HYBRID FAST ALGORITHM FOR S TRANSFORMS

Soo-Chang Pei, Jian-Jiun Ding, Pai-Wei Wamg, and Wen Fu Wang

Department of Electrical Engineering, National Taiwan University,
No. 1, Sec. 4, Roosevelt Rd., 10617, Taipei, Taiwan, R.O.C
TEL: 886-2-23635251-321, Fax: 886-2-23671909, Email: pei@cc.ee.ntu.edu.tw, djj@cc.ee.ntu.edu.tw, d92942013@ntu.edu.tw, r96942061@ntu.edu.tw

ABSTRACT

The S transform is useful in time-frequency analysis. In this paper, we propose a hybrid algorithm to implement it adaptively. Since the window size of the S transform varies with |f|, it is reasonable to use different algorithm for different frequency to implement it. In this paper, we use the sub IDFT algorithm in the low frequency region and the sectioned convolution algorithm in the high frequency region to implement the S transform. From simulation, our algorithm reduces 54% of the computation time and much improves the efficiency of the S transform.

1. INTRODUCTION

The S transform [1][2] is defined as:

$$S(\tau,f)=\frac{|f|}{\sqrt{2\pi}}\int_{-\infty}^{\infty}x(t)\exp[-\frac{f^2(\tau-t)^2}{2}]e^{-i2\pi ft}dt\ . \quad (1)$$

It is useful for time frequency analysis. It is a modification of the short time Fourier transform (STFT, also named as the Gabor transform) [3]:

$$G(\tau,f)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}x(t)\exp[-(\tau-t)^2/2]e^{-i2\pi ft}dt\ . \quad (2)$$

Compared with the STFT, the Gaussian mask of the S transform varies with |f|. This makes the S transform have adaptive resolution (high resolution in the low frequency region and low resolution in the high frequency region). The S transform is useful in signal decomposition [4], power quality analysis [5], seismology [6], biomedical engineering [7][8], atmospheric science [9], and other applications related to time-frequency analysis.

In [1], an algorithm for implementing the S transform in the Fourier domain was introduced. Since the S transform in (1) can be viewed as the following convolution operation

$$S(t,f)=x(t)e^{-i2\pi ft}\underset{\text{along } t\text{-axis}}{*}g_f(t),$$

$$\text{where } g_f(t)=\frac{|f|}{\sqrt{2\pi}}\exp[-f^2t^2/2], \quad (3)$$

from

$$FT[x(t)e^{-i2\pi ft}]=X(u+f)\ ,\ FT\left[g_f(t)\right]=e^{-\frac{2\pi^2u^2}{f^2}}, \quad (4)$$

the S transform can be implemented as follows [1]:

Step 1: Perform the discrete Fourier transform (DFT) for the samples of $x(t)$:

$$X\left(k\Delta_f\right)=\frac{1}{N}\sum_{n=0}^{N-1}x_1[n]e^{-j\frac{2\pi kn}{N}}\ ,\ \text{where } \Delta_t\Delta_f=1/N\ , \quad (5)$$

$x_1\left[((n))_N\right]=x\left(n\Delta_t\right)$, $((\ ))_N$ is the modulus operation, and N is some integer that should be larger than the number of sample points of $x(t)$. Note that if the time duration of $x(t)$ is $[T_1, T_2]$ and $T_1 < 0$, then it is possible that $n < 0$ and we should use $((n))_N$ to re-arrange the index before applying the DFT.

Step 2: For each m, compute the N-point inverse discrete Fourier transform (IDFT) as follows. Then, the result will be the S transform of $x(t)$ at $\tau = n\Delta_t$ and $f = m\Delta_f$.

$$S\left(n\Delta_t,m\Delta_f\right)\cong\sum_{k=-N/2}^{N/2-1}X\left((m+k)\Delta_f\right)e^{-\frac{2\pi^2}{m^2}k^2}e^{j\frac{2\pi kn}{N}}\ . \quad (6)$$

Suppose that m has M possible values. Then the above algorithm requires one DFT, M IDFTs, and M times of multiplications of $\exp(-2\pi^2k^2/m^2)$. Thus, the complexity is:

$$\frac{(M+1)N}{2}\log_2 N+MN\ , \quad (7)$$

and the **average complexity** for each m is:

$$\frac{(M+1)N}{2M}\log_2 N+N\cong\frac{N}{2}\log_2 N+N\ . \quad (8)$$

In this paper, we find that the efficiency of the S transform can be further improved.

First, notice that when |f| is large, the **time duration** of $g_f(t) = \exp[-f^2(\tau - t)^2/2]$ is very **small**. It means that (3) is a convolution of $x(t)\exp(-j2\pi ft)$ with a very short function. The convolution of a long function and a short function can be implemented by the sectioned convolution.

By contrast, when |f| is small, since $FT[g_f(t)] = \exp[-2\pi^2u^2/f^2]$ decays very fast, the **bandwidth** of $g_f(t)$ is very **small**. In this condition, the input of the IDFT in (6) is very short and we can use several shorter length IDFTs instead of the N-point IDFT in (6) to implement the S transform.

Therefore, one can implement the S transform adaptively according to the value of |f|.

(i) When **|f| is large**, the **sectioned convolution** algorithm can be applied. See Section 2.

(ii) When **|f| is small**, the algorithm of **sub IDFT** can be applied. See Section 3.

(iii) Even in the middle frequency region, using both the sectioned convolution and the sub IDFT algorithms are more efficient than using the original algorithm. See Figs. 1 and 2.

We also perform several simulations in Section 4 to show that the proposed hybrid algorithm can much improve the efficiency of the S transform.

2. Sectioned Convolution in High Frequency Region

Suppose that the number of sampling points for $x(t)$ is N_1:

$$x(t) \xrightarrow{sampling} x(n\Delta_t),$$

$$\text{where } n = n_0, n_0+1, .., n_0+N_1-1. \quad (9)$$

In theory, the function $g_f(t)$ in (3) has infinite duration. However, one can give a threshold:

$$threshold = 0.001 \max\left(\left|g_f(t)\right|\right) \quad (10)$$

and ignore the case where $|g_f(t)| <$ threshold. It makes $g_f(t)$ become a time-limited function with the time duration of:

$$-B_f < t < B_f,$$

$$\text{where } B_f = \sqrt{2\log(10^3)} / |f| = 3.7169/|f| \quad (11)$$

and the number of sampling points in the duration is

$$L = 1 + 2\left\lfloor \frac{3.7169}{|f|\Delta_t} \right\rfloor, \quad (12)$$

where $\lfloor\ \rfloor$ means the rounding down operation. Therefore, (3) becomes the linear convolution of an N_1-length sequence $x(n\Delta_t)$ and an L-length sequence $g(n\Delta_t)$.

However, **when $|f|$ is large**, the time-duration of $g_f(t)$ is very short and the value of L in (12) is also very small. In this case, instead of (3), one can divide $x(n\Delta_t)$ into several shorter sequences and use the **sectioned convolution** to implement the convolution of $x(n\Delta_t)$ and $g(n\Delta_t)$. That is,

Step 1: Divide $x(n\Delta_t)$ into B subsections. The length of each subsection is N_2, where $N_2 = \lceil N_1/B \rceil$ and $\lceil\ \rceil$ means the rounding up operation. That is,

$$x_{p,m}[n] = x\left[(n_0 + pN_2 + n)\Delta_t\right] e^{-j\frac{2\pi(n_0+pN_2+n)m}{N}}, \quad (13)$$

where $n = 0, 1, \ldots, N_2-1$, $p = 0, 1, \ldots, B-1$,

and $\exp(-j2\pi(n_0+pN_2+n)m/N)$ comes from substituting $t = (n_0+pN_2+n)\Delta_t$ and $f = m\Delta_f$ into $\exp(-j2\pi ft)$ in (3). Moreover, as in (5), $N = 1/\Delta_t\Delta_f$ should be satisfied.

Step 2: Then, we use the N_3-point DFT / IDFT pair to implement the convolution in (3) instead of the N-point DFT / IDFT pair used in (5) and (6), where N_3 should satisfy

$$N_3 > N_2 + L - 1 \approx (N_1/B) + L - 1. \quad (14)$$

This step can be accomplished by the following sub steps:

Step 2-1: $$X_{p,m}[k] = \frac{1}{N_3}\sum_{n=0}^{N_3-1} x_{p,m}[n] e^{-j\frac{2\pi kn}{N_3}}$$

$$\text{for } p = 0, 1, \ldots, B-1. \quad (15)$$

Step 2-2: $$T_{p,m}[n] = \sum_{k=0}^{N_3-1} X_{p,m}[k] e^{-\frac{2\pi^2}{m^2}k^2\frac{N^2}{N_3^2}} e^{j\frac{2\pi kn}{N_3}}$$

$$\text{for } p = 0, 1, \ldots, B-1. \quad (16)$$

Step 2-3: $$Q_{p,m}[n] = T_{p,m}[n]$$

$$\text{for } n = 0, 1, \ldots, N_2+L_1-1,\ L_1 = (L-1)/2,$$

$$Q_{p,m}[n] = T_{p,m}[n+N_3]$$

$$\text{for } n = -L_1, -L_1+1, \ldots, -1. \quad (17)$$

and $Q_{p,m}[n] = 0$ otherwise. Then $Q_{p,m}[n]$ is near to

$$Q_{p,m}[n] \cong x_{p,m}[n] * \Delta_t g_f(n\Delta_t), \quad (18)$$

where * means the linear convolution and $g_f(t)$ is defined in (3).

Steps 2-2 and 2-3 can be proved as follows

$$DFT_{N_3}\left\{\Delta_t g_f(n\Delta_t)\right\} = \frac{|f|}{\sqrt{2\pi}}\sum_n \exp\left[\frac{-f^2n^2\Delta_t^2}{2}\right] e^{-j\frac{2\pi}{N_3}kn}\Delta_t$$

(DFT_{N_3} means the N_3-point DFT)

$$\cong \frac{|f|}{\sqrt{2\pi}}\int \exp(-f^2t^2/2)e^{-j2\pi ut}dt\Big|_{t\to n\Delta_t, u\to k\Delta_u, \Delta_t\Delta_u = 1/N_3}$$

$$= \exp(-2\pi^2u^2/f^2) \quad \text{(from (4))}$$

$$= \exp\left(\frac{-2\pi^2k^2}{m^2\Delta_f^2\Delta_t^2N_3^2}\right) \quad (\text{from } u = k\Delta_u = k/\Delta_tN_3 \text{ and } f = m\Delta_f)$$

$$= \exp(-2\pi^2k^2N^2/m^2N_3^2) \quad (\text{from } \Delta_t\Delta_f = 1/N). \quad (19)$$

Therefore, from (16),

$$T_{p,m}[n] = IDFT_{N_3}\left\{DFT_{N_3}\left\{x_{p,m}[n]\right\}\exp\left(\frac{-2\pi^2k^2N^2}{m^2N_3^2}\right)\right\}$$

$$\cong IDFT_{N_3}\left\{DFT_{N_3}\left\{x_{p,m}[n]\right\}DFT_{N_3}\left\{\Delta_t g_f(n\Delta_t)\right\}\right\}. \quad (20)$$

Thus, $T_{p,m}[n]$ is the circular convolution of $x_{p,m}[n]$ and $\Delta_t g_f(n\Delta_t)$ and $Q_{p,m}[n]$ in (17) is their linear convolution.

Step 3: Sum the linear convolution results of each subsection. Then we obtain the result of the S transform (denoted by $S(n\Delta_t, m\Delta_f)$):

(a) $S\left((n_0 + pN_2 + n)\Delta_t, m\Delta_f\right) \cong Q_{p,m}[n]$

when (i) $L_1 \le n < N_2-L_1$, $p = 0, 1, \ldots, B-1$,

(ii) $0 \le n < L_1$, $p = 0$, or

(iii) $N_2-L_1 \le n < N_2$, $p = B-1$, where $L_1 = (L-1)/2$,

(b) $S\left((n_0 + pN_2 + n)\Delta_t, m\Delta_f\right) \cong Q_{p,m}[n] + Q_{p-1,m}[n+N_2]$,

when $0 \le n < L_1$, $p = 1, 2, \ldots, B-1$

(c) $S\left((n_0 + pN_2 + n)\Delta_t, m\Delta_f\right) \cong Q_{p,m}[n] + Q_{p+1,m}[n-N_2]$,

when $N_2-L_1 \le n < N_2$, $p = 0, 1, , \ldots, B-2$. (21)

Step 3 can be proven from the fact that

$$x[n]e^{-j\frac{2\pi}{N}mn} = \sum_{p=0}^{B-1} x_{p,m}[n-n_0-pN_2] \quad (22)$$

$$S(n\Delta_t, m\Delta_f) \cong x[n]e^{-j\frac{2\pi}{N}mn} * \Delta_t g_f(n\Delta_t) \quad \text{(from (3))},$$

Table 1 Complexities of the original algorithm and the proposed algorithm in Section 2 for implementing the S transform. Here, $\Delta_t = \Delta_f = 0.02$ and $0 \le t \le 30$.

frequency	$(N/2)\log_2 N + N$ (reflect the complexity of the original algorithm)	$[B(N_1/B+L-1)/2]\cdot\log_2(N_1/B+L-1) + N_1+BL-B$ (reflect the complexity of the **proposed** algorithm)			
		$B = 2$	$B = 4$	$B = 6$	$B = 9$
$\lvert f\rvert = 2$	16610	**11091**	12446	14018	16502
$\lvert f\rvert = 6$	16610	9453	**9372**	9563	10027
$\lvert f\rvert = 10$	16610	9139	8787	**8723**	8818
$\lvert f\rvert = 15$	16610	8982	8497	8306	**8221**

$$S\left(n\Delta_t, m\Delta_f\right) \cong \sum_{p=0}^{B-1} x_{p,m}\left[n-n_0-pN_2\right] * \Delta_t g_f\left(n\Delta_t\right)$$
$$\cong \sum_{p=0}^{B-1} Q_{p,m}\left[n-n_0-pN_2\right] \text{ (from (18))}. \quad (23)$$

Then we analyze the complexity of the proposed algorithm. Suppose that there are M_1 possible values of m. Then, the multiplication of $\exp(-2\pi^2k^2N^2/(m^2N_3^2))$ and the N_3-point IDFT in (16) should be computed M_1B times. Since

$$X_{p,m+N/N_3}[k] = X_{p,m}[k+1]e^{-j\frac{2\pi(n_0+pN_2)}{N_3}}, \quad (24)$$

thus, in (13) and (15), we only have to calculate $x_{p,m}[n]$ and $X_{p,m}[k]$ for $m = 0, 1, \ldots, N/N_3 - 1$. Therefore there are N_1N/N_3 multiplications in (13), where N_1 is the number of input sampling points (see (9)), and the N_3-point DFT in (15) should be performed BN/N_3 times. The product of $\exp(-j2\pi(n_0+pN_2)/N_3)$ in (24) can be merged with the product of $\exp(-2\pi^2k^2N^2/(m^2N_3^2))$ in (16). Therefore, the total complexity of the proposed algorithm is

$$\left(B\frac{N}{N_3}+M_1B\right)\frac{N_3}{2}\log_2 N_3 + N_1\frac{N}{N_3} + M_1BN_3$$
$$\cong \frac{1}{2}\left(B\frac{N}{N_3}+M_1B\right)\left(N_1/B+L-1\right)\log_2\left(\frac{N_1}{B}+L-1\right)$$
$$+N_1N/N_3 + M_1B\left(N_1/B+L-1\right) \quad (25)$$

and the **average complexity** for each m of the **proposed algorithm** is near to

$$\frac{1}{2}\left(\frac{BN}{M_1N_3}+M_1B\right)\left(N_1/B+L-1\right)\log_2\left(\frac{N_1}{B}+L-1\right)$$
$$+N_1N/(M_1N_3) + B\left(N_1/B+L-1\right)$$
$$\cong \frac{B\left(N_1/B+L-1\right)}{2}\log_2\left(\frac{N_1}{B}+L-1\right) + N_1 + BL - B \quad (26)$$

where N_1 is the number of input sampling points, B is the number of sections, and L is defined in (12) when M_1 is sufficient large. In comparison, for the original algorithm in [1], the average complexity is as in (8). In the case where $L << N$, if we choose $B \ge 2$ and the constraint that $N_1 < N$ is satisfied, then $N_1/B + L - 1 << N$ and (26) will be much less than (8).

For example, if Δ_t and Δ_f are fixed to 0.02, from (5), $N = 2500$ and the value of $(N/2)\log_2 N + N$ in (8) is near to 16610. If the duration of the input signal is $0 \le t \le 30$, then the value of N_1 is 1501. In Table 1, we show the value of (26)) for different B and $|f|$ (Note that $|f|$ affects the value of L (see (12)) and hence the value of (26)). It shows that the proposed algorithm indeed has less complexity than the original algorithm. Especially, when $|f|$ is large, using larger B (i.e., dividing the input into more sections) will be even more efficient.

3. Sub IDFT Algorithm in Low Frequency Region

In previous section, we described that when $|f|$ is large the sectioned convolution algorithm is helpful for improving the efficiency of the S transform. In this section, we show that, **when $|f|$ is small**, the sub DFT algorithm will be more helpful for improving the efficiency. Note that, when $|f|$ is small, since $f = m\Delta_f$, the value of $|m|$ in (6) is also small. It means that the exponential term $\exp(-2\pi^2k^2/m^2)$ decays very fast. If we ignore the case where $\exp(-2\pi^2k^2/m^2) < 10^{-3}$, then (6) can be rewritten as:

$$S\left(n\Delta_t, m\Delta_f\right) = \sum_{k=-N_4}^{N_4} X\left((m+k)\Delta_f\right)e^{-\frac{2\pi^2}{m^2}k^2}e^{j\frac{2\pi kn}{N}}, \quad (27)$$

$$\text{where } N_4 = \left\lfloor |m|\sqrt{\frac{\log(10^3)}{2\pi^2}}\right\rfloor = \left\lfloor \frac{0.5916}{\Delta_t}|f|\right\rfloor. \quad (28)$$

(27) is the N-point IDFT whose input has $2N_4 + 1$ points. If $|f|$ is very small, $2N_4 + 1$ is much less than N. For example, when $\Delta_t = \Delta_f = 0.02$ and $|f| = 1$, the value of $N = (1/\Delta_t\Delta_f)$ is 2500 but the value of N_4 in (24) is only 30. Since $2N_4 + 1 = 61 << N$, in this case, using the N-point IDFT to implement (27) directly would be very inefficient. Instead, we can use the **sub IDFT algorithm** described as follows to implement (27) and improve the efficiency.

First, find an integer L_1 such that

(a) $L_1 \ge 2N_4 + 1$ and (b) $B_1 = N/L_1$ is an integer. (29)

Then, (27) can be re-expressed as:

$$S\left((aB_1+c)\Delta_t, m\Delta_f\right) = \sum_{k=-N_4}^{N_4} X\left((m+k)\Delta_f\right)e^{-\frac{2\pi^2}{m^2}k^2}e^{j\frac{2\pi k}{N}(aB_1+c)}$$
$$= \sum_{k=-N_4}^{N_4}\left\{X\left((m+k)\Delta_f\right)e^{j\frac{2\pi k}{N}c}e^{-\frac{2\pi^2}{m^2}k^2}\right\}e^{j\frac{2\pi k}{L_1}a}, \quad (30)$$

where $c = ((n))_{B1}$, $a = (n-c)/B_1$, and $n = aB_1 + c$. (31)

Then, from the fact that **the complexity of B_1 times of the (N/B_1)-point IDFTs is less than that of one N-point IDFT**, we can use the following way to implement the N-point IDFT instead of using (6) directly to improve the efficiency. (Step 1 in (5) is unchanged.)

(Modification for <u>Step 2</u>): First, calculate

$$X_{c,m}(k) = X\left((m+k)\Delta_f\right)e^{j\frac{2\pi k}{N}c}e^{-\frac{2\pi^2}{m^2}k^2}$$
$$\text{for } c = 0, 1, \ldots, B_1-1. \quad (32)$$

Then, perform the L_1-point IDFT for each of $X_{c,m}(k)$ ($c = 0, 1, \ldots, B_1-1$):

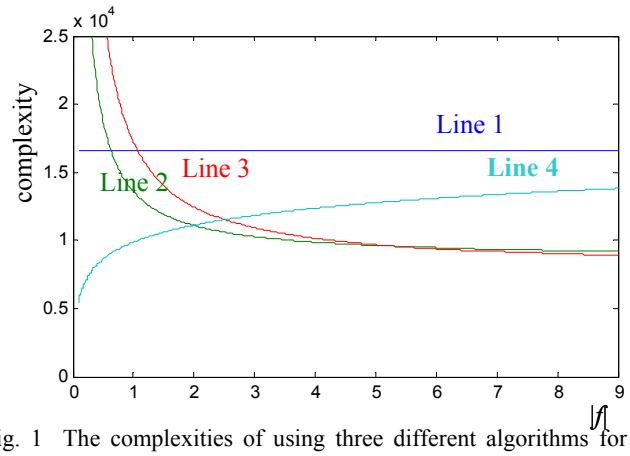

Fig. 1 The complexities of using three different algorithms for implementing the S transform. **Line 1**: the original algorithm, **Lines 2 and 3**: using the sectioned convolution algorithm in Section 2 (B = 2 for line 2 and B = 4 for Line 3), **Line 4**: using the sub IDFT algorithm in Section 3.

$$S\left((aB_1+c)\Delta_t, m\Delta_f\right) = IDFT_{L_1}\left[X_{c,m}(k)\right].$$
$$= \sum_{k=-N_4}^{N_4} X_{c,m}(k) e^{j\frac{2\pi k}{L_1}a} \qquad (33)$$

Note that if there are M_1 possible values for m, then in (33) the L_1-point IDFT should be performed B_1M_1 times. Therefore, the complexity of the modified algorithm is:

$$\frac{N}{2}\log_2 N \ \text{(from Step 1)} \ +$$
$$M_1B_1(2N_4+1) \ \text{(from (32))} + M_1B_1\frac{L_1}{2}\log_2 L_1 \ \text{(from (33))}$$
$$\cong \frac{N}{2}\log_2 N + M_1N + M_1\frac{N}{2}\log_2 L_1. \qquad (34)$$

Here, we use the fact that $B_1(2N_4+1) \approx B_1L_1 = N$ (from (29)). Thus, the **average complexity** for each m of the **proposed sub IDFT algorithm** is:

$$\frac{N}{2M_1}\log_2 N + \frac{N}{2}\log_2 L_1 + N \approx \frac{N}{2}\log_2 L_1 + N \qquad (35)$$

if M_1 is sufficient large. Compared with (8), since $L_1 = N/B_1 << N$, it is obvious that **$(N/2)\log_2 L_1$ is much less than $(N/2)\log_2 N$**. Therefore, using the proposed sub IDFT algorithm can indeed improve the efficiency of the S transform, especially in the condition where $|f|$ is small (Note that, since $L_1 = 2N_4+1$ and N_4 is proportional to $|f|$ (see (28)), if $|f|$ is small, the value of L_1 in (35) is also small).

In Fig. 1, we compare the efficiencies of the three algorithms for implementing the S transform. (i.e., original algorithm, the sectioned convolution algorithm in Section 2, and the sub IDFT algorithm in Section 3) Here,

$$\Delta_t = \Delta_f = 0.02 \ \text{(i.e., } N = (1/\Delta_t\Delta_f) = 2500) \qquad (36)$$

and $0 \le t \le 30$. **Line 1** is $(N/2)\log_2 N + N$, which is the complexity of the original algorithm. It is invariant with f. **Lines 2 and 3** are $[B(N_1/B+L-1)/2]\cdot\log_2(N_1/B+L-1) + N_1+BL-B$ when B = 2 and 4, respectively. They are the complexities when using the sectioned convolution algorithm in Section 2. **Line 4** is $(N/2)\log_2 L_1 + N$, which is the complexity of the sub IDFT algorithm described in this section.

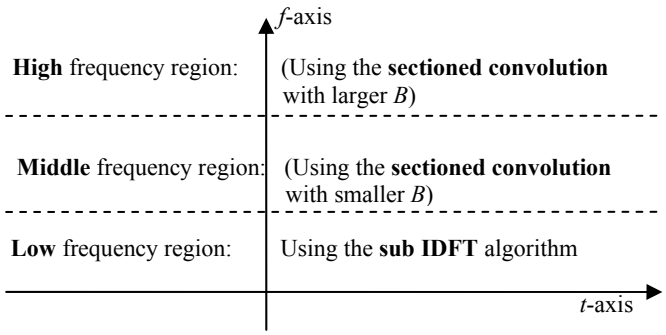

Fig. 2 Using different algorithms in different regions to implement the S transform, where B means **the number of sections** when using the sectioned convolution algorithm.

From Fig. 1, it is obvious that when $|f|$ is small, using the sub IDFT algorithm proposed in this Section will be more efficient for implementing the S transform. When $|f|$ is in the middle region, it is proper to use the sectioned convolution algorithm in Section 2 with smaller B to implement the S transform. In the high frequency region, it is proper to use the sectioned convolution algorithm with larger B, as in Fig. 2. Therefore, we can use the **hybrid algorithm** to implement the S transform. **For different $|f|$, the algorithm for implementing the S transform is also different**.

4. Simulations

We perform several simulations to compare the efficiencies of the proposed hybrid algorithm and the original algorithm to implement the S transform. There are two input signals:

Fig. 3(a): $x(t) = \cos(2\pi t)$ for $0 \le t \le 10$,
$x(t) = \cos(6\pi t)$ for $10 \le t \le 20$,
$x(t) = \cos(4\pi t)$ for $20 \le t \le 30$, (37)

Fig. 4(a): $x(t) = \cos(20\pi t)$ for $0 \le t \le 10$,
$x(t) = \cos(3\pi t)$ for $10 \le t \le 20$,
$x(t) = \cos(10\pi t)$ for $20 \le t \le 30$. (38)

Their S transforms (computed by the original algorithm in (6) and (7)) are shown in Figs. 3(b) and 4(b).

Then, we use the proposed hybrid algorithm to compute the S transforms. In Fig. 3(b), we use the sub IDFT algorithm for $|f| < 2$ and use the sectioned convolution with for $|f| \ge 2$. We plot the results in Figs. 3(c) and 4(c). The results are all the same as those of using the original algorithm. It proofs that the proposed hybrid algorithm is valid.

Then, we show the computation time in Table 2. The results show that the proposed hybrid algorithm **saves over 54% of the computation time** and is much more efficient than the original algorithm.

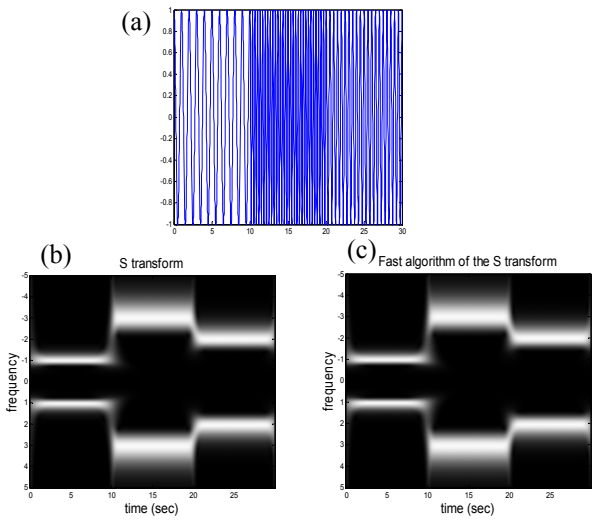

Fig. 3 (a) The input signal $x(t)$ (defined in (37)), (b) the S transform of $x(t)$ computed by the original algorithm, and (c) the S transform of $x(t)$ computed by the proposed hybrid algorithm.

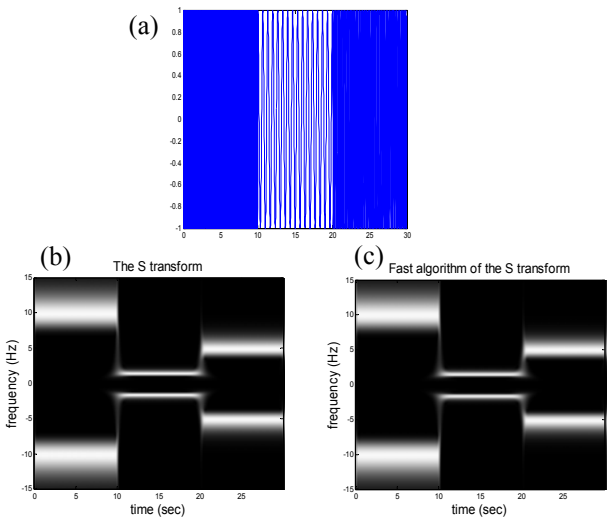

Fig. 4 (a) The input signal $x(t)$ (defined in (38)), (b) the S transform of $x(t)$ computed by the original algorithm, and (c) the S transform of $x(t)$ computed by the proposed hybrid algorithm.

5. Conclusions

In this paper, we propose a hybrid algorithm to improve the implementation efficiency of the S transform. We find that, when $|f|$ is small, it is proper to use the sub IDFT algorithm instead of the original method to implement the S transform. When $|f|$ is large, it is proper to use the sectioned convolution algorithm and the number of sections is increased when $|f|$ grows larger. The proposed algorithm much reduces the computation time of the S transform and is very helpful for time-frequency analysis.

Table 2 Comparing the computation times of the original and the proposed hybrid algorithms for implementing the S transform

Computation time	Original algorithm	**Hybrid algorithm**	Improvement
Signal in Fig. 3	3.20 sec	1.47 sec	54.1%
Signal in Fig. 4	8.21 sec	3.71 sec	54.8%

References

[1] R. G. Stockwell, L. Mansinha, and R. P. Lowe, "Localization of the complex spectrum: the S transform," *IEEE Tran. Signal Processing*, vol. 44, no. 4, Apr. 1996.

[2] S. Ventosa, C. Simon, M. Schimmel, J. J. Dañobeitia and A. Mànuel, "The S-Transform From a Wavelet Point of View," *IEEE Trans. Signal Process*, vol. 56, no. 7, pp. 2771-2780, July 2008.

[3] M. J. Bastiaans, "Gabor's expansion of a signal into Gaussian elementary signals," *Proc. IEEE*, vol. 68, pp. 594-598, 1980.

[4] L. J. Stankovic, T. Thayaparan, and M. Dakovic, "Signal Decomposition by Using the S-Method," *EUSIPCO*, Sept. 2005.

[5] P. K. Dash, B. K. Panigrahi and G. Panda, 'Power quality analysis using S-transform," *IEEE Trans. Power Deliv.*, vol. 18, pp. 406-411, Apr. 2003.

[6] J. H. Gao, W. C. Chen, Y. M. Li, and F. Tian, "Generalized S-transform and seismic response analysis of thin interbeds," *Chinese J. Geophysics*, Chinese Ed., vol. 46, pp. 526-532, July 2003.

[7] G. Livanos, N. Ranganathan, and J. Jiang, "Heart sound analysis using the S transform," *IEEE Computers in Cardiology*, pp. 587-590, 2000.

[8] S. Assous, A. Humeau, M. Tartas, P. Abraham and J. P. L'Huillier, "S-Transform Applied to Laser Doppler Flowmetry Reactive Hyperemia Signals," *IEEE Trans. Biomed. Eng.*, vol. 53, pp.1032-1037, June 2006.

[9] Y. Portnyagin, E. Merzlyakov, C. Jacobi, N. Mitchell, H. Muller, A. Manson, W. Singer, P. Hoffmann, and A. Fachrutdinova, "Some results of S-transform analysis of the transient planetary-scale wind oscillations in the lower thermosphere," *Earth Planets and Space*, vol. 51, pp. 711–717, 1999.