





SAR Target CFAR Detection Via GPU Parallel Operation

Zongyong Cui , Member, IEEE, Hongbin Quan, Student Member, IEEE, Zongjie Cao , Member, IEEE, Shengping Xu , Student Member, IEEE, Chunmei Ding, Student Member, IEEE, and Junjie Wu , Member, IEEE

Abstract—The constant false alarm rate with convolution and pooling (CP-CFAR) method, which can improve the detection efficiency via GPU parallel acceleration in the airborne synthetic aperture radar (SAR) images, is proposed in this paper. The constant false alarm rate (CFAR) method is one of the most widely used methods for target detection in airborne SAR images. However, since the CFAR is performed on a pixel-by-pixel basis, the time consumption will increase rapidly with the expansion of image scene. Even if the GPU is used for acceleration, the efficiency improvement is still limited, which cannot meet the real-time processing requirements. Therefore, the CP-CFAR method for the target detection of SAR images is proposed in this paper. The convolution layer uses the horizontal and vertical Sobel operators to improve the contrast between targets and background, and the pooling layer can reduce the processing dimension of the images. The convolution and pooling layers are added before the two-parameter CFAR, which can reduce the computational elements but without losing the main feature of the original image. More importantly, compared to the traditional CFAR, the proposed CP-CFAR is more suitable for GPU acceleration, which can improve the detection efficiency significantly. Experiments on the moving and stationary target acquisition and recognition SAR images with a size of 1478×1784 show that, compared with the traditional cell-averaging CFAR, two-parameter CFAR and their CPU, multi-thread CPU, and GPU acceleration modes, the proposed CP-CFAR with GPU acceleration can obtain the best detection performance with the highest acceleration ratio, and the operation time is less than 192 ms.

Index Terms—Constant false alarm rate (CFAR), convolution layer, GPU parallel, pooling layer, synthetic aperture radar (SAR), target detection.

I. INTRODUCTION

AIRBORNE synthetic aperture radar (SAR) is an active imaging sensor with all-weather, all-day, penetrating ability and other characteristics and has been widely used in military and civilian fields [1], [2]. With the continuous enrichment of

SAR image resources, SAR automatic target recognition (ATR) has become an urgent need to deal with massive SAR data. A typical ATR system of the SAR image is usually divided into three stages: detection, identification, and classification. In the detection stage, the region of interest (ROI) needs to be screened out first, and the subsequent processing such as identification and classification will be performed in the ROI. Therefore, the performance of the detection stage will directly affect the accuracy and speed of subsequent processing [3], [4].

The constant false alarm rate (CFAR) target detection method is one of the most widely used methods in SAR target detection [5]. However, with the development of airborne SAR imaging technology, the resolution is higher and higher, and the image size will also increase. Due to the large amount of data in SAR images, the calculation speed is an important indicator to evaluate the practicality of the algorithm for the SAR image target detection [6]. Since the CFAR method uses the sliding windows to detect the targets, and all the windows should be processed in sequence, the time consumption of CFAR is positively related to the number of pixels in the images to be processed. For airborne SAR images, how to improve the efficiency of algorithm implementation and realize the real-time performance of target detection is a challenge for researchers.

In order to improve the detection efficiency of CFAR, many researchers have made unremitting efforts. In [7], an improved superpixel-level CFAR detection method is proposed. Compared with CFAR methods implemented with the sliding window technique, the computational burden of the proposed method is significantly reduced without loss of detection performance. An improved two-parameter CFAR method is proposed in [8]. The new two-parameter CFAR detector uses log-normal as the statistical model. Compared with traditional CFAR detectors, the parameter estimation is more accurate and does not need iterative numerical calculation. In [9], a superpixel-based CFAR detection method is proposed for the target detection of high-resolution SAR images, which greatly improves the detection accuracy.

These improved methods can improve the speed of CFAR target detection to a certain extent. However, it is still difficult to meet the real-time requirements for the target detection of SAR images in large scene. Since CFAR is a pixel-based target detection method, it has high parallelism and can use GPU for parallel computation.

NVIDIA provides a parallel computing architecture called Compute Unified Device Architecture (CUDA). GPUs are more

Manuscript received June 12, 2018; revised October 8, 2018; accepted October 29, 2018. Date of publication November 14, 2018; date of current version December 31, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61801098 and in part by the Fundamental Research Funds for the Central Universities under Grant 2672018ZYGX2018J013. (Corresponding author: Zongjie Cao.)

The authors are with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: zycui@uestc.edu.cn; 2575746889@qq.com; zjcao@uestc.edu.cn; 1213119646@qq.com; 2294534934@qq.com; junjie_wu@uestc.edu.cn).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/JSTARS.2018.2879082

efficient than multicore processors for parallel computing because they have thousands of computing cores and high memory bandwidth [10], [11]. Actually, GPU general-purpose computing is relatively mature and has been a powerful tool to improve the speed of target detection methods [12].

To further increase the computational efficiency of the CFAR-based method, the NVIDIA CUDA is used for parallel implementations in [13]. In [14], the research shows that the operation performance of CFAR based on GPU with CUDA is better than that based on CPU. A Parzen-window-kernel-based CFAR algorithm for SAR ship detection is proposed in [15] and implemented via CUDA. The speedup can reach to $112\times$ as compared to the algorithm implemented sequentially on CPU. However, the operation of the above-mentioned CFAR methods with GPU acceleration is still via pixel by pixel, so the efficiency improvement is also limited.

This paper proposes a novel CFAR method called CFAR with convolution and pooling (CP-CFAR), which is based on the analysis of CFAR, GPU acceleration, and dimensionality reduction. The CP-CFAR method consists of three stages: feature extraction, target detection, and follow-up processing. In the feature extraction stage, the image is processed by the convolution layer and the pooling layer. The convolution layer extracts the main features of the image, and the pooling layer reduces the number of image pixels. In the detection stage, the two-parameter CFAR method is used for target detection. In the follow-up processing stage, erosion operation, dilation operation, and median filtering operation are performed on the detected image. The three stages are all suitable for GPU acceleration, which can improve the detection efficiency significantly.

The remainder of this paper is organized as follows. Section II describes the related three different CFAR target detection methods: CA-CFAR, two-parameter CFAR, and CP-CFAR. In Section III, the CP-CFAR method is parallel computed, and the parallel processes are described in detail. In Section IV, the performance of different CFAR methods is tested on the public dataset of moving and stationary target acquisition and recognition (MSTAR). Conclusions are presented in Section V.

II. RELATED CFAR METHODS

The CFAR detection method is a pixel-level target detection algorithm, which requires that the target has strong contrast with respect to the background. The target detection is achieved by determining whether the gray level of each pixel exceeds a certain threshold. In this section, the CA-CFAR and two-parameter CFAR are introduced, and the CP-CFAR method is proposed based on the analysis of convolution and pooling operation.

A. CA-CFAR Method

For most of the CFAR methods, the clutter background distribution of the image needs to be determined before detection. In order to adapt to different clutter scenarios and simplify performance analysis, the basic CFAR detector divides the clutter background into three typical scenarios for target detection: uniform clutter background, clutter edges, and multiple targets. The

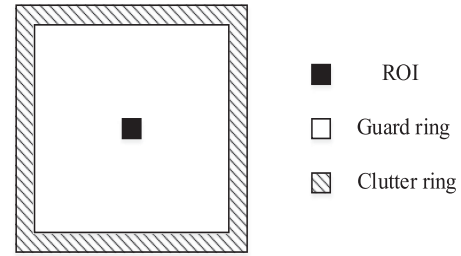


Fig. 1. Window configuration of the CA-CFAR method.

CA-CFAR is proposed to deal with the uniform clutter area. It uses all the pixels of the clutter area within the sliding window to estimate the parameters of the corresponding clutter statistical model.

Typically, the local statistics of the ROI and its background are computed and compared against the selected threshold to determine if the ROI is target or not. In the CA-CFAR, different thresholds are assigned to each pixel and used in conjunction with the ROI's statistics to determine if the pixel is bright or not [16].

The window configuration of the CA-CFAR detector is shown in Fig. 1.

The ROI is the pixel to be detected, the guard ring is set to reduce the influence of the target pixel on clutter parameter estimation, and the clutter ring is used to estimate clutter model parameters.

The strategies for choosing the size of window are as follows [17].

- 1) The size of the ROI should be 1×1 , which means there is only one pixel in the ROI.
- 2) The size of the guard ring should be twice the size of the target, accurately; if the size of the target is $\text{width} \times \text{height}$, the size of guard ring should be $(2 * \max(\text{width}, \text{height}) + 1) \times (2 * \max(\text{width}, \text{height}) + 1)$.
- 3) In order to balance the detection speed and detection effect, the width of the clutter ring is set to 1.

During the CA-CFAR detection process, determining whether the pixel in ROI is the target pixel or not is according to a certain criterion:

$$J(x, y) = \begin{cases} \text{true}, & \mu_{\text{ratio}}(x, y) > T \\ \text{false}, & \text{otherwise} \end{cases} \quad (1)$$

where T is the threshold of the CA-CFAR and is inversely proportional to the false alarm (P_{fa}); the $\mu_{\text{ratio}}(x, y)$ is known as the mean ratio and is defined as

$$\mu_{\text{ratio}}(x, y) = \frac{\mu_{\text{ROI}}(x, y)}{\mu_{\text{C}}(x, y)} \quad (2)$$

where $\mu_{\text{ROI}}(x, y)$ is the mean of the ROI, $\mu_{\text{C}}(x, y)$ is the mean of clutter, and they are calculated according to Fig. 1.

B. Two-Parameter CFAR Method

The two-parameter CFAR detection method is based on the assumption that the background clutter obeys Gaussian distribution, which can adapt to the background clutter changes.

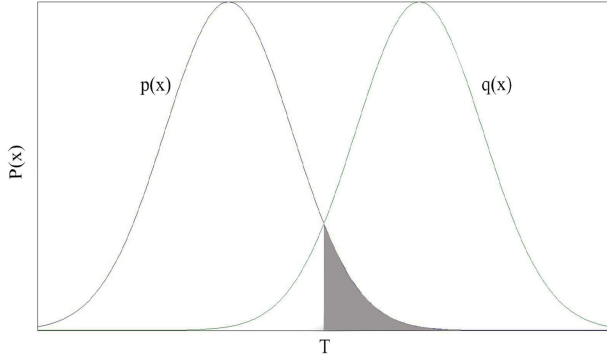


Fig. 2. Schematic diagram of obtaining the threshold of the two-parameter CFAR.

According to Fig. 2, P_{fa} is the given false alarm probability, the probability density function of clutter is $p(x)$, and the probability density function of the target is $q(x)$. The relationship between P_{fa} and $p(x)$ is as follows:

$$P_{fa} = \int_T^{\infty} p(x) dx \quad (3)$$

where T is the detection threshold.

It can be seen that the false alarm probability P_{fa} is corresponding to the area of the shaded part. If the probability of false alarm P_{fa} is given, the detection threshold T is determined by the statistical characteristics of the clutter around the target [18]. The window configuration of the two-parameter CFAR method is the same as that of CA-CFAR in Fig. 1.

The mean and variance of the clutter are used for determining the threshold T in the two-parameter CFAR [8]. Assuming that the clutter background obeys Gaussian distribution, the probability density function $p(x)$ is

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(\ln x - u)^2}{2\sigma^2}\right\}, \quad x \geq 0 \quad (4)$$

where u is the mean value of the clutter ring, and σ is the standard deviation of the clutter ring.

Let $y = \ln x$, $x = e^y$; the probability density function of y is

$$p_y(y) = p(e^y) \left| \frac{de^y}{dy} \right| = -\frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(y - x_m)^2}{2\sigma^2}\right]. \quad (5)$$

According to (3), the threshold T is

$$T = \sigma \times \Phi^{-1}(1 - P_{fa}) + u. \quad (6)$$

Similar to CA-CFAR detection, in the two-parameter CFAR method, the ROI also needs to travel each pixel of the input image to determine whether the pixel in the ROI belongs to target or not by the following criteria:

$$J(x, y) = \begin{cases} \text{true}, & I_{\text{input}}(x, y) > T \\ \text{false}, & \text{otherwise} \end{cases}. \quad (7)$$

C. CP-CFAR Method

In the SAR images, the mutation of the gray value of the target part is more obvious, but the grayscale of the background changes more smoothly. The Sobel operator can be taken as the

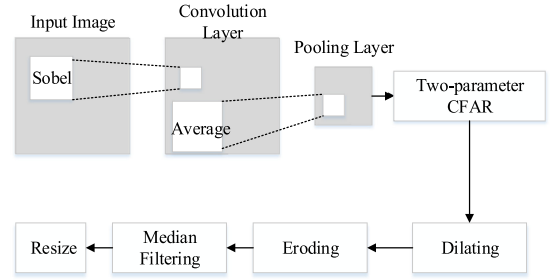


Fig. 3. Flow of the CP-CFAR method.

-1	-2	-1
0	0	0
1	2	1

Vertical Sobel operator

-1	0	1
-2	0	2
-1	0	1

Horizontal Sobel operator

Fig. 4. Horizontal and vertical Sobel operators used in this paper.

convolution kernel of the convolution layer, since after the Sobel filtering, the background noise can be filtered out and the target information can be extracted, and a certain texture and edge can be showed [19]. And the CFAR method performs the target detection based on the difference of the grayscale of the target and the background. Therefore, after the convolution operation is performed using the Sobel operator, it is possible to increase the gray value of the region, where the pixel gray value of the target region is mutated, and improve the accuracy of the CFAR target detection.

Therefore, the CP-CFAR is proposed in this paper. The CP-CFAR mainly has two stages: one is feature extraction stage and after it is the two-parameter CFAR. The flow of the CP-CFAR is shown in Fig. 3.

1) *Feature Extraction Stage*: The feature extraction stage of CP-CFAR includes a convolution layer and a pooling layer. In the convolution layer, the Sobel operator is used for convolution kernel, the size is 3×3 , and the stride is set to 1, as shown in Fig. 4.

The calculation formula is as follows:

$$I_{ij} = (|I_w * S_H|^2 + |I_w * S_V|^2)^{1/2} \quad (8)$$

where I_w means the sliding window with the center of I_{ij} , S_H and S_V indicate the horizontal Sobel operator and vertical Sobel operator in Fig. 4, respectively, and “*” indicates for convolution calculation.

Equation (8) shows that, in the convolution layer, the horizontal and vertical texture information of the input image is used. The main purpose of this step is to improve the contrast between the target and the background and then highlight the target itself and suppress the noise.

Fig. 5 presents a convolution result obtained by (8). It can be seen that, after the convolution layer, the target information is preserved and enhanced, and the background information, such as trees, has been suppressed effectively.

In the pooling layer, the average pooling is chosen in this paper, the filter size is 2×2 , and the stride is 2. The principle

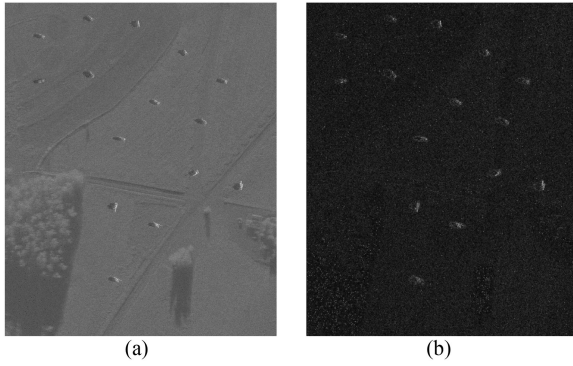


Fig. 5. Convolution result obtained by (8). (a) Original SAR image. (b) Output of the convolution layer.

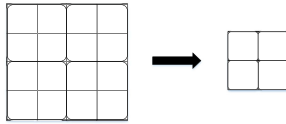


Fig. 6. Average pooling.

of the average pooling layer is, according to a certain sliding stride and size, calculating the average of a certain area and the output of the pooling layer [20], as shown in Fig. 6.

This operation can be described by the following equation:

$$M_{i,j} = \text{mean}(F_{i,j}). \quad (9)$$

The main purpose of pooling is to reduce the processing dimension of the images and decrease the operation time. For example, in Fig. 6, the dimension of output only is one-fourth of the input dimension.

2) *Target Detection*: After the feature extraction layer of CP-CFAR, the width and height of the feature map will be half of the original image, and the number of pixels to be processed in the two-parameter CFAR will be only one-fourth of the original one. In order to improve the accuracy of the detection results, some subsequent processing steps have been added, such as erosion and dilation operations and median filtering.

3) *False Alarm Elimination*: After CFAR detection, there are always some false alarms. In this paper, according to the basic size information of the targets, the morphological processing method and median filtering are used to eliminate the false alarms. The erosion and dilation operations can join the separated target areas and isolate the different target areas. After erosion and dilation, the median filtering is used to reduce the isolated bright spots. Through the above operations, the false alarm can be effectively eliminated, and the target can be completely preserved.

III. PARALLEL COMPUTATION OF THE CP-CFAR METHOD

From Fig. 3, it can be seen that the CP-CFAR can be divided into several steps: convolution layer, pooling layer, two-parameter CFAR, erosion, dilation, median filtering, and resize. This section will describe the detailed GPU acceleration process step by step. Meanwhile, because of the massive parallel cores structure in GPU, CUDA has been widely used in general computing in recent years. So, in this section, each step

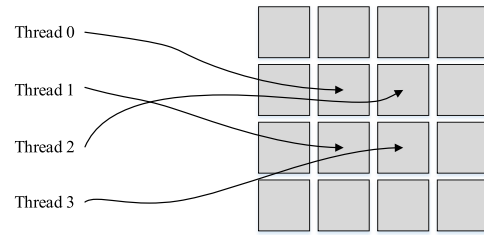


Fig. 7. Schematic of the texture memory.

of the CP-CFAR method is parallel computing in the CUDA architecture.

A. Parallel Computation of the Convolution Layer

The convolution layer operation requires successive convolution operations on 3×3 pixels centered on each pixel. Since each convolution operation is independent of each other, CUDA can be used for parallel acceleration. A thread is created for each pixel of the image, and the output of the pixel is calculated according to (8). In [21], a parallel GPU implementation of an edge detection method with a Sobel operator using CUDA environment is presented. However, the acceleration ratio cannot meet real-time requirements. So, the texture memory is used for acceleration.

Texture memory is cached on the chip, is read only, and has caches, so it has relatively fast speed. So, in some cases, it can reduce memory requests and provide higher memory bandwidth [22]. Texture caches are designed for those graphics applications that have a lot of spatial locality in memory access modes. In a computational application, this means that a thread can read a location that is “very close” to where the nearby thread reads, as shown in Fig. 7.

There is tremendous memory locality in the memory access mode during convolution layer computation, and this access mode can be accelerated by texture memory. The data of the input image are stored in texture memory. Combined with specific issues, the two-dimensional texture memory is used in this paper.

The delay in accessing shared memory is much less than the latency of accessing normal buffers, making shared memory as efficient as the cache or intermediate result scratchpad for each thread block [23]. Therefore, using shared memory can improve program performance when data are reused. Through adding the keyword of CUDA ‘`__shared__`’ before one variable declaration, this variable will be left in the shared memory.

According to [24], data transfer between CPU and GPU is time-consuming. Since the pooling layer also needs to be operated in the GPU, and in order to reduce the data transfer between the host memory and the device memory, the operation is completed without the need to transfer data to host memory, and the data are stored in device memory for the next step.

B. Parallel Computation of the Pooling Layer

The average pooling layer also needs to average the pixels in each adjacent 2×2 area in turn. Since each operation is independent of each other, CUDA can be used for parallel

acceleration by creating a thread for each 2×2 area to calculate the mean value.

Similar to the convolution layer, pixels within each 2×2 area are adjacent and can be read using texture memory, and the two-dimensional texture memory is also used.

Since the next operation is the two-parameter CFAR, the mean and variance of the clutter ring need to be calculated. However, the pixels of the clutter ring are not continuous, and it has more time consumption when using CUDA operation, the operation of taking out the pixels of clutter ring is executed on the CPU. Therefore, it is necessary to copy the data to the host for the next step.

C. Parallel Computation of the Two-Parameter CFAR

In the two-parameter CFAR method, the ROI needs to traverse every pixel of the image. The same operation is performed when each pixel is traversed, that is calculating the mean and variance of the corresponding clutter ring, so these operations can be accelerated using CUDA in parallel. Create a thread for each ROI to calculate the mean and variance of the corresponding clutter ring. For each ROI, the mean and variance of `num_pix` pixels of clutter ring need to be calculated. According to Section II, $\text{num_pix} = 2 * (2 + 2 * (2 * \max(\text{width}, \text{height}) + 1))$.

Due to the presence of the guard ring, the clutter ring of the ROI at the edge of the image is incomplete if the image edge is not expanded by pixels. Therefore, for an image with a width of `col` and a height of `row`, it needs to be filled with pixels with a pixel value of `aver` around it, where `aver` is the mean of the original image. According to Section II, the image after filling has a width of $\text{col} + 2 * \max(\text{width}, \text{height}) + 2$ and a height of $\text{row} + 2 * \max(\text{width}, \text{height}) + 2$.

The pixels of the clutter ring are not continuous, so reading the pixels of the clutter ring corresponding to each ROI requires complicated logic. However, the GPU is most efficient when all CUDA cores follow the same logic and computational path. When cores processing different pixels diverge from each other, each will execute sequentially. Therefore, if we use CUDA in this step to operate, it will be more time consuming. The specific operation is to use the CPU to sequentially read the pixels of the clutter ring corresponding to each ROI into an array of length $\text{col} \times \text{row} \times \text{num_pix}$.

In addition, page-locked memory is a special form of memory mapping that allows direct mapping of host memory to GPU memory space. So, the high-speed global memory bandwidth can be obtained by using `cudaHostAlloc` to allocate page-locked memory.

According to the above, the procedure of two-parameter CFAR acceleration based on CUDA is as follows.

- 1) Pixel fills around the image.
- 2) Use `cudaHostAlloc` to declare an array of length $\text{col} \times \text{row} \times \text{num_pix}$.
- 3) Store the pixels of the clutter ring corresponding to the $\text{col} \times \text{row}$ ROI sequentially in the declared page-locked memory.
- 4) Each thread is responsible for calculating the mean and variance of the pixels of the clutter ring corresponding to an ROI.

- 5) Store the calculated mean and variance sequentially in an array of length $\text{col} \times \text{row}$.
- 6) Calculate the threshold T of each ROI according to (6).
- 7) According to (7), if $I_{\text{ROI}}(x, y) > T(x, y)$, $I(x, y) = 255$, else $I(x, y) = 0$.

After the above operation, a binary image can be obtained, which contains the CFAR target detection results.

D. Parallel Computation of Erosion and Dilation

For the erosion and dilation operations, the processing of each pixel is independent of each other, so CUDA can be used for parallel acceleration. Dilation of the image is to replace the current value with the maximum value around the pixel. Erosion of the image is to replace the current value with the minimum value around the pixel.

The dilation is implemented as

$$g(i, j) = \text{Max}\{f(i - k, j - h)\}, \quad (k, h) \in W. \quad (10)$$

The erosion is implemented as

$$g(i, j) = \text{Min}\{f(i - k, j - h)\}, \quad (k, h) \in W. \quad (11)$$

In [25], the erosion and dilation operations are implemented in parallel on the GPU, and a very high speedup is achieved compared to the CPU. $(2 * n + 1) \times (2 * n + 1)$ template is used in the erosion and dilation operation. Similar to the convolution layer operation, the erosion and dilation operations perform the same operation for each pixel.

In order to avoid the currently processing pixels being disturbed by the processed pixels, a new image with the same size as the original image needs to be generated in the device memory. After creating a thread for each pixel, the maximum or minimum value of the surrounding pixels can be used to perform parallel operations on the dilation and erosion of the image.

E. Parallel Computation of Median Filtering

After erosion and dilation, the median filtering is used to reduce the isolated bright spots. Median filtering is a nonlinear signal processing technique that can effectively suppress noise based on sorting statistics theory. The basic principle of median filtering is to replace the value of a point in a digital image or digital sequence with the middle value of each point value in a neighborhood of the point. The median filter is implemented as

$$g(i, j) = \text{Med}\{f(i - k, j - m)\}, \quad (k, m) \in W \quad (12)$$

where $g(i, j)$ is the grayscale value of the output image, $f(i, j)$ is the grayscale value of the input image, and W is the template window. The size of the template window is assumed to be $w \times h$, as shown in Fig. 8.

It can be seen that the median filter algorithm needs to sort $w \times h$ elements. The sorting algorithm includes quick sorting, bubble sorting, and merge sorting, and in the worst case, quick sorting and bubble sorting require $(w \times h) \times (w \times h - 1) / 2$ times comparison operations; the two-way merge sorting algorithm needs $(w \times h) \times \log(w \times h)$ times comparisons.

In [26], the author proposes an acceleration algorithm that requires at least $(w \times h) \times 2 + w$ times comparisons to obtain the median value. However, if $(w \times h) \times 2 + w$ comparisons

$f(i-\frac{h}{2}, j-\frac{w}{2})$	$f(i-\frac{h}{2}, j)$	$f(i-\frac{h}{2}, j+\frac{w}{2})$
$f(i, j-\frac{w}{2})$	$f(i, j)$	$f(i, j+\frac{w}{2})$
$f(i+\frac{h}{2}, j-\frac{w}{2})$	$f(i+\frac{h}{2}, j)$	$f(i+\frac{h}{2}, j+\frac{w}{2})$

Fig. 8. Template window of median filtering with size $w \times h$.

are needed for each pixel of the image, it still takes a long time for calculating. Actually, the image of the CFAR detection result is a binary image with only two gray values of 255 and 0; the operation of finding the median value does not need to sort the elements in the template.

In this paper, the median filter only needs to sum the elements in the template to find the median value. The specific implementation is

$$g(i, j) = \begin{cases} 0, & \text{Sum}\{f(i-k, j-m)\} \leq 255 \times (w \times h)/2 \\ 255, & \text{otherwise} \end{cases}. \quad (13)$$

Since the value of the $w \times h$ elements in the template window can only be 255 or 0, as long as there are $(w \times h)/2 + 1$ or more elements with a value of 255, the median value will be 255. Using the above algorithm, only one summation is needed to calculate the median value, which can greatly improve the efficiency of the algorithm.

F. Parallel Computation of Resize

Since the width and height of the image after processing become half of the original image, it is necessary to adjust the size of the image to be the same as the original image. Bilinear interpolation is a common method for image scaling, and it is a linear interpolation extension of an interpolation function with two variables. The core idea is to perform a linear interpolation in both directions, respectively.

In [27], a graphic processing unit acceleration-based bilinear interpolation parallel is proposed. The experiment results show that the bilinear interpolation parallel algorithm can greatly improve calculation speed. But, in [27], the output is not a binary image. So, in the interpolation process, the bitwise OR operation is used in this paper to obtain the binary output after resizing.

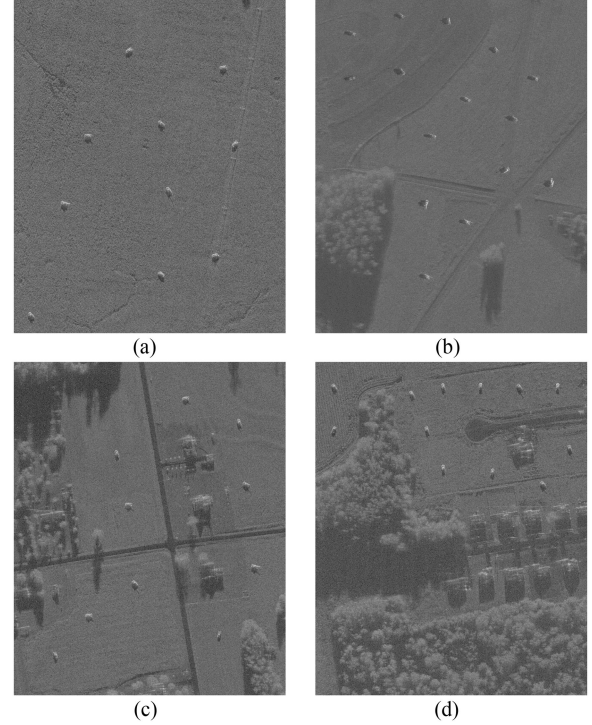


Fig. 9. (a)–(d) Synthesized airborne SAR images, with the background complexity increasing gradually. (a) has the simplest background, and (d) has the most complex background.

IV. EXPERIMENT ANALYSIS

The parallel computation of the CP-CFAR method has been described in detail in the previous section. Based on the theory of Section II, in this section, several sets of comparative experiments are conducted to verify the detection performance and GPU acceleration performance of the proposed CP-CFAR method.

A. Implement Configuration

1) *Hardware Configuration*: The experiments are carried out on an Intel Core i7- 7700 CPU with 16-GB main memory. The NVIDIA GeForce GTX 1080 Ti is characterized by 3584 CUDA Cores with 11-GB graphic memory. The operating system is Windows 10 64-bit, whereas the C/C++ Visual Studio 2015 development environment is used to implement the methods. The version of the CUDA Driver Runtime is 9.0.

2) *SAR Data*: The SAR images used in this paper come from the public database of MSTAR. The sensor collecting this dataset is a high-resolution beamforming SAR with a resolution of $0.3 \text{ m} \times 0.3 \text{ m}$, X-band, and HH polarization mode [28].

The MSTAR dataset provides the slice images with targets and the large-scale scenes images without targets. So, the SAR images with targets of T72, BMP2, and BTR70 are synthesized for the experiments. To verify the effectiveness of the method, this paper selects the SAR images under four different backgrounds, and with size of 1478×1784 , as shown in Fig. 9. Fig. 9(a) shows an image under simple background, and the background complexity of Fig. 9(b)–(d) is increasing gradually.

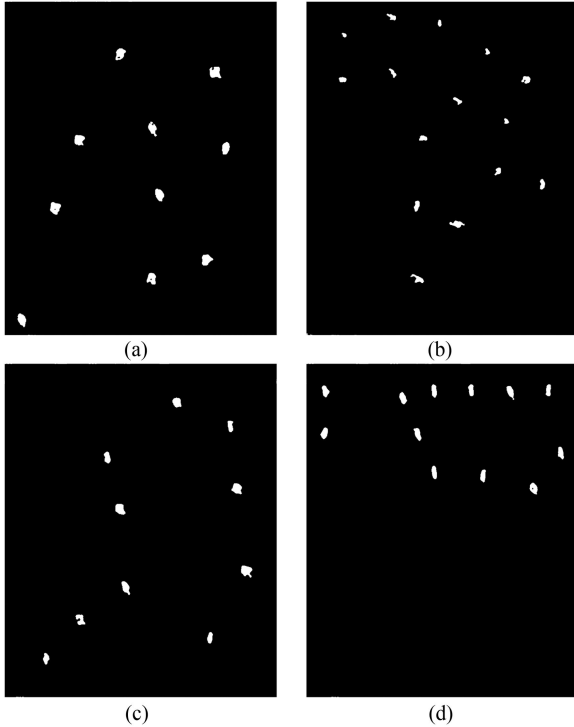


Fig. 10. Target detection results obtained by CP-CFAR. (a)–(d) is corresponding with Fig. 9(a)–(d), respectively.



Fig. 11. Partial enlarged view in Fig. 10(a).

B. Target Detection Via CP-CFAR

In this section, the CP-CFAR detection performance on the four images in Fig. 9 is illustrated, with setting the false alarm probability P_{fa} to 0.0001.

1) *Detection Performance*: The target detection results of CP-CFAR in Fig. 9 are shown in Fig. 10, and a partial enlarged view of the target detection result in Fig. 10(a) is shown in Fig. 11. It can be seen that the detected targets' areas are very similar with them in the original SAR images, which means that the proposed CP-CFAR can remain the shape information while accurately detecting the targets.

In order to show the quantitative detection results, the figure of merit (FoM) is defined as [29]

$$\text{FoM} = \frac{N_{\text{test}}}{(N_{\text{fa}} + N_{\text{real}})} \quad (14)$$

where N_{test} is the correct number of targets detected, N_{fa} is the number of false alarm targets, and N_{real} is the actual number of targets. The FoM can effectively reflect the performance of the

TABLE I
FoM OF TARGET DETECTION USING CP-CFAR

Images	N_{test}	N_{fa}	N_{real}	FoM
Fig. 9 (a)	10	0	10	100%
Fig. 9 (b)	15	0	15	100%
Fig. 9 (c)	10	0	10	100%
Fig. 9 (d)	12	0	12	100%

TABLE II
TIME CONSUMPTION OF EACH STEP IN CP-CFAR (MS)

Steps \ Images	Fig.9 (a)	Fig.9 (b)	Fig.9 (c)	Fig.9 (d)
Convolution	2	2	1	2
Pooling	1	1	1	2
Two-parameter CFAR	185	186	188	184
Dilation & Erosion	1	1	1	1
Median Filtering	2	2	1	2
Total	191	192	192	191

detection method. The larger the FoM, the better the detection performance.

The quantitative detection results of CP-CFAR are shown in Table I. It can be seen that, for the four airborne SAR images in Fig. 9, the proposed CP-CFAR can obtain the excellent performance, with all the targets being detected, and with no false alarm.

2) *Time Consumption Analysis*: The detailed time consumption of each step in CP-CFAR is shown in Table II.

It can be seen that the main time consumption comes from the two-parameter CFAR, accounting for nearly 98%. The other operations, like convolution, dilation, erosion, and median filtering, only take 1 or 2 ms.

The results show that, although the proposed CP-CFAR method adds operations before and after the two-parameter CFAR, these additional operations do not take up much time. On the contrary, these operations can significantly reduce the amount of computation while improving the detection performance.

C. Comparison of Different Methods

For further verification, the target detection performance in Fig. 9 of three methods, CA-CFAR, two-parameter CFAR, and CP-CFAR, is compared.

1) *Detection Performance Comparison*: For CA-CFAR, T is set to 1.2. For the two-parameter CFAR and CP-CFAR, the false alarm probability P_{fa} is set to 0.0001. The detection results of the three methods on the four images in Fig. 9 are shown in Fig. 12. The detection results are marked in the original images with the minimum enclosing rectangle of the binary detection results.

From Fig. 12(a), it can be seen that, for the images with simple background such as Fig. 9(a), the detection results obtained

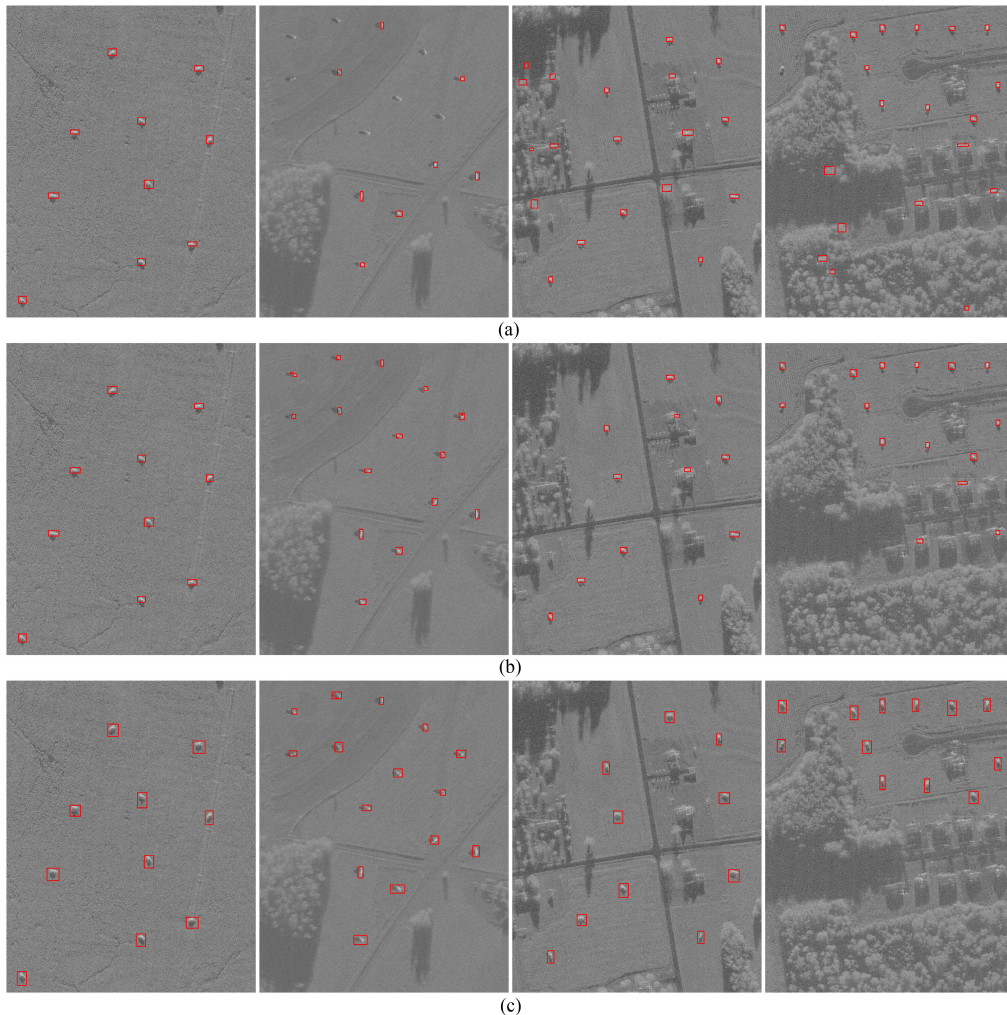


Fig. 12. Results of target detection using CA-CFAR, two-parameter CFAR, and CP-CFAR. The target detection results are marked with rectangular boxes in the original images. (a) Target detection results of CA-CFAR. (b) Target detection results of two-parameter CFAR. (c) Target detection results of CP-CFAR.

by CA-CFAR are better, but for the images with complex background such as Fig. 9(d), the detection results have many false alarms. It means that the CA-CFAR method cannot adapt to different backgrounds. In addition, different thresholds T need to be determined according to different images, which shows the unrobustness of the CA-CFAR and limits its further application.

Comparing Fig. 12(a) and (b), it can be seen that the performance of the two-parameter CFAR is better than the CA-CFAR. Especially for the fourth image, the two-parameter CFAR can detect all the targets, and the number of false alarms is less than that of the CA-CFAR. But, for Fig. 9(c) and (d), the two-parameter CFAR cannot obtain the ideal performance, which means that the complex background will affect the two-parameter CFAR detection ability.

Comparing Fig. 12(a)–(c), it can be seen that the proposed CP-CFAR can obtain the best performance in the all four images. All the targets can be detected, and even if the images with the most complex background, there is no false alarm. In addition, from the human visual angle, the targets' areas marked by the rectangle boxes are more fit for the actual size of the targets.

The detailed detection performance is listed in Table III. It can be seen that, for the images with sample background, all three methods can obtain the accurate detection results. However, with the increasing complexity of the background, the detection performance of CA-CFAR and two-parameter CFAR decreases gradually, while the performance of the proposed CP-CFAR remains stable. Such as for Fig. 9(d), the FoM of CA-CFAR is 52.4%, two-parameter CFAR is 80%, while the CP-CFAR is 100%.

Therefore, from the detection performance comparison, it can be said that the proposed CP-CFAR is robust to different SAR images and can obtain the best performance.

2) *GPU Parallel Acceleration*: For the target detection of SAR images, besides the detection accuracy, the detection time is also an important indicator.

For the above three methods, there are three ways of computing: CPU serial computation, CPU multithread parallel computation (CPU-MT), and GPU parallel computation. So, in this part, the performance of three different methods via CPU serial computation, CPU-MT, and GPU parallel computation is

TABLE III
TARGET DETECTION PERFORMANCE COMPARISON BETWEEN
DIFFERENT METHODS

Images	Methods	N_{test}	N_{fa}	N_{real}	FoM
Fig. 9 (a)	CA-CFAR	10	0	10	100%
	Two-parameter CFAR	10	0	10	100%
	CP-CFAR	10	0	10	100%
Fig. 9 (b)	CA-CFAR	8	0	15	53.3%
	Two-parameter CFAR	15	0	15	100%
	CP-CFAR	15	0	15	100%
Fig. 9 (c)	CA-CFAR	10	12	10	45.5%
	Two-parameter CFAR	10	2	10	83.3%
	CP-CFAR	10	0	10	100%
Fig. 9 (d)	CA-CFAR	11	9	12	52.4%
	Two-parameter CFAR	12	3	12	80%
	CP-CFAR	12	0	12	100%

compared. The procedure of CPU multithreads parallel computation is like in [30] and [31], with eight cores and four threads used in this paper.

The main comparison indicator is Speedup, which can be defined as follows:

$$\text{Speedup}_1 = \frac{T_{\text{CPU}}}{T_{\text{CPU-MT}}}$$

$$\text{Speedup}_2 = \frac{T_{\text{CPU}}}{T_{\text{GPU}}} \quad (15)$$

where T_{CPU} means the detection time consumption via CPU, $T_{\text{CPU-MT}}$ means the detection time consumption via multithread CPU, and T_{GPU} means the detection time consumption via GPU. It can be seen that the bigger the Speedup, the better the acceleration performance.

Table IV detailedly lists the time consumption of different three methods via CPU, multithread CPU, and GPU on the four images in Fig. 9, corresponding to the ‘‘CPU,’’ ‘‘CPU-MT,’’ and ‘‘GPU’’ columns, respectively.

- 1) From the ‘‘CPU’’ column in Table IV, it can be seen that the detection time of each method is more than 1 s. The CA-CFAR can obtain the fastest detection performance, CP-CFAR is the second, and the two-parameter CFAR obtains the most time consumption. This is related to the complexity of each method, and the CA-CFAR has the lowest complexity.

Take two-parameter CFAR and CP-CFAR for comparison. The CP-CFAR contains the two-parameter CFAR stage, but before the two-parameter CFAR, there are convolution operation and pooling operation, which can reduce the processing dimension of the images. Therefore, the CP-CFAR can obtain the lower time consumption.

However, it is worth noting that, although the CA-CFAR can obtain lowest time consumption, its detection accuracy is also the lowest. The time consumption of the proposed CP-CFAR is higher than the CA-CFAR, but the detection accuracy obtained by CP-CFAR is the highest.

- 2) From the ‘‘CPU-MT’’ and ‘‘Speedup₁’’ columns in Table IV, it can be seen that, after the CPU multithreads, the performance improvement of CA-CFAR is not very obvious, and the ‘‘Speedup₁’’ indicator is just a little more than 1. However, the performance improvement of two-parameter CFAR and CP-CFAR is obvious and can obtain the speedup of 3.56–5.39 \times over their CPU modes. It means that, compared with CPU serial processing, the CPU multithread computation can indeed reduce the detection time consumption.
- 3) From the ‘‘GPU’’ and ‘‘Speedup₂’’ columns in Table IV, it can be seen that, after the GPU acceleration, the performance improvement of CA-CFAR is similar with its CPU-MT mode, and the ‘‘Speedup₂’’ indicator is also just a little more than 1. By comparison, the performance improvement of two-parameter CFAR and CP-CFAR is very obvious, and the ‘‘Speedup’’ indicator of both is more than 10, which means that the two-parameter CFAR and CP-CFAR are more suit for being implemented on GPU.
- 4) Comparing ‘‘CPU-MT’’ and ‘‘GPU’’ columns, it can be seen that, for the three methods, their CPU-MT modes can improve the detection speed with varying degrees. But compared to their GPU modes, the speedup degree is still lower.

For the two-parameter CFAR, the process of calculating variance is a floating-point operation, and the ‘‘Speedup₂’’ indicators can reach to 11.98. However, while detecting the targets in SAR images, it still takes more than 1500 ms, and it is still difficult to meet real-time requirements.

For the proposed CP-CFAR, the ‘‘Speedup₂’’ indicator can reach to 18.74, and the time consumption on four SAR images via GPU is less than 192 ms, which can meet the real-time processing requirements. So, in general, the proposed CP-CFAR can obtain the best detection efficiency while obtaining the best detection accuracy.

To further analyze the performance of the proposed CP-CFAR, the two-parameter CFAR with 4-to-1 pooling is taken as an extra comparison. Actually, the two-parameter CFAR with pooling is just the same as the CP-CFAR but without the convolution operation. The detection results are shown in Table V.

From Table V, it can be seen that, in terms of both the detection time and detection accuracy, the proposed CP-CFAR can obtain the better performance. It is worth noting that the two-parameter CFAR with pooling has one step less but takes more time consumption. The main reason is that the two-parameter CFAR with pooling will produce more false alarms after CFAR detection, so the time consumption of false alarm elimination via morphological operations will increase significantly, resulting in an increase of the overall time.

The comparison results also show that, although the proposed CP-CFAR mainly reduces the input dimension through pooling to improve the detection speed, if there is only pooling operation, it will lead to information loss and detection performance reduction. Therefore, the CP-CFAR adds the convolution operation before the pooling operation, in order to preserve the target information via feature extraction, which can improve the detection speed on the basis of high detection accuracy.

TABLE IV
COMPARISON BETWEEN CPU, CPU MULTITHREADS AND GPU PARALLEL MODES OF CA-CFAR, TWO-PARAMETER CFAR, AND CP-CFAR

Methods	Images	Time (ms)			Speedup		FoM
		CPU	CPU-MT	GPU	$Speedup_1$	$Speedup_2$	
CA-CFAR	Fig. 9 (a)	1653	1329	1340	1.24	1.23	100%
	Fig. 9 (b)	1746	1366	1350	1.28	1.29	53.3%
	Fig. 9 (c)	1728	1344	1345	1.28	1.28	45.5%
	Fig. 9 (d)	1739	1291	1340	1.34	1.30	52.4%
Two-parameter CFAR	Fig. 9 (a)	18140	3423	1560	5.30	11.63	100%
	Fig. 9 (b)	17970	3545	1550	5.07	11.59	100%
	Fig. 9 (c)	18050	3347	1507	5.39	11.98	83.3%
	Fig. 9 (d)	18156	3641	1515	4.99	11.98	80%
CP-CFAR	Fig. 9 (a)	3580	937	191	3.82	18.74	100%
	Fig. 9 (b)	3590	951	192	3.77	18.70	100%
	Fig. 9 (c)	3576	972	192	3.68	18.60	100%
	Fig. 9 (d)	3546	997	191	3.56	18.57	100%

TABLE V
COMPARISON BETWEEN TWO-PARAMETER CFAR WITH 4-TO-1 POOLING AND CP-CFAR

Methods	Images	Times of GPU	FoM
Two-parameter CFAR with Pooling	Fig. 9 (a)	396	90%
	Fig. 9 (b)	394	73.3%
	Fig. 9 (c)	395	50%
	Fig. 9 (d)	394	52.4%
CP-CFAR	Fig. 9 (a)	191	100%
	Fig. 9 (b)	192	100%
	Fig. 9 (c)	192	100%
	Fig. 9 (d)	191	100%

V. CONCLUSION

In this paper, an effective way to achieve the real-time SAR target detection in airborne SAR images by combining CFAR and convolution operation is proposed. In order to reduce the computational complexity, the convolution layer and the pooling layer are added before the two-parameter CFAR in the CP-CFAR method, which can reduce the processing dimension of the images without losing the main feature of original image. In order to improve the detection accuracy, the test results were subjected to erosion and dilation operations and median filtering after CFAR detection. In order to improve the detection speed, the GPU is used to accelerate the program in parallel.

The proposed CP-CFAR method can detect the targets in SAR images with high accuracy and high speed. Compared with the traditional CA-CFAR, two-parameter CFAR and their CPU, multithread CPU, and GPU acceleration modes, the proposed CP-CFAR with GPU acceleration can obtain the best detection performance with the highest acceleration ratio and can meet the requirements of real-time target detection.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable and helpful suggestions to this paper.

REFERENCES

- [1] F. Zhou, L. Wang, X. Bai, and Y. Hui, "SAR ATR of ground vehicles based on LM-BN-CNN," *IEEE Trans. Geosci. Remote Sens.*, vol. 99, no. 99, pp. 1–12, 2018.
- [2] Z. Cui, Z. Cao, J. Yang, and J. Feng, "Target recognition in synthetic aperture radar images via non-negative matrix factorisation," *IET Radar Sonar Navigat.*, vol. 9, no. 9, pp. 1376–1385, 2015.
- [3] K. El-Darymli, E. W. Gill, P. McGuire, D. Power, and C. Moloney, "Automatic target recognition in synthetic aperture radar imagery: A state-of-the-art review," *IEEE Access*, vol. 4, pp. 6014–6058, 2017.
- [4] T. Li and L. Du, "Target discrimination for SAR ATR based on scattering center feature and k-center one-class classification," *IEEE Sens. J.*, vol. 18, no. 6, pp. 2453–2461, Mar. 2018.
- [5] B. Hou, X. Chen, and L. Jiao, "Multilayer CFAR detection of ship targets in very high resolution SAR images," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 4, pp. 811–815, Apr. 2014.
- [6] B. Zou, L. Li, and P. He, "Fast automatic target detection of large scene SAR image based on parallel computing," in *Proc. Int. Conf. Comput. Des. Appl.*, 2010, pp. V5-495–V5-499.
- [7] T. Li, Z. Liu, R. Xie, and L. Ran, "An improved superpixel-level CFAR detection method for ship targets in high-resolution SAR images," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 1, pp. 184–194, Jan. 2018.
- [8] J. Ai, X. Yang, J. Song, Z. Dong, L. Jia, and F. Zhou, "An adaptively truncated clutter-statistics-based two-parameter CFAR detector in SAR imagery," *IEEE J. Ocean. Eng.*, vol. 43, no. 1, pp. 267–279, Jan. 2018.
- [9] W. Yu, Y. Wang, H. Liu, and J. He, "Superpixel-based CFAR target detection for high-resolution SAR images," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 5, pp. 730–734, May 2016.
- [10] T. Vogel, "All the way to CUDA," *Comput. Sci. Eng.*, vol. 15, no. 5, pp. 6–8, 2013.
- [11] W. Liu, B. Schmidt, and W. Muller-Wittig, "CUDA-BLASTP: Accelerating BLASTP on CUDA-enabled graphics hardware," *IEEE/ACM Trans. Comput. Biology Bioinform.*, vol. 8, no. 6, pp. 1678–1684, Nov./Dec. 2011.
- [12] X. Guo, J. Wu, Z. Wu, and B. Huang, "Parallel computation of aerial target reflection of background infrared radiation: Performance comparison of openMP, openACC, and CUDA implementations," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 4, pp. 1653–1662, Apr. 2016.
- [13] Y. Liang, H. P. Huynh, K. Rupnow, R. S. M. Goh, and D. Chen, "Efficient GPU spatial-temporal multitasking," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 3, pp. 748–760, Mar. 2015.
- [14] W. Xue and Y. Jiang, "Simulation of fast threshold CFAR processing algorithm with CUDA," in *Proc. Int. Conf. Comput. Distrib. Control Intell. Environ. Monit.*, 2012, pp. 621–624.
- [15] C. P. Chandgude and P. M. Chawan, "Ship detection from SAR imagery using CUDA and performance analysis of the system," *Int. J. Eng. Res. Appl.*, vol. 4, no. 7, pp. 223–228, 2014.
- [16] C. P. Schwegmann, W. Kleyhans, B. P. Salmon, and L. Mdkane, "A CA-CFAR and localized wavelet ship detector for sentinel-1 imagery," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2015, pp. 3707–3710.

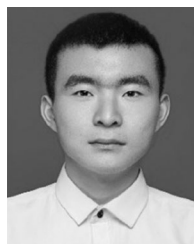
- [17] D. K. Mahapatra, K. R. Pradhan, and L. P. Roy, "An experiment on MSTAR data for CFAR detection in lognormal and Weibull distributed SAR clutter," in *Proc. Int. Conf. Microw., Opt. Commun. Eng.*, 2016, pp. 377–380.
- [18] A. Izzo, M. Liguori, C. Clemente, C. Galdi, M. D. Bisceglie, and J. J. Soraghan, "Multimodel CFAR detection in foliage penetrating SAR images," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 53, no. 4, pp. 1769–1780, Aug. 2017.
- [19] A. Buono, F. Nunziata, L. Mascolo, and M. Migliaccio, "A multipolarization analysis of coastline extraction using x-band cosmo-skymed SAR data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 7, pp. 2811–2820, Jul. 2014.
- [20] Y. Chen, L. Liu, Z. Gong, and P. Zhong, "Learning CNN to pair UAV video image patches," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 12, pp. 5752–5768, Dec. 2017.
- [21] H. B. Fredj, M. Ltaif, A. Ammar, and C. Souani, "Parallel implementation of Sobel filter using CUDA," in *Proc. Int. Conf. Control, Automat. Diagnosis*, 2017, pp. 209–212.
- [22] T. V. Hoang, X. Cavin, and D. W. Ritchie, "gEMfitter: A highly parallel FFT-based 3D density fitting tool with GPU texture memory acceleration," *J. Struct. Biol.*, vol. 184, no. 2, pp. 348–354, 2013.
- [23] X. Meng, L. X. Guo, and T. Q. Fan, "Parallelized TSM-RT method for the fast RCS prediction of the 3-D large-scale sea surface by CUDA," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 10, pp. 4795–4804, Oct. 2015.
- [24] M. Huang, B. Huang, Y. L. Chang, J. Mielikainen, H. L. A. Huang, and M. D. Goldberg, "Efficient parallel GPU design on WRF five-layer thermal diffusion scheme," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 5, pp. 2249–2259, May 2015.
- [25] M. Moreaud and F. Itthirad, "Fast algorithm for dilation and erosion using arbitrary flat structuring element: Improvement of Urbach and Wilkinson's algorithm to GPU computing," in *Proc. Int. Conf. Multimedia Comput. Syst.*, 2014, pp. 289–294.
- [26] H. Zhao, D. Gao, M. Wang, and Z. Pan, "Real-time edge-aware weighted median filtering on the GPU," *Comput. Graph.*, vol. 61, pp. 11–18, 2016.
- [27] S. Yang, "Improved bilinear interpolation method for image fast processing," in *Proc. Int. Conf. Intell. Comput. Technol. Automat.*, 2014, pp. 308–311.
- [28] Z. Cui, C. Tang, Z. Cao, and S. Dang, "SAR unlabeled target recognition based on updating CNN with assistant decision," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 10, pp. 1585–1589, Oct. 2018.
- [29] C. Zhang, H. Zhang, C. Wang, Q. Fu, and L. Xu, "A novel ship detection method based on Shannon entropy in Chinese Gaofen-3 fully polarimetric SAR images," in *Proc. Prog. Electromagn. Res. Symp.-Fall*, 2017, pp. 909–916.
- [30] D. R. Rinku and M. A. Rani, "Analysis of multi-threading time metric on single and multi-core CPUs with matrix multiplication," in *Proc. Int. Conf. Adv. Elect., Electron., Inf., Commun. Bio-Inform.*, 2017, pp. 152–155.
- [31] G. Hong, S. Kang, S. K. Chang, and J. K. Min, "Efficient parallel join processing exploiting SIMD in multi-thread environments," *IEICE Trans. Inf. Syst.*, vol. E101.D, no. 3, pp. 659–667, 2018.



Zongyong Cui (S'13–M'15) received the B.E. and Ph.D. degrees in signal and information processing from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2007 and 2015, respectively.

From 2013 to 2014, he was a Visiting Student with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. He is currently a Lecturer with the School of Information and Communication Engineering, UESTC. His research interests include radar image processing, target recognition, and machine learning.

Dr. Cui is a Reviewer of the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS, the *IET Radar, Sonar and Navigation*, and the *Journal of Electronic Imaging*.



Hongbin Quan (S'16) received the B.E. degree in electronic information science and technology from Zhengzhou University, Zhengzhou, China, in 2016. He is currently working toward the M.S. degree in signal and information processing with the University of Electronic Science and Technology of China, Chengdu, China.

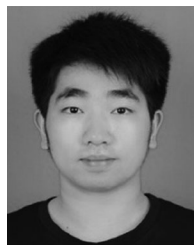
His research interests include parallel computing, synthetic aperture radar target detection, and image processing.



Zongjie Cao (M'10) received the B.E. and Ph.D. degrees from the Xi'an Jiaotong University, Xi'an, China, in 1999, and 2005, respectively.

From 2006 to 2008, he was a Postdoctoral Researcher with the Communication and Information System Postdoctoral Center, University of Electronic Science and Technology of China (UESTC), Chengdu, China. In 2008, he joined the School of Electronic Engineering, UESTC, where he is currently a Professor with the School of Information and Communication Engineering. He has authored or coauthored more than 50 papers. His current research interests include radar signal processing, synthetic aperture radar imaging, and target recognition.

Dr. Cao is a Reviewer for several international journals and conferences, such as the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING, the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS, the *IET Radar, Sonar and Navigation*, and *Remote Sensing*.



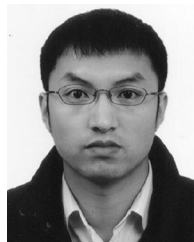
Shengping Xu (S'18) received the B.E. degree in electronic science and technology from the Jiangxi University of Science and Technology, Ganzhou, China, in 2017. He is currently working toward the M.S. degree in signal processing with the University of Electronic Science and Technology of China, Chengdu, China.

His research interests include parallel computing, machine learning, and image processing.



Chunmei Ding (S'16) received the B.E. degree in electronic and information engineering from the Shandong University of Science and Technology, Qingdao, China, in 2016. She is currently working toward the M.S. degree in electronics and communication engineering with the University of Electronic Science and Technology of China, Chengdu, China.

Her research interests include the correlation of between the system resources and target detection.



Junjie Wu (S'06–M'13) received the B.E., M.S., and Ph.D. degrees in electronic engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2004, 2007, and 2013, respectively.

From 2012 to 2013, he was a Visiting Student with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA. He is currently an Associate Professor with UESTC. His research interests include signal processing and radar imaging.

Dr. Wu is a Reviewer of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, the *IET Radar, Sonar and Navigation*, the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS, and the *EURASIP Journal on Wireless Communications and Networking*.