Fast Partial Frequency Spectrum Computation for Real-Time Information Acquisition Systems

Chang-Eup Ha *, Wan-Jin Kim *, Dae-Won Do **, Dong-Hun Lee **, and Hyoung-Nam Kim *
*School of Electrical Engineering, Pusan National University, Busan, Korea
**Agency for Defense Development, Jinhae, Korea
hnkim@pusan.ac.kr

ABSTRACT
In radar and sonar systems, target information such as distance, bearing, and Doppler shift should be obtained on real time for target detection and estimation. For the real time information acquisition, it seems that time-domain processing is better than frequency-domain processing due to the computational burden. However, since the information which can be extracted from time-domain signal is restricted, the use of frequency-domain processing is inevitable. For frequency-domain processing, the first step is to transform the time-domain signal into its corresponding frequency domain signal. In this step, the widely-used method is the fast Fourier transform (FFT) [1]. It is certain that the FFT is an effective transform method but it may not be exactly true in some situations, for example: an update interval is very short or only partial range of frequency spectrum is required. In detection systems, information update interval should be as short as possible for fast and precise processing, but the computational burden, which is related to the number of executed FFT, arithmetically increases by decreasing the update interval. If the burden is acceptable in the given hardware resource, it may not be a problem. However, if the burden exceeds the system capability, it is hard to process the data in the frequency domain on real time. This short update interval problem may be solved by adopting additional hardware resource but it is not a recommended solution. Especially, in case of the partial use of frequency spectrum, a waste of the hardware resource is inevitable. Radar and active sonar systems radiate a pulse and detect a signal reflected from targets. Even though the received signal is distorted and Doppler shifted, its frequency characteristic does not change abruptly. This fact means that frequency band required for the detection is not a full-band but a part of it. It is required to eliminate unnecessary operations for the reduction of computational load but the FFT is not able to calculate only partial frequency band, resulting in hardware resource waste.

These problems of the short update interval and partial use of the spectrum are closely connected to the system performance and they should be solved to implement an optimized system. Some researchers have dealt with these problems and they have presented some remarkable methods: generalized sliding FFT (SFFT) and pruning techniques such as transform decomposition (TD) [2]-[5]. The GSFFT is one of the computing methods of sequential discrete Fourier transform (DFT) using FFT and it has lower computation complexity than recursive FFT computation. The TD is one of the existing high efficiency pruning methods and it is able to compute only desired frequency bins. While the GSFFT and the TD may be a good solution to overcome the short update interval and partial use of the spectrum, respectively, they may not be when both requirements have to be dealt with in one process. To cope with the situation where two demands are required simultaneously, we propose a new method called the pruned GSFFT (PGSFFT). The PGSFFT is based on the combination of the GSFFT and the TD. The PGSFFT utilizes both of their advantages, and thus it is able to reduce the computational complexity in the given situation.

KEY WORDS
Pruning, Transform decomposition, GSFFT, sliding DFT

1. Introduction
In radar and sonar systems, target information such as distance, bearing, and Doppler shift should be obtained on real time for target detection and estimation. For the real time information acquisition, it seems that time-domain processing is better than frequency-domain processing due to the computational burden. However, since the information which can be extracted from time-domain signal is restricted, the use of frequency-domain processing is inevitable. For frequency-domain processing, the first step is to transform the time-domain signal into its corresponding frequency domain signal. In this step, the widely-used method is the fast Fourier transform (FFT) [1]. It is certain that the FFT is an effective transform method but it may not be exactly true in some situations, for example: an update interval is very short or only partial range of frequency spectrum is required. In detection systems, information update interval should be
The rest of this paper is organized as follows. In section 2, we briefly describe the GSFFT and the TD which compose the proposed algorithm. The description on our proposed method and its computational complexity analysis are presented in section 3. In section 4, we show the computational efficiency of PGSFFT compared to other FFT and pruning techniques. Finally in section 5, we conclude the paper.

2. Overview of the GSFFT and TD

2.1 Generalized Sliding FFT

The sliding DFT (SDFT) is a computing method to sequentially get frequency-domain signal whenever the time-limited rectangular window is moved one sample [6]. The concept of the SDFT is that the DFT of the current input sequence in an N-points window can be obtained just with one complex multiplication and two complex additions when the one-sample-advanced DFT is known. Especially if the input sequence length N is a power of two, the SDFT can be shown as N-point FFT which requires N complex multiplications. From this idea, the sliding FFT (SFFT) is developed [2], [7]. The SFFT is very efficient method compared to recursive FFT computations but the remained problem is that the SFFT is only useful when input sequence slides one sample at a time.

The generalized SFFT (GSFFT) is a method which hops power of two samples and the GSFFT makes it possible to eliminate the redundancies between consecutive slides. To help the understanding, an example is illustrated in fig. 1. If the FFT computation is performed every two samples, the overlapped part between the previous and current window is dotted circle which is marked by ‘A’ in fig. 1. In the GSFFT, the ‘A’ marked butterfly is not computed because it has been already computed and saved to the memory in the previous calculation. Eliminating the redundancy part in fig. 1, the general FFT structure can be modified as fig. 2.

For further explanation on the GSFFT, let U(m) be input signals which is moved into the window at the current iteration, then and U(m) is defined as follows [7]

\[
U(m) = [x(n) x(n-1) ... x(n-2^k + 1)]^T,
\]

where x(n) is input signal. k is the number of nodes which do not use a previously saved data and it is logarithm of M to base 2. Using (1), the input vector x(m) in the window at m-th iteration can be expressed by [7]

\[
x(m) = [U^T(m) U^T(m-1) ... U^T(m-2^{q+1})]^T
= [x(n) x(n-1) ... x(n-N+1)]^T
\]

where q is the number of nodes which use a previously saved data and it is the logarithm of N/M to base 2. The state vector S_i(m) at i-th node is defined as and i is the node index whose range is 0 ≤ i < k + q. For 0 ≤ i < q, S_i(m) is expressed by [7]

\[
S_{i+1}(m) = \left( \begin{array}{c}
    1 & 1 \\
    1 & -1
  \end{array} \right) \otimes I_{2^{q+i}} \left[ S_i(m) \right] V_i S_i(m-2^{q+i-1})
\]

where

\[
\begin{align*}
S_0(m) & = U(m) \\
P_i & = P_i \otimes I_{2^i} \\
V_i & = \text{diag}(e_0, e_1, ..., e_{2^{q+i}-1}) \\
P_i & = [e_{odd} e_{even}]
\end{align*}
\]

Here, \( I_{2^i} \) denotes the \( 2^i \) by \( 2^i \) identity matrix and \( \otimes \) is the Kroneker product. \( P_i \) denotes a \( 2^{q+i} \) by \( 2^{q+i} \) permutation matrix where \( e_{odd} \) is \([e_1 e_3 ... e_{2^{q+i}-1}]\) and \( e_{even} \) is \([e_2 e_4 ... e_{2^{q+i}-1}]\) and \( e_i \) means the r-th column of the \( 2^{q+i} \) by \( 2^{q+i} \) identity matrix. \( V_i \) is a diagonal matrix which is
composed of twiddle factor $W_N$ and $\sigma(t)$ is the bit reverse of $r$. For $q \leq i < k + q$, $S(m)$ is expressed by [7]

$$S_i(m) = [S_{i+1}(m) S_{i-1}(m)]^q,$$  \hspace{1cm} (5)

and the next state is evaluated as follows

$$\begin{bmatrix} S_{i+1}(m) \\ S_{i-1}(m) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} V_i S_{i+2Nq-1}(m) \\ V_i S_{i+2Nq-2}(m) \end{bmatrix},$$  \hspace{1cm} (6)

where $V_i$ is $W_{2^N}^{\sigma(t)}$ and $\sigma(t)$ is equal to bit reverse of $(t-1) \mod N / 2^{i+1}$.

The above equations, especially (3) and (6), show that the computational complexity of the GSFFT depends on the updated interval $M$ and the window size $N$. To clarify the effect by various $M$ and $N$, it is needed to quantify the computational complexity of the GSFFT. The number of complex multiplication is equal to the sum of the number of all twiddle factors. In (3), the number of required twiddle factors is $2^N k$ at each node and their sum is

$$\sum_{i=0}^{j=q-1} 2^i k = 2^q (2^0 + 2^1 + \cdots + 2^{q-1}) = 2^{q+1} - 2^q = N - M.$$  \hspace{1cm} (7)

Similarly, the sum of twiddle factors of (6) based on general radix 2 FFT computation is

$$\sum_{i=q}^{j=q-k-1} \frac{N}{2} = \frac{kN}{2} = \log_2 M. $$  \hspace{1cm} (8)

From (6) and (7), the total number of complex multiplications for the GSFFT is $(N/2) \log_2 M + N - M$. One complex multiplication is composed of 2 real additions and 4 real multiplications and 4 real additions per butterfly are required. The number of real additions and multiplications is $3N \log_2 M + 6N - 6M$ and $2N \log_2 M + 4N - 4M$ and, respectively. Therefore, the total number of operations is $5N \log_2 M + 10N - 10M$.

### 2.2 Transform Decomposition

The GSFFT can reduce the computational complexity by omitting redundant calculations which are caused by the overlapped data samples between the previous and current window. However, when the desired part of the frequency spectrum is computed, it is not possible to use the GSFFT for reducing the unnecessary calculations. This problem is due to the fact that the GSFFT is able to produce full frequency spectrum. The pruning techniques are a method which eliminates the computation by the unessential input or output, and thus they may be a good solution to overcome the disadvantage of GSFFT. There exists many pruning methods but we only deal with the transform decomposition (TD) which is known to have high computation efficiency.
where the range of \( j \) is from 0 to \( P-1 \). The second step of the TD is calculation of \( P \) \( P \)-point FFTs of (13). In this step, the computational efficiency of TD varies with the algorithm which is used for the computation of \( P \)-point FFT. In general, the split radix FFT algorithm is known to have the lowest complex multiplications and additions among the power of two FFT algorithms [8]. However, unlike the general radix 2 or 4 FFT algorithm, the number of butterflies is varied in successive stages. Since this problem makes it difficult to implement the radix algorithm on the hardware [9], we consider only the Cooley-Tukey FFT algorithm. The last step is to combine outputs of (13) using (12). In (12), the combining process is performed by the DFT but the computing method is not restricted to the DFT. The Goertzel algorithm based on an IIR filtering approach is one of the methods to improve computational efficiency [10] and it can be derived from (12) as follows

\[
X_n(j) = \sum_{n_0=0}^{P-1} x(n_0Q + n) W_{P}^{n_j} = \sum_{n_0=0}^{P-1} x_{n_0}(n_0) W_{P}^{n_j}, \quad (13)
\]

\[\text{where } m = Q - n_2 - 1. \text{ Then let } y_k(j) \text{ as}\]

\[y_k(j) = \sum_{m=0}^{j-1} x_{Q-m-1}(k < p) (W_{Q}^{k})^{j-m-1}. \quad (15)\]

From (14) and (15), \( X(k) \) can be described as

\[X(k) = y_k(j)|_{j=Q}. \quad (16)\]

This equation means that \( X(k) \) is considered as the \( Q \)-out of the convolution between the sequence \((W_{Q}^{k})^{-1}\) and \( X_{Q-m-1}(k < p) \). Fig. 4 (a) and (b) show the flow graph of (15) and its modified version, respectively. If the flow graph of Fig. 4 (b) is applied to the TD, it is able to reduce 4 multiplications to 2 multiplications per iteration.

To evaluate the computational complexity of the TD, we assumed that the radix 2 FFT algorithm and the modified Goertzel algorithm of fig. 4 (b) are used for minimizing the computational complexity. If the length of the desired frequency bins is \( L \), the number of real multiplications for TD is \( 2N \log_2 P + 2LQ + 2L \). The number of real additions is \( 3N \log_2 P + 4L(Q - 1) + 4L \) and the total number of operations is \( 5N \log_2 P + 6LQ + 2L \).

3. Pruned GSFFT

The concept of the proposed pruned GSFFT is based on the idea that the \( P \)-point FFT of the TD is possible to be replaced with other FFT algorithms. Since the aim of this paper is to present the method which extracts the desired frequency band while sliding, we propose the efficient sliding output pruning technique by combining the GSFFT and the TD. That is, the GSFFT is used for \( P \)-point FFT instead of the Cooley-Tukey algorithm. This proposed method is called pruned GSFFT (PGSFFT) and fig. 5 shows how the PGSFFT works.

Letting \( M \) be the update interval and \( N \) be the window length. In the GSFFT, we can freely select \( M \) if it is only satisfied the condition that power of two. In the PGSFFT, assuming that \( N \) is the product of \( P \) and \( Q \), \( M \) has to meet the additional condition as follows

\[M = Qx \text{ where } x = 2^d, 0 \leq d \leq \log_2 N. \quad (17)\]

The updated inputs of PGSFFT at the \( m \)-th iteration is presented in (1) and they are distributed to the \( n_2 \)-th \( P \)-point GSFFT as follows
\[ S_0^p (m) = \left[ x(n-R(0)) \ x(n-R(1)) \ \ldots \ x(n-R\left(\frac{M}{Q} - 1\right)) \right]^T. \quad (18) \]

where

\[ R(n_i) = Qn_1 + n_2, \quad \text{where} \begin{cases} n_1 = 0,1,\ldots,M/L-1 \\ n_2 = 0,1,\ldots,Q-1 \end{cases} \quad (19) \]

Each P-point GSFFT block can be computed as described in (3) and (6). Each output of the P-point GSFFT block is equal to (13) and (15) can be applied to each output of the P-point GSFFT block to compute only a subset of frequency points.

To evaluate the computational complexity of the proposed method, we analyzed the number of required operations. The number of real multiplications for Q times P-point GSFFT block with updated input length \( M'/Q = Q[2N\log_2(M/Q) + 4P - 4(M/Q)] \). If we assume that \( L \) is the length of the output subset, the number of required real multiplications for the GSFFT can be obtained by

\[
\text{MUL}_{\text{GSFFT}} = \frac{Q}{2} \left( 2P \log_2\left( \frac{M}{Q} \right) + 4P - 4 \left( \frac{M}{Q} \right) \right) + 2L(Q+1). \quad (20)
\]

\[
= 2N \log_2\left( \frac{MP}{N} \right) + 4N - 4M + \frac{2LN}{P} + 2L.
\]

The total number of real additions required for the proposed GSFFT is

\[
\text{ADD}_{\text{GSFFT}} = Q \left( 3P \log_2\left( \frac{M}{Q} \right) + 6P - 6 \left( \frac{M}{Q} \right) \right) + 4LQ. \quad (21)
\]

\[
= 3N \log_2\left( \frac{MP}{N} \right) + 6N - 6M + \frac{4LN}{P}.
\]

Finally the total number of real operations can be expressed as follows

\[
\text{OPR}_{\text{GSFFT}} = 5N \log_2\left( \frac{MP}{N} \right) + 10N - 10M + \frac{6LN}{P} + 2L. \quad (22)
\]

Optimum value of \( P \) which is corresponding to the minimal \( \text{OPR}_{\text{GSFFT}} \) can be calculated by taking the derivative of \( \text{OPR}_{\text{GSFFT}} \) for \( P \). \( P \) should satisfy the condition of \( \frac{\text{OPR}_{\text{GSFFT}}}{P} = 0 \) and be the closest power of two and less than \( N \).

\[
\text{Fig. 6. Optimum value of } P \text{ and modified optimum } P \text{ for the case } N=512\text{ and } M=32. \text{ Ideal optimum value of } P \text{ (solid line) under } N/M \text{ must be set to } N/(M/2) \text{ (dashed line). In this case, optimum value of } P \text{ under } 16 \text{ must be set to } 16.
\]

\[
\begin{array}{|c|c|}
\hline
\text{Algorithm} & \text{Real operations} \\
\hline
\text{Cooley-Tukey [1]} & 5N \log_2 N \\
\text{Transform Decomposition (Cooley-Tukey) [3]} & 5N \log_2 P + \frac{6LN}{P} + 2L \\
\text{Pruned FFT (Skinner) [5]} & 5N \log_2 L + 2N - 2L \\
\text{GSFFT [2]} & 5N \log_2 M + 10N - 10M \\
\text{PGSFFT} & 5N \log_2\left( \frac{MP}{N} \right) + 10N - 10M + \frac{6LN}{P} + 2L \\
\hline
\end{array}
\]

\[
P_{opt} = \left[ P \left| \frac{\text{OPR}_{\text{GSFFT}}}{P} \right| \leq \frac{6}{5} \ln 2L \right] = \left\lfloor \frac{0.8318L}{5} \right\rfloor, \quad (23)
\]

From (17) and \( N=PQ \), \( P_{opt} \) should satisfy following equation.

\[
P_{opt} = \frac{N}{M} x \text{ where } x = 2^d, 0 \leq d \leq \log_2 N . \quad (24)
\]

In (24), if \( P_{opt} \) is less than \( N/M \), \( P_{opt} \) is set to \( N/M \) as shown in fig. 6.

4. Computational efficiency of PGSFFT

To verify the computational efficiency of the proposed PGSFFT algorithm, we compared various algorithms and it is presented in table 1. In the table 1, we can see the computational complexity of pruning FFT algorithms such as the TD and the Skinner’s method is dependent on the subset length \( L \). The GSFFT which is one of the sliding DFT algorithms is dependent on the updated interval \( M \). The optimum value of \( P_{opt} \) is used for the transform decomposition using Cooley-Tukey for \( P \)-point
FFT and PGSFFT. PGSFFT is dependent on both $M$ and $L$. To compare the computational efficiency of algorithms, we fixed the parameters $N$ and $M$ and we performed a computer simulation. Here we note that $P_{opt}$ is given to the TD and PGSFFT using (24). The simulation results are presented in figs. 6 and 7.

When the updated interval of $M$ is less than 32, the computational complexity of the PGSFFT is less than other algorithms until the subset length reaches to approximately 200 samples as shown in fig. 6 (a) and (b). However, as $M$ increases, fig. 6 (c) shows that the computational gain is closed to that of the pruned FFT. If $M$ exceeds 64, PGSFFT has no advantage compared to other algorithms. This trend is due to the increase of butterflies which has to be computed in $P$-point GSFFT of PGSFFT. This fact means that PGSFFT does not guarantee the computational efficiency regardless to $M$ and there exists the upper bound of $M$. To ensure the computational efficiency of the PGSFFT, the $M$ should be selected after comparing the efficiency of PGSFFT and other FFT algorithms by the system requirement.

5. Conclusion

We presented a PGSFFT combining GSFFT and transform decomposition. By the mathematical analysis on the computational complexity, we showed that the number of required operation for the proposed PGSFFT is less than other pruning or sliding DFT algorithms when the update interval $M$ is less than one eighth of window size $N$ (or buffer size) and subband length $L$ is less than $N/3$. Since the PGSFFT computes the updated input signal while sliding, it is able to show the desired frequency spectrum on real time with relatively lower computational complexity than every-sample FFT. This advantage is directly linked to the performance improvement of the system. Since the PGSFFT can be applied to the system which uses FFT algorithm such as radar, sonar, frequency-domain beamformer, and real-time audio system, it may be contributed to the hardware optimization.

Acknowledgement

This work has been supported by the Agency for Defence Development, South Korea, “Research on the optimum signal processing technique for the wide-band SONAR system” 2009-2011.
References


