

Article

The Algorithm and Structure for Digital Normalized Cross-Correlation by Using First-Order Moment [†]

Chao Pan ¹, Zhicheng Lv ¹, Xia Hua ^{2,*} and Hongyan Li ¹

¹ School of Information and Communication Engineering, Hubei University of Economics, Wuhan 430205, China; PanChao@hbue.edu.cn (C.P.); lion112@139.com (Z.L.); hongyanli2000@126.com (H.L.)

² School of Electrical and Information Engineering, Wuhan Institute of Technology, Wuhan 430205, China

* Correspondence: hedahuaxia05021046@163.com

[†] This paper is an extended version of our paper published in the First International Symposium on Future ICT (Future-ICT 2019) in Conjunction with 4th International Symposium on Mobile Internet Security (MobiSec 2019).

Received: 1 February 2020; Accepted: 25 February 2020; Published: 1 March 2020



Abstract: Normalized cross-correlation is an important mathematical tool in digital signal processing. This paper presents a new algorithm and its systolic structure for digital normalized cross-correlation, based on the statistical characteristic of inner-product. We first introduce a relationship between the inner-product in cross-correlation and a first-order moment. Then digital normalized cross-correlation is transformed into a new calculation formula that mainly includes a first-order moment. Finally, by using a fast algorithm for first-order moment, we can compute the first-order moment in this new formula rapidly, and thus develop a fast algorithm for normalized cross-correlation, which contributes to that arbitrary-length digital normalized cross-correlation being performed by a simple procedure and less multiplications. Furthermore, as the algorithm for the first-order moment can be implemented by systolic structure, we design a systolic array for normalized cross-correlation with a seldom multiplier, in order for its fast hardware implementation. The proposed algorithm and systolic array are also improved for reducing their addition complexity. The comparisons with some algorithms and structures have shown the performance of the proposed method.

Keywords: normalized cross-correlation; fast algorithm; first-order moment; systolic array; multiplication complexity

1. Introduction

Normalized cross-correlation (NCC) is an important mathematical tool in signal and image processing for feature matching, similarity analysis, motion tracking, object recognition, and so on [1–3]. In order to improve its real-time and efficient performance, digital NCC has been suggested to be implemented by some fast algorithms and hardware structures, due to its high computational complexity [4,5].

Nowadays, since correlation and convolution have similar computation structures, there are mainly three kinds of fast convolution algorithms can be applied for fast NCC [6,7]: (1) the Fast Fourier Transform (FFT)-based algorithm, (2) the polynomial-based algorithm, (3) the decomposition algorithm. However, to our knowledge, each of these algorithms has its applicable limitations. The FFT-based algorithm is not well-suited to the discrete domain. Plus, it involves with complex multiplications [8,9]. Both the polynomial-based algorithm and the decomposition algorithm require complex computational structures, and they often lack commonality for arbitrary-length correlations [10,11].

Furthermore, some special algorithms for fast NCC have been presented [12,13]. The fast cross-correlation of binary sequences can be extended to other types of NCC sequences [14]. The

estimation algorithm derives the scaling factor between the signal and the kernel, so it computes NCC using only additions at the cost of small noise [15]. Several methods have been used to assist NCC for reducing its searching and computing times in image matching, such as the pyramid method [3,7]. In addition, many parallel algorithms of the inner-product have been published that can perform fast cross-correlation for NCC [16,17], where the Distributed Arithmetic (DA) with look-up table has not multiplication, but needs much Read-Only Memory (ROM) [18].

To hardware implementation of fast NCC, Very-Large-Scale Integration (VLSI) circuits have been applied, where systolic structures are popular due to their regularity and modularity [19–21]. The integration of the systolic array and the DA technique lead to more efficient VLSI implementation of cross-correlation, although they use many ROMs and address decoders [22,23]. The Residue Number System-based DA can reduce ROMs and enhance throughput, while extra encoding processes in the residue domain are necessary [24].

In this paper, we present a new algorithm and structure to implement digital NCC with a simple and fast procedure. It is a breakthrough that an NCC formula expressed in terms of a first-order moment is designed according to the relationship between the inner-product and the first-order moment, so the computational complexity of NCC is transformed into that of a first-order moment. For performing an arbitrary-length digital NCC, our algorithm would first establish the NCC formula based on a first-order moment for correlation sequences, and then introduce a fast algorithm without multiplication from [25,26] to compute this first-order moment in the new NCC formula rapidly. For the hardware implementation of NCC, we develop a simple and scalable systolic array derived from the proposed algorithm, due to the fact that the fast algorithm for the first-order moment is easily performed by systolic structure [27]. The proposed algorithm and systolic array are also improved to reduce their addition complexity, according to an even-odd relationship in the computation of the first-order moment.

The rest of the paper is organized as follows. Section 2 establishes the NCC formula based on a first-order moment. Section 3 introduces a fast algorithm and its systolic implementation for first-order moment. Sections 4 and 5 discuss the fast algorithm and the systolic array inspired by Section 3 to perform the NCC formula in Section 2 rapidly. Comparison and analysis are presented in Section 6 to demonstrate the feasibility of the proposed algorithm and structure. Finally, Section 7 gives the conclusion.

2. Normalized Cross-Correlation Based on First-Order Moment

Being the most complex operation in NCC, the inner-product of two correlation sequences would be transformed into a first-order moment for decreasing computational complexity in fast NCCs. To do this, let us assume two N -point digital sequences $\{f(i)\}$ and $\{g(i)\}$, where $\{f(i)\}$ is an arbitrary input sequence, and $\{g(i)\}$ is the fixed correlation kernel with the value range $g(i) \in \{0, 1, 2, \dots, L\}$. This section establishes an NCC formula for these two sequences that mainly includes a first-order and a zero-order moment. The aim is to replace the complex computation of cross-correlation in NCC with an easy computation of a first-order moment.

2.1. Cross-Correlation

Cross-correlation is an inner-product between two digital sequences. It is defined as

$$c(n) = f(n) \circ g(n) = \sum_{i=0}^{N-1} f(n+i)g(i) \quad (1)$$

Using mathematical transformation, this Equation (1) could be transformed into a first-order moment by means of the statistical characteristics of the inner-product operation. To do this, we define

some subsets S_k ($k = 0, 1, 2, \dots, L$) that divide the index set $i \in \{0, 1, \dots, N-1\}$ into L subsets, depending on the max value in the correlation kernel $\{g(i)\}$. Specifically,

$$S_k = \{i \mid g(i) = k, \quad i \in \{0, 1, 2, \dots, N-1\}\} \quad (2)$$

where $k = 0, 1, 2, \dots, L$. In other words, S_k is a set of indices i that corresponds to $g(i) = k$ in actual. Then a new $(L+1)$ -point sequence $\{a_k(n)\}$ is defined by subsets S_k [28], which is

$$a_k(n) = \begin{cases} \sum_{i \in S_k} f(n+i) & \text{where } S_k \neq \Phi \\ 0 & \text{otherwise} \end{cases}, \quad (3)$$

where $k = 0, 1, 2, \dots, L$, and " Φ " denotes an empty set.

The $a_k(n)$ could be acted as the sum of elements in the sequence $\{f(n+i)\}$ while the parameter i corresponds to $g(i) = k$. The computation of the $\{a_k(n)\}$ is actually a statistics procedure for counting how much k would be accumulated in the computation of the $c(n)$. Therefore, the relationship between $\{f(n+i)\}$ and $\{a_k(n)\}$ can be described as:

$$\sum_{i=0}^{N-1} f(n+i) = \sum_{k=0}^L a_k(n), \quad (4a)$$

$$\sum_{i=0}^{N-1} f(n+i)g(i) = \sum_{k=0}^L a_k(n)k = \sum_{k=1}^L a_k(n)k \quad (4b)$$

It is obvious that $\sum_{k=1}^L a_k(n)$ in Equation (4a) is a zero-order moment of $\{a_k(n)\}$, and $\sum_{k=1}^L a_k(n)k$ in Equation (4b) is a first-order moment of $\{a_k(n)\}$. As a result, the Equation (1) can be transformed into:

$$c(n) = \sum_{k=1}^L a_k(n)k \quad (5)$$

From Equation (5), we obtain a new calculation formula for cross-correlation based on a first-order moment.

2.2. Normalized Cross-Correlation

Normalized cross-correlation is more complex than cross-correlation, because it includes an inner-product between two difference sequences from $\{f(i)\}$, $\{g(i)\}$ and their mean value. It is defined as

$$\rho(n) = \frac{\sum_{i=0}^{N-1} [f(n+i) - \bar{f}(n)][g(i) - \bar{g}]}{\left\{ \sum_{i=0}^{N-1} [f(n+i) - \bar{f}(n)]^2 \sum_{i=0}^{N-1} [g(i) - \bar{g}]^2 \right\}^{\frac{1}{2}}}, \quad (6)$$

where $\bar{f}(n) = \frac{1}{N} \sum_{i=0}^{N-1} f(n+i)$ and $\bar{g} = \frac{1}{N} \sum_{i=0}^{N-1} g(i)$.

This Equation (6) can be rewritten as

$$\begin{aligned} \rho(n) &= \frac{\sum_{i=0}^{N-1} f(n+i)g(i) - \bar{g} \sum_{i=0}^{N-1} f(n+i) - \bar{f}(n) \sum_{i=0}^{N-1} g(i) + N[\bar{f}(n)\bar{g}]}{\left\{ \sum_{i=0}^{N-1} [f(n+i)^2 - 2f(n+i)\bar{f}(n) + \bar{f}(n)^2] \sum_{i=0}^{N-1} [g(i) - \bar{g}]^2 \right\}^{\frac{1}{2}}} \\ &= \frac{\sum_{i=0}^{N-1} f(n+i)g(i) - \frac{1}{N} \sum_{i=0}^{N-1} f(n+i) \sum_{i=0}^{N-1} g(i)}{\left\{ \sum_{i=0}^{N-1} f(n+i)^2 - \frac{1}{N} \left[\sum_{i=0}^{N-1} f(n+i) \right]^2 \sum_{i=0}^{N-1} [g(i) - \bar{g}]^2 \right\}^{\frac{1}{2}}} \end{aligned} \tag{7}$$

If we set

$$b(n) = \sum_{i=0}^{N-1} [f(n+i)]^2 \tag{8}$$

and substitute Equations (4a), (4b) and (8) into Equation (7), the NCC expressed by Equation (6) can be converted to

$$\rho(n) = \frac{\sum_{k=1}^L a_k(n)k - \bar{g} \sum_{k=0}^L a_k(n)}{\left\{ \left[b(n) - \frac{1}{N} \left[\sum_{k=0}^L a_k(n) \right]^2 \sum_{i=0}^{N-1} [g(i) - \bar{g}]^2 \right] \right\}^{\frac{1}{2}}} \tag{9}$$

From Equation (9), we develop a new calculation formula for NCC based on a first-order moment $\sum a_k(n)k$ and a zero-order moment $\sum a_k(n)$. It is obvious that the computation complexity of this NCC formula depends heavily upon the complexity of $\sum a_k(n)k$ and $b(n)$. Therefore, for a fast implementation of Equation (9), we introduce a fast algorithm and structure for $\sum a_k(n)k$ in Section 3, and an optimization method for $b(n)$ in Section 4.1.

3. The Fast Algorithm and Systolic Array for First-Order Moment

Liu et al. presented an algorithm and its systolic array for first-order moment in [25–27]. Their method is suitable to compute the first-order and the zero-order moment in Equation (4) rapidly. In this section, we introduce this algorithm and systolic array that aims to implement fast NCC by using Equation (9). In addition, because the introduced algorithm and array request many additions as the result of removing all multiplications, we also improve them in order for lower addition complexity.

3.1. The Fast Algorithm for First-Order Moment

According to [25], we illustrate a simple 1-network shown in Figure 1 that represents a map of transforming the two-dimensional vector $(1, x)$ into the vector $(1, (1 + x))$. This map is denoted by F that is

$$F(1, x) = (1, (1 + x)).$$

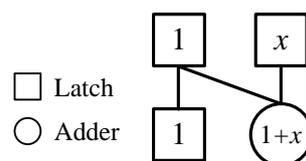


Figure 1. The 1-network.

Some characteristic equations obtained from F are

$$F(a, ax) = (a, a(1 + x)), \quad F(a + b, a + b) = F(a, a) + F(b, b) \tag{10}$$

Also,

$$F^2(1, x) = F(F(1, x)) = F(1, (1 + x)) = (1, 2 + x)$$

and by induction

$$F^{L-1}(1, x) = F(\dots F \dots F(1, x)) = (1, (L - 1 + x)).$$

Hence, we have

$$\begin{aligned} F^{L-1}(1, 1) &= F(\dots F \dots F(1, 1)) = (1, L), \\ F^{L-1}(a, a) &= (a, La). \end{aligned} \quad (11)$$

To compute first-order moment by this 1-network, let

$$\mathbf{a}_k = (a_k(n), a_k(n)) \quad (k = 1, 2, \dots, L),$$

so, Equations (10) and (11) are yielded by

$$\begin{aligned} F(F(\mathbf{a}_k) + \mathbf{a}_{k-1}) &= F(F(\mathbf{a}_k)) + F(\mathbf{a}_{k-1}) = F^2(\mathbf{a}_k) + F(\mathbf{a}_{k-1}) \\ &= (a_k(n) + a_{k-1}(n), 3a_k(n) + 2a_{k-1}(n)) \end{aligned}$$

Generally, the above equation is expanded into

$$\begin{aligned} F(F \dots F(F(F(a_L) + a_{L-1}) + \dots) + a_2) + a_1 &= F^{L-1}(a_L) + \dots + F^2(a_3) + F(a_2) + a_1 \\ &= \left(\sum_{k=1}^L a_k(n), \sum_{k=1}^L a_k(n)k \right). \end{aligned} \quad (12)$$

From Equation (12), $\sum a_k(n)$ in Equation (4a) and $\sum a_k(n)k$ in Equation (4b) can both be obtained from an iterative implementation of the map F . This computational flow uses the $(L - 1)$ recursive process of map F that includes $3L$ additions and 0 multiplications [26]. Therefore, the fast algorithm for first-order moment by Equation (12) can be described in Algorithm 1 as a subroutine Moment [29]. Its computational structure is also shown in Figure 2, which is an iterative structure of a 1-network with six adders and three latches. Its total addition number to compute N -point first-order moments $\sum a_k(n)k$ ($n = 0, 1, \dots, N - 1$) is $3NL$.

Algorithm 1 Moment ($a_L(n), a_{L-1}(n), \dots, a_0(n)$)

```

Define the array a with two elements
Initial a ← (  $a_L(n), a_L(n)$  )
for each  $k \in [2, L]$  do // Equation (12)
    a[1] ← a[1] + a[0] // 1-network  $F(\mathbf{a})$ 
    a[1] ← a[1] +  $a_{L-k+1}(n)$ 
    a[0] ← a[0] +  $a_{L-k+1}(n)$ 
end for
a[0] ← a[0] +  $a_0(n)$ 
return a

```

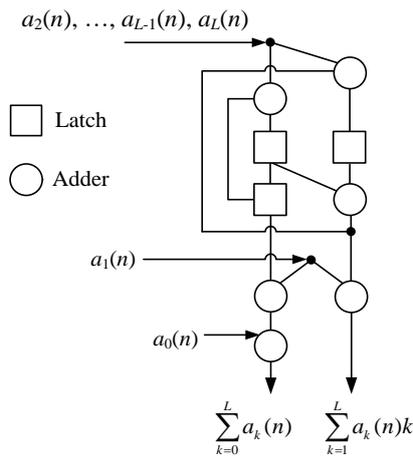


Figure 2. The computational structure for first-order moment.

3.2. The Systolic Array for First-Order Moment

The Equation (12) can be implemented by a systolic array for continuously generating a set of $\sum a_k(n)$ and $\sum a_k(n)k$ in parallel [27]. This systolic array is shown in Figure 3, which is actually a serial arrangement of $(L - 1)$ 1-networks extended from Figure 2. It uses $3L - 2$ adders, $L + 2$ latch, and 0 multiplier. In each clock cycle, we should input a sequence $\{ a_k(n) \}$ into this systolic array and get a $(\sum a_k(n), \sum a_k(n)k)$.

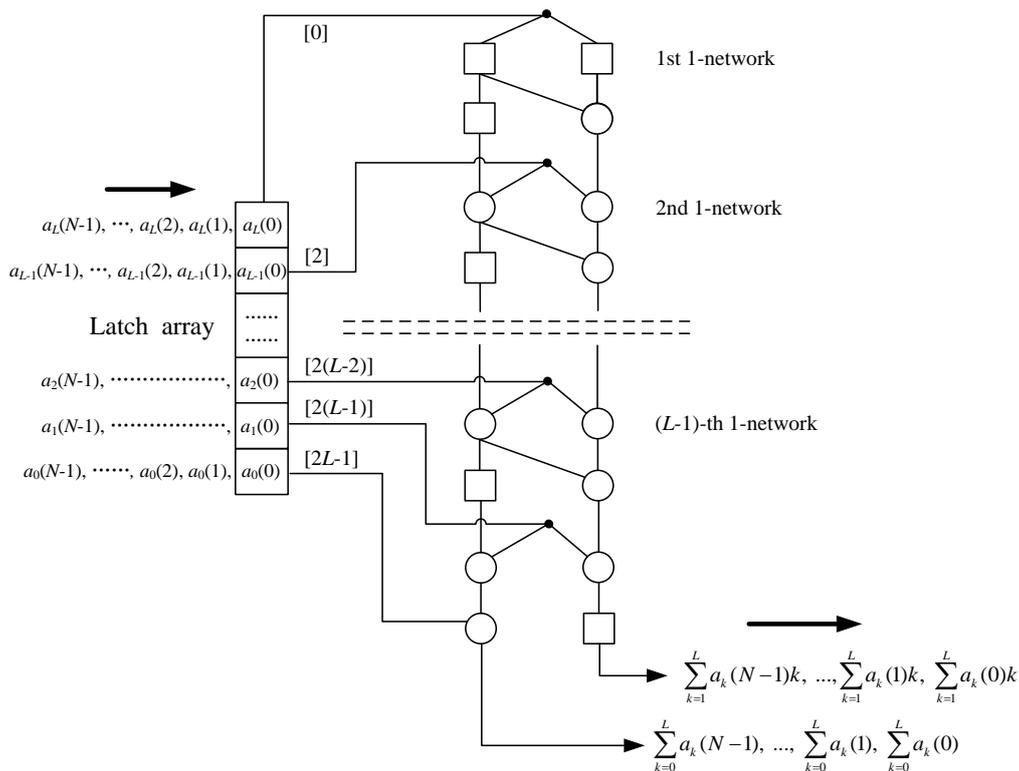


Figure 3. The systolic array for first-order moment.

Especially, to keep an operation synchronization for this parallel structure, the $(L - 1)$ -point $a_k(n)$ ($k = 2, \dots, L$) should be input into the $(L - 1)$ 1-networks respectively rather than simultaneously. Generally, a single $a_k(n)$ ($k > 0$) is input into the $(L - k)$ -th 1-network with a latency $n + 2(L - 1 - k)$ clock cycle. Hence, in Figure 3, we use the extra latch array to generate latency for $a_k(n)$ before it is input

into the corresponding 1-network. The number of latch array and latency time is shown in the note “[]”, which leads to the occurrence that different $a_k(n)$ are input into the different 1-networks at regular intervals. As a result, the total execution time of this systolic array to compute N -point $\sum a_k(n)k$ ($n = 0, 1, \dots, N - 1$) is that

$$2L - 1 + 1 + N - 1 = 2L + N - 1$$

clock cycles.

3.3. The Improvement of the Fast Algorithm and Systolic Array for First-Order Moment

The algorithm in Section 3.1 requires many additions that are computationally expensive when N is larger. In order to reduce its addition number, this algorithm is improved by means of an even-odd relationship that divides the first-moment of sequence $\{ a_k(n) \}$ into two smaller moments. This even-odd relationship is illustrated as:

$$\sum_{k=0}^L a_k(n) = \sum_{k=1}^{L/2} [a_{2k-1}(n) + a_{2k}(n)] + a_0(n), \tag{13a}$$

$$\sum_{k=1}^L a_k(n)k = \sum_{k=1}^{L/2} a_{2k-1}(n) \cdot (2k - 1) + \sum_{k=1}^{L/2} a_{2k}(n) \cdot 2k = 2 \sum_{k=1}^{L/2} [a_{2k-1}(n) + a_{2k}(n)]k - \sum_{k=1}^{L/2} a_{2k-1}(n). \tag{13b}$$

According to Equation (13), the fast algorithm described by Figure 2 can be improved to the new structure shown in Figure 4. This improved algorithm firstly adds $L/2$ additions to obtain the sequence $\{ a_{2k-1}(n) + a_{2k}(n) \}$ as well as $L/2 - 1$ addition to accumulate $\sum a_{2k-1}(n)$. Then each $a_{2k-1}(n) + a_{2k}(n)$ is input into map F successively for performing $L/2 - 1$ iterations. Finally, a left-shift operation and 1 subtraction are applied to generate $\sum a_k(n)k$. The improved algorithm requires $5L/2 - 1$ additions that are superior to Figure 2, even though its structure is more complex at the cost of decreasing $L/2$ additions. Although the sequence $\{ a_{2k-1}(n) + a_{2k}(n) \}$ could be continually divided by the even-odd relationship for further reducing additions, the fast algorithm’s structure would become very complex and unworthy.

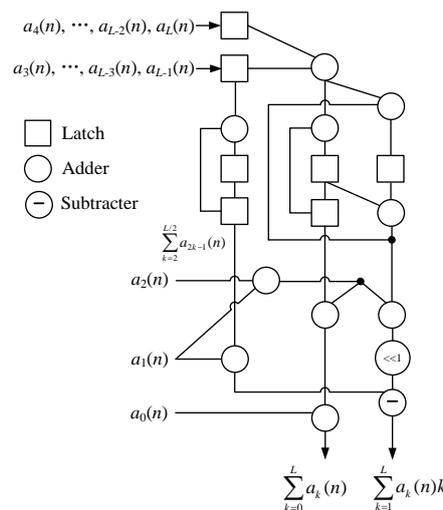


Figure 4. The improved computational structure for first-order moment.

Similarly, the systolic array in Figure 3 can be improved to the structure shown in Figure 5. This improved systolic array is a serial arrangement of the $L/2 - 1$ 1-networks extended from Figure 4. It requires $5L/2 - 3$ adders and $L/2 + 3$ latches that are superior to Figure 3, even though its structure is

more complex. As a result, the total execution time of this systolic array to compute N -point $\sum a_k(n)k$ ($n = 0, 1, \dots, N - 1$) is decreased to

$$L + 1 + 1 + N - 1 = N + L + 1$$

clock cycles.

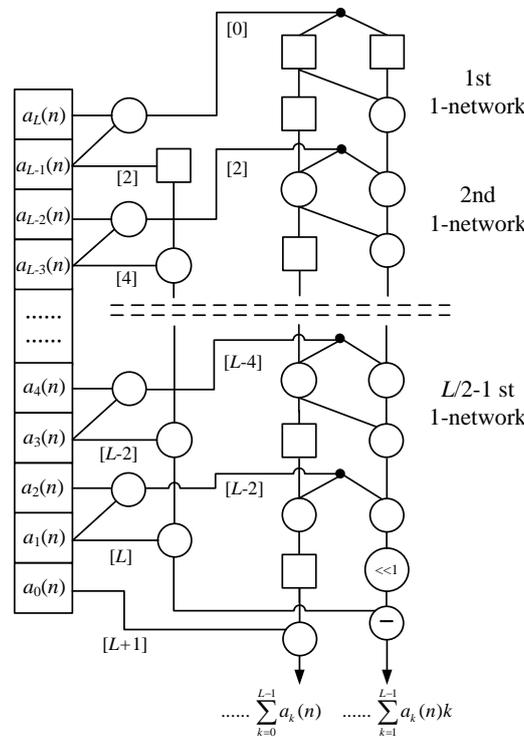


Figure 5. The improved systolic array for first-order moment.

4. The Fast Algorithm for Normalized Cross-Correlation

We apply the improved fast algorithm in Section 3.3 to compute the first-order and the zero-order moments in Equation (9). Thus, the fast algorithm for NCC is presented that can remove most of its multiplications. At first, some optimization methods are introduced in Section 4.1 to further reduce its additions.

4.1. The Optimization Methods

As the sequence $\{g(i)\}$ is a fixed correlation kernel in general, both $\bar{g} = \sum g(i)/N$ and $\sum [g(i) - \bar{g}]^2$ in Equation (9) could be pre-computed and reused for avoiding their repeated computations [30].

Although $b(n)$ in Equation (8) involves many additions and complex squares, it could also be computed by a simple function with the previous $b(n - 1)$, where

$$\begin{aligned} b(n) &= \sum_{i=0}^{N-1} f(n-1+i)^2 + [f(n+N-1)^2 - f(n-1)^2] \\ &= b(n-1) + [f(n+N-1) + f(n-1)][f(n+N-1) - f(n-1)] \end{aligned} \tag{14}$$

We only need to directly compute the first $b(0)$ by N multiplication and $N - 1$ additions, where the square is performed by multiplication. Then, the following $b(n)$ ($n = 1, 2, \dots, N - 1$) would be obtained from Equation (14) by only 1 multiplication, 2 additions and 1 subtraction.

4.2. The Step of the Fast Algorithm for NCC

The proposed fast algorithm for NCC would include five steps:

- Step 1** Initializing all $a_k(n) = 0$ ($k = 0, 1, \dots, L$), where $a_0(n)$ is indispensable for $\sum a_k(n)$.
Step 2 Implementing Equation (3) to acquire the sequence $\{a_k(n)\}$ using N addition.
Step 3 Computing $\sum a_k(n)$, $\sum a_k(n)k$ by Equation (13) and Figure 4 with $5L/2 - 1$ additions.
Step 4 Computing $b(n)$ by Equation (14) with 1 multiplication, 2 additions and 1 subtraction.
Step 5 Inputting $\sum a_k(n)$, $\sum a_k(n)k$ and $b(n)$ into Equation (9) for a NCC $\rho(n)$, which need 2 subtractions, 4 multiplications, 1 division and 1 square root calculation.

The computational flow of this algorithm is illustrated in Algorithm 2. It includes $N + 5L/2 + 1$ additions, 3 subtractions and 5 multiplications per output an NCC $\rho(n)$. Therefore, to compute N -point NCC, it requires $N - 1 + N(N + 5L/2 + 1) - 2 = N(N + 5L/2 + 2) - 3$ additions, and only $N + N - 1 + 4N = 6N - 1$ multiplications.

Algorithm 2 Computing NCC ($n, f, g, b(n-1)$)

```

for each  $a_k$  in the sequence  $\{a_k\}$ :  $a_k \leftarrow 0$ 
for each  $i \in [0, N-1]$  do // Equation (3)
   $k \leftarrow g(i)$ 
   $a_k \leftarrow a_k + f(n + i)$ 
end for
for each  $k \in [1, L/2]$  do // Equation (13a)
   $s \leftarrow s + a_{2k-1}$ 
   $a_k \leftarrow a_{2k-1} + a_{2k}$ 
end for
 $\mathbf{a} \leftarrow \mathbf{Moment}(a_{L/2}, a_{L/2-1}, \dots, a_2, a_1, a_0)$  // Algorithm 1
 $\mathbf{a}[1] \leftarrow \mathbf{a}[1] \ll 1 - s$  // Equation (13b)
Compute  $b(n)$  by  $b(n-1), f(n + N - 1)$  and  $f(n - 1)$  // Equation (14)
Compute  $\rho(n)$  by  $\mathbf{a}[0], \mathbf{a}[1]$  and  $b(n)$  // Equation (9)
return  $\rho(n)$ 

```

5. The Systolic Array for Normalized Cross-Correlation

We apply the improved systolic array in Figure 5 to design a hardware structure for fast NCC in parallel. Figure 6 shows this systolic structure that mainly includes three parts: the module **A** to compute $\{a_{2k-1}(n) + a_{2k}(n)\}$, the module **M** to compute the first-order and zero-order moment of $\{a_k(n)\}$, and the module **S** to compute $b(n)$. In each cycle, we simultaneously input N -point $f(n + i)$ into this systolic array and get an NCC result $\rho(n)$. At first, since the direct computation for $\{a_{2k-1}(n) + a_{2k}(n)\}$ needs many adders, a simplified structure for the module **A** is discussed in Section 5.1.

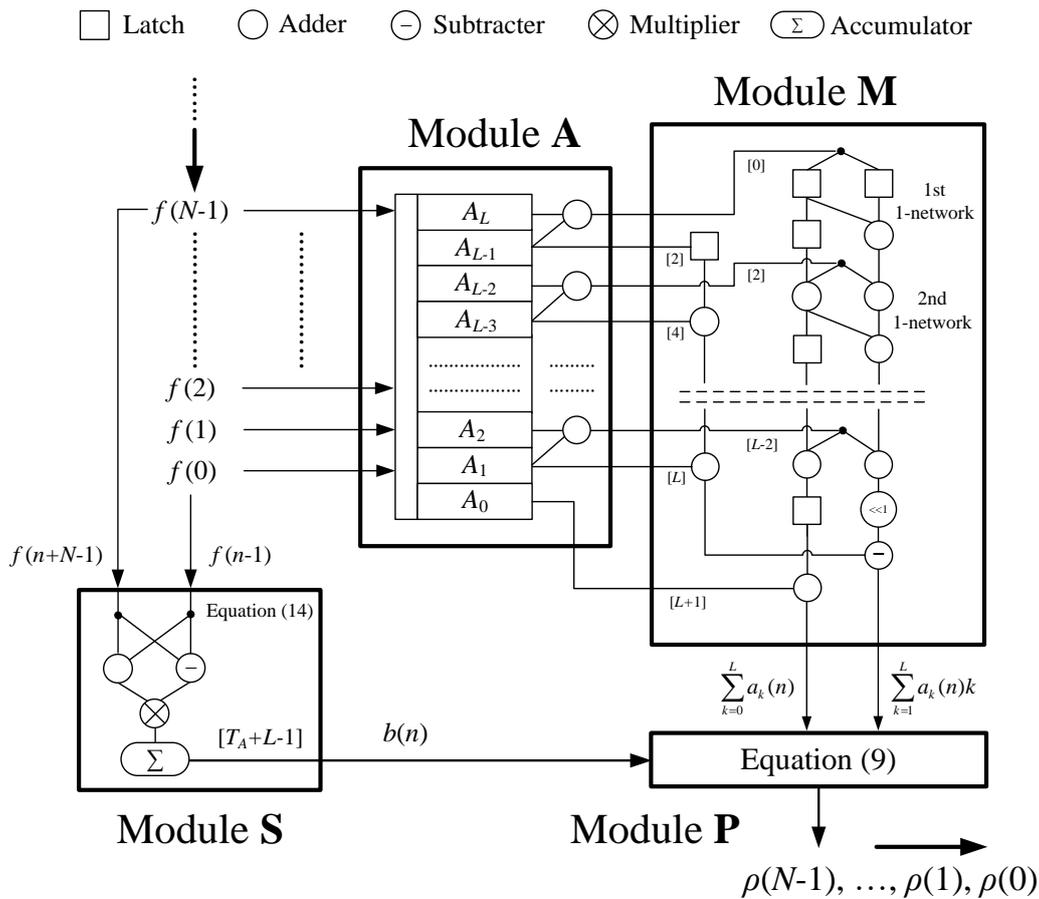


Figure 6. The systolic array for fast normalized cross-correlations (NCCs).

5.1. The Module A

The module A is to acquire an $L/2$ -point sequence $\{ a_{2k-1}(n) + a_{2k}(n) \}$ according to Equations (3) and (13) in every clock cycle. It includes $L + 1$ sub-modules A_k ($k = 0, 1, 2, \dots, L$) that firstly count $\{ f(n + i) \}$ to generate corresponding $\{ a_k(n) \}$, and then sum up the two adjacent $a_k(n)$ to obtain $\{ a_{2k-1}(n) + a_{2k}(n) \}$. We assume the execution time of the module A is T_A clock cycles. The N -point $f(n + i)$ should be inputted into the sub-modules $\{ A_k \}$ in a gradual way.

Since the correlation kernel $\{ g(i) \}$ is so invariable that the computational strategy for Equations (3) and (13) are known in advance, we could simplify the structure of A_k for less adder and data transfer. For example, for $N = 4, L = 4$ and $\{ g(i) \} = \{ 1, 2, 3, 4 \}$, the module A could be simplified as shown in Figure 7 with 2 adder and $T_A = 1$. However, for $N = 4, L = 4$ and $\{ g(i) \} = \{ 2, 1, 4, 2 \}$, the module A would be re-designed as shown in Figure 8 with 2 adder, 3 latches and $T_A = \log_2 4 = 2$. Therefore, the structure of the module A should be not fixed, but changed with different sequences $\{ g(i) \}$ to reduce its hardware complexity. We also show the module A using maximum adders when $\{ g(i) \} = \{ 4, 4, 4, 4 \}$ in Figure 9a, and the module A using 0 adders when $\{ g(i) \} = \{ 2, 4, 6, 8 \}$ in Figure 9b. From Figures 7–9, it can be obtained the adder number of the module A is from 0 to $N - 1$, and the latency T_A is from 0 to $\log_2 N$.

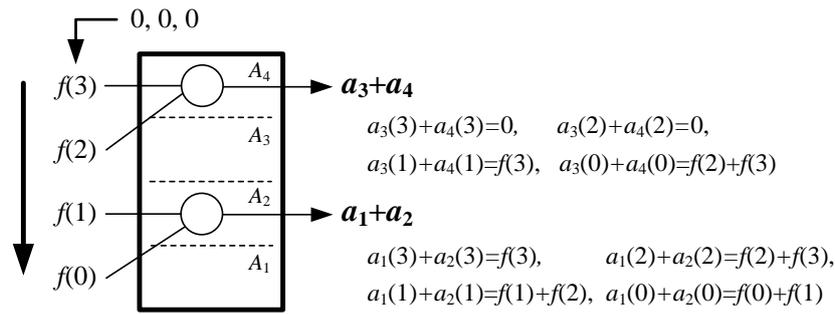


Figure 7. The module A for $\{g(i)\} = \{1, 2, 3, 4\}$.

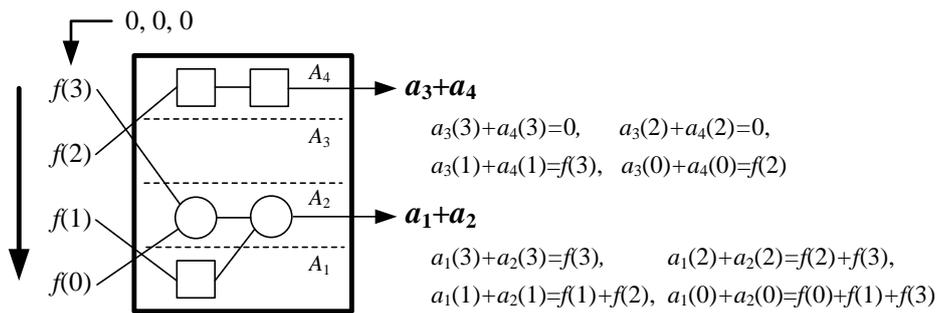


Figure 8. The module A for $\{g(i)\} = \{2, 1, 4, 2\}$.

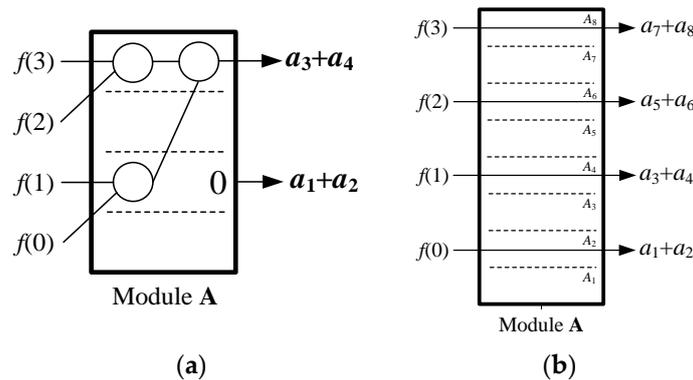


Figure 9. The module A using different addresses: (a) $\{g(i)\} = \{4, 4, 4, 4\}$; (b) $\{g(i)\} = \{2, 4, 6, 8\}$.

5.2. The Model P

The Model P is to implement Equation (9) with 4 multipliers, 1 divider and 1 square root extractor. It receives a $\sum a_k(n)k$ and a $b(n)$, and output a corresponding $\rho(n)$ in each cycle. Some fast methods can be applied for the square root operation. In addition, the fixed \bar{g} and $\sum [g(i) - \bar{g}]^2$ are saved in advance against repeated computation.

5.3. The Systolic Array

The systolic array in Figure 6 uses various modules to perform Equations (3), (9), (13) and (14), respectively, for NCC. Some latches are indispensable to connect these modules for assuring their mutual and parallel operation. The latch number has been shown in the note “[]”. The module M from Figure 5 is to compute first-order moments and zero-order moments based on Equation (13). The module S implements Equation (14) and generates $b(n)$ by 1 multiplier, 1 accumulator and 1 subtractor. Finally, the module P generates NCC $\rho(n)$. The systolic array’s total adder number is ranged from $2L - 2$ to $2L + N - 3$, and its multiplier number is 5.

The initial value of the accumulator in the module S is set as $b(0)$. In the n -th clock cycle, $f(n + N - 1)$ and $f(n - 1)$ would be input into the module S to get $b(n)$ with three clock cycles. Then

$b(n)$ is output from the module **S** to the module **P** with a latency $T_A + L - 1$. The aim is that $b(n)$, $\sum a_k(n)$ and $\sum a_k(n)k$ can arrive in the **P** at the same time.

6. Comparisons

The proposed algorithm and systolic structure are compared with some existing methods to verify their effectiveness. These compared methods are also focused on reducing their multiplication numbers.

6.1. Algorithm Comparison

Because correlation and convolution can share fast algorithms, we compare the proposed algorithm in Section 4 with some convolution algorithms, as well as a fast NCC algorithm to compute an N -point cyclic NCC. The computational complexity of these algorithms are displayed in Table 1, where we set a complex multiplication, which is equivalent to three real multiplications and three real additions, an “AND” operation is equivalent to an addition [31], and a subtraction is also equivalent to an addition.

From Table 1, the multiplication and addition complexity of the FFT-based algorithm are both $O(N \log_2 N)$, the DA-based algorithm is the least addition complexity, and the fast NCC algorithm has zero multiplication. The proposed algorithm uses $O(N^2)$ additions that are more than the FFT-based and the DA-based algorithm, and $O(N)$ multiplications that are more than the fast NCC algorithm. However, the FFT-based algorithm needs float addition and multiplication operations that are more complex than integer operations, the DA-based algorithm requires tedious decode address and very large memories, as well as that the fast NCC algorithm is the most addition complexity and not suitable for high-precision matching [15]. Figure 10 shows the four algorithms’ multiplication and addition number increasing along with N . It is obviously that the proposed algorithm’s multiplication number is lower than both the FFT-based algorithm’s and the DA-based algorithm’s, and its addition number is lower than the fast NCC algorithm’s when $N > 320$.

Table 1. The comparisons of computational complexity.

Algorithm	Multiplication	Addition
Direct calculation	$2N(N + 1)$	$3N(N + 1)$
FFT-based algorithm [8,9]	$(3/2)N \log_2 N - (3/2)N + 16$	$(7/2)N \log_2 N - N/2 + 15$
DA-based algorithm [22]	$7N - 1$	$(5N - 2) \log_2 L + 8N - 1$
Fast NCC algorithm [15]	0	$3N(N + 1)$
The proposed algorithm	$6N - 1$	$N(N + 5L/2 + 5) - 4$

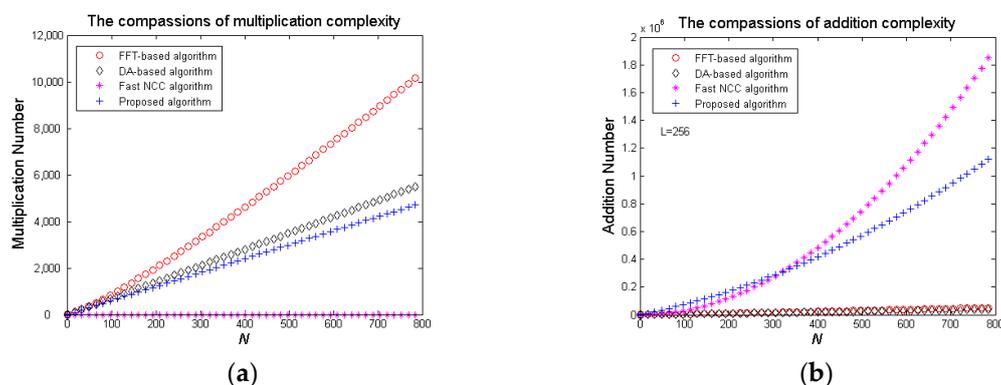


Figure 10. The four algorithm’s multiplication and addition number: (a) Multiplication (b) Addition.

The wireless sensor and communication is an important application field for the proposed algorithm. Therefore, we compare the execution time of the five algorithms from Table 1 by using a mobile phone with the type “HUAWAI nova 2s (HWI-AL00)” and the operation system “Android 9”. Figure 11 shows these algorithms’ execution time to compute a cyclic NCC by the phone with N

from 100 to 6000. The growth curve of the FFT-based algorithm's time is similar to a step curve, in that the length of FFT needs to be extended from N to $2^{\lceil \log_2 N \rceil}$. Although the DA-based algorithm can use the least time, it needs too much memory to make it worthwhile. From the Figure 11, the proposed algorithm's execution time is less than the FFT-based algorithm's when $N < 5500$, and is very close to the fast NCC algorithm, but not involved with noise.

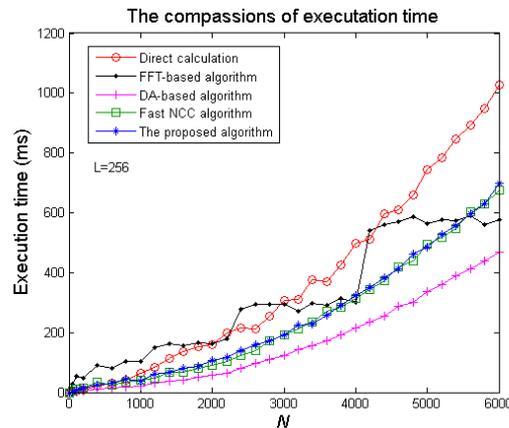


Figure 11. The comparisons of the five algorithm's execution time (ms).

In addition, it is important that the proposed algorithm has five advantages, as follows:

- (1) With less multiplications and memory.
- (2) Simple computational structure due to its simple implementation.
- (3) Precision and Fit to discrete domain as it uses integer operations [32].
- (4) Without limitations on the length of NCC.
- (5) Implementation by simple systolic structure.

6.2. Structure Comparison

We compare the proposed systolic array in Section 5 with some existing hardware structures. Table 2 shows the hardware complexity of these structures to implement an N -point cyclic NCC, where $N = PM$ (P and M are two positive integers derived from [33]). Because the proposed array's adder number and latency are not fixed, but varied with the sequence $\{g(i)\}$, we only display their value range according to Section 5.1. The execution time of the model \mathbf{P} is assumed as three clock cycles.

Table 2. The comparisons of hardware complexity.

Complexity	The Proposed Structure	Structure in [22]	Structure in [33]
ROM number	0	$[(N-1)/2] \lceil 2^{3(N-1)/7} \rceil$	$2^M P \log_2 L$
Adder Number	$2L-2$ To $2L+N-3$	$2N$	$(P+1)\log_2 L$
Latency	$L+5$ To $\log_2 N + L+5$	$2 \log_2 L$	$\log_2 L + P$
Throughput	1	$N/2 \log_2 L$	1

From Table 2, it is an advantage that the proposed systolic structure does not need ROMs, while the other two structures use $O(2^N)$ ROMs that are hardware-expensive when $N > 16$. The structure [22] has minimum latency, but its throughput is more than 1. The structure [33] needs the $O(P)$ adder and latency that would increase rapidly with N .

The proposed structure's hardware complexity is dependent upon L . Furthermore, for long NCCs, or two-dimension NCCs when N and P are larger than L , the adder number of the proposed structure is lower than that of the structure [22], and the latency of the proposed structure is lower than that of the structure [33]. Figure 12 shows the three structures' adder number and latency increasing along

with N , where the proposed structure adopts maximum adder and latency to perform comparisons. It is obvious that the proposed structure's adder number is least when $N > 1800$, and its latency is lower than the structure [33] when $N > 1500$. Therefore, although additional $O(L)$ latches are required for data store and transfer, the proposed systolic array could be more efficient in digital signal and image domain where the maximum value of L is less than 256 in general [34].

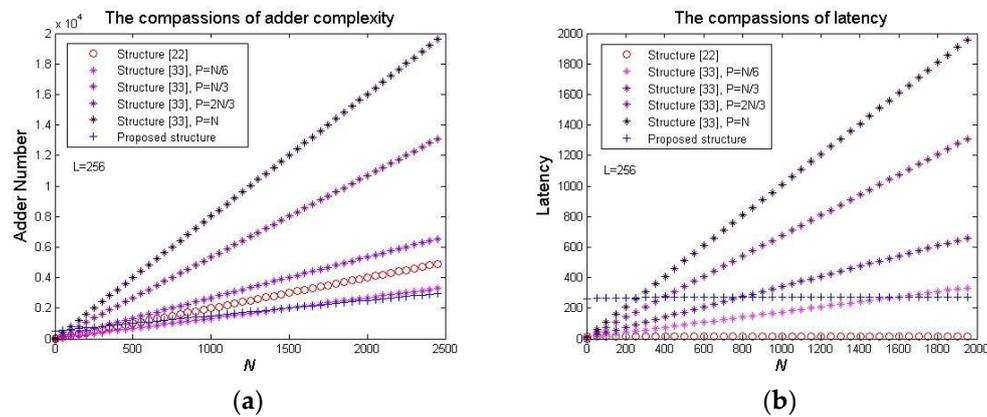


Figure 12. The three structure's adder number and latency: (a) Adder (b) Latency.

7. Conclusions

It is suggested that digital NCCs be implemented by efficient algorithms and hardware structures for decreasing their high multiplication complexity [35]. With the assist of fast computation for first-order moment, this paper presents an algorithm and a systolic array for fast NCCs that aim to reduce multiplication as much as possible. To do this, the key is to transform the complex inner-product in the NCC into a simple first-order moment according to the statistical properties of the digital inner-product, and then a new NCC formula based on a first-order moment is established in order for eliminating inner-product operations. As a result, by introducing an algorithm without multiplication into the computation of the first-order moment in NCC, we proposed a fast algorithm for NCC with the advantages of simple implementation, less multiplication, no length limitation, and so on. Especially, as the introduced algorithm for first-order moment requests many additions, we also improved it by means of an even-odd relationship to reduce addition complexity and execution time. It is an advantage that the introduced algorithm for the first-order moment can be implemented by systolic structure, so a systolic array composed of latches and adders is designed for implementing fast NCC in parallel. This systolic array is hardware-efficient due to its parallel operation, simple structures and seldom multiplier. This paper analyzes the computational and the hardware complexity for the proposed algorithm and systolic array, and compares them with some existing methods to prove their efficiency. The proposed algorithm and array could also be applied for digital filter and various transforms [36].

There are still many additions in the proposed algorithm and systolic structure. Future studies will focus on further reducing their additions.

Author Contributions: The work presented in this paper was completed with collaboration among all authors. Conceptualization, C.P.; Methodology, C.P., Z.L. and X.H.; Formal analysis, Z.L.; Writing—original draft preparation, C.P.; Writing—review and editing, Z.L., X.H., H.L.; Supervision, H.L.; Funding acquisition, X.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Grant No. 61801337, 61572012), the Natural Science Foundation of Hubei Province of China (Grant No. 2018CFB661, 2017CFB677).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Annaby, M.H.; Fouda, Y.M.; Rushdi, M.A. Improved Normalized Cross-Correlation for Defect Detection in Printed-Circuit Boards. *IEEE Trans. Semicond. Manuf.* **2019**, *32*, 199–211. [[CrossRef](#)]
2. Duong, D.H.; Chen, C.S.; Chen, L.C. Absolute Depth Measurement Using Multiphase Normalized Cross-Correlation for Precise Optical Profilometry. *Sensors* **2019**, *19*, 1–20. [[CrossRef](#)] [[PubMed](#)]
3. Banharsakun, A. Feature point matching based on ABC-NCC algorithm. *Evol. Syst.* **2018**, *9*, 71–80. [[CrossRef](#)]
4. Kotenko, I.; Saenko, I.; Branitskiy, A. Applying Big Data Processing and Machine Learning Methods for Mobile Internet of Things Security Monitoring. *J. Internet Serv. Inf. Secur.* **2018**, *8*, 54–63.
5. Sridevi, M.; Sankaranarayanan, N.; Jyothish, A.; Vats, A.; Lalwani, M. Automatic traffic sign recognition system using fast normalized cross correlation and parallel processing. In Proceedings of the 2017 International Conference on Intelligent Communication and Computational Techniques, Jaipur, India, 22–23 December 2017; pp. 200–204.
6. Bovik, A.C. Basic Tools for Image Fourier Analysis. In *The Essential Guide to Image Processing*; Academic: San Diego, CA, USA, 2009.
7. Wu, P.; Li, W.; Song, W.L. Fast, accurate normalized cross-correlation image matching. *J. Intell. Fuzzy Syst.* **2019**, *37*, 4431–4436. [[CrossRef](#)]
8. Liu, G.Q.; Kreinovich, V. Fast convolution and Fast Fourier Transform under interval and fuzzy. *J. Comput. Syst. Sci.* **2010**, *76*, 63–76. [[CrossRef](#)]
9. Kaso, A.; Li, Y. Computation of the normalized cross-correlation by fast Fourier transform. *PLoS ONE* **2018**, *13*, e0203434. [[CrossRef](#)]
10. Narasimha, M.J. Linear Convolution Using Skew-Cyclic Convolutions. *IEEE Signal. Process. Lett.* **2010**, *14*, 173–176. [[CrossRef](#)]
11. Cheng, L.Z.; Jiang, Z.R. An efficient algorithm for cyclic convolution based on fast-polynomial and fast-W transforms. *Circuits Syst. Signal. Process.* **2001**, *20*, 77–88.
12. Li, H.; Lee, W.S.; Wang, K. Immature green citrus fruit detection and counting based on fast normalized cross correlation (FNCC) using natural outdoor colour images. *Precis. Agric.* **2016**, *17*, 678–697. [[CrossRef](#)]
13. Tsai, D.M.; Lin, C.T. Fast normalized cross correlation for defect detection. *Pattern Recognit. Lett.* **2003**, *24*, 2625–2631. [[CrossRef](#)]
14. Byard, K. Application of fast cross-correlation algorithms. *Electron. Lett.* **2015**, *51*, 242–244. [[CrossRef](#)]
15. Yoo, J.C.; Choi, B.D.; Choi, H.K. 1-D fast normalized cross-correlation using additions. *Digit. Signal. Process.* **2010**, *20*, 1482–1493. [[CrossRef](#)]
16. Ismail, L.; Guerchi, D. Performance Evaluation of Convolution on the Cell Broadband Engine Processor. *IEEE Trans. Parallel Distrib. Syst.* **2011**, *22*, 337–351. [[CrossRef](#)]
17. Chaudhari, R.E.; Dhok, S.B. An Optimized Approach to Pipelined Architecture for Fast 2D Normalized Cross-Correlation. *J. Circuits Syst. Comput.* **2019**, *28*, 1950211. [[CrossRef](#)]
18. Mehendale, M.; Sharma, M.; Peher, P.K. DA-Based Circuits for Inner-Product Computation. In *Arithmetic Circuits for DSP Application*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2017; pp. 77–112.
19. Cao, L.; Liu, J.G.; Xiong, J.; Zhang, J. Novel structures for cyclic convolution using improved first-order moment algorithm. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2014**, *61*, 2370–2379. [[CrossRef](#)]
20. Carranza, C.; Llamocca, D.; Pattichis, M. Fast 2D Convolutions and Cross-Correlations Using Scalable Architectures. *IEEE Trans. Image Process.* **2017**, *26*, 2230–2245. [[CrossRef](#)]
21. Meher, P.K. Efficient Systolization of Cyclic Convolutions Using Low-Complexity Rectangular Transform Algorithms. In Proceedings of the 2007 International Symposium on Signals, Circuits and Systems ISSCS '07, Iasi, Romania, 13–14 July 2007; pp. 1–4.
22. Chen, H.C.; Guo, J.I.; Chang, T.S.; Jen, C.W. A Memory-Efficient Realization of Cyclic Convolution and Its Application to Discrete Cosine Transform. *IEEE Trans. Circuits Syst. Video Technol.* **2005**, *15*, 445–453. [[CrossRef](#)]
23. Syed, N.A.A.; Meher, P.K.; Vinod, A.P. Efficient Cross-Correlation Algorithm and Architecture for Robust Synchronization in Frame-Based Communication Systems. *Circuits Syst. Signal. Process.* **2018**, *37*, 2548–2573. [[CrossRef](#)]

24. Vun, C.H.; Premkumar, A.B.; Zhang, W. A New RNS based DA Approach for Inner Product Computation. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2013**, *60*, 2139–2152. [[CrossRef](#)]
25. Liu, J.G.; Pan, C.; Liu, Z.B. Novel Convolutions using First-order Moments. *IEEE Trans. Comput.* **2012**, *61*, 1050–1056. [[CrossRef](#)]
26. Hua, X.; Liu, J.G. A Novel Fast Algorithm for the Pseudo Winger-Ville Distribution. *J. Commun. Technol. Electron.* **2015**, *60*, 1238–1247. [[CrossRef](#)]
27. Liu, J.G.; Liu, Y.Z.; Wang, G.Y. Fast Discrete W Transforms via Computation of Moments. *IEEE Trans. Signal. Process* **2005**, *53*, 654–659. [[CrossRef](#)]
28. Yazdanpanah, H.; Diniz, P.S.R.; Lima, M.V.S. Low-Complexity Feature Stochastic Gradient Algorithm for Block-Lowpass Systems. *IEEE Access* **2019**, *7*, 141587–141593. [[CrossRef](#)]
29. Viticchie, A.; Basile, C.; Valenza, F.; Liroy, A. On the impossibility of effectively using likely-invariants for software attestation purposes. *J. Wirel. Mob. Netw. Ubiquitous Comput. Dependable Appl.* **2018**, *9*, 1–25.
30. Adhikari, G.; Sahu, S.; Sahani, S.K.; Das, B.K. Fast normalized cross correlation with early elimination condition. In Proceedings of the 2012 International Conference on Recent Trends in Information Technology, Chennai, India, 19–21 April 2012; pp. 136–140.
31. Blahut, R.E. *Fast Algorithms for Digital Signal Processing*; Addison-Wesley: Reading, MA, USA, 1984.
32. Yuan, Y.; Qin, Z.; Xiong, C.Y. Digital image correlation based on a fast convolution strategy. *Opt. Lasers Eng.* **2017**, *97*, 52–61. [[CrossRef](#)]
33. Meher, P.K.; Park, S.Y. A novel DA-based architecture for efficient computation of inner-product of variable vectors. In Proceedings of the 2014 IEEE International Symposium on Circuits and Systems, Melbourne, Australia, 1–5 June 2014; pp. 369–372.
34. Mukherjee, D.; Mukhopadhyay, S. Fast Hardware Architecture for 2-D Separable Convolution Operation. *IEEE Trans. Circuits Syst. II Exp. Briefs* **2018**, *65*, 2042–2046. [[CrossRef](#)]
35. Yang, Y.J.; Zhang, Y.H.; Li, D.M.; Wang, Z.J. Parallel Correlation Filters for Real-Time Visual Tracking. *Sensors* **2019**, *19*, 1–22. [[CrossRef](#)]
36. Hartl, A.; Annessi, R.; Zseby, T. Subliminal Channels in High-Speed Signature. *J. Wirel. Mob. Netw. Ubiquitous Comput. Dependable Appl.* **2018**, *9*, 30–53.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).