IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 25, 2019, accepted June 29, 2019, date of publication July 31, 2019, date of current version August 14, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2930833

A Non-Feedback-Loop and Low-Computation-Complexity Algorithm Design for a Novel 2-D Sliding DFT Computation

WEN-HO JUANG[1,2] AND SHIN-CHI LAI[1]
[1]Department of Computer Science and Information Engineering, Nanhua University, Chiayi 62249, Taiwan
[2]Himax Technologies, Inc., Tainan 70101, Taiwan

Corresponding author: Shin-Chi Lai (shivan0111@nhu.edu.tw)

This work was supported in part by the Ministry of Science and Technology of China under Grant 108-2221-E-343-003, Grant 107-2221-E-343-002, and Grant 106-2221-E-343-002.

ABSTRACT This paper presents a forward-path, novel, two-dimensional (2-D) sliding discrete Fourier transform (SDFT) algorithm based on the column-row 2-D DFT concept and the shifted window property. After applying a descending dimension method (DDM), a mixed-radix and butterfly-based structure can be further employed to effectively implement the proposed algorithm. Conceptually, it has many advantages, including greater stability, more accuracy, and less computational complexity because there are no extra feedback loops in the calculation. The evaluation results are based on the following conditions: (1) the window size (N) to 16×16; (2) the test pattern is an SVC grayscale video, and the formats are CIF and 4CIF with 30fps; (3) one multiplication involves four real multiplications and two real additions. The proposed 2-D SDFT method clearly reduced the number of multiplications by 43.8% and only increased the number of additions by 33%, compared with the state-of-the-art Park's method. Additionally, for the first 100 frames of the CIF and 4CIF sequences, the proposed method saves 10.8% and 10.9% of the processing time, respectively, on average. Overall, the proposed DDM-based 2-D SDFT algorithm can be applied to calculate not only 1-D but also 2-D SDFT spectrum, and are especially appropriate for hybrid applications.

INDEX TERMS Sliding discrete Fourier transform (SDFT), two-dimension SDFT (2-D SDFT), column-row 2-D DFT, descending dimension method (DDM), DDM-based 2-D SDFT.

I. INTRODUCTION

The discrete Fourier transform (DFT) has been widely employed in many digital signal processing applications to analyze signal power spectral density (PSD). To observe variations in the time-frequency spectrum in detail, a sliding/hopping sinusoidal transform, such as the sliding DFT (SDFT) [1]–[4] and hopping DFT (HDFT) [5]–[8], has been proposed to offer appropriate amounts of information in a short time period. These algorithms are applied on a sample-by-sample basis to analyze the time-frequency spectrum of the desired signals. Recently, SDFT has been used in various applications, e.g., acoustic echo cancellation [9], vibration mode estimators [10], [11], and in defect detection in the rotor bars of an induction motor [12]. The SDFT algorithm directly uses the previous DFT bins to iteratively compute each new output bin, thus making the SDFT's computational requirement significantly lower than that for the traditional DFT. In addition, both the SDFT algorithm and the DFT computation have to perform all of the time indices, and thus the output data rate of the spectral bin is the same as that of the input data rate. This implies that SDFT would have better resolution in the time domain than the traditional DFT and fast Fourier transform (FFT). Considering the problem of an adjustable resolution for the time-frequency representation, the SDFT algorithm has to perform the DFT computation for all of the time indices even though it requires fewer DFT bins, which are calculated every L ($L > 0$) samples. In other words, the resolution of the time-frequency spectrum will be affected by different samples of L. Therefore, several HDFT approaches [5]–[8] are proposed to reduce the workload required for computation.

Recently, Park proposed the idea of time hopping to lower the computational requirement for SDFT algorithms, i.e., the generalized SDFT (gSDFT) in [3] and the optimal SDFT (oSDFT) in [4]. Based on the same concept,

The associate editor coordinating the review of this manuscript and approving it for publication was Yilun Shang.

it was found that 1) SDFT can be combined with the adaptive DFT for the purpose of performing synchro-phasor measurements [13], [14], and 2) it can be applied to extract a more accurate feature vector for the speech/music recognition process [15]–[18]. On the other hand, the trend of the development of HDFT algorithms can be roughly categorized into two classes: 1) the development of fast algorithms, as in the methods used in [5], [6] to derive a radix-2 FFT parallel structure to reduce computational complexity, and 2) considering the simple accelerator design, the methods used in [7], [8] to derive a compact recursive structure intended to achieve a unified recursive kernel for hops of arbitrary lengths. However, most of the existing SDFT/HDFT algorithms have potential problems, including stability, accuracy, and computational complexity, due to the fact that their structures have a recursive loop with multiplication, i.e., the transfer function is only marginally stable.

Currently, the DSP is being gradually emphasized for multi-dimensional applications, especially in the field of image processing. Similarly, the Fourier spectrum analysis based on the sliding transform process has also been extended from one- to multi-dimensions. It is mainly used in the field of computer vision and image recognition, e.g., texture description/segmentation [19], [20], template matching [21], [22], and shoeprint/fingerprint recognition [23], [24]. The process is a type of front-end processing, and it is applied to extract the eigenvalue of a video/image. Traditionally, 2-D SDFT can be directly calculated using the vector-radix 2-D FFT (VRFFT) algorithm [25], but it has heavy computational complexity in the sliding transform scenario. To obtain the advantages of low computational complexity, recursive calculation methods have been proposed. Park proposed a typical recursive-based 2-D SDFT algorithm [26] based on Jacobsen and Lyons' 1-D SDFT concept [1].

Compared to existing algorithms, Park's method significantly reduces the computational workload because the 2-D structure is very similar to the traditional 1-D SDFT. This implies that there is a marginally stable transfer function with the instabilities and accumulated errors which are caused by a recursive kernel process. In addition, Park's algorithm focuses on the 2-D sequential process, so the algorithm architecture is only suitable for 2-D computations. Therefore, it is critical to effectively overcome these problems.

This paper provides a discussion of the development of a proposed low-complexity, non-feedback-loop novel 2-D SDFT algorithm. After applying a descending dimension method (DDM), a mixed-radix and butterfly-based structure can be employed to effectively implement a forward-path 2-D SDFT algorithm. Conceptually, the proposed design would be more stable, more accurate, and would have less computational complexity than Park's recursive structure [26].

The rest of the paper is organized as follows: Section II provides a brief review of Park's 2-D SDFT. In Section III, the proposed forward-path 2-D SDFT algorithm is derived in detail. The overall computational complexity of the proposed algorithm is analyzed in Section IV. Section V compares the performance of the proposed design with other existing approaches. Finally, conclusions and remarks are provided in Section VI.

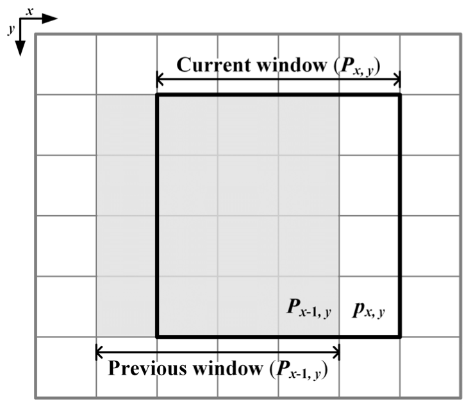


FIGURE 1. The sliding window behavior in the horizontal direction.

II. EXISTING 2-D SDFT ALGORITHM

The N-by-N point DFT standard formula is defined as (1), where $p_{x,y}$ and $P_{x,y}$ (u, v) are, respectively, the time-domain and frequency-domain sequences of the (x, y)-th pixel of the input image. In addition, we assume the sliding window is shifted in the horizontal direction, as shown in Fig. 1.

$$P_{x,y}(u, v) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} p_{\widehat{x}+m,\widehat{y}+n} W_N^{-um} W_N^{-vn}, \tag{1}$$

where $\widehat{x} = x - N + 1$, $\widehat{y} = y - N + 1$, $u, v = 0, 1, \ldots, N - 1$, and the twiddle factor of W_N is denoted by

$$W_N = e^{j2\pi/N}. \tag{2}$$

Based on Park's algorithm [26], the relationship between successive DFT bins can be derived as

$$P_{x,y}(u, v) = W_N^u \begin{pmatrix} P_{x-1,y}(u, v) + \sum\limits_{n=0}^{N-1} p_{\widehat{x}+N-1,\widehat{y}+n} W_N^{-vn} \\ - \sum\limits_{n=0}^{N-1} p_{\widehat{x}-1,\widehat{y}+n} W_N^{-vn} \end{pmatrix}, \tag{3}$$

where P_{x-1}, y (u, v) is defined as the (u, v)th DFT bin of the $(x-1, y)$-th pixel, which is represented as

$$P_{x-1,y}(u, v) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} p_{\widehat{x}+m-1,\widehat{y}+n} W_N^{-um} W_N^{-vn}. \tag{4}$$

Here, the linearity of the DFT is utilized to reduce these redundant DFT computations. The two 1-D DFTs in (3) are merged into one as follows:

$$D_{x,y}(v) = \sum_{n=0}^{N-1} d_{x,y} W_N^{-vn}, \tag{5}$$

where the pixel difference, $d_{x,y}$, is denoted by

$$d_{x,y} = p_{x,y} - p_{x-N,y}. \tag{6}$$

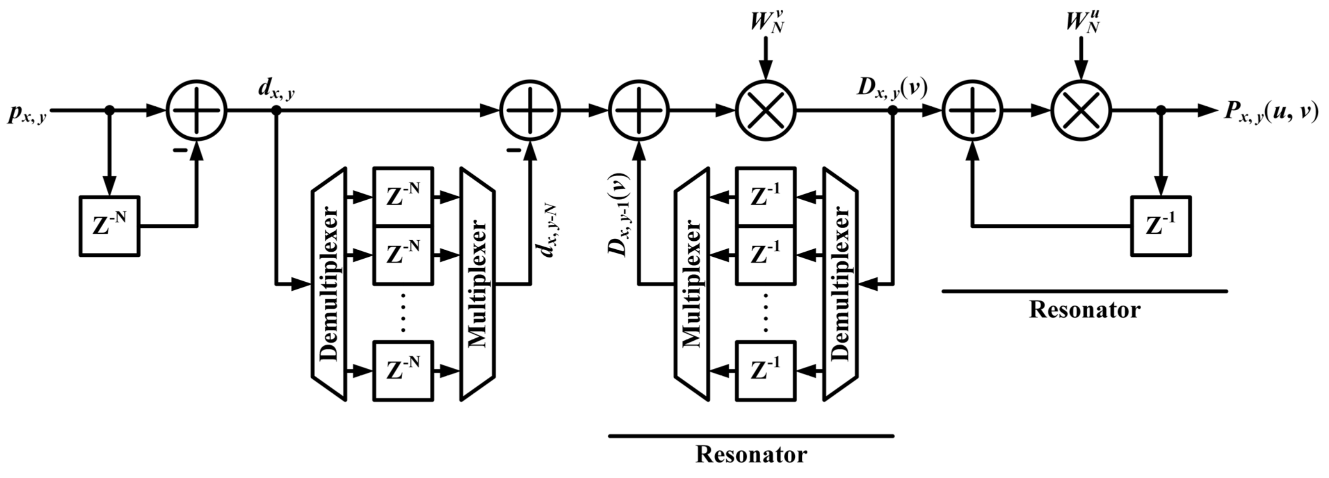


FIGURE 2. Park's 2-D SDFT structure with a dual-recursive-kernel process.

By substituting (4) and (5) into (3), the 2-D Fourier transform expression can be rewritten as

$$P_{x,y}(u,v) = W_N^u \left(P_{x-1,y}(u,v) + D_{x,y}(v)\right). \tag{7}$$

To efficiently implement the function of $D_{x,y}(v)$ in (5), Park adopts a recursive computation according to the relationship between $D_{x,y}(v)$ and $D_{x,y-1}$(v), which is represented by

$$D_{x,y}(v) = W_N^v \left(D_{x,y-1}(v) + d_{x,y} - d_{x,y-N}\right). \tag{8}$$

From (6) to (8), these derivations can be easily mapped into a dual recursive structure. The flow graph is shown in Fig. 2. In general, the 2-D SDFT is applied to a plane, and the architectural design essentially uses two similar resonator cascade filters to handle the corresponding directions, respectively. Therefore, the source of errors would come from both the horizontal and vertical directions simultaneously. Since the SDFT computation is being employed in more and more 2-D applications, the accumulated errors caused by the recursive structure tends become a serious issue.

III. ALGORITHM DERIVATION OF THE PROPOSED FORWARD-PATH 2-D SDFT

In this section, we derive a forward-path computation for the 2-D SDFT algorithm. Considering the computational complexity, the proposed algorithm should be effectively implemented by using a DDM. For the kernel computations of the 2-D SDFT algorithm, we also develop and design the proposed 1-D and mixed-radix structure SDFT algorithm as well [27]. In this way, the algorithm not only has fewer computations but also can be used in a hybrid application.

A. DDM-BASED 2-D SDFT

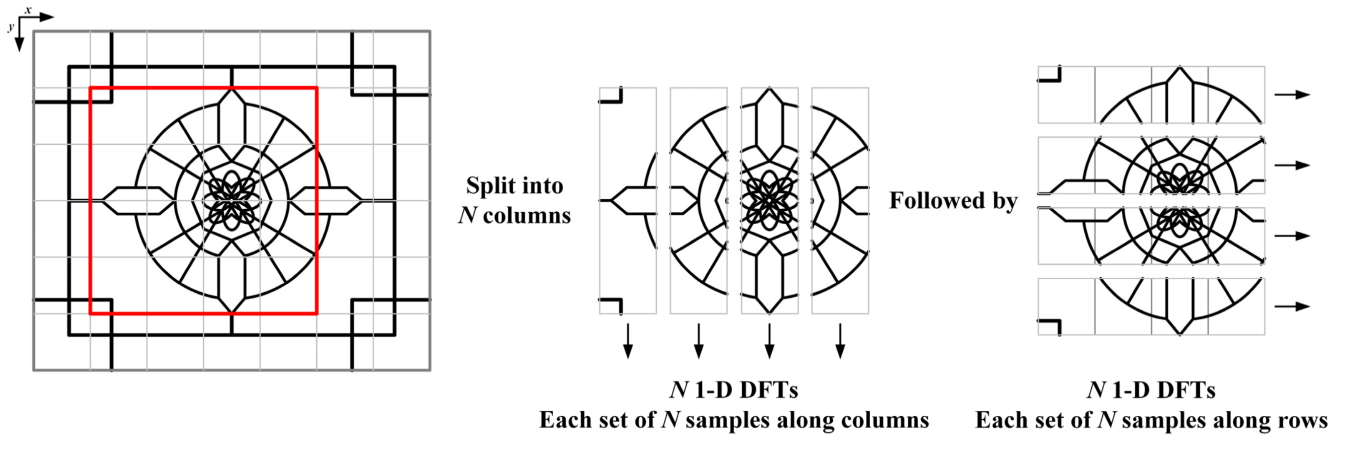
According to the separable property of the 2-D DFT [28], the standard formula [see (1)] can be separated into a series of 1-D DFTs, which can be written as

$$P_{x,y}(u,v) = \sum_{m=0}^{N-1}\left(\sum_{n=0}^{N-1} p_{\widehat{x}+m,\widehat{y}+n} W_N^{-vn}\right) W_N^{-um}. \tag{9}$$

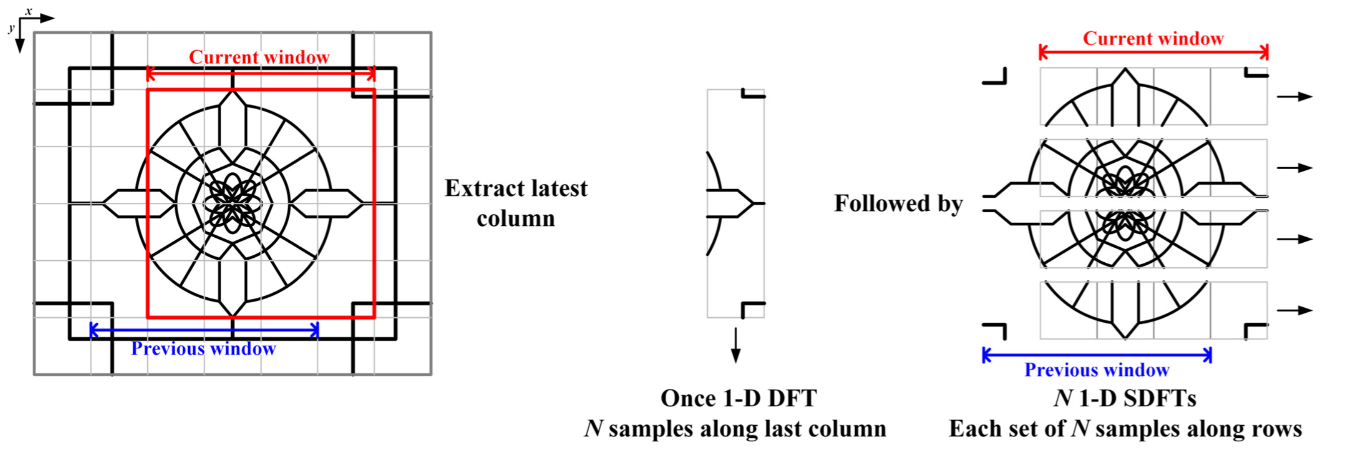
Equation (9) is easily implemented by column-row DFT operations [28] as shown in Fig. 3 (a). The overall processes can be calculated as $2N$ times the length-of-N 1-D DFT computation, and it takes ($2N \times N^2$) complex multiplications and ($2N \times N^2$) complex additions for each complex-input sample. This column-row-based 2-D algorithm has more computational complexity than the vector-radix 2-D FFT algorithm [25], but it can be easily simplified into a 1-D computation. In addition, it also gains the advantage of reducing the complexity of memory access during the transform process and can also be applied to achieve parallel processing on a broadcast mode multiprocessor [29].

We next consider the sliding transform process and assume that the sliding window is shifted in the horizontal direction. It is already known that the first and the last samples are shifted between the current window and the pre-window for the 1-D SDFT process [as Fig. 3 shown in [1]]. However, the first column information for the current window and the removal of the last column information in the pre-window are required for the 2-D SDFT process, as shown on the left side of Fig 3 (b). In the column-row-based case, the 2-D sliding process is equivalent to perform DFT computation once along one new column followed by the N-point SDFT along the N rows, as shown in Fig 3 (b).

Here, we make another assumption: that the 1-D sliding window is shifted in the vertical direction for each column. Simultaneously, we apply the concept of SDFT to substitute once 1-D DFT calculation on the left side of Fig. 3 (b). This implies that the column-row-based 2-D SDFT is simplified into (N+ 1) times the 1-D SDFT computations. Hence, the proposed DDM has been successfully done.


x
y
Split into
N columns
Followed by
N 1-D DFTs
Each set of N samples along columns
N 1-D DFTs
Each set of N samples along rows

(a)



(b)

FIGURE 3. **The schematic principle of column-row-based 2-D DFT. (a) Without the sliding property. (b) With the sliding property.**

B. MIXED-RADIX BUTTERFLY-BASED 1-D SDFT STRUCTURE

Let x_n be a time-varying signal at time index n; the k-th spectral bin of an N-point 1-D DFT standard formula is defined as

$$X_n(k) = \sum_{m=0}^{N-1} x_{\widehat{n}+m} W_N^{-km}, \tag{10}$$

where $\widehat{n} = n - N + 1$ and the twiddle factor (W_N) are same as (2). Assume that the transform length of N is restricted to be a power-of-two integer, and the number can be divisible by four as

$$N = 2^r = 4Q, \tag{11}$$

where the parameter r is more than or equal to two. Using variable substitution, the index m is represented by

$$m = Qm_1 + m_2 \begin{cases} m_1 = 0, 1, 2, 3. \\ m_2 = 0, 1, \ldots, Q-1. \end{cases} \tag{12}$$

By taking (12) into (10), the DFT formula can be further rewritten as follows:

$$\begin{aligned} X_n(k) &= \sum_{m_2=0}^{Q-1} \sum_{m_1=0}^{3} x_{\widehat{n}+Qm_1+m_2} W_N^{-k(Qm_1+m_2)} \\ &= \sum_{m_2=0}^{Q-1} \sum_{m_1=0}^{3} x_{\widehat{n}+Qm_1+m_2} W_N^{-kQm_1} W_N^{-km_2} \\ &= \sum_{m_2=0}^{Q-1} P_{n,m_2}(k) W_N^{-km_2}, \end{aligned} \tag{13}$$

where

$$P_{n,m_2}(k) = \sum_{m_1=0}^{3} x_{\widehat{n}+Qm_1+m_2} W_4^{-km_1}. \tag{14}$$

Let us apply the properties of the twiddle factor and the modulo function, which are expressed as (15) and (16), respectively. Equation (14) can be simplified as (17). It is treated as a 4-point DFT and can be realized using one butterfly radix-4 DIT-FFT algorithm [30].

$$W_N^{n+N} = W_N^n. \tag{15}$$

$$ab \bmod n = [(a \bmod n)\,(b \bmod n)] \bmod n. \quad (16)$$

$$P_{n,m_2}\left(\widehat{k}\right) = \sum_{m_1=0}^{3} x_{\widehat{n}+Qm_1+m_2} W_4^{-\widehat{k}m_1}, \quad (17)$$

where $\widehat{k}$ is the results of the modulo operation, and its value ranging from 0 to 3 can be represented as

$$\widehat{k} = k \bmod 4. \quad (18)$$

By using a data re-mapping scheme, it is very easy to generate a radix-4 DFT computation. It is worth noting that there are no extra complex multiplications for calculating either (14) or (18).

A traditional radix-2 DIT-FFT algorithm is then adopted, where the index m2 can be further divided into even-numbered and odd-numbered parts. Based on the decomposition approach [31], the resultant formula is given by

$$\begin{aligned} X_n(k) &= \sum_{m_2=0}^{Q-1} P_{n,m_2}\left(\widehat{k}\right) W_N^{-km_2} \\ &= \sum_{m_2 even} P_{n,m_2}\left(\widehat{k}\right) W_N^{-km_2} + \sum_{m_2 odd} P_{n,m_2}\left(\widehat{k}\right) W_N^{-km_2} \\ &= \sum_{p=0}^{(Q/2)-1} P_{n,2p}\left(\widehat{k}\right) W_N^{-k2p} \\ &\quad + \sum_{p=0}^{(Q/2)-1} P_{n,2p+1}\left(\widehat{k}\right) W_N^{-k(2p+1)} \\ &= \ddot{L}_n(k) + W_N^{-k}\dot{L}_n(k), \end{aligned} \quad (19)$$

where $\ddot{L}_n(k)$ and $\dot{L}_n(k)$ are, respectively, defined as the even- and odd-parts of the interleaved sequences, as shown in (20) and (21).

$$\ddot{L}_n(k) = \sum_{p=0}^{(Q/2)-1} P_{n,2p}\left(\widehat{k}\right) W_{N/2}^{-kp}. \quad (20)$$

$$\dot{L}_n(k) = \sum_{p=0}^{(Q/2)-1} P_{n,2p+1}\left(\widehat{k}\right) W_{N/2}^{-kp}. \quad (21)$$

Then, we employ the properties of the shifted window, i.e., equation (20) can be further derived as (22) by using the intermediate calculations from previous windows.

$$\begin{aligned} \ddot{L}_n(k) &= \sum_{p=0}^{(Q/2)-1} P_{n,2p}\left(\widehat{k}\right) W_{N/2}^{-kp} \\ &= \sum_{p=0}^{(Q/2)-1} P_{n-1,2p+1}\left(\widehat{k}\right) W_{N/2}^{-kp} \\ &= \dot{L}_{n-1}(k). \end{aligned} \quad (22)$$

The above results show that the even parts [see (20)] can be obtained by exploiting the delayed calculations of the odd parts [see (21)]. After taking (22) into (19), the N-point DFT formula of $\dot{L}_n(k)$'s has the following relationship

$$X_n(k) = \dot{L}_{n-1}(k) + W_N^{-k}\dot{L}_n(k). \quad (23)$$

In addition, in order to fully understand the sequential decimation operations during the next decimation process, we define $L_n^s(k)$ as the k-th bin of the $Q/2^s$-point interleaved sequence transform at the decimation stage s, which yields

$$L_n^s(k) = \sum_{p=0}^{(Q/2^s)-1} P_{n,2^s(p+1)-1}\left(\widehat{k}\right) W_{N/2^s}^{-kp}, \quad (24)$$

where the superscript s ranges from 0 to $\log_2 Q - 1$. The notation $L_n^s(k)$ can be further used to instead of $\dot{L}_n(k)$, and equation (23) can be generally rewritten as

$$\begin{aligned} X_n(k) &= L_n^0(k) \\ &= L_{n-1}^1(k) + W_N^{-k}L_n^1(k). \end{aligned} \quad (25)$$

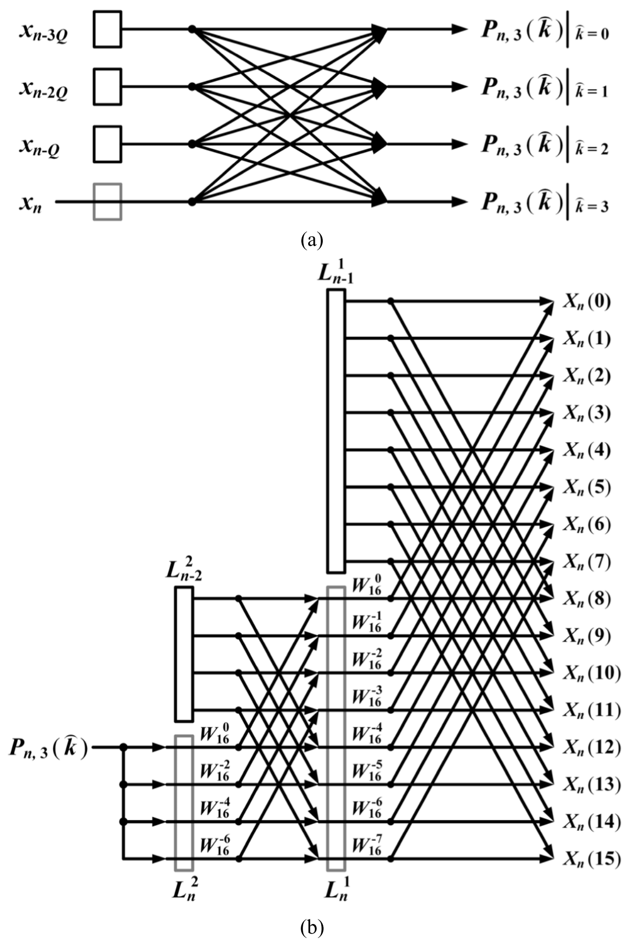
The interleaved decimation process is repeated at each decimation stage until the sequences cannot be decimated. For example, $L_n^1(k)$ can be decimated as

$$\begin{aligned} L_n^1(k) &= \sum_{p=0}^{(Q/2)-1} P_{n,2p+2-1}\left(\widehat{k}\right) W_{N/2}^{-kp} \\ &= \sum_{p=0}^{(Q/4)-1} P_{n,4p+1}\left(\widehat{k}\right) W_{N/2}^{-k2p} \\ &\quad + \sum_{p=0}^{(Q/4)-1} P_{n,4p+2+1}\left(\widehat{k}\right) W_{N/2}^{-k(2p+1)} \\ &= \sum_{p=0}^{(Q/4)-1} P_{n-2,4p+3}\left(\widehat{k}\right) W_{N/4}^{-kp} \\ &\quad + W_{N/2}^{-k}\sum_{p=0}^{(Q/4)-1} P_{n,4p+3}\left(\widehat{k}\right) W_{N/4}^{-kp} \\ &= L_{n-2}^2(k) + W_{N/2}^{-k}L_n^2(k). \end{aligned} \quad (26)$$

Based on above derivations, we can obtain the recursive formula of $L_n^s(k)$ during the sliding transform process, which is given as follows:

$$L_n^s(k) = L_{n-2^s}^{s+1}(k) + W_{N/2^s}^{-k}L_n^{s+1}(k). \quad (27)$$

According to (17), (24), (25), and (27), the derivations can be easily mapped into a novel algorithm. The flow graph is shown in Fig. 4. Note that the function of the gray rectangle is similar to the sample-and-hold operation, where the value can be used to update the intermediate result ($L_{n-2^s}^{s+1}(k)$) in the sliding transform scenario. The black rectangle can be directly viewed as a group of registers, which is stored the pre-calculated intermediate results of previous windows.



x_{n-3Q}
x_{n-2Q}
x_{n-Q}
x_n
$P_{n,3}(\hat{k})|_{\hat{k}=0}$
$P_{n,3}(\hat{k})|_{\hat{k}=1}$
$P_{n,3}(\hat{k})|_{\hat{k}=2}$
$P_{n,3}(\hat{k})|_{\hat{k}=3}$
(a)
L_{n-1}^1
L_{n-2}^2
$P_{n,3}(\hat{k})$
W_{16}^0 W_{16}^{-2} W_{16}^{-4} W_{16}^{-6}
W_{16}^0 W_{16}^{-1} W_{16}^{-2} W_{16}^{-3} W_{16}^{-4} W_{16}^{-5} W_{16}^{-6} W_{16}^{-7}
L_n^2
L_n^1
$X_n(0)$ … $X_n(15)$
(b)

FIGURE 4. **Proposed forward-path structure of the 1-D SDFT with a mixed-radix structure for "$N = 2^2 \times 4$".**

IV. OVERALL COMPLEXITY ANALYSIS OF THE PROPOSED 1-D/2-D SDFT ALGORITHM

Next, the computational complexity of the proposed algorithm is considered. For simplicity of analysis, we assume the input is a complex sequence and that intermediate calculations of previous windows already exist. The overall steps of the proposed algorithm are listed as follows:

Step 1: Make the 4-point DFT calculation based on the radix-4 structure. Here, eight complex additions are required.

Step 2: Compute the intermediate calculations at the last decimation stage, i.e., $s = \log_2 Q - 1$. Furthermore, the real and imaginary parts of the twiddle factor $W_N^{-N/8}$ and $W_N^{-3N/8}$ are identical. Here, one complex multiplication can be treated as two real additions and two real multiplications. In addition, eight complex additions, four real additions, and four real multiplications are required.

Step 3: Compute $L_n^s(k)$ using the recursive relationship in (27) at remaining decimation stage except when $s = 0$. Here, $N/2^s$ complex additions and $N/2^{(s+1)}$ complex multiplications are required at each stage. Thus, the number of the computational complexities totals $(N - 16)$ complex additions and $(N/2 - 8)$ complex multiplications.

Step 4: According to (25), the DFT bins calculated using $L_n^1(k)$ and $L_{n-1}^1(k)$, where $L_{n-1}^1(k)$ is pre-calculated in the previous window. Here, (N) complex additions and $(N/2)$ complex multiplications are required.

TABLE 1. **Pseudo code for the proposed forward-path 1-D SDFT algorithm.**

```
Input: data sequences $x_n$, the pre-calculated intermediate
results $L_{n-2^s}^{s+1}(k)$.
Output: the updated intermediate results $L_n^s(k)$, DFT bins $X_n(k)$.
/* Step 1. $C_A$ = 8 */
Compute $P_{n,m_2}(\hat{k})$ using $x_n$ [see (17)]

/* Step 2. $C_A$ = 8, $R_M$ = 4, $R_A$ = 4 */
Compute $L_n^{\log_2 Q-1}(k)$ using $P_{n,m_2}(\hat{k})$

/* Step 3. $C_M$ = $N/2 - 8$, $C_A$ = $N - 16$ */
for $s$ = 1 to $\log_2 Q - 2$ do
    Compute $L_n^s(k)$ using the recursive relationship [see (27)]
end for

/* Step 4. $C_M$ = $N/2$, $C_A$ = $N$ */
Compute $X_n(k)$ using $L_n^1(k)$ and $L_{n-1}^1(k)$

return $X_n(k)$
```

Table 1 shows a pseudo code for the proposed forward-path 1-D SDFT algorithm, where R_A, R_M, C_A, and C_M are, respectively, the numbers of real additions, real multiplications, complex additions, and complex multiplications.

The overall computation requirement of the proposed forward-path 1-D SDFT is summarized as

$$R_A = 6N - 12 \tag{28}$$

and

$$R_M = 4N - 28, \tag{29}$$

where one complex addition (C_A) involves two real additions (R_A), and one complex multiplication (C_M) involves four real multiplications (R_M) and two R_A.

As mentioned in the end of part A of Section III, the column-row-based 2-D SDFT can be simplified into $(N + 1)$ times 1-D SDFT computations. If the proposed method is applied to the 2-D SDFT structures, the overall computation requirement is counted as

$$R_A = 6N^2 - 6N - 12 \tag{30}$$

and

$$R_M = 4N^2 - 24N - 28. \tag{31}$$

In this section, a new approach to the SDFT algorithm based on a blend of the radix-2 and radix-4 structures is proposed. The proposed method has the same precision as that of the traditional FFT due to the fact that the proposed structure is quite similar to the traditional butterfiy-based FFT algorithm. Furthermore, the forward-path 1-D SDFT is guaranteed to be stable due to the presence of non-feedback loops. This means that, compared with the recursive Park's structure, accumulated imprecise twiddle factor errors do not occur regardless of whether the proposed 1-D or 2-D SDFT algorithm are used, [26].

V. DISCUSSION AND COMPARISON

A. COMPUTATIONAL COMPLEXITY

For the sliding issues in various 2-D SDFT algorithms, the performance metrics in terms of number of multiplications and additions are evaluated in Table 2, where R_M and R_A denote the number of real multiplications and additions, respectively.

TABLE 2. Computational complexity comparison of *N*-by-*N* point-related DFT algorithms for 2-D sliding applications.

Algorithm	Architecture	Operation	Number of operations
DFT [28]	Direct Computation	R_M	$8N^3$
		R_A	$8N^3 - 4N^2$
FFT [31]	Radix-2	R_M	$4N^2 \log_2 N$
		R_A	$6N^2 \log_2 N$
VRFFT [25]	Radix-2x2	R_M	$3N^2 \log_2 N$
		R_A	$5.5N^2 \log_2 N$
2-D SDFT [26,[33]	Recursive	R_M	$4N^2 + 4N$
		R_A	$4N^2 + 3N + 2$
VR-SFFT [32]	Radix-2x2	R_M	$4N^2 - 4N$
		R_A	$8N^2 - 8N$
Proposed	Mixed-radix	R_M	$4N^2 - 24N - 28$
		R_A	$6N^2 - 6N - 12$

To provide a fair comparison with other algorithms [25], [26], [28], [31], [32], one complex multiplication (C_M) is equivalent to two real additions and four real multiplications, and one complex addition (C_A) is equivalent to two real additions. In order to conveniently observe the trend in the growth rate of R_M and R_A, Fig. 5 shows the amount of computation required for various 2-D SDFT algorithms with varying window size ($N \times N$). For example, if the computational complexity for a window size of 16 × 16, the proposed algorithm shows that the number of additions and multiplications are 1,428 and 612, representing 74.64% and 80.08% reductions compared to the VRFFT, respectively. The Park's algorithm [26] has the lowest number of additions ($R_A = 1,074$) but requires the highest number of multiplications ($R_M = 1,088$). Overall, it is still more computationally complex than the proposed algorithm.

For the detailed comparison of computational complexity, the proposed algorithm totally takes ($4N^2 - 24N - 28$) multiplications and ($6N^2 - 6N - 12$) additions. In Byun *et al.* [Table 1, 32], various algorithms and their combinations were compared to the proposed VR-2 × 2 SFFT algorithm. The results clearly showed that their algorithm requires ($4N^2 - 4N$) multiplications and ($8N^2$ - $8N$) additions. Therefore, the computational complexity of Byun *et al.* [32]

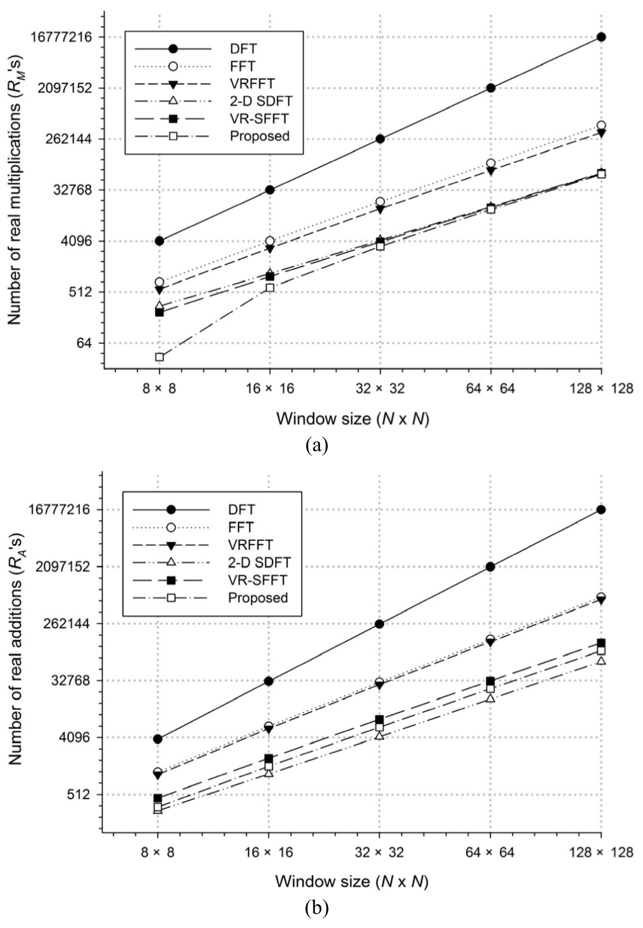


FIGURE 5. The trend growth rate of computational complexity with varying window sizes for various 2-D sliding approaches. (a) R_M. (b) R_A.

is more complex than that of this work. Moreover, Kuchan *et al.* algorithm [33] has the same complexity as well as 2-D SDFT [26]. It requires ($4N^2 - 4N$) multiplications and ($2N^2 + 3N + 2$) additions. Conceptually, the complexity of multiplication is quite higher than that of addition. That is the reason that the proposed method has better processing time saving than Kuchan *et al.* [33] and 2-D SDFT [26] as shown in Fig. 6. Although these approaches [26, 33] have very similar derivations, the proposed algorithm still has its individual contribution based on the derivations of DDM-based 2-D SDFT and mixed-radix butterfly-based 1-D SDFT structure.

B. EXECUTION TIME

It is known that experimental simulations are a better aggregative index for both multiplication and addition than other alternatives. Therefore, a simulation environment is applied to measure the processing time of the 2-D SDFT. Here, the simulation environment is set as follows: Various algorithms were performed on a 64-bit Windows 7 system with an Intel i7 3.60 GHz processor with 16.0 GB random access memory (RAM). The window size was set to 16 × 16; a practical SVC video was adopted for the test signal [34], and the formats included the CIF and 4CIF with 30 fps. In order to shorten the simulation time, the sliding process was only

TABLE 3. Execution time analysis of various algorithms with different CIF sequences for 2-D sliding applications.

Sequences	DFT	FFT	VRFFT	2-D SDFT	Proposed
	Time (ms)	Time (ms)	Time (ms)	Time (ms)	Time (ms)
CITY	791994.35	103761.09	86966.94	29317.35	26145.49
CREW	791921.49	103773.20	87096.69	29370.61	26147.58
SOCCER	791736.29	103713.64	87334.97	29325.44	26144.80
HARBOUR	791841.80	103763.22	87149.21	29352.84	26156.14
Average	791873.48	103752.79	87136.95	29341.56	26148.50

TABLE 4. Execution time analysis of various algorithms with different 4CIF sequences for 2-D sliding applications.

Sequences	DFT	FFT	VRFFT	2-D SDFT	Proposed
	Time (ms)	Time (ms)	Time (ms)	Time (ms)	Time (ms)
CITY	3170184.43	413424.29	347829.00	117329.30	104487.59
CREW	3172548.82	414053.99	348393.49	117336.27	104499.69
SOCCER	3177551.54	414256.53	348347.26	117480.16	104608.80
HARBOUR	3171558.25	413563.16	348009.11	117201.30	104446.47
Average	3172960.76	413824.49	348144.72	117336.76	104510.64

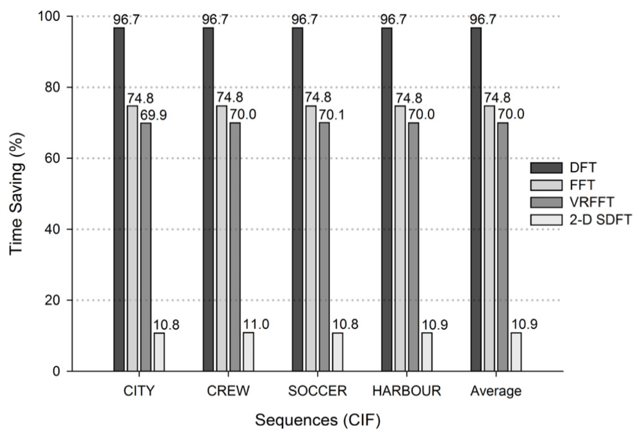


FIGURE 6. Time saving analysis of various algorithms with different CIF sequences for 2-D sliding applications.

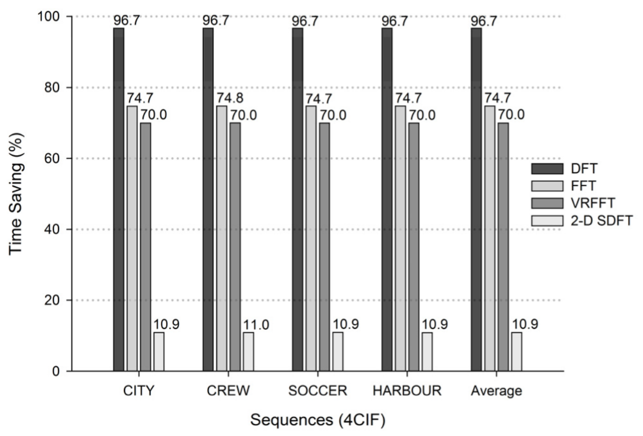


FIGURE 7. Time saving analysis of various algorithms with different 4CIF sequences for 2-D sliding applications.

performed on the grayscale component, and we focused on the time of the first 100 frames for each test sequence.

First, we measured the processing time of each algorithm under the CIF sequences, for which the results are shown in Table 3. Simultaneously, the time savings as shown in Fig. 6 was calculated using the formula in (32).

$$TimeSaving\,(\%) = \frac{T_{\text{Algorithm}} - T_{\text{Proposed}}}{T_{\text{Algorithm}}} \times 100 \qquad (32)$$

where $T_{\text{Algorithm}}$ and T_{Proposed} denote the measured processing time of other algorithms and the proposed method, respectively. The results show that the execution time for this work can be reduced by 96.7%, 74.8%, 70%, and 10.9%, respectively, compared with the DFT, FFT, VRFFT, and 2-D SDFT. In summary, this work consistently outperforms the other methods under consideration in terms of processing time.

We next change the test format from CIF to 4CIF. Under the same simulation conditions, the time consumed in the various sequences is recorded in Table 4. The time savings results are provided in Fig. 7. For all the sequences, the execution time reduction, respectively, was 96.7%, 74.7%, 70%, and 10.9%, compared with [25], [26], [28], [31]. According to the Fig. 6 and Fig. 7, it shows that this work achieves significant time savings and has a quite consistent performance result based on the simulations of CIF and 4CIF sequences.

VI. CONCLUSION

This paper presented a low-complexity, low-execution-time 2-D sliding DFT algorithm for a time-frequency analyzer design. The proposed 2-D SDFT employs a descending dimension method and a mixed-radix structure to achieve a faster sliding transform process. The results indicate that the proposed method had superior performance compared to other approaches and thus provides a new direction in which to apply these fast transforms to the applications of biomedical/speech signal processing, computer vision, and image recognition/processing.

ACKNOWLEDGMENT

The authors would like to thank Prof. Ching-Hsing Luo and Shuenn-Yuh Lee for their kind provision of some suggestions related to this research topic.

REFERENCES

[1] E. Jacobsen and R. Lyons, "The sliding DFT," *IEEE Signal Process. Mag.*, vol. 20, no. 2, pp. 74–80, Mar. 2003.

[2] K. Duda, "Accurate, guaranteed stable, sliding discrete Fourier transform," *IEEE Signal Process. Mag.*, vol. 27, no. 6, pp. 124–127, Nov. 2010.

[3] C.-S. Park, "Fast, accurate, and guaranteed stable sliding discrete Fourier transform," *IEEE Signal Process. Mag.*, vol. 32, no. 4, pp. 145–156, Jul. 2015.

[4] C.-S. Park, "Guaranteed-stable sliding DFT algorithm with minimal computational requirements," *IEEE Trans. Signal Process.*, vol. 65, no. 20, pp. 5281–5288, Oct. 2017.

[5] C.-S. Park and S.-J. Ko, "The hopping discrete Fourier transform," *IEEE Signal Process. Mag.*, vol. 31, no. 2, pp. 135–139, Mar. 2014.

[6] Q. Wang, X. Yan, and K. Qin, "High-precision, permanently stable, modulated hopping discrete Fourier transform," *IEEE Signal Process. Lett.*, vol. 22, no. 6, pp. 748–751, Jun. 2015.

[7] W.-H. Juang, S.-C. Lai, K.-H. Chen, W.-K. Tsai, and C.-H. Luo, "Low-complexity hopping DFT design based on a compact recursive structure," *Electron. Lett.*, vol. 53, no. 1, pp. 25–27, Jan. 2017.

[8] W.-H. Juang, S.-C. Lai, C.-H. Luo, and S.-Y. Lee, "VLSI architecture for novel hopping discrete Fourier transform computation," *IEEE Access*, vol. 6, pp. 30491–35000, May 2018.

[9] E. S. Gower and M. O. J. Hawksford, "Acoustic echo cancellation by minimising mutual information within sliding DFT window," *Electron. Lett.*, vol. 49, no. 25, pp. 1585–1586, Dec. 2013.

[10] C. Shitole and P. Sumathi, "Sliding DFT-based vibration mode estimator for single-link flexible manipulator," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 6, pp. 3249–3256, Dec. 2015.

[11] K. M. Singh, "Simultaneous estimation of moving-vibration parameters by sliding goertzel algorithm in PLL technique," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 2, pp. 334–343, Feb. 2019.

[12] M. A. Moussa, M. Boucherma, and A. Khezzar, "A detection method for induction motor bar fault using sidelobes leakage phenomenon of the sliding discrete Fourier transform," *IEEE Trans. Power Electron.*, vol. 32, no. 7, pp. 5560–5572, Jul. 2017.

[13] S. Liu and T. Guo, "An adaptive DFT algorithm for measuring power system synchrophasors based on rectangular coordinate," in *Proc. IEEE PES Asia–Pacific Power Energy Eng. Conf. (APPEEC)*, Nov. 2015, pp. 1–5. doi: 10.1109/APPEEC.2015.7380900.

[14] S. Liu, T. Guo, J. Li, and J. Wu, "Adaptive DFT algorithm for measuring synchrophasors based on shifting window symmetrical," in *Proc. Int. Conf. Electr. Utility Deregulation Restruct. Power Technol. (DRPT)*, Nov. 2015, pp. 722–726. doi: 10.1109/DRPT.2015.7432339.

[15] W. A. Sethares, "Short-time Fourier transform," in *Rhythm and Transforms*. New York, NY, USA: Springer, 2007, pp. 122–124.

[16] I.-J. Ding, "Enhancement of speech recognition using a variable-length frame overlapping method," in *Proc. Int. Symp. Comput., Commun., Control Autom. (3CA)*, Jul. 2010, pp. 375–377. doi: 10.1109/3CA.2010.5533800.

[17] M. A. Ali, M. Hossain, and M. N. Bhuiyan, "Automatic speech recognition technique for Bangla words," *Int. J. Adv. Sci. Technol.*, vol. 50, pp. 51–59, Jan. 2013.

[18] P. Grosche and M. Müller, "Extracting predominant local pulse information from music recordings," *IEEE Trans. Audio, Speech, Language Process.*, vol. 19, no. 6, pp. 1688–1701, Aug. 2011.

[19] S. L. Tanimoto, "An optimal algorithm for computing Fourier texture descriptors," *IEEE Trans. Comput.*, vol. C-27, no. 1, pp. 81–84, Jan. 1978.

[20] F. Lehmann, "Turbo segmentation of textured images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 16–29, Jan. 2011.

[21] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising with block-matching and 3D filtering," *Proc. SPIE*, vol. 6064, pp. 606414-1–606414-12, Feb. 2006.

[22] A. Meraoumia, S. Chitroub, and A. Bouridane, "Efficient person identification by fusion of multiple palmprint representations," in *Proc. Int. Conf. Image Signal Process.*, Jan. 2010, pp. 182–191. doi: 10.1007/978-3-642-13681-8_22.

[23] P. D. Chazal, J. Flynn, and R. B. Reilly, "Automated processing of shoeprint images based on the Fourier transform for use in forensic science," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 3, pp. 341–350, Mar. 2005.

[24] M. Ghafoor, I. A. Taj, and M. N. Jafri, "Fingerprint frequency normalisation and enhancement using two-dimensional short-time Fourier transform analysis," *IET Comput. Vis.*, vol. 10, no. 8, pp. 806–816, Dec. 2016.

[25] K. R. Rao, D. N. Kim, and J. J. Hwang, "Vector-radix 2D-FFT algorithms," in *Fast Fourier Transform—Algorithms and Applications*. New York, NY, USA: Springer, 2011, pp. 185–193.

[26] C. S. Park, "2D discrete Fourier transform on sliding windows," *IEEE Trans. Image Process.*, vol. 24, no. 3, pp. 901–907, Mar. 2015.

[27] E. Chu and A. George, "The mixed-radix and split-radix FFTs," in *Inside the FFT Black Box: Serial and Parallel Fast Fourier Transform Algorithms*. Boca Raton, FL, USA: CRC Press, 2000, pp. 111–116.

[28] K. R. Rao, D. N. Kim, and J. J. Hwang, "Two-dimensional discrete Fourier transform," in *Fast Fourier Transform—Algorithms and Applications*. New York, NY, USA: Springer, 2011, pp. 127–131.

[29] R. Tolimieri, M. An, and C. Lu, "Parallel implementation of FFT," in *Mathematics of Multidimensional Fourier Transform Algorithms*. New York, NY, USA: Springer-Verlag, 1993, pp. 163–164.

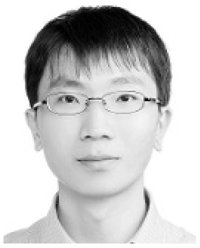
[30] R. Neuenfeld, M. Fonseca, and E. Costa, "Design of optimized radix-2 and radix-4 butterflies from FFT with decimation in time," in *Proc. IEEE 7th Latin Amer. Symp. Circuits Syst. (LASCAS)*, Apr. 2016, pp. 171–174. doi: 10.1109/LASCAS.2016.7451037.

[31] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2010.

[32] K.-Y. Byun, C.-S. Park, J.-Y. Sun, and S.-J. Ko, "Vector radix 2×2 sliding fast Fourier transform," *Math. Problems Eng.*, vol. 2016, Dec. 2016, Art. no. 2416286.

[33] A. Kuchan, D. J. Tuptewar, S. S. Anwar, and S. P. Bandewar, "Sliding discrete Fourier transform for 2D signal processing," in *Proc. Int. Conf. ISMAC Comput. Vis. Bio-Eng.* Cham, Switzerland: Springer, 2018, pp. 1339–1346. doi: 10.1007/978-3-030-00665-5_126.

[34] W. Song, D. Tjondronegoro, and S. Azad, "User-centered video quality assessment for scalable video coding of H.264/AVC standard," in *Proc. 16th Int. Multimedia Modeling Conf.*, Jan. 2010, pp. 55–65. doi: 10.1007/978-3-642-11301-7_9.



WEN-HO JUANG was born in Taiwan. He received the M.S. degree from the Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, in 2010, and the Ph.D. degree from National Cheng Kung University, in January 2019. From October 2011 to September 2014, he was an Engineer with the Data Center Networking BU, MediaTek Inc., Hsinchu, Taiwan. He is currently an Engineer with Touch BU, Himax Technologies Inc., Tainan. He is also an Adjunct Assistant Professor with the Department of Computer Science and Information Engineering, Nanhua University, Chiayi, Taiwan. His research interests include digital signal processing and VLSI system design.

SHIN-CHI LAI was born in Taichung, Taiwan, in 1980. He received the B.S. degree in electronic engineering from ChienKuo Technology University, Changhua, Taiwan, in 2002, the M.S. degree in electronic engineering from the National Yunlin University of Science and Technology, Yunlin, Taiwan, in 2005, and the Ph.D. degree from National Cheng Kung University, Tainan, Taiwan, in 2011, where he was an Assistant Research Fellow with the Department of Electrical Engineering, from October 2011 to July 2013. From August 2013 to July 2016, he was an Assistant Professor with the Department of Computer Science and Information Engineering, Nanhua University, Chiayi, Taiwan, where he has been an Associate Professor, since August 2016. His main research interest includes signal processing and its circuit design, especially for speech, audio, biomedical, and multimedia applications.