.

SILVA, J. F.; DUARTE, M. A. Q.; VILLARREAL, F. Um método para codificação de imagens digitais no domínio wavelet. In: CONFERÊNCIA BRASILEIRA DE DINÂMICA, CONTROLE E APLICAÇÕES- DINCON, 10., 2011, Águas de Lindóia. *Anais...* Águas de Lindóia: SBMAC, 2011. p. 731–734.

SINGH, B.; KAUR, A.; SINGH, J. A review of ecg data compression techniques. *International Journal of Computer Applications*, New York, v. 116, n. 11, p. 39–44, 2015.

SOARES, W. C. *Um método não-limiar para redução de ruído em sinais de voz no domínio wavelet*. 2009. 205 f. Tese (Doutorado em Engenharia Elétrica) — Faculdade de Engenharia, Universidade Estadual Paulista- UNESP, Ilha Solteira, 2009.

SOARES, W. C.; VIEIRA FILHO, J.; DUARTE, M. A. Q.; VILLARREAL, F. Análise de métodos de redução de ruído por limiar no domínio wavelet. *Tema - Tendencias em Matemática Aplicada e Computacional*, Rio de Janeiro, v. 9, n. 3, p. 471–480, 2008.

STOLLNITZ, E. J.; DEROSE, A. D.; SALESIN, D. H. Wavelets for computer graphics: a primer, part 1. *IEEE Computer Graphics and Applications*, Piscataway, v. 15, n. 3, p. 76–84, 1995.

STOLLNITZ, E. J.; DEROSE, T. D.; SALESIN, D. H. *Wavelets for computer graphics:* theory and applications. [S.l.]: Morgan Kaufmann, 1996. 245 p.

STROMBERG, J. O. A modified franklin system and higher-order spline system on $\mathbb{R}^n$ as unconditional basis of hardy spaces. *Conference in Harmonic Analysis in Honor of Antoni Zygmund*, Chicago, v. 2, n. 3, p. 475–493, 1981.

TRAVASSOS, N. C. L.; DUARTE, M. A. Q.; VILLARREAL, F. Encoding and decoding compressed images in the wavelet domain. In: ENCONTRO NACIONAL DE MODELAGEM COMPUTACIONAL- ENMC, 19., 2016, João Pessoa. *Anais...* João Pessoa: UFPB, 2016. p. 755–763.

WALKER, J. S. *A primer on wavelets and their scientific applications*. [S.l.]: CRC, 2008.

WU, K.; OTOO, E. J.; SHOSHANI, A. Optimizing bitmap indices with efficient compression. *ACM Transactions on Database Systems*, New York, v. 31, n. 1, p. 1–38, 2006.

APÊNDICE A – Demonstração

Vamos mostrar que a função (28) é invertível, e que (30) é sua inversa.

Para cada $m \in \{2^n : n \in \mathbb{N}\}$ definamos $I_m = \{3, 4, \cdots, m^2+2\}$ e $L_m = \{1, \cdots, m\}$. Considere a função $N : L_m \times L_m \to I_m$ dada por :

$$N(i,j) = a_{ij}^3 = -m + 2 + mi + j = m(i-1) + j + 2.$$

Observe que tal função está bem definida pois,

$$3 \leq a_{ij}^3 \leq m^2 + 2, \text{ para todo } (i,j) \in L_m \times L_m.$$

Vamos mostrar que tal função é uma bijeção. Uma vez que as cardinalidades de $L_m \times L_m$ e I_m são finitas e iguais a m^2, é suficiente mostrarmos apenas que N é injetiva. Suponhamos que existam $(i_1, j_1), (i_2, j_2) \in L_m \times L_m$ tais que $N(i_1, j_1) = N(i_2, j_2)$. Então,

$$\begin{array}{l} k-3 = m(i_1-1) + j_1 - 1 = mq_1 + r_1 \\ k-3 = m(i_2-1) + j_2 - 1 = mq_2 + r_2 \end{array}, \tag{38}$$

onde

$$\begin{cases} 0 \leq j_1 - 1 < m \\ 0 \leq j_2 - 1 < m \end{cases}.$$

Nas equações de (38), as divisões euclidianas de $k-3$ por m têm:

- como quocientes: $i_1 - 1$ e $i_2 - 1$,
- como restos: $j_1 - 1$ e $j_2 - 1$.

Pelo Algoritmo da Divisão segue que $(i_1, j_1) = (i_2, j_2)$, provando que N é injetiva. Logo N é uma bijeção e, portanto, existe $N^{-1} : I_m \to L_m \times L_m$. Agora, considere a seguinte função $M : I_m \to L_m \times L_m$ dada por

$$M(k) = \left(\frac{k - 3 - \text{mod } (k-3, m)}{m} + 1, \text{ mod } (k-3, m) + 1 \right),$$

onde a função mod $(k-3, m)$ retorna o resto da divisão de $k-3$ por m. Vamos mostrar que $M = N^{-1}$. Primeiramente observe que se $(i,j) \in L_m \times L_m$ então

$$\text{mod } (a_{ij}^3 - 3, m) = \text{mod } (m(i-1) + j - 1, m) = j - 1.$$

Portanto,

$$M \circ N(i,j) = \left(\frac{a_{ij}^3 - 3 - \text{mod } (a_{ij}^3 - 3, m)}{m} + 1, \text{ mod } (a_{ij}^3 - 3, m) + 1 \right)$$

$$
\begin{aligned}
&= \left(\frac{m(i-1)+j-1-(j-1)}{m} + 1, j-1+1 \right) \\
&= (i,j)\,,
\end{aligned}
$$

ou seja, M é a inversa à esquerda de N e portanto, pela unicidade da inversa, $M = N^{-1}$ como desejado.

APÊNDICE B – Construindo funções

Para construção da função (31) o primeiro passo foi escolher uma função invertível já conhecida, a saber $\log(x)$, que pudesse relacionar b_{ij} e dimensão m de algum modo. Após testes, tais elementos foram envolvidos na função invertível da seguinte forma: $\log_{m^5}|b_{ij}|$. A decisão de escolher o número 5 como expoente de m foi apenas uma questão de segurança matemática, pois, usando 4 como expoente o resultado desejado já é obtido, e a partir de 128 a qualidade da imagem reconstruída é insatisfatória. Ainda era preciso envolver a_{ij}^3, e a melhor maneira, foi adicionando-o : $\log_{m^5}|b_{ij}| + a_{ij}^3$. Devido ao fato de termos usado o módulo na função log, foi preciso multiplicar nossa nova função pelo sinal do elemento: $\texttt{sign}(b_{ij})(\log_{m^5}|b_{ij}| + a_{ij}^3)$. A construção de $\log_{m^5}|b_{ij}|$ garante que este número estará sempre no intervalo $[0, 1)$, então a alternativa para achar a_{ij}^3 é aplicando a função `round` do MATLAB®. Para achar o valor de b_{ij} foram usadas as propriedades da inversa da função $\log(x)$.

O processo descrito acima não é tão simples, entretanto, existem maneiras menos trabalhosas para a escolha de uma função desse tipo. Basta usar a seguinte lógica: Para cada $m \in \{2^n : n \in \mathbb{N}\}$ seja $I_m = \{3, 4, \cdots, m^2 + 2\}$. Considere o problema de determinar um função invertível $f : \mathbb{R} \times I_m \to \mathbb{R}$ que a cada par $(b_{ij}, a_{ij}^3) \in \mathbb{R} \times I_m$ faz corresponder um único número $f(b_{ij}, a_{ij}^3)$. Um linha de raciocínio para obter tal função é a seguinte: dado que a variável a_{ij}^3 é sempre um número natural, se considerarmos uma função $g : \mathbb{R} \to [0, 1)$ e definirmos

$$f(b_{ij}, a_{ij}^3) = g(b_{ij}) + a_{ij}^3, \tag{39}$$

então podemos facilmente determinar a_{ij}^3 a partir de $f(b_{ij}, a_{ij}^3)$ aplicando a função $\texttt{floor}(x)$ do MATLAB®, que arredonda x na direção de menos infinito. De fato, observe que:

$$\begin{aligned} & 0 \leq g(b_{ij}) < 1 \\ \implies \quad & a_{ij}^3 \leq g(b_{ij}) + a_{ij}^3 < a_{ij}^3 + 1 \\ \implies \quad & a_{ij}^3 \leq f(b_{ij}, a_{ij}^3) < a_{ij}^3 + 1 \\ \implies \quad & f(b_{ij}, a_{ij}^3) \in [a_{ij}^3, a_{ij}^3 + 1). \end{aligned} \tag{40}$$

De (40) concluímos que

$$\texttt{floor}(f(b_{ij}, a_{ij}^3)) = a_{ij}^3. \tag{41}$$

Para determinarmos a variável b_{ij} a partir de $f(b_{ij}, a_{ij}^3)$, observe que (39) e (41) implicam em

$$f(b_{ij}, a_{ij}^3) - \texttt{floor}(f(b_{ij}, a_{ij}^3)) = g(b_{ij}). \tag{42}$$

Se supormos que a função g é invertível então de (42) podemos determinar a variável b_{ij} a partir de $f(b_{ij}, a_{ij}^3)$, segundo a equação (43).

$$g^{-1}(f(b_{ij}, a_{ij}^3) - \texttt{floor}(f(b_{ij}, a_{ij}^3))) = g^{-1}(g(b_{ij})) = b_{ij}. \tag{43}$$

Logo, se $g : \mathbb{R} \to [0, 1)$ é uma função invertível, a função f definida pela equação (39) será invertível, e sua inversa é dada pela equação (44).

$$f^{-1}(z) = (g^{-1}(z - \texttt{floor}(z)), \texttt{floor}(z)). \tag{44}$$

Portanto, o problema de determinar uma função invertível $f : \mathbb{R} \times I_m \to \mathbb{R}$ pode ser resumido a encontrar uma função $g : \mathbb{R} \to [0, 1)$ que seja invertível e definir f pela equação (39).

Exemplo: Um exemplo de função que satisfaz essa condições é

$$f(b_{ij}, a_{ij}^3) = \frac{\tan^{-1}(b_{ij})}{\pi} + \frac{1}{2} + a_{ij}^3 = z.$$

A inversa é obtida seguindo as condições dadas:

$$f^{-1}(z) = \left(\tan(\pi(z - \texttt{floor}(z) - \frac{1}{2})), \texttt{floor}(z)\right),$$

onde a primeira coordenada fornece b_{ij}, e a segunda a_{ij}^3.