一种基于 GPU 的 SAR 高效成像处理算法

孟大地* 胡玉新 丁赤飚
(中国科学院电子学研究所 北京 100190)
(中国科学院空间信息处理与应用系统技术重点实验室 北京 100190)

摘 要: 合成孔径雷达(SAR)成像处理是一项需要进行大量计算的处理任务。图形处理器(GPU)具有数十倍于 CPU 的浮点计算能力以及传输带宽,而 CUDA 技术的发展使得 GPU 能够方便地进行通用计算。该文提出了一种在 GPU 上进行 SAR 成像的高效方法。与一般 GPU 处理方法相比,该方法使得处理过程中的 CPU-GPU 往返数据传输由 4 次减少到 1 次,而且同时利用了工作站上的 CPU 与 GPU 计算资源。实验结果表明,该方法能够带来相对一般 GPU 处理方法 2.3 倍的处理效率提升,从而验证了该方法的有效性。

关键词: SAR;CUDA;GPU;SAR 成像处理
中图分类号: TN957.52 **文献标识码:** A **文章编号:** 2095-283X(2013)02-0210-08
DOI: 10.3724/SP.J.1300.2013.20098

Efficient Algorithm for Processing SAR Data Based on GPU

Meng Da-di Hu Yu-xin Ding Chi-biao
(*Institute of Electronics, Chinese Academy of Sciences*, *Beijing* 100190, *China*)
(*Key Laboratory of Technology in Geospatial Information Processing and Application System*, *Chinese Academy of Sciences, Beijing* 100190, *China*)

Abstract: Data processing is time-consuming in the field of Synthetic Aperture Radar (SAR). Graphics Processing Units (GPUs) have tremendous float-point computational ability and a very high memory bandwidth, and the developing Compute Unified Device Architecture (CUDA) technology has enabled the application of GPUs to general-purpose parallel computing. A new method for processing SAR data using GPUs is presented in this paper. Compared with the nominal GPU-based SAR processing method, the number of data transfers between the CPUs and a GPU is reduced from 4 to 1, and the CPUs are exploited to cooperate with the GPU synchronously. By using the proposed method, we can speed up the data processing by 2.3 times, which is verified by the testing with simulated SAR data.

Key words: Synthetic Aperture Radar (SAR); Compute Unified Device Architecture (CUDA); Graphics Processing Unit (GPU); SAR data processing

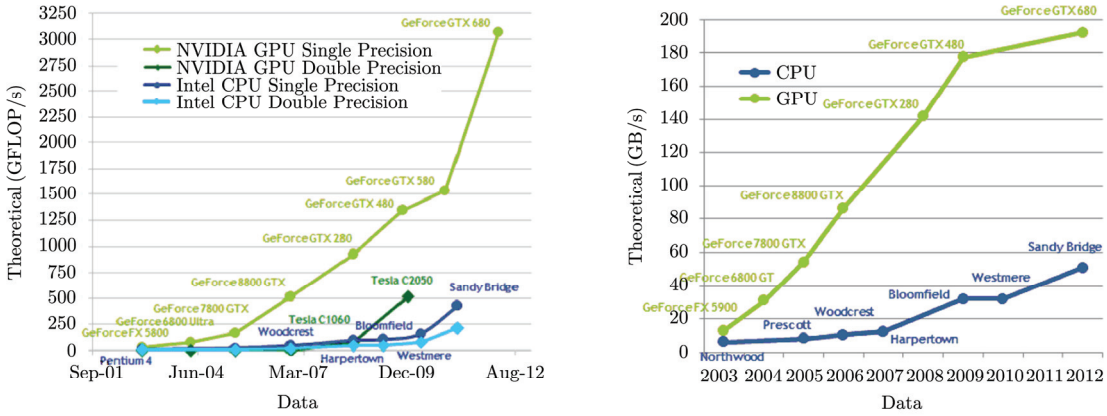
1 引言

合成孔径雷达(Synthetic Aperture Radar, SAR)是一种全天时、全天候的微波遥感对地观测手段。SAR 图像中包含丰富的地表特征信息,并具有穿透能力,因此在国民经济以及军事领域,具有光学遥感不可替代的重要作用。但与光学遥感相比,SAR 图像需要通过对 SAR 原始数据进行相干数据处理得到。常用的处理算法有以ω-κ算法[1,2]为代表的频域批处理算法(距离多普勒算法[1]以及 Chirp Scaling 算法[1]均属此类),以及反投影时域处理算法[3]。

SAR 图像的方位向分辨能力有赖于对场景的长时间观测期间的脉冲相干积累,SAR 数据在两个垂直维度上也存在耦合,因此在实际应用中,SAR 数据量非常巨大;各 SAR 应用领域科学技术的飞速发展,对 SAR 分辨率、多波段、多极化等方面提出了更高的要求,这就使得 SAR 数据量成倍增加。因此,高效的 SAR 数据处理手段成为 SAR 技术发展的一项关键问题。

由于 3D 图形领域的拉动作用,图形处理器(Graphic Processing Unit, GPU)技术在近几年来发展非常迅速。由图 1 可见,在处理能力以及存储器带宽上,近七年间 GPU 所取得的进展与 CPU 相比超越约一个量级[4]。GPU 在运算以及传输速度上的飞速发展,在一定程度上应归因于 GPU 主要专

2012-12-17 收到,2013-04-07 改回;2013-04-19 网络优先出版
国家大科学工程航空遥感系统地面数据处理与管理分系统项目资助课题
*通信作者: 孟大地 mengdadi@hotmail.com



(a) 浮点处理能力对比
(a) Floating-point operations

(b) 存储器带宽对比
(b) Memory bandwidth

图 1 浮点处理能力及存储器带宽近年发展对比

Fig. 1　Floating-point operations per second and memory bandwidth for the CPU and GPU

注于数据并行性任务的处理，而较少考虑了逻辑控制、条件判断等 CPU 中不可或缺的重要功能。2006 年 11 月，英伟达(NVIDIA)公司针对旗下主流型号 GPU 产品推出了一种通用并行计算架构 —— CUDA(Compute Unified Device Architecture)，以及相关的并行编程模型和指令集，从而大大推动了 GPU 在高性能计算领域的广泛应用。借助 CUDA，在传统 CPU 上运行的需要大量计算的代码可以方便地移植到 GPU 上执行，而将代码的逻辑控制以及条件判断部分仍然由 CPU 执行。这种新的程序开发模式使得传统上较为耗时的处理任务得到了数十倍甚至上百倍的速度提升[4]。

利用 CUDA 这种高效的 GPU 编程模式，SAR 数据处理软件的速度有望得到大幅度提升。国内外已有少量该方面的研究成果[5−11]。文献[5−7]在 GPU 上实现了反投影处理算法，使得原本极其耗时的处理过程可以在可容忍的较短时间内完成；文献[8−11]介绍了距离-多普勒算法在 GPU 上的实现方法。

当前支持 CUDA 编程的主流 GPU 产品的内存一般不大于 4 GB[4]，而在实际 SAR 数据处理过程中，所需处理的 1 景 SAR 数据往往超出单个 GPU 存储能力。因此，在将现有 SAR 成像处理算法在 GPU 上实现时，除了要考虑原始算法中各大运算量代码在 GPU 上的合理实现之外，还需要考虑如何高效地在 CPU 内存与 GPU 内存之间进行数据交互。

对于这种数据量超出 GPU 存储能力的情况，文献[8,9]通过减少处理脉冲数的方法强制减少 1 景 SAR 数据的数据量。由于处理所得每景 SAR 图像的方位向两端孔径不全，因此不能达到全分辨率，在实际应用中需要截去，并且截去量与处理脉冲数无关，因此这种通过缩减方位景宽的方法将导致截去比例增加，降低处理效率。

当前 GPU 设备与主机之间主要通过 PCI-E 2.0×16 方式进行连接[12]，根据在 Tesla C1060[13]上的实际测试结果，主机内存与 GPU 显存之间进行 1 次 2 GB 数据的往返传输需要消耗时间约为 1.1 s，而对 2 GB float 型存储 SAR 数据利用单卡 GPU 1 次成像(采用ω-κ成像处理算法，包括运动补偿处理)时间约为 2.5 s。由此可见，在成像处理过程中若有多次数据在主机内存与 GPU 显存之间的往返传输，将会带来效率的成倍下降。因此，需要尽可能减少主机内存与 GPU 显存之间的数据传输次数，从而充分发挥 GPU 的高速浮点运算能力，提高运算速度。

本文对机载ω-κ成像处理算法在 GPU 上的实现进行了深入研究，提出了一种新的ω-κ算法在 GPU 上的实现方法 —— 方位时域分割法。该方法只需 1 次 SAR 原始数据导入 GPU 显存以及 1 次 float 型 SAR 数据导出 GPU 显存操作，从而大大降低了 SAR 数据在显存传输上所消耗的时间。

另外，该方法将部分任务交由 GPU 处理，而数据导出后的剩余少量任务仍由 CPU 处理，这两部分任务可并行执行。在配置 4 颗 Intel Xeon X5550[14]、16 G 内存、1 部 Tesla C1060[13] GPU 的工作站上的实验表明，在合理分配 CPU 核数与 GPU 设备数比例时，该方法在 GPU 端任务所花时间与 CPU 端相当。这时，1 块 SAR 数据的处理时间为 GPU 处理时间与 CPU 处理时间的最大值，从而在避免数据冗余传输的同时，充分利用了 1 台主机上的所有计算资源。

本文第 2 节简要介绍了ω-κ算法的基本步骤；第 3 节介绍了ω-κ算法的常规 GPU 实现方法；第 4 节

提出了方位时域分割法，并给出了处理步骤和误差分析；第 5 节利用点目标仿真 SAR 数据的处理结果验证了算法的有效性和高效性；第 6 节对本文内容进行了总结。

2 ω-κ算法介绍

ω-κ算法是一种被广泛采用的机载 SAR 成像处理算法[1, 2]。与其他常用算法(如距离-多普勒算法[1]、Chirp Scaling 算法[1])相比，由于其推导过程采用与 SAR 成像原理一致的双曲距离历程模型，并且除了载机速度不能沿距离向变化(在机载情况下该假设完全成立[1])以及插值误差之外，未采取任何近似，因此ω-κ算法能够适应各个波段的 SAR 数据处理，并能够适应大方位波束角、大斜视角等情况。

另外，在实际 SAR 数据获取过程中，由于载机不能严格保持匀速直线运动状态，在成像处理过程中还需要加入运动补偿处理环节(通过对距离压缩后信号进行距离向重采样以及相位补偿实现)，以抵消载机运动的非理想性。

ω-κ算法的处理流程如图 2 所示。该算法处理步骤如下：

(1) 首先对原始数据进行距离向压缩(包括两次距离向 FFT 以及距离向参考函数相乘)以及运动补偿(包括插值以及相位生成、相位相乘)，并通过距离向 FFT 转至距离向频域；

(2) 通过方位向 FFT(需要两次额外转置)转至 2 维频域；

(3) 在 2 维频域进行距离迁移校正、2 次距离压缩、方位压缩(通过距离向参考函数生成、相位相乘以及 stolt 插值实现)；

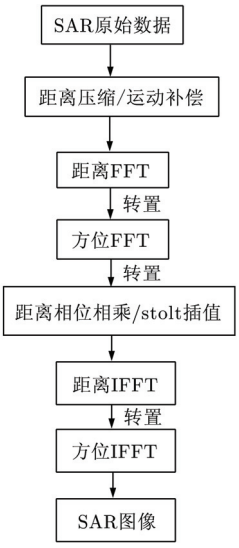


图 2 常规基于 CPU 的ω-κ算法处理流程图

Fig. 2 Diagram of normal ω-κ on CPU

(4) 通过距离向 IFFT 转至距离向时域；

(5) 最后通过方位向 IFFT(需要 1 次额外转置)得到时域 SAR 图像。

3 ω-κ算法的常规 GPU 实现方法

利用 CUDA 编程模型将ω-κ算法移至 GPU 实现时，首要考虑的是将 SAR 数据传入 GPU 显存。当 GPU 显存容量足以容纳所处理 SAR 数据以及处理所需额外内存时，可以将所有ω-κ算法处理流程交由 GPU 处理，最后将处理结果 SAR 图像传入主机内存并进行存储。

当对 SAR 系统的分辨率、测绘带提出更高要求时，GPU 显存存储容量往往不能满足 1 景 SAR 数据的存储。这时需要在处理过程中多次将处理中间结果分块导入 GPU 显存进行处理，之后再将处理结果导出 GPU 显存，在进行转置之后，再进行后续处理。

常规 GPU 处理步骤如下：

(1) 对 SAR 原始数据在方位向进行分块，各块分别传入 GPU 显存进行距离压缩、运动补偿以及距离向 FFT，并将结果传入主机内存；

(2) 对 SAR 数据在距离向进行分块，各块分别传入 GPU 显存进行方位向 FFT，并将结果传入主机内存；

(3) 对 SAR 数据在方位向进行分块，各块分别传入 GPU 显存进行距离向参考相位相乘、stolt 插值以及距离向 IFFT，并将结果传入主机内存；

(4) 对 SAR 数据在距离向进行分块，各块分别传入 GPU 显存进行方位向 FFT，并将结果传入主机内存，拼接之后得到时域 SAR 图像。

按照这种思路，除了在 GPU 端进行应有的ω-κ算法处理操作以及 SAR 原始数据传入 GPU 显存、SAR 图像数据传出 GPU 显存之外，还需进行 3 次额外的 GPU 显存与主机内存之间的往返数据传输。

另外，每次将数据传入主机内存之前需要将数据进行转置处理(共需 3 次)，以适应后续步骤的处理需求。

这种基于 GPU 的常规ω-κ算法处理流程如图 3 所示。可见，这种 GPU 端的ω-κ成像处理方法在原理上与常规ω-κ算法完全一致，而将所有主要的处理步骤分别移至 GPU 端实现，但增加了 3 次额外的主机内存与 GPU 显存之间的往返数据传输。

4 方位时域分割法

根据 SAR 成像处理原理[1]可知，对ω-κ成像处理所得到的 SAR 图像进行方位向逆压缩可以得到

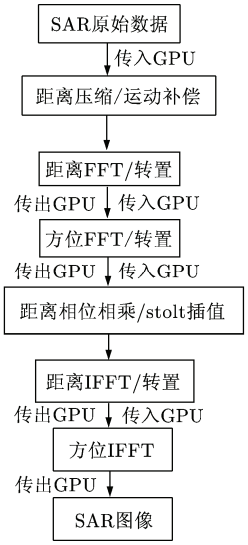


图 3 常规基于 GPU 的ω-κ算法处理流程图

Fig. 3　Diagram of normal ω-κ on GPU

距离迁移校正后、方位压缩前的 SAR 数据，这时场景中各目标的回波信号都被校正到 1 个距离单元，沿方位向排列。如果在成像处理开始之前对 SAR 原始数据在方位向分块，每块单独进行ω-κ成像处理以及其他辅助处理，得到方位压缩前数据，则可以将各块处理得到的数据在方位向进行拼接，拼接得到的 SAR 数据即近似相当于对所有 SAR 原始数据统一进行成像处理所得到的方位压缩前数据(该等效的近似性将在 5.1 节进行分析)。再对该数据进行方位压缩处理，即得到所有 SAR 原始数据所对应的 SAR 图像。

基于这种成像处理方式，本文提出了一种ω-κ算法在 GPU 上新的实现方法 —— 方位时域分割法。该方法将 SAR 原始数据回波在方位向分为若干块(每块数据量小于 GPU 显存)，每块交由 GPU 一次性进行方位压缩前的所有处理步骤，并将处理结果导入主机内存并在方位向拼接，再由主机端进行方位压缩，得到最终的 SAR 图像。

方位时域分割法的处理步骤如图 4 所示，处理步骤如下：

(1) 将 SAR 原始数据在方位向进行分块(每块数据量小于显存容量)之后，各块分别传入 GPU 并进行常规ω-κ算法处理步骤(无需分块)；

(2) 在方位向 IFFT 生成 SAR 图像之前通过方位向参考相位相乘得到方位压缩前信号；

(3) 将各块处理结果传入主机内存，并拼合为全景 SAR 数据的方位压缩前数据；

(4) 通过方位向 FFT、方位向参考相位相乘、方位向 IFFT 得到最终 SAR 图像。

由此可见，与常规方法相比，方位时域分割法在避免了 3 次往返数据传输的同时，在 GPU 端及 CPU 端各增加了一次参考相位生成及相乘，在 CPU 端增加了两次 FFT。

5　误差分析及实验结果

5.1 方位时域分割法误差的实验分析

从原理上来讲，在时带积大于 100 时，驻定相位原理的近似性对 SAR 成像的影响可忽略不计[1]。对于内存为 4 G 的 GPU 来说，所能容纳的脉冲数一般能满足该要求。这时，方位时域分割法对各块处理所得的方位压缩前时域数据拼接后近似等同于用ω-κ算法直接对 SAR 原始数据进行处理所得方位压缩前时域数据。由于驻定相位原理近似性的理论分析较为困难，以下结合仿真对方位时域分割法的误差特性进行研究。

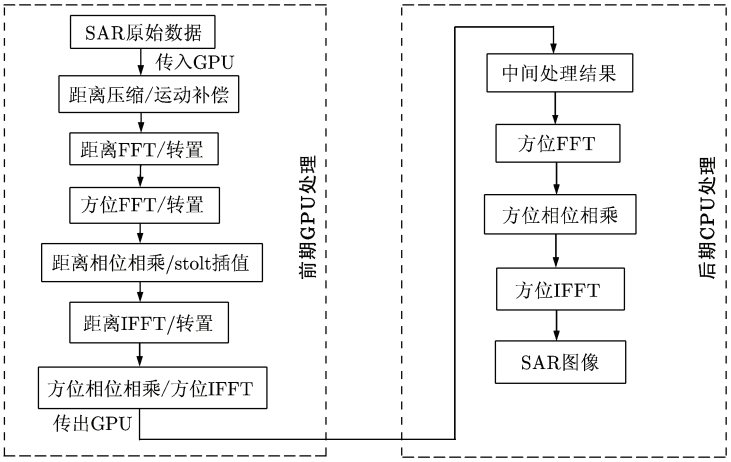


图 4 方位时域分割法处理流程图

Fig. 4　Diagram of the proposed method

对 1 块距离、方位向分别为 32768 的仿真 X 波段 SAR 数据(有符号 8 位整型存储，仿真中加入幅度约为 3 m 的运动误差)进行成像及运动补偿处理。仿真所用系统参数如表 1 所示。在该数据中，所设点目标回波的零多普勒位置位于中心距离单元的方位向中心，因此在利用方位时域分割法处理将仿真数据分为 4 块进行处理时，该点目标回波信号的前后两半将分别位于中间两个分块内。

表 1 仿真参数

Tab. 1 Simulation parameters

参数	数值
系统波长	0.03 m
脉冲带宽	375 MHz
脉冲采样率	500 MHz
近地点斜距	5000 m
载机速度	150 m/s
PRF	500 Hz
多普勒带宽	375 Hz

对表 1 仿真数据分别进行方位时域分割法处理以及基于 CPU 的ω-κ算法处理，点目标在相邻两个分块内的方位压缩前信号如图 5 所示。这时信号总时带积约为 930，两分块内的局部信号时带积约为 465。由于驻定相位原理的幅度特性在信号边缘有较大的震荡以及低通效应[1]，因此方位时域分割法的分块操作导致各块边缘位置处有少量的信号泄漏，如图 5(a)所示；各段信号的边缘泄漏在频域也相应造成了较大的频谱幅度震荡，如图 5(b)所示。图 5 中信号的相位特性以及方位压缩后的方位波形如图 6 所示，图 6(b)波形的各项压缩指标如表 2 所示。由该仿真结果可见，方位时域分割法的分块操作所引起的相位误差以及幅度调制导致点目标的压缩波形有所变化。这主要体现在方位向峰值旁瓣比(Peak Sidelobe Ratio, PSLR)以及积分旁瓣比(Integrated Sidelobe Ratio, ISLR)的恶化上(约 0.1 dB)。在实际应用中，这样的 SAR 图像质量恶化程度通常是可以容忍的，而且可以通过加窗的手段进行抑制。

另外，由于运动误差的方位空变性[15,16]，两个处理结果中方位向 PSLR 与理想值[1](–13.2 dB)相比恶化约 0.1 dB。

5.2 方位时域分割法性能分析

仿真所用工作站上配置了 4 颗 4 核 Intel Xeon X5550 CPU[14]以及 1 台单卡 NVIDIA Tesla C1060 GPU 设备[13]。基于 CPU 的ω-κ算法中使用了 Intel MKL 数学运算库并通过 Intel 编译器进行编译，方位时域分割法使用 CUDA 3.2 编译，其中主机代码调用 gcc 编译。在该工作站上，分别利用基于 CPU 的ω-κ算法、常规 GPU 方法、单 CPU 方位时域分

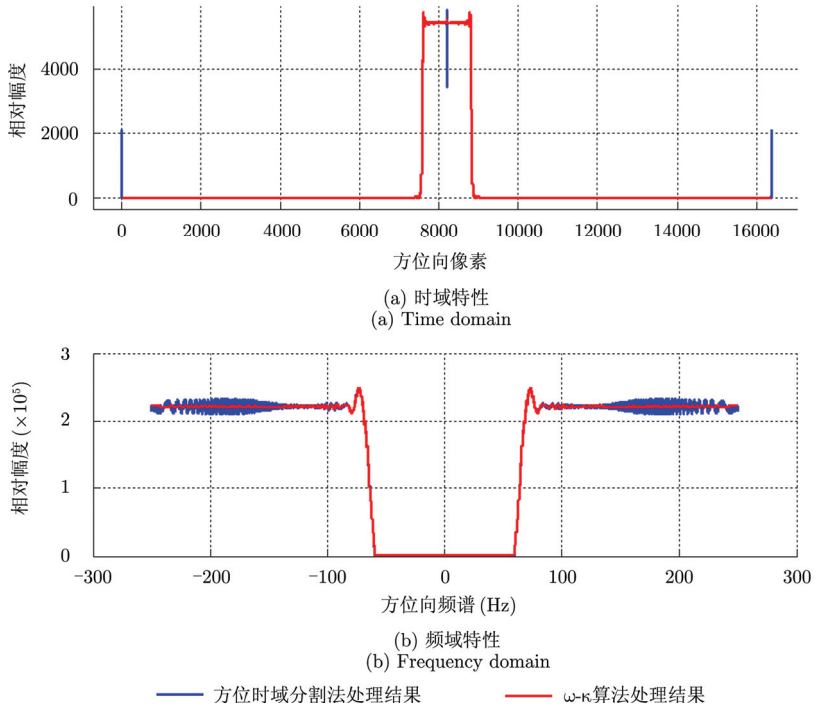
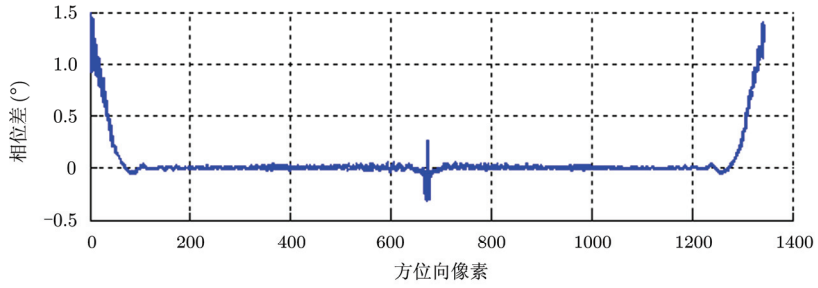
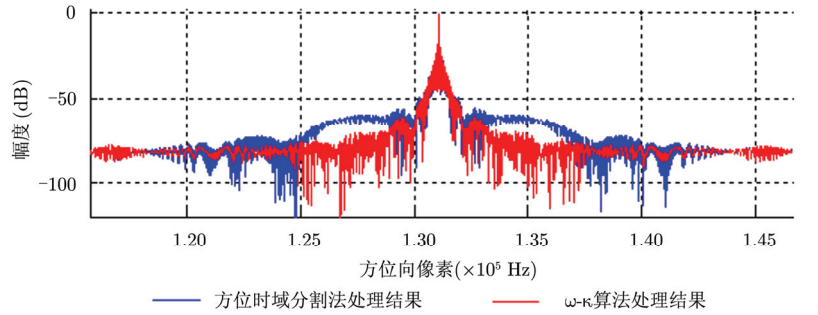


图 5 方位压缩前信号图

Fig. 5 Azimuth character of signal before azimuth compression



(a) 方位压缩前两信号的相位差
(a) Phase difference before azimuth compression



(b) 压缩波形
(b) Profile after azimuth compression

图 6　方位时域分割法与ω-κ算法处理结果对比

Fig. 6　Result comparison of proposed method and normal ω-κ algorithm

表 2　点目标仿真结果

Tab. 2　Simulation results of point target simulation

指标	CPU 处理结果		GPU 处理结果	
	距离向	方位向	距离向	方位向
分辨率 (m)	0.3375	0.3375	0.3375	0.3375
ISLR (dB)	−9.7577	−9.9132	−9.7638	−9.8121
PSLR (dB)	−13.2630	−13.1494	−13.2727	−13.0233

割法、多 CPU 方位时域分割法 4 种方法对表 1 所示仿真数据进行成像及运动补偿处理(方位时域分割法中将 SAR 原始数据分为 4 块，每块 float 型数据量为 2 GB)。所花时间如表 3 所示(其中未包括 SAR 原始数据读入主机内存时间以及 SAR 图像文件输出时间)。

由仿真结果可见，在单核 CPU 配置情况下，与ω-κ算法的 CPU 实现相比，常规 GPU 实现方法效率提高将近 30 倍，但其中约 60%时间花在数据在主机内存与 GPU 显存之间的交互上，GPU 的高速计算性能未能得到充分发挥。方位时域分割法避免了数据在主机内存与GPU 显存之间的冗余交互，但需要较长的 CPU 处理时间。

在配置多颗 CPU 使得方位时域分割法的 CPU 处理时间与 GPU 处理时间相当时，可在 CPU 进行后期处理的同时让 GPU 进行下 1 块 SAR 数据的前

表 3 ω-κ算法 3 种实现方法处理效率对比

Tab. 3　Efficiency comparison of 3 approaches of ω-κ algorithm

处理方案	常规ω-κ算法处理时间 (s)	常规 GPU 处理时间 (s)		方位时域分割法处理时间 (s)	
		冗余数据传输	成像处理	GPU 处理时间	CPU 处理时间
单核 CPU	720.25	15.33	9.88	10.96	99.82
14 核 CPU	63.25	15.33	9.88	10.79	9.54

期处理，这时总处理时间为 GPU 处理时间与 CPU 处理时间的最大值(表 3 实验结果中为 10.79 s)。按照表 3 实验结果，在配置 14 核 CPU 的情况下(1 个 CPU 核心用于 GPU 控制)，相比基于 CPU 的ω-κ算法，常规 GPU 实现方法的效率提高约 2.5 倍，而方位时域分割法的效率提高约 5.8 倍。

由以上仿真分析可知，方位时域分割法由于避免了冗余的主机与显存之间的数据传输，并结合 CPU 资源进行并行处理，可以在约 10 s 内完成 8 GB SAR 数据的成像及运动补偿处理，相比常规 GPU 实现方法，性能提高约 2.3 倍。

6 结论

本文提出了一种ω-κ算法在GPU平台上的新的实现方法——方位时域分割法。与常规 GPU 实现方法相比，新方法避免了主机内存与 GPU 显存之间的多次冗余数据传输。实验结果表明，在常规 GPU 实现方法中，冗余数据传输占用了成像处理的大部分时间，而方位时域分割法只需要在成像初始阶段将 SAR 原始数据传入 GPU 显存，在 GPU 处理结束之后将中间结果传回主机内存，与常规 GPU 实现方法相比减少了 3 次主机内存与 GPU 显存之间的往返数据传输，成倍缩减了 GPU 处理时间。

另外，方位时域分割法将 SAR 成像及运动补偿处理任务划分为前期 GPU 计算部分和后期 CPU 计算部分，避免了常规 GPU 实现方法对 CPU 计算资源的浪费。按照这种任务分配方式，在进行批量 SAR 数据处理时，可在后期 CPU 计算的同时，GPU 对下一块 SAR 数据进行前期计算，提高了 SAR 数据处理效率。在实验所用工作站的性能条件下，利用 Tesla C1060 单卡 GPU 与 14 核 CPU 进行处理时，GPU 端处理时间与 CPU 端处理时间相当，可以较为充分地利用工作站的计算资源。

参 考 文 献

[1] Cumming I G and Wong F H. Digital Processing of Synthetic Aperture Radar Data: Algorithms and Implementation[M]. Boston: Artech House, 2005.

[2] Bamler R. A comparison of range-doppler and wavenumber domain SAR focusing algorithms[J]. *IEEE Transactions on Geoscience and Remote Sensing*, 1992, 30(4): 706–713.

[3] Lars M H U, Hans H, and Gunnar S. Synthetic-aperture radar processing using fast factorized back-projection[J]. *IEEE Transactions on Aerospace and Electronic Systems*, 2003, 39(3): 760–776.

[4] Nvidia. NVIDIA CUDA C programming guide [OL]. http://developer.download.nvidia.com/compute/cuda/3_2/toolkit/docs/CUDA_C_Programming_Guide.pdf.

[5] Fasih A R and Hartley T D R. GPU-accelerated synthetic aperture radar backprojection in CUDA[C]. IEEE International Radar Conference, Arlington, VA, USA, May 2010: 1408–1413.

[6] Blom M and Follo P. VHF SAR image formation implemented on a GPU[C]. International Geoscience and Remote Sensing Symposium, Seoul, Korea, July 2005: 3352–3356.

[7] Hartley T D R, Fasih A R, Berdanier C A, *et al.*. Investigating the use of GPU-accelerated nodes for SAR image formation[C]. Proceedings of the IEEE International Conference on Cluster Computing, New Orleans, LA, USA, Sept. 2009: 1–8.

[8] Liu Bin, Wang Kai-zhi, Liu Xing-zhao, *et al.*. An efficient signal processor of synthetic aperture radar based on GPU[C]. European Conference on Synthetic Aperture Radar, Eurogress, Aachen, Germany, June 2010: 1054–1057.

[9] Ning Xia, Yeh Chun-mao, Zhou Bin, *et al.*. Multiple-GPU accelerated range-doppler algorithm for synthetic aperture radar imaging[C]. IEEE International Radar Conference, Kansas City, MO, USA, May 2011: 698–701.

[10] Clemente C, Bisceglie M D, Santo M D, *et al.*. Processing of synthetic aperture radar data with GPGPU[C]. IEEE Workshop on Signal Processing Systems, Tampere, Finland, Oct. 2009: 309–314.

[11] 俞惊雷, 柳彬, 王开志, 等. 一种基于 GPU 的高效合成孔径雷达信号处理器[J]. 信息与电子工程, 2010, 8(4): 415–418.
Yu J L, Liu B, Wang K Z, *et al.*. A highly efficient GPU-based signal processor of Synthetic Aperture Radar[J]. *Information and Electronic Engineering*, 2010, 8(4): 415–418.

[12] 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: 120–124.
Zhang Shu and Zhe Yan-li. CUDA-High Perfermance Computing on GPU[M]. Beijing: China WaterPower Press, 2009: 120–124.

[13] Nvidia. Tesla C1060 specifications[OL]. http://www.nvidia.com/docs/IO/43395/BD-04111-001_v06.pdf.

[14] Intel. Intel xeon processor X5550 specifications[OL]. http://ark.intel.com/Product.aspx?id=37106.

[15] Fornaro G, Franceschetti G, and Perna S. On center-beam approximation in SAR motion compensation[J]. *IEEE Geoscience and Remote Sensing Letters*, 2006, 3(2): 276–280.

[16] Prats P, Macedo K A C, Reigber A, *et al.*. Comparison of topography-and aperture-dependent motion compensation algorithms for Airborne SAR[J]. *IEEE Geoscience and Remote Sensing Letters*, 2007, 4(3): 349–353.

作 者 简 介

孟大地(1979–)，男，陕西渭南人，2006 年于中国科学院电子学研究所获得工学博士学位，2001 年于西安交通大学获得工学学士学位，现任职于中国科学院电子学研究所，副研究员，研究方向为机载合成孔径雷达信号处理。
E-mail: mengdadi@hotmail.com

胡玉新(1981–)，男，内蒙古赤峰人，2007 年于中国科学院电子学研究所获得工学博士学位，2002 年于内蒙古大学获得工学学士学位，现任职于中国科学院电子学研究所，副研究员，研究方向为合成孔径雷达信号处理系统设计。
E-mail: yxhu@mail.ie.ac.cn

丁赤飚(1969–)，男，研究员，博士生导师，现任中国科学院电子学研究所副所长，主要从事合成孔径雷达、遥感信息处理和应用系统等领域的研究工作，先后主持多项国家 863 重点项目和国家级遥感卫星地面系统工程建设项目，曾获国家科技进步一等奖、二等奖各一项。
E-mail: cbding@mail.ie.ac.cn