Algorithms for Discrete Quadratic Time–Frequency Distributions

John M. O' Toole
Perinatal Research Centre
University of Queensland
Royal Brisbane & Women's Hospital
Herston, QLD 4029
Australia
j.otoole@ieee.org

Mostefa Mesbah
Perinatal Research Centre
University of Queensland
Royal Brisbane & Women's Hospital
Herston, QLD 4029
Australia
m.mesbah@uq.edu.au

Boualem Boashash
College of Engineering
University of Sharjah
P.O. Box 27272
Sharjah
United Arab Emirates
boualem.boashash@sharjah.ac.ae

Abstract: Time–frequency distributions (TFDs) are computationally costly to compute. We address this problem by presenting algorithms to reduce the computational load for computing TFDs. Before we can compute the TFDs, however, we first must define a discrete version of the TFD. Defining a discrete TFD (DTFD) is, unfortunately, not a straightforward process—for example, a popular DTFD definition does not satisfy all desirable mathematical properties that are inherent to the continuous TFD. In this paper, we define a new DTFD definition, the DTFD-C. This definition is closely related to another DTFD definition which we recently proposed, the DTFD-B. The DTFD-B and DTFD-C satisfy all desirable properties. We provide algorithms for both these definitions and show that the DTFD-C requires only 50% of the computational complexity and memory required to compute the DTFD-B.

Key–Words: Discrete time–frequency distributions (DTFD), discrete Wigner–Ville distributions (DWVD), discrete-time signal processing (DSP), time–frequency signal analysis, algorithms, computational load, fast Fourier transforms (FFTs)

1 Introduction

The signal analyst requires high performance at a low cost. Time–frequency methods provide high performance analysis tools for nonstationary signals [1–4], which are signals that have time-varying frequency characteristics. But, unfortunately, using time–frequency methods comes at a cost—a computational cost. Time–frequency methods use time–frequency distributions (TFDs)—two-dimensional functions that represents the time–frequency domain. To compute these TFDs requires a large computational load in comparison to the one-dimensional time or frequency methods.

In this paper, we present simple algorithms to reduce the computational cost of TFDs. Because a digital device, such as a computer, requires a discrete TFD (DTFD), we first need to define a discrete version of the continuous TFD before we design the algorithms. Ideally, the DTFD should satisfy all desirable mathematical properties of the continuous distribution. (These desirable properties are a set of commonly presented properties [5–9].)

1.1 Findings and Scope of Paper

We start this paper with a review of two existing DTFDs definitions: a popular DTFD definition [10] which we call the DTFD-A; and a definition which we recently proposed [9], the DTFD-B. In addition, we define a new DTFD definition, the DTFD-C. Next, we compare the properties of the three definitions—the DTFD-C and DTFD-B satisfy all desirable properties. Then, we present algorithms for the DTFD-C and DTFD-B. Finally, we look at the computational load for each algorithm and show that the DTFD-C requires only 50% of the computational complexity and memory required to compute the DTFD-B.

In this paper, we consider only the class of quadratic real-valued TFDs—a popular class of TFDs [11]. We do not consider the AF-GDTFD definition [8] because it contains aliasing [9]. Also, we assume that the signal under analysis is real valued.

2 Review of Existing Definitions

Any quadratic TFD can be expressed in terms of the Wigner–Ville distribution $W_z(t,f)$ and time–frequency smoothing kernel $\gamma(t,f)$ as

$$\rho_z(t,f) = W_z(t,f) \underset{t}{*} \underset{f}{*} \gamma(t,f) \tag{1}$$

where the symbol $*$ represents the convolution operation. The Wigner–Ville distribution (WVD) of $s(t)$

is

$$W_z(t,f) = \int_{-\infty}^{\infty} z(t+\tfrac{\tau}{2})z^*(t-\tfrac{\tau}{2})\mathrm{e}^{-\mathrm{j}2\pi\tau f}\mathrm{d}\tau$$

where $z(t)$ is the analytic associate $s(t)$ [11].

Because the TFD $\rho(t,f)$ is a function of two variables, t and f, we can transform the TFD through four different domains using the Fourier transform as

$$\begin{tikzcd} W_z(t,f) \underset{t}{*}\underset{f}{*} \gamma(t,f) \arrow[r] & \mathcal{K}_z(\nu,f) \underset{f}{*} \mathcal{G}(\nu,f) \\ K_z(t,\tau) \underset{t}{*} G(t,\tau) \arrow[u] \arrow[r] & A_z(\nu,\tau)g(\nu,\tau) \arrow[u] \end{tikzcd} \quad (2)$$

where the variables (t,τ), (ν,τ), (ν,f), and (t,f) represent the time–lag, doppler–lag, doppler–frequency, and time–frequency domains, respectively; the arrows in (2) represent Fourier transforms. Thus we can implement the time–frequency convolution as a multiplication in the doppler–lag domain. We shall use this approach to generate the DTFD.

To define a DTFD, we therefore require a discrete WVD (DWVD) and a discrete kernel. Because the kernel is independent of the signal, we can sample the kernel is any of the four domains, assuming a closed-form expression for the kernel exists in that domain. Also, if we assume that the kernel is time and frequency bandlimited, then the discrete kernel will be alias free. The difficult arises, however, when forming the DWVD, as we cannot sample the WVD in the time–frequency domain; instead, we must form the DWVD from the discrete-time signal. Before we review DTFD definitions, we start with a review of DWVD definitions.

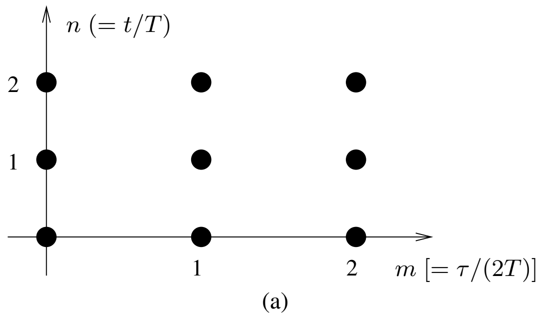
2.1 Discrete Wigner–Ville Distributions

To ensure that the discrete WVD (DWVD), and therefore the DTFD, is alias free, we transform the discrete N-point real-valued signal $s(nT)$ to the $2N$-point complex-valued signal $y(nT)$ using the method in [12]. This new signal $y(nT)$ has the form

$$\begin{aligned} y(nT) &= 0, && \text{for } N \le n \le 2N-1, \\ Y(\tfrac{k}{2NT}) &\approx 0, && \text{for } N \le k \le 2N-1, \end{aligned} \quad (3)$$

where $Y(k/2NT)$ is the discrete Fourier transform of $y(nT)$. We exploit the property of $y(nT)$ in (3) in the algorithms of Section 5.

There are two popular DWVD definitions, which we name the DWVD-A and the DWVD-B.



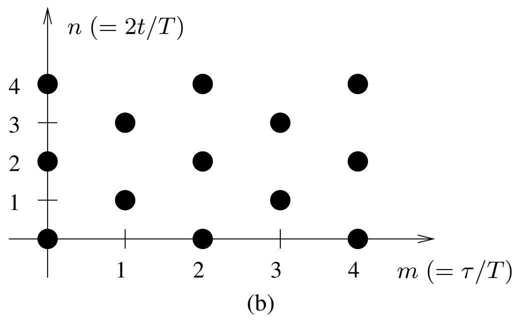


Figure 1: Different time–lag (t,τ) sampling grids with sampling period T: (a) discrete grid $(nT, 2mT)$ and (b) discrete grid $(nT/2, mT)$.

2.1.1 DWVD-A

Claasen and Mecklenbräuker [5] sampled the time–lag function $K(t,\tau)$ in time t with sample frequency $1/T$ and in lag τ with sample frequency $1/2T$. This methods uses the discrete grid shown in Fig. 1a to obtain the discrete function $K^{\mathrm{A}}(nT, 2mT)$. This discrete function, in terms of the discrete signal $y(nT)$, is

$$K_y^{\mathrm{A}}(nT, 2mT) = y((n+m)T)y^*((n-m)T).$$

To compute the DWVD, we simply discrete Fourier transform (DFT) $K_y^{\mathrm{A}}(nT, 2mT)$:

$$W_y^{\mathrm{A}}(nT, \tfrac{k}{2NT}) = \sum_{m=0}^{N-1} K_y^{\mathrm{A}}(nT, 2mT)\mathrm{e}^{-\mathrm{j}\pi mk/N}$$

for $n, k = 0, 1, \ldots, N-1$.

2.1.2 DWVD-B

Chan [13] proposed another approach for sampling the time–lag function. The method uses a nonuniform sampling grid, illustrated in Fig. 1c. The resultant time–lag function $K^{\mathrm{B}}(nT/2, mT)$, for even–odd values of n, is

$$\begin{aligned} K_y(nT, 2mT) &= y((n+m)T)y^*((n-m)T) \\ K_y((n+\tfrac{1}{2})T, &(2m+1)T) \\ &= y((n+m+1)T)y^*((n-m)T). \end{aligned}$$

The DWVD is the DFT of time–lag function:

$$W_y^{\mathrm{B}}(\tfrac{nT}{2}, \tfrac{k}{4NT}) = \sum_{m=0}^{2N-1} K_y^{\mathrm{B}}(\tfrac{nT}{2}, mT)\mathrm{e}^{-\mathrm{j}\pi mk/(2N)}$$

for $n, k = 0, 1, \ldots, 2N-1$.

The array size for the two DWVD definitions differ: DWVD-A is an N by N array whereas DWVD-B is an $2N$ by $2N$ array. Hence the DWVD-B is more computational expensive to compute. The advantage of DWVD-B, however, is that this definition satisfies more desirable properties than DWVD-A satisfies [6].

The two DWVDs are closely related [6, 14]: DWVD-A is a decimated, in time and frequency, version of DWVD-B, as

$$W_y^{\mathrm{A}}(nT, \tfrac{k}{2NT}) = W_y^{\mathrm{B}}(\tfrac{2nT}{2}, \tfrac{2k}{4NT}). \tag{4}$$

We proposed another DWVD definition [7], the DWVD-C, similar to the preceding relation:

$$W_y^{\mathrm{C}}(\tfrac{nT}{2}, \tfrac{k}{2NT}) = W_y^{\mathrm{B}}(\tfrac{nT}{2}, \tfrac{2k}{4NT}).$$

DWVD-C also satisfies all the desirable properties that DWVD-B satisfies [7].

2.2 Discrete Time–Frequency Distributions

2.2.1 DTFD-A

The method to form the DTFD-A, sometimes called the generalised DTFD, is as follows [10]. First, define the discrete doppler–lag kernel $g^{\mathrm{A}}(l/NT, 2mT)$ for $l, m = 0, 1, \ldots, N-1$, which we represent as $\gamma^{\mathrm{A}}(nT, k/2NT)$ in the time–frequency domain. Second, form the DWVD-B from $y(nT)$. Third, convolve the time–frequency kernel with the DWVD:

$$\rho_y^{\mathrm{A}}(nT, \tfrac{k}{2NT}) = W_y^{\mathrm{A}}(nT, \tfrac{k}{2NT}) \underset{n}{\circledast} \underset{k}{\circledast} \gamma^{\mathrm{A}}(nT, \tfrac{k}{2NT}).$$

for $n, k = 0, 1, \ldots, N-1$. The symbol $\circledast$ represents the circular convolution operation.

2.2.2 DTFD-B

To form the DTFD-B, we do as follows [9]. First, define the discrete doppler–lag kernel $g^{\mathrm{B}}(l/2NT, mT)$ for $l, m = 0, 1, \ldots, 2N-1$, which we represent as $\gamma^{\mathrm{B}}(nT/2, k/4NT)$ in the time–frequency domain. Second, form the DWVD-B from $y(nT)$. And finally, convolve these two functions:

$$\rho_y^{\mathrm{B}}(\tfrac{nT}{2}, \tfrac{k}{4NT}) = W_y^{\mathrm{B}}(\tfrac{nT}{2}, \tfrac{k}{4NT}) \underset{n}{\bar{\circledast}} \underset{k}{\bar{\circledast}} \gamma^{\mathrm{B}}(nT, \tfrac{k}{2NT}).$$

for $n, k = 0, 1, \ldots, 2N-1$. The symbol $\bar{\circledast}$ represents a *modified* circular convolution operation that compensates for the nonstandard periodic form of the DWVD-B [6,9].

3 The DTFD-C Definition

We propose another DTFD definition by decimating DTFD-B by a factor of two in the frequency direction. Thus, we express this new DTFD definition, which we call the DTFD-C, as

$$\rho_y^{\mathrm{C}}(\tfrac{nT}{2}, \tfrac{k}{2NT}) = \rho_y^{\mathrm{B}}(\tfrac{nT}{2}, \tfrac{2k}{4NT}). \tag{5}$$

4 Properties

Here we present a set of properties, inherent to the continuous TFD, which the discrete definition should, ideally, satisfy [5–8, 11]. We define this set of properties for the DTFD-B. We can simply modify the properties for the other two DTFD definitions.

- Time–frequency covariance: A signal of the form

  $$y(nT) = x((n-n_0)T)\mathrm{e}^{\mathrm{j}\pi k_0 n/N}$$

  produces the following shift in the DTFD:

  $$\rho_y^{\mathrm{B}}(\tfrac{nT}{2}, \tfrac{k}{4NT}) = \rho_x^{\mathrm{B}}((n-2n_0)\tfrac{T}{2}, (k-2k_0)\tfrac{1}{4NT}).$$

- Time marginal: By summing along the frequency direction in the DTFD, we get the instantaneous power of the signal:

  $$\sum_{k=0}^{2N-1} \rho_y^{\mathrm{B}}(\tfrac{2nT}{2}, \tfrac{k}{4NT}) = |y(nT)|^2 .$$

- Frequency marginal: By summing along the time direction in the DTFD, we get the energy spectrum:

  $$\sum_{n=0}^{2N-1} \rho_y^{\mathrm{B}}(\tfrac{nT}{2}, \tfrac{2k}{4NT}) = \frac{1}{2N}\left|Y(\tfrac{k}{2NT})\right|^2 .$$

- Time support: If the time-domain signal

  $$y(nT) = 0, \quad \text{for } n < 2n_1 \text{ and } n > 2n_2,$$

  then

  $$\rho_y^{\mathrm{B}}(\tfrac{nT}{2}, \tfrac{k}{4NT}) = 0, \quad \text{for } n < 2n_1 \text{ and } n > 2n_2.$$

- Frequency support: If the frequency-domain signal

  $$Y(\tfrac{k}{2NT}) = 0, \quad \text{for } k < 2k_1 \text{ and } k > 2k_2,$$

  then

  $$\rho_y^{\mathrm{B}}(\tfrac{nT}{2}, \tfrac{k}{4NT}) = 0, \quad \text{for } k < 2k_1 \text{ and } k > 2k_2.$$

- Instantaneous frequency: The periodic first moment [5] [15, pp. 463] of the DTFD, with respect to frequency, is equal to the instantaneous frequency of the signal:

$$\arg\left[\sum_{k=0}^{2N-1}\rho^{\mathrm{B}}(\tfrac{2nT}{2},\tfrac{k}{4NT})\mathrm{e}^{\mathrm{j}\pi k/N}\right]\mod 2\pi$$
$$=2\left[\frac{\varphi(n+1)-\varphi(n-1)}{2}\mod \pi\right].$$

  We assume that the signal $y(nT)$ has the form $y(nT)=A(n)\mathrm{e}^{\mathrm{j}\varphi(n)}$.

- Group delay: The periodic first moment of the DTFD, with respect to time, is equal to the group delay of the signal:

$$\arg\left[\sum_{n=0}^{2N-1}\rho^{\mathrm{B}}(\tfrac{nT}{2},\tfrac{2k}{4NT})\mathrm{e}^{-\mathrm{j}\pi n/N}\right]\mod -2\pi$$
$$=\frac{\theta(k+1)-\theta(k-1)}{2}\mod -\pi.$$

  We assume that the spectral signal $Y(k/2NT)$ has the form $Y(k/2NT)=a(k)\mathrm{e}^{\mathrm{j}\theta(k)}$.

- Moyal's formula: This property, also know as unitarity or inner-product invariance, states that

$$2N\sum_{n=0}^{2N-1}\sum_{k=0}^{2N-1}\rho_x^{\mathrm{B}}(\tfrac{nT}{2},\tfrac{k}{4NT})\rho_y^{\mathrm{B}}(\tfrac{nT}{2},\tfrac{k}{4NT})$$
$$=\left|\sum_{n=0}^{N-1}x(nT)y(nT)\right|^2.$$

- Signal recovery: We can recover the time-domain signal, up to a constant phase, from the DTFD:

$$\sum_{k=0}^{2N-1}\rho_y^{\mathrm{B}}(\tfrac{nT}{2},\tfrac{k}{4NT})\mathrm{e}^{\mathrm{j}\pi kn/2N}=y(nT)y^*(0)$$

Both the DTFD-C and the DTFD-B satisfy all these properties; the DTFD-A does not satisfy all these properties [8], as we detail in Table 1.

Table 1: Properties for the three DTFD definitions.

	DTFD-A	DTFD-C	DTFD-B
nonnegative	✓	✓	✓
TF covariance	✓	✓	✓
time marginal	✓	✓	✓
freq. marginal		✓	✓
time support	✓	✓	✓
freq. support	✓	✓	✓
IF	✓	✓	✓
group delay		✓	✓
Moyal's formula		✓	✓
signal recovery		✓	✓

Legend: IF: instantaneous frequency; TF: time–frequency

5 Algorithms

To compute the DTFDs definitions efficiently, we implement the two convolutions operations in (1) using the discrete Fourier transform (DFT) method [10,16,17]. The steps are as follows, using the DTFD-A definition as an example. To simplify the notation we use sequence notation wherever possible—for example, we write $\rho^{\mathrm{A}}[n,k]$ instead of $\rho^{\mathrm{A}}(nT,k/2NT)$.

1. Generate the time–lag function $K[n,m]$ from the signal $y[n]$.
2. Obtain the time–lag function $R[n,m]$ as follows:
$$R[n,m]=\underset{l\to n}{\mathrm{IDFT}}\left\{\underset{n\to l}{\mathrm{DFT}}\left\{K[n,m]\right\}g[l,m]\right\}\tag{6}$$
   where $g[l,m]$ is the doppler–lag kernel and IDFT represents the inverse DFT.
3. Form the DTFD from the relation
$$\rho_y^{\mathrm{A}}[n,k]=\underset{m\to k}{\mathrm{DFT}}\left\{R[n,m]\right\}.\tag{7}$$

Fig. 2 illustrates this simple process.

In the algorithms for DTFD-C and DTFD-B, we use two different discrete representations for the time–lag function because this function has a nonuniform discrete grid. From Fig. 3a, we can see that the nonuniform gird $R(nT/2,mT)$ is not in a suitable form for storage in an array. Thus, we have two options: either we shift the sample points at $R((n+1/2)T/2,(2m+1)T)$, highlighted as the grey points in Fig. 3a, across in the lag direction by T or we shift these sample points down in the time direction by $T/2$. We call the function shifted in time the *shifted-down* array $R^{\mathrm{d}}[n,m]$ and the function shifted in lag the *shifted-across* array $R^{\mathrm{a}}[n,m]$—Fig. 3 illustrates this process. Note that both arrays contain the same sample points but are ordered differently. Also, we define the variable N_{h}, which we use in the following algorithms, as

$$N_{\mathrm{h}}=\lceil N/2\rceil.$$

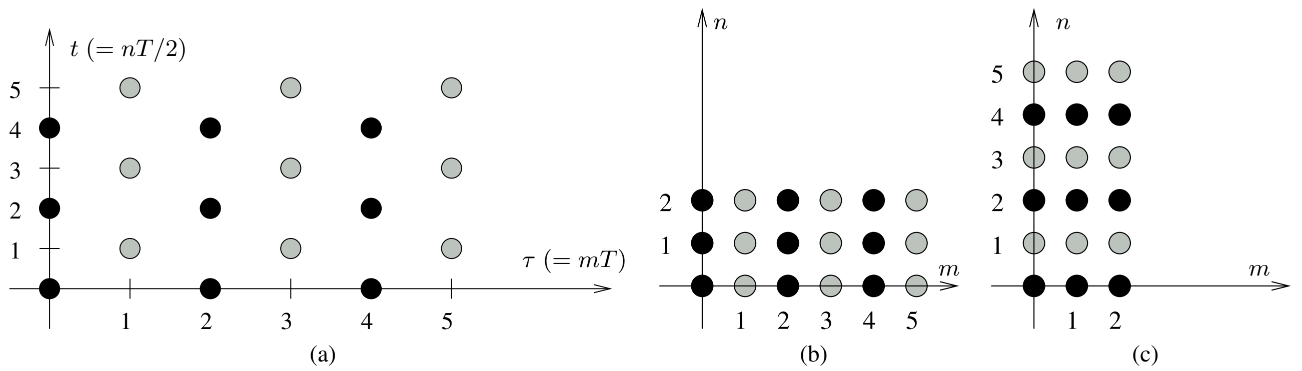


Figure 3: Two ways to store the time–lag function $R(nT/2, mT)$ as an array. (a) $R(nT/2, mT)$, (b) the shifted-down array $R^{\mathrm{d}}[n,m]$, and (c) the shifted-across array $R^{\mathrm{a}}[n,m]$.

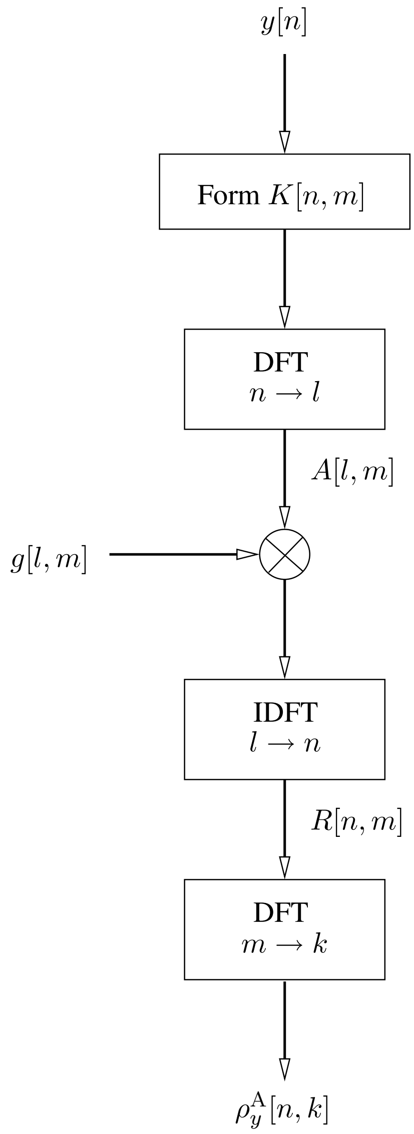


Figure 2: Flowchart for computing the DTFD-A.

where the function $\lceil x \rceil$ returns an integer larger than or equal to x.

5.1 DTFD-A Algorithm

Reilly and Boashash presented an algorithm for the DTFD-A [10]. Because this algorithm forms the basis of the algorithms for the DTFD-C and DTFD-B, we outline it here.

The algorithm takes advantage of the conjugate symmetrical property of the time–lag function $R[n,m]$. There are two benefits to this. First, we can halve the number of DFTs required to generate the function $R[n,m]$, described in (6), by only computing this time–lag function for positive lag m values. We then recover the negative values from the conjugate symmetry [10].

Second, because the DTFD is real valued, we can reduce the number of DFTs in (7). We may use an inverse real-valued fast Fourier transform (FFT) algorithm to implement these DFT operations [18]. An alternative to the inverse real-valued FFT method is a method that uses a real-valued FFT with minimal computational overhead [7]. The advantage of the latter method is that real-valued FFTs are more readily available in signal processing software packages compared with the inverse real-valued FFT algorithms.

5.2 DTFD-C Algorithm

In this implementation, we reduce the computational load by two procedures. First, we compute the time–lag function $R[n,m]$ for positive lag values only, as we previously described. Second, we use the real value property of the DTFD to further reduce the computational load when going from the time–lag to the time–frequency domain. For this transformation, we must employ a new technique because, unlike the function for the DTFD-A, the time–lag function for the DTFD-C is not conjugate symmetrical. This new technique is similar to one we proposed for the DWVD-C [7]. We shall now detail these two procedures.

Lets first examine the characteristics of the shifted-across function $R^{\mathrm{a}}[n,m]$ for the DTFD-C. For even n values, the function is conjugate symmetrical with respect to the lag axis—that is, $R^{\mathrm{a}}[2n,m] = (R^{\mathrm{a}})^*[2n,N-m]$. Thus, for even n values we can implement the DFT operations efficiently, as mentioned in Section 5.1.

A problem arises, however, when n is odd, as the DFT of $R^{\mathrm{a}}[2n+1,m]$ is complex valued because $R^{\mathrm{a}}[2n+1,m] \neq (R^{\mathrm{a}})^*[2n+1,N-m]$ [7]. Thus, we need to modulate the output of the DFT to ensure a real-valued DTFD [6]:

$$\rho_y^{\mathrm{C}}[2n{+}1,k] = \mathrm{e}^{-\mathrm{j}\pi k/N} \underset{m\to k}{\mathrm{DFT}} \left\{ R^{\mathrm{a}}[2n+1,m] \right\}. \quad (8)$$

Unlike for n even, the computational load cannot be reduced because the DFTs in (8) produces a complex valued output.

To implement (8) efficiently, we do the following. We rewrite (8) as

$$\rho_y^{\mathrm{C}}[2n+1,k] = \left[\frac{\cos^2(\pi k/N)}{\sin(\pi k/N)} + \sin(\pi k/N) \right] \cdot \Im\left(\underset{m\to k}{\mathrm{DFT}} \left\{ R^{\mathrm{a}}[2n+1,m] \right\} \right) \quad (9)$$

for $k = 1,\dots,N-1$, where the function $\Im(c)$ returns the imaginary part of c. Next, we define a function $\bar{R}^{\mathrm{a}}[n,m]$ so that the DFT of $\bar{R}^{\mathrm{a}}[n,m]$ equals the imaginary part of the DFT of $R^{\mathrm{a}}[n,m]$—that is,

$$\underset{m\to k}{\mathrm{DFT}} \left\{ \bar{R}^{\mathrm{a}}[n,m] \right\} = \Im\left(\underset{m\to k}{\mathrm{DFT}} \left\{ R^{\mathrm{a}}[2n+1,m] \right\} \right)$$

where $\bar{R}^{\mathrm{a}}[n,m]$ is a function of $R^{\mathrm{a}}[n,m]$. Therefore, because the DFT of $\bar{R}^{\mathrm{a}}[n,m]$ has a real-valued output, we can reduce the computational load.

When $k=0$, equation (9) is undefined. For this special case, we simply sum along the lag values:

$$\rho_z^{\mathrm{C}}[2n+1,0] = \sum_{m=0}^{N-1} R^{\mathrm{a}}[2n+1,m].$$

The following algorithm details the method. In this algorithm, we use the shifted-down array $\bar{R}^{\mathrm{d}}[n,m]$, rather than the shifted-across array $\bar{R}^{\mathrm{a}}[n,m]$, so we can easily integrate it with the first part of the algorithm, as we shall see.

- **Input**: $2N$-point analytic signal $y[n]$ and N by $(N+1)$ kernel $g^{\mathrm{C}}[l,m]$.
- **Output** : $2N$ by N DTFD array $\rho_y^{\mathrm{C}}[n,k]$.

1. Form the time–lag array $K^{\mathrm{d}}[n,m]$ from the signal $y[n]$:
$$K^{\mathrm{d}}[n,2m] = y[n+m]y^*[n-m]$$
$$K^{\mathrm{d}}[n,2m+1] = y[n+m+1]y^*[n-m]$$
for $n=0,1,\dots,N-1$ and $m=0,1,\dots,N_{\mathrm{h}}$. The array $K^{\mathrm{d}}[n,m]$ is the shifted-down version of the function $K(nT/2,mT)$, as illustrated in Fig. 3.
2. DFT $K^{\mathrm{d}}[n,m]$ to the doppler–lag domain to obtain the discrete ambiguity function (DAF) $A[l,m]$:
$$A[l,m] = \underset{n\to l}{\mathrm{DFT}} \left\{ K_z^{\mathrm{d}}[n,m] \right\}$$
for $m=0,1,\dots,N_{\mathrm{h}}$.
3. Multiply by the kernel, for all l,m values:
$$S[l,m] = A[l,m]g^{\mathrm{C}}[l,m].$$
4. IDFT back to the time–lag domain:
$$R^{\mathrm{d}}[n,m] = \underset{l\to n}{\mathrm{IDFT}} \left\{ S[l,m] \right\}$$
for $m=0,1,\dots N_{\mathrm{h}}$.
5. Recover the negative lag values from the positive ones:
$$R^{\mathrm{d}}[n,2N-2m] = (R^{\mathrm{d}})^*[n,2m]$$
for $m=1,2,\dots,N_{\mathrm{h}}-1$ and
$$R^{\mathrm{d}}[n,2N-2m-1] = (R^{\mathrm{d}})^*[n,2m+1]$$
for $m=0,1,\dots,N_{\mathrm{h}}-1$.
6. Finally, transform the time–lag function $R^{\mathrm{d}}[n,m]$ to the time–frequency domain to obtain the DTFD. Do as follows for n even and n odd:
   (a) for n even,
   $$\rho^{\mathrm{C}}[2n,k] = \underset{m\to k}{\mathrm{DFT}} \left\{ R^{\mathrm{d}}[n,2m] \right\}$$
   (b) for n odd, do the following:
      i. Let
      $$h[k] = \frac{\cos^2(\frac{\pi k}{N})}{\sin(\frac{\pi k}{N})} + \sin(\tfrac{\pi k}{N})$$
      for $k=1,2,\dots,N-1$, and $h[k]=0$ for $k=0$.

ii. Over $n = 0, 1, \ldots, N-1$, let

$$\bar{R}^{\mathrm{d}}[n,0] = \Im\left(R^{\mathrm{d}}[n,0]\right)$$

and

$$\bar{R}^{\mathrm{d}}[n,m] = \frac{1}{2j}\Big(R^{\mathrm{d}}[n,2m+1] - (R^{\mathrm{d}})^*[n,2N-2m-1]\Big),$$

for $m = 1, 2, \ldots, N_{\mathrm{h}}$. Then, using the conjugate symmetry of $\bar{R}^{\mathrm{d}}[n,m]$ recover the negative lag values:

$$\bar{R}^{\mathrm{d}}[n,m] = (\bar{R}^{\mathrm{d}})^*[n,2N-2m-1],$$

for $m = N_{\mathrm{h}}+1, N_{\mathrm{h}}+2, \ldots, N-1$.

iii. DFT to the time–frequency domain and multiply by constant $h[k]$:

$$\rho_y^{\mathrm{C}}[2n+1,k] = \underset{m\to k}{\mathrm{DFT}}\left\{\bar{R}^{\mathrm{d}}[n,m]\right\} h[k].$$

iv. Do for frequency sample $k = 0$,

$$\rho_y^{\mathrm{C}}[2n+1,0] = \sum_{m=0}^{N-1} R^{\mathrm{d}}[n,2m+1]$$

5.3 DTFD-B Algorithm

The DTFD-B algorithm is closely related to the DTFD-C algorithm with one major difference—the time–lag function $R(t = nT/2, \tau = mT)$ extends in the lag direction from $|\tau| \le 2NT$, unlike the DTFD-A or DTFD-C time–lag functions which extend only from $|\tau| \le NT$. They have different regions of support because the time–lag function of the DWVDs associated with the DTFD definitions are different. We show these regions of support in Fig. 4.

To reduce the computational load, we compute the time–lag function $R[n,m]$ in the lag direction for $m = 0, 1, \ldots, N-1$, which corresponds to the continuous region $0 \le \tau \le NT$. Then, we recover the lag region $NT < \tau \le 2NT$ from the initial time–lag function. To do so, we make use of the relation [6, 10]

$$R(\tfrac{nT}{2}, (2N-m)T) = R^*((n-N)\tfrac{T}{2}, mT) \quad (10)$$

We now present the algorithm.

- **Input:** $2N$-point analytic signal $y[n]$ and $2N$ by $(N+1)$ kernel $g^{\mathrm{B}}[l,m]$.
- **Output:** $2N$ by $2N$ DTFD $\rho_y^{\mathrm{B}}[n,k]$.

1. Form the time–lag function $K^{\mathrm{d}}[n,m]$ from the signal $y[n]$:

$$K^{\mathrm{d}}[n,2m] = y[n+m]y^*[n-m]$$
$$K^{\mathrm{d}}[n,2m+1] = y[n+m+1]y^*[n-m]$$

for $n = 0, 1, \ldots, N-1$ and

$$K^{\mathrm{d}}[n,m] = 0$$

for $n = N, N+1, \ldots, 2N-1$. The $2N$ by $(N+1)$ array $K^{\mathrm{d}}[n,m]$ is a shifted-down version of version of the function $K(nT/2, mT)$, as illustrated in Fig. 3b.

2. DFT $K^{\mathrm{d}}[n,m]$ to the doppler–lag domain to obtain the DAF $A[l,m]$:

$$A[l,m] = \underset{n\to l}{\mathrm{DFT}}\left\{K_z^{\mathrm{d}}[n,m]\right\}$$

for $m = 0, 1 \ldots, N$.

3. Multiple the DAF by the kernel, for all l, m values:

$$S[l,m] = A[l,m]g^{\mathrm{B}}[l,m].$$

4. IDFT back to the time–lag domain:

$$R^{\mathrm{d}}[n,m] = \underset{l\to n}{\mathrm{IDFT}}\left\{S[l,m]\right\}$$

for $m = 0, 1 \ldots, N$.

5. Expand the time–lag function in the positive lag direction using the relation in (10),

$$R^{\mathrm{d}}[n,2N-m] = (R^{\mathrm{d}})^*[n-N,m]$$

for $m = 0, 1, \ldots, N-1$ and $n = 0, 1, \ldots, 2N-1$. The array $R^{\mathrm{d}}[n,m]$ is now $2N$ by $(2N+1)$. This expansion process is equivalent to expanding the continuous function $R(t,\tau)$ from $0 \le \tau < NT$ to $0 \le \tau < 2NT$.

6. Limit $R^{\mathrm{d}}[n,m]$ from $n = 0, 1, \ldots, 2N-1$ to $n = 0, 1, \ldots, N-1$; the array $R^{\mathrm{d}}[n,m]$ is now N by $(2N+1)$.

7. Recover the negative lag values from the positive ones:

$$R^{\mathrm{d}}[n,4N-2m] = (R^{\mathrm{d}})^*[n,2m],$$

for $m = 1, 2, \ldots, N-1$ and

$$R^{\mathrm{d}}[n,4N-2m-1] = (R^{\mathrm{d}})^*[n,2m+1],$$

for $m = 0, 1, \ldots, N-1$. The array $R^{\mathrm{d}}[n,m]$ is now N by $4N$.

8. Finally, transform the time–lag function $R^{\mathrm{d}}[n,m]$ to the time–frequency domain to obtain the DTFD. Do as follows for n even and n odd:

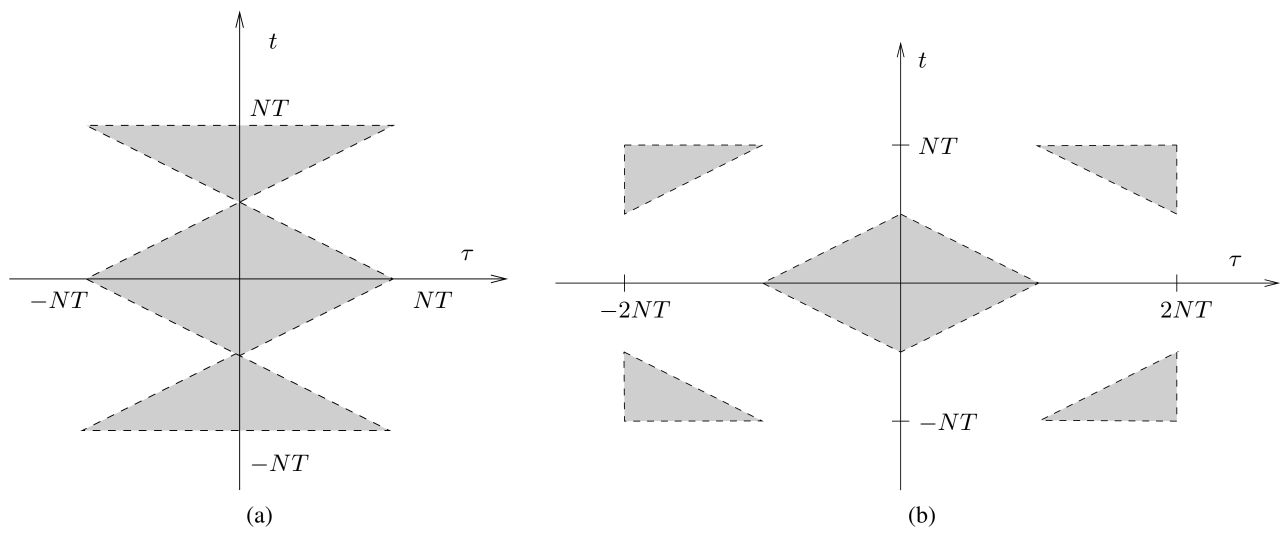


Figure 4: Region of support for time–lag (t, τ) functions of the (a) DWVD-A and DWVD-C, and (b) DWVD-B. The functions outside the grey shaded area are zero.

(a) for n even,

$$\rho^{\mathrm{B}}[2n, k] = \underset{m \to k}{\mathrm{DFT}} \left\{ R^{\mathrm{d}}[n, 2m] \right\}$$

(b) for n odd, do the following:

i. Let

$$h[k] = \frac{\cos^2(\frac{\pi k}{2N})}{\sin(\frac{\pi k}{2N})} + \sin(\frac{\pi k}{2N})$$

for $k = 1, 2, \ldots, 2N-1$, and $h[k] = 0$ for $k = 0$.

ii. Over $n = 0, 1, \ldots, N-1$, let

$$\bar{R}^{\mathrm{d}}[n, 0] = \Im \left(R^{\mathrm{d}}[n, 0] \right)$$

and

$$\bar{R}^{\mathrm{d}}[n, m] = \frac{1}{2j} \Big(R^{\mathrm{d}}[n, 2m+1] - (R^{\mathrm{d}})^*[n, 4N - 2m - 1] \Big)$$

for $m = 1, 2, \ldots, N$. Then, using the conjugate symmetry of $\bar{R}^{\mathrm{d}}[n, m]$, recover the negative lag values:

$$\bar{R}^{\mathrm{d}}[n, m] = (\bar{R}^{\mathrm{d}})^*[n, 2N - 2m - 1],$$

for $m = N + 1, N + 2, \ldots, 2N - 1$.

iii. DFT to the time–frequency domain and multiply by constant $h[k]$:

$$\rho_y^{\mathrm{C}}[2n+1, k] = \underset{m \to k}{\mathrm{DFT}} \left\{ \bar{R}^{\mathrm{d}}[n, m] \right\} h[k].$$

iv. Do for frequency sample $k = 0$,

$$\rho_y^{\mathrm{C}}[2n+1, 0] = \sum_{m=0}^{2N-1} R^{\mathrm{d}}[n, 2m+1]$$

6 Computational Load

We now quantify the computational complexity and memory required by each algorithm. We define the term computational complexity as the total number of additions and multiplications used by the algorithm. We look only at the number of DFTs used by each algorithm as this accounts for most of the computational complexity [10, 17].

We make three assumptions. First, we assume that an N-point DFT uses $cN \log_2 N$ real multiplications and real additions. The constant c is specific to the type of DFT algorithm. Second, we assume that the complexity for the DFT of a conjugate-symmetrical signal is equal to half the complexity of a DFT for complex-valued signal [7, 18]. Third, we assume that $2N_{\mathrm{h}} \approx N$.

We present the results in Table 2 where we can see that there is a two-fold increase in computational complexity between the three definitions. Specifically, the computational load for the DTFD-C is twice that for the DTFD-A; the computational load for the DTFD-B is twice that for the DTFD-C. Fig. 5 displays this relation.

7 Conclusion

The DTFD-C and DTFD-B satisfy all desirable properties, whereas the DTFD-A does not. The DTFD-C requires only half of the computational complexity and memory to compute compared with that for the DTFD-B. Hence, the newly proposed definition, the DTFD-C, has the clear advantage over the other two definitions: it provides the signal analyst with a high

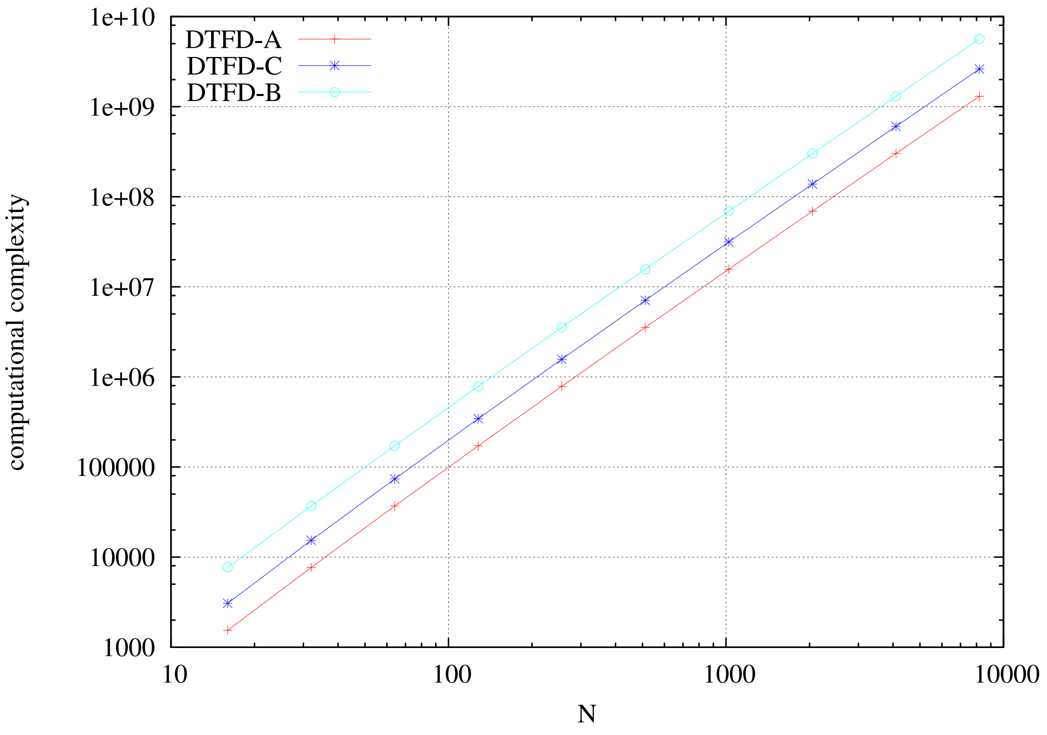


Figure 5: Computational complexity for the three algorithms. We define computational complexity as the total number of additions and multiplications used by the algorithm and N is the signal length. The plot is on a log–log scale.

	Computational complexity	Memory (array size)
DTFD-A	$6N_{\mathrm{h}}^2 \log_2 N$	N by N
DTFD-C	$12N_{\mathrm{h}}^2 \log_2 N$	$2N$ by N
DTFD-B	$24N_{\mathrm{h}}^2 \log_2 2N$	$2N$ by $2N$

Table 2: Computational load for the three DTFD definitions.

performance tool, by retaining all useful properties, at a low (computational) cost.

References:

[1] A. Akan, E. Onen, and L. F. Chaparro, "A new time-frequency based channel estimation approach for wireless OFDM systems," *WSEAS Trans. on Communications*, vol. 5, pp. 1033–1038, 2006.

[2] E. R. Lontis and A. G. Bezerianos, "Signal-dependent time-frequency analysis of transient episodes in heart rate variability," *WSEAS Trans. on Signal Processing*, vol. 1, pp. 196–202, 2005.

[3] D. Boutana, F. Marir, and M. Nibouche, "Arabic stop consonants speech signal characterization in a joint time–frequency plane," *WSEAS Trans. on Circuits and Systems*, vol. 3, pp. 1756–1761, Nov. 2004.

[4] J. M. O' Toole, M. Mesbah, B. Boashash, and P. Colditz, "A new neonatal seizure detection technique based on the time–frequency characteristics of the electroencephalogram," in *Proc. Int. Sym. on Signal Processing and its Applications, ISSPA-07*, vol. III, Sharjah, United Arab Emirates, Feb. 12–15 2007, pp. 132–135.

[5] T. Claasen and W. Mecklenbräuker, "The Wigner distribution—a tool for time–frequency signal analysis. Part II: discrete-time signals," *Philips J. Research*, vol. 35, pp. 276–350, 1980.

[6] F. Peyrin and R. Prost, "A unified definition for the discrete-time, discrete-frequency, and discrete-time/frequency Wigner distributions," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 34, no. 4, pp. 858–866, Aug. 1986.

[7] J. O' Toole, M. Mesbah, and B. Boashash, "A discrete time and frequency Wigner–Ville distribution: properties and implementation," in

Proc. Int. Conf. on Digital Signal Processing and Comm. Systems, vol. CD-ROM, Dec. 19–21, 2005.

[8] J. Jeong and W. Williams, "Alias-free generalized discrete-time time–frequency distributions," *IEEE Trans. Signal Processing*, vol. 40, pp. 2757–2765, Nov. 1992.

[9] J. M. O' Toole, M. Mesbah, and B. Boashash, "A new definition of discrete quadratic time–frequency distributions," in *Proc. Sixteenth European Signal Processing Conf. EUSIPCO-08*, accepted for publication.

[10] B. Boashash and A. Reilly, "Algorithms for time–frequency signal analysis," in *Time–Frequency Signal Analysis: Methods and Applications*, B. Boashash, Ed. Melbourne 3205: Wiley Press, 1992, ch. 7, pp. 163–181.

[11] B. Boashash, Ed., *Time–Frequency Signal Analysis and Processing: A Comprehensive Reference*. Oxford, UK: Elsevier, 2003.

[12] J. M. O' Toole, M. Mesbah, and B. Boashash, "A new discrete analytic signal for reducing aliasing in the discrete Wigner–Ville distribution," *IEEE Trans. Signal Processing*, accepted for publication.

[13] D. Chan, "A non-aliased discrete-time Wigner distribution for time–frequency signal analysis," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, (ICASSP-82)*, vol. 7, May 1982, pp. 1333–1336.

[14] T. Claasen and W. Mecklenbräuker, "The aliasing problem in discrete-time Wigner distributions," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 31, no. 5, pp. 1067–1072, 1983.

[15] B. Boashash, "Time–frequency signal analysis," in *Advances in Spectrum Estimation*, S. Haykin, Ed. Englewood Cliffs, NJ: Prentice-Hall, 1991, ch. 9, pp. 418–517.

[16] J. O' Toole and B. Boashash, "Optimisation of the realisation of quadratic discrete time-freqeuncy distributions as a MATLAB toolbox," in *Proc. Int. Conf. on Scientific and Eng. Comp.*, vol. CD-ROM, 2004.

[17] J. M. O' Toole, M. Mesbah, and B. Boashash, "A computationally efficient implementation of quadratic time–frequency distributions," in *Proc. Int. Sym. on Signal Processing and its Applications, ISSPA-07*, vol. I, Sharjah, United Arab Emirates, Feb. 12–15 2007, pp. 290–293.

[18] S.-C. Chan and K.-L. Ho, "On computing the discrete Wigner-Ville distribution," *Electron. Lett.*, vol. 26, no. 10, pp. 636–638, May10 1990.