

# 基于 CPU-GPU 的有序统计类恒虚警检测优化

火静斌<sup>1</sup>, 张晓滨<sup>1</sup>, 田 泽<sup>2</sup>

(1. 西安工程大学, 陕西 西安 710048;

2. 集成电路与微系统设计航空科技重点实验室, 陕西 西安 710068)

**摘 要:** 雷达信号处理算法的高性能实现是雷达系统设计中的关键技术。而恒虚警检测技术是雷达信号目标检测系统中控制虚警率的重要手段之一, 也是最耗费系统资源的地方。传统的恒虚警检测技术主要采用 DSP 和 FPGA 等定制化设备, 但存在的问题主要有开发周期长、调试难度大、耗费资源。为了满足脉冲多普勒雷达回波数据的实时处理需求以及国产化 GPU 的生态扩充, 文章针对恒虚警检测技术分析了有序统计类恒虚警检测方法 (OSCFAR) 的可并行性问题, 并基于 OpenCL 平台提出了对 OSCFAR 进行 GPU 加速的方法。该方法中提出了 OSCFAR 在 GPU 中避免条件分支的并行化技术, 优化了适用于 GPU 的并行化排序方法, 减少了系统访问全局内存所花费的时间。最后从性能测试和误差分析角度评估了 OSCFAR 的实时性和准确性, 实验结果表明, 在所使用的硬件平台上相比于传统 CPU 实现达到了 60 倍以上的加速比, 处理精度可以达到与原有方案相同的水平。

**关键词:** 雷达信号处理; 并行计算; 恒虚警检测; 双调排序; OpenCL

中图分类号: TP391

文献标识码: A

文章编号: 1673-629X(2023)04-0040-06

doi: 10.3969/j.issn.1673-629X.2023.04.006

## Order Statistics Constant False-alarm Optimization Based on CPU-GPU

HUO Jing-bin<sup>1</sup>, ZHANG Xiao-bin<sup>1</sup>, TIAN Ze<sup>2</sup>

(1. Xi'an Polytechnic University, Xi'an 710048, China;

2. Integrated Circuit and Microsystem Design Key Laboratory of Aviation Science and Technology,  
Xi'an 710068, China)

**Abstract:** High-performance implementation of radar signal processing algorithms is a key technology in radar system design. Constant False Alarm Rate (CFAR) detection is one of the important means to control the false-alarm rate in the radar signal target detection system. The traditional CFARs use customized equipment such as DSP and FPGA, but the long development cycle, difficult debugging, and resource consumption are the main problems. To meet the real-time processing requirements of pulsed Doppler radar echo data and the ecological expansion of domestic GPUs, we analyze the parallelism problem of the Ordered Statistical Constant False Alarm Rate (OSCFAR). Based on the platform accelerated OSCFAR on GPU, we propose a parallelization technique for OSCFAR to avoid conditional branches in GPU, optimize the parallelization sorting method suitable for GPU, and reduce the time it takes for the system to access global memory. Finally, the real-time performance and accuracy of OSCFAR are evaluated from the perspective of performance testing and error analysis. Compared with the traditional CPU on the hardware platform, the results show that the acceleration ratio is more than 60 times, and the processing accuracy can reach the same level as the original one.

**Key words:** radar signal processing; parallel computing; constant false alarm rate; bitonic sort; OpenCL

## 0 引言

在雷达系统中, 恒虚警率检测技术 (CFAR) 通过动态调整检测门限来检测背景杂波干扰下的目标。为了估计杂波功率, 需要对每个雷达回波样本进行大量的计算。雷达系统需要高吞吐量和低延迟, 传统的

CFAR 处理通常使用现场可编程门阵列 (FPGA) 或数字信号处理器 (DSP)<sup>[1-2]</sup>, 但也存在一定的问题, 如开发周期长、调试难度大、耗费资源等。随着 GPU 统一渲染架构的出现, 基于 CPU-GPU 的异构计算体系结构开始普及, OpenCL、CUDA 等开发平台简化了 GPU

收稿日期: 2022-06-16

修回日期: 2022-10-18

基金项目: 西安工程大学研究生创新基金 (chx2022021)

作者简介: 火静斌 (1998-), 男, 研究生, 研究方向为并行计算; 通信作者: 张晓滨 (1970-), 男, 硕士, 副教授, 研究方向为大数据分析 with 智能计算技术。

编程难度,基于异构计算平台的 GPU 开发环境已广泛应用于雷达信号处理领域<sup>[3-6]</sup>。

在雷达应用中,有多种方法可以实现恒虚警检测,比如经典的单元平均恒虚警检测技术(Cell Averaging CFAR, CACFAR)、平均选大(Greatest of CFAR, GOCFAR)和平均选小恒虚警检测技术(Smallest of CFAR, SOCFAR)。但是在需要多目标或高性能的应用环境中,有序统计类恒虚警检测技术的应用范围更广<sup>[7]</sup>。与其他检测方法不同的是,OSCFAR 的内部实现更为复杂,属于计算密集型方法,需要高性能计算设备。文献[8]在 Tesla C1060 GPU 上研究了 PD 雷达中 CACFAR 的 GPU 实现,通过对算法进行了优化,提高了算法的并行性,达到了一定的加速效果。文献[9]利用 OpenCL 在集成显卡以及中等级别产品 AMD GPU 实现 CACFAR,但由于使用零填充方法,容易造成程序虚警。文献[10]利用积分图像算法加速了并行均值滤波算法,但需要两个内核函数上分别实现计算并行前缀和以及平均值,会产生额外的计算以及对共享内存的不必要访问。文献[11]在 GPU 端采用 Blleloch 扫描算法来对 CA-CFAR 求和过程进行优化,避免了重复计算,但存在的问题是数据必须是以二的指数幂格式传输。文献[12]提出的局部最大最小快速滤波是一种充分考虑数据重用的算法,可以用来解决 GOCFAR、SOCFAR 的数据重用,但存在边界溢出的问题。

以上都是对均值类恒虚警检测算法进行了 GPU 加速实现,而对 OSCFAR 加速的研究还比较少。文献[13]在 FPGA 上设计实现了可以规避排序算法的 OSCFAR,但是如果需要排序值计算信噪比时不能满足系统要求。文献[14]设计了重复快速排序算法,但快速排序不适合在 GPU 上进行计算。文献[15]提出了具有分布式直方图的 OSCFAR 的计算方法,但是需要一定的计算量。文献[16]提出了基于“扫雷算法”的 OSCFAR 算法,适合并行计算,但是需要利用第三方工具做预处理排序。

该文的研究重点是在 OSCFAR 中使用 GPU 进行加速,提出了一种改进的 OSCFAR 方法,使其适用于 GPU 并行实现。首先,设计了一个预处理程序,对每个多普勒通道进行数据扩充。在预处理程序中,为了减少 GPU 处理中的分支操作,对每个多普勒通道数据进行扩充,大小为左右保护单元以及参考单元数量之和,从而优化设备计算效率。其次,设计并实现了 CPU-GPU 异构架构下 OSCFAR 的并行加速方法。根据 GPU 的显存大小,将原始数据按时间顺序划分为若干个子数据并传输到 GPU。为了优化计算效率,预处理程序分配给 CPU 执行,之后 OSCFAR 由 GPU 执行。

为了证明所提出方法的可行性,在雷达模拟噪声信号为瑞利噪声背景条件下,对有序统计类恒虚警检测方法进行了仿真实验。在实验中,所提出的方法消耗时间是仅在 CPU 上运行的 OSCFAR 所消耗时间的 1/60,且目标信号质量并没有降低。

## 1 有序统计类恒虚警处理方法分析

在瑞利背景条件下,假设  $v(t)$  是单脉冲检测中某个分辨单元的一个观测值,当使用平方律检波器进行检测时,可写成如下形式:

$$D(v) = I^2(v) + Q^2(v) \quad (1)$$

其中,  $I(v)$  表示信号的同相分量,  $Q(v)$  表示信号的正交分量。在一般的杂波环境中,可以认为接收到的杂波的包络服从瑞利分布。在均匀的瑞利杂波背景条件下,单元平均方法就是利用检测单元周围的前沿滑窗和后沿滑窗中的一组独立同分布的参考单元采样的平均值来估计杂波功率水平的。有序统计类(Order Statistics) OSCFAR 是采用统计手段来估计杂波水平功率。该方法和常规均值类 CFAR 的区别是其在多目标环境下检测性能较好,但是在杂波边缘环境和均匀环境下会存在一定的损失,如图 1 所示。

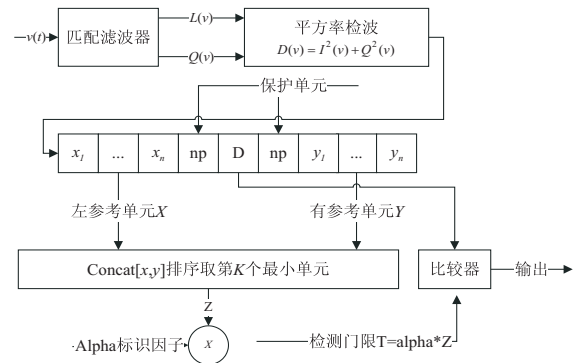


图 1 有序统计类 CFAR 检测方法

数据经过平方率检波得到距离多普勒功率谱,对每个待检测单元  $D$  选取参考单元  $(x_i(i=1,2,\dots,n), y_j(j=1,2,\dots,n))$ ,为了防止目标的能量泄露到参考单元造成杂波功率估计值较高,一般在检测单元两侧选取一定的保护单元(np)。OSCFAR 算法是将待检测单元两侧参考单元(ns)拼接后进行排序,得到一个递增序列。然后取第  $K$  个有序值作为杂波水平的估计值,一般条件下  $K$  的取值和 rate 相关,具体计算方式为  $K = \text{rate} * 2n$ 。OSCFAR 的虚警概率 PFA 与门限因子  $\alpha$  的关系如下:

$$\text{PFA} = K \binom{2n}{k} \frac{\Gamma(2n - k + 1 + \alpha) \Gamma(k)}{\Gamma(2n + 1 + \alpha)} \quad (2)$$

一般条件下,虚警概率 PFA 为固定概率值,由此计算得到门限监测因子  $\alpha$ ,检测门限  $T$  为门限监测因子与有序值  $K$  的乘积。将待检测目标单元与检测门限

进行比较,如果待检测单元的功率值大,则判决为目标,否则判为杂波。

从有序统计类恒虚警检测图中可看出,均值类恒虚警检测方法 CACFAR 是每一个采样点需要完成一次求和运算,这样可以通过并行扫描的方式避免大量的重复运算。例如 Blleloch 算法是在 GPU 端计算扫描的方法,避免了 CPU 完成扫描处理时主机段与设备端之间的传输时间。而 OSCFAR 与 CACFAR 的区别在于中间的排序过程,会耗费系统大量的资源。为此,如何选择适合 GPU 并行的排序算法以及对完整流程的并行化设计是 OSCFAR 加速研究的重点。

## 2 基于 GPU 的有序统计类恒虚警检测并行化技术

脉冲多普勒雷达信号处理系统常用的算法模块包括脉冲压缩、动目标信号显示、动目标检测、恒虚警检测。雷达信号在经过前三个步骤之后可以得到大小为  $N_p * N_s$  的二维矩阵,针对该矩阵进行距离维 CFAR 检测时,各个脉冲之间不存在联系,可以同时进行  $N_p$  个脉冲的 CFAR 检测,因此存在数据集的并行。针对有序统计类的检测算法而言,处理主要有平方率检波、每个单元的杂波功率估计(包括窗口排序)、检测门限计算以及待检测单元的比较,都可以由 GPU 的多个线程完成。因此,OSCFAR 具有较好的数据以及线程级别的可并行性。

### 2.1 有序统计类恒虚警检测的 GPU 实现

基于 OpenCL 的 OSCFAR 处理流程如图 2 所示,采用以单个脉冲为单位进行 OSCFAR,单个脉冲内的平方率检波,两侧参考单元的排序,以及待检测单元的分支可以由多个线程并行实现。

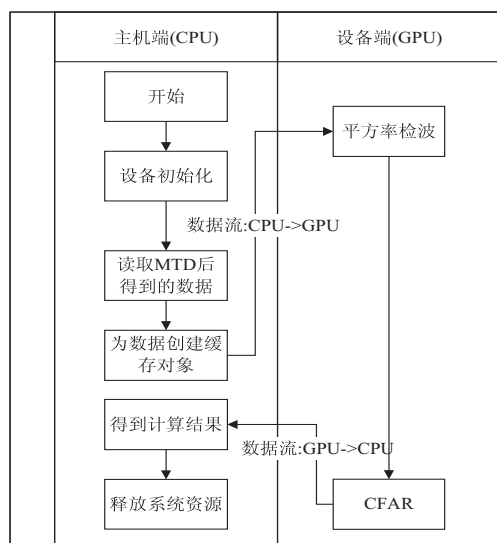


图 2 基于 OpenCL 的 OSCFAR 处理流程

大小为  $N_p * N_s$  的数据从主机端传到设备端后,按

照流程可以将算法分解为两个模块依次对应两个内核函数,第一个内核函数平方率检波主要将复数信号转化为实数信号,数据格式为实部虚部交替存储,即连续两个数据分别是信号的实部和虚部,一般采用 OpenCL 内建矢量数据类型表示。若采用一维寻址的方式,设置合适的 workgroup 大小  $p$ ,则需要的 workgroup 数量为  $N_p * N_s / p$  记为  $q$ ,每个线程完成一个平方律检波计算,总的计算组织架构如图 3 所示。

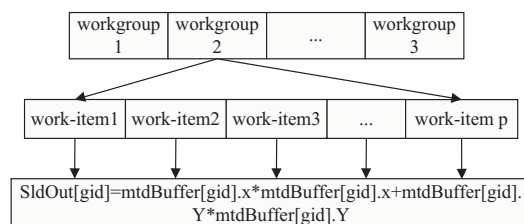


图 3 平方率检波处理流程

第二个内核函数主要用于计算杂波功率水平估计以及与实际值进行判断,若采用一维寻址的方式,设置 workgroup 大小,每个线程进行一个信号的杂波水平估计与判断。假设单侧参考单元大小为  $nr$ ,单侧保护单元大小为  $np$ ,由此可以将距离维度的数据划分为三个部分:对于前  $nr + np$  个元素来说,如果要填充左右数据的话可以从右侧取  $2 * (nr + np)$  个元素对其排序后将第  $K$  个有序值作为待检测单元的估计值,对于最后  $nr + np$  个元素取其左侧  $2 * (nr + np)$  个元素,中间部分的话直接从左右两侧各取  $nr + np$  个元素。将两侧数据排序取  $K$  值作为杂波功率的估计值,最后将估计值与实际值作对比确定是否为目标对象。

### 2.2 算法优化

优化 1:数据填充。

在估计杂波功率水平时,需要将距离维度的数据划分为三个部分,即需要三个分支操作来进行各自的计算。由于 GPU 的设计不善于处理分支判断,因此需要将每个线程进行统一操作。采取的办法是数据扩充,扩充的方法如图 4 所示。如果填充数据零的话,对于均值类 CFAR 和有序统计类 OSCFAR 来说,将导致杂波功率估计值变小,从而导致检测门限变小容易造成虚警<sup>[9]</sup>。因此,可以采取的方法是将原数据做复制,不论是镜像复制还是按序复制都会对这两类数据结果产生影响较小,在实验中采用按序复制一侧的方法。

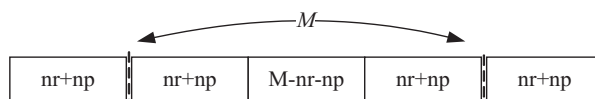


图 4 数据填充方式

优化 2:排序分析。

得到数据后将两侧数据合并,利用合适的排序算

法得到杂波功率的估计值。采取的排序算法可以是传统的冒泡排序算法以及快速排序,由于只需要递增排序前  $K$  个元素,对于冒泡排序来说,时间复杂度可以降低为  $O(K \times N)$ 。当数据量过大时,排序过程可能成为程序瓶颈,快速排序具有良好的执行效率可以提高性能,但由于 OpenCL 平台限制, GPU 执行的机器代码仅包含分支和循环,一般情况下,对于函数调用来说编译器会将函数转换为内联函数,但是对于递归调用来说无法做到。因此,对于高效的快速排序无法应用到本项目场景中。

为此,解决的办法是采用双调排序。双调排序是基于 Batchier 定理的一种适用于数据并行的排序算法,首先需要生成一个双调序列(有一个非严格递增序列和一个非严格递减数列),然后将该序列划分为两个子序列(左边的双调序列小于右边的序列),继续对每个子序列进行划分直到序列长度为 1,就可以得到单调递增的序列。

优化 3:内存优化。

对于平方率检波内核函数来说,AMD 系列 GPU 支持的最大 workgroup 大小为 256,为此可以将交替存

储的实虚数拷贝到本地内存中,减少函数对全局内存的访问,提高设备计算能力。除此之外,可以在程序中使用页锁定(Page-Locked)内存分配方式,操作系统不需要对其分页也不需要将其交换到磁盘上,可以将其一直置于物理内存中, GPU 根据其物理地址直接与主机进行数据的复制与传递,可以获得更好的加速性能。

对齐访问全局内存对于 AMD 系列以及 Nvidia 系列 GPU 来说都会提高 GPU 性能。本实验中,在主机端对数组数据进行手工填充比较复杂,该文采用了 OpenCL 引入的 clEnqueueWriteBufferRect 命令将主机端数据拷贝到设备端。在创建缓存时,用来确定的列数是满足对齐后所需的数据。对于一维的大小为 256 的 workgroup 来说,列数应该四舍五入到最近的 256 的倍数。

### 3 实验评估

实验中用到的 GPU 硬件平台分别为 Intel(R) HD Graphics520 和 AMD Radeon R7 M370。CPU 为 Core(TM)i5-6200U CPU@2.30 GHz。所使用的数据是由 Matlab 仿真的线性调频信号,其主要参数如表 1 所示。

表 1 模拟信号参数

参数名称	参数大小
雷达射频/Hz	$3.140 \times 10^{9/2}$
回波脉冲数	16
发射信号带宽/s	$2.0 \times 10^6$
发射信号时宽(脉冲宽度)/s	$42.0 \times 10^{-6}$
雷达发射脉冲重复周期/s	$4.096 \times 10^{-3}$
采样频率/Hz	$2.0 \times 10^6$
噪声功率/DB	12

#### 3.1 优化实验结果对比

由之前的模拟数据可知,采样频率( $f_s$ )为  $2.0 \times 10^6$ ,雷达发射脉冲重复周期(PRT)为  $4.096 \times 10^{-3}$ ,则单个脉冲周期的采样点数  $f_s \times \text{PRT}$  为 8 192。脉冲数量为 16,则总的采样点数为  $16 \times 8 192$ 。雷达信号在经过脉冲压缩、MTI、MTD 之后可以得到大小为  $16 \times 8 192$  的二维复数矩阵。由于存储的数据格式为实虚部交替存储,交由主机端存储时数据  $16 \times 8 192 \times 2$ 。首先,数据在经过平方率检波内核函数时,相邻两个数

据之间分别是采样点的实部与虚部,将二者的平方和存入该线程的寄存器中。其次,针对该矩阵进行距离维 CFAR 检测时,采用一维寻址方式,设置全局工作组大小  $16 \times 8 192$ ,由于 workgroup 中的 work-item 不存在数据共享,并未设置本地 workgroup 大小。处理杂波功率水平估计采用有序统计类方式,测试稳定后取三次实验结果,表 2 给出了 CFAR 在 CPU 和 GPU 上运行时间对比。

表 2 CPU 与 GPU 运行时间以及加速比

试验次数	CPU 实现/ms	GPU 实现/ms	加速比
1	897.46	100.37	8.94
2	888.39	103.11	8.62
3	899.44	104.25	8.63

其中,Graphics520 执行期间,两个内核函数(平方率检波,杂波估计与判别)分别耗费的时间为 0.251

ms、99.111 ms。由表 2 可以看出,并行化后的恒虚警检测在 GPU 上有着良好的性能。经过算法优化后的

OSCFAR 方法实验结果如表 3 所示。优化后系统的执行时间较优化前缩小到原来的 1/7, 实验证明, 该方法加速效果明显。

表 3 优化前后的执行事件对比

实验平台	平均执行时间/ms	加速比
优化前	102.577	8.730
优化后	14.872	60.473

### 3.2 算法优化实验结果分析

对于 OSCFAR 来说, 优化的方面分别是将条件分支通过数据扩充解决、排序算法的合理使用和内存优化。本实验采用不同的策略在 16 个雷达脉冲, 在每个脉冲的采样点数为 8 192 的条件下进行了分支与数据填充实验分析, 得到的实验结果如表 4 所示。

表 4 分支与数据填充实验分析

实验策略	CFAR 内核 执行时间/ms	主设备端 传输时间/ms
未使用数据 分支(分支)	99.111	1.009
	97.171	1.076
	100.25	1.085
数据填充 (替代分支)	18.541	1.226
	18.339	1.220
	18.256	1.277

对于数据扩充来说会影响到两个方面, 首先是在主机端(CPU)扩充了数据, 对于每个雷达脉冲的 8 192 个采样点来说, 只会在其两侧分别添加保护单元和参考单元的总数, 导致数据传输时间会受到一定的影响。其次就是减少数据流分支, 对于工作组中的线程执行同样的操作会对 GPU 执行效率产生了近 5 倍的性能提升, 对整体的优化有着显著的提升。

表 5 排序算法实验分析

实验策略	CFAR 内核执行时间/ms
冒泡排序	24.932
	25.089
	26.221
	18.541
优化后的冒泡排序	18.339
	18.256
	16.214
	15.494
双调排序	16.012

除此之外, 有序统计类恒虚警检测中的排序算法也会对 GPU 执行效率有一定影响, 实验中采用了改进过的冒泡排序和双调排序来对参考单元数量进行排序, 数据采用填充过的雷达信号数据, 实验结果如表 5

所示。分析算法中存在的瓶颈, 通过设计更加合理、更加具有并行性的排序算法, 对估计每个待检测单元的功率值的 GPU 执行效率有了近 1.5 倍的性能提升。

对于 OSCFAR 中的两个内核函数都可以在内存优化方面有一定的空间, 采用的方法就是减少对全局内存的访问, 利用本地内存提升效率。利用页锁定机制, 系统不需要进行分页也不需要置换内存, 可以获得较好的加速性能。缺点就是如果数据量过大的话, 一旦没有足够的物理内存供程序使用, 系统性能有所下降。通过循环展开将平方率检波后的二维数据映射到 OpenCL 二维空间, 提升计算部件的利用率。实验结果如表 6 所示: 算法经过页锁定机制以及内存访问对齐后的性能有所提升。

表 6 主存与循环展开实验分析

实验策略	CFAR 内核执行 时间/ms	平方率检波 执行时间/ms
未使用 内存优化	16.214	0.211
	15.494	0.218
	16.021	0.225
内存优化以及 循环展开	14.796	0.204
	14.805	0.109
	15.021	0.205

### 3.3 误差分析

图 5 展示了 OSCFAR 在 CPU 端和 GPU 端的目标检测对比结果, 二者处理结果近乎一致。在处理待检测功率估计值时将 CPU 与 GPU 的数据做误差分析, 得到的结果如图 6 所示, 其绝对误差小于  $3 \times 10^{-3}$ , 且 GPU 具有更好的数据精度。

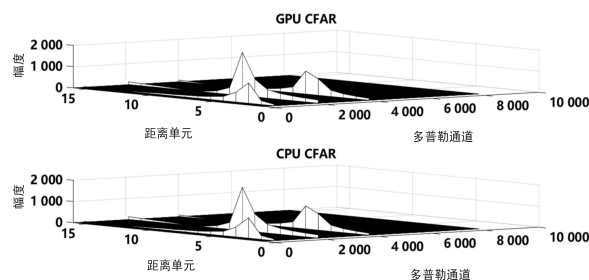


图 5 OSCFAR CPU 与 GPU 处理结果

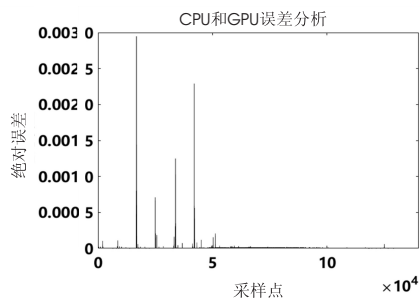


图 6 CPU 和 GPU 误差分析结果

## 4 结束语

提出了一种基于 GPU 的恒虚警检测并行处理方法,通过对原数据进行填充、采用双调排序并行化等优化策略来实现 GPU 加速。基于 OpenCL 通用异构平台的 GPU 加速满足了雷达信号处理高性能要求,在集成显卡以及中等级别的 AMD 显卡上可以达到 60 倍左右的加速比,本研究为后续基于国产 GPU 的加速积累了经验。

### 参考文献:

- [1] ZHANG M, LI X. An efficient real-time two-dimensional CA-CFAR hardware engine [C]//IEEE international conference on electron devices and solid-state circuits (EDSSC). Xi'an; IEEE, 2019: 1-3.
- [2] YANG M, YANG J, HOU Y, et al. Implementation architecture of signal processing in pulse Doppler radar system based on FPGA [J]. The Journal of Engineering, 2019 (21): 7335-7338.
- [3] 韩文俊, 王 嘎, 丁琳琳. 基于 GPU 的雷达信息处理并行设计优化技术研究 [J]. 电子技术与软件工程, 2018 (16): 82-83.
- [4] 田乾元, 徐朝阳, 赵 泉. 基于 GPU 的软件雷达信号处理 [J]. 舰船电子对抗, 2020, 43 (1): 58-63.
- [5] 顾文恺. 基于 GPU 的脉冲压缩并行化研究 [J]. 航空计算技术, 2017, 47 (2): 121-124.
- [6] PENG Pei, ZHANG Yunlei, LI Ke. Parallel mechanism study of radar pulse compression based on CPU/GPU processor [J]. Ship Electronic Engineering, 2017, 37 (10): 30-32.
- [7] DE MAIO A, AUBRY A. Radar detection, performance analysis, and CFAR techniques [C]//IEEE radar conference (radar conf). Boston; IEEE, 2019: 1-120.
- [8] VENTER C J, GROBLER H, ALMALKI K A. Implementation of the CA-CFAR algorithm for pulsed-Doppler radar on a GPU architecture [C]//IEEE Jordan conference on applied electrical engineering and computing technologies (AEECT). Amman; IEEE, 2011: 1-6.
- [9] BANTLE M, BAYERL P, FUNKEN S, et al. Radar signal processing with OpenCL on integrated graphic processors [C]//19th international radar symposium (IRS). Bonn; IEEE, 2018: 1-8.
- [10] WU X, WANG K, LI Y, et al. Accelerating haze removal algorithm using CUDA [J]. Remote Sens., 2021, 13: 85.
- [11] 龚 昊, 刘 莹, 冯建周, 等. 基于 GPU 加速的脉冲多普勒雷达信号处理 [J]. 计算机工程与科学, 2021, 43 (7): 1141-1149.
- [12] HANG J, HU S. A GPU-accelerated real-time single image de-hazing method using pixel-level optimal de-hazing criterion [J]. Journal of Real-Time Image Processing, 2012, 9 (4): 661-672.
- [13] BALES M R, BENSON T, DICKERSON R, et al. Real-time implementations of ordered-statistic CFAR [C]//2012 IEEE radar conference. Atlanta; IEEE, 2012.
- [14] DIDI J H, LEVANON N. Repeated sorting on the sliding window for OS-CFAR [J]. IET Radar, Sonar & Navigation, 2019, 13 (8): 1272-1278.
- [15] VILLAR S A, MENNA B V, TORCIDA S, et al. Efficient approach for OS-CFAR 2D technique using distributive histograms and breakdown point optimal concept applied to acoustic images [J]. IET Radar, Sonar & Navigation, 2019, 13 (12): 2071-2082.
- [16] COLENA C L, RUSSELL M J, BRAUN S A. Minesweeper: a novel and fast ordered-statistic CFAR algorithm [C]//IEEE high performance extreme computing conference (HPEC). Waltham; IEEE, 2020: 1-6.