基于 GPU 的 ISAR 成像算法实现

刘百玲 江海清 倪书爱
(北京理工大学 信息与电子学院 北京 100081)

摘 要：逆合成孔径雷达(inverse synthetic aperture radar,ISAR)是一种具有很高的成像分辨率的雷达,其成像过程是一项大数据量的高密度计算的处理任务。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 技术的发展使得多线程、单指令的 GPU 架构能够方便快速地进行并行计算。提出了一种在 cuda 平台上用 GPU 进行非相参成像处理的高效方法,利用 GPU 的并行特性,实现了成像处理过程的并行化,与一般 GPU 处理方法相比,其处理速度可以达到实时成像的效果,运算速率明显提高。

关键词：ISAR;雷达成像;图形处理器(GPU)
中图分类号：TN95 **文献标识码**：A **国家标准学科分类代码**：510.4030

The realization of ISAR imaging algorithm based on GPU

Liu Bailing Jiang Haiqing Ni Shu'ai
(Beijing Institute of Technology School of Information and Electronics,Beijing 100081,China)

Abstract: ISAR (Inverse Synthetic Aperture Radar inverse synthetic aperture radar) is a kind of high resolution imaging radar, the imaging process is a heavy task of processing the high density computing and a large amount of data. GPU has dozens of times the CPU floating-point computation capacity and transmission bandwidth. The development of CUDA technology makes the multi-threads, single instruction GPU architecture can quickly for parallel computing. This paper presents an efficient method of non coherent imaging using the parallel characteristic of GPU on the CUDA platform and realizes the parallel imaging processing. Compared with the general GPU processing method, the processing speed can achieve real-time imaging effect, and the operation rate is obviously improved.

Keywords: inverse synthetic aperture radar(ISAR); imaging radar; graphic processing umit(GPU)

1 引 言

逆合成孔径雷达是静止的雷达,可以对运动的目标进行距离向和方位向二维高分辨率成像,是一种具有高分辨率的成像雷达,用来对一般的雷达目标(飞机、卫星、导弹、船舰等)进行远距离的,全天时的,全天候的细致的观察和识别。ISAR 图像是对回波数据进行处理得到。常用的方法有 RD[1](Range Doppler 距离多普勒)算法和 BP(Back Projection 后向投影)算法。本文介绍了 RD 算法中的非相参成像。ISAR 成像的基本原理是距离多普勒原理,成像的距离分辨率通过大宽带信号得到,横向分辨率则是基于目标相对于雷达视线的转动而引起的积累角,处理的关键是运动补偿。

2 GPU 计算模型

图形处理器(graphic processing unit,GPU)是一个相对于 CPU 的概念,它专门用来在个人电脑、工作站或游戏机上处理图形、影像等运算的微处理器。它是显卡的心脏,和显卡板载内存成为一个子系统,共同决定计算机系统的图形处理性能。在计算机系统中,GPU 是 CPU 的协处理器[3]。最初用于计算机图形图像的加速,处理图形转换,光照,三角形建立、裁剪和渲染工作[9]。为了适应图形图像应用中并行度高且计算量大的需要,GPU 针对吞吐量优化了它的硬件架构,现在可编程图形处理器(GPU)已经演化成高并行度,多线程,拥有强大计算能力和极高存储器带宽的多核处理器。

特别地,GPU 非常适合处理那些能够表示为数据并行计算(同一程序在多个数据上并行执行)的问题,数据并行计算的算术计算密度(算术操作和存储器操作的比例)非常高。由于同一程序在每个元素上执行,因此对复杂流控的要求非常少,更因为在多个元素上执行和高计算密度,访存

收稿日期:2014-11

延迟可以被计算隐藏,因此用不着大的数据缓存。

与常规的 CPU 实现相比大大缩短了任务计算时间,具有很大的运算优势。如表 1 所示进行的对比。

表 1 CPU 运算优势对比

FFT 点数	CPU 平台 1 万次平均时间/ms	cuda 平台 1 万次平均时间/ms	CPU 平台平均运行时间/cuda 平台平均运行时间
32K	0.0689	0.0268	2.5709
64K	0.1360	0.035	3.9706
256K	1.4136	0.107	13.2112
1M	5.6544	0.371	15.2410

CPU 平台配置:Intel Corei7 3770K,主频 3.5 GHz,内存 16 G,Windows764 bit

CUDA 平台配置:GPU Nvidia Geforce GTX690,CPU Intel Corei7 3770 K,主频 3.5 GHz,内存 16 G,Windows7 64 bit

可见在 CUDA 平台上 FFT 运行效率优于传统的 CPU 平台,而且随着运算点数的增大,CUDA 平台的优势更加明显。

2.1 CUDA 编程模型

2007 年 6 月,NVIDIA 推出了 CUDA(compute unified device architecture,统一计算设备架构)。CUDA 是一种将 GPU 作为数据并行计算设备的软硬件体系。

CUDA 这种计算模型是以 CPU+GPU 的一种异构模式来工作的[2]。CPU 负责整体程序的串行逻辑控制和任务调度,GPU 则用于执行一些能够被高度并行化的并行计算任务。确定程序的并行部分,可以考虑将这部分计算交给 GPU,运行在 GPU 上的 CUDA 并行计算函数称为 kernel 函数,一个 kernel 不是一个完整的程序,只是 CUDA 程序中一个可以被并行执行的部分。一个完整的程序是由一系列设备端 kernel 函数和主机端串行处理的步骤共同组成的。CUDA 程序调度示意如图 1 所示。

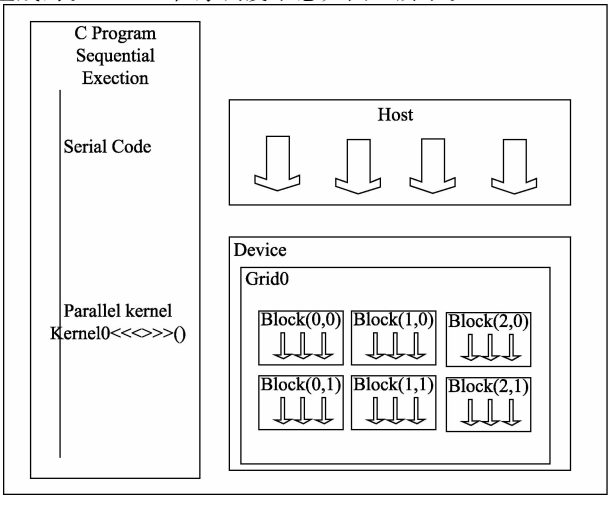

图 1 CUDA 程序调度示意

2.2 CUDA 线程结构

内核函数中的线程以线程网格(grid)的形式组织,每个线程网格由若干个线程块(block)组成,每个线程块又由若干个线程组(thread)组成。每个 block 之间并行执行,互不能通信,也没有执行顺序。CUDA 线程结构如图 2 所示。

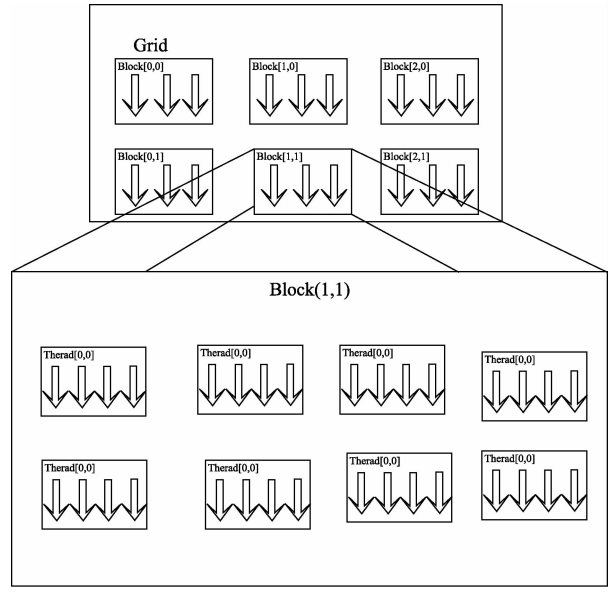

图 2 线程组织结构

3 算法介绍及 GPU 实现

距离多普勒算法中的非相参成像是被普遍使用的一种成像处理算法,对回波信号进行处理,根据信号时延和多普勒频率确定各个散射点在距离跟方位向上的分布,距离高分辨率通过发射大宽带的信号并作脉冲压缩来得到;方位向分辨率则是通过多个脉冲相干积累,用 FFT 实现。其实现过程运算量大,尤其是要达到实时成像的要求,对运算速度就具有很高的要求。对此,用 CUDA 平台上的 GPU 并行运算的机制来提高运算速度,这个架构上提供了 FFT(快速傅里叶变换)、矩阵转置、向量矩阵乘法等库的使用,大大提高了运算速度。如图 3 所示 ISAR 成像距离多普勒算法处理流程[4]。

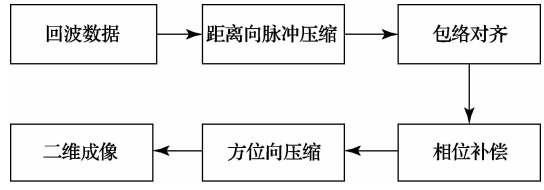

图 3 距离多普勒算法处理流程

3.1 距离向脉冲压缩

对接收到的各线性调频子脉冲回波做匹配滤波,使得

输出信噪比最大[10]。首先生成匹配滤波函数，滤波函数进行傅里叶变换，变换到频域，并求其共轭，得到频率域滤波函数；然后将回波信号进行傅里叶变换，变换到频域；最后信号与匹配滤波在频域相乘，就实现了频率匹配滤波。子脉冲压缩时可进行加窗。

此过程中用到了快速傅里叶变换，对 M×N 的二维矩阵按列进行 FFT，直接调用 FFT 库函数：

```
cufftHandle plan;    //分配执行句柄
//设置执行句柄
cufftPlan1d(&plan, M, CUFFT_C2C, N);
cufftExecC2C(plan,(cufftComplex * )data,
(cufftComplex * )data,CUFFT_FORWD);
cufftDestroy(plan);    //释放执行句柄
```

在频域，回波信号(Echo_data)与匹配滤波(Ref)相乘，在此文中回波信号是由多个脉冲组成的矩阵，其中矩阵的每一列代表一个脉冲，匹配滤波是以向量形式表示，两者相乘即矩阵与向量相乘，以下是核心代码：

```
typedef float2 Complex;//定义复数
dim3 Grid(1,1,1);    //设置网格的执行配置
dim3 Block(M,1,1); //设置线程块的执行配置
//调用 kernel 函数
ComplexMul<<<Grid,Block>>>(M,N,
Echo_idata,Ref,Echo_odata);
//kernel 函数的实现
__global__ void ComplexMul(int m,int n, Complex *
idata,Complex * ref,Complex * odata
{
unsigned int tid = blockDim.x * blockIdx.x +
threadIdx.x;
//线程索引
if(tid<m)
  { for(int i=0;i<n;i++)
    {  //数据相乘
odata[i * m+tid] = idata[i * m+tid] * ref[tid];
    }
  }
}
```

3.2　包络对齐

非相参处理时，包络对齐采用积累互相关法[6]，将与待对齐的最近的几个脉冲进行包络求和，作为基准信号，将待对齐的脉冲与基准信号做相关处理，求最大相关点所对应的位移，取点积最大值对应的位移量对包络进行移位补偿，使之与基准信号对齐，此前所说的回波均为其幅度信息。

进行相关处理的时候，主要是求向量的点积及最大值，直接调用 CUBLAS 库中的点积和最大值函数：

```
//假设两个向量的长度都是 M
cublasHandle_t handle;
cublasCreate(&handle);    //创建句柄
//点积执行语句
cublasdot(handle,M,vector_1,1,
vector_2,1,dot_sum);
cublasDestroy(handle);    //销毁句柄
//此处求取的是向量中最大值的下标(index)，向量的长度是 M
cublasHandle_t handle;
cublasCreate(&handle);    //创建句柄
//最大值执行语句
cublasIsmax(handle,M,vector,1,max_index);
cublasDestroy(handle); //销毁句柄
```

3.3　相位补偿

非相参处理时，实现相位补偿的方法有多普勒中心法[5]和相位梯度自聚焦法[8]。此处用到多普勒中心法。假设目标存在一个等效的多普勒中心，目标绕中心旋转时，将相邻回波距离对齐后共轭相乘，由此得到相邻两个回波的相位误差估计，便可把相邻回波在多普勒域对准。此过程中也用到了向量的点积，程序代码同上。

3.4　方位向压缩

方位向压缩，即在包络对齐和相位补偿后的平动补偿之后，直接在方位向对各距离单元进行 FFT(可加窗)，得到散射点的多普勒分布，于是得到了目标的 ISAR 二维成像[7]。

此处理过程中用到了 fftshift 函数，直接用快速傅里叶变换即 FFT 得出的数据与频率不是一一对应的，fftshift 的作用正是让正半轴部分和负半轴部分的图像分别关于各自的中心对称。该函数核心代码如下：

```
//fftshift 函数
__global__ void FftShift(int num,
        Complex * idata,Complex * odata)
{
unsigned int tid=
blockDim.x * blockIdx.x+threadIdx.x;
if(tid<num/2)
  {
      odata[tid] = idata[tid+num/2];
      odata[tid+num/2] = idata[tid];
  }
}
```

4　结　　论

基于 CUDA 平台的 GPU 异构系统的 ISAR 成像算法的实现，是一种新的并行计算方法，能够满足高速处理、实时成像的要求，对于程序的要求，GPU 最大的困难在于程序

(下转第 89 页)