第 47 卷第 3 期
2015 年 6 月

南 京 航 空 航 天 大 学 学 报
Journal of Nanjing University of Aeronautics & Astronautics

Vol. 47 No. 3
Jun. 2015

DOI:10.16356/j.1005-2615.2015.03.011

超高分辨率机载 SAR 成像算法及其 GPU 实现

田宵骏[1]　梁媚蓉[1]　毛新华[1,2]

(1.南京航空航天大学电子信息工程学院,南京,210016;
2.南京航空航天大学雷达成像与微波光子技术教育部重点实验室,南京,210016)

摘要:雷达成像分辨率的不断提高,给 SAR 高精度实时成像处理带来了新的挑战。采用高效精确的成像算法以及对算法进行硬件加速是解决该问题的有效途径。本文提出了一种适用于超高分辨率机载 SAR 成像的精确高效成像处理方案,并利用并行化硬件平台 GPU 对该成像方案进行了硬件加速。实测数据处理结果充分验证了该处理方案的聚焦精度和处理效率。

关键词:合成孔径雷达;超高分辨率;成像算法;图形处理器;通用并行计算架构

中图分类号:TN957.52　　**文献标志码**:A　　**文章编号**:1005-2615(2015)03-0384-08

Imaging Algorithm and Its Implementation on GPU for Ultra-High Resolution Airborne SAR

Tian Xiaojun[1], *Liang Meirong*[1], *Mao Xinhua*[1,2]

(1. College of Electronic and Information Engineering, Nanjing University of Aeronautics & Astronautics, Nanjing, 210016, China;
2. Radar Imaging and Microwave Photonic Technology Key Laboratory of Ministry of Education, Nanjing University of Aeronautics & Astronautics, Nanjing, 210016, China)

Abstract: The improvement of synthetic aperture radar (SAR) imaging resolution brings new challenges to SAR high-precise and real-time imaging processing. Selecting efficient imaging algorithm and utilizing hardware platform for algorithm acceleration are two effective methods to solve the problem. A new efficient and precise imaging algorithm is proposed for the ultra-high resolution airborne SAR. Parallel hardware platform GPU is utilized to accelerate the algorithm. The processing results of the SAR measured data fully verify the accuracy and efficiency of the new imaging algorithm.

Key words: synthetic aperture radar (SAR); ultra-high resolution; image formation algorithm; graphic processing unit (GPU); compute unified device architecture (CUDA)

合成孔径雷达(Synthetic aperture radar, SAR)能够突破传统雷达分辨率的限制,具有两维(距离和方位)的高分辨率,能对场景作高分辨率的二维成像。作为一种成像雷达,不断提高其成像分辨率始终是 SAR 成像不懈追求的目标。当前,SAR 成像分辨率已经达到 0.1 m 甚至更高(超高分辨率)。超高分辨率 SAR 成像质量高,涵盖更多的信息内容,能够极大地提高目标的识别概率,具有重要的军事和民用价值。然而,高的成像分辨率给信号处理提出了更高的要求。首先,高的分辨率要求更长的合成孔径时间,在较长的合成孔径时间内,雷达平台匀速直线运动的假设往往不再成立,

基金项目:国家自然科学基金(61301210)资助项目;江苏省自然科学基金(BK20130815)资助项目;航空科学基金(20142052021)资助项目;江苏高校优势学科建设工程资助项目。

收稿日期:2015-01-29;**修订日期**:2015-04-02

通信作者:毛新华,男,副教授,E-mail:xinhua@nuaa.edu.cn。

这对成像算法的精确度提出了更加苛刻的要求；同时，雷达回波数据会急剧增加，给实时成像处理带来了更大的难度。为了解决这两个问题，主要着手于两个方面：一方面从成像算法角度，选择设计一个高效精确的成像处理方案；另一方面通过更加高效的硬件处理平台，对大数据进行加速处理，实现实时成像。

目前，具有代表性的机载超高分辨率 SAR 系统主要有：美国 Sandia 国家实验室研制的 Lynx SAR 系统[1]；德国应用科学研究所（FGAN）开发的多功能相控阵（PAMIR）SAR 系统[2]。Lynx SAR 系统成像处理方案采用基于重叠子孔径处理的极坐标格式算法[3]（Polar format algorithm，PFA），该系统能够实现超高分辨率实时成像，但是成像处理依赖于高精度的惯导和雷达参数的实时调整；PAMIR 系统成像处理方案采用基于时域的后向反投影（Back projection，BP）算法，该算法虽然精度高，但是计算效率非常低，FGAN 为此专门设计了分布式计算系统对算法进行加速，但目前仍无法做到实时成像处理。

SAR 成像处理本质上是对复数据的处理。现代硬件平台对复数据的处理容量大、精度高且速度快，硬件内部也集成了常用复数据处理函数（矩阵转置，快速傅里叶变换等）。所以硬件平台非常适用于 SAR 实时成像处理。随着硬件处理器的发展，基于硬件平台的实时成像处理方案日趋成熟。当前主流的硬件平台有：DSP、FPGA 和 GPU。DSP 具有较强的浮点运算能力，流水线结构和易学的编程语言（C、C++）等优势，但是其串行的数据处理逻辑是其实时成像处理的瓶颈；文献[4]中采用了 FPGA＋DSP 的硬件组合，利用 FPGA 实现成像处理并结合 DSP 强大的浮点运算性能完成硬件加速，在 11 s 内完成了 16 384×32 768 采样点机载聚束式 SAR 数据处理，实现了 4 096×8 192点的实时成像。但是基于 FPGA 的雷达处理程序不具备良好的扩展性和移植性，并且编程语言学习成本较高（主要编程语言为 VHDL 和 Verilog，基于 API 的 OpenCL 语言现阶段在 FPGA 上还没有广泛得到应用），开发和代码调试周期长。

GPU 采用多核并行的数据处理方式，非常适合大数据的实时快速处理；基于 CUDA C 的编程起点低，同时具有良好的扩展性和移植性，因此很多学者提出基于 GPU 的 SAR 实时成像处理方案。文献[5]对常用的 3 种经典成像算法：距离多普勒（Range doppler，RD），CS（Chirp scaling）和 ωk 算法提出了 3 种基于 GPU 的加速方案，并获得了每秒约 36 兆采样点的实时数据处理速度。

本文主要完成了两方面工作：一方面提出了一种基于 PFA 和分块两维自聚焦（2-D AutoFocus）的超高分辨率 SAR 成像处理方案；另一方面利用并行化处理平台 GPU 对该方案进行硬件加速。最后通过实测数据验证了该方案的聚焦精度和处理效率。

1　成像处理方案

在超高分辨率和大回波数据的双重条件下，成像算法需兼备高精度和高效率的特点。频域成像算法如距离徙动算法（Range migration algorithm，RMA）[6]具有良好的成像精度和效率，但是无法适用于非理想航迹运动下的成像处理；时域算法如 BP 具有良好的成像精度，但是效率较低。PFA 则是在时域进行运动补偿，易于校正雷达平台非理想航迹运动，在兼顾较高成像精度的同时，算法流程简洁高效，适合超高分辨率 SAR 成像处理。

合成孔径时间变长，雷达平台不满足匀速直线的理想运动假设，会导致方位向的散焦和额外的距离徙动。两者在 SAR 成像算法处理过程中无法得到相应的补偿，同时残留距离徙动还会引起图像距离向出现二次散焦，所以实际上 SAR 图像中残留的相位误差本质上是一种两维误差。利用精确的惯导或者自聚焦算法是当下解决该问题的两种常用方案。但是实际中很难获取精确的惯导信息，所以自聚焦是实现超高分辨率 SAR 成像的关键。当额外距离徙动没有超过一个距离分辨单元时，此时往往忽略该距离徙动，只需估计和补偿方位向一维相位误差；当额外距离徙动超过一个距离分辨单元时，需要两维自聚焦算法进行估计补偿。传统的两维自聚焦算法如文献[7,8]中提出的方法没有考虑两维相位误差的内部解析结构，而是假设两维相位误差是完全未知的，因此是对两维相位误差的一种盲估计。这种盲估计方法由于估计的参数非常多，在实际应用中存在两个主要缺陷：首先，由于要对所有参数同时进行估计，估计算法计算量非常大；其次，数据中通常没有足够的冗余度来精确估计如此多的参数。实际上，残留的两维相位误差并不是完全未知的，其内部存在特定的解析结构，如果能够利用这种先验结构信息，就可以对估计参数空间进行降维处理，从而减少估计的计算量同时提高估计精度。文献[9]分析了 PFA 处理框架下残留两

维相位误差的解析结构，并在此基础上提出了一种降维两维自聚焦算法，很好地解决了两维相位误差的高效精确估计问题。

综上考虑，在超高分辨率机载 SAR 成像处理中，本文采用 PFA 作为成像处理算法，同时采用基于先验知识的两维自聚焦算法对 PFA 残留误差进行补偿。考虑到 PFA 残留误差的空变性，采用了分块处理的策略，即对 PFA 图像先分块进行自聚焦处理，然后再拼接得到精确聚焦的整个图像。为了消除相邻子块边界目标的散焦问题，相邻子块选取时会有部分重叠。整个处理流程如图 1 所示。

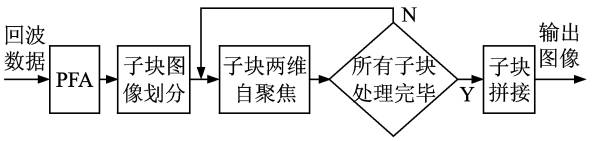


图 1 超高分辨率机载 SAR 成像处理方案

Fig. 1 Imaging solution of ultra-high resolution airborne SAR

上述处理流程中核心步骤是 PFA 和两维自聚焦，因此下面对这两个部分进行更详细的介绍。

1.1 PFA

PFA 是经典的聚束式 SAR 成像算法。聚束式 SAR 成像数据采集模型如图 2 所示。以场景中心为坐标中心建立三维直角坐标系，考虑实际情况，飞机以非直线轨迹运动，瞬时的方位角和俯仰角分别为 θ 和 φ（孔径中心时刻为 θ_c 和 φ_c），成像平面内的目标坐标为 $(x_t, y_t, 0)$，与飞机的斜距为 R_t。场景中心与飞机的斜距为 R_a。

回波信号经过解调，距离向做 Fourier 变换并进行匹配滤波和运动补偿后，得到的信号表达式可

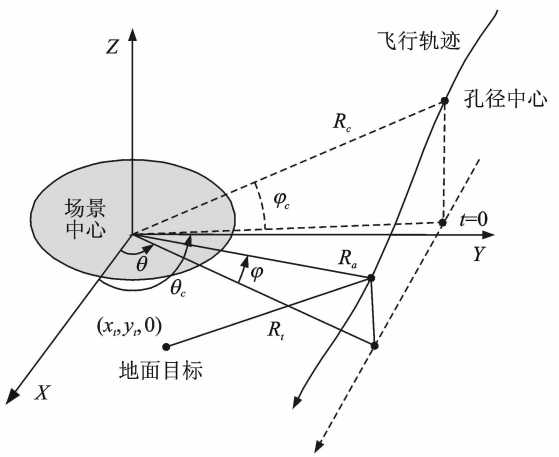


图 2 聚束式 SAR 成像数据采集模型

Fig. 2 Data acquisition model of spotlight SAR

以表示为

$$s(t, f_r) = \exp\left[\mathrm{j}\frac{4\pi}{c}(f_c + f_r)(R_a - R_t)\right] \quad (1)$$

式中：c 为电波的传播速度；f_c 为线性调频信号的载频；f_r 为距离向采样频率；$R_a - R_t$ 为雷达平台到场景中心的距离和此刻平台到目标的距离之差。

结合图 1 中的几何关系以及 PFA 基于平面波前的假设，$R_a - R_t$ 可以近似表示为

$$R_a - R_t \approx x_t\cos\varphi\cos\theta + y_t\cos\varphi\sin\theta \quad (2)$$

将式(2)代入式(1)中，并用空间波数域表示，即

$$S(K_x, K_y) = \exp[\mathrm{j}(x_t K_x + y_t K_y)] \quad (3)$$

其中 K_x 和 K_y 为

$$K_x = \frac{4\pi}{c}(f_r + f_c)\cos\varphi\cos\theta$$

$$K_y = \frac{4\pi}{c}(f_r + f_c)\cos\varphi\sin\theta \quad (4)$$

分别表示方位和距离空间频率。

从式(3)可以看出，雷达的回波信号与目标函数之间存在两维 Fourier 变换关系，只需要对运动补偿和匹配滤波过后的数据作两维 Fourier 变换即可成像。但是实际中回波数据采样点在 (t, f_r) 域是均匀分布的，映射到空间频域 (K_x, K_y) 上则是按照极坐标格式排列。为了利用 FFT 快速实现离散傅里叶变换，提高算法的运算效率，要求通过距离向和方位向的两维数据重采样，将极坐标格式数据转换为矩形格式，再对重采样过后的矩形数据作两维 IFFT 即可实现对目标的成像。因此，可以得到 PFA 算法处理流程如图 3 所示。

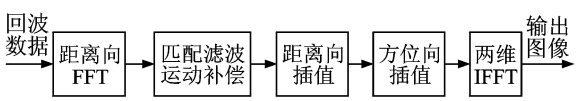


图 3 PFA 算法处理流程

Fig. 3 Flow chart of PFA

1.2 两维自聚焦

文献[9]中分析了 PFA 处理后残留两维相位误差的解析结构，得到了两维相位误差与方位一维相位误差的解析映射关系，即

$$\varphi_{2D} = \frac{K_y}{K_{yc}}\varphi_{1D}\left(\frac{K_{yc}}{K_y}K_x\right) \quad (5)$$

式中：φ_{2D} 表示两维相位误差；φ_{1D} 表示方位向一维相位误差；$K_{yc} = K_y\left(\frac{N_r}{2}\right)$，$N_r$ 为距离向采样点数。

根据这一关系，要估计两维相位误差，实际上只需要直接估计方位一维相位误差即可，两维相位

误差可由先前估计得到的方位相位误差通过式(5)直接计算得到。因此得到这种基于降维处理的两维自聚焦处理流程如图 4 所示。

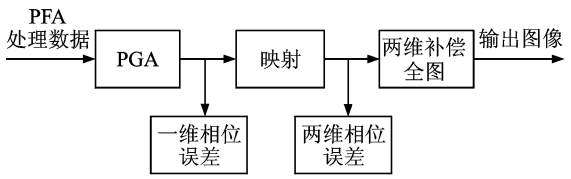


图 4　两维自聚焦算法流程

Fig. 4　Flow chart of 2-D autofocus algorithm

方位相位误差的估计采用经典的相位梯度自聚焦算法(Phase gradient autofocus，PGA)。PGA 算法是一种稳健的高分辨率 SAR 相位校正方法，在 SAR 领域得到广泛应用[10]。其处理主要包括 4 个步骤：中心移位、加窗处理、相位误差估计和迭代校正。PGA 算法流程如图 5 所示。

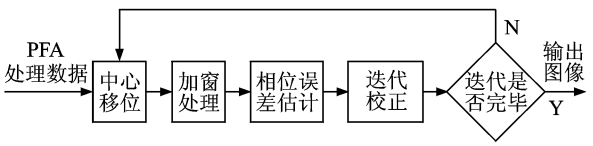


图 5　PGA 处理流程

Fig. 5　Flow chart of PGA

下面详细阐述 PGA 算法每个步骤的具体操作。

(1)中心移位：找到每个距离单元中模值最大的采样点，并将该点循环移位至该距离单元中心，以消除多普勒中心频率偏移。

(2)加窗处理：对经过循环移位的每个距离单元在图像域中心加窗，图像域加窗相当于相位域做滤波处理，可以提高信噪比从而提高相位估计精度。

(3)相位误差估计：将加窗处理过后的数据矩阵按方位向做 IFFT 得到 $g_n(m)$(相位域)，根据式(6)计算每个方位向的相位误差梯度

$$r_n(m)=g_n(m-1)\cdot[g_n(m)]^*$$
$$m=1,\cdots,N_a-1;n=1,\cdots,N_r \tag{6}$$

式中：$r_n(0)=1(n=1,\cdots,N_r)$；$*$ 为复数共轭；N_a 和 N_r 分别为方位向和距离向的采样点数。

$$\varphi(m)=\sum_{n=1}^{N_r}\arg[r_n(m)] \tag{7}$$

$$\varphi'(m)=\varphi(1)+,\cdots,+\varphi(m)$$
$$m=1,\cdots,N_a-1 \tag{8}$$

再采用式(7,8)(其中 arg[·]为取相位)，估计出一维相位误差 $\varphi'(m)$。

(4)迭代校正：将原始矩阵按方位向做 IFFT 转换到相位域，用估计出来的一维相位误差进行补偿，再做 FFT 回到图像域，为下一次迭代做准备。

一般情况下 PGA 对步骤(1～4)进行 4～6 次迭代就可以得到较理想的方位向聚焦效果。

2　成像处理流程的 GPU 实现

2.1　GPU 架构介绍

GPU 最大的特点是采用了单指令多线程(Single instruction multiple threads，SIMT)组织模式管理硬件中同时存在的线程，实现对线程数据的并行处理。运行在 GPU 上的 CUDA 并行计算函数称为内核函数(kernel)。一个 kernel 并不是一个完整的程序，而是整个 CUDA 程序中的一个可以被并行执行的步骤[11]。

如图 6 所示，kernel 是以线程格(grid)进行组织管理和分配的，目前一个 kernel 中只能包含一个 grid；每个 grid 中包含有若干个线程块(block)；每个 block 又是由若干个单线程(thread)组成。这样的三级结构，又与 GPU 本身的硬件计算单元流多处理器(Streaming multiprocessor，SM)和流处理器(Streaming processor，SP)相互对应，其中由 SM 来管理 thread 并行执行。每个 SM 拥有 32 个 SP(CUDA 2.0 及以上版本)。一个 block 会被分配到一个 SM 中(SM 中并行活动的 block 上限为 8 个)执行；而 block 中的 thread 则会被分配到 SP 中执行。

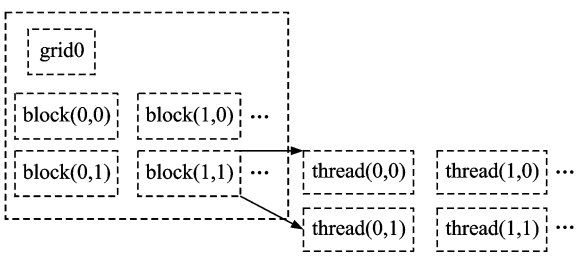


图 6　CUDA 三级线程架构

Fig. 6　CUDA three-level structure

2.2　并行化分析

GPU 数据处理的策略一方面是将对整个成像矩阵数据量的每一步操作尽可能对应到不同的 kernel 中完成。根据 GPU 的三级结构，其中 block 与二维成像矩阵相对应，所以 block 中的 thread 也采用二维分布，与二维成像矩阵中的每个采样点相对应。此时将矩阵中每个采样点利用

thread 的两维寻址对应分配其中，每个 thread 中的采样点都能独立完成运动补偿和插值等算法操作，利用并行化处理的 kernel 对每个成像处理步骤加速；另一方面在 CPU 端仅仅完成涉及一个距离向或方位向数据量的计算操作（如斜距计算）。因为处理量较小的操作，如若占用 GPU 的资源，就会闲置大量的 thread，产生长的等待延迟，降低 GPU 的利用效率。

这里需要注意的是，GPU 中所谓的“并行”执行并不严谨，实际情况是“并发”执行的。在理想条件下，GPU 并行运算所能获得的加速比等同于并行度。但是由于 GPU 本身资源的限制以及其调度分配等原因，在实际情况下加速比要低于并行度。尽管如此，利用 GPU 对雷达数据进行处理仍然能获得很高的加速比。

2.2.1 PFA 的并行化方案

PFA 的主要处理步骤有 FFT，IFFT，匹配滤波和运动补偿，插值，转置（外部雷达数据从 CPU 内存拷贝至 GPU 显存采用按距离向顺序存储。在距离向插值完成到方位向插值之前，需要将矩阵进行转置，变换为按方位向顺序存储）。FFT 和 IFFT 调用 CUDA 本身自带的 CUFFT[12] 库函数实现；插值采用的是 sinc 插值（根据实测数据精度要求可以选择 8 点或 4 点），利用传统的二分法能够在每个 thread 中快速查找所需插值点。因此可以得到如图 7 所示 PFA 对应的 CPU+GPU 处理流程。

在 kernel 并行化处理的基础上，本文参考了文献[5]中流水线（stream）的概念对数据处理进一步加速。流水线的概念现在非常普遍，在主流的数字信号处理器上都采用了多级流水线的设计方案。流水线是一种异步并行机制，该机制可以使得 kernel 上的数据处理和 CPU 与 GPU 之间的数据传输能够并行执行，流处理实际是通过数据在 CPU 和 GPU 相互拷贝的时间掩盖实际数据处理的时间。创建若干个流，将雷达数据平均分配到每个流上。如图 8 所示，流处理的过程简单总结就是“CPU→GPU 的数据拷贝”、“数据处理”、“GPU→CPU 的数据拷贝”这 3 个部分。

在实际处理过程中，无法像图 8 中所示，数据处理的时间段完全被数据拷贝的时间段掩盖。一方面数据拷贝的时间相对于数据处理时间要短很多；另一方面，基于单片 GPU 的流数据处理部分本身就要占用 GPU 大部分的资源，基本都要等待上一个流（除第一个流）的数据处理部分完成后才能占据 GPU 的资源进行处理，所以在数据拷贝和数据处理之间（除第一个流），每个流会出现延迟等待时间，这是所不希望看到的。在今后可以将单片 GPU 处理扩展到多片处理时可以很大程度上缩减

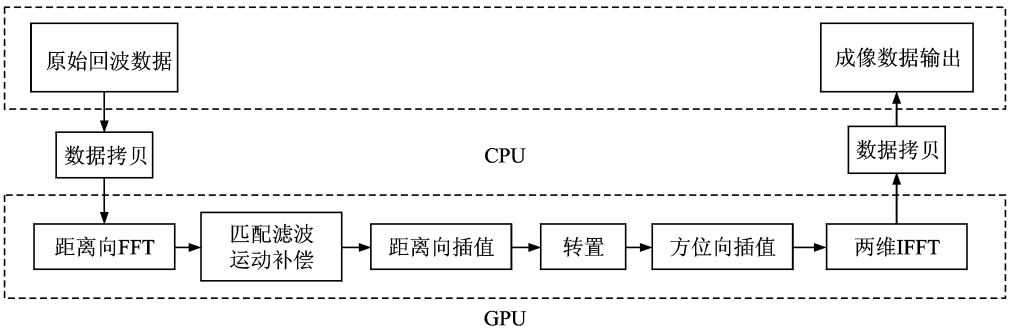


图 7 基于 CPU+GPU 的 PFA 实现流程

Fig. 7 Flow chart of PFA based on CPU+GPU

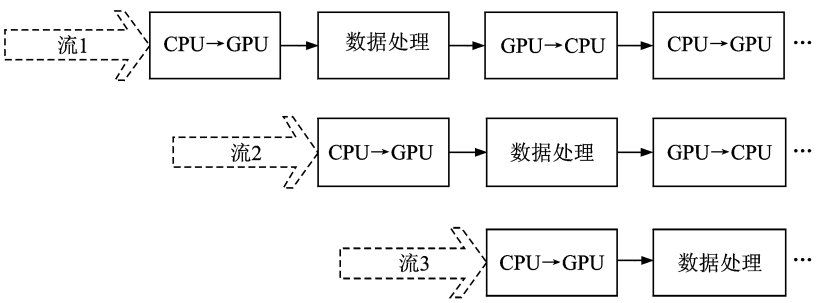


图 8 流异步处理过程流程图

Fig. 8 Flow chart of stream asynchronous processing

甚至消除这类等待时间。

2.2.2　两维自聚焦的并行化方案

相对于 PFA 算法，两维自聚焦算法部分复杂度较高，并行化的难度和局限性也较大。可以从以下 4 个方面入手。

(1)迭代矩阵

PGA 原理是基于强特显点散焦情况估计相位误差，所以只需要原始图像域中能量最大的一部分距离单元(一般选取前 5%)。后将这一部分距离单元恢复构成迭代矩阵，可以大大减少迭代时处理的数据量。此时需要在迭代开始前增加寻找最大模值的步骤，用来找出前 5%的距离单元而后恢复出迭代矩阵；在进入第一次迭代后，跳过寻找最大值的步骤，直接移位和加窗(因为在迭代外的寻找最大模值步骤中已经记录下迭代矩阵第一次需要移位的最大模值点及其下标)。

(2)寻找最大模值

对于 N 个数据，找出最大值通常需要 $N-1$ 次比较操作。这样的串行比较思想并不适用于并行加速。这里采用归约(reduction)的算法思想来找出每个距离单元中的最大模值(CUDA 的 SDK 中含有 reduction 的说明文档和对应程序)，可以大幅度降低时间复杂度，提高效率。

(3)相位误差估计

根据 PGA 算法介绍中式(7,8)可知，处理步骤为：取相位、求和、累加以及最后需要将相位误差(浮点数据)转换为复数据。可以将以上步骤精简为求和，累乘，即替换为式(9,10)，也可以达到相同的处理效果，处理步骤简单且高效。其中累乘是对单个方位向数据量进行的运算，所以放到 CPU 端处理会更加高效。

$$\varphi(m)=\sum_{n=1}^{N_r} r_n(m) \tag{9}$$

$$\varphi'(m)=\varphi(1)\cdot,\cdots,\cdot\varphi(m)$$
$$m=1,\cdots,N_a-1 \tag{10}$$

(4)总补偿相位

整个迭代的目的在于求出一维总补偿相位，即在最后一次迭代校正之前就已经求得该相位，所以最后一次的迭代校正是不需要的，可以直接跳到迭代后的下一步操作。

由此可以得到如图 9 所示两维自聚焦算法对应的 CPU+GPU 处理流程。

3　试验结果及性能分析

为了验证成像算法聚焦精度及 GPU 加速性能，针对某超高分辨率机载 SAR 实测数据进行了处理。该雷达工作在聚束模式，其主要参数如表 1 所示。

表 1　雷达主要参数

Tab. 1　Main parameters of radar

参数	参数值
波段	X
距离分辨率/m	0.1(信号带宽 1.5 GHz)
方位分辨率/m	0.16
作用距离/km	81

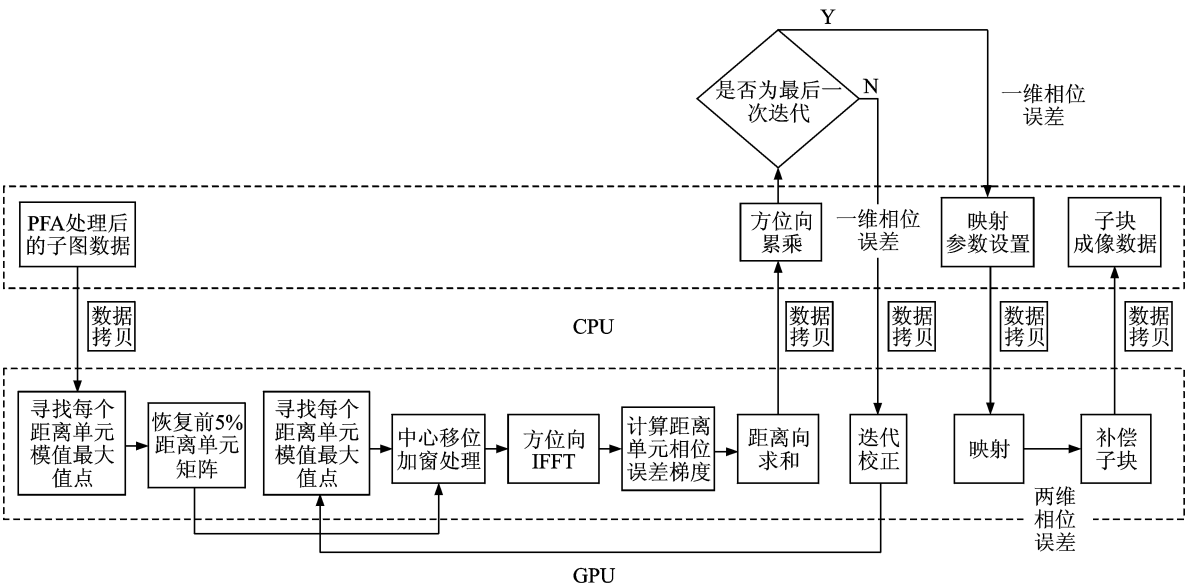


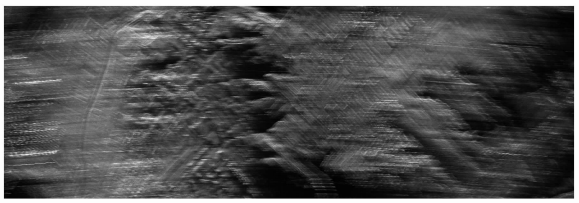
图 9　基于 CPU+GPU 的两维自聚焦算法实现流程

Fig. 9　Flow chart of 2-D autofocus algorithm based on CPU+GPU

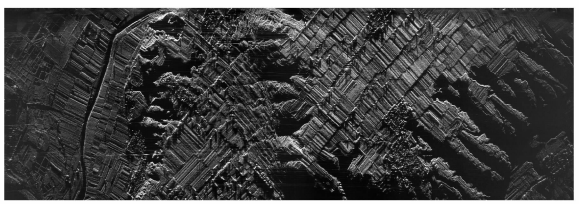
成像处理时，实际处理脉冲数为65 536(64 K)，对应孔径长度约为7 068 m，在如此长的孔径时间内，雷达载机运动存在较大机动。为了验证算法鲁棒性，在成像处理时没有利用惯导数据，而是假设雷达平台以平均速度做匀速直线运动。考虑到内存限制，实际处理时，距离向只截取了16 K个采样点，因此处理的整个数据大小为64 K×16 K。

3.1 成像处理效果

图10(a)为直接PFA处理结果，可以看到，图像存在非常严重的散焦，而且散焦是二维的，因此必须采取两维自聚焦才能提供精确的重聚焦。图10(b)为通过分块两维自聚焦后的结果，可以看到，整个图像均聚焦良好。



(a) PFA处理后图像



(b) 分块两维自聚焦处理后图像

图10 成像处理效果

Fig. 10 Image after PFA processing and image after 2-D autofocus processing

3.2 成像处理效率

本文采用的CPU处理型号为Intel Xeon E5-1650[13](3.2 GHz主频，32 GB内存，6核心)，是现在主流的工作站CPU型号。GPU处理型号是Tesla C2075[14](计算能力2.0，1.15 GHz时钟频率，6 GB GDDR5显存，448个CUDA计算核心(cores)，1.03Tflops单精度峰值性能)。整个并行化程序搭建在CUDA5.5版本，所用代码编译器为VS2010。

表2给出了GPU和CPU处理时间对比(表中的处理时间不包括数据在CPU硬盘和内存之间的拷贝时间)。

从表2中可以看出，因为PGA算法本身的复杂度及其并行化的局限性导致加速比远远低于PFA所取得的加速比。处理数据方位向采样点数为64 K，方位向采样频率为1 333.3 Hz，可以得到数据采集时间大约为49.15 s，而实际的GPU处理时间为96.73 s，获得了每秒约84兆采样点的实时处理速度(只是针对距离截取的16 K数据)，相比于CPU的总处理时间，获得了理想的加速比和实时处理效率。

表2 CPU与GPU数据处理时间对比

Tab. 2 Comparison of processing time between CPU and GPU

处理过程	CPU/s	GPU/s	加速比
PFA	10 335.61	50.47	204.78
2-D autofocus(block)	876.12	46.26	18.94
PFA+2-D autofocus (block)	11 211.73	96.73	115.91

4 结束语

本文主要研究了超高分辨率SAR成像的处理方案，详细阐述了该方案的处理流程和GPU端对应改进的并行化处理步骤。通过实测数据验证了该处理方案能够获得良好的聚焦精度和成像效果，实现了超高分辨率成像。最后通过硬件平台GPU完成了数据的实时成像，今后可以利用更好的GPU平台获得更高的实时处理效率。该成像处理方案不依赖于惯导，具有很好的鲁棒性，在超高分辨率机载SAR成像处理中具有很好的应用前景。

参考文献：

[1] Tsunoda S I, Pace F, Stence J, et al. Lynx: A high-resolution synthetic aperture radar[C]//SPIE Radar Sensor Technology. Orlando: International Society for Optics and Photonics, 1999:20-27.

[2] Ender J H G, Berens P, Brenner A R, et al. Multi-channel SAR/MTI system development at FGAN: from AER to PAMIR[C]//IGARSS′02. Toronto: IEEE Press, 2002:1697-1701.

[3] Burns B L, Cordaro J T. A SAR image-formation algorithm that compensates for the spatially-variant effects of antenna motion[C]//SPIE′s International Symposium on Optical Engineering and Photonics in Aerospace Sensing. Orlando: International Society for Optics and Photonics, 1994:14-24.

[4] 周芳，唐禹，张佳佳，等. 机载高分辨聚束式SAR实时成像处理系统的FPGA实现[J]. 电子与信息学报，2011,33(5):1248-1252.
Zhou Fang, Tang Yu, Zhang Jiajia, et al. Real-time image formation for airborne high resolution spotlight SAR based on FPGA[J]. Journal of Electronics & Information Technology, 2011,33(5):1248-1252.

[5] 孟大地，胡玉新，石涛，等. 基于 NVIDIA GPU 的机载 SAR 实时成像处理算法 CUDA 设计与实现[J]. 雷达学报，2013，2(4)：481-491.
Meng Dadi, Hu Yuxin, Shi Tao, et al. Airborne SAR real-time imaging algorithm design and implementation with CUDA on NVIDIA GPU[J]. Journal of Radars, 2013, 2(4): 481-491.

[6] Cafforio C, Prati C, Rocca F. SAR data focusing using seismic migration techniques[J]. IEEE Trans on Aerosp Electron Syst, 1991, 27(2): 194-207.

[7] Gallon A, Impagnatiello F. Motion compensation in chirp scaling SAR processing using phase gradient autofocusing[C]//IGARSS'98. Seattle: IEEE Press, 1998: 633-635.

[8] Zhu Daiyin. SAR signal based motion compensation through combining PGA and 2-D map drift[C]//APSAR 2009. Xi'an, China: IEEE Press, 2009: 435-438.

[9] 毛新华，曹海洋，朱岱寅，等. 基于先验知识的 SAR 两维自聚焦算法[J]. 电子学报，2013，41(6)：1041-1047.
Mao Xinhua, Cao Haiyang, Zhu Daiyin, et al. Prior knowledge aided two-dimensional autofocus approach for synthetic aperture radar[J]. Acta Electronica Sinica, 2013, 41(6): 1041-1047.

[10] 武昕伟，朱兆达，朱岱寅. 相位梯度自聚焦算法在条带模式 SAR 中的应用[J]. 数据采集与处理，2002，17(4)：437-440.
Wu Xinwei, Zhu Zhaoda, Zhu Daiyin. Phase gradient autofocus algorithm for stripmap SAR autofocus[J]. Journal of Data Acquisition & Processing, 2002, 17(4): 437-440.

[11] 张舒，褚艳利. GPU 高性能运算之 CUDA[M]. 北京：中国水利水电出版社，2009.
Zhang Shu, Chu Yanli. GPU high-performance computing: CUDA[M]. Beijing: China Water Power Press, 2009.

[12] NVIDIA. Cufft library user's guide[EB/OL]. http://docs.nvidia.com/cuda/pdf/CUFFT_Library.pdf, 2014-08/2014-12-01.

[13] INTEL. Intel xeon processor E5-1650[EB/OL]. http://ark.intel.com/products/64601/Intel-Xeon-Processor-E5-1650, 2012-05/2014-12-01.

[14] NVIDIA. Tesla C2075 computing processor board [EB/OL]. http://www.nvidia.cn/docs/IO/43395/BD-05880-001_v02.pdf, 2011-09/2014-12-01.