Fast algorithm for pseudodiscrete Wigner–Ville distribution using moving discrete Hartley transform

K.M.M. Prabhu
R. Shanmuga Sundaram

Indexing terms: Pseudodiscrete Wigner–Ville distribution, Discrete Hartley transform, Hilbert transform, Fast algorithms, Time-varying signals

Abstract: A new fast algorithm is proposed to compute pseudodiscrete Wigner–Ville distribution (PDWVD) in real-time applications. The proposed algorithm uses the moving discrete Hartley transform to compute the Hilbert transform and thereby implements the PDWVD in real domain. The computational complexity of the proposed algorithm is derived and compared with the existing algorithm to compute the PDWVD.

1 Introduction

The Wigner–Ville distribution (WVD) based time-frequency representation has shown to be a viable tool in the study of time-varying signals. It has, in recent years, found applications in diverse areas such as radar, sonar, speech, seismic and biomedical data analysis [1]. The increasing number of applications of the WVD has precipitated a number of fast algorithms for computing the discrete WVD. In this paper, a new algorithm based on MDHT has been proposed for computing the PDWVD and is more suitable for real-time applications.

2 WVD theory

For a real signal $x(n)$, discrete Wigner–Ville distribution (DWVD) is defined as [1]

$$DWVD_x(n,\omega) = \sum_{m=-\infty}^{\infty} z(n+m)z^*(n-m)\exp(-j2\omega m) \tag{1}$$

where $z(n)$ is the analytic signal of $x(n)$ and * denotes complex conjugation. Eqn. 1 implies that evaluation of the DWVD is a noncausal operation, i.e. the value of the signal for all time must be known before DWVD can be evaluated as it is done for the evaluation of the Fourier transform and Hilbert transform. Such an expression does not lend itself to real-time evaluation when only a finite delay can be tolerated. To overcome this problem, we generally weight the signal z by a window function w before computing the DWVD. This windowed discrete Wigner–Ville distribution is known as the pseudodiscrete Wigner–Ville distribution (PDWVD) and may be expressed as [1]

$$PDWVD_x(n,k) = \sum_{m=-L+1}^{L-1} z(n+m)z^*(n-m)w(m)w^*(-m)\exp\left(\frac{-j2\pi km}{N}\right) \tag{2}$$

where $w(m)$ is a real symmetric window of length $N = 2L - 1$. The above equation can also be written as

$$PDWVD_x(n,k) = \sum_{m=-L+1}^{L-1} z(n+m)z^*(n-m)g(m)\exp\left(\frac{-j2\pi km}{N}\right) \tag{3}$$

where $g(m) = w(m) \times w^*(-m)$, is a positive real window of odd length $2L - 1$ and $\times$ denotes multiplication. Using the analytic signal instead of the original signal in eqn. 2 ensures that the PDWVD does not contain any aliasing terms [1]. For a real signal $x(n)$, the analytic signal $z(n)$ is defined as $z(n) = x(n) + j\hat{x}(n)$, where $\hat{x}(n)$ is known as Hilbert transform of $x(n)$ [1]. Recently, Pei and Jaw proposed a new fast algorithm to compute the Hilbert transform through fast Hartley transform (FHT) [2].

Many algorithms are available in the literature to compute PDWVD as per eqn. 2 [3–6]. Among these, the algorithm proposed by Pei and Yang [3] uses only real-domain operations and reduces the computational complexity from three complex FFTs to three real FHTs. The following five-step procedure describes the algorithm.

2.1 Pei and Yang algorithm [3]

(i) For a particular value of n, say n_1, take the samples of $x(n_1 + m)$ for m ranging from $-(L - 1)$ to $(L - 1)$ and call the sequence as $G(n_1, m)$.

(ii) Obtain the Hilbert transform of $G(n_1, m)$ with respect to m [i.e, $\hat{G}(n_1, m)$] using Pei and Jaw algorithm [2].

(iii) Form $S(n_1, m) = G(n_1, m)\{G(n_1, -m) - \hat{G}(n_1, -m)\} + \hat{G}(n_1, m)\{\hat{G}(n_1, -m) + \hat{G}(n_1, -m)\}$.

(iv) Compute the FHT of $S(n_1, m) \times g(m)$, resulting in $PDWVD_X(n_1, k)$.

(v) Repeat the above steps for all values of n.

It is clear that computational complexity of PDWVD is high because for each value of n, we need to compute three FHTs [two FHTs for computing the Hilbert transform and one FHT for step (iv)] plus some more

© IEE, 1996
IEE Proceedings online no. 19960647
Paper first received 3rd January and in revised form 14th May 1996
The authors are with the Department of Electrical Engineering, Indian Institute of Technology, Madras-600 036, India

multiplications and additions for computing $S(n_1, m)$ and the product $S(n_1, m) \times g(m)$. But, in the proposed algorithm, Hilbert transform is computed recursively with a smaller number of arithmetic operations and this reduces the computational complexity from three FHTs to a single FHT.

3 Discrete Hartley transform (DHT)

For a real signal $x(n)$, the forward DHT is defined as [7],

$$X(k) = \sum_{n=0}^{N-1} x(n)\mathrm{cas}\left(\frac{2\pi nk}{N}\right) \quad k = 0, 1, \ldots, N-1 \quad (4)$$

and the inverse DHT is given by

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\mathrm{cas}\left(\frac{2\pi nk}{N}\right) \quad n = 0, 1, \ldots, N-1 \quad (5)$$

where cas (α) = cos (α) + sin (α). Recently, the DHT has been considered as an alternative tool to DFT for spectral analysis and fast convolution of real data. It can be seen from the definitions of DHT and IDHT that one can return to the original sequence from the DHT by applying the transformation formula for a second time. This advantage of not having to go through an inverse transformation which is different from the forward one is helpful. Besides, the convenience of not having to manage the real and imaginary parts either in separate arrays or interleaved in one array of double length or in other ingenious ways that have been adopted in various embodiments of the Fourier transform, commends the DHT for consideration in many applications. Hence, the DHT is a suitable substitute for the DFT. Also many fast algorithms are available to compute the DHT and are known as fast-Hartley-transform (FHT) algorithms. The fastest FHT algorithm available to compute DHT is known as split-radix algorithm and it requires

$$\left(\frac{N}{2}\log_2 N - \frac{3N}{2} + 2\right) \text{ multiplications} \quad (6)$$

and

$$\left\{\frac{3N}{2}\log_2 N - \frac{39N}{18} + 4 + \frac{2(-1)^M}{3}\right\} \text{ additions} \quad (7)$$

to compute an N-point DHT, where $N = 2^M$ [8].

4 The proposed algorithm

The proposed algorithm is similar to the Pei and Yang algorithm for computing the PDWVD. The only difference is that in the proposed algorithm, the Hilbert transform of the signal is computed using the moving DHT instead of DHT. The idea behind the moving DHT is explained in Section 4.1.

4.1 Moving discrete Hartley transform (MDHT)

Let $X(k)$ be the N-point DHT of the data sequence $\{x(0), x(1), x(2), \ldots, x(N-1)\}$ and is already computed. Let $x_{new}(n)$ be the new data sequence obtained by moving the previous data sequence to the left by one sample point which includes the new sample, say x_{add}, and removes the old sample $x(0)$, i.e.

$$x_{new}(n) = \begin{cases} x(n+1) & n = 0, 1, 2, \ldots, N-2 \\ x_{add} & n = N-1 \end{cases} \quad (8)$$

Eqn. 8 can also be written as

$$x_{new}(n) = \begin{cases} x(n+1) & n = 0, 1, 2, \ldots, N-2 \\ x(0) + \delta\{n-(N-1)\} \\ \times\{x_{add} - x(0)\} & n = N-1 \end{cases} \quad (9)$$

where $\delta(n)$ is the discrete-time unit sample function. Taking the DHT of eqn. 9 and by applying the shift theorem of DHT [7] and simplifying, we obtain

$$\begin{aligned} X_{new}(k) =& \{X(k) + A\}\cos\left(\frac{2\pi k}{N}\right) \\ &- \{X(N-k) + A\}\sin\left(\frac{2\pi k}{N}\right) \\ & k = 0, 1, 2, \ldots, N-1 \end{aligned} \quad (10)$$

where $A = \{x_{add} - x(0)\}$. Note that the above equation gives a recursive relation between $X(k)$ and $X_{new}(k)$. If we want to compute the DHT of the next segment, which is obtained by shifting the $x_{new}(n)$ to the left which removes the sample $x(1)$ and includes x_{add1}, we can use eqn. 10 with $X(k)$ replaced by $X_{new}(k)$, $x(0)$ replaced by $x(1)$ and x_{add} by x_{add1}. When considering eqn. 10, the multiplications in the Hartley domain seem to need two multiplications per point, but it is easy to show that by grouping the computations of $X_{new}(k)$ and $X_{new}(N-k)$, this number can be reduced to (3/2) multiplications per point. The above equation can be written as

$$\begin{aligned} X_{new}(k) =& \{X(k) + A\}\cos\left(\frac{2\pi k}{N}\right) \\ &- \{X(N-k) + A\}\sin\left(\frac{2\pi k}{N}\right) \\ X_{new}(N-k) =& \{X(N-k) + A\}\cos\left(\frac{2\pi k}{N}\right) \\ &- \{X(k) + A\}\sin\left(\frac{2\pi k}{N}\right) \\ & k = 0, 1, \ldots, N/2 \end{aligned} \quad (11)$$

Eqn. 11 can be written as

$$\begin{bmatrix} X_{new}(k) \\ X_{new}(N-k) \end{bmatrix} = \begin{bmatrix} C(k) & -S(k) \\ S(k) & C(k) \end{bmatrix} \begin{bmatrix} X(k) + A \\ X(N-k) + A \end{bmatrix}$$
$$k = 0, 1, 2, \ldots, N/2 \quad (12)$$

where $C(k)$ = cos $(2\pi k/N)$ and $S(k)$ = sin $(2\pi k/N)$. Thus, eqn. 12 can be easily recognised as a complex multiplication. We know that a complex multiplication can be computed using three real multiplications and three real additions [8]. The direct implementation of eqn. 12 as a complex multiplication requires $3(N/2 + 1)$ multiplications and additions. However, when k = 0, no multiplication is required because $C(k)$ becomes one and $S(k)$ becomes 0. Similarly, when k = $N/2$ no multiplication is required. Because of this, six multiplications have been saved. In a similar way, we can calculate the number of additions as well. Also, note that for computing $X(k) + A$ for $k = 0, 1, \ldots, N-1$, we need N additions. Hence, we can obtain $X_{new}(k)$ from $X(k)$ using only $(3N/2 - 3)$ multiplications and $(5N/2 - 3)$ additions. For example, for a signal of length N = 512, the above method requires 765 multiplications and 1277 additions, whereas the direct computation of $X_{new}(k)$ using the split-radix algorithm requires 1538 multiplications and 5806 additions [8]. Note that the recursive method therefore has computational savings of 50% in terms of multiplications and 78% in terms of additions, for $N = 512$. When the input sequence is real, we can also compute the moving DFT using MDHT efficiently. Recently, Liu *et al.* proposed a real-time DHT analyser based on running DHT using a different approach [9].

4.2 Computing Hilbert transform through MDHT

Recently, Pei and Jaw [2] proposed a fast algorithm to compute the discrete Hilbert transform through fast Hartley transform. They implemented the frequency-domain definition of the Hilbert transformer and this has reduced the computational complexity from two complex FFTs to two real FHTs. For an N-point real signal $x(n)$, the discrete Hilbert transform $\hat{x}(n)$ is given by

$$\hat{x}(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(N-k)H(k)\text{cas}(2\pi nk/N)$$
$$n = 0, 1, \ldots, N-1 \qquad (13)$$

where $X(k)$ is the N-point DHT of $x(n)$ and

$$H(k) = \begin{cases} 1 & k = 1, 2, \ldots, (N/2) - 1 \\ -1 & k = (N/2) + 1, \ldots, N-1 \\ 0 & k = 0, N/2 \end{cases}$$

where cas (x) = cos (x) + sin (x). Note that eqn. 13 requires one N-point forward DHT and one inverse DHT to compute the discrete Hilbert transform of the given signal. We call this method the direct method of computing the discrete Hilbert transform. In the above equation, if we replace both the DHTs with MDHT, the computational complexity can be reduced and this is the main idea behind this method. Following eqn. 13, the discrete Hilbert transform of $x_{new}(n)$ is given by

$$\hat{x}_{new}(n) = \frac{1}{N}\sum_{k=0}^{N-1} X_{new}(N-k)H(k)\text{cas}(2\pi nk/N) \quad (14)$$

Substituting $X_{new}(k)$ from eqn. 10 into eqn. 14 we obtain

$$\begin{aligned}\hat{x}_{new}(n) =& \frac{1}{N}\sum_{k=0}^{N-1} \{X(N-k)\cos(2\pi k/N) \\ &+ X(k)\sin(2\pi k/N)\}\, H(k)\text{cas}(2\pi nk/N) \\ &+ \left(\frac{1}{N}\right)\{x_{add} - x(0)\} \\ &\times \sum_{k=0}^{N-1} \text{cas}(2\pi k/N)H(k)\text{cas}(2\pi nk/N) \quad (15)\end{aligned}$$

Comparing the first term of eqn. 15 with eqn. 13 and applying the shift theorem of the DHT, the above equation can be written as

$$\hat{x}_{new}(n) = \hat{x}(n+1) + (1/N)\{x_{add} - x(0)\}A(n)$$

where $A(n)$ is given by

$$A(n) = \sum_{k=0}^{N-1} \text{cas}(2\pi k/N)H(k)\text{cas}(2\pi nk/N)$$

By applying the identity

$$\text{cas}(\alpha)\text{cas}(\beta) = \text{cas}(\alpha+\beta) + 2\sin(\alpha)\sin(\beta)$$

and noting that $H(k)$ is an odd function of k, it can be shown that

$$A(n) = \begin{cases} 0 & n = 1, 3, \ldots, N-1 \\ 2\cot(\pi(n+1)/N) & n = 0, 2, \ldots, N-2 \end{cases}$$

Therefore

$$\hat{x}_{new}(n) = \begin{cases} \hat{x}(n+1) & n = 1, 3, \ldots, N-1 \\ \hat{x}(n+1) & \\ +\frac{2}{N}\{x_{add} - x(0)\} & \\ \times\cot(\pi(n+1)/N) & n = 0, 2, \ldots, N-2 \end{cases} \quad (16)$$

Using the above equation, we can compute the Hilbert transform of $x_{new}(n)$ from the Hilbert transform of $x(n)$ by using only $N/2$ multiplications and $N/2 + 1$ additions. However, note that to compute $\hat{x}_{new}(0)$ and $\hat{x}_{new}(N-2)$ only multiplication by cot $(\pi n/N)$ is required. Note that the multiplication by $2/N$ can be implemented as a right shift and hence it is not considered as a multiplication while calculating the computational complexity. Hence, the required number of multiplications can be reduced from $N/2$ to $N/4$. If we use MDHT to compute the Hilbert transform of the given signal, the computational complexity reduces from $N(\log_2 N - 3) + 4$ real multiplications to $N/4$ real multiplications. Similarly, additions are reduced from $3N\log_2 N - (39N/9) + 8 + (4/3)(-1)^{\log_2 N}$ to $(N/2) + 1$ assuming that the split-radix FHT has been used to compute the two DHTs in the direct method. For $N = 64$, the direct method requires 196 multiplications and 884 additions whereas the recursive method requires 16 multiplications and 33 additions. Hence, the recursive method has computational savings of 92% for multiplications and 96% for additions. As such, this algorithm can be used to compute the Hilbert transform of a signal very efficiently. Systems for performing Hilbert transform operations have been proven useful in such diverse fields as radar moving-target detectors, analytic signal rooting, envelope detection and generation of the phase of a spectrum from its amplitude.

4.3 Computing PDWVD through MDHT

Since the computation of PDWVD requires the analytic signal of the original signal and the analytic signal can be obtained from the Hilbert transform of the original signal, we can also apply the above algorithm to compute PDWVD. This reduces the computational complexity. Note that the above recursive relation for Hilbert transform holds good only when the new data is obtained by shifting the old data by one sample to the left. In most of the applications, computing PDWVD for each value of n (i.e. time resolution = 1) may not be necessary. We therefore have to modify eqn. 16 for a greater number of shifts between the successive computations.

Now consider the case of shifting $x(n)$ by two samples to the left with x_{add1}, x_{add2} being the new samples added and calling the newly obtained sequence as $x_{new}(n)$, it can be shown that

$$\hat{x}_{new}(n) = \begin{cases} \hat{x}(n+2) & \\ +\frac{2}{N}[x_{add1} - x(0)] & \\ \times\cot(\pi(n+2)/N) & n = 1, 3, \ldots, N-1 \\ \hat{x}(n+2) & \\ +\frac{2}{N}[x_{add2} - x(1)] & \\ \times\cot(\pi(n+1)/N) & n = 0, 2, \ldots, N-2 \end{cases}$$

The above equation requires $2\{(N/2) + 1\}$ additions and $2(N/4)$ multiplications to compute $\hat{x}_{new}(n)$ from $\hat{x}(n)$. It can be shown that, for L shifts between the successive computations, we need $L(N/4)$ multiplications and $L\{(N/2) + 1\}$ additions to obtain $\hat{x}_{new}(n)$ from $\hat{x}(n)$. In this way we can obtain the Hilbert transform of the signal efficiently. The other steps to compute PDWVD are the same as outlined in the Pei and Yang algorithm [3].

4.4 Comparison of computation time

If we use the split-radix algorithm to compute DHT, the proposed algorithm requires

$$\left\{\frac{N}{2}(\log N - 3) + 2\right\} + 2N + L(N/4) \text{ multiplications}$$

$$\left\{\frac{3N}{2}\log_2 N - \frac{39N}{18} + 4 + \frac{2}{3}(-1)^{\log_2 M}\right\}$$
$$+ 2N + L\{(N/2) + 1\} \text{ additions}$$

for a shift of L samples between successive computations. In the multiplications count, the first term corresponds to computing the FHT using split-radix algorithm and the second term corresponds to computing $S(n, m)$ and $S(n, m) \times g(m)$. Note that to compute $S(n, m)$ only N multiplications are required, because it is the sum of the even part and the odd part. The last term in the multiplication count corresponds to computing the Hilbert transform. Similarly, the first term in the additions count corresponds to FHT computation and the second term corresponds to formulation of $S(n, m)$. However, the algorithm proposed by Pei and Yang requires 3 $\{(N/2)(\log_2 N - 3) + 2\} + 2N$ multiplications and 3 $\{(3N/2)\log_2 N - (39N/18) + 4 + (2/3)(-1)^{\log_2 M}\} + 2N$ additions [The arithmetic count has been changed to take into the effect of multiplying by $g(m)$].

The proposed algorithm is computationally better than the Pei and Yang algorithm only if L satisfies the relation

$$L < (4\log_2 N - 12)$$

For a window of length 64 (i.e. $N = 64$), the proposed algorithm is better than the Pei and Yang algorithm only when the number of shifts between the successive PDWVD computations are less than 12. This may be enough for the real-time applications. For example, consider a signal of length 512 whose PDWVD has to be computed with a time resolution of 4 ($L = 4$) using a 64-point rectangular window ($N = 64$). By using the operation counts given above, it can be shown that the proposed algorithm requires 28928 multiplications and 89856 additions to compute the PDWVD, whereas the Pei and Yang algorithm requires 45824 multiplications and 186112 additions. Hence, the proposed algorithm has computational savings of 16896 multiplications (37%) and 96256 additions (52%). As discussed earlier, in some applications it may be necessary to compute PDWVD for each value of n. In this case, the proposed algorithm has computational savings of 92160 (50%) multiplications and 435712 (58%) additions. Even though the split-radix FHT algorithm is more efficient than the radix-2 and the radix-4 algorithms, it may not be possible to implement it efficiently on some machines such as DSP processors [10]. In such cases, it is necessary to use either the radix-2 FHT algorithm or radix-4 FHT algorithm to compute the PDWVD. It can be shown that the proposed algorithm will give more computational reductions in these cases. The only disadvantage of the proposed algorithm is that its error characteristics are less favourable due to the factor cot $\pi(n + 1)/N\}$. As N increases, the value of this factor increases approximately linearly. Hence, on fixed-point processors this algorithm requires a lot of scaling for large values of N. However, this algorithm is suitable on floating-point machines where the dynamic-range problems are less.

5 Conclusion

In this paper, we present a new fast algorithm for computing the PDWVD using the moving discrete Hartley transform. Compared with the existing method, the computational complexity of the proposed algorithm is reduced from three FHTs to a single FHT with a few more arithmetic operations. For a window of length 64, computational savings of 50% for multiplications and 58% for additions can be obtained compared with the existing algorithm.

6 Acknowledgments

The authors thank the anonymous reviewers for their suggestions and constructive criticisms which considerably improved the presentation of the paper.

7 References

1 BOASHASH, B.: 'Time–frequency signal analysis' *in* HAYKIN, S. (Ed.): 'Advances in spectrum analysis and array processing' (Prentice–Hall, New Jersey, 1991), vol. 1. Chap. 9, pp. 418–517

2 PEI, S.C., and JAW, S.B.: 'Computation of discrete Hilbert transform through fast Hartley transform', *IEEE Trans.*, 1989, **CAS-36**, (9), pp. 1251–1252

3 PEI, S.C., and YANG, I.I.: 'Computing pseudo-Wigner distribution by the fast Hartley transform', *IEEE Trans.*, 1992, **SP-40**, (9), pp. 2346–2349

4 BOASHASH, B., and BLACK, P.J.: 'An efficient real-time implementation of the Wigner–Ville distribution', *IEEE Trans.*, 1987, **ASSP-35**, (11), pp. 1611–1618

5 SUN, M., LI, C.C., SEKHAR, L.N., and SCLABASSI, R.J.: 'Efficient computation of the discrete pseudo-Wigner distribution', *IEEE Trans.*, 1989, **ASSP-37**, (11), pp. 1735–1742

6 NARAYANAN, S.B., and PRABHU, K.M.M.: 'New method of computing Wigner–Ville distribution', *Electron. Lett.*, 1989, **25**, (5), pp. 336–338

7 BRACEWELL, R.N.: 'The fast Hartley transform', *Proc. IEEE*, 1984, **72**, (8), pp. 1010–1018

8 MALVAR, H.S.: 'Signal processing with lapped transforms' (Artech House, 1992), Chap. 2, pp. 31–80

9 LIU, J.C., and LIN, T.P.: 'Running DHT and real-time DHT analyser', *Electron. Lett.*, 1988, **24**, (12), pp. 762–763

10 SURENDRA KUMAR, B., RAGHU RAMA KRISHNA, S., and PRABHU, K.M.M.: 'Fast Hartley transform implementation on DSP chips'. Proceedings of international conference on *Signal processing applications and technology*, Santa Clara, USA, 1993, pp. 478–485