SAR 图像压缩采样恢复的 GPU 并行实现

陈 帅[①②] 李 刚[*①] 张 颢[①] 孟华东[①] 王希勤[①]
[①](清华大学电子工程系 北京 100084)
[②](西安电子工程研究所 西安 710100)

摘 要：压缩采样(CS)技术被尝试应用于合成孔径雷达(SAR)图像的压缩。然而，高分辨 SAR 图像数据量大，导致压缩采样后的恢复过程计算量大，传统的中央处理器(CPU)无法实时成像。为解决这一问题，该文在图形处理器(GPU)平台上设计了 CS 的并行方法，并实现了 SAR 图像压缩。实验结果表明，在保证 SAR 图像压缩性能的前提下，该文设计的 GPU 并行处理速度能够提高到 CPU 串行处理的 8.8 倍。

关键词：合成孔径雷达(SAR)；压缩采样(CS)；并行计算；图形处理器(GPU)
中图分类号：TN957.52 **文献标识码**： A **文章编号**：1009-5896(2011)03-0610-06
DOI: 10.3724/SP.J.1146.2010.00461

A GPU-based Parallel Implementation of Compressive Sampling Reconstruction for SAR Image Compression

Chen Shuai[①②] Li Gang[①] Zhang Hao[①] Meng Hua-dong[①] Wang Xi-qin[①]
[①](*Department of Electronic Engineering, Tsinghua University, Beijing* 100084, *China*)
[②](*Xi'an Electronic Engineering Research Institute, Xi'an* 710100, *China*)

Abstract: Compressive Sampling (CS) technique has been adopted to compress Synthetic Aperture Radar (SAR) images. However, due to the mass data of high resolution SAR images, the reconstruction of compressive sampling generates huge computational load, making it impossible to run on traditional CPU at real time. To solve this problem, this paper attempts to implement the reconstruction produce in parallel based on Graphics Processing Unit (GPU) device. The results show that the GPU-based implementation is faster (up to 8.8 times) than the CPU-based implementation.

Key words: Synthetic Aperture Radar (SAR); Compressive Sampling (CS); Parallel computation; Graphic Processing Unit (GPU)

1 引言

合成孔径雷达(SAR)借助飞机、卫星的移动将阵元天线合成大的天线孔径，形成地面场景的 2 维高分辨图像。为节省存储与传输的成本，通常需要对 SAR 图像进行压缩。近年来，一种新兴的称之为压缩采样(Compressive Sampling，CS)[1]的技术被广泛应用于图像处理、医疗成像、雷达信号处理等诸多领域[2]。CS 理论指出，在信号满足稀疏性的前提下，用远小于奈奎斯特采样率的采样频率对数据进行采样，即能够完全恢复出原始信号[1]。已经有学者尝试应用 CS 技术来实现 SAR 图像压缩[3–6]。文献[4]中指出，CS 方法与传统的图像压缩方法以及基于标量量化、向量量化的方法相比，在 SAR 图像的压缩上有更好的性能表现。然而，用 CS 技术实现 SAR 图像压缩还存在以下困难：高分辨的 SAR 图像具有场景多变、数据量大等特征，导致应用 CS 技术实现 SAR 图像压缩计算量非常大，传统 CPU 的串行实现运行时间长，无法实时恢复出原始图像；而借助大型计算机或者集群虽然能够实现快速计算，但所需成本高。因此，为促进 CS 技术在 SAR 图像压缩领域的实际应用，必须研究低成本、高速计算的实现方法。

近年来，图形处理器(Graphic Processing Unit，GPU)的性能在图形处理市场的驱动下，发展迅速。GPU 由以前的专用图形处理器演变成一个高速并行化的多核、多线程通用应用平台，在解决计算密集型问题上具有很高的性价比[7]。基于 GPU 这一优势，本文试图在 GPU 平台上设计 CS 的并行算法，进而实现 SAR 图像的低成本、快速压缩。

目前，已有一些研究人员在 GPU 平台上对 CS

2010-05-11 收到，2010-07-13 改回
国家自然科学基金(40901157)，国家 973 计划项目(2010CB731901)和教育部新教师基金(200800031050)资助课题
*通信作者：李刚 gangli@tsinghua.edu.cn

算法进行实现，并取得了较好的性能表现[8−10]。文献[8]中，UCLA 大学的 Borghi 等人在 GPU 平台上对 CS 的 Moreau-Yosida Regularization 算法进行了实现。文献[10]中，Wisconsin 大学的 Sangkyun Lee 等人在 GPU 平台上实现了 CS 的 SpaRSA 算法。文献[9]中，Calgary 大学的 Andrecut 等人实现了匹配追踪(Matching Pursuit，MP)算法的 GPU 并行化 。而本文的工作是在 GPU 平台上设计正交匹配追踪(Orthogonal Matching Pursuit，OMP)的并行算法[11,12]，并实现 SAR 图像压缩。与文献[8−10]的工作比较，本文的优势在于所实现的 OMP 算法比文献[8,10]中采用的凸优化算法计算复杂度更小，比文献[9]中的 MP 算法收敛更快，同时本文更进一步地将 OMP 算法的并行实现应用于 SAR 图像的压缩采样，并取得很好的性能表现。

2　CS 及在 SAR 图像压缩中的应用

SAR 图像经过离散余弦变换(Discrete Cosine Transformation，DCT)或者小波变换具有一定的稀疏性，因而能够利用 CS 技术对 SAR 图像压缩。在这一节中，结合 SAR 图像特征和 CS 理论，讨论如何对 SAR 图像进行压缩采样。

2.1 CS 理论简要回顾

对于原始信号 $\boldsymbol{f}\in R^N$，通过观测矩阵 $\boldsymbol{\Phi}\in R^{M\times N}$，得到观测向量 $\boldsymbol{x}\in R^M$。

$$\boldsymbol{x}=\boldsymbol{\Phi f} \tag{1}$$

其中 $M<<N$，$\boldsymbol{f}$ 中显著元素个数为S，$S<<N$ [2]。CS理论研究的是：已知观测 $\boldsymbol{x}$，估计满足式(1)的最稀疏解 $\boldsymbol{f}$，即找到一个 $\tilde{\boldsymbol{f}}\in \boldsymbol{R}^N$ 满足：

$$\min\left\|\tilde{\boldsymbol{f}}\right\|_0,\ \text{s.t. } \boldsymbol{x}=\boldsymbol{\Phi}\tilde{\boldsymbol{f}} \tag{2}$$

其中 $\|\cdot\|_0$ 表示 L_0 范数，即计算非零元素个数。估计信号 $\tilde{\boldsymbol{f}}$ 的准确性需要采样数和 $\boldsymbol{\Phi}$ 矩阵约束等距性(Restricted Isometry Property，RIP)[2]的保证。CS 理论指出，当 $\boldsymbol{\Phi}$ 具有良好 RIP 性质，且采样数 M 满足：

$$M\geq C\cdot S\cdot \lg N \tag{3}$$

能够以非常大的概率恢复出原始信号。式(3)中，C 是与 N，S 无关的常数。

2.2 图像的 DCT 与压缩采样

在图像处理中，常采用 2 维 DCT 或小波变换等方法对图像进行压缩处理，本文采用 2 维 DCT 作为压缩变换。2 维 DCT 定义为

$$\boldsymbol{G}_{u,v}=\alpha(u)\alpha(v)\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}\boldsymbol{X}_{u,v}\cos\left[\frac{(2n+1)u\pi}{2N}\right]\cdot\cos\left[\frac{(2m+1)v\pi}{2N}\right],\ u,v=0,1,\cdots,N-1 \tag{4}$$

其中 $\boldsymbol{X}$ 为 SAR 幅度图像，$\boldsymbol{G}$ 为变换后的 DCT 域表示，$\boldsymbol{G}_{u,v}$ 表示 $\boldsymbol{G}$ 的第 u 行第 v 列元素，$\alpha(u)$ 定义为

$$\alpha(u)=\begin{cases}\sqrt{1/N}, & u=0\\ \sqrt{2/N}, & u=1,2,\cdots,N\end{cases} \tag{5}$$

用矩阵来表示 2 维 DCT 为 $\boldsymbol{G}=\boldsymbol{DXD}^{\mathrm{T}}$，其中 $\boldsymbol{D}$ 是 DCT 变换矩阵，相应的逆变换为

$$\boldsymbol{X}=\boldsymbol{D}^{\mathrm{T}}\boldsymbol{GD} \tag{6}$$

图像通过2维DCT变换到DCT域往往表现出稀疏性，即$\boldsymbol{G}$的能量集中在少数显著元素上。对式(6)向量化：

$$\mathrm{vec}(\boldsymbol{X})=\mathrm{vec}(\boldsymbol{D}^{\mathrm{T}}\boldsymbol{GD})=(\boldsymbol{D}\otimes\boldsymbol{D}^{\mathrm{T}})\mathrm{vec}(\boldsymbol{G}) \tag{7}$$

其中 $\mathrm{vec}(\boldsymbol{X})$ 描述如下：对于矩阵 $\boldsymbol{X}=[x_{i,j}]$(其中，$i=1,2,\cdots,m$，$j=1,2,\cdots,n$，$x_{i,j}$ 表示$\boldsymbol{X}$第i行第j列元素)，$\mathrm{vec}(\boldsymbol{X})=(x_{1,1},x_{2,1},\cdots,x_{m,1},x_{1,2},x_{2,2},\cdots,x_{m,2},\cdots,x_{1,n},x_{2,n},\cdots,x_{m,n})^{\mathrm{T}}$；$\otimes$ 表示Kronecker积[13]，因为$\boldsymbol{D}$为正交矩阵，$\boldsymbol{D}\otimes\boldsymbol{D}^{\mathrm{T}}$ 亦为正交矩阵。式(7)说明 $\mathrm{vec}(\boldsymbol{X})$ 能够通过正交变换变换成稀疏信号 $\mathrm{vec}(\boldsymbol{G})$，故能使用CS技术对 $\mathrm{vec}(\boldsymbol{G})$ 进行压缩采样。压缩采样过程通过对图像$\boldsymbol{X}$的随机采样实现，描述为

$$\boldsymbol{y}=\boldsymbol{J}\cdot\mathrm{vec}(\boldsymbol{X})=\boldsymbol{J}(\boldsymbol{D}\otimes\boldsymbol{D}^{\mathrm{T}})\mathrm{vec}(\boldsymbol{G}) \tag{8}$$

其中 $\boldsymbol{J}$ 是作用在 $\mathrm{vec}(\boldsymbol{X})$ 上随机采样过程的矩阵表示。$\boldsymbol{J}$的具体描述如下：设采样过程中 M 个采样值在原始信号中的下标序列为 $\{s_1,s_2,\cdots,s_M\}$，$s_i\in\{1,2,\cdots,N\}$，则 $\boldsymbol{J}=\left[\boldsymbol{e}_{s_1}^{\mathrm{T}}\ \boldsymbol{e}_{s_2}^{\mathrm{T}}\ \cdots\ \boldsymbol{e}_{s_M}^{\mathrm{T}}\right]^{\mathrm{T}}$，其中 $\boldsymbol{e}_k$ 是基本向量，定义 $\boldsymbol{e}_k=\left(0,0,\cdots,0,\overset{\text{第}k\text{元素}}{1},0,\cdots,0\right)^{\mathrm{T}}$。例如：若从矢量 $\boldsymbol{v}=(v_1,v_2,\cdots,v_8)^{\mathrm{T}}$ 中采样得矢量 $\boldsymbol{u}=(v_7,v_2,v_4)^{\mathrm{T}}$，则该操作可表示为 $\boldsymbol{u}=\boldsymbol{Jv}$，其中

$$\boldsymbol{J}=\begin{bmatrix}0&0&0&0&0&0&1&0\\0&1&0&0&0&0&0&0\\0&0&0&1&0&0&0&0\end{bmatrix}$$

从上面的分析知，SAR 图像压缩采样的稀疏恢复为

$$\min\left\|\mathrm{vec}(\boldsymbol{G})\right\|_0,\ \text{s.t. } \left\|\boldsymbol{y}-\boldsymbol{J}(\boldsymbol{D}\otimes\boldsymbol{D}^{\mathrm{T}})\mathrm{vec}(\boldsymbol{G})\right\|_2^2\leq\varepsilon \tag{9}$$

其中 $\|\cdot\|_2$ 表示 L_2 范数，ε 表示误差约束参数。一旦通过式(9)估计出 $\mathrm{vec}(\boldsymbol{G})$，SAR幅度图像$\boldsymbol{X}$可由式(6)获得。

3　SAR 图像的 GPU 恢复实现

3.1 OMP 算法并行实现

求解 CS 问题的常用方法是基追踪(Basic Pursuit，BP)一类的优化算法，但这种解决方法不是唯一的。近年来提出的贪婪类算法由于实现简单，

被广泛应用[14]。这类贪婪算法包括MP算法，OMP算法，压缩采样匹配追踪(Compressive Sampling Matching Pursuit，CosaMP)算法等。本文使用OMP算法对压缩采样后的SAR幅度图像进行恢复。OMP算法主要由选基和最小二乘估计两个过程组成，选基是从观测矩阵 $\boldsymbol{\Phi}$ (即式(8)中的 $\boldsymbol{J}(\boldsymbol{D}\otimes\boldsymbol{D}^{\mathrm{T}})$)中贪婪地挑选出与观测信号相关度较大的列(所挑选的列在观测矩阵中对应的列标构成支持集 Λ)；最小二乘估计是在支持集上对显著元素的值进行估计。OMP算法具体可描述为以下 5 步[11]：

(1)初始化残差 $\boldsymbol{r}_0=\boldsymbol{x}$，迭代步数 t=0，支持集 $\Lambda=\{\ \}$；

(2)检查迭代是否终止；

(3)计算残差与观测矩阵各列的相关度，找出相关度最大的列对应的列标 i_t；

(4)扩充支持集 $\Lambda_t=\Lambda_{t-1}\cup\{i_t\}$，在新的基矩阵 $\boldsymbol{\Phi}_{\Lambda_t}$ 上，采用最小二乘的方法估计出 $\boldsymbol{f}_t$；

(5)更新残差：$\boldsymbol{r}_t=\boldsymbol{x}-\boldsymbol{\Phi}_{\Lambda_t}\boldsymbol{f}_t$，迭代步数 t=t+1，继续迭代。

在以上描述中，Λ_t 表示第 t 步的支持集，$\boldsymbol{\Phi}_{\Lambda_t}$ 表示由 $\boldsymbol{\Phi}$ 中列标为 Λ_t 中元素的列构成的基矩阵，$\boldsymbol{r}_t$ 表示 t 步迭代后的残差。从上面描述知，OMP 算法的计算量主要来自向量、矩阵运算。在向量、矩阵运算中，元素之间相关性小，适合并行处理。同时，考虑到高分辨 SAR 图像数据量大，并行处理比串行处理能取得很高的加速比。在具体的实现上，使用了 NVIDIA 公司开发的 CUDA[7]并行编程平台。CUDA 并行模型基于单指令多数据结构(Single Instruction Multiple Data，SIMD)，在向量、矩阵运算方面性能优势明显。OMP 算法的 GPU 并行实现描述如图 1 所示。

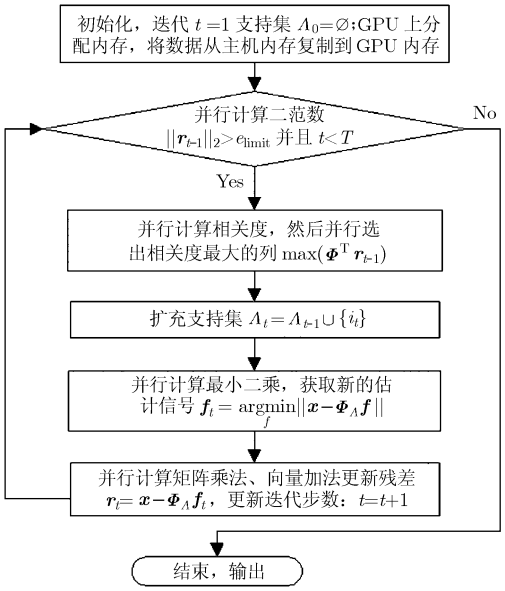


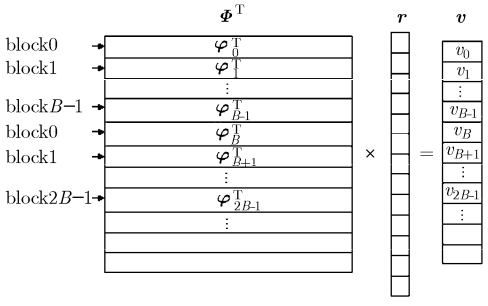
图 1 OMP 算法的 GPU 实现

3.1.1 矩阵与向量乘法的并行化 根据 CUDA 模型三级并行粒度的特点，矩阵与向量乘法的并行实现分为粗粒度并行和细粒度并行。在 $\boldsymbol{v}=\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{r}$ 中，矩阵各行与 $\boldsymbol{r}$ 之间执行向量乘法是粗粒度并行，由 CUDA 平台中的线程块(Thread Block)来完成；向量乘法中元素与元素执行乘法操作是细粒度并行，由 CUDA 平台中的线程来完成，如图 2 所示。

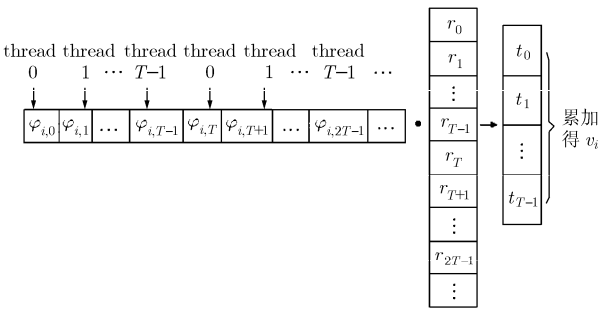
在上面的描述中，$\boldsymbol{r}=(r_0,r_1,r_2,\cdots,r_{N-1})^{\mathrm{T}}$，$\boldsymbol{v}=(v_0,v_1,v_2,\cdots,v_{M-1})^{\mathrm{T}}$，$(t_0,t_1,\cdots,t_{T-1})$ 为同一个线程块中的各线程计算的中间结果。存储优化方面，将同一个线程块访问的资源 v_i，t_i 存储在共享内存中，减少访问延迟。同时，计算 v_i 需要将同一个线程块中各线程计算的中间结果 t_i 累加。传统的实现方式是由线程 0 执行此操作，然而，在 CUDA 的并行机制中，多个线程构成的一组线程都执行相同的指令。因而，在线程 0 执行累加操作时，其余线程的资源将浪费。为提高实现的并行度，在此处采用树状加法。实现方式如图 3 所示。

对于 N 个线程构成的并行级，传统的累加运算需要 N–1 步，而树形加法只需要 $\log_2 N$ 并行步。

3.1.2 QR 分解并行化 OMP 算法中，需要通过最



(a) 粗粒度的并行



(b) 细粒度的并行

图 2 矩阵向量乘法的 GPU 并行实现(B 为线程块数目，T 为每个线程块中线程数目)

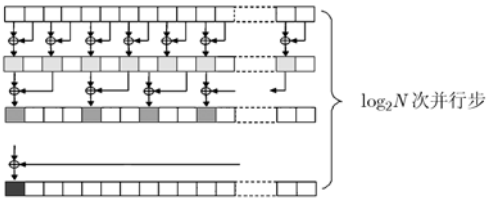


图 3 树状加法

小二乘来估计新的信号。最小二乘 $\boldsymbol{f}_t = \arg\min_{f} \left\| \boldsymbol{x} - \Phi_{A_t} \boldsymbol{f} \right\|_2^2$ 的解为 $\boldsymbol{f}_t = (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{x}$，其中矩阵的求逆常通过 QR 分解实现，对于 $n \times n$ 的矩阵 QR 分解求逆，计算量为 $\Omega(n^3)$。在 OMP 的迭代过程中，随着支持集的扩充，计算时间增长迅速，从而限制了运算速度。为解决此计算瓶颈，本文借助 GPU 来加速 QR 分解。文献[15]中讨论了 QR 分解的 Household 反射并行算法和 Givens 旋转并行算法，本文将采用 Givens 旋转并行算法实现 QR 分解。

对 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 矩阵进行 QR 分解描述为

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R} \tag{10}$$

其中 $\boldsymbol{Q}$ 为正交矩阵，$\boldsymbol{R}$ 为上三角矩阵。令 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{y}$，最小二乘估计即为求解下面的线性方程：

$$\boldsymbol{R}\boldsymbol{f}_t = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{y} \tag{11}$$

其中 $\boldsymbol{Q}^{\mathrm{T}}$ 不需显式计算，对 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 进行变换的同时，对 $\boldsymbol{y}$ 进行相应计算。

对于矩阵 $\boldsymbol{X}$ 的 $i, j (i < j)$ 两行，如果 $x_{i,l} = 0$，$x_{j,l} = 0 (l = 0,1,\cdots,k-1)$，可通过 Givens 旋转消去 $x_{j,k}$ (定义 (i, j) 为一个消去对)。Givens 旋转中，矩阵中能同时消去且互不影响的 (i, j) 有多对，可并行进行消去。算法执行中，第 1 步，满足并行消去条件的消去对为 $(x_{n,1}, x_{\lceil n/2 \rceil,1}), (x_{n-1,1}, x_{\lceil n/2 \rceil-1,1}), \cdots, (x_{\lceil n/2 \rceil+1,1}, x_{2\lceil n/2 \rceil-n+1,1})$；第 2 步，满足并行消去条件的消去对为 $(x_{\lceil n/2 \rceil,1}, x_{\lceil n/4 \rceil,1}), (x_{\lceil n/4 \rceil-1,1}, x_{\lceil n/2 \rceil-1,1}), \cdots, (x_{\lceil n/4 \rceil+1,1}, x_{2\lceil n/4 \rceil-\lceil n/2 \rceil+1,1}), (x_{n,2}, x_{n-\lceil n/4 \rceil,1}), (x_{n-1,2}, x_{n-\lceil n/4 \rceil-1,2}), \cdots, (x_{n-\lceil n/4 \rceil+1,2}, x_{n-\lceil n/2 \rceil+1,2})$；第 3 步，依次类推…，直至完成 QR 分解。在上面的描述中，$\lceil a \rceil$ 表示求 $\geq a$ 的最小整数。

不妨以 $n = 8$ 举例说明，如图 4 所示。

该并行算法需要的并行步数为 $\log_2 n + (n-1) \cdot \log_2 \log_2 n$ [15]。在 OMP 算法中，$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 是一个 $t \times t$ 维的矩阵，其中 t 为当前迭代步数。当 t 很大，上述算法能取得明显的并行加速。

本文的GPU实现中，Givens旋转并行实现分为线程块级和线程级。在线程块级，每个线程块负责消去对中两行元素的并行旋转操作；在线程级，每个线程负责行向量元素在旋转中的并行乘法操作。存储优化方面，由于每个线程块对应一个旋转矩阵，其内部线程都需要访问旋转矩阵，因而将旋转矩阵存储在共享内存中可以减少内存访问时间。另一方面，为防止内存读写冲突，每一并行步完成之后，需要保证所有线程同步，实现中借助CUDA模型中的__syncthreads 同步机制实现。考虑到OMP算法中，$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 的维度逐步增加，在算法的初始阶段，t 很小，并行算法不但不能缩短计算时间，反而因其调度的复杂性消耗了计算时间。实现过程中，应根据 t 自适应地调整实现策略：在 t 比较小的时候，串行实现；在 t 超过某一大小，采用GPU并行实现。

3.2 分块恢复

并行 CS 算法能处理的 SAR 图像大小受限于 GPU 内存大小。例如，当图像大小为 1024×1024 像素，压缩比例为 2 时，观测矩阵 $\boldsymbol{\Phi}$ 所需存储空间为 16 G(元素类型为 float 型)，然而，现有的大部分 GPU 均无法满足此存储空间的要求。为消除硬件平台的限制，在实现过程中，将待处理的 SAR 图像分割成若干块，对每块分别进行并行恢复。本文中设计每个图片块的大小为 64×64 像素。分块恢复算法描述如图 5 所示。

4 实验结果与分析

实验数据来自 ERS-SAR 幅度图像，实验 CPU 平台为 Intel Core 2 Duo E8400 处理器，GPU 平台为 NVIDIA Geforce 9600GT GPU。原始图像如图 6(a)所示，采用 CS 技术以不同压缩比对原始图像采样后，使用 OMP 算法并行恢复的效果如图 6(b)，6(c)所示。

从实验结果知，压缩比例为 2 时，恢复的图像质量很高，与原始图像基本相同；在压缩比为 4 时，依然有比较好的恢复效果，但与压缩比例为 2 时比

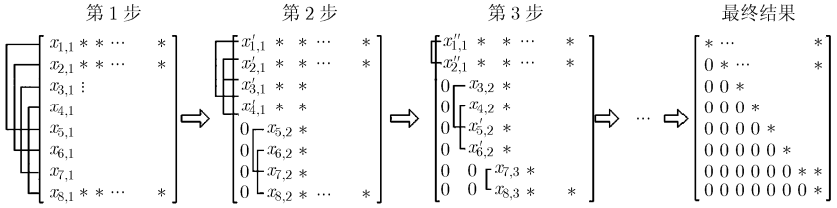


图 4 8×8 矩阵并行 QR 分解过程

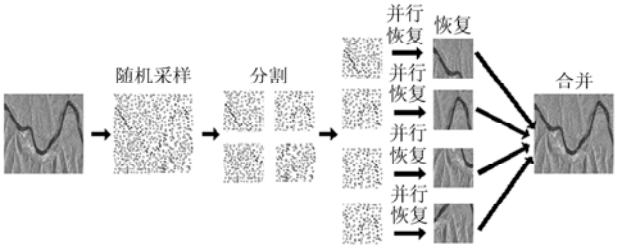


图 5 分块恢复

较，损失了一定的图像清晰度，但在分辨率要求不高的情况下依然能满足要求。以峰值信噪比(Peak Signal-to-Noise Ratio，PSNR)来衡量图片压缩的质量，PSNR 定义为 $\mathrm{PSNR} \triangleq 10 \cdot \lg\left(\dfrac{\mathrm{MAX}_I^2}{\mathrm{MSE}}\right)$，其中 MAX_I 是图像 $\boldsymbol{I}$ 中像素灰度的最大值，$\mathrm{MSE} \triangleq \dfrac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left[\boldsymbol{I}(i,j)-\boldsymbol{K}(i,j)\right]^2$，$m$，$n$ 为图像的维度，$\boldsymbol{I}(i,j)$ 表示压缩前图像第 i 行第 j 列像素的灰度值，$\boldsymbol{K}(i,j)$ 表示压缩后图像第 i 行第 j 列像素的灰度值。比较图 6(b)，图 6(c)的峰值信噪比(Peak Signal-to-Noise Ratio，PSNR)，压缩比例为 2 时，$\mathrm{PSNR}_1 = 21.76$；压缩比例为 4 时，$\mathrm{PSNR}_2 = 18.63$，可见，高的压缩比需要以损失图像质量为代价。

应用 GPU 的关键是提高计算速度，下面的实验分析 CPU 和 GPU 计算性能的差别。表 1 是原始场景(图 6 (a))中 64×64 像素的图片块的恢复时间和价格比较。

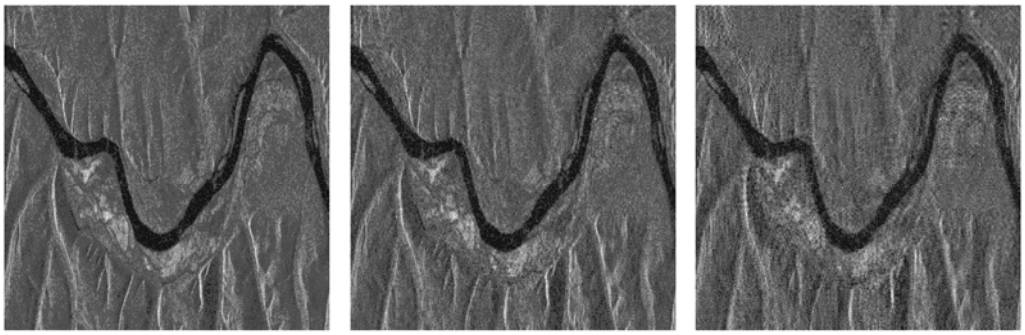
表 1 64×64 像素的图像 CPU 与 GPU 性能、价格比较

芯片类型	CPU(Intel Core Duo 2 E8400)	GPU(Geforce 9600GT)	比值
时间(s)	150.8972	17.06	8.8
相对成本	299.95	112.68	2.66

实验结果表明，针对 64×64 像素图像块的处理，GPU 的计算速度是 CPU 的 8.8 倍，而价格方面，实验中使用的 CPU 却是 GPU 的 2.66 倍。可见，GPU 比 CPU 具有明显的性价比优势。

5 结束语

本文给出了一种应用于 SAR 图像压缩采样的 GPU 并行方法。该方法首先将 SAR 图像的压缩处理转化为一个 CS 模型，然后在 GPU 上基于 CUDA 并行平台对 CS 的 OMP 算法进行并行实现，并进行了多处并行优化工作，最终有效地解决了 CS 技术应用于 SAR 图像压缩中计算需求大、恢复时间长的问题。实验结果表明，本文给出的 GPU 并行实现与 CPU 串行实现相比有明显的性价比优势。



(a) 原始场景图　(b) 压缩比例为 2 的压缩采样恢复　(c) 压缩比例为 4 的压缩采样恢复

图 6 原始图像与不同压缩比例下所恢复的图像

参 考 文 献

[1] Donoho D L. Compressed sensing[J]. *IEEE Transactions on Information Theory*, 2006, 52(4): 1289–1306.

[2] Candes E J and Wakin M B. An introduction to compressive sampling[J]. *IEEE Signal Processing Magazine*. 2008, 25(2): 21–30.

[3] Rilling G, Davies M E, and Mulgrew B. Compressed sensing based compression of SAR raw data[C]. SPARS'09 - Signal Processing with Adaptive Sparse Structured Representations, Saint Malo, France, 2009, ID 00369560.

[4] Bhattacharya S, Blumensath T, and Mulgrew B, *et al*.. Fast encoding of synthetic aperture radar raw data using compressed sensing[C]. IEEE/SP 14th Workshop on Statistical Signal Processing, Madison, WI, USA, 2007: 448–452.

[5] Li J, Xing M, and Wu S. Application of compressed sensing in sparse aperture imaging of radar[C]. APSAR 2009. 2nd Asian-Pacific Conference on Synthetic Aperture Radar, Xi'an, China, 2009: 651–655.

[6] Patel V M, Easley G R, and Healy D M, *et al*.. Compressed synthetic aperture radar[J]. *IEEE Journal of Selected Topics in Signal Processing*, 2010, 4(2): 244–254.

[7] Nvidia. CUDA Programming Guide Version 2.3.1 [EB/OL]. http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/NVIDIA_CUDA_Programming_Guide_2.3.pdf, 2010–03–05.

[8] Borghi A, Darbon J, and Peyronnet S, *et al*.. A simple compressive sensing algorithm for parallel many-core architectures. Department of Mathematics, UCLA, CAM Report 08-64, September, 2008.

[9] Andrecut M. Fast GPU implementation of sparse signal recovery from random projections [J]. *Engineering Letters*, 2009, 17(3): 151–158.

[10] Sangkyun Lee S W. Implementing algorithms for signal and image reconstruction on graphical processing units. Computer Sciences Department, University of Wisconsin-Madison, Tech. Rep., November, 2008.

[11] Tropp J A and Gilbert A C. Signal recovery from random measurements via orthogonal matching pursuit[J]. *IEEE Transactions on Information Theory*, 2007, 53(12): 4655–4666.

[12] Pati Y C, Rezaiifar R, and Krishnaprasad P S. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition[C]. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers, Pacific Grove, GA, USA, 1993: 40–44.

[13] 张贤达. 矩阵分析与应用[M]. 北京: 清华大学出版社, 2004: 107–117.

[14] Tropp J A and Wright S J. Computational methods for sparse solution of linear inverse problems[J]. *Proceedings of IEEE*, 2010, 98(6): 948–958.

[15] Modi J. Parallel Algorithms and Matrix Computation[M]. Oxford [Oxfordshire] New York Clarendon Press Oxford University Press, 1988: 196–204.

陈　帅：　男，1987 年生，硕士生，研究方向为雷达信号的压缩采样、并行计算等.

李　刚：　男，1979 年生，助理研究员，研究方向为微波成像.

张　颢：　男，1972 年生，副教授，研究方向为雷达信号处理.

孟华东：　男，1977 年生，副教授，研究方向为雷达信号处理.

王希勤：　男，1968 年生，教授，研究方向为雷达系统与信号处理.