sensors

MDPI

Letter

Fast Recursive Computation of Sliding DHT with Arbitrary Step

Vitaly Kober [1,2,3]

[1] Department of Computer Science, CICESE, Ensenada 22860, Mexico; vkober@hotmail.com
[2] Department of Mathematics, Chelyabinsk State University, Chelyabinsk 454001, Russia
[3] Institute for Information Transmission Problems, Russian Academy of Sciences, Moscow 127051, Russia

Received: 25 August 2020; Accepted: 26 September 2020; Published: 28 September 2020

Abstract: Short-time (sliding) transform based on discrete Hartley transform (DHT) is often used to estimate the power spectrum of a quasi-stationary process such as speech, audio, radar, communication, and biomedical signals. Sliding transform calculates the transform coefficients of the signal in a fixed-size moving window. In order to speed up the spectral analysis of signals with slowly changing spectra, the window can slide along the signal with a step of more than one. A fast algorithm for computing the discrete Hartley transform in windows that are equidistant from each other is proposed. The algorithm is based on a second-order recursive relation between subsequent equidistant local transform spectra. The performance of the proposed algorithm with respect to computational complexity is compared with the performance of known fast Hartley transform and sliding algorithms.

Keywords: discrete Hartley transform; sliding algorithm; sensor noise removal; signal processing; short-time transform

1. Introduction

In today's world, the study of real-life phenomena begins with sensors that convert the description of phenomena into digital signals. Since real sensors are sensitive to internal thermal, electronic, and environmental noise, the sensors' output must be intelligently processed for subsequent use in a computer system. Short-time signal processing [1] is an appropriate technique for such a system, since it is capable of adaptively filtering long-term stationary (in wide-sense) and quasi-stationary signals. An example of quasi-stationary signal is speech. Frequency components of this signal change over time when the position of tongue and mouth changes, but on a short period of time they might be considered stable because tongue cannot move very quickly. Short-time processing is used in a variety of applications such as radar emitter recognition [2], heart sound classification [3], sensor noise removal [4], gearbox fault diagnosis [5], speech processing and spectral analysis [6], and adaptive linear filtering [7].

Basically, short-time transform is a time series of windowed signal transforms that can be used to perform time-frequency analysis. Local signal processing in a moving window (special case of short-time processing) can be carried out as follows: at each position of the running window, the coefficients of an orthogonal transform of the signal are modified to estimate only the central window element [8]. For time-varying signals, it is preferred that the window length be small enough so that the windowed signal would be approximately stationary. On the other hand, the window size should be large enough to provide a reasonable frequency resolution of the local signal. Typically, with local sliding processing, the window moves with a step of one along the signal. However, if it is necessary to process the signal at a high speed and the signal spectrum changes slowly over time,

the window can move with a step of more than one. This approach has recently been proposed to design preview tools for multiresolution signal and image restoration [9].

Discrete Hartley transform (DHT) [10] is used in various practical applications such as speech spectral analysis [11], feature extraction and sea surface modeling [12], data compression [13], and signal interpolation [14]. Four types of DHT are classified [15,16]. Discrete Hartley and Fourier transforms of real signals and their relations with discrete sinusoidal transformations are discussed in detail [17,18]. Note that for real signals, all DHT coefficients can be used in the design of transform domain scalar filters. Representation of a signal in the DHT sliding domain is especially suitable for processing of time-varying, quasi-stationary data. Since calculating the DHT at each position of a running window is an intensive task, recursive algorithms can be used. First-order shift properties were obtained [19]. Since this approach is not very efficient in terms of computational complexity, four types of the sliding DHT were derived based on second-order recursive equations [20].

In this work, a recursive algorithm is proposed for fast computation of the DHT in a window sliding along the signal more than one sample at a time. The main contributions of the work are as follows:

- Second-order recursive equation between three consecutive equidistant DHT spectra is obtained using the z-transform technique.
- Efficient sliding DHT algorithm is proposed using the properties of discrete sinusoidal functions and the recursive equation.
- Computational complexity and running time of the proposed sliding algorithm are compared with the known sliding and fast DHT algorithms.

The paper is organized as follows. In Section 2, a second-order recursive equation between three consecutive equidistant DHT spectra is derived. In Section 3, a fast forward algorithm for computing the sliding DHT is suggested. Section 3 also presents a fast inverse sliding DHT algorithm. The performance of the proposed and common DHT algorithms in terms of computational complexity and runtime is given and discussed in Section 4. Finally, our conclusions are presented in Section 5.

2. Second-Order Equation for Recursive Computation of Sliding DHT

We recall here the definition of DHT. The following notations are used: $cs_N(rs) \equiv \cos\left(\frac{2\pi}{N}rs\right)$, $sn_N(rs) \equiv \sin\left(\frac{2\pi}{N}rs\right)$, $snc_N(rs) \equiv \frac{sn_N(rs)}{sn_N(s)}$ $\mathrm{cas}_N(rs) \equiv \cos\left(\frac{2\pi}{N}rs\right) + \sin\left(\frac{2\pi}{N}rs\right)$, $\overline{\mathrm{cas}_N}(rs) \equiv \cos\left(\frac{2\pi}{N}rs\right) - \sin\left(\frac{2\pi}{N}rs\right)$, $W_N^s \equiv e^{i\frac{2\pi}{N}s} = cs_N(s) + isn_N(s)$, where the subscript N is the transform order, r is an integer, and $s = 0, 1, \ldots N-1$. For clarity, the normalization factor $1/\sqrt{N}$ is disregarded until the inverse transform.

Sliding DHT with a step of p can be defined as follows:

$$y_s(kp) = \sum_{n=-N_1}^{N_2} x(kp+n)\mathrm{cas}_N(s(n+N_1)), \tag{1}$$

where $\{x(kp);\ k = \ldots, -N_1, -N_1+1, \ldots, 0, 1, \ldots, N_2, N_2+1 \ldots.\}$ is a signal to be processed; $\{y_s(kp),\ s = 0, 1, \ldots N-1\}$ are the transform coefficients around time kp; $N = N_1 + N_2 + 1$ is the size of the moving window; N_1 and N_2 are arbitrary integer values.

A recursive relationship between three consecutive sliding spectra is given as [20]:

$$y_s(k+2) - 2\mathrm{cs}_N(s)y_s(k+1) + y_s(k) = f_s(k), \tag{2}$$

where $f_s(k) = (x(k+N_2+2) - x(k-N_1+1))\overline{\mathrm{cas}_N}(s) + x(k-N_1) - x(k+N_2+1)$.

This is a linear inhomogeneous difference equation defined on k. For fixed s, (2) can be considered as a linear difference equation with constant coefficients. Linear time-invariant systems defined by such equations can be analyzed using the unilateral z-transform [1]. Suppose that we deal with a

causal linear system (region of convergence is outside a circle on the z-plane) and nonzero initial conditions $y_s(0)$ and $y_s(1)$ are given. Applying the unilateral z-transform to (2) and using its shift property, we get:

$$z^2\Big[Y_s(z) - y_s(0) - z^{-1}y_s(1)\Big] - 2\mathrm{cs}_N(s)z[Y_s(z) - y_s(0)] + Y_s(z) = F_s(z), \tag{3}$$

where $Y_s(z)$ and $F_s(z)$ are the z-transforms of $y_s(k)$ and $f_s(k)$, respectively.

We express $Y_s(z)$ as:

$$Y_s(z) = \frac{F_s(z) + y_s(0)z(z - 2\mathrm{cs}_N(s)) + y_s(1)z}{z^2 - 2\mathrm{cs}_N(s)z + 1}. \tag{4}$$

For $s \neq 0$, there are two different roots of the denominator; that is, $q_1(s) = W_N^s$ and $q_2(s) = W_N^{-s}$. For $s = 0$, the roots are equal to $q_1(s) = q_2(s) = 1$. Let us carry out the partial fraction expansion of the equation:

$$D_s(z) = \frac{z}{z^2 - 2\mathrm{cs}_N(s)z + 1} = \begin{cases} \frac{z^{-1}}{1-2\mathrm{cs}_N(s)z^{-1}+z^{-2}}, s \neq 0 \\ \frac{z^{-1}}{1-2z^{-1}+z^{-2}}, s = 0 \end{cases}, \tag{5}$$

and obtain

$$D_s(z) = \begin{cases} \frac{1}{q_1(s)-q_2(s)}\left(\frac{1}{1-q_1(s)_1z^{-1}} - \frac{1}{1-q_2(s)z^{-1}}\right), s \neq 0 \\ \frac{z^{-1}}{(1-z^{-1})^2}, s = 0 \end{cases}. \tag{6}$$

The inverse z-transform can be computed as follows [1]:

$$D_s(z) \overset{Z^{-1}}{\rightarrow} d_s(k) = \begin{cases} \frac{q_1^k(s)-q_2^k(s)}{q_1(s)-q_2(s)}u(k) = \frac{sn_N(ks)}{sn_N(s)}u(k), & s \neq 0 \\ ku(k), & s = 0 \end{cases}, \tag{7}$$

where $u(k)$ is the unit-step function defined as 1, for $k \geq 0$ and 0, for $k < 0$.

Using (5), (4) can be represented as follows:

$$Y_s(z) = F_s(z)\Big[z^{-1}D_s(z)\Big] + y_s(0)[zD_s(z) - 2\mathrm{cs}_N(s)D_s(z)] + y_s(1)D_s(z). \tag{8}$$

Using the shifting and convolution properties, the inverse transform is given as:

$$y_s(k) = \sum_{i=0}^{k} d_s(i-1)f_s(k-i) + y_s(0)[d_s(k+1) - 2\mathrm{cs}_N(s)d_s(k)] + y_s(1)d_s(k). \tag{9}$$

Substituting (7) into (9) and taking into account that $d_s(k) = 0$ for $k < 2$, $q_1(s)q_1(s) = 1$ and $2\mathrm{cs}_N(s) = q_1(s) + q_2(s)$, we arrive at:

$$y_s(k) = \begin{cases} \sum\limits_{r=1}^{k-1} f_s(k-r-1)r - y_s(0)(k-1) + y_s(1)k, \; s = 0 \\ \frac{\sum\limits_{r=1}^{k-1} sn_N(rs)f_s(k-r-1) - y_s(0)sn_N((k-1)s) + y_s(1)sn_N(ks)}{sn_N(s)}, s = 1, \ldots N-1 \end{cases}, \tag{10}$$

with $k \geq 2$.

Given two initial values $y_s(0)$ and $y_s(p)$ $(p > 1)$, one can obtain $y_s(1)$ from (10) as:

$$y_s(1) = \begin{cases} \frac{y_s(p)+y_s(0)(p-1)-\sum_{r=1}^{p-1} f_s(p-r-1)r}{p}, & s = 0 \\ \frac{y_s(p)sn_N(s)+y(0)sn_N((p-1)s)-\sum_{r=1}^{p-1} f(p-r-1)sn_N(rs)}{sn_N(ps)}, & s = 1,\ldots N-1 \end{cases}. \quad (11)$$

Substituting (11) into (10), for arbitrary k we obtain:

$$y_s(k) = \begin{cases} \frac{k}{p}y_s(p) + \frac{k-p}{p}\left(\sum_{r=1}^{p-1} f_s(r-1)r - y_s(0)\right) + \sum_{r=p}^{k-1} f_s(r-1)(k-r), & s = 0 \\ \frac{sn_N((k-p)s)}{sn_N(ps)}\left(\sum_{r=1}^{p-1} f_s(r-1)\frac{sn_N(rs)}{sn_N(s)} - y_s(0)\right) + \frac{sn_N(ks)}{sn_N(ps)}y_s(p) + \sum_{r=p}^{k-1} f_s(r-1)\frac{sn_N((k-r)s)}{sn_N(s)}, & s = 1,\ldots N-1 \end{cases}. \quad (12)$$

Finally, the DHT at the window position $2p$ is recursively computed using $y_s(0)$ and $y_s(p)$ as:

$$y_s(2p) = \begin{cases} \sum_{r=1}^{p-1} (f_s(r-1) + f_s(2p-r-1))r - y_s(0) + 2y_s(p) + f_s(p-1)p, & s = 0 \\ \sum_{r=1}^{p-1} (f_s(r-1) + f_s(2p-r-1))\frac{sn_N(rs)}{sn_N(s)} - y_s(0) + 2cs_N(ps)y_s(p) + f_s(p-1)\frac{sn_N(ps)}{sn_N(s)}, & s = 1,\ldots N-1 \end{cases}. \quad (13)$$

This equation gives relationship between three consecutive equidistant DHT spectra, computed at times 0, p, and $2p$.

3. Fast Sliding Algorithm for Computing DHT

3.1. Special Values of Discrete Sinusoidal Functions

Given below are special values $(0,\ 1, -1)$ of discrete sinusoidal functions used in the performance analysis of the proposed sliding algorithm. Let us consider the following functions: $cs_N(rs)$, $snc_N(rs)$, and $\overline{cas_N}(s)$ with $s = 0, 1, \ldots, N-1$. The special values are shown in Table 1. Here, r, l, and N are arbitrary integer values, and the binary variable b takes the values $\{0, 1\}$. So, it is necessary to find such values of the variables l and b in order to obtain integer values of s in the range from 0 to $N-1$.

Table 1. Special values of the discrete sinusoidal functions.

Functions	**Values**		
	0	**1**	**−1**
$cs_N(rs)$	$s = \frac{N}{4r}(2l+1)$	$s = \frac{N}{r}l$	$s = \frac{N}{2r}(2l+1)$
$snc_N(rs)$	$s = \frac{N}{2r}l; s \neq 0, \frac{N}{2}$	$s = \frac{N}{2}\frac{(2l+b)}{(r+2b-1)}$	$s = \frac{N}{2}\frac{(2l+b)}{(r-2b+1)}$
$\overline{cas_N}(s)$	$s = \frac{N}{8}(4l+1)$	$s = 0; s = \frac{3N}{4}$	$s = \frac{N}{4}; s = \frac{N}{2}$

The functions are used in the proposed algorithm in the operations of addition and multiplication; that is, when the functions are equal to 0, then the corresponding addition and multiplication operations can be discarded, and when they are equal to either −1 or 1, then the corresponding multiplication operations can be canceled. Therefore, the use of the special values can reduce the computational complexity of the algorithm.

3.2. Design of Forward Sliding Algorithm

Equation (13) for $p > 1$ can be rewritten as:

$$y_s(2p) = \begin{cases} \sum_{r=1}^{p} (B(r) + A(r))r - y_s(0) + 2y_s(p), & s = 0 \\ \overline{\cos_N}(s)\left(B(1) + \sum_{r=2}^{p} B(r)snc_N(rs)\right) + \left(A(1) + \sum_{r=2}^{p} A(r)snc_N(rs)\right) - y_s(0) + 2cs_N(ps)y_s(p), & s = 1,\ldots N-1 \end{cases}, \quad (14)$$

where $\{B(r) = \Delta(r) + \Delta(2p-r); A(r) = -\Delta(r-1) - \Delta(2p-r-1);\ B(p) = \Delta(p);\ A(p) = -\Delta(p-1); \Delta(r) = x(r+N_2+1) - x(r-N_1); r = 1,\dots p-1\}$. At each window, the number of additions required for calculation of $\{B(r), A(r); r = 1,\dots p\}$ is $3p-2$, since the coefficients $\{\Delta(r); r = 0,\dots p-1\}$ have already been computed and stored at the position p of the window.

Suppose that N is odd. Let g_r, h_r^+, h_r^- be the greatest common factors of N and r, $r+1$, $r-1$, respectively. The property of symmetry of the discrete function $snc_N(rs)$ is given as follows:$\left\{snc_N(rs) = snc_N(r(N-s))\ ;\ s = 1,\dots \frac{N-1}{2},\ r = 2,\dots p\right\}$. Let us compute $h_r = MAX\left(h_r^+, h_r^-\right)$. For a fixed r, the quantities of 0 and ±1 of $\left\{snc_N(rs)\ ;\ s = 1,\dots \frac{N-1}{2},\ r = 2,\dots p\right\}$ are equal to $\frac{g_r-1}{2}$ and $\frac{h_r-1}{2}$, respectively (see the second row of Table 1). So, the number of multiplications can be estimated as:

$$C_{MUL} = N - 1 + 2\left((p-1)\frac{N+1}{2} - \sum_{r=2}^{p}\frac{g_r + h_r - 2}{2}\right) + N = N(p+1) + 3p - \sum_{r=2}^{p}(g_r + h_r) - 4. \quad (15)$$

Let us denote $SUM_Q(s) \equiv \sum_{r=1}^{p} Q(r) snc_N(rs)$. Using the symmetry property of the functions $\left\{SUM_A(s) = SUM_A(N-s), SUM_B(s) = SUM_B(N-s); s = 1,\dots \frac{N-1}{2}\right\}$, the number of additions can be estimated as:

$$C_{ADD} = 2\left((p-1)\frac{N+1}{2} - \sum_{r=2}^{p}\frac{g_r - 1}{2}\right) + 3N + 3p - 2 = N(p+2) + 5p - \sum_{r=2}^{p} g_r - 4. \quad (16)$$

Additional expenses are required for calculating initial p coefficients.

Suppose that N is even. Equation (14) can be represented as follows:

$$y_s(2p) = \begin{cases} \sum_{r=1}^{p}(B(r) + A(r))r - y_s(0) + 2y_s(p), & s = 0 \\ \sum_{r=1}^{p}(-B(r) + A(r))(-1)^{r-1}r - y_s(0) + 2(-1)^p y_s(p), & s = \frac{N}{2} \\ \overline{\cos_N}(s)\left(B(1) + \sum_{r=2}^{p} B(r) snc_N(rs)\right) + \left(A(1) + \sum_{r=2}^{p} A(r) snc_N(rs)\right) - y_s(0) + 2cs_N(ps) y_s(p), & s = 1,\dots,N/2-1, N/2+1,\dots,N-1 \end{cases}. \quad (17)$$

Let $\overline{g}_r, \overline{h}_r^+$, $\overline{h}_r^-$ be the greatest common factors of $\frac{N}{2}$ and r, $r+1$, $r-1$, respectively. In this case, the property of symmetry of the discrete function $snc_N(rs)$ is given as follows:$\left\{snc_N(rs) = snc_N(r(N-s)) = (-1)^{r-1} snc_N\left(r\left(\frac{N}{2} \pm s\right)\right), snc_N\left(r\frac{N}{2}\right) = (-1)^{r-1} r\ ;\ s = 0,\dots \left[\frac{N}{4}\right],\ r = 2,\dots p\right\}$, here $[x/y]$ is the integer quotient. Let us compute $\overline{h}_r = MAX\left(\overline{h}_r^+, \overline{h}_r^-\right)$. For a fixed r, the quantities of 0 and ±1 of $\left\{snc_N(rs)\ ;\ s = 1,\dots \left[\frac{N}{4}\right],\ r = 2,\dots p\right\}$ are equal to $\left[\frac{\overline{g}_r}{2}\right]$ and $\left[\frac{\overline{h}_r}{2}\right]$, respectively (see second row of Table 1). The number of zeros of $\{cs_N(ps)\ ;\ s = 1,\dots N-1\}$ is equal to $2g_p$ (see the first row of Table 1). If $N/8$ is integer (see the third row of Table 1), then the number of zeros of the function $\{\overline{cas_N}(s)\ ;\ s = 1,\dots N-1\}$ equals $C_{cas} = 2$, otherwise $C_{cas} = 0$. It means that the zeros of the function can cancel the operations of multiplication and addition in the first term of (17) (see the last line of the equation) at the corresponding frequencies s. The number of ±1 of the function $\{\overline{cas_N}(s)\ ;\ s = 0,\dots N-1\}$ is equal to $2C_{cas}$. Finally, the number of multiplications can be estimated as follows:

$$\begin{aligned} C_{MUL} = N - 2 + 2\left((p-1)\left(\left[\frac{N}{4}\right] + 1 - \frac{C_{cas}}{2}\right) - \sum_{r=2}^{p}\left(\left[\frac{\overline{g}_r}{2}\right] + \left[\frac{\overline{h}_r}{2}\right]\right)\right) + N - 2g_p - 2C_{cas} = \\ 2\left(N + (p-1)\left[\frac{N}{4}\right] - \sum_{r=2}^{p}\left(\left[\frac{\overline{g}_r}{2}\right] + \left[\frac{\overline{h}_r}{2}\right]\right) + p - g_p - 2\right) - (p+1)C_{cas}. \end{aligned} \quad (18)$$

Using the symmetry property of the functions $\{SUM_A(s) = SUM_A(N-s)$, $SUM_B(s) = SUM_B(N-s); s = 1, \ldots \frac{N}{2} - 1\}$, the number of additions can be estimated as:

$$C_{ADD} = 3N - 1 + 2\left((p-1)\left(\frac{N}{2} + 1 - \frac{C_{cas}}{2}\right) - \sum_{r=2}^{p}\left[\frac{\bar{g}_r}{2}\right]\right) - 2g_p + 3p - 2 = N(p+2) - 2\sum_{r=2}^{p}\left[\frac{\bar{g}_r}{2}\right] - 2g_p + (p-1)(5 - C_{cas}). \quad (19)$$

Additional expenses are needed for calculating initial p coefficients.

To more clearly show the design of the algorithm, we provide an example for computing the sliding DHT coefficients. Assume that $N_1 = 7$, $N_2 = 8$, $N = 16$, $p = 2$, the DHT is computed at the window position $2p$ $\{y_s(2p)$, $s = 0, 1, ...15\}$. We borrow from the window position p two coefficients $\triangle_0 = x_9 - x_{-7}$; $\triangle_1 = x_{10} - x_{-6}$, and calculate the following auxiliary data:

$\triangle_2 = x_{11} - x_{-5}; \triangle_3 = x_{12} - x_{-4}$

$A_1 = \Delta_0 + \Delta_2$; $B_1 = \Delta_1 + \Delta_3$

$S_0 = 2\Delta_2; S_1 = 1.84776\Delta_2; S_2 = 1.41421\Delta_2; S_3 = 0.76537\Delta_2$

$C_0 = 2\Delta_1; C_1 = 1.84776\Delta_1; C_2 = 1.41421\Delta_1; C_3 = 0.76537\Delta_1$

$Q_0^{\pm} = A_1 \pm C_0; Q_1^{\pm} = A_1 \pm C_1; Q_2^{\pm} = A_1 \pm C_2; Q_3^{\pm} = A_1 \pm C_3$

$M_0^{\pm} = B_1 \pm S_0; M_1^{\pm} = B_1 \pm S_1; M_2 = B_1 - S_2; M_3^{\pm} = B_1 \pm S_3$

$F_1 = 0.5412M_1^+; F_3 = -0.5412M_3^+; F_5 = -1.30656M_3^-$

$F_6 = -1.41421M_2; F_7 = -1.30656M_1^-$.

DHT coefficients are computed as follows:

$y_0(2p) = M_0^+ - Q_0^+ - y_0(0) + 2y_0(p)$

$y_1(2p) = F_1 - Q_1^+ - y_1(0) + 1.41421y_1(p)$

$y_2(2p) = -Q_2^+ - y_2(0)$

$y_3(2p) = F_3 - Q_3^+ - y_3(0) - 1.41421y_3(p)$

$y_4(2p) = -B_1 - A_1 - y_4(0) - 2y_4(p)$

$y_5(2p) = F_5 - Q_3^- - y_5(0) - 1.41421y_5(p)$

$y_6(2p) = F_6 - Q_2^- - y_6(0)$

$y_7(2p) = F_7 - Q_1^- - y_7(0) + 1.41421y_7(p)$

$y_8(2p) = -M_0^- - Q_0^- - y_8(0) + 2y_8(p)$

$y_9(2p) = -F_1 - Q_1^- - y_9(0) + 1.41421y_9(p)$

$y_{10}(2p) = -Q_2^- - y_{10}(0)$

$y_{11}(2p) = -F_3 - Q_3^- - y_{11}(0) - 1.41421y_{11}(p)$

$y_{12}(2p) = B_1 - A_1 - y_{12}(0) - 2y_{12}(p)$

$y_{13}(2p) = -F_5 - Q_3^+ - y_{13}(0) - 1.41421y_{13}(p)$

$y_{14}(2p) = -F_6 - Q_2^+ - y_{14}(0)$

$y_{15}(2p) = -F_7 - Q_1^+ - y_{15}(0) + 1.41421y_{15}(p)$.

The computational complexity of the algorithm is 25 multiplications and 61 additions.

3.3. Design of Inverse Sliding Algorithm

The inverse sliding DHT computes only the element $x(kp)$ of the window. The inverse algorithm can be written as follows:

$$x(kp) = \frac{1}{N}\sum_{s=0}^{N-1} y_s(kp) cas_N(N_1 s). \quad (20)$$

where $N = N_1 + N_2 + 1$.

If $x(kp)$ is the central window element, that is, $N = 2N_1$, then one can simplify the inverse transform to:

$$x(kp) = \frac{1}{N}\sum_{s=0}^{N-1} y_s(kp)(-1)^s. \quad (21)$$

Finally, for $N_1 = 0$ the inverse transform is given as:

$$x(kp) = \frac{1}{N}\sum_{s=0}^{N-1} y_s(kp). \tag{22}$$

The proposed algorithms require only one multiplication and $N - 1$ additions.

4. Results and Discussion

In this section, using computer simulation, we analyze the performance of the proposed algorithm with respect to the computational complexity and execution time and compare it with that of fast DHT and conventional recursive algorithms. Among fast DHT algorithms, the most popular are fast radix-2 [21,22]. The recursive sliding (with a step of one) DHT algorithms [19,20] are carried out p times for computing the DHT spectra at equidistant positions kp of the window. Tables 2 and 3 show the computational complexity in terms of multiplications and additions, respectively, of the proposed, fast radix-2 DHT, and known sliding algorithms at a fixed window position for $p = 2$ when N varies.

Table 2. Comparison of the tested algorithms with respect to multiplications for computing sliding DHT with $p = 2$.

Algorithms	**Number of Operations, $N = 2^M$**	***N*-Length of Sliding Window**					
		16	**32**	**64**	**128**	**256**	**512**
Fast DHT [22]	$(M-3)N/2+2$	10	34	98	258	642	1538
Ref. [19]	$(2N-1)p$	62	126	254	510	1022	2046
Ref. [20]	$(5/4(N-4)-1)p$	28	68	148	308	628	1268
Proposed ALG	Equation (18)	25	68	148	308	628	1268

Table 3. Comparison of the tested algorithms with respect to additions for computing sliding DHT with $p = 2$.

Algorithms	**Number of Operations, $N = 2^M$**	***N*-Length of Sliding Window**					
		16	**32**	**64**	**128**	**256**	**512**
Fast DHT [22]	$(M+9)N/2\text{-}M^2 - \text{b}3M - 6$	70	178	420	948	2082	4494
Ref. [19]	$(3N-1)p$	94	190	382	766	1534	3070
Ref. [20]	$(5N/2+2)p$	84	164	324	644	1284	2564
Proposed ALG	Equation (19)	61	125	253	509	1021	2045

It can be seen that the proposed algorithm for $p > 2$ outperforms the conventional sliding DHT algorithms. Note that the proposed algorithm is more efficient than the fast and conventional recursive algorithms when the window length increases.

In modern processors, the execution times of floating point multiplication and addition are comparable. So, further we will estimate the computational complexity with respect to flop counts (real additions and multiplications). Table 4 shows the computational complexity of the tested algorithms for $N = 256$ when p varies.

Table 4. Comparison of the tested algorithms in terms of flops for computing sliding DHT, $N = 256$.

Algorithms	***p*-Step of Sliding Window**				
	2	**3**	**4**	**5**	**6**
Fast DHT [22]	2724	2724	2724	2724	2724
Ref. [19]	2556	3834	5112	6390	7668
Ref. [20]	1912	2868	3824	4780	5736
Proposed ALG	1649	2036	2403	2798	3177

It can be seen that the proposed algorithm is more efficient than the fast DHT algorithm when the step $p < 5$. This boundary value of the step, when the proposed algorithm is still better than the fast DHT algorithm, increases with increasing the window length.

Obviously, the execution time of any algorithm depends on the characteristics of a computer used in a particular implementation of the algorithm. Now, with the help of computer simulation, we want to illustrate how the theoretical computational complexity of the tested algorithms relates to the execution time of the implemented algorithms. Computer simulation was performed on a laptop with an Intel Core i7-2630QM processor with 8 GB of RAM using the MATLAB R2012b. Experiments were carried out 100 times to ensure statistically correct results. Average time results for each algorithm were calculated. Figure 1 shows the performance of the tested algorithms in terms of runtime: DHT is the discrete Hartley transform given by definition, Fast DHT is implemented in Matlab, SDHT is sliding DHT [20], ALG is the proposed algorithm.

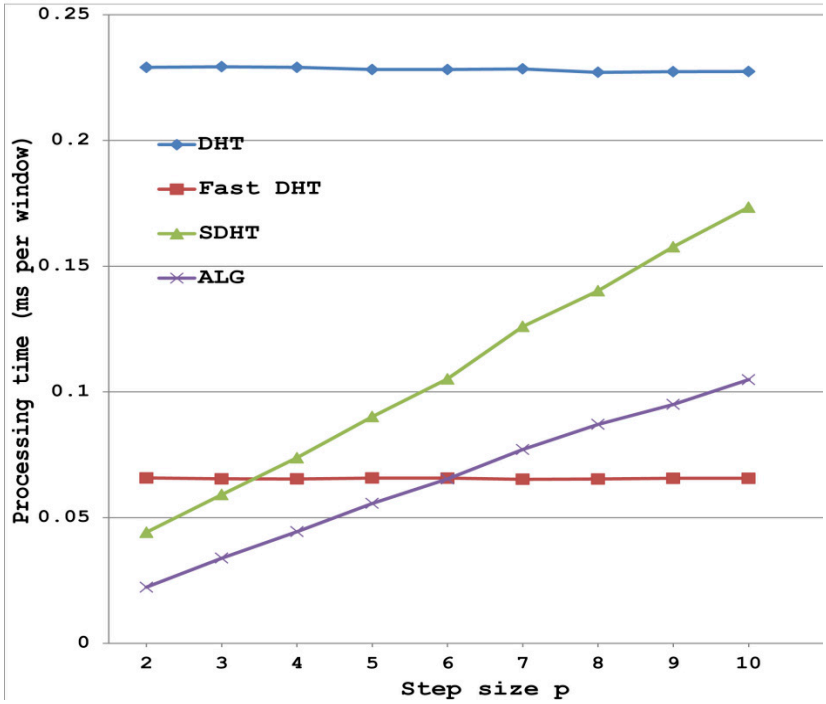

Figure 1. Performance of the tested algorithms in terms of runtime (milliseconds) per window when $N = 256$ and p varies from 2 to 10.

It can be seen that the obtained results are in good agreement with the results in Table 4. Only the performance of the fast algorithm implemented in the MATLAB is slightly lower than that of the algorithm [22].

There are four types of DHT [16] that are suitable for the time varying processing of different signal models. In this paper, the second order recursive equation and fast recursive algorithm have been proposed for only one type of discrete Hartley transform (DHT-I). In the future, the same approach can be used to derive recursive equations and design fast recursive algorithms for other types of DHT, which will allow the system to work effectively with various signal models.

5. Conclusions

A second-order recursive equation between three consecutive equidistant DHT spectra was obtained utilizing of the unilateral z-transform technique. Using the properties of discrete sinusoidal functions and the recursive equation, a fast sliding DHT algorithm was proposed. A fast inverse sliding DHT transform was also presented. The computational complexity of the proposed sliding algorithm was compared with the known running and fast DHT algorithms. The proposed algorithm

was implemented on a laptop, and it was shown that the theoretical computational complexity and execution time of the implemented algorithm are in good agreement.

Funding: This research received no external funding.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Oppenheim, A.V.; Schafer, R.W. *Discrete-Time Signal Processing*, 3rd ed.; Prentice Hall Press: Upper Saddle River, NJ, USA, 2009.
2. Wang, X.; Huang, G.; Zhou, Z.; Tian, W.; Yao, J.; Gao, J. Radar emitter recognition based on the energy cumulant of short time Fourier transform and reinforced deep belief network. *Sensors* **2018**, *18*, 3103. [CrossRef] [PubMed]
3. Thalmayer, A.; Zeising, S.; Fischer, G.; Kirchner, J. A robust and real-time capable envelope-based algorithm for heart sound classification: Validation under different physiological conditions. *Sensors* **2020**, *20*, 972. [CrossRef] [PubMed]
4. Maciusowicz, M.; Psuj, G. Use of time-dependent multispectral representation of magnetic Barkhausen noise signals for the needs of non-destructive evaluation of steel materials. *Sensors* **2019**, *19*, 1443. [CrossRef] [PubMed]
5. Lv, Y.; Pan, B.; Yi, C.; Ma, Y. A novel fault feature recognition method for time-varying signals and its application to planetary gearbox fault diagnosis under variable speed conditions. *Sensors* **2019**, *19*, 3154. [CrossRef] [PubMed]
6. Allen, J. Applications of the short time Fourier transform to speech processing and spectral analysis. In Proceedings of the Acoustics Speech and Signal IEEE International Conference on ICASSP, Paris, France, 3–5 May 1982; pp. 1012–1015. [CrossRef]
7. Kober, V. Robust and efficient algorithm of image enhancement. *IEEE Trans. Consum. Electron.* **2006**, *52*, 655–659. [CrossRef]
8. Jacobsen, E.; Lyons, R. The sliding DFT. *IEEE Signal Process. Mag.* **2003**, *20*, 74–80.
9. Karnaukhov, V.; Kober, V. A fast preview restoration algorithm for space-variant degraded images. In Proceedings of the SPIE's 61 Annual Meeting: Applications of Digital Image Processing XXXIX, San Diego, CA, USA, 29 August–1 September 2016. [CrossRef]
10. Bracewell, R.N. *The Hartley Transform*; Oxford Univ. Press: New York, NY, USA, 1986.
11. Agbinya, J.I.; McLean, D.J. Generalised short-time Hartley transforms for speech processing. In Proceedings of the IEEE Conf. ICCS, Singapore, 14–18 November 1994; pp. 893–896. [CrossRef]
12. Varela, J.; Rodriguez, G.; Guedes Soares, C. Comparison study between the Fourier and the Hartley transforms for the real-time simulation of the sea surface elevation. *Appl. Ocean Res.* **2018**, *74*, 227–236. [CrossRef]
13. Pattanaik, S.K.; Kamalakanta, M. DHT Based JPEG image compression using a novel energy quantization method. In Proceedings of the IEEE International Conference on Industrial Technology, Mumbai, India, 15–17 December 2006; pp. 2827–2832. [CrossRef]
14. Maharana, G.; Meher, P.K. Algorithm for efficient interpolation of real-valued signals using discrete Hartley transform. *Comput. Electr. Eng.* **1997**, *23*, 129–134. [CrossRef]
15. Wang, Z. Fast algorithms for the discrete W transform and for the discrete Fourier transform. *IEEE Trans. Acoust. Speech Signal Process.* **1984**, *32*, 803–816. [CrossRef]
16. Hu, N.-C.; Chang, H.-I.; Ersoy, O.K. Generalized discrete Hartley transforms. *IEEE Trans. Signal Process.* **1992**, *40*, 2931–2940.
17. Britanak, V.; Rao, K.R. The Fast generalized discrete Fourier transforms: A unified approach to the discrete sinusoidal transforms computation. *Signal Process.* **1999**, *79*, 135–150. [CrossRef]
18. Kober, V. Fast algorithms for the computation of sliding discrete sinusoidal transforms. *IEEE Trans. Signal Process.* **2004**, *52*, 1704–1710. [CrossRef]
19. Xi, J.; Chicharo, J.F. Computing running Hartley transform and running discrete W transforms based on the adaptive LMS algorithm. *IEEE Trans. Syst. II* **1997**, *44*, 257–260.
20. Kober, V. Fast algorithms for the computation of sliding discrete Hartley transforms. *IEEE Trans. Signal Process.* **2007**, *55*, 2937–2944. [CrossRef]

21. Bi, G.; Chen, Y.Q.; Zeng, Y. Fast generalized DFT and DHT algorithms. *Signal Process.* **1998**, *65*, 383–390. [CrossRef]
22. Grigoryan, A.M. A novel algorithm for computing the 1-D discrete Hartley transform. *IEEE Signal Process. Lett.* **2004**, *11*, 156–159. [CrossRef]