UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Curso de Bacharelado em Ciência da Computação

Trabalho de Conclusão de Curso

Um Blabla Blablabla com Aplicações em Blablabla

Vinícius Rodrigues dos Santos

Pelotas, 2015

Vinícius Rodrigues dos Santos

Um Blabla Blablabla com Aplicações em Blablabla

Trabalho de Conclusão de Curso apresentado ao Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Renata Hax Sander Reiser
Coorientador: Prof. Dr. Alice Kozakevicius
Colaborador: Prof. Dr. Maurício Pilla

Pelotas, 2015

ABSTRACT

DOS SANTOS, Vinícius Rodrigues. **Titulo do Trabalho em Ingles**. 2015. 37 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2015.

This document describes the interval extension of the Haar Wavelet Transform (HWT) implemented using C-XSC. Both Cascade and *à trous* algorithms are extended. Optimizations for the normalized formulations of the one- and two dimensional transforms to increase speed and accuracy of the results are also presented. As an application, image compression is addressed, whose quality is computed and compared by different metrics. The error analysis of the different interval formulations as well as the speedup obtained through the interval formulations and the proposed simplifications are presented.

The final thesis will present a complete study over the Daubechies Wavelet Transform (DWT), extending it to interval mathematics, deducting simplifications and implementing optimizations based on the HWT approaches, targeting the Nvidia CUDA architecture for massive concurrence and parallelism.

Keywords: wavelet, interval, performance, optimization.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

DWTs	Discrete Wavelet Transforms
DaWT	Daubechies Wavelet Transform
HWT	Haar Wavelet Transform
EUC	Euclidean Distance
MSE	Mean Squared Error
PSNR	Peak Signal to Noise Ratio

SUMÁRIO

1 INTRODUCTION

The quality of numerical results in Scientific Computing (SC) depends on understanding the different causes of errors and controling their propagation in data, as well as improving computations in the involved procedures. This study considers the Interval Mathematics (IM) approach and proposes a solution based on Moore's arithmetic (MOORE, 1979) for the implementation of the Haar wavelet transform (HWT), considering two well established algorithms, Cascade and *à trous*, and their standard formulations assuming point values as input data.

Interval results carry over the safety of their quality together with the degree of their uncertainty (MOORE, 1979). The diameter of the interval solution contains uncertainties of input parameters, being also an indicative of the error influence and of the extension of its propagation within the incoming data. Interval solutions also indicate truncation and rounding errors contained in the computed results.

In the last decade, many studies associating wavelet transforms with interval mathematics have arisen, promoting a new research direction and pointing out the relevance of interval computations for a wide range of applications. Shu-Li et al. (SHU-LI et al., 2005) propose the interval extension of the interpolating wavelet family for the construction of an adaptive algorithm for solving partial differential equations (PDE). Another collocation method for solving PDE is presented by Liu (LIU, 2013), based on the interval extension for the Shannon-Garbor wavelet family. With respect to image processing applications, Minamoto and Aoki (MINAMOTO; AOKI, 2010) propose a blind digital image watermarking method using interval Daubechies wavelets.

The main motivation for interval techniques integrated with wavelet transforms is to provide trustworthy and validated results to technological and SC applications using such transformations. The pool of techniques involving wavelets is ample, specially in signal and image processing (H. OM, 2012; KUMAR; SUDHANSU; KASABEGOUDAR, 2012). Even the HWT, the simplest discrete wavelet transform (DWT) in terms of algorithm complexity, is still nowadays largely explored, its robustness with respect to many different mathematical structures and space formulations being a relevant aspect (NOVIKOV; SKOPINA, 2012) and explored in many applications. Besides, as

demonstrated in (BRITO; KOSHELEVA, 1998), the imaging inaccuracy of the Haar wavelet basis is the smallest possible, motivating its usage in many 2D problems and its formulation in the interval context.

Hence, this present study introduces an integrated analysis of the original HWT algorithms and the proposed interval extensions, developed as part of the library called Int-HWT, dealing with imprecise input data and calculation erros. Preliminary results of the new methods of the Int-Haar library presented in (SANTOS et al., 2013) are revisited here, including the interval extensions of the decimated HWT (STOLLNITZ; DEROSE; SALESIN, 1995) (Cascade algorithm) for the 1D and 2D cases.

The current research addresses also the *à trous* algorithm, a non-decimated HWT formulation (STARCK; FADILI; MURTAGH, 2007; KOZAKEVICIUS; SCHMIDT, 2013), proposing interval extensions for both 1D and 2D cases. Furthermore, a threshold procedure for treating signals and images, considered as a significant part of many compression and filtering algorithms, is analyzed and its interval extension is presented.

During the study of the original normalized and non-normalized formulations for the HWT (STOLLNITZ; DEROSE; SALESIN, 1995), the possibility to optimize them arises from the observation of the patterns of the transform filters in each level of decomposition. Therefore, algebraic simplifications are executed in order to eliminate operations with irrational numbers, originally responsible for the normalization of the transformation and considered in each iteration of the original algorithms. Through these simplifications, a gain of precision in the interval formulations is obtained, producing therefore more reliable results.

The C-XSC library (HOLBIG; DIVERIO; CLAUDIO, 2009) is employed to support the implementation of Int-HWT. Such library consists of a set of tools for the development of numerical algorithms, integrating precision and automatic verification of the results, frequently applied in SC. The compatibility between C++ and C-XSC library allows high level programming techniques for numerical applications (HOLBIG; JúNIOR; CLAUDIO, 2005), aggregating other available libraries to support SC.

This paper is organized in the following configuration: Section 2.1 summarizes some aspects about the interval computation, supporting the extensions proposed in the remaining of the article. Sections 2.2 and 2.3 present the decimated and undecimated formulation of the HWT. The simplifications introduced on the interval approach for obtaining a faster normalization procedure and the compression method are presented in sections 2.1.4 and 2.5, respectively. Afterwards, Section 2.4 describes the developed optimizations. Tests and results are shown in Section 2.6. Finally, conclusions and the future works are presented in Chapter 3.

2 DEVELOPMENT

2.1 Interval Arithmetics

Interval analysis was developed by Ramon Moore in the 1960's (MOORE, 1959, 1962), when computers were still at an early stage of development and every additional cost associated with keeping track of the computational errors were considered as too high. Furthermore, the produced error bounds were overly pessimistic and therefore quite useless.

Nowadays, the research for new interval methods have reached a high scientific level, producing tight error bounds faster than approximations of non-rigorous computations. Even in pure mathematics, non-trivial results have recently been proved using computer-aided methods based on interval techniques (TUCKER, 2011). Additionally, scientific computations demanding rigor as well as speed from numerical computations should be performed based on techniques for performing validated numerical calculations provided by interval methods.

Distinct approaches of arithmetic operations have been discussed in the literature, see for example (WALSTER, ????; HICKEY; JU; EMDEM., 2001) and (FIGUEIREDO; STOLFI, 2004). In this paper we consider the interval analysis proposed by Moore (MOORE, 1979) which is able to cover the development of an arithmetic for intervals, providing methods to obtain accuracy in numerical calculations.

2.1.1 Moore Interval Arithmetic Operations

In Moore arithmetic, an interval X is a continuum subset of real numbers which is in the infimum-supremum representation given as:

$$X = \{x \in \mathbb{R} \colon a \leq x \leq b, a, b \in \mathbb{R}, a \leq b\}; \tag{1}$$

and in the midpoint-radius representation, expressed by:

$$X = \{x \in \mathbb{R} \colon |x - a| \leq \alpha, a \in \mathbb{R}, 0 \leq \alpha \in \mathbb{R}\}. \tag{2}$$

The two representations are identical for real intervals but not for floating-point intervals. The infimum-supremum arithmetic is used in computers, since arithmetic based on midpoint-radius representation demands overestimation for multiplication and division. In general, it is not exactly representable in floating point causing additional computational effort. The set of all real intervals is indicated by $\mathbb{IR}$. Frequently, an interval $X \in \mathbb{IR}$ is indicated by its endpoints, $X = [a, b]$. When $a = b$ then $[a, b] \in \mathbb{IR}$ is called a degenerate interval. For an unary operator $\omega : \mathbb{IR} \to \mathbb{IR}$, we have that $\omega(X) = \{\omega(x) : x \in X\}$. For an arithmetic operation $* : \mathbb{IR}^2 \to \mathbb{IR}$ such that $* \in \{+, -, /, \cdot\}$, the following property is valid for any two intervals X and Y: $X * Y = \{x * y \colon x \in X \text{ and } y \in Y\}$.

Interval functions can be computed by performing arithmetic operations or by applying rational approximation methods (WALSTER, ????). The former identifies the class of rational interval functions and the latter, the class of irrational interval functions. Thus, Moore arithmetic guarantees correctness in the sense that any computation performed with standard floating-point methods can also be done with their interval version. By considering the set $\Re = [-\infty, \infty]$ of extended real numbers, the related family of subintervals of $[a, b]$ is given as

$$\mathbb{I}_{[a,b]} = \{[x, y] \subseteq \Re \colon a \leq x \leq y \leq b\}. \tag{3}$$

In particular, let U be the real unit interval $U = [0, 1] \subseteq \Re$. By Eq. (3), the set of all subintervals of U is indicated as $\mathbb{I}_U = \{[x, y] \subseteq \Re \colon 0 \leq x \leq y \leq 1\}$. The projections $l, r : \mathbb{I}_{[a,b]} \to [a, b]$ are respectively given by

$$l([x, y]) = x \text{ and } r([x, y]) = y, \forall [x, y] \in \mathbb{I}_{[a,b]}. \tag{4}$$

Additionally, when $X = [x, y] \in \mathbb{I}_{[a,b]}$, the projection functions $l(X)$ and $r(X)$ are also denoted by $\underline{X}$ and $\overline{X}$, respectively. Thus, by projections in Eq.(4), for all $X, Y \in \mathbb{I}_{[a,b]}$, interval arithmetic operations can also be defined as follows:

$$\begin{aligned}
X + Y &= [\underline{X} + \underline{Y}, \overline{X} + \overline{Y}];\\
X - Y &= [\underline{X} - \overline{Y}, \overline{X} - \underline{Y}];\\
1/Y &= [1/\overline{Y}, 1/\underline{Y}], \quad \text{if} \quad 0 \notin Y;\\
X \cdot Y &= [\min\{\underline{X} \cdot \underline{Y}, \underline{X} \cdot \overline{Y}, \overline{X} \cdot \underline{Y}, \overline{X} \cdot \overline{Y}\}, \max\{\underline{X} \cdot \underline{Y}, \underline{X} \cdot \overline{Y}, \overline{X} \cdot \underline{Y}, \overline{X} \cdot \overline{Y}\}].
\end{aligned}$$

Other operators considered in this study are described below: Let $X \in \mathbb{I}_{[a,b]}$ and

$n \in \mathbb{N}$. Then, the power and root operations are, respectively, defined as the following:

$$X^n = \begin{cases} [\underline{X}^n, \overline{X}^n], & \text{if } \underline{X} > 0 \text{ or } n \text{ is odd;} \\ [\overline{X}^n, \underline{X}^n], & \text{if } \overline{X} < 0 \text{ and } n \text{ is even;} \\ [0, \left(\max\{\underline{X}, \overline{X}\}\right)^n], & \text{if } 0 \in X \text{ and } n \text{ is even.} \end{cases}$$

$$\sqrt[n]{X} = \begin{cases} [\sqrt[n]{\underline{X}}, \sqrt[n]{\overline{X}}], & \text{if } \underline{X} > 0 \text{ or } n \text{ is odd;} \\ \text{indefined,} & \text{otherwise.} \end{cases}$$

The partial order used here in the context of interval mathematics is the Product (or Kulisch-Miranker) order and, for all $X, Y \in \mathbb{I}_{[a,b]}$, it is defined as follows:

$$X \leq_{\mathbb{I}_{[a,b]}} Y \text{ iff } \underline{X} \leq \underline{Y} \text{ and } \overline{X} \leq \overline{Y}. \tag{5}$$

Additionally, an interval function preserving the partial order $\leq_{\mathbb{I}_{[a,b]}}$ is called $\mathbb{I}_{[a,b]}$-monotonic function with respect to the partially ordered set $(\mathbb{I}_{[a,b]}, \leq_{\mathbb{I}_{[a,b]}})$.

2.1.2 Interval Extensions and Representations

It is possible to extend all standard (exponential, logarithm, trigonometric) real function f to an interval function F. As pointed out in (TUCKER, 2011), elementary functions, which are obtained by combining constants, arithmetic operations and compositions of standard functions, do not provide, in general, well-defined interval extensions. This can be observed for a set of singular points of a real function f, which frequently can be included in the corresponding natural extension. However, the interval evaluation $F(X)$ produces the exact range of f over the domain $X \in \mathbb{I}_{[a,b]}$.

By the Fundamental Theorem of Interval Analysis(TUCKER, 2011), let f be an elementary function and F be an interval extension such that $F(X)$ is well-defined, for $X \in \mathbb{I}_{[a,b]}$. Then, F satisfies the following properties:

(i) Inclusion Monotonicity: $Z \subseteq Z' \subseteq X \rightarrow F(Z) \subseteq F(Z') \subseteq F(X)$;

(ii) Range Enclosure: $f[X] = f(X)$.

An interval $X \in \mathbb{I}_{[a,b]}$ is said to be an interval representation of a real number α, if $\alpha \in X$. This notion can be easily extended for tuples of n intervals $(\vec{X}) = (X_1, \ldots, X_n)$. In (MOORE, 1959), a function $F : \mathbb{I}^n_{[a,b]} \rightarrow \mathbb{I}_{[a,b]}$ is an **interval representation** of a function $f : [a,b]^n \rightarrow [a,b]$ if, for each $\vec{X} \in {\mathbb{I}_{[a,b]}}^n$ and $\vec{x} \in \vec{X}$, $f(\vec{x}) \in F(\vec{X})$. From the previous paragraph, an interval function may be seen as a representation of a subset of real numbers X defined as the range $f(X)$. Considering $F : \mathbb{I}^n_{[a,b]} \rightarrow \mathbb{I}_{[a,b]}$, let $\underline{F}, \overline{F} : U^n \rightarrow U$ be the functions respectively given by

$$\underline{F}(x_1, \ldots, x_n) = l(F([x_1, x_1], \ldots, [x_n, x_n])) \tag{6}$$

$$\overline{F}(x_1, \ldots, x_n) = r(F([x_1, x_1], \ldots, [x_n, x_n])). \tag{7}$$

The fundamental theorem of interval arithmetic implicitly rests on the notion of correctness. It means that, when F is an inclusion monotonic interval extension of a real function f, then $f(X_1, \ldots, X_n) \subseteq F(X_1, \ldots, X_n)$. Thus, $F(X)$ contains the range $f(X)$ and consequently, the class of inclusion monotonic interval functions have the property of correctness. Moreover, since all arithmetic operations are inclusion monotonic functions, it holds that any rational interval function is also correct. In this paper, we consider the interval extension of three real functions, which are studied in the following.

2.1.3 Interval Metrics

Not only interval algorithms but also metrics for evaluating procedure's accuracy can assume concepts from Interval Mathematics (MOORE, 1979) to perform interval analysis efficiently managing computation errors. The motivation to consider numerical intervals instead of simple punctual values is linked to the capability of intervals to represent infinite punctual values. This sort of representation is very useful in SC when the accuracy of the input (or output) data is not granted. In these cases of uncertainty or inaccuracy the interval procedures should ensure that all possible punctual results belong to the interval results. In addition, due to memory limitation, it is also common to compute round (or simply truncated) values to store the result afterwards. This heuristic may result in different values, depending on the machine's configuration where the program has been executed.

The following 3 subsections will present the metrics being used to measure result quality.

2.1.3.1 Euclidean Distance

Let $\tilde{Y} = (\tilde{y})_{ij}, Y = (y)_{ij} \in \mathbb{R}^{n \times m}$ be multi-dimensional real matrices. $\tilde{Y} = (\tilde{y})_{ij} \in \mathbb{R}^{n \times m}$ is called an estimator of Y, a matrix of nm predictions $\tilde{y}_{ij}$ of the original values $(y)_{ij}$. The Euclidean distance between Y and its estimator $\tilde{Y}$ is defined by the following expression:

$$D(\tilde{Y}, Y) = \sqrt{\sum_{j=0}^{m} \sum_{i=0}^{n} (\tilde{y}_{ij} - y_{ij})^2}. \tag{8}$$

Analogously, $\tilde{\mathbb{Y}} = (\tilde{\mathbf{Y}})_{ij}, \mathbb{Y} = (\mathbf{Y})_{ij} \in \mathbb{IR}^{n \times m}$ are defined as being multi-dimensional real-interval matrices such that $\tilde{\mathbb{Y}}$ is called an interval estimator, an interval matrix of nm interval predictions $\tilde{\mathbf{Y}}_{ij}$ of the original interval quantities $\mathbf{Y}_{ij}$. An interval extensional of the Euclidean distance is given by the expression below:

$$\mathbf{D}(\tilde{\mathbb{Y}}, \mathbb{Y}) = \sqrt{\sum_{j=0}^{m} \sum_{i=0}^{n} \left(\tilde{\mathbf{Y}}_{ij} - \mathbf{Y}_{ij} \right)^2}. \tag{9}$$

2.1.3.2 Mean Squared Error

The Mean Squared Error (MSE) is considered as a risk function, corresponding to the expected value of the squared error loss. An MSE operator is an estimator measuring the average of the squares of the errors in SC, providing the difference between the estimator and what is estimated. MSE allows us to compare the pixel values of our original image to our degraded (noise) image. Additionally, The error is the amount by which the values of the original image differ from the degraded image.

For all $Y = (y)_{ij} \in \mathbb{R}^{n \times m}$ corresponding to the related matrix of true values y_{ij}, the accuracy of the estimator $\tilde{Y} = (\tilde{y})_{ij}$ can be obtained by application of an MSE operator as the following

$$MSE(\tilde{Y}, Y) = \frac{1}{mn} \sqrt{\sum_{j=0}^{m} \sum_{i=0}^{n} (\tilde{y}_{ij} - y_{ij})^2} = \frac{1}{mn} D(\tilde{Y}, Y). \tag{10}$$

Analogously, for all $\mathbb{Y} = (\mathbf{Y})_{ij} \in \mathbb{IR}^{n \times m}$ corresponding to the related matrix of true values $\mathbf{Y}_{ij}$, the accuracy of the estimated values can be obtained by application of an interval extension of the MSE operator in Eq. (10), which is given by the following expression:

$$\mathbf{MSE}(\tilde{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{mn} \sqrt{\sum_{j=0}^{m} \sum_{i=0}^{n} (\tilde{\mathbf{Y}}_{ij} - \mathbf{Y}_{ij})^2} = \frac{1}{mn} \mathbf{D}(\tilde{\mathbb{Y}}, \mathbb{Y}). \tag{11}$$

2.1.3.3 Peak signal-to-noise ratio

An important performance metric for evaluation and comparison of image or video codecs is the Rate/Distortion (R/D) where quality is measured in terms of Peak Signal-to-Noise Ratio ($PSNR$). Such metric is expressed as the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation.

$PSNR$ is usually expressed in terms of the logarithmic decibel scale and most commonly used to measure the quality of reconstruction of lossy compression codecs. In image compression, the signal is the original data, and the noise is the error introduced by compression. However, the range of validity of this metric is limited since it is only conclusively valid when used to compare results from the same content. Thus, the expression of $PSNR$ is most easily defined via the logarithmic decibel scale related to the MSE as follows:

$$PSNR(\tilde{Y}, Y) = 10 \cdot \log_{10} \frac{MAX_I^2}{MSE(\tilde{Y}, Y)}, \tag{12}$$

when I indicates a mn-dimensional monochrome image associated to Y and MAX_I is

the maximum possible pixel value of the image I.

Thus, a natural interval extension of Eq. (12) is given as:

$$\mathbf{PSNR}(\tilde{\mathbf{Y}}, \mathbf{Y}) = 10 \cdot \log_{10} \frac{[MAX_I, MAX_I]^2}{\mathbf{MSE}(\tilde{\mathbf{Y}}, \mathbf{Y})}, \tag{13}$$

when $[MAX_I, MAX_I]$ denotes the degenerate interval obtained by MAX_I and $\mathbf{MSE}(\tilde{\mathbf{Y}}, \mathbf{Y})$ is the interval extension of $MSE(\tilde{Y}, Y)$.

Finally, it should be noticed that a computation of the MSE between two identical images , the value will be zero and hence the PSNR will be undefined. Moreover, as the main limitation, both metrics relies strictly on numerical comparison, on which exactly focuses our study in this paper.

2.1.4 C-XSC Library

To make interval results able to contain all possible punctual results, algorithms performing only interval arithmetic have to be designed. In this context, the HWT and its many formulations are extended according to the interval arithmetic, considering the C-XSC interval library (WEBLINK TO C-XSC, ????).

C-XSC is an extensive C++ class library for SC including a set of basic data types (from intervals to multiple precision complex intervals) whose high accuracy computation must be available for some basic arithmetic operations, mainly the operations that accomplish the summation and (optimize) dot product. Therefore, some concepts of Mathematics of Computation and Computational Arithmetic have been incorporated into this system, such as high accuracy arithmetic, interval mathematics and automatic numerical verification. Besides, C-XSC is an open source library and available from (WEBLINK TO C-XSC, ????) adding additional packages in the source code (HOFSCHUSTER; KRAMER; NEHER, 2008).

Function and operator overloading allow the common mathematical notation of expressions involving interval types. This is also true for (interval) matrix/vector expressions. Numerical verification methods, also called self-validating methods, are constructive and they allow to handle uncertain data with mathematical rigor. The C-XSC library provides support for users to develop efficient numerical verification methods to obtain correct and self-validating numerical applications.

In the last years significant improvements have been made in parallelized versions of interval linear system solvers supplied by C-XSC, for more details see (PBLAS , PARALLEL BASIC LINEAR ALGEBRA SUBPROGRAMS(????; KOLBERG; FERNANDES; CLAUDIO, 2008; MILANI; KOLBERG; FERNANDES, 2010). Once the interval extensions allow different error computations, different compression algorithms can be derived, as shown in Section 2.5.

2.2 Int-HWT - Decimated Formulation

The Haar basis was proposed in 1910, introduced by the Hungarian mathematician Alfred Haar (HAAR, 1910) in a diferent approach than the one considered by Daubechies in 1988, when she presented her orthogonal basis with compact support for the space of square integrable functions(DAUBECHIES, 1992), called wavelets. Nevertheless, through her work, the Haar functions started to be seen as a particular case of orthogonal wavelets. Nowadays, the HWT is well stated with some different fast algorithm formulations, been well established for many applications, specially those involving data compression, as considered by JPEG-2000 (CHRISTOPOULOS; SKODRAS; EBRAHIMI, 2000).

2.2.1 One Dimensional Decimated Int-HWT

According to the description in (STOLLNITZ; DEROSE; SALESIN, 1995), when assuming the decimated formulation of the transform and given an initial vector C with n punctual values, the one-dimensional HWT calculates the averages and differences of each pair of adjacent elements in the input vector C, $(C_{2j-1}, C_{2j}), j = 1, ..., n/2$, generating therefore two output sets: one for the scaling (averages) and another for the wavelets (differences) coefficients, both with $n_1 = n/2$ points, half size of the original input vector C.

Figure 1 presents the $log_2(n)$ level decomposition of the Int-HWT algorithm, formulated for the decimated case as an extension of the original algorithm presented in (STOLLNITZ; DEROSE; SALESIN, 1995). The novelty here on the interval code remains in the definition of vectors as interval vectors. Constants are defined as interval quantities and all arithmetics are treated as interval operations. In this sense, in Figure 1 the interval quantities $h = (1/\sqrt{[2;2]}, 1/\sqrt{[2;2]})$ and $g = (1/\sqrt{[2;2]}, -1/\sqrt{[2;2]})$ are the normalized interval filters associated to the Int-HWT, representing the interval extension of the standard algorithm. The discrete convolution from C with h generates the averages (scaling coefficients) and for computing the differences (wavelet coefficients) the filter g is considered.

Since the HWT main feature is to decompose information in many resolution levels, assuming as input the set of the scaling coefficients that contains the n_1 previously computed averages, the next level of the decimated HWT produces a second pair of half-sized vectors, one for the new $n_2 = n_1/2$ averages and other for the corresponding $n_2 = n_1/2$ wavelet coefficients. This procedure is recursively defined and can be applied until a specific level of coarser resolution is achieved or until a single scalar scaling coefficient is obtained. The whole process of the direct HWT is also called **decomposition**.

Figure 2 shows the comparison between the standard and the interval procedures

Decomposition $(C[1..n])$
1 $C \leftarrow C/\sqrt{[2;2]}$;
2 $i \leftarrow n$;
3 **while** $i > 1$ **do**
4 $\quad DecompositionStep(C[1..i])$;
5 $\quad i \leftarrow i/2$;
6 **end**

DecompositionStep $(C[1..n])$
1 **for** $i \leftarrow 1$ ***to*** $n/2$ **do**
2 $\quad C'[i] \leftarrow (C[2i-1] + C[2i])/\sqrt{[2;2]}$;
3 $\quad C'[i+n/2] \leftarrow (C[2i-1] - C[2i])/\sqrt{[2;2]}$;
4 **end**
5 $C \leftarrow C'$;

Figura 1: 1D decimated Int-HWT: $l = log_2(n)$ decomposition levels. Normalized interval filters: $h = (1/\sqrt{[2,2]}, 1/\sqrt{[2,2]})$ for computing scaling coefficients; $g = (1/\sqrt{[2,2]}, -1/\sqrt{[2,2]})$ for wavelet coefficients.

of the one-dimensional HWT assuming $l = log_2 n$ decomposition levels. It is possible to

```
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;
    }
}
```

Figura 2: 1D decimated HWT algorithms:(top panel) original; (bottom panel) interval extension. Both cases assuming initial data as being punctual values and $log_2 n$ decomposition levels.

notice in Figure 2 the care with the type of data being used in the formal parameters in both procedures. The constants must be expressed by intervals too. The instruction $sqrt(float(n))$, square root of punctual floating value, becames $sqrt(interval(n))$, square root of punctual interval. Inside the while loop the interval procedure calls another interval procedure, maintaining the compatibility of the data being calculated. The same type of code adaptation is considered for the rest of the punctual procedures, generating in this way the proposed interval extension.

Using the sets of all wavelet coefficients - resulted from the decomposition pro-

cess - together with the averages in the lowest level of the direct decimated Int-HWT, it is possible to accurately perform the inverse transformation (also called composition), generating the exact reconstruction of the original vector, which is an important characteristic of the HWT.

The pseudo-code that performs the level composition of the 1D decimated Int-HWT is represented in Figure 3. According to (STOLLNITZ; DEROSE; SALESIN, 1995), in the composition process, starting from the coarsest resolution level of the transformation, the original values are restored level by level, combining the wavelet and scaling coefficients from the level immediately below. Therefore, the decomposition process is completely reversible, allowing the data reconstruction in each level of the transformation, until the end of the process is achieved, when finally the finest resolution level is exactly reconstructed.

```
Composition (C[1..n])
1 i ← 2 ;
2 while i < n do
3 |   CompositionStep(C[1..2i]) ;
4 |   i ← 2i ;
5 end
6 C ← C * √[2;2] ;
```

```
CompositionStep (C[1..n])
1 for i ← 1 to n/2 do
2 |   C'[2i − 1] ← (C[i] + C[i + n/2])/√[2;2] ;
3 |   C'[2i] ← (C[i] − C[i + n/2])/√[2;2] ;
4 end
5 C ← C' ;
```

Figura 3: Inverse 1D decimated Int-HWT. Composition process to reconstruct $l = log_2 n$ levels.

2.2.2 Two Dimensional Decimated Int-HWT

The HWT can be extended for two-dimensional vectors. According to the procedure described in (STOLLNITZ; DEROSE; SALESIN, 1995), the fast algorithm for the 2D decimated HWT is obtained through the application of the one-dimensional transformation per direction (in all rows of the input matrix and after that, in all columns of the resulting one). In fact, the order how the many levels of the one-dimensional transformation are applied to the two-dimensional data generates different intermediate results and, therefore, distinct algorithms for the 2D transform.

Following what was presented in (STOLLNITZ; DEROSE; SALESIN, 1995), the 2D decimated Int-HWT can be calculated through the standard method, showed in Figure 4. In this formulation, the many levels of the one-dimensional transform are applied to all rows. After the $l = log_2 n$ decomposition levels of the 1D HWT were applied to the entire set of rows of the matrix, the same 1D procedure is applied to the columns to the resulting matrix, as many levels as done for the rows. This completes the 2D decimated transformation, assuming $l = log_2 n$ decomposition levels. Figure 5 illustrates the application of this algorithm.

According to (STOLLNITZ; DEROSE; SALESIN, 1995) and shown in Figure 4,the

```
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 4: 2D decimated HWT: Standard and non-standard decomposition processes.

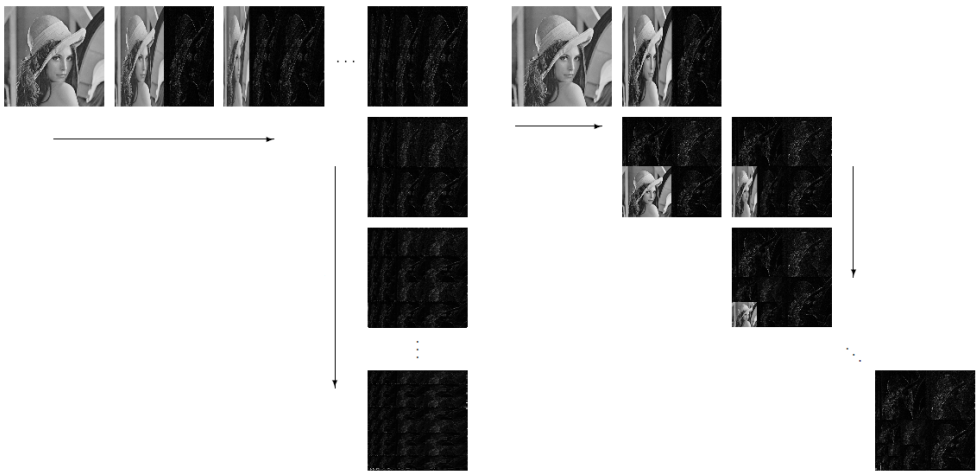

Figura 5: 2D decimated HWT with $n \geq 3$ decomposition levels.(left panel)standard algorithm; (right panel) non-standard method.

non-standard method is another option for the 2D decimated HWT, whose interval extension is also proposed here. In this formulation the operations by rows and columns through out each level of the transform are intercalated. Thereby, every row is decomposed in one level and right after that all columns are also decomposed in one level, completing one level of decomposition for the entire 2D data. In this non standard formulation of the 2D HWT, the resulting scaling coefficients after one decomposition level, which are one fourth of the entire initial data, are then considered as the input for the computation of the next decomposition level of the transform. The three remaining blocks with wavelet coefficients are not further decomposed. Each one also containing one fourth of the original data. The complete process repeats itself until in the last level just a single scaling coefficient and just one wavelet coefficient for the remaining three sets are obtained. A representation of this formulation is presented in Figure 5(right panel).

The direct and inverse Int-HWT, Figures 1, 3, 4 and 6, can be thought as a convolution of the input data with a pair of filters , $h = (1/2, 1/2)$ and $g = (1/2, -1/2)$, that represent the transform (STOLLNITZ; DEROSE; SALESIN, 1995). The normalized

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

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

Figura 6: 2D decimated HWT - composition process: (left panel) standard formulation; (right panel) non-standard formulation.

Haar filters are $h = (1/\sqrt{2}, 1/\sqrt{2})$ and $g = (1/\sqrt{2}, -1/\sqrt{2})$ and the normalized transform preserves the energy of the entire set of coefficients obtained through the HWT. Therefore, according to (STOLLNITZ; DEROSE; SALESIN, 1995), besides the standard and non-standard algorithms, the HWT can be performed using normalized and non-normalized filters. Therefore combining the two possible algorithms with these two possible filter choices, there are four ways to compute the decimated 2D Int-HWT available: (i) non-standard, non-normalized; (ii) non-standard, normalized; (iii) standard, non-normalized; and (iv) standard, normalized.

2.3 Int-HWT - Undecimated Formulation

Another alternative version for the HWT is the undecimated approach, which avoids the decimation operation after the convolution with the filters has been computed. This undecimated transform has one well established formulation, called the *à trous* algorithm, which is considered in many astrophysical and statistical applications (STARCK; FADILI; MURTAGH, 2007; KOZAKEVICIUS; SCHMIDT, 2013), specially for been a translation invariant transform.

2.3.1 One Dimensional Undecimated Int-HWT

According to what is shown in (STARCK; FADILI; MURTAGH, 2007), given an initial vector C^0 with $n_0 = 2^J$ punctual values, the vector C^1 that contains the first decomposition level of the 1D undecimated HWT (à-trous HWT) stores the averages for each pair of adjacent elements $(C^0_j, C^0_{j+1}), j = 1, ..., n_0$, (assuming $C^0_{n_0+1} = C^0_0$). The difference now with respect to the Cascade formulation is that the range of j to select the values C^0_j covers all positions $j = 1, ..., n_0$. This is exactly what avoids the decimation step and causes C^1 to have the same size as C^0. The way the wavelet coefficients are computed also differs from the previous formulation. Now the vector with the differences D^1 is computed by $D^1_j = C^0_j - C^1_j,\ j = 1, ..., n_0$. The *à trous* HWT is again constructed recur-

sively, until a certain decomposition level $0 < l < log_2 n_0$ is achieved. In this sense, the vector of averages C^j at a level $0 < j \leq l$ is obtained from the set C^{j-1}, the averages at the previous level. Analogously, the vector with the wavelet coefficients is given by $D^j = C^{j-1} - C^j$. This process is again called Decomposition or Direct Transformation and it is illustrated in Figure 7 .

$$C^0 = [9 \quad 7 \quad 5 \quad 3] \quad \rightarrow \quad \begin{matrix} C^1 = [8 & 6 & 4 & 6] \\ D^1 = [1 & 1 & 1 & -3] \end{matrix} \quad \rightarrow \quad \begin{matrix} C^2 = [7 & 5 & 5 & 7] \\ D^2 = [1 & -1 & -1 & 1] \end{matrix} \quad \rightarrow \quad ...$$

Figura 7: 1D à trous HWT: example with 2 decomposition levels

```
Decomposition (Input[1..n], Levels)
1 Declare C[1..Levels][1..n] ;
2 Declare D[1..Levels][1..n] ;
3 Declare LastC[1..n] ← Input ;
4 for j ← 1 to Levels do
5 |   DecompositionStep(C[j], D[j], LastC) ;
6 |   LastC ← C[j] ;
7 end
8 return C, D ;
```

```
DecompositionStep (C[1..n], D[1..n], LastC[1..n])
1 for i ← 1 to n − 1 do
2 |   C[i] ← (LastC[i] + LastC[i + 1])/√[2; 2] ;
3 |   D[i] ← LastC[i] − C[i] ;
4 end
5 C[i] ← (LastC[n] + LastC[0])/√[2; 2] ;
6 D[i] ← LastC[n] − C[n] ;
```

Figura 8: 1D à-Trous HWT- Decomposition algorithm.

Summing all vectors of wavelet coefficients resulted by the n decomposition levels together with the averages in the lowest level n, it is possible to perform, accurately, the Inverse Transformation, $C^0 = C^n + \sum_{i=1}^{n} D^i$, also called Composition, which provides the exact reconstruction of the original input vector of data, being an important characteristic of Wavelet Transforms. The composition pseudo code is shown in Figure 9.

```
Composition (C[1..n], D[1..Levels][1..n])
1 Declare Result[1..n] ← C ;
2 for i ← 1 to Levels do
3 |   Result ← Result + D[i] ;
4 end
5 return Result ;
```

Figura 9: 1D à-Trous HWT- Composition algorithm.

2.3.2 Two Dimensional à trous algorithm

According to (STARCK; FADILI; MURTAGH, 2007), the algorithm for the 2D à trous HWT is again obtained through the application of the one-dimensional transformation in

all rows and columns of the input matrix, as done for the decimated version of the HWT. Analogously, Standard and Non-Standard procedures can be designed according to the order how the 1D à trous transforms are applied to the 2D data.

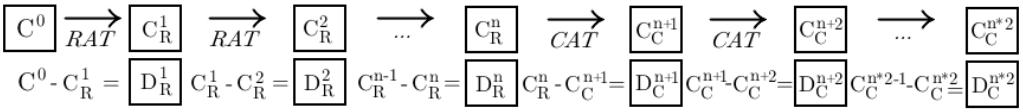

Figura 10: 2D à trous HWT- standard algorithm for matrix decomposition.

The undecimated standard decomposition for the 1D Int-HWT, based on the Standard decimated decomposition, is performed by executing the 1D à trous Int-HWT for each row of the input matrix. This process is called Row À-Trous Transform (*RAT*), characterizing this procedure as a horizontal transformation. Considering n decomposition levels, the RAT procedure is repeated n times, generating matrices C_R^1 ... C_R^n and D_R^1 ... D_R^n. After decomposing all rows from the input matrix, the decomposition of all its columns is performed, called Column À-Trous Transform (*CAT*), characterizing a vertical transformation and generating matrices C_C^{n+1} ... C_C^{n*2} and D_C^{n+1} ... D_C^{n*2}.

The matrices C_R^j and C_C^j store scaling coefficients, while D_R^j and D_C^j store wavelet coefficients. A visual representation of the process is shown in Figure 10, and its pseudo algorithm is shown in Figure 13.

```
StandardDecomposition (Input[1..n][1..m], Levels)
1  Declare C[1..Levels * 2][1..n][1..m] ;
2  Declare D[1..Levels * 2][1..n][1..m] ;
3  Declare C'[1..Levels][1..n] ;
4  Declare D'[1..Levels][1..n] ;
5  for i ← 1 to n do
6  |   C', D' ← Decomposition(Input[i][1..m], Levels) ;
7  |   for j ← 1 to Levels do
8  |   |   C[j][n][1..m] ← C'[j][1..m] ;
9  |   |   D[j][n][1..m] ← D'[j][1..m] ;
10 |   end
11 end
12 for i ← 1 to m do
13 |   C', D' ← Decomposition(C[Levels][1..n][i], Levels) ;
14 |   for j ← Levels to Levels * 2 do
15 |   |   C[j][1..n][i] ← C'[j][1..n] ;
16 |   |   D[j][1..n][i] ← D'[j][1..n] ;
17 |   end
18 end
19 return C, D ;
```

Figura 11: 2D à trous HWT: Pseudo code for the standard matrix decomposition.

The undecimated non-standard decomposition, based on the original non-standard algorithm from the Cascade approach, is performed by intercalating both *RAT* and *CAT*

operations, extracting vertical and horizontal details for each level of decomposition, as shown in Figure 12. For each level j of decomposition the algorithm performs a *RAT* step, generating matrices C_R^j and D_R^j, and right after a *CAT* is performed, generating matrices C_C^j and D_C^j. Both C_R^j and C_C^j store scalar coefficients, while D_R^j and D_C^j hold wavelet coefficients, and those four matrices represent one level full decomposed. The À-trous Non Standard algorithm is shown in Figure 13

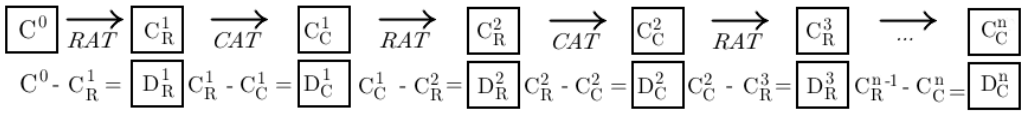

Figura 12: Non Standard algorithm for matrix decomposition.

```
NonStandardDecomposition (Input[1..n][1..m], Levels)
1 . . .
2 return C, D ;
```

Figura 13: Pseudo code for the Non Standard matrix decomposition.

The same idea from the one-dimensional transform can be applied for both standard and non-standard sets of data to perform an inverse process, generating the input matrix. By using all matrices of wavelet coefficients resulted by the process of decomposition together with the averages in the lowest level of decomposition, it is possible to reconstruct, accurately. $MatrixComposition$ generates the exact reconstruction of the original input matrix of data. The algorithm describing its execution is shown in Figure 14.

```
MatrixComposition (C[1..n][1..m], D[1..Levels][1..n][1..m])
1 Declare Result[1..n][1..m] ← C ;
2 for i ← 1 to Levels do
3 |   Result ← Result + D[i] ;
4 end
5 return Result ;
```

Figura 14: Pseudo code for À-trous matrix composition.

2.4 Optimal Interval Formulations

When the normalized filters are considered in the original algorithms of the HWT (STOLLNITZ; DEROSE; SALESIN, 1995; STARCK; FADILI; MURTAGH, 2007), divisions by $\sqrt{2}$ are executed in all iterations within the decomposition. Once $\sqrt{2}$ is not a computable value, each level of the decomposition or composition adds a certain degree of error into the data. The error generated in each iteration is then propagated through all levels until the end of the procedure.

The solution proposed to avoid this issue for the interval extension is based on performing algebraic simplifications to eliminate the computation of these non computable values, $2^{j/2}$, whenever possible, reducing the error involved in the process. Therefore the developed interval procedures produce more trustworthy results in comparison with the original algorithms analyzed here (STOLLNITZ; DEROSE; SALESIN, 1995) and (KOZAKEVICIUS; SCHMIDT, 2013).

The original algorithms and their interval versions developed here are compared. Assuming the same input data is analyzed with all different formulations of the HWT: the Cascade (decimated) and à-trous (undecimated) algorithms for both normalized and non-normalized approaches.

The basic idea for the algebraic simplifications is based on using the non-normalized transform, which introduces no extra computation errors for the HWT. By shifting the normalization step to the end of the transform, all operations with $\sqrt{2}$ are applied only once, decreasing the error produced by its computation in each level of the decomposition.

2.4.1 1D Decimated Int-HWT Simplification

The normalized decomposition procedure for the decimated version of the Int-HWT, either 1D or 2D, is executed in the following order: first a non-normalized decomposition is made, as indicated by the pseudo-code in Figure 15; after this stage the normalization of all coefficients is performed, multiplying all of them by the normalization factor $2^{-j/2}$, where j is the resolution level of the coefficients through out the decomposition.

Depending on j, the computation of $\sqrt{2}$ may not be necessary, avoiding any computation error in this case. This process is illustrated in Figure 15, assuming 2 levels of the 1D HWT and considering the same example as presented in (STOLLNITZ; DEROSE; SALESIN, 1995).

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

Input $\rightarrow$ **Decomposition** $\rightarrow$ **Normalization**

Figura 15: Example of the one-dimensional optimization.

Looking at the original decomposition procedure, shown in Figure 1, the complexity of $DecompositionStep$ is $\mathrm{O}(n+1)$, and for the main $decomposition$ algorithm the complexity is $\mathrm{O}(n + \log(n) * (n+2) + 1)$. By the proposed simplification they are reduced to $\mathrm{O}(n)$ and $\mathrm{O}(n + n\log(n))$, respectively. By performing the non-normalized decomposition there is no need to execute the normalization step at line 1 of the original $decomposition$ algorithm, neither the divisions by $\sqrt{2}$ at its step procedure. Therefore, overall complexity for non-normalized transformation of a n sized vector is $\mathrm{O}(n\log(n))$, representing the pseudo-code shown in Figure 16.

```
Decomposition (C[1..n])
1 i ← n ;
2 while i > 1 do
3 |   DecompositionStep(C[1..i]) ;
4 |   i ← i/2 ;
5 end
```

```
DecompositionStep (C[1..n])
1 for i ← 1 to n/2 do
2 |   C'[i] ← (C[2i − 1] + C[2i])/[2; 2] ;
3 |   C'[i + n/2] ← (C[2i − 1] − C[2i])/[2; 2] ;
4 end
5 C ← C' ;
```

Figura 16: Int-HWT, decimated version: non-normalized Decomposition process and its step procedure.

The normalization procedure, presented in Figure 17, performs the normalization step of all coefficients after their convolution using filters. Since it is executed once on every coefficient of a n sized vector, its complexity can be $\mathrm{O}(n)$. By performing a non-normalized transform and the normalization procedure afterwards, the complexity is $\mathrm{O}(n\log(n)+n)$, which is the same complexity from the normalized transformation. The goal is therefore the gain on exactitude obtained by avoiding high number of divisions using $\sqrt{2}$ from the original $DecompositionStep$ algorithm, as shown in Figure 26.

```
Normalization (C[1..n])
1 levels ← log2(n) ;
2 for i ← 1 to levels do
3 |   start ← i² ;
4 |   end ← (i + 1)² ;
5 |   for j ← start to end do
6 |   |   C[j] ← C[j] ∗ 2 ∧ (−(i/2)) ;
7 |   end
8 end
```

```
Denormalization (C[1..n])
1 levels ← log2(n) ;
2 for i ← 1 to levels do
3 |   start ← i² ;
4 |   end ← (i + 1)² ;
5 |   for j ← start to end do
6 |   |   C[j] ← C[j] / 2 ∧ (−(i/2)) ;
7 |   end
8 end
```

Figura 17: Int-HWT, decimated version: on dimensional normalization procedure.

The normalized composition procedure follows the same basic idea to reduce computation of $\sqrt{2}$, it performs the non-normalized inverse transform, but this time a denormalization step, shown in Figure 17, is executed before the transform begins. The original algorithms for composition and its step procedure, Figure 3, behave similarly to decomposition. The complexity of the $Composition$ procedure is $\mathrm{O}(n\log(n)+n)$ if the normalized formulation is considered, otherwise it is $\mathrm{O}(n\log(n))$. The denormalization step, also shown in Figure 17, is almost the same $Normalization$ algorithm, whose complexity is $\mathrm{O}(n)$. The difference is at line 6, where it is dividing each coefficient by the normalization factor, preparing the data to be transformed. Therefore the optimal composition has the complexity from the original version, $\mathrm{O}(n\log(n)+n)$, but its results are more exact due to the normalization step, avoiding the calculation of $\sqrt{2}$ on every iteration.

2.4.2 2D Decimated Int-HWT Simplification

The simplifications implemented for the 2D int-HWT consider the same principle presented in the 1D case, i.e., the same strategy of multiplying the transformed values

by the corresponding $2^{-j/2}$ normalization factor, where j is the corresponding level. To compose or decompose a matrix the 1D transform is applied in all rows and in all columns of the input matrix.

However, according to what was discussed in Section 2.2.2, there are two algorithms for composition and decomposition of matrices, what suggests distinct normalization procedures for both approaches. During the study of the original algorithms (STOLLNITZ; DEROSE; SALESIN, 1995), a pattern of normalization factors was recognized. These patterns were analyzed and employed in the development of the normalization procedures for both standard and non-standard algorithms, assuming the decimated version of the HWT.

These patterns are illustrated in Figure 18, where 3 decomposition levels are presented for 8x8 matrices. The parameters j' and j'' indicate the normalization levels. The rule to calculate these normalization factors is described in the equation:

$$2^{-(j'+j'')/2},$$

where $0 \leq j', j'' \leq (\log_2 n) - 1$ and n indicates the matrix order.

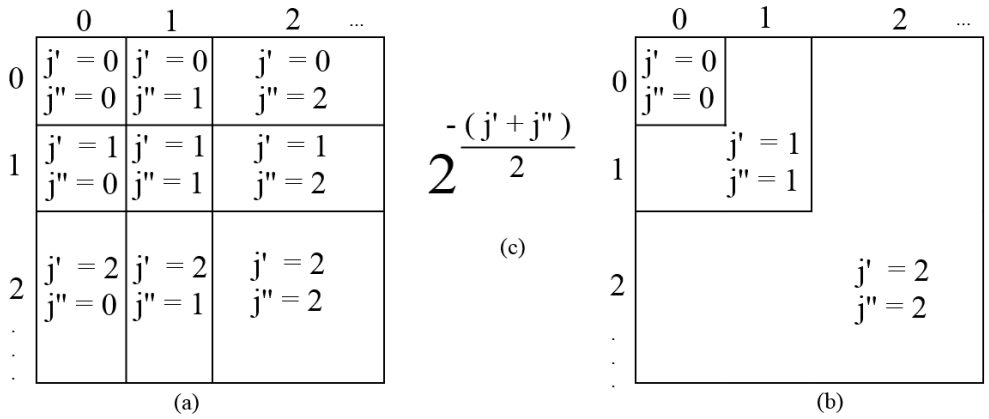

Figura 18: Int-HWT, decimated version: (a) standard, (b) non-standard normalization patterns and (c) rule for normalization factors.

By analyzing the original standard algorithm, Figure 4, it depends on the original normalized decomposition procedure, whose complexity is $\mathrm{O}(n\log(n)+n)$, executing it for every row and column of the input matrix. Considering a nxn matrix the complexity of the original standard algorithm is $\mathrm{O}(n(n\log(n)+n)+n(n\log(n)+n))$, therefore $\mathrm{O}(2n^2\log(n)+2n^2)$. Performing the non-normalized decomposition algorithm, discussed in Section 2.4.1, it reduces the calculations by avoiding intermediate normalization steps.

The non-normalized standard decomposition is performed with $\mathrm{O}(2n^2\log(n))$, and the last step is to normalize the results using the $StandardNormalization$ algorithm, presented in Figure 19, and since it is operating one time on each coefficient of the input matrix its complexity can be expressed by $\mathrm{O}(n^2)$. The complexity of performing the

```
   StandardNormalization (C[1..h, 1..w])
1  Levels ← log2(h) ;
2  for LevelR ← 1 to Levels do
3  |  StartR ← 2 ∧ LevelR ;
4  |  EndR ← 2 ∧ (LevelR + 1) ;
5  |  for Row ← StartR to EndR do
6  |  |  for LevelC ← 1 to Levels do
7  |  |  |  StartC ← 2 ∧ LevelC ;
8  |  |  |  EndC ← 2 ∧ (LevelC + 1) ;
9  |  |  |  for Col ← StartC to EndC do
10 |  |  |  |  Factor ← 2 ∧ ((LevelR + LevelC) / 2) ;
11 |  |  |  |  C[Row][Col] ← C[Row][Col] ∗ Factor ;
12 |  |  |  end
13 |  |  end
14 |  end
15 end
```

Figura 19: 2D decimated Int-HWT: Normalization step for the standard decomposition process.

non-normalized standard decomposition and the standard normalization step can be expressed by $\mathrm{O}(2n^2\log(n)+n^2)$, which is faster than the original normalized procedure by avoiding a second iteration of $\mathrm{O}(n^2)$ on the input matrix.

The original normalized standard composition, presented in Figure 6, can be analyzed in the same way as its decomposition procedure since it performs the same number of operations but produces an approximation of the input matrix used for decomposition. The original version is executed in $\mathrm{O}(2n^2\log(n)+2n^2)$ time, while its optimized version operates in $\mathrm{O}(2n^2\log(n)+n^2)$.

Looking at the original non-standard algorithm, Figure 4, it is executing a intermediate normalization at line 1, dividing each coefficient by the order of the input matrix, which configures a $\mathrm{O}(n^2)$ operation. Considering a quadratic matrix of order n, the rest of the algorithm operates in a $while$ loop which is executed $\log_2(n)$ times. For each $while$ loop there are two for loops also executing $\log_2(n)$ times, each one performing the $DecompositionStep$ algorithm ($\mathrm{O}(n)$) on portions of the input matrix, depending on the level of transformation. The original non-standard decomposition is executed in $\mathrm{O}(n^2+\log(\log(n)+\log(n)))$ time.

The non-normalized non-standard decomposition ($\mathrm{O}(\log(\log(n)+\log(n))))$ does not need the normalization step, and its $Non-StandardNormalization$ algorithm, shown in Figure 20, is performed with complexity $\mathrm{O}(n^2)$. The overall complexity of these procedures is $\mathrm{O}(n^2+\log(\log(n)+\log(n)))$, which is the same from the original normalized non-standard algorithm. The gain of time presented in Figure 27 was obtained performing simple divisions by 2 which is faster than using $\sqrt{2}$ as divisor.

```
NonStandardNormalization (C[1..h, 1..w])
1  Levels ← log2(h) ;
2  for i ← Levels to 1 do
3  |  Factor ← 2 ∧ i ;
4  |  Start ← Factor ;
5  |  End ← 2 ∧ i + 1 ;
6  |  for Row ← 1 to End / 2 do
7  |  |  for Col ← Start to End do
8  |  |  |  C[Row][Col] ← C[Row][Col] / Factor ;
9  |  |  end
10 |  end
11 |  for Row ← Start to End do
12 |  |  for Col ← 1 to End do
13 |  |  |  C[Row][Col] ← C[Row][Col] / Factor ;
14 |  |  end
15 |  end
16 end
```

Figura 20: Int-HWT, decimated version: Normalization step for the non-standard decomposition process.

2.4.3 1D Undecimated Int-HWT Simplification

The normalized decomposition procedure for the undecimated version of the Int-HWT, either one-dimensional or two-dimensional, is executed in the same order from the optimization developed for the decimated version: first a non-normalized decomposition is made, as indicated in Figure 7; after this stage the normalization of all coefficients is performed, multiplying all of them by the normalization factor $2^{j/2}$, where j is the resolution level of the coefficients through out the decomposition.

The complexities of both $DecompositionStep$ and $Decomposition$ procedures are $\mathrm{O}(2n+2)$ and $\mathrm{O}(Levels.(2n+3))$, respectively. Both are part of the original decomposition procedure, shown in Figure 8. As in the previous cases, by performing a non-normalized decomposition there is no need to execute divisions by $\sqrt{2}$ in the step procedure, pseudo-code shown in Figure 21, reducing the corresponding complexities to $\mathrm{O}(2n)$ and $\mathrm{O}(2n.Levels)$ (the same complexity as the original normalized transform).

```
DecompositionStep (C[1..n], D[1..n], LastC[1..n])
1  for i ← 1 to n − 1 do
2  |  C[i] ← (LastC[i] + LastC[i + 1]) / [2; 2] ;
3  |  D[i] ← LastC[i] − C[i] ;
4  end
5  C[i] ← (LastC[n] + LastC[0]) / [2; 2] ;
6  D[i] ← LastC[n] − C[n] ;
```

Figura 21: 2D undecimated Int-HWT: non-normalized decomposition step.

The normalization procedure, presented in Figure 22, performs the normalization step of all coefficients after the transform. In contrast with the decimated version, the

undecimated approach deals with a set of vectors as result of the transform, and each resultant vector need to be normalized individually. The normalization procedure multiplies each scalar coefficient by its normalization factor $2^{i/2}$ for every resultant vector from the transform, configuring $\mathrm{O}(n.Levels)$ from line 1 to 5 of the pseudo-code. With the new set of scalar values all wavelet coefficients are then recalculated, configuring $\mathrm{O}(Levels.(n+3)+2)$ from line 6 to 16, which can be simplified to $\mathrm{O}(n.Levels)$. The overall complexity of the procedure is $\mathrm{O}(2n.Levels)$.

```
   À-TrousNormalization
   (C[1..n], C'[1..Levels][1..n], D'[1..Levels][1..n])
1  for i ← 1 to Levels do
2  |  for j ← 1 to n do
3  |  |  C'[i][j] ← C'[i][j] * 2 ∧ ((i/2)) ;
4  |  end
5  end
6  LastC ← C ;
7  NextC ← C'[1] ;
8  for i ← 1 to Levels do
9  |  D ← LastC − NextC ;
10 |  if i + 1 < Levels then
11 |  |  LastC ← C'[i] ;
12 |  |  NextC ← C'[i + 1] ;
13 |  end
14 end
```

Figura 22: 1D undecimated Int-HWT: normalization procedure.

By performing a non-normalized vector transform and the normalization procedure, the complexity is $\mathrm{O}(4n.Levels)$, which is more expensive than the original normalized formulation. The gain obtained with the new implementation is on the exactitude of results by avoiding high number of divisions using $\sqrt{2}$ from the original $DecompositionStep$ algorithm, as shown in Figure 26.

The composition process, shown in Figure 9, starts with a copy of all scalar coefficients from the lowest level summing it with all wavelet vectors produced during decomposition. The complexity of this procedure is $\mathrm{O}(n.Levels)$ and can be used for both normalized and non-normalized transformations, there is no need for optimization.

2.4.4 2D undecimated Int-HWT Simplification

As done for the decimated case, the undecimated transform, preformed by the à-trous algorithm, considers the same principle presented in the one-dimensional optimization, i.e., the same strategy of multiplying the transformed values by the corresponding $2^{j/2}$ normalization factor, where j is the corresponding level. To decompose a data matrix the two-dimensional implementation creates a set of matrices carrying scalar coefficients and another set of matrices to store wavelet coefficients. The procedure is expensive on both performing and storing the results. Applying the same

idea used in previous implementations, it is possible to reduce the error involved in the process.

In Figure 13 the original standard procedure is presented. It performs the decomposition of every line and level of a nxm matrix, configuring a complexity of $\mathrm{O}(n.2m.Levels)$. The analog procedure for all columns of the same matrix has $\mathrm{O}(m.2n.Levels)$. The entire procedure is expressed by $\mathrm{O}(n.2m.Levels + m.2n.Levels)$, which can be simplified to $\mathrm{O}(4n.m.Levels)$. The non-standard procedure executes the same amount of calculations, but intercalating the decomposition of rows and columns. Both standard and non-standard algorithms can be used in conjunction with $DecompositionStep$ shown in Figure 16, so the complexity of executing the normalized and non-normalized are the same.

The next step in order to normalize the transformed values is to perform the $A\text{-}TrousMatrixNormalization$ procedure, shown in Figure 23, which is executed in $\mathrm{O}(4n.m.Levels)$. Therefore, the optimized decomposition procedure developed costs $\mathrm{O}(8n.m.Levels)$, twice the cost of the original normalized formulation, but tests indicate that our approach results in more exact values as shown in Figure 28.

```
À-TrousMatrixNormalization
(C[1..n][1..m], C'[1..Levels][1..n][1..m], D'[1..Levels][1..n][1..m])
1  for i ← 1 to Levels * 2 do
2  |  for j ← 1 to n do
3  |  |  for w ← 1 to m do
4  |  |  |  C'[i][j][w] ← C'[i][j][w] * 2 ∧ ((i/2)) ;
5  |  |  end
6  |  end
7  end
8  LastC ← C ;
9  NextC ← C'[1] ;
10 for i ← 1 to Levels * 2 do
11 |  D ← LastC − NextC ;
12 |  if i + 1 < Levels then
13 |  |  LastC ← C'[i] ;
14 |  |  NextC ← C'[i + 1] ;
15 |  end
16 end
```

Figura 23: 2D undecimated Int-HWT: normalization procedure.

2.5 Application - Data Compression

The main goal of compression procedures is to express an initial data set with the smaller amount of points as possible, this task can be done either with or without loss of information (STOLLNITZ; DEROSE; SALESIN, 1995), (KOZAKEVICIUS; BAYER, 2014). In the wavelet context, the wavelet expansion of the initial data is analysed in

order to decide which wavelet coefficients are more significant then others. Therefore, keeping only the most significant ones, the truncated series represents the compressed(or filtered) data. In (PERIN; KOZAKEVICIUS, 2013) ECG signals were analysed assuming adaptive compression techniques.

In many signal analysis applications, the significant wavelet coefficients are then used to draw further conclusions about the original data, as for example in (ASADZADEH; HASHEMI; KOZAKEVICIUS., 2013), where micro-calcifications in mammograms were detected based on the truncated wavelet representation of the images.

The significance of the coefficients is judged by a threshold value, which can be obtained based on many types of heuristics, linear as the Universal threshold proposed by (DONOHO, 1995) or adaptive as proposed in (BAYER; KOZAKEVICIUS, 2010). For a review about the many possibilities for choosing threshold values see (KOZAKEVICIUS; BAYER, 2014).

By ignoring (and excluding) non significant wavelet coefficients from the wavelet expansion, this strategy turns the method into a lossy compression procedure. This procedure is called hard thresholding, according to (DONOHO, 1995). The result is an approximation of the original function. Figure 24, wich can be found in (STOLLNITZ; DEROSE; SALESIN, 1995), shows a sequence of approximations, generated by varying the compression rate s/N, where s is the number of significant coefficients, and N the total amount of points in the initial data representation.

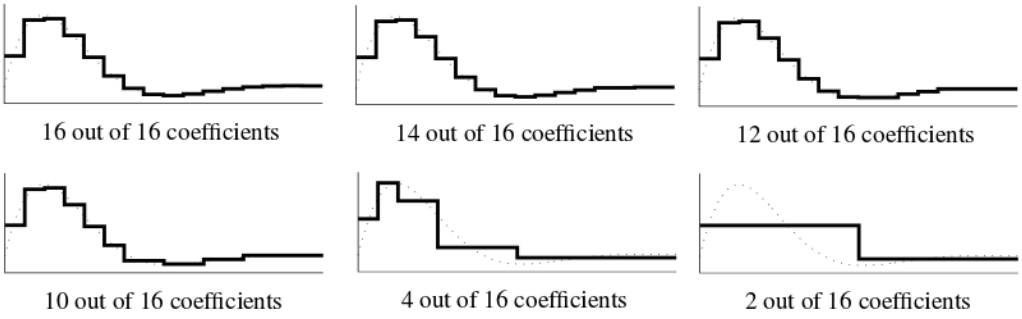

Figura 24: Approximation of functions after compression.

The hard thresholding is trivial for punctual data, but it can not be applied in the same way to interval information. The punctual algorithm for compression uses a real value τ as threshold VALUE, and the decision of which details must be ignored is a set of simple punctual comparisons.

When treating with interval data, τ turn to be also an interval, carrying the error in the threshold calculus. This way the compress decision can not be evaluated with the same punctual comparison as before (MOORE, 1979). Two solutions were developed to manage the decision in this interval extension, the first was named as Hard Decision and the second as Soft Decision, inspired by the nomenclature considerd by Donoho for its threshold operators.

```
Data: C[0...n] : real
i ← 0 ;
for i < n do
    if |C[i]| < τ then
        C[i] = 0 ;
    end
    i = i + 1 ;
end
```

```
Data: C[0...n] : interval
i ← 0 ;
for i < n do
    if C̄[i] < τ̲ then
        C[i] = [0 : 0] ;
    end
    i = i + 1 ;
end
```

```
Data: C[0...n] : interval
midτ = (τ̄ + τ̲)/2 ;
i ← 0 ;
for i < n do
    midC = (C̄[i] + C̲[i])/2 ;
    if midC < midτ then
        C[i] = [0 : 0] ;
    end
    i = i + 1 ;
end
```

Figura 25: Hard Thresholding, Hard Decision and Soft Decision compress procedures.

The Hard Decision is made by verifying if the interval data is entirely less than τ, comparing the left bound of the information and the right bound of the threshold. This procedure grants that every possible punctual coefficient is less than all possible punctual values belonging to τ.

The Soft Decision is made by verifying if most of the interval data is less than the midpoint of τ, comparing the left bound of the information and the center of the threshold. This procedure grants that most possible punctual coefficients are less than the midpoint of $\overline{\tau}$ and $\underline{\tau}$.

2.6 Tests and Results

The numerical validation of algorithms described in Sections 2.2 and 2.3 are based on the application of the HWT for image processing. In this manner, the interval parameters for test executions were obtained from punctual values setting degenerated intervals and using them as input to the interval extensions of both decimated and undecimated implementations of the HWT.

The implementation of interval procedures in the Int-Haar library performs the computation of interval error in the process, presenting the widest interval diameter contained in the transformation results. For all tests the time measurement was done by executing each function 30 times, mean and standard deviation values were obtained from those times and used to generate the figures presented in this section.

In order to compute tests using the 1D HWT and its interval extension a vector filled with 1048576 random values is being used as input. For tests that use the 2D HWT algorithms the input is generate from a 1024x1024 image, using 8 bits gray-scale codification, configuring 1048576 values.

The tests were executed on a Intel ® Core ™ i7 950 Processor @ 3.07GHz, 6GB RAM DDR3 @ 1066MHz, Windows 8.1 OS, compiled using Microsoft Visual C++ Compiler for Visual Studio 2015 on Win64 Release compilation target.

The results shown in Figure 26 describe the execution time of both decomposition

and composition procedures from Section 2.2.1, targeting the decimated and undecimated 1D HWT. The data indicates that the decimated algorithms developed in this work are faster then the originals from the literature (STOLLNITZ; DEROSE; SALESIN, 1995), and the error of calculation is significantly smaller than the original formulation. The data also indicates that the developed undecimated algorithms are slower then the originals from the literature (STARCK; FADILI; MURTAGH, 2007).

The undecimated formulation do not add error to calculations on composition step if there is no error from the input. The increase of error on composition in conjunction with decomposition is the case where there are errors at input data for the composition step. The result of the Euclidean Distance (EUC) and Mean Square Error (MSE) for both formulations indicate that the output from developed decomposition and composition procedures in sequence result in an better image quality, closer to the input used. Peak Signal to Noise Ratio (PSNR) is largest used in compression evaluation, measuring the quality of the resultant image comparing with the original input, where the highest value means better results.

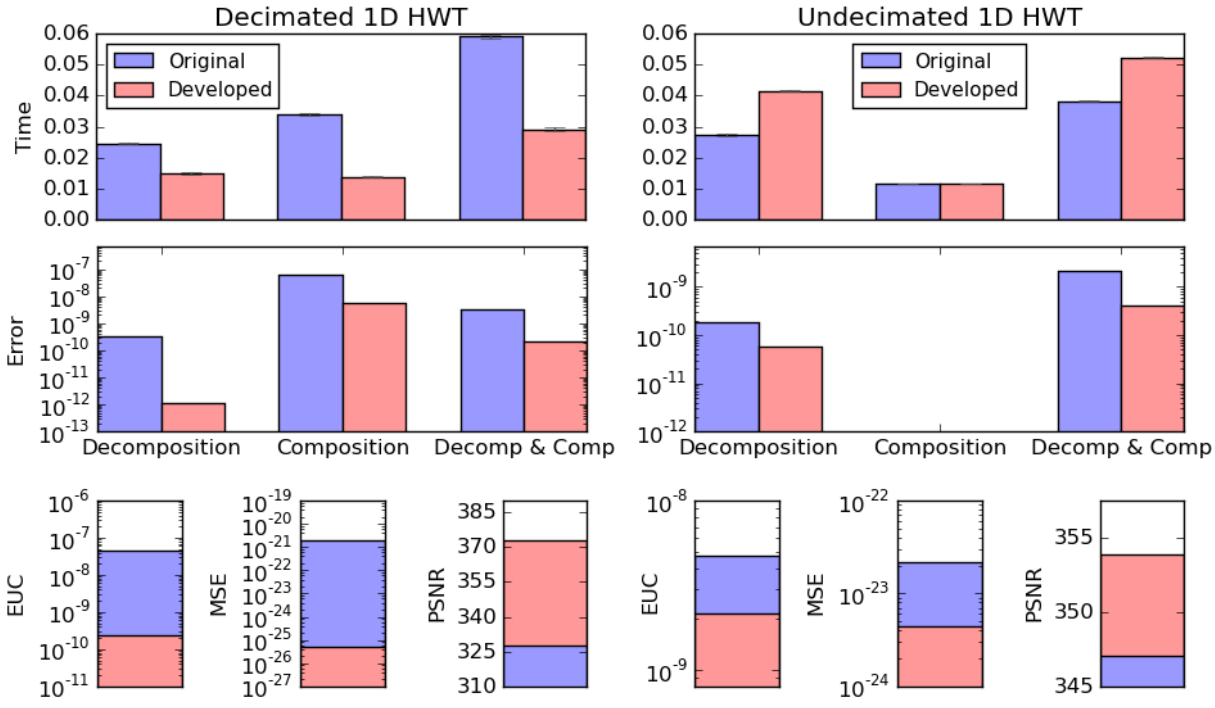

Figura 26: Times of execution and error measurement for the 1D HWT using 1048576 random values.

The data from Figure 27 shows the execution time of both decomposition and composition procedures, using the standard and non standard algorithms for matrices from Section 2.2.2, targeting the decimated 2D HWT. The data indicates that the algorithms developed in this work are faster then the originals, and the error of its calculation is smaller compared with original algorithms. It is important to verify that, as shown in Figure 18, the calculation of $\sqrt{2}$ for the developed non standard method do not need to

be performed due to the implemented algebraic simplification. Therefore, the calculation error during this process is null, indicated by the lack of bars in the same chart. The EUC and MSE indicate that composition in conjunction with decomposition result in an image closer to the original input, meaning less calculus deviation during the process, and PSNR confirms the quality of results.

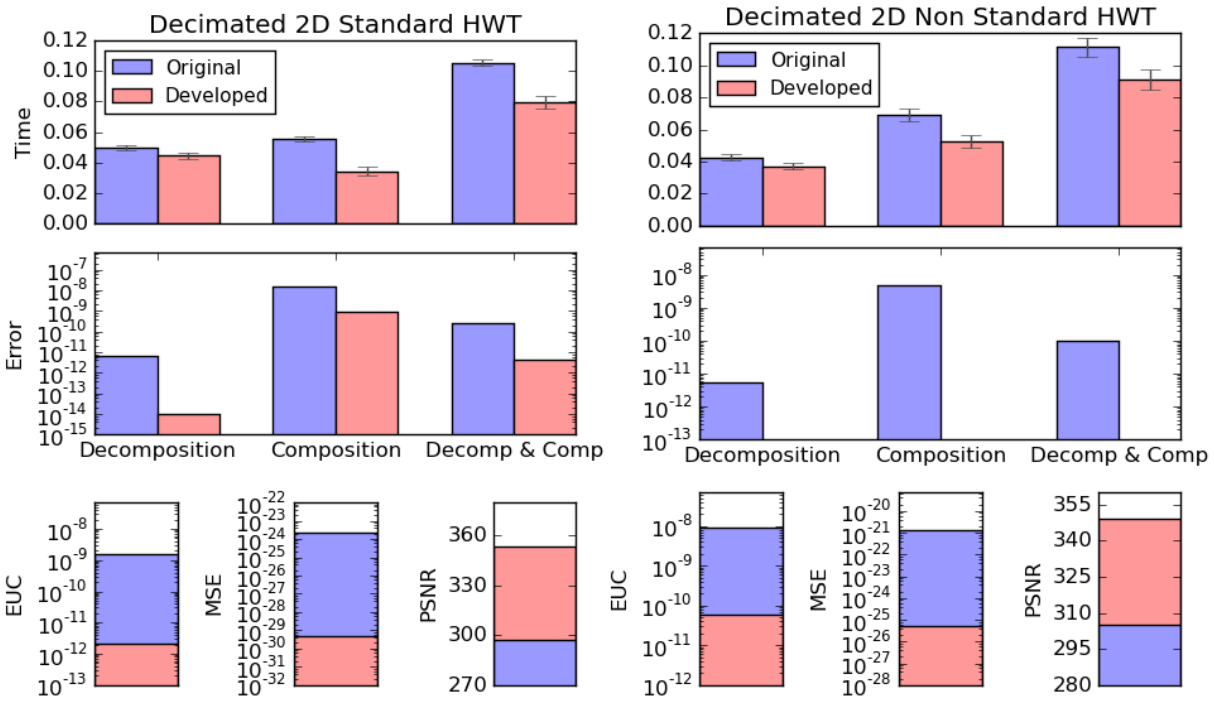

Figura 27: Times of execution and error measurement for the 2D Standard and Non Standard HWT using a 1024x1024 image.

The data from Figure 28 shows the execution time of both decomposition and composition procedures, using the standard and non standard algorithms for matrices from Section 2.3.2, used on the undecimated 2D HWT. The data indicates that the algorithms developed in this work are slower then the literature, both formulations do not add error to calculations on composition step if there is no error on the input. The error of calculations are smaller compared with original algorithms. The EUC and MSE indicate that composition in conjunction with decomposition result in an image closer to the original input, meaning less calculus deviation during the process, and PSNR confirms the quality of results.

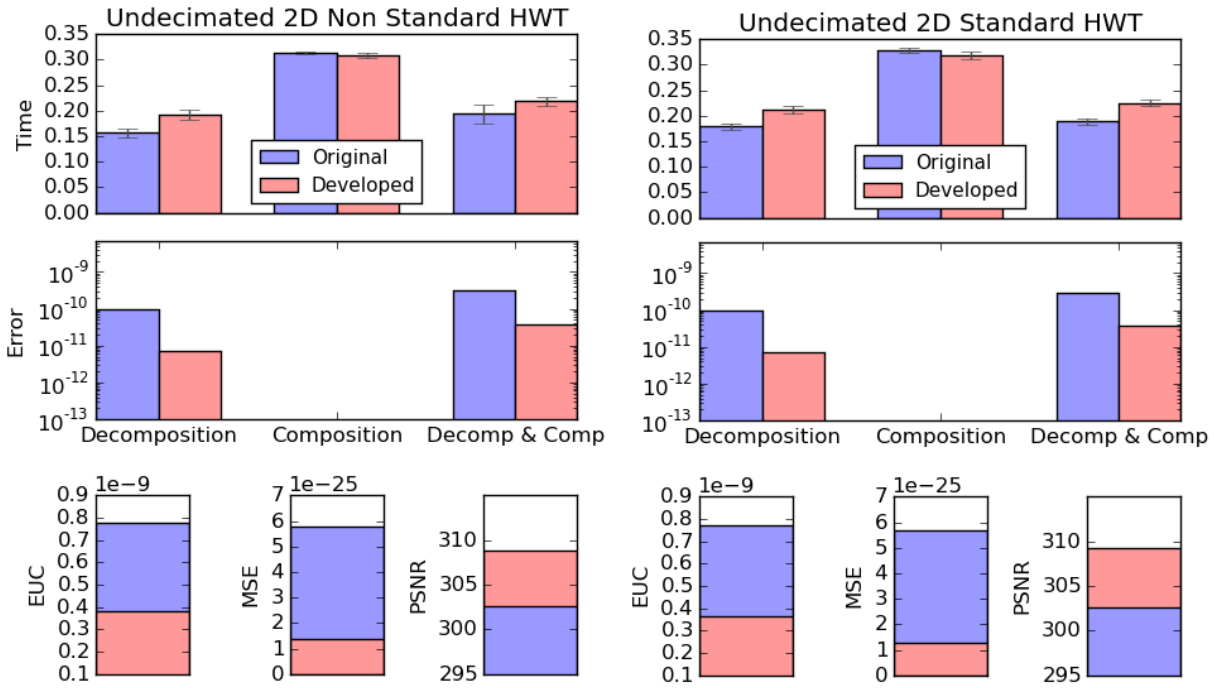

Figura 28: Times of execution and error measurement for the 2D Non Standard HWT À-Trous using a 1024x1024 image.

3 CONCLUSION

This work presented an interval implementation of the HWT, for both decimated and undecimated approaches and covering all cases of study, obtaining good results related to error analysis, that is, the optimizations improve accuracy. As consequence of the implemented optimizations our algorithms are faster then the original decimated formulation, but slower than the original undecimated algorithms.

As the wavelet transforms are appropriated to data analysis in contexts that the scales of representation are relevant to the problem, the precision gain obtained with the proposed simplifications in this work represent a significant contribution to this research area.

Further research considers the study of the Daubechies Wavelet Transform together with corresponding parallel and/or distribution Int-DWT library extension, by making use of massive parallel architectural of GPUs and considering CUDA programming language.

REFERÊNCIAS

ASADZADEH, M.; HASHEMI, E.; KOZAKEVICIUS., A. On efficiency of com- bined Daubechies wavelets and statistical parameters applied in mammography. **Applied and Computational Mathematics: an international journal**, [S.l.], v.12, n.3, p.289–306, 2013.

BAYER, F. M.; KOZAKEVICIUS, A. J. SPC-threshold: uma proposta de limiarização para filtragem adaptativa de sinais. **Tendências em Matemática Aplicada e Computacional**, [S.l.], v.11, n.2, p.121–132, 2010.

BRITO, A.; KOSHELEVA, O. Interval + Image = Wavelet: for image processong under interval uncertainty, wavelets are optimal. **Reliable Computing**, [S.l.], v.4, p.291–301, 1998.

CHRISTOPOULOS, C.; SKODRAS, A.; EBRAHIMI, T. The JPEG2000 still image coding system: an overview. **Consumer Electronics, IEEE Transactions on**, [S.l.], v.46, n.4, p.1103–1127, Nov 2000.

DAUBECHIES, I. **Ten Lectures on Wavelets**. Philadelphia: SIAM, 1992. (CBMS-NSF Regional Conference Series in Applied Mathematics).

DONOHO, D. De-noising by soft-thresholding. **IEEE Transactions on Information Theory**, [S.l.], v.41, p.613627, 1995.

FIGUEIREDO, L. H.; STOLFI, J. Affine Arithmetic: Concepts and Applications. **Numerical Algorithms**, [S.l.], v.37, p.147–158, 2004.

H. OM, M. B. An Improved Image Denoising Method Based on Wavelet Thresholding. **Journal of Signal and Information Processing**, [S.l.], v.3, p.109–116, 2012.

HAAR, A. Zur Theorie der orthogonalen Funktionensysteme. **Mathematische Annalen**, [S.l.], v.69, n.3, p.331–371, 1910.

HICKEY, T.; JU, Q.; EMDEM., M. V. Interval arithmetic: From principles to implementation. **Journal of the ACM**, [S.l.], v.48, n.5, p.1038–1068, 2001.

HOFSCHUSTER, W.; KRAMER, W.; NEHER, M. **C-XSC and Closely Related Software Packages**. Universitat Wuppertal: Springer-Verlag, 2008. 68–102p. (Proceedings 08021 - Numerical Validation in Current Hardware Architectures, LNCS 5492).

HOLBIG, C.; DIVERIO, T.; CLAUDIO, D. Parallel Environment with High Accuracy for Resolution of Numerical Problems. **Revista IEEE América Latina**, [S.l.], v.7, p.114–121, 2009.

HOLBIG, C.; JúNIOR, P. M.; CLAUDIO, D. Solving Real Life Applications With High Accuracy. In: INTERNATIONAL CONFERENCE ON PARALLEL COMPUTING, PARCO 2005, 2005, Málaga. **Anais...** [S.l.: s.n.], 2005. p.98.

KOLBERG, M.; FERNANDES, L. G.; CLAUDIO, D. Dense Linear System: A Parallel Self-verified Solver. **International Journal of Parallel Programming**, [S.l.], v.36, p.412–425, 2008.

KOZAKEVICIUS, A. J.; BAYER, F. M. Signal denoising via wavelet coefficients thresholding. **Ciência & Natura, Special Issue**, [S.l.], v.1, p.37–51, 2014.

KOZAKEVICIUS, A.; SCHMIDT, A. A. Wavelet transform with special boundary treatmentfor 1D data. **Comp. Appl. Math**, [S.l.], v.1, p.1–15, 2013.

KUMAR, M.; SUDHANSU, S. K.; KASABEGOUDAR, V. Wavelet Based Texture Analysis And Classification With Linear Regration Model. **International Journal of Engineering Research and Applications (IJERA)**, [S.l.], v.2, p.1963–1970, 2012.

LIU, L. Interval wavelet numerical method on Fokker-Planck equations for nonlinear random systems. **Advances in Mathematical Physics**, [S.l.], v.2013, n.ID 651357, p.1–7, 2013.

MILANI, C. R.; KOLBERG, M. L.; FERNANDES, L. G. Solving Dense Interval Linear Systems with Verified Computing on Multicore Architectures. In: HIGH PERFORMANCE COMPUTING FOR COMPUTATIONAL SCIENCE - VECPAR 2010 - 9TH INTERNATIONAL CONFERENCE, BERKELEY, CA, USA, JUNE 22-25, 2010, REVISED SELECTED PAPERS, 2010. **Anais...** [S.l.: s.n.], 2010. p.435–448.

MINAMOTO, T.; AOKI, K. A blind digital image watermarking using interval wavelet decomposition. **Int. Journal of Signal proc., Image proc. and Pattern recognition**, [S.l.], v.3, n.2, p.59–72, 2010.

MOORE, R. **Methods and Applications of Interval Analysis**. Philadelphia: SIAM, 1979.

MOORE, R. E. Automatic error analysis in digital computation. , [S.l.], 1959. Technical Report LMSD84821, Lockheed Missiles and Space Co.

MOORE, R. E. **Interval Arithmetic and Automatic Error Analysis in Digital Computing**. 1962. Tese (Doutorado em Ciência da Computação) — PhD thesis, Department of Mathematics, Stanford University, Stanford, POA.

NOVIKOV, I. Y.; SKOPINA, M. A. Why are Haar bases in various structures the same? **Mathematical Notes-Short Communications**, [S.l.], v.91, n.6, p.895–898, 2012.

PBLAS (Parallel Basic Linear Algebra Subprograms. , [S.l.].

PERIN, G.; KOZAKEVICIUS, A. J. Filtragem Wavelet de Sinais Cardíacos através de Algoritmos Adaptativos. **RITA (***in portuguese***)**, [S.l.], v.20, n.3, p.95–111, 2013.

SANTOS, V. dos; PILLA, M.; REISER, R.; KOZAKEVICIUS, A. Int-Haar: Extensão Intervalar da Transformada de Haar. In: ESCOLA REGIONAL DE ALTO DESEMPENHO, ERAD 2013, 2013, POA. **Anais...** SBC, 2013. p.167–170.

SHU-LI, M.; L.QI-SHAO; SEN-WEN, Z.; LI, J. Adaptive interval wavelets precise integration method for partial differential equations. **Applied Mathematics and Mechanics**, [S.l.], v.26, n.3, p.364–371, 2005.

STARCK, J.-L.; FADILI, J.; MURTAGH, F. The Undecimated Wavelet Decomposition and its Reconstruction. In: IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 16, NO. 2, 2007. **Anais...** [S.l.: s.n.], 2007.

STOLLNITZ, E. J.; DEROSE, T. D.; SALESIN, D. H. Wavelets for Computer Graphics: A Primer, Part 1. **IEEE Computer Graphics and Applications**, [S.l.], v.15, n.3, p.76–84, 1995.

TUCKER, W. **Validated Numerics**: A Short Introduction to Rigorous Computations. New Jersey: Princeton Unversity Press, 2011. 137p.

WALSTER, G. Interval arithmetic: The new floating-point arithmetic paradigm. , [S.l.].

Weblink to C-XSC. , [S.l.].