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

Trabalho de Conclusão de Curso

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

Vinícius Rodrigues dos Santos

Pelotas, 2016

Vinícius Rodrigues dos Santos

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

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

Orientadora: Profa. Dra. Renata Hax Sander Reiser
Coorientador: Prof. Dr. Maurício Pilla
Colaboradora: Profa. Dra. Alice Kozakevicius

Pelotas, 2016

**Insira AQUI a ficha catalográfica
(solicitada na página da biblioteca)**

Insira AQUI a folha de aprovação
(será entregue após a banca)

Dedico este trabalho a meu avô, Orlando Ferreira dos Santos, conhecido "Camarão", por ter apoiado meus estudos até o início de minha formação acadêmica. Por conta de eventos inevitáveis da vida, infelizmente não poderei lhe mostrar o diploma, porém registro aqui minha dedicatória ao velho camarada.

"Leave this world a little better than you found it."
— BADEN-POWELL'S LAST MESSAGE (1945)

RESUMO

DOS SANTOS, Vinícius Rodrigues. **Biblioteca Int-DWTs - Simplificações Algébricas Aumentando Desempenho e Exatidão de Transformadas Discretas Wavelet**. 2016. 64 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2016.

Este trabalho descreve a modelagem e o desenvolvimento de metodologia bem como a análise de resultados para consolidação da biblioteca Int-DWTs, contendo Métodos Intervalares para Transformadas Discretas Wavelet. Os métodos presentes em Int-DWTs incluem estensões intervalares e suas correspondentes otimizações para Transformadas Discretas Wavelet (TDWs), mais especificamente para as Transformadas Wavelet de Haar e Daubechies. As otimizações implementadas promoveram simplificações algébricas visando o incremento de desempenho e a garantia de exatidão das TDWs estudadas. São apresentadas e comparadas as análises de complexidade dos algoritmos desenvolvidos em relação aos presentes na literatura. A qualidade de resultados é verificada ao comparar três métricas: EUC (*Euclidean Distance*), MSE (*Mean Squared Error*) e $PSNR$ (*Peak Signal-to-Noise Ratio*). O nível de erro produzido pelas formulações intervalares, como também os desempenhos obtidos através das simplificações propostas, são analisadas e comparadas com resultados baseados na literatura. Em contextos onde as escalas de representação são relevantes para o problema, os ganhos de exatidão obtidos com as otimizações propostas representam uma contribuição significativa para a área de Computação Científica.

Palavras-chave: wavelet, haar, daubechies, otimização, desempenho.

ABSTRACT

This project describes the main results consolidating the Interval Methods of Discrete Wavelet Transforms Library (Int-DWTs), providing algebraic simplifications in order to increase performance and guarantee accuracy of Discrete Wavelet Transforms (DWTs). The methods in the Int-DWTs include interval extensions and corresponding optimizations to the Haar Wavelet Transform (HWT) and Daubechies Wavelet Transform (DWT), which are implemented by using C-XSC. Optimizations for the normalized formulations of the one- and two-dimensional transforms to increase speed and guarantee accuracy of 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 analyzed. The results consider a study over the interval extension of DWT, deducting simplifications and implementing optimizations based on the analogous methodology performed over HWT approaches. In addition, not only interval algorithms are developed but also metrics for evaluating procedure's accuracy are considered. By assuming concepts from Interval Mathematics, this work enables us to perform interval analysis and efficiently manage computing errors of DWTs. The metrics being used to measure result quality in the Int-DWTs are the following: Euclidean Distance, Mean Squared Error and Peak signal-to-noise ratio. In contexts where the scales of representation are relevant to the problem, the accuracy gain obtained with the proposed optimizations in the Int-DWTs represent a significant contribution to the scientific computing research area.

Keywords: Wavelet, haar, daubechies, optimization, performance.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

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

PDE   Partial Differential Equations

CONTENTS

1 INTRODUCTION

The quality of numerical results in Scientific Computing (SC) depends on understanding the different error causes and on controlling their propagation, as well as improving computations upon the involved procedures.

This study considers the Interval Mathematics (IM) approach and proposes a solution based on Moore's arithmetic (MOORE, 1979) for the interval implementation of the Daubechies Wavelet Transform (DaWT) and Haar Wavelet Transform (HWT), considering its two approaches, Cascade and À-Trous, and their standard formulations. The implementation of interval extensions of both HWT and DaWT is the first step on the development of the Int-DWT library that will provide interval results for several Discrete Wavelet Transforms (DWTs).

Interval results carry over the safety of their quality together with the degree of their uncertainty (MOORE, 1979). The diameter of the interval solution represents with fidelity the 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.

The C-XSC library (HOLBIG; DIVERIO; CLAUDIO, 2009) is employed to support the interval implementation of all formulations addressed by the Int-DWTs library. Such open source 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.

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. As proposed by (SHU-LI et al., 2005), 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, 2013), based on the interval extension for the Shannon-Garbor wavelet family. With respect to image processing applications, (MINAMOTO; AOKI, 2010) proposes a blind digital image watermarking method using interval Daubechies wavelets.

The main motivation for interval techniques integrated with DWTs is to provide trustworthy and validated results to technological and SC applications assuming such transformations. The techniques involving wavelets is ample, specially in signal and image processing (H. OM, 2012; KUMAR; SUDHANSU; KASABEGOUDAR, 2012).

The HWT, the simplest Discrete Wavelet Transform (DWT) due to the complexity of its filters, is still nowadays largely explored, being its robustness with respect to many different mathematical structures and space formulations a relevant aspect (NOVIKOV; SKOPINA, 2012) addressed 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 as well as the current formulation in the interval context. Preliminary results of the new interval extension of the HWT presented in (SANTOS et al., 2013a) are revisited here

The optimal and interval extension of the DaWT is also addressed as one of the goals of the current work. In contrast to the HWT, the DaWT has more complex filters (NIELSEN, 1998) requiring different algebraic manipulations in order to optimize its calculations. The developed optimizations ensures more exact results due to the reduced amount of irrational values to be computed during the transformation.

The current research addresses as well the *À-Trous* algorithm, an undecimated 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, considered as a significant part of many compression and filtering algorithms, is analyzed and its interval extensions are presented, namely the Hard and Soft Decision procedures.

During the study of the original normalized and non-normalized forms of the HWT (STOLLNITZ; DEROSE; SALESIN, 1995), the possibility to optimize them arises from the observation of patterns of normalization factors in each level of transformation. Therefore, algebraic simplifications are executed to eliminate operations with irrational numbers, originally responsible for the normalization of the transform and considered in each iteration of the original algorithms (SANTOS et al., 2013b).

Through these simplifications, a gain of accuracy is obtained for the normalized approach of the decimated HWT, producing therefore more reliable results. In the current work, a similar heuristic is considered to simplify the standard formulation of the undecimated HWT and DaWT. This work also describes the complexity of the developed algorithms for both HWT and DaWT comparing with the ones from the literature.

This work is organized as follows. Chapter 2 summarizes aspects about interval arithmetics and the metrics used to validate the results. Chapter 3 presents the original formulations of both HWT and DaWT. Chapter 4 details the development of the proposed optimizations. Chapter 5 shows the configuration of tests and the analysis of results. Finally, conclusions are presented in Chapter 6.

2 INTERVAL MATHEMATICS

This Chapter presents the fundamental concepts of Interval Mathematics, by describing its interval aritmethics in Section 2.1, and introduces interval extensions of several metrics related to image quality evaluation in Section 2.2. In addition, the C-XSC library for computation of interval arithmetics is introduced in Section 2.3.

2.1 Concept of Interval Aritmethics

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, 1996; HICKEY; JU; EMDEM., 2001) and (FIGUEIREDO; STOLFI, 2004).

Despite distinct approaches of arithmetic for intervals, see e.g. (WALSTER, 1996; HICKEY; JU; EMDEM., 2001) and (FIGUEIREDO; STOLFI, 2004), this work considers the interval analysis proposed by Moore in the 1960's, providing methods to obtain accuracy in numerical calculations (MOORE, 1979, 1959, 1962).

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

$$X = \{x \in \mathbb{R} : a \le x \le b, a, b \in \mathbb{R}, a \le b\}. \tag{1}$$

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: $X * Y = \{x * y : x \in X \text{ and } y \in Y\}$.

Interval functions can be computed by performing arithmetic operations or by applying rational approximation methods (WALSTER, 1996). 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

$$\mathbb{I}_{[a,b]} = \{[x, y] \subseteq \Re : a \leq x \leq y \leq b\}. \tag{2}$$

In particular, let U be the real unit interval $U = [0, 1] \subseteq \Re$. By Eq. (2), the set of all subintervals of U is indicated as $\mathbb{I}_U = \{[x, y] \subseteq \Re : 0 \leq x \leq y \leq 1\}$. The projections $l, r : \mathbb{I}_{[a,b]} \to [a, b]$ are respectively given by $l([x, y]) = x$ and $r([x, y]) = y$, for all $[x, y] \in \mathbb{I}_{[a,b]}$. 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, for all $X, Y \in \mathbb{I}_{[a,b]}$, interval arithmetic operations can be defined as follows:

$$X \pm Y = [\underline{X} \pm \underline{Y}, \overline{X} \pm \overline{Y}]; \qquad\qquad 1/Y = [1/\overline{Y}, 1/\underline{Y}], \quad \text{if} \quad 0 \notin Y;$$

$$X \cdot Y = [\min\{\underline{X} \cdot \underline{Y}, \underline{X} \cdot \overline{Y}, \overline{X} \cdot \underline{Y}, \overline{X} \cdot \overline{Y}\}, \max\{\underline{X} \cdot \underline{Y}, \underline{X} \cdot \overline{Y}, \overline{X} \cdot \underline{Y}, \overline{X} \cdot \overline{Y}\}].$$

Additionally, the power and root operations are, respectively, defined as follows:

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

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

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:

$$X \leq_{\mathbb{I}_{[a,b]}} Y \text{ iff } \underline{X} \leq \underline{Y} \text{ and } \overline{X} \leq \overline{Y}. \tag{3}$$

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 $(\mathbb{I}_{[a,b]}, \leq_{\mathbb{I}_{[a,b]}})$.

2.2 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 metrics being used to measure result quality are presented below.

2.2.1 Euclidean Distance

Let $\tilde{Y} = (\tilde{y})_{ij} \in \mathbb{R}^{n \times m}$ be an estimator of $Y = (y)_{ij} \in \mathbb{R}^{n \times m}$ whose $n.m$ elements $\tilde{y}_{ij}$ are predictions of the original values $(y)_{ij}$. The Euclidean distance between Y and its estimator $\tilde{Y}$ is defined by the following expression:

$$D(\tilde{Y}, Y) = \sqrt{\sum_{j=0}^{m} \sum_{i=0}^{n} (\tilde{y}_{ij} - y_{ij})^2}. \tag{4}$$

Analogously, $\tilde{\mathbb{Y}} = (\tilde{\mathbf{Y}})_{ij}$ is called an interval estimator of $\mathbb{Y} = (\mathbf{Y})_{ij} \in \mathbb{IR}^{n \times m}$ with an nm-dimensional matrix of interval predictions $\tilde{\mathbf{Y}}_{ij}$ of the original interval quantities $\mathbf{Y}_{ij}$. An interval extensional of Eq.(5) is given in the following:

$$\mathbf{D}(\tilde{\mathbb{Y}}, \mathbb{Y}) = \sqrt{\sum_{j=0}^{m} \sum_{i=0}^{n} \left(\tilde{\mathbf{Y}}_{ij} - \mathbf{Y}_{ij} \right)^2}. \tag{5}$$

2.2.2 Mean Squared Error

The Mean Squared Error (MSE) is considered as a risk function, corresponding to the expected value of the squared error loss. It is an estimator measuring the average of the squares of the errors in SC, providing the difference between the estimator and what is estimated.

The MSE allows us to compare the pixel values of our original image to our degraded (noise) image based on the amount by which the values of the original image differ from the degraded image.

Let $Y = (y)_{ij} \in \mathbb{R}^{n \times m}$ be the related matrix of true values y_{ij}. The accuracy of

$\tilde{Y} = (\tilde{y})_{ij}$ can be obtained by the application of an MSE operator as the following

$$MSE(\tilde{Y}, Y) = \frac{1}{mn}\sqrt{\sum_{j=0}^{m}\sum_{i=0}^{n}(\tilde{y}_{ij} - y_{ij})^2} = \frac{1}{mn}D(\tilde{Y}, Y). \tag{6}$$

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 applying an interval extension of the MSE operator in Eq. (4), which is given by the following expression:

$$\mathbf{MSE}(\tilde{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{mn}\sqrt{\sum_{j=0}^{m}\sum_{i=0}^{n}(\tilde{\mathbf{Y}}_{ij} - \mathbf{Y}_{ij})^2} = \frac{1}{mn}\mathbf{D}(\tilde{\mathbb{Y}}, \mathbb{Y}). \tag{7}$$

2.2.3 Peak signal-to-noise ratio

The Rate/Distortion (R/D) measures the image quality in terms of Peak Signal-to-Noise Ratio ($PSNR$), expressing the ratio between the maximum possible power of a signal and the power of corrupting noise affecting the fidelity of its representation.

The $PSNR$ is usually given by 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.

Thus, the expression of $PSNR$ is most easily defined via the logarithmic decibel scale related to the MSE as follows:

$$PSNR(\tilde{Y}, Y) = 10 \cdot \log_{10}\frac{MAX_I^2}{MSE(\tilde{Y}, Y)}, \tag{8}$$

when I indicates a mn-dimensional monochrome image associated to Y and MAX_I is the maximum possible pixel value of the image I.

Therefore, a natural interval extension of Eq. (8) is given as:

$$\mathbf{PSNR}(\tilde{\mathbf{Y}}, \mathbf{Y}) = 10 \cdot \log_{10}\frac{[MAX_I, MAX_I]^2}{\mathbf{MSE}(\tilde{\mathbf{Y}}, \mathbf{Y})}, \tag{9}$$

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 work.

2.3 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[1] interval library.

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.

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 and also including additional packages in the source code (HOFSCHUSTER; KRAMER; NEHER, 2008).

Besides, C-XSC is an open source library 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 (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 4.3.

---

[1]C-XSC library can be accessed at: http://www2.math.uni-wuppertal.de/∼xsc/xsc/cxsc.html

3   DISCRETE WAVELET TRANSFORMS

This Chapter introduces both Haar Wavelet Transform (HWT), presented in Section 3.1, and Daubechies Wavelet Transform (DaWT), Section 3.2, and their interval formulations.

3.1   Haar Wavelet Transform

This Section describes both decimated and undecimated formulations of the HWT, presented in Sections 3.1.1 and 3.1.2 respectively.

3.1.1   Decimated Formulation

The Haar basis was proposed in 1910, introduced by the Hungarian mathematician Alfred Haar (HAAR, 1910) in a different approach than the one considered by (DAUBECHIES, 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).

3.1.1.1   One Dimensional

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, for-

mulated 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**.

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 $C \leftarrow C'$;

Figure 1: 1D decimated Int-HWT using h and g filters.

Figure 2 shows the comparison between the standard and the interval procedures of the one-dimensional HWT assuming $l = log_2 n$ decomposition levels. It is possible to 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, becomes $sqrt(interval(n))$, 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 process, 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 char-

```
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;
    }
}
```

Figure 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.

acteristic 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.

3.1.1.2 Two Dimensional

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).

```
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' ;
```

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

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.

```
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
```

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

According to (STOLLNITZ; DEROSE; SALESIN, 1995) and shown in Figure 4,the 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
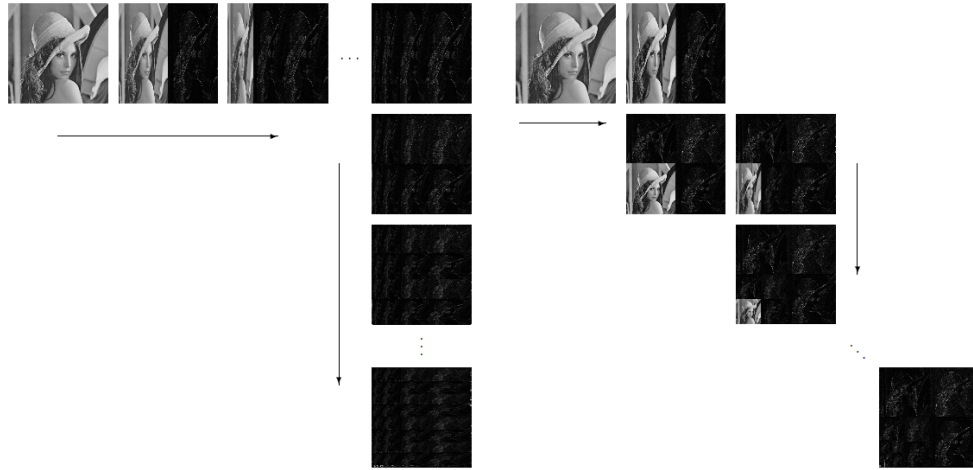
Figure 5: 2D decimated HWT with $n \geq 3$ decomposition levels.(left panel)standard algorithm; (right panel) non-standard method.

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).

InverseStandard $(C[1..h, 1..w])$
1 **for** $row \leftarrow 1$ **to** h **do**
2 $\quad\mid\quad Composition(C[row, 1..w])$;
3 **end**
4 **for** $col \leftarrow 1$ **to** w **do**
5 $\quad\mid\quad Composition(C[1..h, col])$;
6 **end**

InverseNon-Standard $(C[1..h, 1..w])$
1 **while** $h > 1$ **do**
2 $\quad\mid\quad$ **for** $row \leftarrow 1$ **to** h **do**
3 $\quad\mid\quad\quad\mid\quad CompositionStep(C[row, 1..w])$
4 $\quad\mid\quad$ **end**
5 $\quad\mid\quad$ **for** $col \leftarrow 1$ **to** w **do**
6 $\quad\mid\quad\quad\mid\quad CompositionStep(C[1..h, col])$;
7 $\quad\mid\quad$ **end**
8 **end**
9 $C \leftarrow C * h$;

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

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 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.

Thus 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.

3.1.2   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.

3.1.2.1   One Dimensional

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 $(C_j^0, C_{j+1}^0), j = 1, ..., n_0$, (assuming $C_{n_0+1}^0 = C_0^0$).

The difference now with respect to the Cascade formulation is that the range of j to select the values C_j^0 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 $D_j^1 = C_j^0 - C_j^1$, $j = 1, ..., n_0$.

The *À-Trous* HWT is again constructed recursively, 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 .

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

Figure 7: 1D À-Trous HWT: example with 2 decomposition levels

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.

Decomposition $(Input[1..n], Levels)$
1 $Declare\ C[1..Levels][1..n]$;
2 $Declare\ D[1..Levels][1..n]$;
3 $Declare\ LastC[1..n] \leftarrow Input$;
4 **for** $j \leftarrow 1$ **to** $Levels$ **do**
5 $\quad DecompositionStep(C[j], D[j], LastC)$;
6 $\quad LastC \leftarrow C[j]$;
7 **end**
8 **return** C, D ;

DecompositionStep $(C[1..n], D[1..n], LastC[1..n])$
1 **for** $i \leftarrow 1$ **to** $n - 1$ **do**
2 $\quad C[i] \leftarrow (LastC[i] + LastC[i+1])/\sqrt{[2;2]}$;
3 $\quad D[i] \leftarrow LastC[i] - C[i]$;
4 **end**
5 $C[i] \leftarrow (LastC[n] + LastC[0])/\sqrt{[2;2]}$;
6 $D[i] \leftarrow LastC[n] - C[n]$;

Figure 8: 1D À-Trous HWT- Decomposition algorithm.

Composition $(C[1..n], D[1..Levels][1..n])$
1 $Declare\ Result[1..n] \leftarrow C$;
2 **for** $i \leftarrow 1$ **to** $Levels$ **do**
3 $\quad Result \leftarrow Result + D[i]$;
4 **end**
5 **return** $Result$;

Figure 9: 1D À-Trous HWT- Composition algorithm.

3.1.2.2 Two Dimensional

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.

Figure 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, gen-

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** $i \leftarrow 1$ **to** n **do**
6      $C', D' \leftarrow Decomposition(Input[i][1..m], Levels)$;
7   **for** $j \leftarrow 1$ **to** $Levels$ **do**
8         $C[j][n][1..m] \leftarrow C'[j][1..m]$;
9         $D[j][n][1..m] \leftarrow D'[j][1..m]$;
10   **end**
11 **end**
12 **for** $i \leftarrow 1$ **to** m **do**
13      $C', D' \leftarrow Decomposition(C[Levels][1..n][i], Levels)$;
14   **for** $j \leftarrow Levels$ **to** $Levels * 2$ **do**
15         $C[j][1..n][i] \leftarrow C'[j][1..n]$;
16         $D[j][1..n][i] \leftarrow D'[j][1..n]$;
17   **end**
18 **end**
19 **return** C, D ;

Figure 11: 2D À-Trous HWT: Pseudo code for the standard matrix decomposition.

erating 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 11.

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 13.

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 12.

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.

NonStandardDecomposition $(Input[1..n][1..m], Levels)$

1   $Declare \ \ C[1..Levels * 2][1..n][1..m] \ ;$
2   $Declare \ \ D[1..Levels * 2][1..n][1..m] \ ;$
3   $Declare \ \ C'[1..m] \ ;$
4   $Declare \ \ D'[1..n] \ ;$
5   $Declare \ \ LastCMatrix[1..n][1..m] \leftarrow Input \ ;$
6   $Declare \ \ LastC[1..n] \ ;$
7   $Declare \ \ index \ ;$
8   **for** $i \leftarrow 1$ **to** $levels$ **do**
9     $index \leftarrow (i - 1) * 2 + 1 \ ;$
10   **for** $j \leftarrow 1$ **to** n **do**
11      $LastC \leftarrow LastCMatrix[j][1..m] \ ;$
12      $DecompositionStep(C', D', LastC) \ ;$
13      $C[index][j][1..m] \leftarrow C' \ ;$
14      $D[index][j][1..m] \leftarrow D' \ ;$
15   **end**
16    $LastCMatrix \leftarrow C[index][1..n][1..m] \ ;$
17    $index \leftarrow (i - 1) * 2 + 2 \ ;$
18   **for** $j \leftarrow 1$ **to** m **do**
19      $LastC \leftarrow LastCMatrix[1..n][j] \ ;$
20      $DecompositionStep(C', D', LastC) \ ;$
21      $C[index][1..n][j] \leftarrow C' \ ;$
22      $D[index][1..n][j] \leftarrow D' \ ;$
23   **end**
24    $LastCMatrix \leftarrow C[index][1..n][1..m] \ ;$
25   **end**
26   **return** $C, D \ ;$

Figure 12: Pseudo code for the Non Standard matrix decomposition.

Figure 13: Non Standard algorithm for matrix decomposition.

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 $Declare\ Result[1..n][1..m] \leftarrow C$;
2 **for** $i \leftarrow 1$ **to** $Levels$ **do**
3 $\quad | \quad Result \leftarrow Result + D[i]$;
4 **end**
5 **return** $Result$;

Figure 14: Pseudo code for À-Trous matrix composition.

3.2 Daubechies Wavelet Transform

This Section introduces the Daubechies Wavelet Transform (DaWT) as described in the literature (NIELSEN, 1998), in addition to the proposed interval filters. The one and two transformation methods are presented in Subsections 3.2.1 and 3.2.2 respectively.

3.2.1 One Dimensional

According to the description in (NIELSEN, 1998), when assuming the original formulation of the transform and given an initial vector C with n punctual values, the one-dimensional DaWT calculates scaling and wavelet coefficients 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: a vector containing the scaling and a second vector with wavelet coefficients, both with $n_1 = n/2$ points, half size of the original input vector C.

Decomposition $(C[1..n])$
1 $size \leftarrow n$;
2 **while** $size >= 4$ **do**
3 $\quad | \quad DecompositionStep(C[1..size])$;
4 $\quad | \quad size \leftarrow size/2$;
5 **end**

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

Figure 15: 1D DaWT decomposition procedure.

Figure 15 presents the $log_2(n)$ level decomposition of the Int-DaWT algorithm, formulated as an extension of the original algorithm presented in (NIELSEN, 1998).

Constants are defined as interval quantities and all arithmetics are treated as interval operations. In this sense the interval extension of the DaWT is performed by using the set of filters h' and g', shown in Eqs. (1) and (2), based on the original set of filters h and g, shown in Eqs. (5) and (6), which are normalized filters associated to the Int-DaWT.

The discrete convolution from C with h generates the next set of scaling coefficients and for computing the wavelet coefficients the filter g is considered.

$$h' = (\frac{[1;1] + \sqrt{[3;3]}}{[4;4]\sqrt{[2;2]}}, \quad \frac{[3;3] + \sqrt{[3;3]}}{[4;4]\sqrt{[2;2]}}, \quad \frac{[3;3] - \sqrt{[3;3]}}{[4;4]\sqrt{[2;2]}}, \quad \frac{[1;1] - \sqrt{[3;3]}}{[4;4]\sqrt{[2;2]}}) \quad (1)$$

$$g' = (\frac{[1;1] - \sqrt{[3;3]}}{[4;4]\sqrt{[2;2]}}, \quad \frac{-[3;3] + \sqrt{[3;3]}}{[4;4]\sqrt{[2;2]}}, \quad \frac{[3;3] + \sqrt{[3;3]}}{[4;4]\sqrt{[2;2]}}, \quad \frac{-[1;1] - \sqrt{[3;3]}}{[4;4]\sqrt{[2;2]}}) \quad (2)$$

$$ih' = (\frac{[3;3] - \sqrt{[3;3]}}{[4;4]\sqrt{[2;2]}}, \quad \frac{[3;3] + \sqrt{[3;3]}}{[4;4]\sqrt{[2;2]}}, \quad \frac{[1;1] + \sqrt{[3;3]}}{[4;4]\sqrt{[2;2]}}, \quad \frac{[1;1] - \sqrt{[3;3]}}{[4;4]\sqrt{[2;2]}}) \quad (3)$$

$$ig' = (\frac{[1;1] - \sqrt{[3;3]}}{[4;4]\sqrt{[2;2]}}, \quad \frac{-[1;1] - \sqrt{[3;3]}}{[4;4]\sqrt{[2;2]}}, \quad \frac{[3;3] + \sqrt{[3;3]}}{[4;4]\sqrt{[2;2]}}, \quad \frac{-[1;1] + \sqrt{[3;3]}}{[4;4]\sqrt{[2;2]}}) \quad (4)$$

Since the DaWT 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 decomposition level produces a second pair of half-sized vectors, one for the new $n_2 = n_1/2$ scaling coefficients 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 two scaling coefficients are obtained.

$$h = (\quad \frac{1 + \sqrt{3}}{4\sqrt{2}}, \quad \frac{3 + \sqrt{3}}{4\sqrt{2}}, \quad \frac{3 - \sqrt{3}}{4\sqrt{2}}, \quad \frac{1 - \sqrt{3}}{4\sqrt{2}} \quad) \quad (5)$$

$$g = (\quad \frac{1 - \sqrt{3}}{4\sqrt{2}}, \quad \frac{-3 + \sqrt{3}}{4\sqrt{2}}, \quad \frac{3 + \sqrt{3}}{4\sqrt{2}}, \quad \frac{-1 - \sqrt{3}}{4\sqrt{2}} \quad) \quad (6)$$

$$ih = (\quad \frac{3 - \sqrt{3}}{4\sqrt{2}}, \quad \frac{3 + \sqrt{3}}{4\sqrt{2}}, \quad \frac{1 + \sqrt{3}}{4\sqrt{2}}, \quad \frac{1 - \sqrt{3}}{4\sqrt{2}} \quad) \quad (7)$$

$$ig = (\quad \frac{1 - \sqrt{3}}{4\sqrt{2}}, \quad \frac{-1 - \sqrt{3}}{4\sqrt{2}}, \quad \frac{3 + \sqrt{3}}{4\sqrt{2}}, \quad \frac{-1 + \sqrt{3}}{4\sqrt{2}} \quad) \quad (8)$$

Using the sets of all wavelet coefficients, resulted from the decomposition process, together with the scaling coefficients in the lowest level of the decomposed data, it is possible to perform the inverse transformation, also known as composition, reconstructing the original vector, which is an important characteristic of DWTs.

Composition $(C[1..n])$
1   $size \leftarrow 4$;
2   **while** $size <= n$ **do**
3      $CompositionStep(C[1..size])$;
4      $size \leftarrow size * 2$;
5   **end**

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

Figure 16: Inverse 1D DaWT, Composition process to reconstruct $l = log_2 n$ levels.

The pseudo-code that performs the level composition of the 1D DaWT is represented in Figure 16. The filter sets ih' and ig', presented in Eqs. (3) and (4), are based on the original filter sets ih and ig, described in Eqs. (7) and (8).

According to (NIELSEN, 1998), 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, calculated by using filter sets ih' and ig', 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.

3.2.2 Two Dimensional

The DaWT can be extended for two-dimensional vectors by using the same concept presented for the HWT in Section 3.1.1.2. According to the procedure described in (NIELSEN, 1998), the fast algorithm for the 2D DaWT 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 (NIELSEN, 1998), the 2D DaWT 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 DaWT were applied to the entire set of rows of the matrix, the same 1D procedure is applied to the columns of the resulting matrix, as many levels as done for the rows. This completes the 2D transformation, assuming $l = log_2 n$ decomposition levels. Figure 5 illustrates the application of this algorithm.

According to (NIELSEN, 1998) and shown in Figure 4,the non-standard method is another option for the 2D DaWT. 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 DaWT, 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 of the DaWT, Figures 15, 16, 4 and 6, can be thought as a convolution of the input data with two normalized pairs of filters, h and g shown in Eqs. (5) and (6), that represent the transform.

The normalized transform preserves the energy of the entire set of coefficients obtained through the process. Therefore, by using the same strategy implemented for the HWT, besides the standard and non-standard algorithms, the DaWT can be performed using normalized and non-normalized filters.

Thus combining the two possible algorithms with these two possible filter choices, there are four ways to compute the decimated 2D DaWT available: (i) non-standard, non-normalized; (ii) non-standard, normalized; (iii) standard, non-normalized; and (iv) standard, normalized. The non-normalized filters are presented in Section 4.2.1.

4 DEVELOPED OPTIMAL FORMULATIONS

When the normalized filters are considered in the original algorithms of the HWT and DaWT (STOLLNITZ; DEROSE; SALESIN, 1995; STARCK; FADILI; MURTAGH, 2007; NIELSEN, 1998), 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 by performing algebraic simplifications in order to eliminate the computation of these non computable values, $2^{j/2}$, whenever possible, reducing the error involved in the process.

Therefore the developed algorithms produce more trustworthy results in comparison to the literature.

The basic idea for the algebraic simplifications is based on using the non-normalized transform, which introduces no computing 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 or composition.

4.1 Haar Wavelet Transform

This Section aims to describe the developed optimizations for the HWT where the optimized decimated formulation is presented in Section 4.1.1 and its undecimated form in Section 4.1.2.

4.1.1 Decimated Formulation

This Section presents the developed optimizations for both one and two dimensional approaches of the HWT in Sections 4.1.1.1 and 4.1.1.2.

4.1.1.1 One Dimensional

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 17.

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 17, 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**

Figure 17: Example of the one-dimensional optimization.

Looking at the original decomposition procedure, shown in Figure 1, the number of operations of the $DecompositionStep$ is $n+1$, and for the main $decomposition$ algorithm the number of operations is $n + \log(n) \cdot (n + 2) + 1$. By the proposed simplification they are reduced to $n + n\log(n)$ operations. 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 number of operations for non-normalized transformation of a n sized vector is $n\log(n)$, representing the pseudo-code shown in Figure 18.

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

Figure 18: Int-HWT, decimated version: non-normalized Decomposition process and its step procedure.

The normalization procedure, presented in Figure 19, 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 number of operations is n.

By performing a non-normalized transform and the normalization procedure afterwards, the number of operations is $n\log(n) + n$, which is the same amount of operations from the normalized transformation.

The goal is therefore the gain on accuracy obtained by avoiding high number of divisions using $\sqrt{2}$ from the original $DecompositionStep$ algorithm, as shown in Figure 35.

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

1 $levels \leftarrow log_2(n)$;
2 **for** $i \leftarrow 1$ **to** $levels$ **do**
3 $\quad start \leftarrow i^2$;
4 $\quad end \leftarrow (i+1)^2$;
5 $\quad$ **for** $j \leftarrow start$ **to** end **do**
6 $\quad \quad | \quad C[j] \leftarrow C[j] * 2 \wedge (-(i/2))$;
7 $\quad$ **end**
8 **end**

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

1 $levels \leftarrow log_2(n)$;
2 **for** $i \leftarrow 1$ **to** $levels$ **do**
3 $\quad start \leftarrow i^2$;
4 $\quad end \leftarrow (i+1)^2$;
5 $\quad$ **for** $j \leftarrow start$ **to** end **do**
6 $\quad \quad | \quad C[j] \leftarrow C[j] / 2 \wedge (-(i/2))$;
7 $\quad$ **end**
8 **end**

Figure 19: 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 19, is executed before the transform begins.

The original algorithms for composition and its step procedure, Figure 3, behave similarly to decomposition. The number of operations of the $Composition$ procedure is $n \log(n) + n$ if the normalized formulation is considered, otherwise it is $n \log(n)$. The denormalization step, also shown in Figure 19, is almost the same $Normalization$ algorithm, whose complexity is 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 same amount of operations from the original version, $n \log(n)$, but its results are more exact due to the normalization step, avoiding the calculation of $\sqrt{2}$ on every iteration.

4.1.1.2 Two Dimensional

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 (STOLLNITZ; DEROSE; SALESIN, 1995), there are two main algorithms for composition and decomposition of matrices, know as Standard and Non-Standard procedures, what suggests distinct normalization procedures for both approaches.

The study of the original algorithms (STOLLNITZ; DEROSE; SALESIN, 1995) helped to recognize patterns of normalization factors. These patterns were analysed and employed in the development of the normalization procedures for both algorithms, assuming the decimated version of the HWT.

These patterns are illustrated in Figure 20, 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 expressed by R defined in Eq. (9):

$$R = 2^{\frac{-(j'+j'')}{2}}$$ (9)

where $0 \leq j', j'' \leq (\log_2 n) - 1$ and n indicates the matrix order.
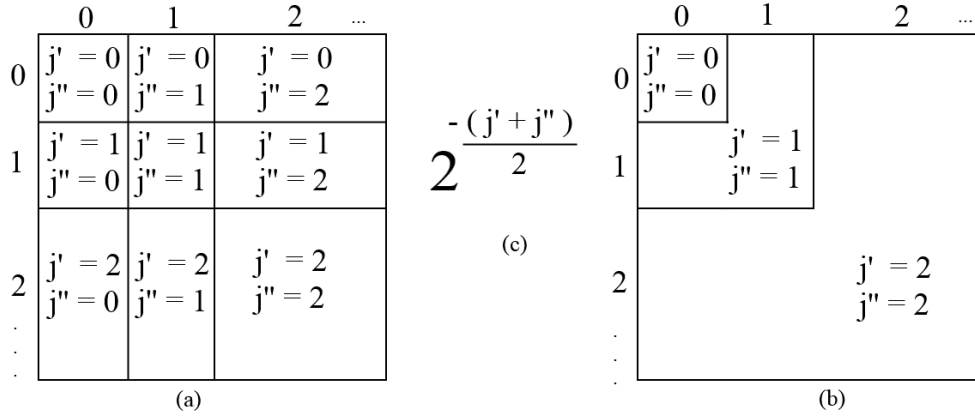
Figure 20: Int-HWT, decimated version: (a) standard, (b) non-standard normalization patterns and (c) rule for normalization factors.

By analyzing the original standard algorithm, Figure 4, it depends on the original normalized decomposition procedure, whose number of operations is $n \log(n) + n$, executing it for every row and column of the input matrix.

Considering a nxn matrix the number of operations of the original standard algorithm is $n \cdot (n \log(n) + n) + n \cdot (n \log(n) + n)$, therefore $2n^2 \log(n) + 2n^2$. Performing the non-normalized decomposition algorithm, discussed in Section 4.1.1.1, it reduces the calculations by avoiding intermediate normalization steps.

The non-normalized standard decomposition is performed with $2n^2 \log(n)$ operations, and the last step is to normalize the results using the $StandardNormalization$ algorithm, presented in Figure 21, and since it is operating one time on each coefficient of the input matrix its number of operations can be expressed by n^2.

The number of operations by performing the non-normalized standard decomposition and the standard normalization step can be expressed by $2n^2 \log(n) + n^2$, which is faster than the original normalized procedure by avoiding a second iteration of n^2 operations 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 $2n^2 \log(n) + 2n^2$ time, while its optimized version operates in $2n^2 \log(n) + n^2$.

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
```

Figure 21: 2D decimated Int-HWT: Normalization step for the standard decomposition process.

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 $O(n^2)$ 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 (n operations) on portions of the input matrix, depending on the level of transformation. The original non-standard decomposition is executed with $n^2 + \log(\log(n) + \log(n))$ operations.

The non-normalized non-standard decomposition is performed on $\log(\log(n) + \log(n))$ operations, and its $Non - StandardNormalization$ algorithm, shown in Figure 22, is performed on n^2 operations.

The number of operations of these procedures is $n^2 + \log(\log(n) + \log(n))$, which is the same from the original normalized non-standard algorithm. The gain of time presented in Figure 36 was obtained performing simple divisions by 2 which are faster than using $\sqrt{2}$ as divisor when analysing processor microinstructions.

```
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
```

Figure 22: Int-HWT, decimated version: Normalization step for the non-standard decomposition process.

4.1.2 Undecimated Formulation

This Section presents the developed optimizations for both one and two dimensional approaches of the HWT À-Trous in Sections 4.1.2.1 and 4.1.2.2.

4.1.2.1 One Dimensional

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 number of operations of both $DecompositionStep$ and $Decomposition$ procedures are $2n$ and $Levels \cdot 2n$, 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 23, reducing the corresponding amount of operations is $2n$ and $2n \cdot Levels$, the same complexity as the original normalized transform.

The normalization procedure, presented in Figure 24, 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

DecompositionStep $(C[1..n], D[1..n], LastC[1..n])$
1 **for** $i \leftarrow 1$ **to** $n - 1$ **do**
2 $\quad C[i] \leftarrow (LastC[i] + LastC[i+1]) / [2; 2]$;
3 $\quad D[i] \leftarrow LastC[i] - C[i]$;
4 **end**
5 $C[i] \leftarrow (LastC[n] + LastC[0]) / [2; 2]$;
6 $D[i] \leftarrow LastC[n] - C[n]$;

Figure 23: 2D undecimated Int-HWT: non-normalized decomposition step.

factor, as shown in Eq. (9), for every resultant vector from the transform, configuring $n \cdot Levels$ operations from line 1 to 5 of the pseudo-code. With the new set of scalar values all wavelet coefficients are then recalculated, configuring $Levels \cdot (n+3) + 2$ from line 6 to 16, which can be simplified to $n \cdot Levels$. The overall amount of operations of the procedure is $2n \cdot Levels$.

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

Figure 24: 1D undecimated Int-HWT: normalization procedure.

By performing a non-normalized vector transform and the normalization procedure, the number of operations is $4n \cdot Levels$, which is more expensive than the original normalized formulation. The gain obtained with the new implementation is on the accuracy of results by avoiding high number of divisions using $\sqrt{2}$ from the original $DecompositionStep$ algorithm, as shown in Figure 35.

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 number of operations of this procedure is $n \cdot Levels$) and can be used for both normalized and non-normalized transformations, there is no need for optimization.

4.1.2.2 Two Dimension

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.

In order 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 11 the original standard procedure is presented. It performs the decomposition of every line and level of a nxm matrix, configuring an amount of $n \cdot 2m \cdot Levels$ operations.

The analog procedure for all columns of the same matrix has $m \cdot 2n \cdot Levels$. The entire procedure is expressed by $n \cdot 2m \cdot Levels + m \cdot 2n \cdot Levels$, which can be simplified to $4n \cdot m \cdot Levels$.

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 18, so the complexity of executing the normalized and non-normalized are the same.

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

Figure 25: 2D undecimated Int-HWT: normalization procedure.

The next step in order to normalize the transformed values is to perform the $A\text{-}TrousMatrixNormalization$ procedure, shown in Figure 25, which is executed with $4n \cdot m \cdot Levels$ operations.

Therefore, the optimized decomposition procedure developed costs $8n \cdot m \cdot Levels$, twice the cost of the original normalized formulation, but tests indicate that our approach results in more exact values as shown in Figure 37.

Table 1: Number of operations of the Decimated HWT.

	1D	2D Standard	2D Non-Standard
Original	$n \log(n) + n$	$2n^2 \log(n) + 2n^2$	$n^2 + \log(\log(n) + \log(n))$
developed	$n \log(n) + n$	$2n^2 \log(n) + n^2$	$n^2 + \log(\log(n) + \log(n))$

Table 2: Number of operations of the Undecimated HWT for decomposition.

	1D	2D
Original	$2 \cdot n \cdot Levels$	$4 \cdot n \cdot m \cdot Levels$
developed	$4 \cdot n \cdot Levels$	$8 \cdot n \cdot m \cdot Levels$

The Tables 1 and 2 present a comparison of the studied algorithmic complexity for both approaches of the WHT. The expressions show that the developed 1D and 2D Non-Standard HWT Decimated are executed with the same amount of operations compared to their original formulation. The developed 2D Standard HWT is performed with less operations than the original algorithm from the literature. The developed HWT Undecimated needs more operations to be performed comparing to the one from the literature.

4.2 Daubechies Wavelet Transform

This Section discribes the developed optimizations for both 1D and 2D DaWT, in addition to new filter sets for calculation accuracy, presented in Subsections 4.2.1 and 4.2.2 respectively.

4.2.1 One Dimensional

The normalized decomposition procedure for the DaWT 1D is executed in the following order:

(i) firstly, a non-normalized decomposition is made, as indicated by the pseudo-code in Figure 17; and,

(ii) 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 last one stage represents the same process used in the HWT presented in Figure 17, depending on j, the computation of $\sqrt{2}$ may not be necessary, avoiding computing error in this case.

As presented in Figure 26, in the decomposition steps there are calculations involving irrational numbers.

Decomposition $(C[1..n], Normal)$
1 $size \leftarrow n$;
2 **while** $size >= 4$ **do**
3 $\quad$ $DecompositionStep(C[1..size], Normal)$;
4 $\quad$ $size \leftarrow size/2$;
5 **end**

DecompositionStep $(C[1..n], Normal)$
1 $i \leftarrow 1$;
2 $j \leftarrow 1$;
3 $half \leftarrow n/2$;
4 **if** $Normal = True$ **then**
5 $\quad$ $denom \leftarrow 4\sqrt{2}$;
6 **else**
7 $\quad$ $denom \leftarrow 4$;
8 **end**
9 **while** $j <= n - 3$ **do**
10 $\quad$ $a \leftarrow C[j]$;
11 $\quad$ $b \leftarrow C[j+1]$;
12 $\quad$ $c \leftarrow C[j+2]$;
13 $\quad$ $d \leftarrow C[j+3]$;
14 $\quad$ $t1 \leftarrow a - c$;
15 $\quad$ $t2 \leftarrow b - d$;
16 $\quad$ $C'[i] \leftarrow (a + d + 3.(b + c) + \sqrt{3}.(t1 + t2))/denom$;
17 $\quad$ $C'[i + half] \leftarrow (a - d + 3.(c - b) + \sqrt{3}.(t2 - t1))/denom$;
18 $\quad$ $j \leftarrow j + 2$;
19 $\quad$ $i \leftarrow i + 1$;
20 **end**
21 $a \leftarrow C[n-1]$;
22 $b \leftarrow C[n]$;
23 $c \leftarrow C[0]$;
24 $d \leftarrow C[1]$;
25 $t1 \leftarrow a - c$;
26 $t2 \leftarrow b - d$;
27 $C'[i] \leftarrow (a + d + 3.(b + c) + \sqrt{3}.(t1 + t2))/denom$;
28 $C'[i + half] \leftarrow (a - d + 3.(c - b) + \sqrt{3}.(t2 - t1))/denom$;
29 $C \leftarrow C'$;

Figure 26: DaWT Decomposition pseudo-code for normalized and non-normalized transformations.

The next step of the optimization is to reduce the number of operations using $\sqrt{3}$ by considering the filters h, g, ih and ig, described in the following paragraphs.

Firstly, the algebraic manipulations presented in Eq.(10), express the simplifications related to parameter h, expressed by Eq.(5), by taking a, b, c, and d as different different coefficients.

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

In the sequence, Eq.(11) provides the main simplifications related to parameter g, based on previous expression in Eq.(6):

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

Other ones are described by (12) and (13), according with their previous definitions in Eqs.(7) and Eq.(8), respectively:

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

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

$$=\; \frac{3a - a\sqrt{3} + 3b + b\sqrt{3} + c + c\sqrt{3} + d - d\sqrt{3}}{4\sqrt{2}}$$

$$=\; \frac{c + d + 3\left(a+b\right) + \sqrt{3}\left(b+c-a-d\right)}{4\sqrt{2}} \tag{12}$$

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

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

$$=\; \frac{a - a\sqrt{3} - b - b\sqrt{3} + 3c + c\sqrt{3} - 3d + d\sqrt{3}}{4\sqrt{2}}$$

$$=\; \frac{a - b + 3\left(c-d\right) + \sqrt{3}\left(c+d-a-b\right)}{4\sqrt{2}} \tag{13}$$

Looking at the original decomposition procedure, shown in Figure 15, the number of operations of the $DecompositionStep$ is $3 + (n/2) \cdot 4 + 3$, which can be simplified to $\mathrm{O}(n)$, and for the main $Decomposition$ algorithm the amount of operations is $1 + \log(n) \cdot (n+1)$, that can be simplified to $\mathrm{O}(n\log(n))$.

The number of operations of the developed algorithms for decomposition, shown in Figure 26, are expressed by $5 + (n/2) \cdot 9 + 8$, or just n, for the developed $DecompositionStep$ procedure and $1 + \log(n) \cdot (n+1)$, or just $n\log(n)$ for the developed $Decomposition$ routine.

Therefore, both original and developed algorithms share the same complexity for normalized filters.

The optimization step to avoid calculations of $\sqrt{2}$ is to perform a non-normalized decomposition by using 4 instead of $4\sqrt{2}$ as divider in filters h and g.

The next step is to normalize the coefficients after their convolution using the pseudo-code presented in Figure 27.

Since it is executed once on every coefficient of a n sized vector, its complexity is expressed by n. Performing a non-normalized transformation and the normalization procedure afterwards, the number of operations is $n\log(n) + n$, making the developed

optimization slower than the original formulation.

The goal is therefore the gain on accuracy obtained by avoiding a high number of operations using $\sqrt{2}$ and $\sqrt{3}$ from the original $DecompositionStep$ algorithm, as shown in Figure 15.

Normalization $(C[1..n])$
1   $levels \leftarrow log_2(n)$;
2   **for** $i \leftarrow 1$ **to** $levels$ **do**
3   **if** $i = 1$ **then**
4       $start \leftarrow 1$;
5   **else**
6       $start \leftarrow 2 \wedge i$;
7   **end**
8     $end \leftarrow 2 \wedge (i + 1)$;
9   **for** $j \leftarrow start$ **to** end **do**
10      $C[j] \leftarrow C[j].2 \wedge (-(i/2))$;
11   **end**
12 **end**

Figure 27: DaWT one dimensional normalization procedure.

As presented in Section 3.1.1.1 for the HWT, the optimized composition procedure follows the same basic idea to reduce computation of $\sqrt{2}$, it performs the non-normalized inverse transform by using a denormalization step, shown in Figure 19 before the transformation begins.

In contrast to the HWT optimization strategy for composing the results, the DaWT has to use normalized filters excluding the denormalization step. It is due to the order of coefficients which are calculated by the inverse DaWT, shown in Figure 29, which needs the values to be normalized in order to scale the signals properly.

Composition $(C[1..n], Normal)$
1   $size \leftarrow 4$;
2   **while** $size <= n$ **do**
3     $CompositionStep(C[1..size], Normal)$;
4     $size \leftarrow 2.size$;
5   **end**

Figure 28: DaWT one dimensional composition procedure.

In the 1D DaWT composition procedure and its step procedure, as shown in Figure 16, behave similarly to decomposition where its number of operations is expressed by $n \log(n)$.

The developed optimal $Composition$ procedure, shown in Figure 28, shares the same number of operations of $n \log(n)$, but its results are more exact due to the simplified filters, avoiding the unnecessary calculation of $\sqrt{2}$ and $\sqrt{3}$ on every iteration.

CompositionStep $(C[1..n], Normal)$

```
1  i ← 1 ;
2  j ← 1 ;
3  half ← n/2 ;
4  if Normal = True then
5  │   denom ← 4√2 ;
6  else
7  │   denom ← 4 ;
8  end
9  a ← C[half] ;
10 b ← C[n] ;
11 c ← C[1] ;
12 d ← C[half + 1] ;
13 t1 ← c − a ;
14 t2 ← b − d ;
15 C′[1] ← (c + d + 3.(a + b) + √3.(t1 + t2))/denom ;
16 C′[2] ← (a − b + 3.(c − d) + √3.(t1 − t2))/denom ;
17 while j <= n − 3 do
18 │   a ← C[j] ;
19 │   b ← C[j + 1] ;
20 │   c ← C[j + 2] ;
21 │   d ← C[j + 3] ;
22 │   t1 ← a − c ;
23 │   t2 ← b − d ;
24 │   C′[i] ← (c + d + 3.(a + b) + √3.(t1 + t2))/denom ;
25 │   C′[i + half] ← (a − b + 3.(c − d) + √3.(t1 − t2))/denom ;
26 │   j ← j + 2 ;
27 │   i ← i + 1 ;
28 end
29 C ← C′ ;
```

Figure 29: DaWT one dimensional step composition procedure.

4.2.2 Two Dimensional

The simplifications implemented for the 2D DaWT 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.

In order to compose or decompose a matrix, the 1D transform is applied in all rows and in all columns of the input matrix in sequence. However, according to (STOLL-NITZ; DEROSE; SALESIN, 1995), there are two main algorithms for composition and decomposition of matrices, known as Standard and Non-Standard procedures.

During the study of the original DaWT formulation (NIELSEN, 1998), patterns of normalization factors were recognized. These patterns were analysed and employed in the development of the normalized procedures for the Standard algorithm. In contrast

to the 2D HWT optimization, the DaWT Non-Standard procedure does not use the normalization step due to its behavior.

The normalization patterns are illustrated in Figure 30, where three decomposition levels are presented for $16\text{x}16$ matrices. The parameters j' and j'' indicate the normalization levels. The rule to calculate these normalization factors is expressed by R defined in Eq. (9).

Figure 30: DaWT (a) Standard normalization pattern, (b) level combinations.

Standard $(C[1..h, 1..w])$
1 **for** $row \leftarrow 1$ **to** h **do**
2 $\quad$ $Decomposition(C[row, 1..w])$;
3 **end**
4 $StandardStepNormalization(C[1..h, 1..w], True)$;
5 **for** $col \leftarrow 1$ **to** w **do**
6 $\quad$ $Decomposition(C[1..h, col])$;
7 **end**
8 $StandardStepNormalization(C[1..h, 1..w], False)$;

Figure 31: DaWT 2D Standard normalization step decomposition process.

The analysis of the original Standard algorithm inherited from the HWT depends on the original normalized decomposition procedure, as shown in Figure 4. By executing such algorithm for every row and column of the input matrix, one can observe the corresponding number of operations analysis as $n \log(n)$. Considering a matrix $C_{n \times n}$, the number of operations of the original standard algorithm is $n(n \log(n)) + n(n \log(n))$, or just $n^2 \log(n)$.

In opposition to the HWT, the optimal normalized DaWT Standard decomposition has to perform its normalization step, shown in Figure 32, in between the decomposition of rows and columns, and a second time at the end of the procedure, as presented in Figure 31.

StandardStepNormalization $(C[1..h, 1..w], Rows)$
1   $Levels \leftarrow \log_2(h)$;
2   **for** $Level \leftarrow 1$ **to** $Levels$ **do**
3     $Start \leftarrow 2 \wedge Level$;
4     $End \leftarrow 2 \wedge (Level + 1)$;
5     $Factor \leftarrow 2 \wedge (Level/2)$;
6   **for** $j \leftarrow Start$ **to** End **do**
7   **if** $Rows = True$ **then**
8   **for** $i \leftarrow 1$ **to** h **do**
9           $C[i][j] \leftarrow C[i][j]/Factor$;
10   **end**
11   **else**
12   **for** $i \leftarrow 1$ **to** w **do**
13           $C[j][i] \leftarrow C[j][i]/Factor$;
14   **end**
15   **end**
16   **end**
17 **end**

Figure 32: DaWT 2D Standard normalization step decomposition process.

Since the normalization step procedure is a kernel that will apply the right normalization factor to each coefficient in a nxn matrix, its number of operations can be expressed as n^2. Therefore, the developed optimal Standard normalized decomposition procedure is performed with number of operations of $n^2 \log(n) + n^2$.

Considering this, the developed procedure is slower than the original but it is more accurated due to the new proposed filter set.

The Non-Standard procedure has the same limitation from the Standard where the column operations depend on normalized row operations.

Because of this limitation the Non-Standard algorithm would have to normalize each step level of decomposition before the next operation, instead of adding a normalization step in between each row or column decomposition it is better to perform a normalized procedure using the new set of filters.

This way the Non-Standard algorithm has the same complexity as the same developed procedure from the HWT, as seen in Section 4.1.1.2.

By the same reason that the 1D DaWT does not use any denormalization procedure, the 2D Standard and Non-Standard DaWT does not perform denormalization of its coefficients before composition either.

Both approaches inherit the 2D HWT composition procedures, keeping their original complexity as discussed in Section 4.1.1.2. However, the developed 2D DaWT Composition routines in Int-DWTs library present better results as consequence of using the new set of simplified filters, as discussed in Section 4.2.1.

Table 3: Number of operations of the DaWT for decomposition.

	1D	2D Standard	2D Non-Standard
Original	$n \log(n)$	$n^2 \log(n)$	$n^2 + \log(\log(n) + \log(n))$
developed	$n \log(n) + n$	$n^2 \log(n) + n^2$	$n^2 + \log(\log(n) + \log(n))$

The Table 3 shows that the developed DaWT is perform using more operations compared to its original formulation based on the literature.

4.3   Interval Compression

The main goal of compression procedures is to express an initial data set with the smaller number 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 than others. Therefore, keeping only the most significant ones, the truncated series represents the compressed 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 33, which 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 at the initial data representation.

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 the details details which must be ig-
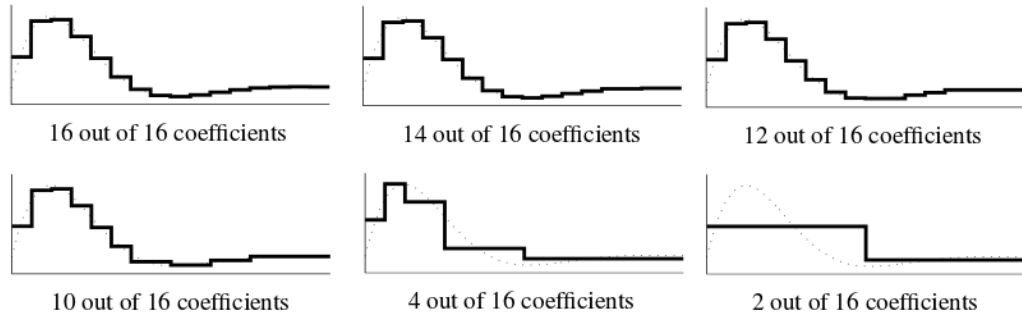
Figure 33: Approximation of functions after compression.

nored, considers a set of simple punctual comparisons.

When treating with interval data, the parameter τ becomes a degenerated interval based on its punctual input value, but can also incorporate some level of error if necessary. 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 considered by (DONOHO, 1995) for its threshold operators.

Data: $C[0...n]$: real
$i \leftarrow 0$;
for $i < n$ **do**
    if $|C[i]| < \tau$ **then**
        $C[i] = 0$;
    end
    $i = i + 1$;
end

Data: $C[0...n]$: interval
$i \leftarrow 0$;
for $i < n$ **do**
    if $\overline{C[i]} < \underline{\tau}$ **then**
        $C[i] = [0 : 0]$;
    end
    $i = i + 1$;
end

Data: $C[0...n]$: interval
$mid\tau = (\overline{\tau} + \underline{\tau})/2$;
$i \leftarrow 0$;
for $i < n$ **do**
    $midC = (\overline{C[i]} + \underline{C[i]})/2$;
    if $midC < mid\tau$ **then**
        $C[i] = [0 : 0]$;
    end
    $i = i + 1$;
end

Figure 34: 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 infimum of the information and the supremum of the interval parameter. 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 infimum of the information and the center of the threshold. This procedure grants that most possible punctual coefficients are between the punctual values of $\overline{\tau}$ and $\underline{\tau}$.

5 TESTS AND RESULTS

The numerical validation of algorithms proposed and described in Section 4 is 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-HWT 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 carried out by executing each function 30 times, mean and standard deviation values were obtained from those times and used to generate the figures presented in the following subsections.

The tests were executed on an Intel ® Core ™ i7 950 Processor @ 3.07GHz, 6GB RAM DDR3 @ 1066MHz, Windows 10 OS, compiled using Microsoft Visual C++ Compiler for Visual Studio 2013 on x64 Release compilation target.

5.1 Haar Wavelet Transform

In order to compute tests using the 1D HWT and its interval extension, a vector filled with 1048576 random values is used as input. For tests using 2D HWT algorithms, the input is generated from a 1024x1024 matrix of random values, configuring 1048576 values.

5.1.1 One Dimensional

The results shown in Figure 35a describe the performance of both decomposition and composition procedures from Section 4.1.1.1, targeting the decimated and undecimated 1D HWT.

The data indicates that the decimated algorithms developed in this work are 46% faster than the results found in (STOLLNITZ; DEROSE; SALESIN, 1995), when composing the results from decomposing the input.

Decomposition, Composition and the combination of both operations, are executed $30\times$, taking $15.46ms$, $16.4317ms$ and $31.1012ms$ with standard deviations of $0.757038ms$, $0.155433ms$ and $0.117835ms$ respectively.

The accuracy gained is from 95% up to 99.8%, meaning that the developed algorithms generate much more exact results compared to those from the literature.

Euclidean Distance (EUC) and Mean Square Error (MSE) are around 99.8% of gain, and Peak-to-Noise Ratio ($PSNR$) is about 24.4%.

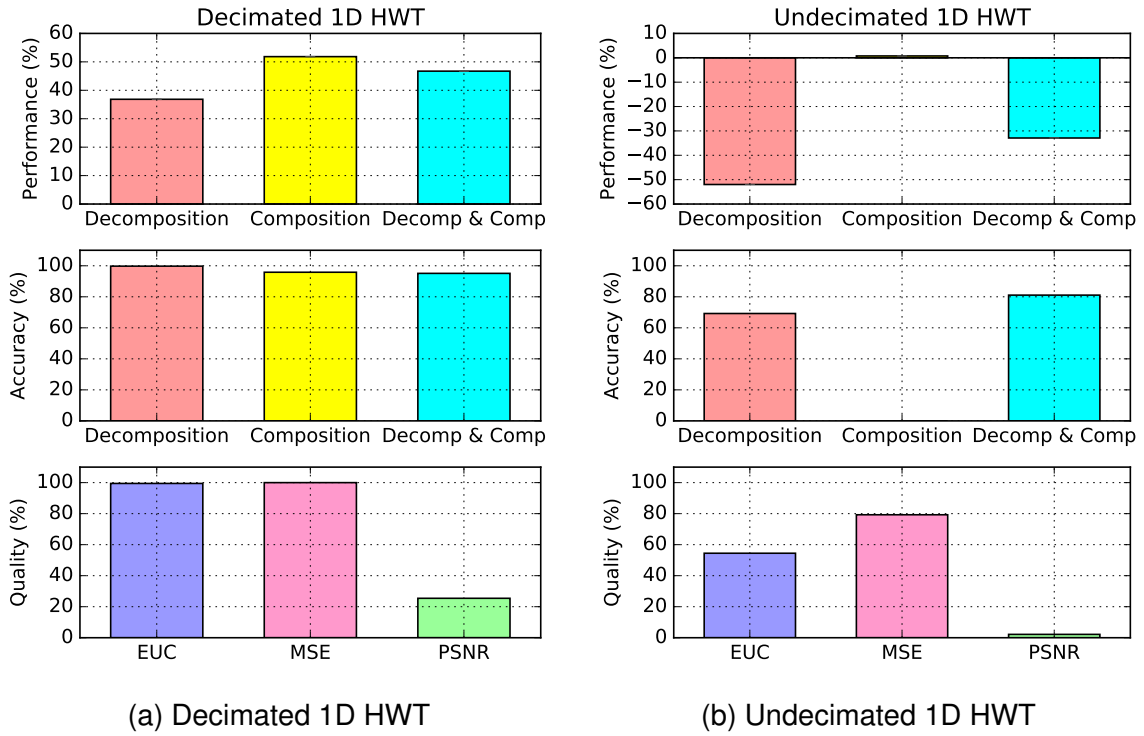(a) Decimated 1D HWT   (b) Undecimated 1D HWT

Figure 35: Performance, accuracy and metric gains for the 1D HWT.

The data also indicate that the developed undecimated algorithms are about 30% slower than the original formulations (STARCK; FADILI; MURTAGH, 2007) when composing the results from the decomposition step, shown in Figure 35b.

The average times of execution for decomposition, composition and the combination of both are respectively $43.4929ms$, $12.0031ms$ and $55.0647ms$, and the corresponding standard deviation are $0.385967ms$, $0.440633ms$ and $0.371797ms$.

The undecimated formulation adds no error to calculations on the composition step, considering that the input does not contain errors, which explains the lack of bars in Figure 35b. The accuracy gain when performing the combination of both decomposition and composition is around 81%.

The EUC is about 54.5%, and MSE is 79.3%, while $PSNR$ is 2.15%. These results show that despite the loss of performance, the developed algorithms are more accurate than the algorithms from the literature (STARCK; FADILI; MURTAGH, 2007).

5.1.2 Two Dimensional

This Section presents the results of both decimated and undecimated formulations of the HWT in Sections 5.1.2.1 and 5.1.2.2 respectively.

5.1.2.1 Decimated

Data from Figure 36 show the performance of both standard and non-standard approaches for the decimated 2D HWT. The results presented in Figure 36a show a performance boost of $58,1\%$ during the composition step. The average time of execution of the decomposition, composition and the combination of both are $42.2897ms$, $32.5122ms$ and $74.6253ms$, and their standard deviations are $0.271538ms$, $0.487003ms$ and $0.314095ms$ respectively. The accuracy gain of decomposition, composition and the combination of both are 99.8%, 93.5% and 98.3%, meaning that the results provided from the developed algorithms are much more exact compared to the literature (STOLLNITZ; DEROSE; SALESIN, 1995). The EUC, MSE and PSNR metrics are 99.8%, 99.9% and 19.2%, showing that the developed algorithms are more accurate than the algorithms from the literature (STOLLNITZ; DEROSE; SALESIN, 1995).

(a) Decimated 2D Standard HWT   (b) Decimated 2D Non-Standard HWT

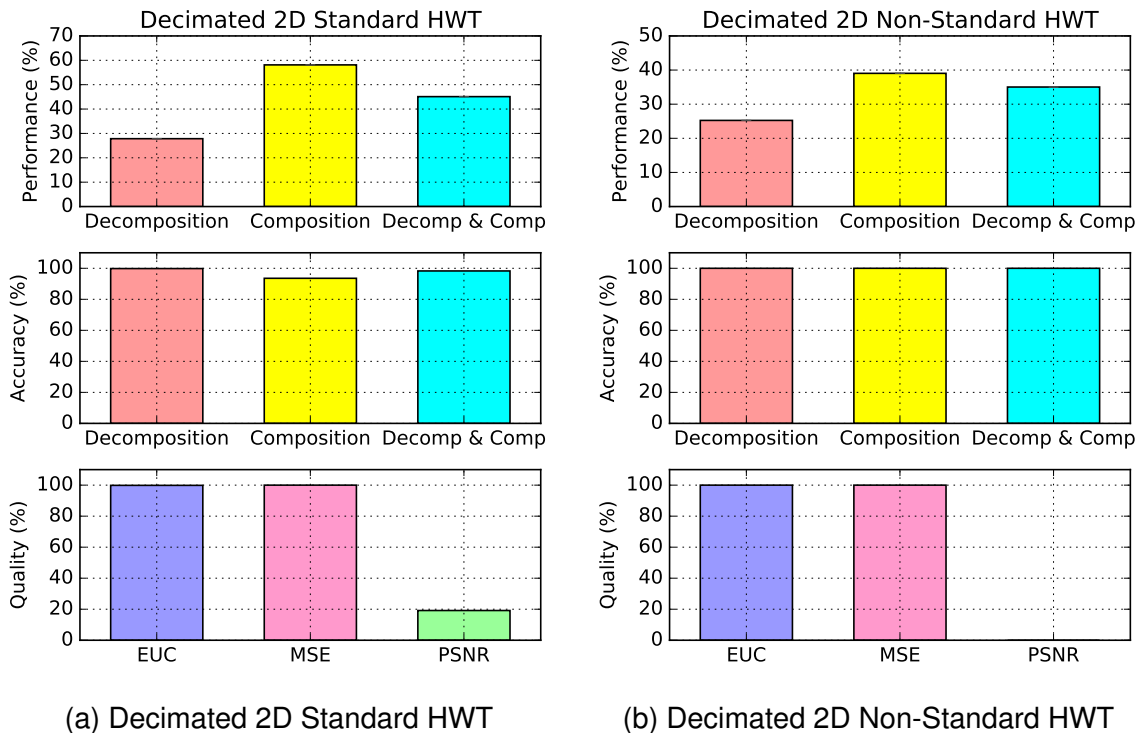Figure 36: Times of execution and error measurement for the 2D HWT using 1024x1024 matrix with random values.

Figure 36b shows the performance, accuracy and metric gains for the decimated 2D non-standard HWT comparing to the original algorithms found in (STOLLNITZ; DEROSE; SALESIN, 1995). The average time of execution for decomposition, composition and both operations are $37.8028ms$, $49.1ms$ and $86.0812ms$, and their standard

deviations are respectively $0.985911ms$, $0.362122ms$ and $0.255491ms$. It is important to verify that, as shown in Figure 20, the developed non-standard method does not need to perform calculations of $\sqrt{2}$ due to the implemented algebraic simplifications. Therefore, the corresponding calculations do not increase the error during this process, indicated by accuracy values at 100% for all three methods. Due to the lack of error, the EUC and MSE are also shown at 100%. The PSNR metric cannot be calculated due to division by 0, since it uses MSE as divisor and its value is 0.

5.1.2.2 Undecimated

Figure 37 shows the performance impact of both standard and non-standard approaches for the undecimated 2D HWT.

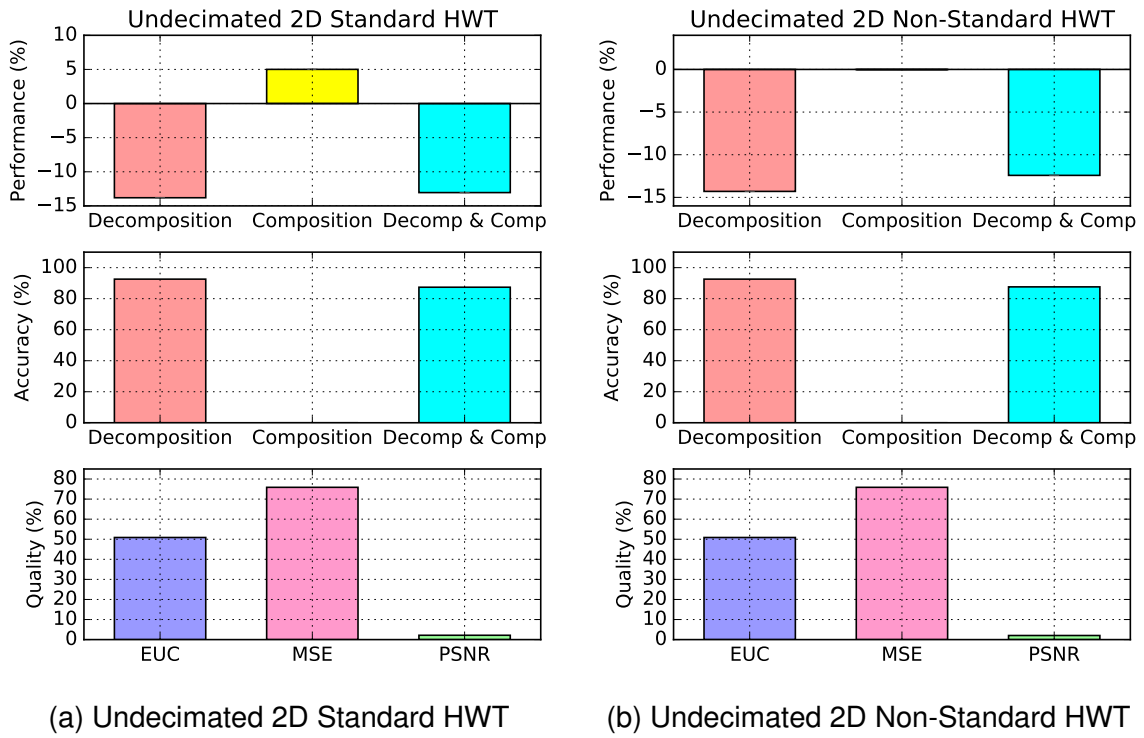(a) Undecimated 2D Standard HWT   (b) Undecimated 2D Non-Standard HWT

Figure 37: Times of execution and error measurement for the 2D HWT using 1024x1024 matrix with random values on 4 levels of decomposition.

The performance shown in Figure 37a and Figure 37b indicates that the developed algorithms are at least 10% slower than the originals, despite the composition step which has a little advantage of 5% on the standard method.

For the standard approach, the average execution time for decomposition, composition and the combination of both are $195.461ms$, $290.717ms$ and $200.601ms$, and their standard deviations $3.43775ms$, $10.5094ms$, $3.13332ms$.

For the non-standard approach execution time averages are $173.385ms$, $286.755ms$ and $192.677ms$, and their standard deviations are $3.03845ms$, $4.40987ms$ and $2.65613ms$.

Both approaches also share the same percentage of accuracy gain, 92.6% and

87.4% respectively for decomposition and the combination of both algorithms performing together. The EUC, MSE and $PSNR$ of both approaches are around 51%, 76% and 2% respectively.

5.2   Daubechies Wavelet Transform

In order to evaluate the 1D DaWT and its interval extension, three tests are executed using vectors filled with 1048576, 4194304 and 16777216 random values as input respectively. For tests using 2D DaWT algorithms, the inputs were generated from 1024x1024, 2048x2048 and 4096x4096 matrices of random values.

5.2.1   One Dimensional

The results shown in Figure 38 describe the performance of both decomposition and composition procedures from Section 3.2.1, targeting the one dimensional DaWT.
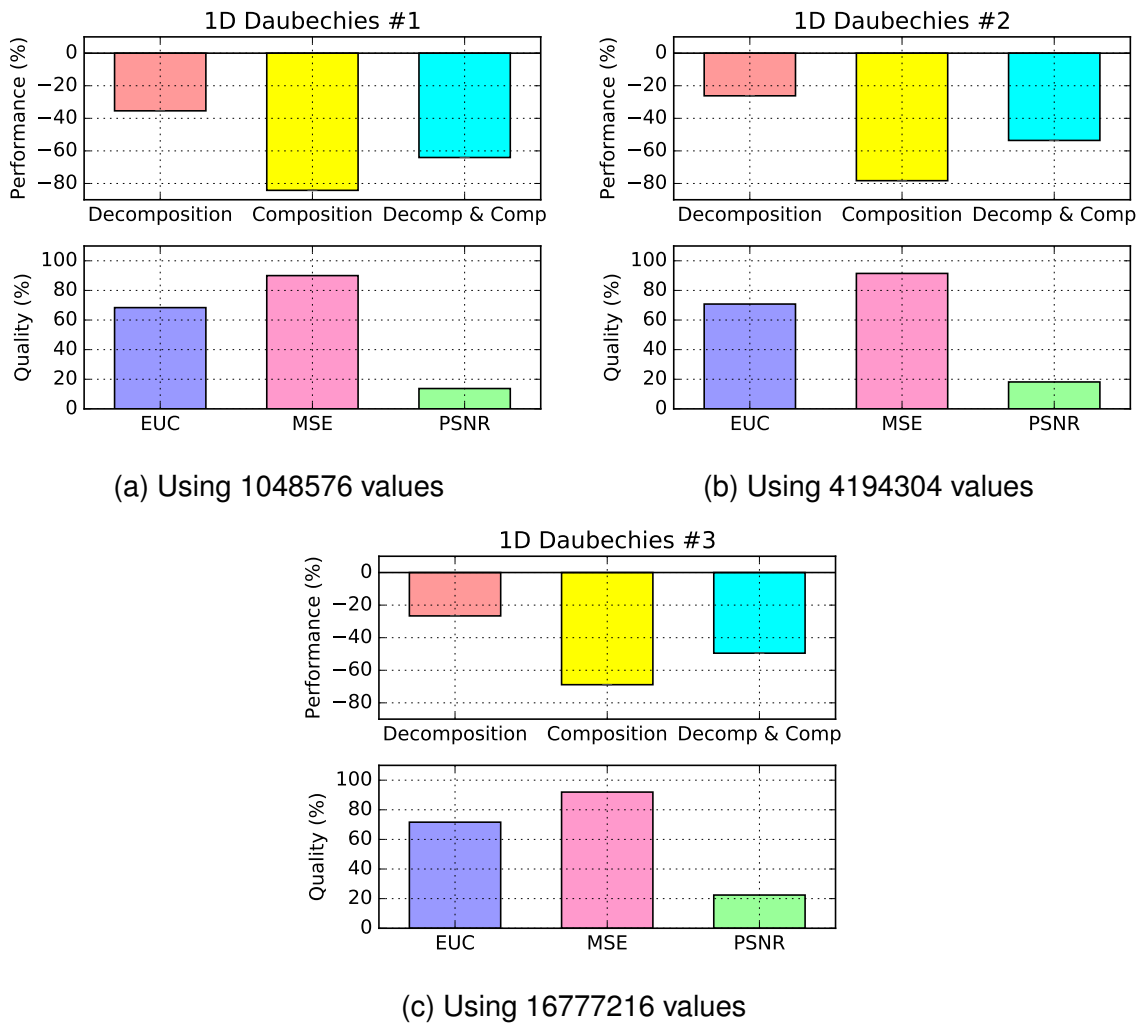

(a) Using 1048576 values

(b) Using 4194304 values

(c) Using 16777216 values

Figure 38: Performance and metric results for the 1D DaWT.

The data indicate that the developed algorithms are 35%, 26% and 26% slower than

the results based in (NIELSEN, 1998), when decomposing the vectors, and its coefficient of variation (CV) are 13.6%, 22.4% and 17.6% respectively. The average times of executions of the optimal algorithms for decomposition are $16.0975ms$, $69.0551ms$ and $283.37ms$, and the corresponding standard deviation are $0.219135ms$, $1.55045ms$ and $5.00761ms$.

The Composition tests indicates that the developed algorithms are 84%, 78% and 68% slower than the algorithms in the literature (NIELSEN, 1998), its CV are 5.7%, 5.3% and 9.8%. The average times of executions of the optimal algorithms for composition are $21.2527ms$, $88.8472ms$ and $363.605ms$, and the corresponding standard deviation are $0.121824ms$, $0.474684ms$ and $3.57715ms$.

The combination of both Decomposition and Composition operations are 64%, 53% and 49% and its CV are 7.5%, 9.4% and 9.7%. The average times of executions are $37.2334ms$, $157.169ms$ and $646.311ms$, and the corresponding standard deviation are $0.281221ms$, $1.47963ms$ and $6.30055ms$.

The interval extension shows that the results are 343x, 22x and 203629x less accurate, meaning that the developed algorithms generate much larger intervals compared to the original formulation of the DaWT, its due to the proposed filter presented in Section 3.2.1.

The results from EUC are 68.3%, 68.3% and 68.3%, data from MSE indicate gains of 89.9%, 91.4% and 91.9%, and $PSNR$ gains are 13.7%, 18.1% and 22.4%. The results from EUC, MSE and $PSNR$ show a very good gain over the original formulations of the DaWT.

5.2.2 Two Dimensional

This Section presents the results of both Standard and Non-Standard approaches of the DaWT in Sections 5.2.2.1 and 5.2.2.2 respectively.

5.2.2.1 Standard

The results shown in Figure 39 describe the performance of both decomposition and composition procedures from Section 3.2.2, targeting the two dimensional Standard DaWT.

The data indicate that the developed algorithms are 25%, 23% and 17% slower than the results based in (NIELSEN, 1998), when decomposing matrices, and its coefficient of variation (CV) are 7.5%, 10.5% and 2.2% respectively. The average times of executions are $43.1052ms$, $195.124ms$ and $1052.4ms$, and the corresponding standard deviation are $0.323927ms$, $2.03999ms$ and $2.34561ms$.

The Composition tests indicates that the developed algorithms are 63%, 56% and 39% slower than the algorithms in the literature (NIELSEN, 1998), its CV are 4%, 9.6% and 1.6%. The average times of executions are $55.1799ms$, $241.814ms$ and $1238.55ms$,

(a) Using 1024x1024 matrix

(b) Using 2048x2048 matrix
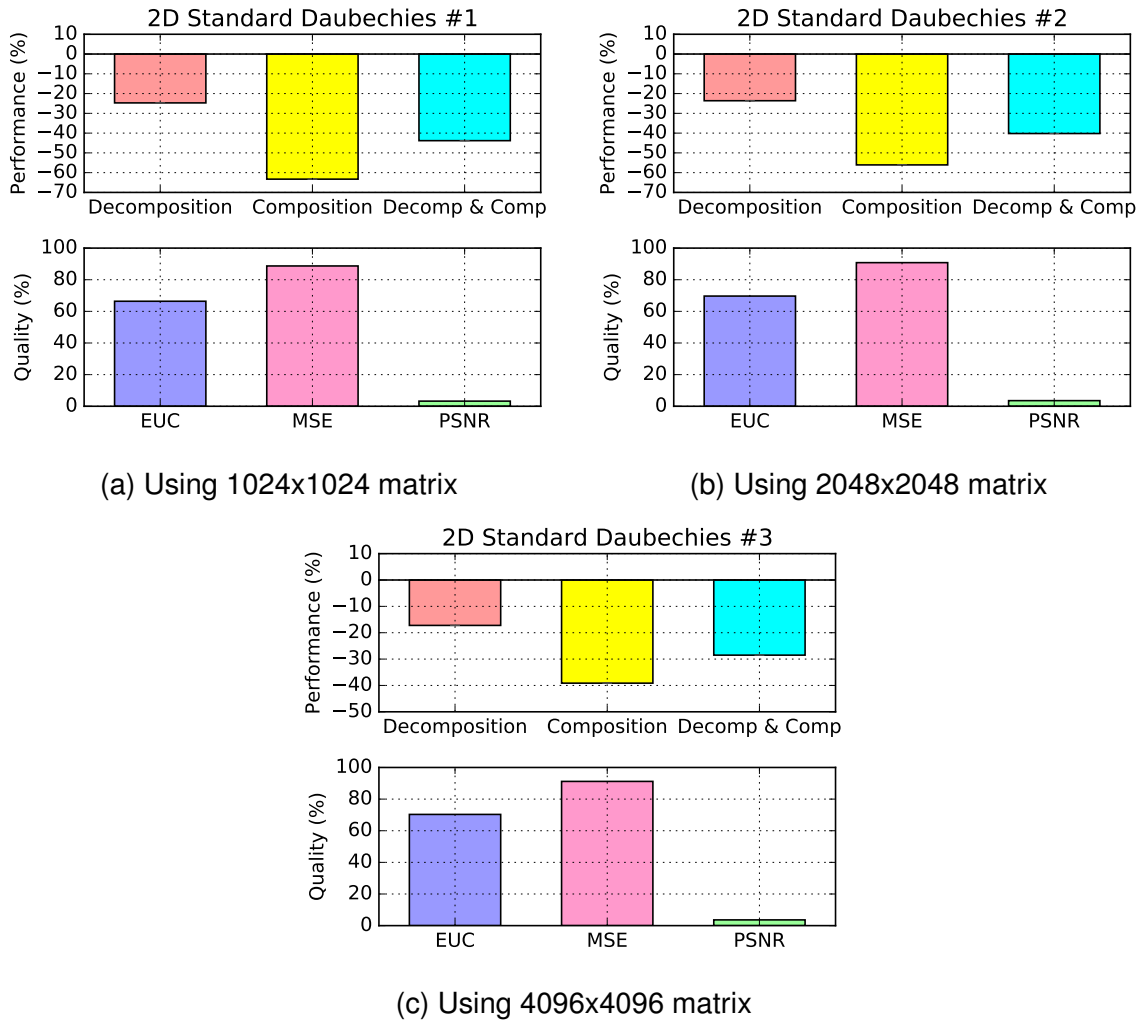
(c) Using 4096x4096 matrix

Figure 39: Performance and metric results for the 2D Standard DaWT.

and the corresponding standard deviation are $0.220524ms$, $2.32843ms$ and $2.01274ms$.

The combination of both Decomposition and Composition operations are 44%, 40% and 28% and its CV are 3.3%, 7.1% and 3.3%. The average times of executions are $98.1822ms$, $436.064ms$ and $2295ms$, and the corresponding standard deviation are $0.325826ms$, $3.11464ms$ and $7.64587ms$.

The interval extension shows that the results are $220x$, $18x$ and $80236x$ less accurate, meaning that the developed algorithms generate much larger intervals compared to the original formulation of the DaWT. The results from EUC are 66.4%, 70% and 70%, data from MSE indicate gains of 88.7%, 90.8% and 91.2%, and PSNR gains are 3.2%, 3.6% and 3.6%. The results from EUC, MSE and PSNR show a very good gain over the original formulations of the DaWT.

5.2.2.2   Non Standard

The results shown in Figure 40 summarize the performance of both decomposition and composition procedures from Section 3.2.2, targeting the two dimensional Standard DaWT.
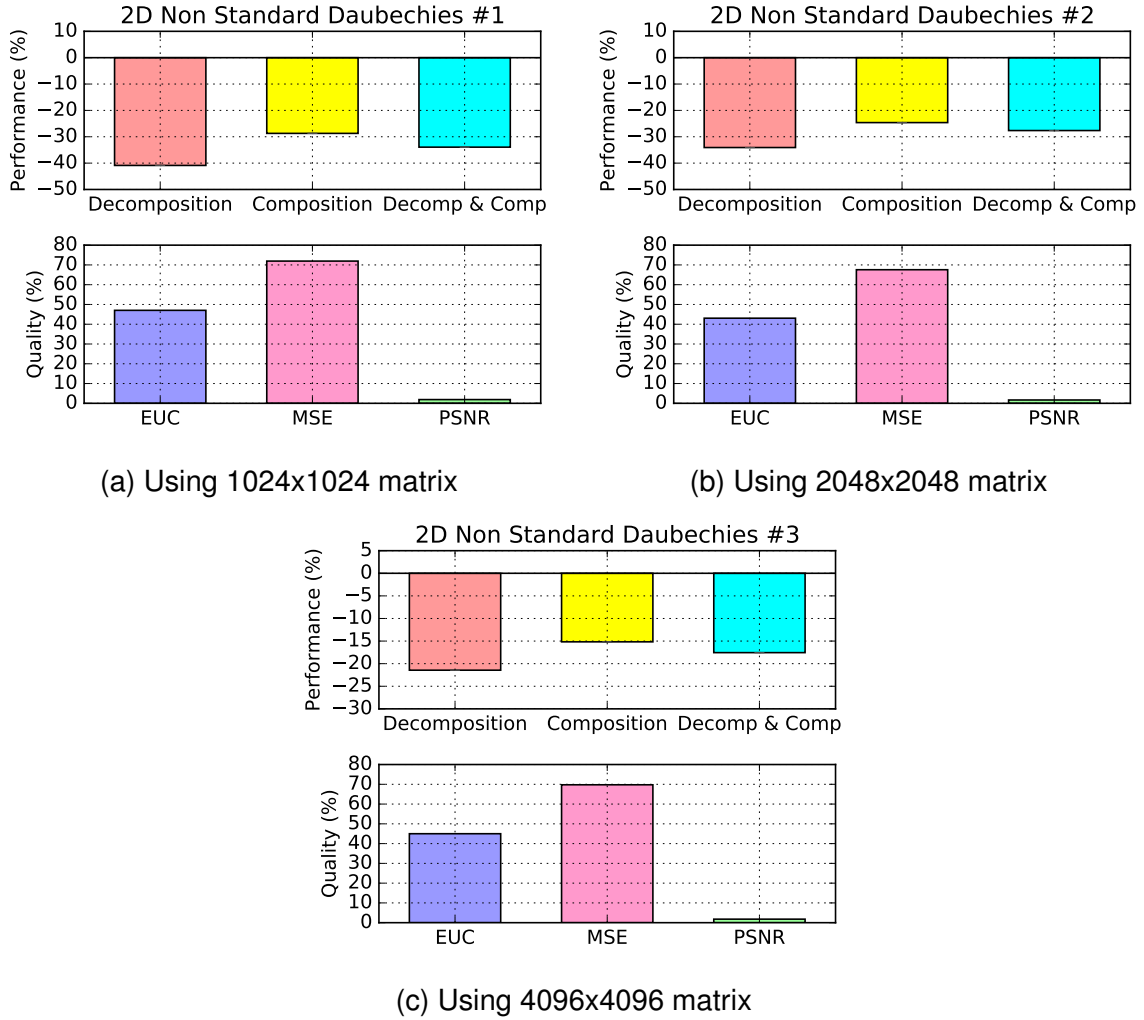

(a) Using 1024x1024 matrix

(b) Using 2048x2048 matrix

(c) Using 4096x4096 matrix

Figure 40: Performance and metric results for the 2D Non Standard DaWT.

The data indicate that the developed algorithms are 40%, 34% and 21% slower than the results based in (NIELSEN, 1998), when decomposing the matrices, and its coefficient of variation (CV) are 6.3%, 7.1% and 3% respectively. The average times of executions are $48.6368ms$, $221.135ms$ and $1277.12ms$, and the corresponding standard deviation are $0.306729ms$, $1.57093ms$ and $3.8811ms$.

The Composition tests indicate that the developed algorithms are 29%, 25% and 15% slower than the algorithms in the literature (NIELSEN, 1998), its CV are 4.7%, 15.6% and 2.3%. The average times of executions are $65.2928ms$, $302.765ms$ and $1791.56ms$, and the corresponding standard deviation are $0.310508ms$, $4.74117ms$ and $4.07062ms$.

The combination of both Decomposition and Composition operations are 34%, 28%

and 18% and their CV are 4%, 6.7% and 2.7%. The average times of executions are $113.875ms$, $522.279ms$ and $3067.84ms$, and the corresponding standard deviation are $0.4643ms$, $3.53883ms$ and $8.20209ms$.

The interval extension shows that the results are 3034x, 67x and 5916370x less accurate, meaning that the developed algorithms generate much larger intervals compared to the original formulation of the DaWT.

The results from EUC are 47%, 43.3% and 45%, data from MSE indicate gains of 71.9%, 67.6% and 70%, and $PSNR$ gains are 1.8%, 1.6% and 1.7%. The results from EUC, MSE and $PSNR$ show a very good gain over the original formulations of the DaWT.

6 CONCLUSION

This work presented an interval implementation of the Haar Wavelet Transform (HWT), previously published in (SANTOS et al., 2013a), for both decimated (SANTOS et al., 2013b) and undecimated (SANTOS et al., 2015) approaches and covering all study cases. The same study and optimization strategies were applied on the Daubechies Wavelet Transform (DaWT) with success.

The optimization for the decimated HWT made possible a gain of both accuracy of results and accuracy on performing the calculations, as it was expected. The same optimization was applied to the undecimated HWT, also know as À-Trous approach, and good results were obtained from the accuracy analysis, but there were performance loss due to the addition of operations on top of the original formation from the literature (STOLLNITZ; DEROSE; SALESIN, 1995).

A new set of filters was proposed by simplifying the DaWT original filters (NIELSEN, 1998) in order to increase accuracy of results. The algebraic manipulation of its filters made possible a gain in terms of accuracy of results in both 1D and 2D procedures. However, there was a need to add normalization steps on the top of the original DaWT formulation which resulted in a loss of performance.

Good results related to error analysis were obtained, that is, the optimizations improve accuracy. The metrics used in this work helped to evaluate the accuracy of all presented implementations.

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

Further research considers the study of other Discrete Wavelet Transforms (DWTs), together with the corresponding parallel and/or distribution of the Int-DWTs Library, using massive parallel architectural of GPUs and considering CUDA programming language for fast computation.

REFERENCES

ASADZADEH, M.; HASHEMI, E.; KOZAKEVICIUS., A. On efficiency of com- bined Daubechies wavelets and statistical parameters applied in mammography. **Applied and Computational Mathematics**, [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. Orthonormal bases of compactly supported wavelets. **Communications on Pure and Applied Mathematics**, [S.l.], v.41, n.7, p.909–996, 1988.

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.613?627, 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 Inf. 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. (Numerical Validation in Current Hardware Architectures).

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: INTL. 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. **Intl. 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. **Intl. Journal of Eng. Research and Appls. (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 PERF. COMP. FOR COMP. SCIENCE, 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.]: Lockheed Aircraft Corporation, 1959. (LMSD-48421).

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, Stanford.

NIELSEN, O. M. **Wavelets in scientific computing**. Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby: [s.n.], 1998. Supervisor: Per Christian Hansen, pcha@dtu.dk, DTU Compute.

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.

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.

SANTOS, V. dos; PILLA, M.; REISER, R.; KOZAKEVICIUS, A. Int-Haar: Improving Precision of the Haar Interval Wavelet Extension. In: THEORETICAL COMPUTER SCIENCE, WORKSHOP-SCHOOL ON, WEIT 2013, 2013, Rio Grande, Brazil. **Anais. . .** IEEE Computer Society, 2013. p.92–96.

SANTOS, V. dos; PILLA, M.; REISER, R.; KOZAKEVICIUS, A. Int-HWT: Increasing Performance and Exactitude of 1D and 2D Haar Wavelet Transforms. **Simpósui Sul de Microeletrônica**, Santa Maria, Brazil, 2015.

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.; FADILI, J.; MURTAGH, F. The Undecimated Wavelet Decomposition and its Reconstruction. In: IEEE TRANSACTIONS ON IMAGE PROCESSING, 2007. **Anais. . .** [S.l.: s.n.], 2007. p.297–309.

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. **Advancing Science through Computation**, [S.l.], 1996.