Int-DWTs Library - Algebraic Simplifications Increasing Performance and Accuracy of Discrete Wavelet Transforms

Orientando: **Vinícius Rodrigues dos Santos**
Orientador: **Renata Hax Sander Reiser**
Coorientador: **Maurício Lima Pilla**
Colaborador: **Alice de Jesus Kozakevicius**

Universidade Federal de Pelotas
Ciência da Computação
Centro de Desenvolvimento Tecnológico

11/07/2015 - PELOTAS / RS

Matemática Intervalar

Vantagens - **Aritmética Intervalar** (Moore/1959 e Sunaga/1958)

- Tecnicas Intervalares são aplicadas na representação de dados inexatos, aproximações e erros nos procedimentos computacionais.
- Tem alcançado resultados significativos:
  - **controle automático** para o limite dos erros;
  - **algoritmos numéricos autovalidáveis**;
  - **tratamento da precisão** em sistema de ponto flutuante;
  - garantindo representação com a **máxima exatidão**;
  - **maior confiabilidade** em relação aos erros de propagação, de arredondamentos e truncamentos.

Principais Aplicações - Computação Científica

- **representação de valores contínuos** (números irracionais);
- **modelagem de parâmetros** (medidas obtidas por instrumentos de precisão limitada).

Biblioteca C-XSC

- Biblioteca para manipulação de intervalos numéricos.
- C-XSC foi elaborada em C++ e é direcionada ao desenvolvimento de algoritmos que necessitam de validação de resultados.
- Possibilidade de paralelização das operações intervalares.
- Exemplos de estruturas presentes na biblioteca:
  - **interval** (intervalo de reais).
  - **complex** (número complexo).
  - **rvector** (vetor de números reais).
  - **dotprecision** (acumulador real de alta precisão).

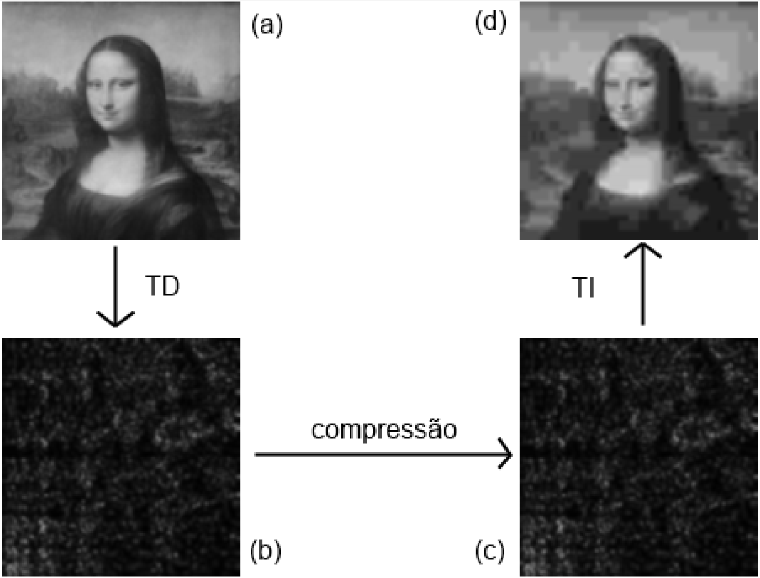
Introdução

Transformadas Wavelets

- As Transformadas Wavelets (TWs) são ferramentas matemáticas para a decomposição hierárquica de funções.
- São utilizadas em análise de sinais para identificação de detalhes.
- Uma característica importante das TWs é a composição (transformação inversa) dos sinais transformados para os originais.

Aplicações

- As Transformadas Wavelets são aplicadas para resolução de problemas relacionados com, por exemplo:
  - **compressão de imagens**,
  - controle automatizado de nível de detalhe,
  - renderização de curvas.


(a)
(d)
TD
TI
compressão
(b)
(c)

Principal Contribuição

Técnicas intervalares integradas às Transformadas Discretas Wavelets

- Desenvolver a biblioteca **Int-DWTs** com as implementações intervalares das TDWs, utilizando C-XSC.
- Prover resultados intervalares confiáveis e automaticamente validados às aplicações que necessitam das Transformadas Discretas Wavelets (TDW).

Objetivo

- Estudo, análise e **implementação da extensão intervalar** das Transformadas Wavelet de Haar (TWH) e de Daubechies (TWD).
- Implementar as seguintes abordagens:
  - **Abordagens**: não-normalizada não-padrão, não-normalizada padrão, normalizada não-padrão e normalizada padrão;
  - **Etapas**: transformação direta (decomposição) e de transformação inversa (composição).

Transformada Wavelet de Haar

Cascata

... ou Decimada

Decomposição	Composição
C'[i] = (C[2i] + C[2i + 1]) / 2 C'[n/2 + i] = (C[2i] - C[2i + 1]) / 2	C'[2*i] = C[i] + C[n + i] C'[2*i + 1] = C[i] - C[n + i]

n = tamanho do vetor; i = 0... n/2 - 1;

—	(de escala)	(wavelets)
Nível	Médias	Diferenças
2	[9 7 3 5]	-
1	[8 4]	[1 -1]
0	[6]	[2]

- Vetor original: [9 7 3 5]
- Vetor transformado: [6 2 1 -1]

Extensão Intervalar do Algoritmo da TWH

```
void INT_Haar_DecompositionStep(interval *vec, int n, bool normal)
{
    interval div;
    interval *vecp = new interval[n];

    if (normal) div = sqrt(interval(2));
    else        div = interval(2);

    for (int i = 0; i < n/2; i++)
    {
        vecp[i]       = (vec[2*i] + vec[2*i + 1]) / div;
        vecp[i + n/2] = (vec[2*i] - vec[2*i + 1]) / div;
    }

    for (int i = 0; i < n; i++)
        vec[i] = vecp[i];

    delete [] vecp;
}
```

```
void Haar_Decomposition(double *vec, int n, bool normal)
{
    if (normal)
    for (int i = 0; i < n; i++)
        vec[i] = vec[i] / sqrt(float(n));

    while (n > 1)
    {
        Haar_DecompositionStep(vec, n, normal);
        n /= 2;
    }
}

void INT_Haar_Decomposition(interval *vec, int n, bool normal)
{
    if (normal)
    for (int i = 0; i < n; i++)
        vec[i] = vec[i] / sqrt(interval(n));

    while (n > 1)
    {
        INT_Haar_DecompositionStep(vec, n, normal);
        n /= 2;
    }
}
```

À-Trous

... ou Não Decimada

Decomposição	Composição
$C^j[i] = (C^{j-1}[i] + C^{j-1}[i+1])/2$ $D^j[i] = (C^{j-1}[i] - C^j[i])$	C = $C^n + \sum_{i=1}^{n} D^i$

n = tamanho do vetor; i = 0... n/2 - 1;

—	de escala (C)	wavelets (D)
Nível	**Médias**	**Diferenças**
0	[9 7 3 5]	-
1	[8 6 4 6]	[1 1 1 -3]
2	[7 5 5 7]	[1 1 -1 -1]

- Vetor original: C^0 = [9 7 3 5]
- Resultado: C^2, D^1, D^2

procedure *DecompositionStep*(C: **array** $[1\,..\,h]$ **of reals**)
  for $i \leftarrow 1$ **to** $h/2$ **do**
    $C'[i] \leftarrow (C[2i-1] + C[2i])/\sqrt{2}$
    $C'[h/2+i] \leftarrow (C[2i-1] - C[2i])/\sqrt{2}$
  end for
  $C \leftarrow C'$
end procedure

procedure *Decomposition*(C: **array** $[1\,..\,h]$ **of reals**)
  $C \leftarrow C/\sqrt{h}$ *(normalize input coefficients)*
  while $h > 1$ **do**
    DecompositionStep($C[1\,..\,h]$)
    $h \leftarrow h/2$
  end while
end procedure

Transformada Wavelet de Daubechies

$$h = \left(\frac{1+\sqrt{3}}{4\sqrt{2}}, \frac{3+\sqrt{3}}{4\sqrt{2}}, \frac{3-\sqrt{3}}{4\sqrt{2}}, \frac{1-\sqrt{3}}{4\sqrt{2}} \right) \quad (1)$$

$$g = \left(\frac{1-\sqrt{3}}{4\sqrt{2}}, \frac{-3+\sqrt{3}}{4\sqrt{2}}, \frac{3+\sqrt{3}}{4\sqrt{2}}, \frac{-1-\sqrt{3}}{4\sqrt{2}} \right) \quad (2)$$

$$ih = \left(\frac{3-\sqrt{3}}{4\sqrt{2}}, \frac{3+\sqrt{3}}{4\sqrt{2}}, \frac{1+\sqrt{3}}{4\sqrt{2}}, \frac{1-\sqrt{3}}{4\sqrt{2}} \right) \quad (3)$$

$$ig = \left(\frac{1-\sqrt{3}}{4\sqrt{2}}, \frac{-1-\sqrt{3}}{4\sqrt{2}}, \frac{3+\sqrt{3}}{4\sqrt{2}}, \frac{-1+\sqrt{3}}{4\sqrt{2}} \right) \quad (4)$$

DecompositionStep $(C[1..n])$

$i \leftarrow 1$;
$j \leftarrow 1$;
$half \leftarrow n/2$;
while $j <= n-3$ **do**
    $C'[i] \leftarrow C[j].h1 + C[j+1].h2 + C[j+2].h3 + C[j+3].h4$;
    $C'[i+half] \leftarrow C[j].g1 + C[j+1].g2 + C[j+2].g3 + C[j+3].g4$
    $j \leftarrow j+2$;
    $i \leftarrow i+1$;
end
$C'[i] \leftarrow C[n-1].h1 + C[n].h2 + C[1].h3 + C[2].h4$;
$C'[i+half] \leftarrow C[n-1].g1 + C[n].g2 + C[1].g3 + C[2].g4$;
$C \leftarrow C'$;

$$h = a\left(\frac{1+\sqrt{3}}{4\sqrt{2}}\right) + b\left(\frac{3+\sqrt{3}}{4\sqrt{2}}\right) + c\left(\frac{3-\sqrt{3}}{4\sqrt{2}}\right) + d\left(\frac{1-\sqrt{3}}{4\sqrt{2}}\right) \quad (5)$$

$$g = a\left(\frac{1-\sqrt{3}}{4\sqrt{2}}\right) + b\left(\frac{\sqrt{3}-3}{4\sqrt{2}}\right) + c\left(\frac{3+\sqrt{3}}{4\sqrt{2}}\right) + d\left(\frac{-1-\sqrt{3}}{4\sqrt{2}}\right) \quad (6)$$

$$ih = a\left(\frac{3-\sqrt{3}}{4\sqrt{2}}\right) + b\left(\frac{3+\sqrt{3}}{4\sqrt{2}}\right) + c\left(\frac{1+\sqrt{3}}{4\sqrt{2}}\right) + d\left(\frac{1-\sqrt{3}}{4\sqrt{2}}\right) \quad (7)$$

$$ig = a\left(\frac{1-\sqrt{3}}{4\sqrt{2}}\right) + b\left(\frac{-1-\sqrt{3}}{4\sqrt{2}}\right) + c\left(\frac{3+\sqrt{3}}{4\sqrt{2}}\right) + d\left(\frac{\sqrt{3}-3}{4\sqrt{2}}\right) \quad (8)$$

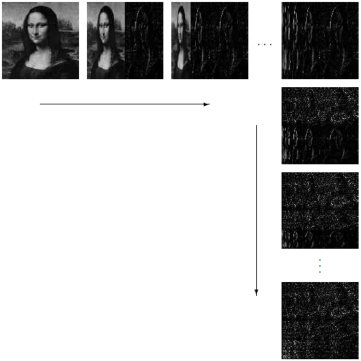
Otimização

Exemplificação

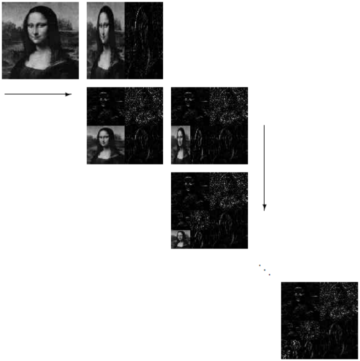
$$R = 2^{\frac{-j}{2}}$$

$$[9 \quad 7 \quad 3 \quad 5] \rightarrow [6 \quad 2 \quad 1 \quad -1] \rightarrow [6 \quad 2 \quad \frac{1}{\sqrt{2}} \quad \frac{-1}{\sqrt{2}}]$$

Entrada → Decomposição → Normalização

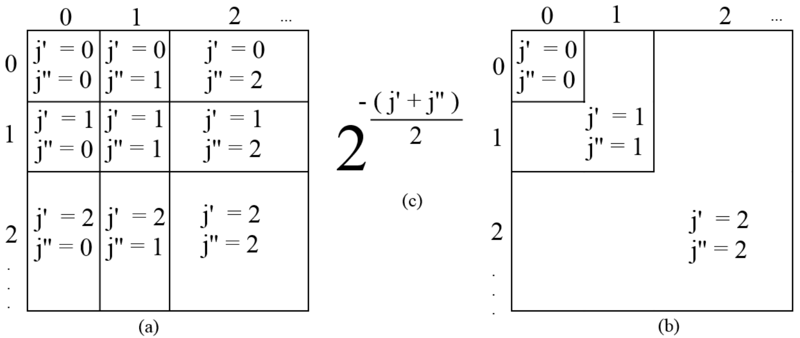


(a) TWH 2D Padrão

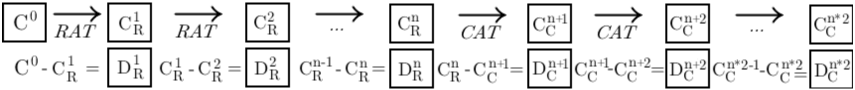


(b) TWH 2D Não-Padrão

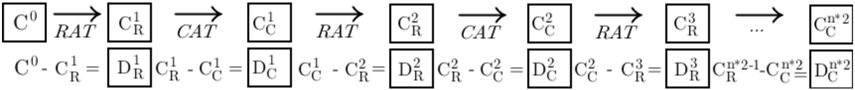
Padrão de fatores de normalização (TWH 2D Cascata)



Níveis de aplicação para (a) algorítmo Padrão, (b) algorítmo Não-Padrão e (c) regra de normalização de coeficientes bidimensionais.



TWH 2D Padrão



TWH 2D Não Padrão

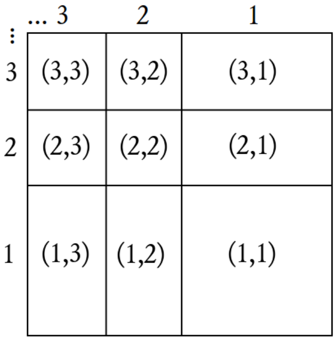
$$
\begin{aligned}
h &= a\left(\frac{1+\sqrt{3}}{4\sqrt{2}}\right)+b\left(\frac{3+\sqrt{3}}{4\sqrt{2}}\right)+c\left(\frac{3-\sqrt{3}}{4\sqrt{2}}\right)+d\left(\frac{1-\sqrt{3}}{4\sqrt{2}}\right) \\
&= \frac{a\left(1+\sqrt{3}\right)+b\left(3+\sqrt{3}\right)+c\left(3-\sqrt{3}\right)+d\left(1-\sqrt{3}\right)}{4\sqrt{2}} \\
&= \frac{a+a\sqrt{3}+3b+b\sqrt{3}+3c-c\sqrt{3}+d-d\sqrt{3}}{4\sqrt{2}} \\
&= \frac{a+d+3\left(b+c\right)+\sqrt{3}\left(a+b-c-d\right)}{4\sqrt{2}} \qquad (9)
\end{aligned}
$$

$$
\begin{aligned}
g &= a\left(\frac{1-\sqrt{3}}{4\sqrt{2}}\right)+b\left(\frac{\sqrt{3}-3}{4\sqrt{2}}\right)+c\left(\frac{3+\sqrt{3}}{4\sqrt{2}}\right)+d\left(\frac{-1-\sqrt{3}}{4\sqrt{2}}\right)\\
&= \frac{a\left(1-\sqrt{3}\right)+b\left(\sqrt{3}-3\right)+c\left(3+\sqrt{3}\right)+d\left(-1-\sqrt{3}\right)}{4\sqrt{2}}\\
&= \frac{a-a\sqrt{3}+b\sqrt{3}-3b+3c+c\sqrt{3}-d-d\sqrt{3}}{4\sqrt{2}}\\
&= \frac{a-d+3\left(c-b\right)+\sqrt{3}\left(b+c-a-d\right)}{4\sqrt{2}}
\end{aligned}
\tag{10}
$$

$$
\begin{aligned}
ih &= a\left(\frac{3-\sqrt{3}}{4\sqrt{2}}\right)+b\left(\frac{3+\sqrt{3}}{4\sqrt{2}}\right)+c\left(\frac{1+\sqrt{3}}{4\sqrt{2}}\right)+d\left(\frac{1-\sqrt{3}}{4\sqrt{2}}\right) \\
&= \frac{a\left(3-\sqrt{3}\right)+b\left(3+\sqrt{3}\right)+c\left(1+\sqrt{3}\right)+d\left(1-\sqrt{3}\right)}{4\sqrt{2}} \\
&= \frac{3a-a\sqrt{3}+3b+b\sqrt{3}+c+c\sqrt{3}+d-d\sqrt{3}}{4\sqrt{2}} \\
&= \frac{c+d+3\left(a+b\right)+\sqrt{3}\left(b+c-a-d\right)}{4\sqrt{2}} \qquad (11)
\end{aligned}
$$

$$
\begin{aligned}
ig &= a\left(\frac{1-\sqrt{3}}{4\sqrt{2}}\right)+b\left(\frac{-1-\sqrt{3}}{4\sqrt{2}}\right)+c\left(\frac{3+\sqrt{3}}{4\sqrt{2}}\right)+d\left(\frac{\sqrt{3}-3}{4\sqrt{2}}\right)\\
&= \frac{a\left(1-\sqrt{3}\right)+b\left(-1-\sqrt{3}\right)+c\left(3+\sqrt{3}\right)+d\left(\sqrt{3}-3\right)}{4\sqrt{2}}\\
&= \frac{a-a\sqrt{3}-b-b\sqrt{3}+3c+c\sqrt{3}-3d+d\sqrt{3}}{4\sqrt{2}}\\
&= \frac{a-b+3(c-d)+\sqrt{3}(c+d-a-b)}{4\sqrt{2}} \qquad (12)
\end{aligned}
$$

Padrão de fatores de normalização (TWD 2D Padrão)



(a)

$$(j', j'')$$

(b)

$$2^{-\frac{(j'+j'')}{2}}$$

(c)

(a) níveis de aplicação, (b) combinação de níveis, (c) regra de normalização.

Casos de Estudo

Estudo de caso:

- **Performance**: Relação de desempenho das execuções.
- **Accuracy**: Relação entre o maior diâmetro de intervalo de erro na computação.
- **Metrics**: Aplicação de métricas (EUC, MSE e PSNR) para medição de qualidade.

$$\mathbf{EUC}(\tilde{\mathbf{Y}}, \mathbf{Y}) = \sqrt{\sum_{j=0}^{m} \sum_{i=0}^{n} \left(\tilde{\mathbf{Y}}_{ij} - \mathbf{Y}_{ij}\right)^2} \tag{13}$$

$$\mathbf{MSE}(\tilde{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{nm} \sum_{j=0}^{m} \sum_{i=0}^{n} (\tilde{\mathbf{Y}}_{ij} - \mathbf{Y}_{ij})^2 \tag{14}$$

$$\mathbf{PSNR}(\tilde{\mathbf{Y}}, \mathbf{Y}_{ij}) = 10 \cdot \log_{10} \frac{MAX^2}{\mathbf{MSE}(\tilde{\mathbf{Y}}, \mathbf{Y})} \tag{15}$$

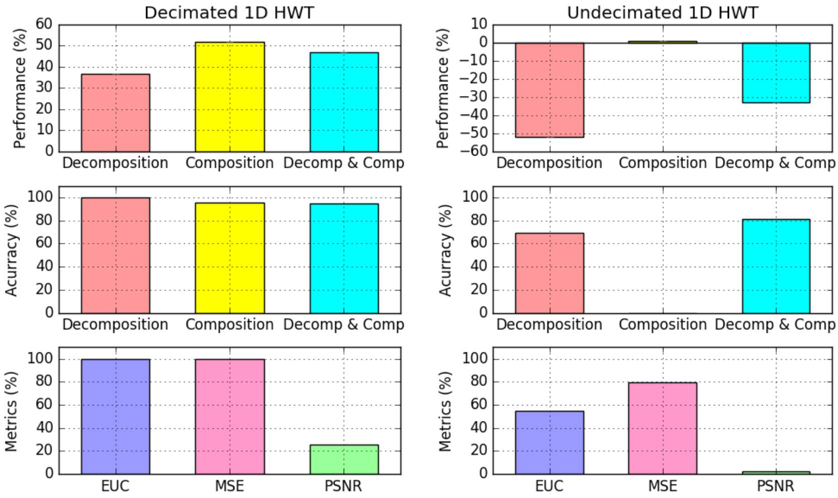
Configuração da Máquina

Desktop PC

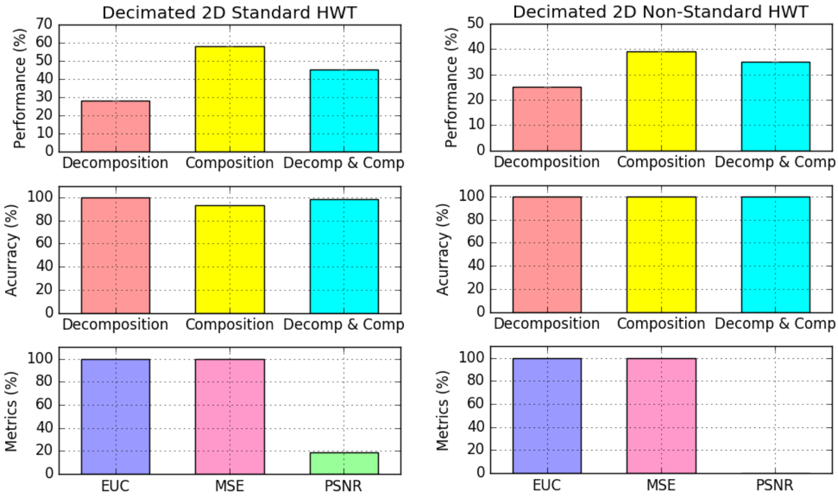
Processador:	Intel ® Core ™ i7 950 Processor @ 3.07GHz
Memória:	6GB DDR3 @ 1066MHz
SO:	Windows 10
Compilador:	Microsoft Visual C++ VS2013 (x64 Release)

Parâmetros de teste para a TWH

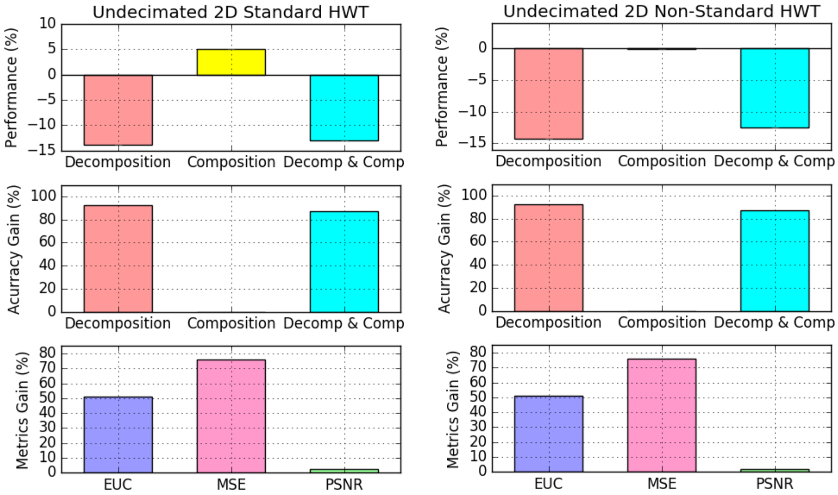
- **TWH 1D**
  - **Cascata**: 1048576 valores aleatórios
  - **À-Trous**: 1048576 valores aleatórios e 4 iterações
- **TWH 2D**
  - **Cascata**: 1024x1024 valores aleatórios
  - **À-Trous**: 1024x1024 valores aleatórios e 4 iterações



Testes para a **TWH 1D Cascata** e **À-Trous** utilizando 1048576 valores aleatórios.



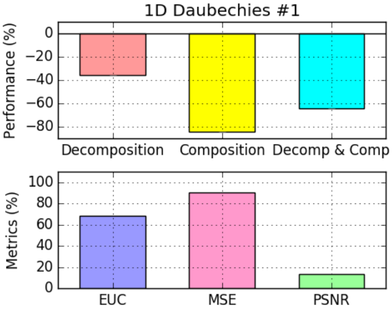
Testes para a **TWH 2D Cascata** utilizando 1024x1024 valores aleatórios.



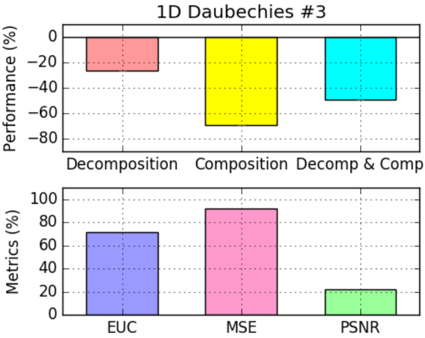
Testes para a **TWH 2D À-Trous** utilizando 1024x1024 valores aleatórios e 4 iterações.

Parâmetros de teste para a TWD

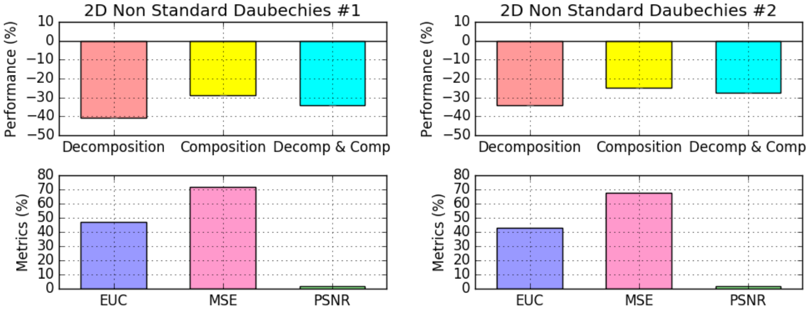
- **TWD 1D**
  - **Teste #1**: 1048576 valores aleatórios
  - **Teste #2**: 4194304 valores aleatórios
  - **Teste #3**: 16777216 valores aleatórios
- **TWD 2D**
  - **Teste #1**: 1024x1024 valores aleatórios
  - **Teste #2**: 2048x2048 valores aleatórios
  - **Teste #3**: 4096x4096 valores aleatórios



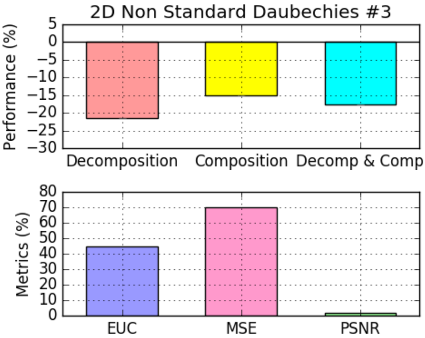
Testes #1 e #2 para a **TWD 1D** utilizando 1048576 e 4194304 valores aleatórios.



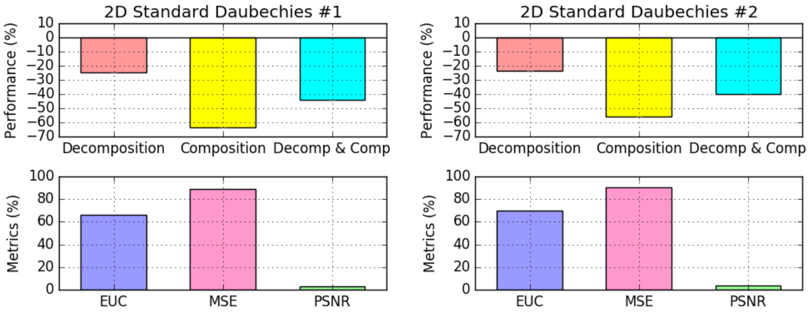
Teste #3 para a **TWD 1D** utilizando 16777216 valores aleatórios.



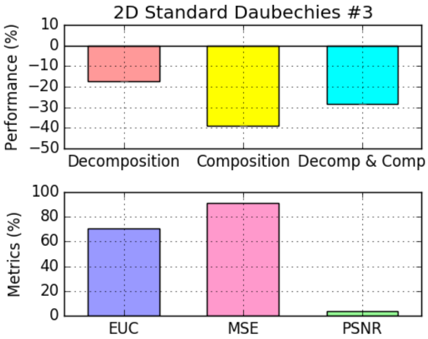
Testes #1 e #2 para a **TWD 2D Não-Padrão** utilizando 1024x1024 e 2048x2048 valores aleatórios.



Teste #3 para a **TWD 2D Não-Padrão** utilizando 4096x4096 valores aleatórios.



Testes #1 e #2 para a **TWD 2D Padrão** utilizando 1024x1024 e 2048x2048 valores aleatórios.



Teste #3 para a **TWD 2D Padrão** utilizando 4096x4096 valores aleatórios

Principais Publicações 2015-2016

- **ERAD 2015**: Estudo de Desempenho sobre a Biblioteca Int-Haar
- **SIM 2015**: Int-HWT: Increasing Performance and Exactitude of 1D and 2D Haar Wavelet Transforms
- **WEIT 2015**: Transformada de Haar Não Decimada Otimizada e sua Extensão Intervalar
- **ERAD 2016**: Transformada de Haar À Trous Otimizada e Análise de Desempenho
- **CNMAC 2016**: Otimização para Aumento de Exatidão da Transformada Wavelet de Haar À-Trous

Próximas Publicações

- **WSCAD-WIC 2016**: Simplificações de filtros e Análise de Desempenho para Transformada Wavelet de Daubechies
- Int-HWT: Algebraic Simplifications Increasing Performance And Exactitude of the Haar Wavelet Transforms

Conclusão

- As otimizações propostas para ambas transformadas foram implementadas e validadas com sucesso.
- A TWH Cascata apresentou ganhos tanto em desempenho quanto em exatidão dos cálculos.
- Ambas TWH À-Trous e TWD obtiveram perdas de desempenho porém com ganhos quanto a exatidão.
- A análise de complexidade algorítmica mostra a causa dos ganhos e perdas de desempenho.
- Não foi possível viabilizar as implementações concorrentes das transformadas utilizando GPU, porém serão os próximos objetivos do trabalho.
- Na continuidade, busca-se a extensão intervalar e otimização de outras TDWs.

Int-DWTs Library - Algebraic Simplifications Increasing Performance and Accuracy of Discrete Wavelet Transforms

Orientando: **Vinícius Rodrigues dos Santos**
Orientador: **Renata Hax Sander Reiser**
Coorientador: **Maurício Lima Pilla**
Colaborador: **Alice de Jesus Kozakevicius**

Universidade Federal de Pelotas
Ciência da Computação
Centro de Desenvolvimento Tecnológico

11/07/2015 - PELOTAS / RS

Métricas

Error

```
real INT_error(interval *x, int n)
{
    real r = 0.0;

    for (int i = 0; i < n; i++)
    if (INT_diameter(x[i]) > r) r = INT_diameter(x[i]);

    return r;
}
```

Distância Euclidiana

$$\mathbf{EUC}(\tilde{\mathbf{Y}}, \mathbf{Y}) = \sqrt{\sum_{j=0}^{m} \sum_{i=0}^{n} \left(\tilde{\mathbf{Y}}_{ij} - \mathbf{Y}_{ij}\right)^2}.$$

Mean Squared Error

$$\mathbf{MSE}(\tilde{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{nm}\sum_{j=0}^{m}\sum_{i=0}^{n}(\tilde{\mathbf{Y}}_{ij} - \mathbf{Y}_{ij})^2$$

Peak Noise-To-Signal

$$\mathbf{PSNR}(\tilde{\mathbf{Y}}, \mathbf{Y}_{ij}) = 10 \cdot \log_{10} \frac{MAX^2}{\mathbf{MSE}(\tilde{\mathbf{Y}}, \mathbf{Y})}.$$