Comparando Desempenho de Compressão de Imagens Utilizando Transformadas Wavelets

Huberto Kaiser Filho
Engenharia da Computação
Universidade Federal de Pelotas
Pelotas, RS
Email: hkaiser@inf.ufpel.edu.br

Leonardo da Rosa Silvera João
Engenharia da Computação
Universidade Federal de Pelotas
Pelotas, RS
Email: ldrsjoao@inf.ufpel.edu.br

Vinicius Rodrigues dos Santos
Ciência da Computação
Universidade Federal de Pelotas
Pelotas, RS
Email: vrdsantos@inf.ufpel.edu.br

***Resumo*—Este trabalho compreende a modelagem da compressão de imagens utilizando a transformada que utiliza componentes Wavelets da Transformada Discreta de Daubechie e a sua comparação com a Transformada Wavelet de Haar. São apresentadas e comparadas as análises qualitativas das imagens obtidas através dos algoritmos resultantes. A qualidade de resultados é verificada ao comparar duas métricas: MSE (*Mean Squared Error*) e $PSNR$ (*Peak Signal-to-Noise Ratio*).**

***Keywords*—Transformada, Daubechie, Wavelets, Haar, Compressão.**

I. Introdução

Na última década, muitos estudos associando as transformadas wavelet surgiram [1] [2] [3], promovendo uma nova direção de pesquisas e apontamentos para a relevância do processamento de sinais para uma grande variedade de aplicações. O trabalho apresentado por [1] e [2] apresenta os métodos de integração para resolver equações diferenciais parciais usando intervalos wavelets. O método para inserir marcas d'água em imagens usando a decomposição de intervalos wavelet é introduzido por [3].

A Transformada Wavelet de Haar (HWT), é a Transformada Discreta Wavelet (DWT) mais simples devido a complexidade de seus filtros e ainda é largamente explorada nos dias de hoje devido a sua robustez em respeito a diverstas estruturas matemáticas e .

A HWT, é a Transformada Discreta Wavelet (DWT) mais simples devido à complexidade de seus filtros, é ainda hoje amplamente explorada, sendo sua robustez em relação a muitas estruturas matemáticas e formulações espaciais diferentes em um aspecto relevante abordado em muitas aplicações [4].

Além disso, como demonstrado em [5], as imprecisões de imagens com base em wavelets de Haar são as menores possíveis, mnotivando o seu uso em muitos problemas 2D bem como a formulação atual no contexto de intervalo. As técnicas que envolvem as wavelets são amplas, especialmente no processamento de sinais e imagens [6], [7]

Levando em consideração estes aspectos, o objetivo deste trabalho é implementar um algoritmo de compressão o qual faz uso da Transformada Wavelet de Daubechies e comparalo com o algoritmo de compressão utilizando a Transformada Wavelet de Haar presente no Matlab.

II. Transformada Wavelet de Haar (HWT)

A base da transformada de Haar foi proposta em 1910, introduzida pelo matemático Húngaro Alfred Haar [8]. Nos dias de hoje, a Transformada Wavelet de Haar é utilizada para diferentes aplicações, especialmente as que envolvem compressão como no caso do JPEG-2000 [9].

Originalmente, a HWT foi projetada para um vetor unidimensional de dados, contúdo a HWT pode ser estendida para um vetor de duas dimensões. De acordo com o procedimento descrito em [10], o algoritimo mais rápido para a HWT 2D é obtido através da aplicação da transformação unidimensional por direção ou seja, em todas as linhas da matriz de entrada, e após em todas as colunas da matriz resultante.

Seguindo o que foi apresentado em [10], a HWT 2D pode ser calculada através do método standard (Figura 1) onde os diversos níveis da transformada unidimensional são aplicados a todas as linhas. Depois das $l = log_2 n$ decomposições da HWT 1D, o mesmo pocedimento é aplicado para as colunas da matriz resultante.

```
Standard (C[1..h, 1..w])
1 for row ← 1 to h do
2 |  Decomposition(C[row, 1..w]) ;
3 end
4 for col ← 1 to w do
5 |  Decomposition(C[1..h, col]) ;
6 end
```

```
Non-Standard (C[1..h, 1..w])
1 C ← C/h ;
2 while h > 1 do
3 |  for row ← 1 to h do
4 |  |  DecompositionStep(C[row, 1..w]) ;
5 |  end
6 |  for col ← 1 to w do
7 |  |  DecompositionStep(C[1..h, col]) ;
8 |  end
9 end
```

Figura 1: HWT 2D: Processo de decomposição standard e non-standard.

De acordo com [10] e mostrado na Figura 1, o método non-standard é outra opção e a utilizada neste trabalho para a HWT 2D, onde o intervalo de extensão é levado em consideração. Em sua formulação, as operações por linhas e colunas através de cada nível da transformação são intercaladas. Neste método, os coeficientes de escala resultantes após um nível de decomposição, são um quarto dos dados iniciais, estes são então considerados como a entrada para o cálculo do próximo nível de decomposição da transformação.

Ao final, os três blocos restantes com coeficientes de wavelet não são mais decompostos. Cada um também contém

um quarto dos dados originais. O processo completo repete-se até que no último nível apenas um único coeficiente de escala e apenas um coeficiente wavelet para os restantes três conjuntos são obtidos. Uma representação desta formulação é apresentada na Figura 2 (painel direito).

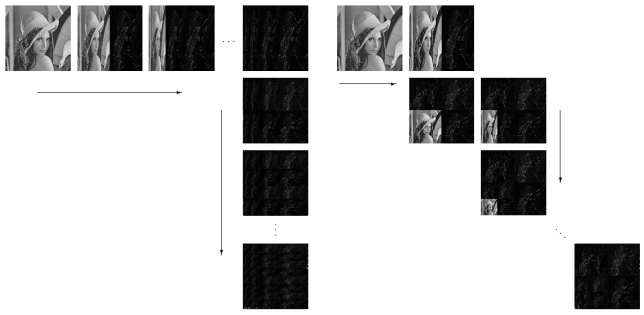


Figura 2: HWT 2D com $n \geq 3$ níveis de decomposição.(painel esquerdo) método standard; (painel direito) método non-standard.

III. Transformada Wavelet de Daubechie (DaWT)

De acordo com a descrição em [11], assumindo a formulação original da transformada e dado um vetor inicia C com n valores pontuais, a DaWT unidimensional calcula coeficientes wavelets escalares para cada par de elementos adjacentes C, $(C_{2j-1}, C_{2j}), j = 1, ..., n/2$, gerando dois grupos de output: um vetor contendo a escala e um segundo vetor com coeficientes wavelets, ambos com $n_1 = n/2$ pontos, ou seja, metade do tamanho do vetor de entrada original C.

Decomposition $(C[1..n])$
1 $size \leftarrow n$;
2 **while** $size >= 4$ **do**
3 $\quad DecompositionStep(C[1..size])$;
4 $\quad size \leftarrow size/2$;
5 **end**
DecompositionStep $(C[1..n])$
1 $i \leftarrow 1$;
2 $j \leftarrow 1$;
3 $half \leftarrow n/2$;
4 **while** $j <= n-3$ **do**
5 $\quad C'[i] \leftarrow C[j]*h1 + C[j+1]*h2 + C[j+2]*h3 + C[j+3]*h4$;
6 $\quad C'[i+half] \leftarrow C[j]*g1 + C[j+1]*g2 + C[j+2]*g3 + C[j+3]*g4$;
$\quad j \leftarrow j+2$;
7 $\quad i \leftarrow i+1$;
8 **end**
9 $C'[i] \leftarrow C[n-1]*h1 + C[n]*h2 + C[1]*h3 + C[2]*h4$;
10 $C'[i+half] \leftarrow C[n-1]*g1 + C[n]*g2 + C[1]*g3 + C[2]*g4$;
11 $C \leftarrow C'$;

Figura 3: DaWT 1D Procedimento de decomposição.

A figura 3 apresenta os $log_2(n)$ níveis de decomposição do algoritimo.

O próximo nível de decomposição produz um segundo par de vetores, cada um com metade do tamanho original, um para o novo n2 = n1/2 coeficientes escalares e outro para os n2 = n1/2 coeficientes wavelets. Este procedimento é recursivamente definido e pode ser aplicado até que um nível específico for atingido ou até obter dois coeficientes escalares.

Utilizando os coeficientes wavelets resultados do processo de decomposição, juntamente com os coeficientes escalares no menor nível de decomposição, é possível efetuar a transformação inversa, também conhecida como composição, onde o vetor original é reconstituído.

Composition $(C[1..n])$
1 $size \leftarrow 4$;
2 **while** $size <= n$ **do**
3 $\quad CompositionStep(C[1..size])$;
4 $\quad size \leftarrow size * 2$;
5 **end**
CompositionStep $(C[1..n])$
1 $i \leftarrow 1$;
2 $j \leftarrow 1$;
3 $half \leftarrow n/2$;
4 $C'[1] \leftarrow C[half+1]*ih1 + C[n]*ih2 + C[0]*ih3 + C[half]*ih4$;
5 $C'[2] \leftarrow C[half+1]*ig1 + C[n]*ig2 + C[0]*ig3 + C[half]*ig4$;
6 **while** $i < half - 1$ **do**
7 $\quad C'[j] \leftarrow C[i]*ih1 + C[i+half]*ih2 + C[i+1]*ih3 + C[i+half+1]*ih4$;
$\quad C'[j+1] \leftarrow C[j]*ig1 + C[j+1]*ig2 + C[j+2]*ig3 + C[j+3]*ig4$;
$\quad j \leftarrow j+2$;
8 $\quad i \leftarrow i+1$;
9 **end**
10 $C \leftarrow C'$;

Figura 4: DaWT 1D inversa, processo de composição para reconstruir os $l = log_2 n$ níveis.

O pseudo código que reproduzo o nível de composição da DaWT 1D é representado na Figura 4. De acordo com [11], no processo de composição, começando pelo nível mais grosso da transformação, os valores originais são restaurados nível por nível combinando os coeficientes escalares e de wavelet.

A DaWT pode ser extendida para vetores bidimensionais, usando o mesmmo concito apresentado para a HWT. De acordo com o procedimento descrito em [11], o algoritmo rápido para a DaWT é obtido através da aplicação da transformação unidirecional por direção (em todas as linhas da matriz e depois disso, em todas as colunas da matriz resultante).

IV. Metodologia

A fim de obter os resultados desejados, os algoritmos propostos na sessão anterior foram desenvolvidos para as quatro etapas de processamento descritas a seguir.

Como pode ser observado na Figura 5 em (a) é considerada uma imagem de entrada a qual em (b) é aplicado a Transformada Direta de Daubechies, onde os coeficientes wavelets são calculados. Em (c), é feita a compressão da imagem removendo uma determinada porção de energia do espaço *wavelet*, definido pelo parâmetro "e" presente no algoritimo da Figura 6. Em (d) ocorre a Transformada Inversa de Daubechies, onde os coeficientes *wavelets* são relacionados com os escalares da imagem constituindo assim uma imagem aproximada da original utilizada na entrada do programa.

V. Testes e Resultados

A fim de validar os resultados obtidos com o algoritmo de compressão utilizando a DaWT, comparamos os resultados obtidos com a compressão utilizando a HWT utilizando duas métricas: MSE (*Mean Squared Error*) que mede a média dos

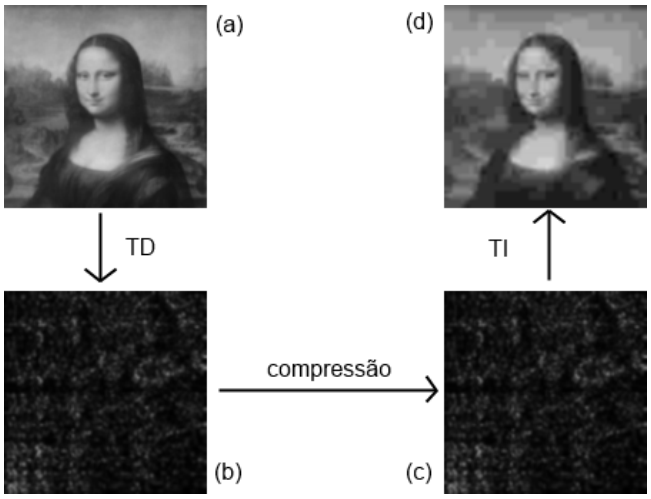


Figura 5: Etapas presentes no filtro desenvolvido.

procedure *Compress*(C: **array** $[1..m]$ **of reals**; ϵ: **real**)
  $\tau_{\min} \leftarrow \min\{\,|C[i]|\,\}$
  $\tau_{\max} \leftarrow \max\{\,|C[i]|\,\}$
  do
    $\tau \leftarrow (\tau_{\min} + \tau_{\max})/2$
    $s \leftarrow 0$
    for $i \leftarrow 1$ **to** m **do**
      if $|C[i]| < \tau$ **then** $s \leftarrow s + (C[i])^2$
    end for
    if $s < \epsilon^2$ **then** $\tau_{\min} \leftarrow \tau$ **else** $\tau_{\max} \leftarrow \tau$
  until $\tau_{\min} \approx \tau_{\max}$
  for $i \leftarrow 1$ **to** m **do**
    if $|C[i]| < \tau$ **then** $C[i] \leftarrow 0$
  end for
end procedure

Figura 6: Pseudo algoritmo utilizado para a implementação do procedimento de compressão.

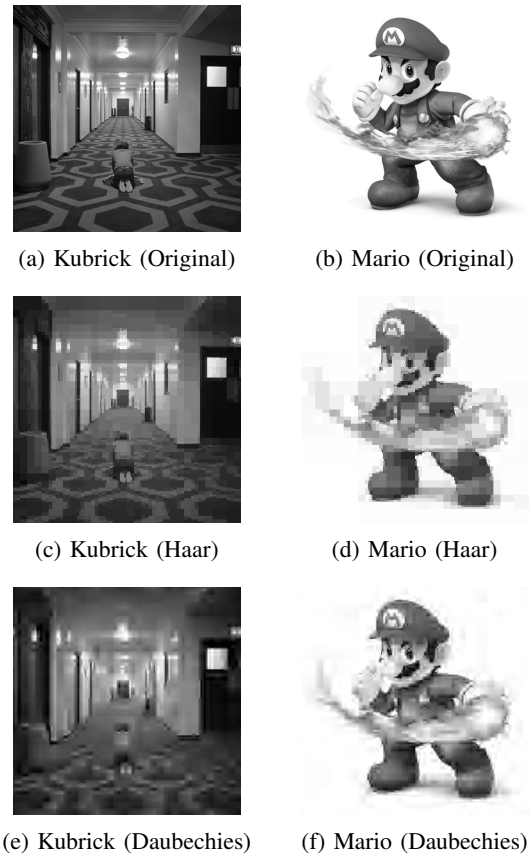
quadrados dos erros ou desvios e $PSNR$ (*Peak Signal-to-Noise Ratio*) o qual é a relação entre a potência máxima possível de um sinal e a potência do ruído que afeta a fidelidade de sua representação. Ambas as métricas foram obtidas utilizando as funções do matlab *immse* e *psnr* respectivamente.

Para a realização dos testes utilizou-se uma máquina de Processador Core i5 5200U @2.2GHz e turbo boost de 2.7 GHz, 8GB de memória RAM DDR3L e SSD 240 GB com velocidade de leitura de dados 530 MB/s e escreita 440 MB/s.

O processo de comparação se deu em obter as métricas da mesma imagem comprimida utilizando a HWT e a DaWT, foram utilizadas duas imagens. As imagens de entrada e saida para os dois tipos de compressão podem ser observados naFigura 7 e os resultados podem ser observados na Tabela I.

Através dos resultados obtidos[1], podemos observar que

[1]valores de Razão de Compressão utilizados na compressão das imagens de teste diferem devido aos diferentes algoritmos utilizados, porém possuem a mesma energia no espaço *Wavelet*.



(a) Kubrick (Original) (b) Mario (Original)

(c) Kubrick (Haar) (d) Mario (Haar)

(e) Kubrick (Daubechies) (f) Mario (Daubechies)

Figura 7: Imagens originais de entrada para testes.

Tabela I: Comparação de métricas para imagens de teste.

	Kubrick		
	Razão de Compressão[1]	**MSE**	**PSNR**
Daubechie	3,265663E-02	3,012244E+02	2,334190E+01
Haar	0.8752	1,739894E+02	2,572558E+01

	Mario		
	Razão de Compressão[1]	**MSE**	**PSNR**
Daubechie	4,849000E-03	2,264445E+02	2,458119E+01
Haar	0.4776	3,271667E+02	2,298311E+01

para o que algoritmo da DaWT se aproxime o máximo do HWT, devemos aumentar a razão de compressão. Mas mesmo procurando uma razão para que a compressão seja a mesma, ou seja, para que as métricas sejam iguais para ambos os casos de compressão, não foi possível obter valores iguais.

VI. Conclusão

Este trabalho apresentou a definição de duas Transformadas Discretas *Wavelet*, a Tranformada *Wavelet* de Haar (HWT) e a Transformada *Wavelet* de Daubechies (DaWT), como também suas aplicações na área de Processamento de Sinais e mais especificamente em Processamento de Imagens.

Ambos as transformações foram utilizadas para compressão das imagens de teste e as qualidades de resultados analisados e comparados. A métrica *MSE* demonstra o quão próximos os resultados estão da imagem original, definindo assim qual

o melhor método de compressão. A métrica *PSNR* demonstra o nível de detalhe presente na imagem.

Os resultados analisados na Tabela I mostram que a DaWT se adapta melhor com a imagem de parâmetro, retirando mais detalhes de imagens que têm mais detalhes, enquanto que a HWT se adapta menos à imagem de parâmetro.

Figura 8: Imagem comprimida gerada utilizando DaWT com taxa de compressão 0.005.

A Figura 8 demonstra o resultado da compressão utilizando DaWT sobre imagens coloridas. É possível notar um bom desempenho do procedimento de compressão utilizando o valor 0.005 como taxa de compressão.

O código utilizado neste trabalho pode ser acessado em pelo endereço: https://github.com/Leonardojoao01/Projeto_final_DSP

REFERÊNCIAS

[1] M. Shu-li, L.Qi-shao, Z. Sen-wen, and J. Li, "Adaptive interval wavelets precise integration method for partial differential equations," *Applied Mathematics and Mechanics*, vol. 26, no. 3, pp. 364–371, 2005.

[2] L. Liu, "Interval wavelet numerical method on fokker-planck equations for nonlinear random systems," *Advances in Mathematical Physics*, vol. 2013, no. ID 651357, pp. 1–7, 2013.

[3] T. Minamoto and K. Aoki, "A blind digital image watermarking using interval wavelet decomposition," *Int. Journal of Signal proc., Image proc. and Pattern recognition*, vol. 3, no. 2, pp. 59–72, 2010.

[4] I. Y. Novikov and M. A. Skopina, "Why are haar bases in various structures the same?" *Mathematical Notes-Short Communications*, vol. 91, no. 6, pp. 895–898, 2012.

[5] A. Brito and O. Kosheleva, "Interval + image = wavelet: for image processing under interval uncertainty, wavelets are optimal," *Reliable Computing*, pp. 291–301, 1998.

[6] M. B. H. Om, "An improved image denoising method based on wavelet thresholding," *Journal of Signal and Inf. Processing*, vol. 3, pp. 109–116, 2012.

[7] M. Kumar, S. K. Sudhansu, and V. Kasabegoudar, "Wavelet based texture analysis and classification with linear regration model," *Intl. Journal of Eng. Research and Appls. (IJERA)*, vol. 2, pp. 1963–1970, 2012.

[8] I. Daubechies, "Orthonormal bases of compactly supported wavelets," *Communications on Pure and Applied Mathematics*, vol. 41, no. 7, pp. 909–996, 1988. [Online]. Available: http://dx.doi.org/10.1002/cpa.3160410705

[9] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The jpeg2000 still image coding system: an overview," *Consumer Electronics, IEEE Transactions on*, vol. 46, no. 4, pp. 1103–1127, Nov 2000.

[10] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin, "Wavelets for computer graphics: A primer, part 1," *IEEE Computer Graphics and Applications*, vol. 15, no. 3, pp. 76–84, 1995.

[11] O. M. Nielsen, "Wavelets in scientific computing," Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 1998, supervisor: Per Christian Hansen, pcha@dtu.dk, DTU Compute. [Online]. Available: http://www.compute.dtu.dk/English.aspx