Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080615

A Mathematical Framework for Parallel Computing of Discrete-Time Discrete-Frequency Transforms in Multi-Core Processors

Pablo Soto-Quiros[*]

Escuela de Matemáticas, Instituto Tecnológico de Costa Rica, Apdo. 159-7050, Cartago, Costa Rica

Received: 16 Oct. 2013, Revised: 14 Jan. 2014, Accepted: 15 Jan. 2014
Published online: 1 Nov. 2014

Abstract: This paper presents a mathematical framework for a family of discrete-time discrete-frequency transforms in terms of matrix signal algebra. The matrix signal algebra is a mathematics environment composed of a signal space, a finite dimensional linear operators and special matrices where algebraic methods are used to generate these signal transforms as computational estimators. The matrix signal algebra contribute to analysis, design and implementation of parallel algorithms in multi-core proccesors. In this work, an implementation and experimental investigation of the mathematical framework are performed using MATLAB® with the Parallel Computing Toolbox™. We found that there is advantage to use multi-core processors and a parallel computing environment to minimize the high execution time. Also, speedup and efficiency increases when the number of logical processor and length of the signal increase. Moreover, a superlinear speedup is obtained in this experimental investigation.

Keywords: DFT, matrix signal algebra, superlinear speedup

1 Introduction

In signal processing, an important aspect of the study of a signal is understanding how its frequency varies with time [1,2]. The time-frequency analysis was developed to aid get this information using time-frequency representations of a signal, through of time-frequency transforms [2,3]. Time-frequency transforms can represent a signals over a time-frequency plane. These transforms combine time-domain and frequency-domain analyses to yield a picture of the temporal localization of a signals spectral components. They may also serve for signal synthesis, coding and processing [1,3].

A computational implementation of time-frequency transforms is performed using discrete periodic signals and discrete-time discrete-frequency (DT-DF) transforms. A signal is a discrete periodic signal if it completes a pattern within a measurable time frame, called a period and repeats that pattern over identical subsequent periods. Examples of DT-DF transforms are the discrete ambiguity function (DAF) [4], the discrete short-time Fourier transforms (DSTFT) [5], the discrete Zak transform (DZT) [6], the discrete chirp-Fourier transform (DCFT) [7], the modified discrete chirp-Fourier transform (MDCFT) [8] and the new discrete chirp-Fourier transform (NDCFT) [9]. These transforms have several applications in engineering: waveform designs [10], time-frequency representations of audio [5], Gabor expansions and Weyl-Heisenberg frames [11], radar signal processing as estimator of range and velocity parameters of the moving object [2,12] and synthetic aperture radar (SAR) and inverse SAR imaging [8]. Many implementations of these DT-DF transforms have been studied and developed in [2,6,7,8,9,12,13], but very few developed a parallel computing (see, e.g., [13,14]).

In this paper, we present a new general mathematical framework for all DT-DF transforms mentioned above (DAF, DSTFT, DZT, DCFT, MDCFT, NDCFT). This mathematical framework is different to others implementations because express each DT-DF transform in terms of a matrix signal algebra, which is a mathematics environment composed of a signal space, finite dimensional linear operators and special matrices, where algebraic methods are used to generate these signal transforms as computational estimators [12]. This matrix signal algebra contributes to analysis, design and

[*] Corresponding author e-mail: jusoto@itcr.ac.cr

implementation of parallel algorithms. Thus, an implementation and experimental investigation of this mathematical framework are performed using MATLAB® with the Parallel Computing Toolbox™ in a computer with multi-core proccesors.

The present paper is organized as follows. In Section 2, we define the matrix signal algebra and we explain some applications to parallel computing. In Section 3, we explain the different types of DT-DF transforms to use in this paper. Furthermore, we develop a mathematical framework of DT-DF transforms in terms of the matrix signal algebra. In Section 4, we explain an implementation and experimental investigation of this mathematical framework using parallel computing in multi-core processors with MATLAB®. Finally, in Section 5, we present some conclusions.

Throughout the paper, the following notation is used. $\mathbb{Z}_N = \{0,1,...,N-1\}$ is the additive group $\mathbb{Z}$ of integers modulo N, $\mathbb{C}^{M\times N}$ is the matrix space of M rows and N columns with complex numbers entries and $\mathbb{C}^N = \mathbb{C}^{N\times 1}$. The rows and columns of $\boldsymbol{A} \in \mathbb{C}^{M\times N}$ are indexed by elements of $\mathbb{Z}_M$ and $\mathbb{Z}_N$, respectively. $\boldsymbol{A}(m,n)$, $\boldsymbol{A}(m,:)$, $\boldsymbol{A}(:,n)$, $\overline{\boldsymbol{A}}$ and $\boldsymbol{A}^T$ represent entry (m,n), row m, column n, conjugate matrix and transpose matrix of $\boldsymbol{A}$, respectively. $\boldsymbol{I}_N \in \mathbb{C}^{N\times N}$ and $\boldsymbol{1}_N \in \mathbb{C}^N$ are identity matrix and ones vector, respectively.

2 Matrix Signal Algebra

We define the matrix signal algebra as a mathematics environment composed of a signal space, finite dimensional linear operators and special matrices where algebraic methods are used to generate algorithms in signal processing area.

Let $\boldsymbol{A},\boldsymbol{B} \in \mathbb{C}^{M\times N}$, $\boldsymbol{C} \in \mathbb{C}^{P\times Q}$ and $\{\boldsymbol{A}_n\}_{n\in\mathbb{Z}_N}$ such that $\boldsymbol{A}_n \in \mathbb{C}^{M_n\times P}$. Some spaces, operators and matrices associated to the matrix signal algebra are the following:

- The space of discrete periodics signals, $l^2(\mathbb{Z}_N)$, is the set of $\mathbb{C}$-valued signals on $\mathbb{Z}_N$. Moreover, $\boldsymbol{x} \in l^2(\mathbb{Z}_N)$ if and only if $\boldsymbol{x} \in \mathbb{C}^N$ [15]. This space corresponds to signals with finite energy and N-periodic sequences, i.e., for each $k_1 \in \mathbb{Z}$, $\boldsymbol{x}(k_1) = \boldsymbol{x}(k_2)$, where $k_2 \in \mathbb{Z}_N$ and $k_1 \equiv k_2 \bmod N$.
- The Hadamard product of $\boldsymbol{A}$ and $\boldsymbol{B}$ is defined as $\boldsymbol{A}\odot\boldsymbol{B} \in \mathbb{C}^{M\times N}$ such that

$$(\boldsymbol{A}\odot\boldsymbol{B})(m,n) = \boldsymbol{A}(m,n)\cdot\boldsymbol{B}(m,n).$$

  The Hadamard product is also known as pointwise or coordinatewise product.
- The Kronecker product of $\boldsymbol{A}$ and $\boldsymbol{C}$ is defined as $\boldsymbol{A}\otimes\boldsymbol{C} \in \mathbb{C}^{MP\times NQ}$ such that

$$\boldsymbol{A}\otimes\boldsymbol{C} = \begin{pmatrix} \boldsymbol{A}(0,0)\boldsymbol{C} & \cdots & \boldsymbol{A}(0,N-1)\boldsymbol{C} \\ \vdots & \ddots & \vdots \\ \boldsymbol{A}(M-1,0)\boldsymbol{C} & \cdots & \boldsymbol{A}(M-1,N-1)\boldsymbol{C} \end{pmatrix}.$$

  It replaces every entry (m,n) of $\boldsymbol{A}$ by the matrix $\boldsymbol{A}(m,n)\boldsymbol{C}$. In the special case $\boldsymbol{A} = \boldsymbol{I}_N$, it is called parallel operation [16].
- Let $N = RS$. The stride permutation matrix is defined as $\boldsymbol{L}_S^N \in \mathbb{C}^{N\times N}$ such that it permutes the elements of the input signal $\boldsymbol{x} \in \mathbb{C}^N$ as $mR+n \to nS+m$, $m \in \mathbb{Z}_S$ and $n \in \mathbb{Z}_R$ [16,17]. This matrix permutation governs the data flow required to parallelize a Kronecker product computation [16].
- The vec operator, $\mathscr{V} : \mathbb{C}^{M\times N} \to \mathbb{C}^{MN}$, transforms a matrix into a vector, by stacking all the columns of this matrix one underneath the other. On the other hand, the vec inverse operator, $\mathscr{R}_{M,N} : \mathbb{C}^{MN} \to \mathbb{C}^{M\times N}$, transforms a vector of dimension MN into a matrix of size $M\times N$. $\mathscr{R}_{N,N}$ is related to the stride permutation matrix: $\mathscr{R}_{N,N}\{\boldsymbol{L}_N^{N^2}\boldsymbol{v}\} = (\mathscr{R}_{N,N}\{\boldsymbol{v}\})^T$, for $\boldsymbol{v} \in \mathbb{C}^{N^2}$.
- The accumulation operator of matrices, $\bigsqcup : \prod_{n\in\mathbb{Z}_N} \mathbb{C}^{M_n\times P} \to \mathbb{C}^{M\times P}$ with $M = \sum_{n\in\mathbb{Z}_N} M_n$, is defined as

$$\bigsqcup_{n\in\mathbb{Z}_N} \boldsymbol{A}_n = \begin{pmatrix} \boldsymbol{A}_0 \\ \vdots \\ \boldsymbol{A}_{N-1} \end{pmatrix}.$$

The following examples illustrate how the matrix signal algebra contributes to analysis, design and implementation of parallel algorithms.

Example 2.1. Let $\boldsymbol{A} \in \mathbb{C}^{R\times M}$, $\boldsymbol{x} \in \mathbb{C}^{RN}$ and $\boldsymbol{y} \in \mathbb{C}^{MN}$. We consider the matrix operation $\boldsymbol{x}\odot(\boldsymbol{I}_N\otimes\boldsymbol{A})\boldsymbol{y}$. This matrix operation can be decomposed as follows:

$$\begin{pmatrix} \boldsymbol{x}_0 \\ \vdots \\ \boldsymbol{x}_{N-1} \end{pmatrix} \odot \begin{pmatrix} \boldsymbol{A} & & \\ & \ddots & \\ & & \boldsymbol{A} \end{pmatrix} \begin{pmatrix} \boldsymbol{y}_0 \\ \vdots \\ \boldsymbol{y}_{N-1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_0 \odot \boldsymbol{A}\boldsymbol{y}_0 \\ \vdots \\ \boldsymbol{x}_{N-1} \odot \boldsymbol{A}\boldsymbol{y}_{N-1} \end{pmatrix},$$

where $\boldsymbol{x}_m \in \mathbb{C}^R$ and $\boldsymbol{y}_m \in \mathbb{C}^M$. The matrix operation $\boldsymbol{x}\odot(\boldsymbol{I}_N\otimes\boldsymbol{A})\boldsymbol{y}$ can be divided into N sub-operations $\boldsymbol{x}_m \odot \boldsymbol{A}\boldsymbol{y}_m$, for $m \in \mathbb{Z}_N$. The structure operation of $\boldsymbol{x}\odot(\boldsymbol{I}_N\otimes\boldsymbol{A})\boldsymbol{y}$ allows an implementation using parallel computing, because each $\boldsymbol{x}_m \odot \boldsymbol{A}\boldsymbol{y}_m$ is computed independently. ■

Example 2.2. Matrix signal algebra is using to compute the discrete Fourier Transform (DFT) [16,18,19]. The DFT of $\boldsymbol{x} \in l^2(\mathbb{Z}_N)$ is represented as $\mathscr{F}_{\boldsymbol{x}} : \mathbb{Z}_N \to \mathbb{C}$ such that $\mathscr{F}_{\boldsymbol{x}}(k) = \frac{1}{\sqrt{N}}\sum_{n\in\mathbb{Z}_N} \boldsymbol{x}(n)\omega_N^{-nk}$, where $\omega_N = e^{2\pi i/N}$. The matrix representation of DFT of $\boldsymbol{x}$ is $\mathscr{F}_{\boldsymbol{x}} = \frac{1}{\sqrt{N}}\boldsymbol{F}_N\boldsymbol{x}$, where $\boldsymbol{F}_N \in \mathbb{C}^{N\times N}$ such that $\boldsymbol{F}_N(m,n) = \omega_N^{-mn}$. If $N = RS$, then the matrix formalism can be used to express $\boldsymbol{F}_N$ as factorizations of matrices using operators and matrices from matrix signal algebra [16,18]:

$$\boldsymbol{F}_N = \tfrac{1}{\sqrt{N}}\,\boldsymbol{L}_S^N(\boldsymbol{I}_R\otimes\boldsymbol{F}_S)\boldsymbol{L}_R^N\,\boldsymbol{T}_R^N(\boldsymbol{I}_S\otimes\boldsymbol{F}_R)\boldsymbol{L}_S^N.$$

Here, $\boldsymbol{T}_R^N$ is a diagonal matrix containing the twiddle factors. This factorization of $\boldsymbol{F}_N$ is the recursive general-radix decimation in time Cooley-Tukey FFT for $N = RS$. In addition, this representation of $\boldsymbol{F}_N$ allows the implementation using parallel computing [17]. ■

Table 1: Values of $\boldsymbol{A}$ and $\boldsymbol{H}$ from DT-DF transforms

DT-DF Transforms	$\boldsymbol{A}(m,k)$	$\boldsymbol{H}(m,n)$
DAF	1	$\bar{\boldsymbol{y}}(n+m)$
DSTFT	1	$\boldsymbol{w}(n-m)$
DZT	ω_N^{mk}	1
DCFT	1	$\omega_N^{-mn^2}$
MDCFT	1	$(1+(-1)^n)\,\omega_N^{-mn^2/2}$
NDCFT	$(-1)^k$	$\omega_N^{-m(n-N/2)^2/2}$

$\omega_N = e^{2\pi i/N}$ is a root of unity.
$\boldsymbol{y} \in l^2(\mathbb{Z}_N)$ is a discrete periodic echo signal.
$\boldsymbol{w} \in l^2(\mathbb{Z}_N)$ is a discrete periodic window function.

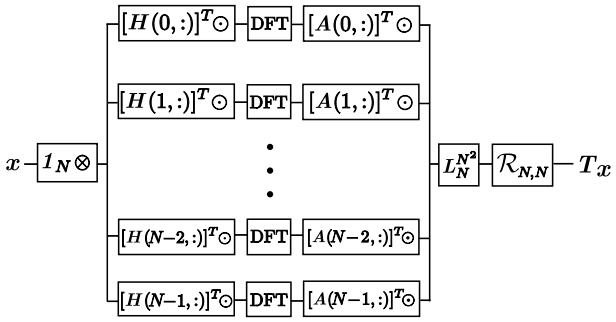

Fig. 1: Parallel model of DT-DF transforms for a signal $\boldsymbol{x} \in l^2(\mathbb{Z}_N)$ using the matrix signal algebra.

3 Discrete-Time Discrete-Frequency Transforms

3.1 Definition

In signal processing, time-frequency analysis is a body of techniques and methods used for characterizing and manipulating signals whose statistics vary in time, such as transient signals. For discrete periodics signals, we use discrete-time discrete-frequency (DT-DF) transforms. Each signal $\boldsymbol{x} \in l^2(\mathbb{Z}_N)$ can be express in two dimension using a DT-DF transform $\mathscr{T}$, such that

$$\begin{array}{rcl} \mathscr{T} : \{l^2(\mathbb{Z}_N), \mathbb{Z}_N \times \mathbb{Z}_N\} \to & \mathbb{C} \\ \{\boldsymbol{x}, (m,k)\} & \mapsto \mathscr{T}(m,k) \end{array}$$

where $m,k \in \mathbb{Z}_N$. In this paper, a DT-DF transform of $\boldsymbol{x}$ is expressed as

$$\begin{array}{rcl} \mathscr{T}_{\boldsymbol{x}} : \mathbb{Z}_N \times \mathbb{Z}_N \to & \mathbb{C} \\ (m,k) & \mapsto \mathscr{T}_{\boldsymbol{x}}(m,k) \end{array}$$

As already mentioned above, the discrete ambiguity function (DAF), the discrete short-time Fourier transforms (DSTFT), the discrete Zak transform (DZT), the discrete chirp-Fourier transform (DCFT), the modified discrete chirp-Fourier transform (MDCFT) and the new discrete chirp-Fourier transform (NDCFT) are some types of DT-DF transforms. These transforms have the same structure:

$$\mathscr{T}_{\boldsymbol{x}}(m,k) = \frac{1}{\sqrt{N}}\,\boldsymbol{A}(m,k) \sum_{n \in \mathbb{Z}_N} \boldsymbol{x}(n)\boldsymbol{H}(m,n)\omega_N^{-nk}, \qquad (1)$$

where $\boldsymbol{x} \in l^2(\mathbb{Z}_N)$ and $\boldsymbol{A}, \boldsymbol{H} \in \mathbb{C}^{N\times N}$ are given in Table 1. For DAF, $\boldsymbol{y} \in l^2(\mathbb{Z}_N)$ is a discrete periodic echo signal [4] and for DSTFT, $\boldsymbol{w} \in l^2(\mathbb{Z}_N)$ is a discrete periodic window function [20].

3.2 Mathematical Framework

There are two fundamentally different ways of representing the DT-DF transforms: as summations, explained above, or matrix form. Both representations of DT-DF transform allow to develop a fast algorithm, but the matrix representation permits an implementation using parallel computing.

Let $\boldsymbol{T}_{\boldsymbol{x}} \in \mathbb{C}^{N\times N}$ the matrix representation of DT-DF transforms, such that $\boldsymbol{T}_{\boldsymbol{x}}(m,k) = \mathscr{T}_{\boldsymbol{x}}(m,k)$. The following result represents $\boldsymbol{T}_{\boldsymbol{x}}$ in terms of matrix signal algebra.

Theorem 3.2.1. Let $\boldsymbol{x} \in l^2(\mathbb{Z}_N)$. Then

$$\boldsymbol{T}_{\boldsymbol{x}} = \tfrac{1}{\sqrt{N}}\,\boldsymbol{A} \odot \mathscr{R}_{N,N}\left\{\boldsymbol{L}_N^{N^2}\,(\boldsymbol{I}_N \otimes \boldsymbol{F}_N)\,(\boldsymbol{h} \odot (\boldsymbol{1}_N \otimes \boldsymbol{x}))\right\}, \qquad (2)$$

where $\boldsymbol{h} \in \mathbb{C}^{N^2}$ such that $\boldsymbol{h} = \bigsqcup_{m\in\mathbb{Z}_N} [\boldsymbol{H}(m,:)]^T$.

Proof. Let $\boldsymbol{z} = \boldsymbol{L}_N^{N^2}\,(\boldsymbol{I}_N \otimes \boldsymbol{F}_N)\,(\boldsymbol{h} \odot (\boldsymbol{1}_N \otimes \boldsymbol{x}))$. This vector can be expressed as

$$\boldsymbol{z} = \boldsymbol{L}_N^{N^2} \bigsqcup_{m\in\mathbb{Z}_N} \boldsymbol{s}_m, \qquad (3)$$

where $\boldsymbol{s}_m \in \mathbb{C}^N$, such that $\boldsymbol{s}_m = \boldsymbol{F}_N\left([\boldsymbol{H}(m,:)]^T \odot \boldsymbol{x}\right)$. Applying the $\mathscr{R}_{N,N}$ operator in (3), we obtain

$$\begin{aligned} \mathscr{R}_{N,N}\{\boldsymbol{z}\} &= \mathscr{R}_{N,N}\left\{\boldsymbol{L}_N^{N^2} \bigsqcup_{m\in\mathbb{Z}_N} \boldsymbol{s}_m\right\} \\ &= \left(\mathscr{R}_{N,N}\left\{\bigsqcup_{m\in\mathbb{Z}_N} \boldsymbol{s}_m\right\}\right)^T. \end{aligned}$$

Let $\boldsymbol{S} \in \mathbb{C}^{N\times N}$ such that $\boldsymbol{S} = \mathscr{R}_{N,N}\left\{\bigsqcup_{m\in\mathbb{Z}_N} \boldsymbol{s}_m\right\}$. Then

$$\begin{aligned} \mathscr{R}_{N,N}\{\boldsymbol{z}\}(m,k) &= \boldsymbol{S}^T(m,k) \\ &= \boldsymbol{s}_m(k) \\ &= \sum_{n\in\mathbb{Z}_N} \boldsymbol{x}(n)\boldsymbol{H}(m,k)\omega_N^{-nk}. \end{aligned}$$

Finally, if we make the Hadamard product of $\boldsymbol{A}$ and $\boldsymbol{S}^T$, then we obtain

$$\begin{aligned} \tfrac{1}{\sqrt{N}}(\boldsymbol{A} \odot \boldsymbol{S}^T)(m,k) &= \tfrac{1}{\sqrt{N}}\,\boldsymbol{A}(m,k) \cdot \boldsymbol{S}^T(m,k) \\ &= \tfrac{1}{\sqrt{N}}\,\boldsymbol{A}(m,k) \cdot \boldsymbol{s}_m(k) \\ &= \boldsymbol{T}_{\boldsymbol{x}}(m,k) \end{aligned}$$

□

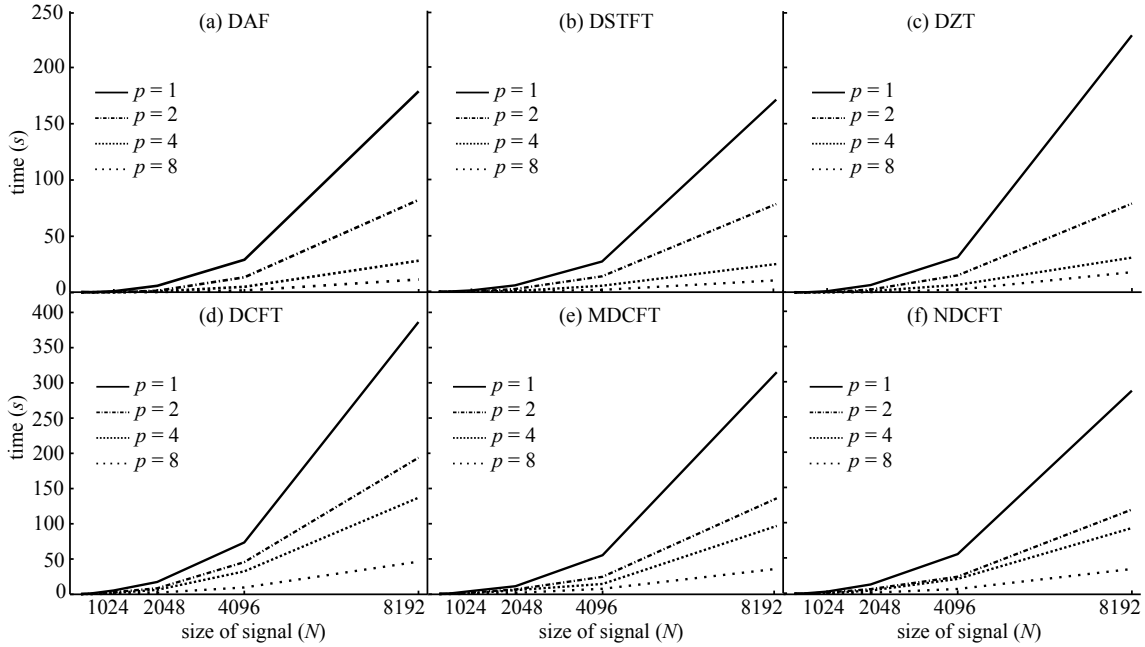

Fig. 2: Computing time of the $N \times N$ DT-DF Transforms.

Fig. 1 shows a model of DT-DF transforms using the matrix signal algebra. We can observe N independent processes, making this approach a parallel operation. Now, using the property $\mathscr{R}_{N,N}\{\boldsymbol{L}_N^{N^2}\boldsymbol{v}\} = (\mathscr{R}_{N,N}\{\boldsymbol{v}\})^T$, equation (2) can write as

$$\boldsymbol{T_x} = \tfrac{1}{\sqrt{N}}\,\boldsymbol{A} \odot \mathscr{R}_{N,N}\left\{(\boldsymbol{I}_N \otimes \boldsymbol{F}_N)\,(\boldsymbol{h} \odot (\boldsymbol{1}_N \otimes \boldsymbol{x}))\right\}^T. \quad (4)$$

The computational complexity of $\mathscr{R}_{N,N}\{\boldsymbol{L}_N^{N^2}\boldsymbol{v}\}$ and $(\mathscr{R}_{N,N}\{\boldsymbol{v}\})^T$ can be implemented linearly; thus, the equations (2) and (4) are computationally similar.

The following algorithm shows the implementation of equation (4).

Algorithm 1: DT-DF Transform Algorithm

Require: $\boldsymbol{x} \in \mathbb{C}^N$
Ensure: $\boldsymbol{T}_x \in \mathbb{C}^{N\times N}$

1. **for** $m \leftarrow 0 : N-1$
2. $\quad \boldsymbol{h} \leftarrow [\boldsymbol{H}(m,:)]^T$
3. $\quad \boldsymbol{v}_1 \leftarrow \boldsymbol{x} \odot \boldsymbol{h}$
4. $\quad \boldsymbol{v}_2 \leftarrow \frac{1}{\sqrt{N}}\boldsymbol{F}_N\boldsymbol{v}_1$
5. $\quad \boldsymbol{T}_x(:,m) \leftarrow [\boldsymbol{A}(m,:)]^T \odot \boldsymbol{v}_2$
6. **end for**
7. $\boldsymbol{T}_x \leftarrow (\boldsymbol{T}_x)^T$

Steps 2-5 are independent in each iteration, therefore the above algorithm allows parallel computation. Also, in the case $\boldsymbol{A}(m,n) = 1$, for all $m,n \in \mathbb{Z}_N$, the Hadamard product of Step 5 can be omitted.

4 Implementation and Experimental Investigation

4.1 General Information

The investigations have been carried out on multi-core processors computer of Instituto Tecnológico de Costa Rica (Costa Rica Institute Technology). The computer consists of 4 two-processor units (8 logical processors) with Intel® Core™ i7-3632QM CPU processor, system clock of 2.20 GHz and 8 GB of RAM.

In this experiment, we do the implementation and testing of Algorithm 1 for all DT-DF transforms defined above. We use a chirp signal $\boldsymbol{x} \in l^2(\mathbb{Z}_N)$ such that $\boldsymbol{x}(n) = \omega^{-25n^2-30n} + \omega^{-5n^2-63n}$ as experimental signal. We select a chirp signal because the time-frequency plane is a natural representation space for chirps signals and, therefore it is a signal frequently used in DT-DF transforms [21]. For the DAF, we use the same chirp signal $\boldsymbol{x}$ as echo signal[1] and, for the DSTFT, we use a discrete Hamming signal $\boldsymbol{w} \in l^2(\mathbb{Z}_N)$ as the discrete window function, where it is defined as $\boldsymbol{w}(n) = 0.54 - 0.46\cos(2\pi n/(N-1))$.

The implementation of Algorithm 1 to compute each DT-DF transform is performed using MATLAB®. MATLAB® provides two main ways to take advantage of multicore and multiprocessor computers: built-in multithreading and parallelism using MATLAB® workers

[1] If discrete echo signal is the same signal $\boldsymbol{x}$, then DAF is called discrete cross-ambiguity function [2].

Table 2: Speedup of Algorithm 1

DT-DF Transform	p	$N=256$	$N=512$	$N=1024$	$N=2048$	$N=4096$	$N=8192$
DAF	2	2.267	1.255	2.878	3.751	2.172	2.141
	4	1.921	1.206	2.592	5.335	6.318	5.661
	8	2.593	1.660	4.291	11.146	15.140	15.288
DSTFT	2	0.975	2.109	3.543	2.177	1.935	2.186
	4	0.921	2.191	2.685	6.457	4.844	6.854
	8	1.243	1.880	4.569	9.527	13.702	16.317
DZT	2	2.443	1.445	2.213	2.402	2.047	2.885
	4	2.035	1.257	2.644	4.571	4.497	7.318
	8	2.670	1.779	4.701	10.555	11.269	12.465
DCFT	2	1.670	1.692	1.780	2.072	1.617	1.993
	4	2.102	2.895	3.123	2.805	2.584	2.820
	8	1.867	4.432	8.263	7.761	7.599	8.355
MDCFT	2	2.009	1.680	1.895	1.653	2.238	2.315
	4	1.996	2.181	3.266	3.017	3.789	3.251
	8	2.853	3.967	8.173	6.178	7.120	8.764
NDCFT	2	2.325	1.678	1.581	1.968	2.306	2.408
	4	1.998	3.200	2.638	2.811	2.673	3.080
	8	2.899	3.649	6.480	8.017	7.810	8.136

N is length of the signal and p is the number of logical processor.

[22,23]. We use parallelism using MATLAB® workers. We can run multiple MATLAB® workers (MATLAB® computational engines) on a multi-core computer to execute applications in parallel, with the Parallel Computing Toolbox™. This approach allows more control over the parallelism than with built-in multithreading [22]. With programming constructs such as parallel for-loops (`parfor`) and batch, we write the parallel MATLAB programs of the mathematical framework for DT-DF transforms.

4.2 Results and Discussion

The computational performance analysis of Algorithm 1 is evaluated using the metrics speedup (or acceleration) and efficiency. Let T_1 the execution time of the sequential algorithm and T_p the execution time of the parallel algorithm, where p is the number of logical processors. The speedup is the ratio between the execution times of sequencial and parallel implementations, and it is a value typically between 1 and p. It is represented by the formula $S = T_1/T_p$. The efficiency is determined by the ratio between the speedup and the number of processing elements, and it is a value typically between 0 and 1. It is represented by the formula $E = T_1/(pT_p)$. When $S > p$ and $E > 1$, it is called superlinear speedup.

Fig. 2 shows the execution time T_p, in seconds s, of Algorithm 1 as a function of N, where N is the size of signal of each DT-DF transform. In this figure, it is observed that there is significant reduction in the parallel execution time of each DT-DF transform. For example, to compute DAF with a chirp signal of size $N = 8192$ produce a time of serial execution $T_1 = 178.845$ s. But, using parallel computing, we obtain $T_2 = 82.339$ s (43.04% of T_1), $T_4 = 28.310$ s (15.82% of T_1) and $T_8 = 11.700$ s (6.54% of T_1). This shows the advantage of to use multi-core processors and a parallel computing environment to minimize the high execution time in each DT-DF transform. This is due because parallel computing is a form of computation in which many calculations are carried out simultaneously [24,25], operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently, and minimize execution time [25,26].

Tables 2 and 3 represent speedup and efficiency of Algorithm 1 obtained from the experimental chirp signal with each DT-DF transform. In Table 2, it is observed that the acceleration of most DT-DF transforms increases when p increases, regardless of the value of N. Moreover, we obtain superlinear speedup in about 42% of simulations and most of it is obtained when N increases. It indicate that speedup increases and superlinear speedup is obtained when p and N increase, using Algorithm 1 and MATLAB® with the Parallel Computing Toolbox™ in a computer with similar characteristic to those used in this paper. Superlinear speedup is not common in parallel computing. A few researches obtain a superlinear speedup in its parallel implementation (see, e.g., [27,28]). Some research mentioned various reasons for superlinear speedup: cache effect resulting from the different memory hierarchies of a modern compute [27], the termination time can be reduced when several searches are executed at the same time or the efficient utilization of resources by multiprocessors [29].

Now, Table 3 shows increasing values of efficiency with the increase of p of most DT-DF transforms. For all DT-DF transforms, we obtain an efficiency above of 23% in the range $256 \leq N \leq 1024$ and an efficiency above of

Table 3: Efficiency of Algorithm 1

DT-DF Transform	p	$N=256$	$N=512$	$N=1024$	$N=2048$	$N=4096$	$N=8192$
DAF	2	1.333	0.628	1.439	1.876	1.086	1.071
	4	0.480	0.302	0.648	1.334	1.580	1.415
	8	0.324	0.208	0.536	1.393	1.893	1.911
DSTFT	2	0.487	1.055	1.772	1.088	0.968	1.093
	4	0.230	0.548	0.671	1.191	1.211	1.713
	8	0.155	0.235	0.571	1.036	1.713	2.040
DZT	2	1.221	0.722	1.106	1.201	1.023	1.443
	4	0.509	0.314	0.661	1.143	1.124	1.829
	8	0.334	0.222	0.588	1.319	1.409	1.558
DCFT	2	0.835	0.846	0.890	1.036	0.809	0.997
	4	0.525	0.724	0.781	0.701	0.646	0.705
	8	0.233	0.554	1.033	0.970	0.950	1.044
MDCFT	2	1.004	0.840	0.948	0.827	1.119	1.157
	4	0.499	0.545	0.816	0.754	0.947	0.813
	8	0.357	0.496	1.022	0.772	0.890	1.096
NDCFT	2	1.162	0.839	0.790	0.984	1.153	1.204
	4	0.499	0.800	0.660	0.703	0.659	0.770
	8	0.362	0.456	0.810	1.002	0.976	1.017

N is length of the signal and p is the number of logical processor.

57% in the range $2048 \leq N \leq 8192$. In special case $N = 8192$, we obtain an efficiency above of 70%. Furthermore, we obtain an efficiency above 100% of 42% of simulations and an efficiency above 80% in 60% of simulations. It indicates a good efficiency to calculate DT-DF transforms using Algorithm 1 and MATLAB® with the Parallel Computing Toolbox™ in a computer with similar characteristic to those used in this paper. Many research in parallel computing mention a good efficienfy when $E > 70\%$ (see, e.g., [30,31,32,33]).

5 Conclusion

This work presents a new general mathematical framework for a family of DT-DF transforms: the discrete ambiguity function (DAF), the discrete short-time Fourier transforms (DSTFT), the discrete Zak transform (DZT), the discrete chirp-Fourier transform (DCFT), the modified discrete chirp-Fourier transform (MDCFT) and the new discrete chirp-Fourier transform (NDCFT). This mathematical framework is expressed in equations (2), (4) and Algorithm 1.

This framework is possible because this DT-DF transforms have the same structure and it is expressed in equation (1). This mathematical framework is performed in terms of matrix signal algebra, which is a mathematics environment composed of a signal space, finite dimensional linear operators and special matrices, where algebraic methods are used to generate algorithms in signal processing area. The matrix signal algebra contributes to analysis, design and implementation in parallel of Algorithm 1 to compute each DT-DF transform. An experimental investigation is performed and it indicated the following results, using MATLAB® with the Parallel Computing Toolbox™ in a computer with multi-core proccesors:

– there is advantage to use multi-core processors and a parallel computing environment to minimize the high execution time (for DAF, we obtain $T_1 = 178.845$ s, $T_2 = 82.339$ s, $T_4 = 28.310$ s and $T_8 = 11.700$ s),
– speedup increases and superlinear speedup is obtained when the number of logical processor p and length of the signal N increase (42% of simulations),
– a good efficiency too is obtained when p and N increase (above 80% in 60% of simulations).

Acknowledgement

The author wish to thank Vicerrectoría de Investigación y Extensión of Instituto Tecnológico de Costa Rica, Prof. Roger Moya (Instituto Tecnológico de Costa Rica) for useful suggestions of improving the presentation of the paper and Prof. Domingo Rodríguez (University of Puerto Rico, Mayagüez Campus) for insightful discussions and to motivate the study of the digital signal processing area.

References

[1] F. Hlawatsch and G.F. Boudreaux-Bartels, “Linear and quadratic time-frequency signal representations”, IEEE Signal Processing Magazine, **9**, 21-67 (1992).
[2] M.S. Richman, T.W. Parks and R.G. Shenoy, “Discrete-time, discrete-frequency, time-frequency analysis”, IEEE Transactions on Signal Processing, **46**, 1517-1527 (1998).
[3] L. Cohen, Time-frequency analysis: theory and applications, Prentice Hall, (1995).

[4] L. Auslander and R. Tolimieri, "Computing decimated finite cross-ambiguity functions", IEEE Transactions on Acoustics, Speech and Signal Processing, **36**, 359-364 (1988).
[5] M. Bahoura and Y. Simard, "Blue whale calls classification using short-time Fourier and wavelet packet transforms and artificial neural network", Digital Signal Processing, **26**, 1256-1263 (2010).
[6] H. Blcskei and F. Hlawatsch, "Discrete Zak transforms, polyphase transforms, and applications", IEEE Transactions on Signal Processing, **45**, 851-866 (1997).
[7] X. Xiang-Gen, "Discrete chirp-Fourier transform and its application to chirp rate estimation", IEEE Transactions on Signal, **48**, 3122-3133 (2000).
[8] F. Pingyi and X. Xiang-Gen, "A modified discrete chirp-Fourier transform scheme", 5th International Conference on Signal Processing, **1**, 57-60 (2000).
[9] F. Pingyi and F. Chongxi, "A new discrete chirp Fourier transform", 6th International Conference on Signal Processing Proceedings, **1**, 49-53 (2002).
[10] J.J. Benedetto, I. Konstantinidis and M. Rangaswamy, "Phase-coded waveforms and their design", IEEE Signal Processing Magazine, **26**, 22-31 (2009).
[11] R. Tolimieri and R. S. Orr, "Poisson summation, the ambiguity function, and the theory of Weyl-Heisenberg frames", Journal of Fourier Analysis and Applications, **1**, 233-247 (1994).
[12] D. Rodriguez, J. Seguel and E. Cruz, "Algebraic methods for the analysis and design of time-frequency signal processing algorithms", IEEE International Symposium on Circuits and Systems, **1**, 196-199 (1993).
[13] D. Marquez, J. Valera, A. Camelo, C. Aceros, M. Jimenez and D. Rodriguez, "Implementations of cyclic cross-ambiguity functions in FPGAs for large scale signals", IEEE Second Latin American Symposium on Circuits and Systems, **1**, 1-4 (2011).
[14] C.A. Aceros-Moreno and D. Rodriguez, "Fast discrete chirp Fourier transforms for radar signal detection systems using cluster computer implementations", 48th Midwest Symposium on Circuits and Systems, **2**, 1047-1050 (2005).
[15] J.J. Benedetto and J.J. Donatelli, "Ambiguity function and frame-theoretic properties of periodic zero-autocorrelation waveforms", IEEE Journal of Selected Topics in Signal Processing, **1**, 6-20 (2007).
[16] R. Tolimieri, M. An and C. Lu, Algorithms for discrete Fourier transform and convolution, Springer, (1997).
[17] F. Franchetti, M. Püschel, Y. Voronenko, S. Chellappa and J. Moura, "Discrete Fourier transform on multicore", IEEE Signal Processing Magazine, special issue on "Signal Processing on Platforms with Multiple Cores", **26**, 90-102 (2009).
[18] J. Johnson, R. Johnson, D. Rodriguez and R. Tolimieri, "A Methodology for designing, modifying, and implementing Fourier transform algorithms on various architectures", Proceedings of Circuits, Systems, and Signal Processing, **9**, 449-500 (1990).
[19] C. Van Loan, Computational framework of the fast Fourier transform, SIAM, (1992).
[20] W. Yang, Signals and systems with MATLAB, Springer, (2009).
[21] P. Flandrin, "Time frequency and chirps", Proceedings of SPIE, **4391**, 161-175 (2001).
[22] MATLAB, MATLAB Multicore: Run MATLAB on multicore and multiprocessor machines, (2013), http://www.mathworks.com/.
[23] C. Moler, Parallel MATLAB: multiple processors and multiple cores, (2007), http://www.mathworks.com/.
[24] G. S. Almasi and A. Gottlieb, Highly parallel computing, Benjamin-Cummings Publishing Co, (1989).
[25] R. Trobec, M. Vajteric, P. Zinterhof, Parallel computing: numerics, applications, and trends, Springer, (2009).
[26] M. O. Tokhi, M. A. Hossain and M. H. Shaheed, Parallel computing for real-time signal processing and control, Springer, (2003).
[27] A. Camargos, R. Batalha, C. Martins, E.J. Silva and G.L. Soares, "Superlinear speedup in a 3-D parallel conjugate gradient solver", IEEE Transactions on Magnetics, **45**, 1602-1605 (2009).
[28] M. Otte and N. Correll, "C-FOREST: parallel shortest path planning with superlinear speedup", IEEE Transactions on Robotics, **29**, 798-806 (2013).
[29] J. Sienicki, V.D. Agrawal and M.L. Bushnell, "Superlinear speedup in multiprocessing environment", Proceedings of 1st International Workshop on Parallel Processing, **1**, 116-120 (1994).
[30] F. Munz, T. Stephan, U. Maier, T. Ludwig, A. Bode, S. Ziegler, S. Nekolla, P. Bartenstein and M. Schwaiger, "Improved functional imaging through network based parallel processing", Lecture Notes in Computer Science (Network-Based Parallel Computing Communication, Architecture, and Applications), **1362**, 116-120 (1998).
[31] D. Li, S. Peng, Z. Li, "Design and optimization of high efficiency parallel video coding system", Fifth International Joint Conference on Computational Sciences and Optimization, **1**, 23-26 (2012).
[32] Y.M. Pi, H. Long, S.J. Huang, "A SAR Parallel Processing Algorithm and its Implementation", Conference Proceedings Pecora 15/Land Satellite Information IV/ISPRS Commission I/FIEOS, (2002).
[33] O. Bandman, "A lattice-gas model of fluid flow through tortuous channels of hydrophilous and hydrophobic porous materials", Lecture Notes in Computer Science (Parallel Computing Technologies), **5698**, 168-181 (2009).

Pablo Soto-Quiros works in Departament of Mathematics at Instituto Tecnológico de Costa Rica since 2009 and University of Costa Rica since 2012. He received a Bachelor degree in Computer Assisted Math Teaching from Instituto Tecnológico de Costa Rica in 2007 and a M.Sc. in Applied Mathematics from University of Puerto Rico at Mayaguez, Puerto Rico, USA, in 2012. His main research interests are applied and numerical harmonic analysis, signal processing, linear programming, numerical linear algebra and numerical analysis.