基于 GPU 并行计算的压缩感知苹果图像重构方法研究

代媛[1,2], 何东健[1*], 杨龙[2]

1 西北农林科技大学机械与电子工程学院，陕西杨凌 712100
2 西北农林科技大学信息工程学院，陕西杨凌 712100
hdj168@nwsuaf.edu.cn

摘要[4]：压缩感知技术为机器视觉应用于苹果图像的采集提供一个新的途径，但其重构算法耗时长，影响苹果图像实时获取。本文采用 2D-OMP 图像重构算法、借助 GPU 通用并行计算平台，利用 CUDA 技术，设计一个并行化的苹果图像重构方法。实验结果表明，并行化的算法可将苹果图像重构效率提高约 16-35 倍，能在数秒内恢复原始图像，为苹果果园远程实时监控及基于图像的苹果质量快速检测与分类等应用提供条件。

关键词：苹果图像；GPU；并行计算；重构算法

1 引言

在我国，苹果的生长管理与生产方式大多采用粗放式和人工的传统方式，给果农和商家均带来较大的经济损失。因此，实现对苹果生长过程进行远程监控、成熟苹果机器人自动采摘、苹果自动化检测与分级等，将有助于推动我国苹果产业的发展。图像采集是机器视觉技术发展的关键技术之一[1]，而苹果图像采集是苹果果园远程监控系统对目标进行识别及苹果品质检测与分级等的重要基础[2]。目前苹果图像仍然依照香农采样定理的传统方法采集，采集到的图像数据量大，难以存储和传输[3]。Donoho 等人在 2006 年提出一种新的压缩感知（Compressed Sensing，CS）理论，为数据采集技术带来了革命性的突破[4]，CS 理论表明利用较少的信号值可实现稀疏信号的重建。近年来，压缩感知在诸多领域受到了关注与重视[5]，也逐渐应用到农业领域。蔡骋等应用压缩感知理论对杂草种子进行分类识别[6]，最高识别率可达到 90%；杨小青将压缩感知应用到水果分级系统，实现对水果颜色、大小和缺陷等的分级并取得良好的结果[7]。但是，他们都未在加速方面展开工作，随着样本尺寸增大及数目的增多，重构信号时间急剧增加，给需实时处理场合带来困难。

近年来，计算机图形处理器（GPU、Graphics Processing Unit）以其性能、编程、价格等优越性而受到青睐[8]，且 GPU 越来越广泛地应用于图形学之外的其它通用计算领域，即基于 GPU 的通用计算（GPGPU，General Purpose GPU）[9-10]，GPGPU 的发展也为压缩感知重构算法执行效率的提高提供了新的技术途径[11]。因此，针对苹果图像采用传统方法采集数据量大的问题，本文采用压缩感知技术采集数据并研究压缩感知中重构算法这一关键技术，借助 GPU 通用计算平台，采用 CUDA 技术，设计 2D-OMP (Two dimensional orthogonal matching pursuit) 并行化的苹果重构算法，提高其执行效率以使其满足实时处理的需求，解决传统方法因数据量大而带来的存储、传输及基于图像的后续各类应用效率低下的问题。

2 压缩感知苹果图像的重构

2.1 压缩感知理论

压缩感知理论主要包括信号的稀疏表示、编码测量和重构算法三个方面。其主要思想是首先对可稀疏表示或可压缩的原始信号进行稀疏变换，再利用随机传感矩阵把稀疏信号从高维度上投影到一个相对较低维度的随机弥散空间上，最后利用信号重构算法在概率意义上实现信号的精确重构或者在误差允许范围的近似重构。压缩感知理论下的信号压缩采样过程如图 1 所示：假设 $x \in \mathbb{R}^n$ 是一个在 $\Psi \in \mathbb{R}^{n \times n}$ 域内 k 稀疏的一维信号，其中 $x = \Psi z$，并且在 z 中有 $k \ll n$ 个非零值。通过测量矩阵 $\Phi \in \mathbb{R}^{m \times n}(k < m < n)$ 对 x 进行采样，可得到 $y = \Phi x = Az \in \mathbb{R}^m$，其中 $A = \Phi\Psi$ 为传感矩阵，最后利用测量值，采用压缩感知重构算法就可高精度的恢复原始信号。

图 1 压缩感知及信号重构过程

2.2 2D-OMP 重构算法

目前常用的压缩感知重构方法主要是针对 L_0 范数最小提出的一系列贪婪算法[12]，其中正交匹配跟踪算法(Orthogonal Matching Pursuit, OMP)是一个典型的贪婪算法，它首先寻找字典中与残差最匹配的原子，将其添加到已选择的原子序列中去，并对已选择的原子进行 Schmidt 正交变换，之后将信号投影到正交原子构成的空间上，得到信号在已选原子的分量和迭代残差，反复迭代。该方法实现简单、效率较高，但是

国家支撑计划课题资助（2012BAH29B04-00）
作者简介：代媛，博士生，主要从事并行计算、图像压缩编码研究。
通讯作者：何东健，教授，博士生导师，主要从事图像处理、智能监测及虚拟现实方面研究。

由于 OMP 算法是为一维信号设计的，而二维信号的恢复需通过二维的离散采样将其信号转换为一维信号，导致其存储空间较大并解码端复杂度高。而 Fang Yong 等人在 2012 年提出一个 2D-OMP 算法[13]，可直接对二维信号操作。2D-OMP 算法不仅在效率上优于 1D-OMP 算法还节省内存空间，因此，本文采用 2D-OMP 算法重构压缩感知的苹果图像。

算法 1: 2D orthogonal matching pursuit

Inputs:

$\mathbf{A} \in \mathbb{R}^{m \times n}$: 采样矩阵

$\mathbf{Y} \in \mathbb{R}^{m \times m}$: 样本

k: 稀疏度

Output:

$\tilde{\mathbf{Z}} \in \mathbb{R}^{n \times n}$: 重构信号 $\mathbf{Z}$

Variables:

$\mathbf{R} \in \mathbb{R}^{m \times m}$: 残差

$(\mathbf{i}, \mathbf{j})$: 标注将要选择原子的坐标，$\mathbf{i}$ 表示行，$\mathbf{j}$ 代表列。

Initialization:

$\mathbf{R} \leftarrow \mathbf{Y}$

$(\mathbf{i}, \mathbf{j}) \leftarrow \{(1,1),(1,2),\ldots,(n,n)\}$

For $t \leftarrow 1$ **to** k **do**

$$(i_t, j_t) \leftarrow \arg\max_{(i',j') \in (\mathbf{i},\mathbf{j})} \frac{\left|\left\langle \mathbf{R}, \mathbf{B}_{i',j'} \right\rangle\right|}{\left\|\mathbf{B}_{i',j'}\right\|_2}$$

$$(\mathbf{i}, \mathbf{j}) \leftarrow (\mathbf{i}, \mathbf{j}) \setminus (i_t, j_t)$$

$$\hat{u} \leftarrow \arg\min_{u} \left\| \mathbf{Y} - \sum_{t'=1}^{t} u_{t'} \mathbf{B}_{i_{t'}, j_{t'}} \right\|_2$$

$$\mathbf{R} \leftarrow \mathbf{Y} - \sum_{t'=1}^{t} \hat{u}_{t'} \mathbf{B}_{i_{t'}, j_{t'}}$$

For $t \leftarrow 1$ **to** k **do**

$$\tilde{z}_{i_t, j_t} \leftarrow \hat{u}_t$$

令 $\mathbf{X} \in \mathbb{R}^{n \times n}$ 是一个在 $\mathbf{\Psi}$ 域内 k 稀疏的二维信号，即 $\mathbf{X} = \mathbf{\Psi Z \Psi}^{\mathrm{T}}$，并且在 $\mathbf{Z}$ 中有 $k \ll n^2$ 个非零值，其中 $(\cdot)^T$ 代表矩阵的转制。通过测量矩阵 $\mathbf{\Phi}$ 对 $\mathbf{X}$ 进行采样可得到 $\mathbf{Y} = \mathbf{\Phi X \Phi}^T = \mathbf{AZA}^T \in \mathbb{R}^{m \times m}$，其中 $\mathbf{A} = \mathbf{\Phi \Psi}$。令 $\mathbf{A} = (\mathbf{a}_1, \cdots, \mathbf{a}_n)$，$\mathbf{a}_i$ 代表 $\mathbf{A}$ 的第 i 列。在 2D-OMP 中，字典包含 n^2 个原子，并且每个原子是一个 $m \times m$ 的矩阵。算法 1 给出 2D-OMP 的主要步骤，算法需要两个辅助变量，首先，利用 (i, j) 标注将要选择原子的坐标，i 表示行，j 代表列；其次，$\mathbf{R} \in \mathbb{R}^{m \times m}$ 用来存储从 $\mathbf{Y}$ 拿走匹配原子后的剩余残差。算法开始将 $\mathbf{Y}$ 的值赋给 $\mathbf{R}$，此后在每次迭代中，解码端在字典中寻找和残差最为匹配的原子，并通过最小二乘更新已选择的原子集[13]。

2.3 基于 GPU 并行计算的苹果图像重构

在压缩感知图像的重构过程中，其算法复杂度高，运算效率低。而 2D-OMP 算法包含大量的矩阵运算，可进行并行计算。GPU 具有良好的并行处理能力，且其在性能、编程、及价格方面有着优越性，故采用 GPU 平台开发其相应的并行算法，以提高 2D-OMP 算法重构图像的效率。

2.3.1 GPU 架构及 CUDA 技术

传统的 GPU 通用计算开发方式要求开发人员不仅熟悉需要移植具体应用的并行算法，还要精通计算机图形学和 GPU 架构才能完成 GPGPU 的开发[14]。该编程模型由于过多的存储器访问降低了效率，并且不允许线程之间通信，使其适用范围受到了限制。直到 2007 年，NVIDIA 推出了一种将 GPU 作为数据并行计算设备的软硬件体系 CUDA(Computer Unified Device Architecture，统一计算设备结构)，它采用统一处理架构，能有效地利用过去分布在顶点着色器和像素着色器的计算资源，并引入了片内共享存储器，支持随机写入和线程间通信，这种架构使得 GPU 更加适合进行通用计算。CUDA 构架分为两部分，主机 Host 和从设备 Device，通常 CPU 被看作是主机，而 GPU 则认为是从设备[15]。主程序是由 CPU 来执行，当遇到数据并行处理的部分，CUDA 就会将程序编译成 GPU 能执行的程序，并传送到 GPU。

2.3.2 重构算法的并行计算

2D-OMP 算法主要分为四个步骤，下面对其各步骤进行分析，确定各步骤的复杂度及可并行部分。

（1）投影：在 2D-OMP 算法中，字典包含 n^2 个原子，而每个原子是一个 $m \times m$ 的矩阵，设 $\mathbf{B}_{i,j}$ 为第 (i, j) 个原子并定义为：$\mathbf{B}_{i,j} = \mathbf{a}_i \mathbf{a}_j^T$，$\mathbf{Y}$ 可用 $\mathbf{B}_{i,j}$ 的加权和表示，那么 $\mathbf{Y}$ 在 $\mathbf{B}_{i,j}$ 的投影为 $\left\langle \mathbf{Y}, \mathbf{B}_{i,j} \right\rangle / \left\| \mathbf{B}_{i,j} \right\|_2$，其中 $\left\langle \mathbf{Y}, \mathbf{B}_{i,j} \right\rangle = \mathbf{a}_i^T \mathbf{Y} \mathbf{a}_j$，$\left\| \mathbf{B}_{i,j} \right\|_2$ 代表 $\mathbf{B}_{i,j}$ 的范数。设 $\mathbf{P}$ 是一个 $n \times n$ 的矩阵，并且 $\left\| \mathbf{B}_{i,j} \right\|_2$ 是其第 (i, j) 个元素，那么投影这步即可以这样方式实现：$\mathbf{A}^T \mathbf{RA} ./ \mathbf{P}$，因此当 $m < n$ 时，这步的复杂度为 $O(mn^2)$，且矩阵相乘是这部分并行的核心。

（2）选择最匹配原子：在未选择的原子中寻找绝对值最大的投影所对应原子。由于一共有 n^2 个原子，而 $n^2 \gg k$，因此，该部分的复杂度为 $O(n^2)$，且矩阵元素求最值可并行完成。

（3）更新权重：用 t 个已选择原子的加权和逼近 $\mathbf{Y}$，设

$$\mathbf{R}=\mathbf{Y}-\sum_{t'=1}^{t}u_{i_{t'},j_{t'}}\mathbf{B}_{i_{t'},j_{t'}}\quad,\qquad(1)$$

则寻找最优的 $\mathbf{u}=(u_{i_1,j_1},\cdots,u_{i_t,j_t})^T$ 以求 $\mathbf{R}$ 的最小范数，等价于一个最小二乘问题[13]：

$$\hat{\mathbf{u}}=\mathbf{H}^{-1}\mathbf{f}\quad。\qquad(2)$$

令 $\left\langle\mathbf{B}_{i_{t'},j_{t'}},\mathbf{B}_{i_{s'},j_{s'}}\right\rangle=\left\langle\mathbf{a}_{i_{t'}},\mathbf{a}_{i_{s'}}\right\rangle\left\langle\mathbf{a}_{j_{t'}},\mathbf{a}_{j_{s'}}\right\rangle$，

则：

$$\mathbf{H}=\begin{pmatrix}\left\langle\mathbf{B}_{i_1,j_1},\mathbf{B}_{i_1,j_1}\right\rangle & \cdots & \left\langle\mathbf{B}_{i_1,j_1},\mathbf{B}_{i_t,j_t}\right\rangle\\ \vdots & \vdots & \vdots\\ \left\langle\mathbf{B}_{i_t,j_t},\mathbf{B}_{i_1,j_1}\right\rangle & \cdots & \left\langle\mathbf{B}_{i_t,j_t},\mathbf{B}_{i_t,j_t}\right\rangle\end{pmatrix},\qquad(3)$$

并有：

$$\mathbf{f}=\left(\left\langle\mathbf{Y},\mathbf{B}_{i_1,j_1}\right\rangle,\cdots,\left\langle\mathbf{Y},\mathbf{B}_{i_t,j_t}\right\rangle\right)^T\quad。\qquad(4)$$

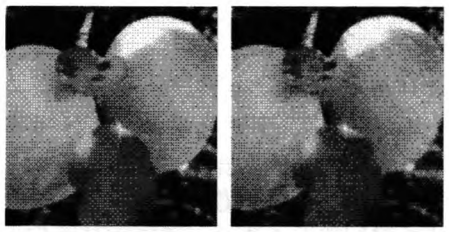
计算 $\mathbf{H}$ 和 $\mathbf{f}$ 的复杂度分别是 $O(mt^2)$ 和 $O(tm^2)$，计算 $\mathbf{H}^{-1}$ 的复杂度是 $O(t^3)$，且计算 $\mathbf{H}^{-1}\mathbf{f}$ 的复杂度为 $O(t^2)$，该部分矩阵求逆及矩阵向量相乘均可并行执行。（4）更新残差：该部分主要是矩阵向量相乘，其复杂度依赖于 t，而 $t\le k\ll n^2$，因此，它的计算时间只占总时间很少一部分。

根据以上分析，当 k 较小时，2D-OMP 算法的复杂度主要由投影部分决定，其复杂度为 $O(mn^2)$。当 k 增大时，更新权重中矩阵求逆的复杂度迅速增加，成为影响算法执行效率的主要因素。因此，加速矩阵求逆和投影是提高整个算法的关键。

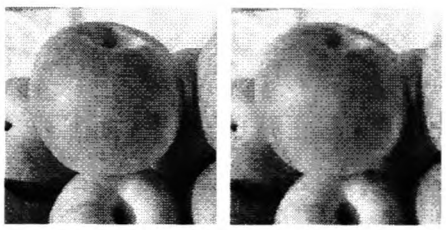
采用 2D-OMP 算法在 GPU 平台上从观测数据中对苹果图像进行并行快速重构：首先在 GPU 上给变量分配内存空间，输入参数包括 $m\times m$ 的采样矩阵 $\mathbf{Y}$，$m\times n$ 的测量矩阵 $\mathbf{A}$ 及其转制 $\mathbf{A}^T$，输出 $\mathbf{Z}$ 是一个 $n\times n$ 的矩阵。在算法的核函数执行之前，$\mathbf{Y}$ 需从 CPU 中拷贝至 GPU，$\mathbf{A}$ 和 $\mathbf{A}^T$ 是只读参数，因此可将其放入 GPU 的纹理存储器以提高读速度。2D-OMP 算法执行 k 次迭代，在每次迭代中，首先计算投影 $\mathbf{A}^T\mathbf{RA}./\mathbf{P}$，使用两个核函数分别计算两次矩阵相乘同时完成点除运算；第二，调用 CUBLAS 库函数求矩阵最大值以获得最匹配原子；第三，调用四个核函数求解矩阵逆；最后，更新残差中包括矩阵向量相乘运算，调用一个核函数可完成其并行计算。待算法计算完后，恢复矩阵 $\mathbf{Z}$ 再从 GPU 中拷贝至 CPU。

3 实验结果

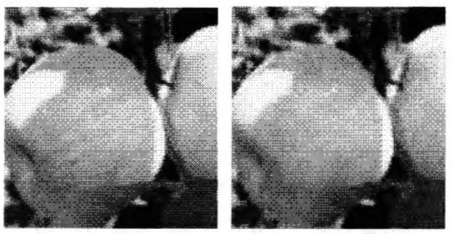
实验测试在英特尔 3.07GHz 的 i7 核 CPU 上进行，CPU 串行 C 代码使用 O_2 级优化。GPU 为 NVIDIA 公司具有 CUDA 性能为 2.0 的 GTX480，有 15 个多重流处理器（SMs），每个多重流处理器包括 32 个流处理器（SPs），一共有 480 个处理核。每个 SM 有 32KB 的寄存器和 48KB 的共享存储器，可提供高速的数据访问。此外，还有 64KB 的常数存储器和纹理存储器。



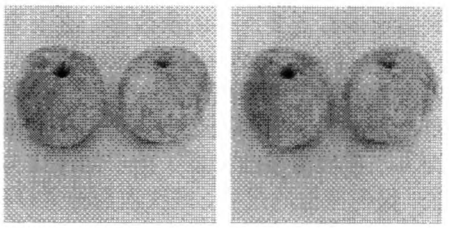
（a）左边为实例 1 原始图，右边为实例 1 重构图



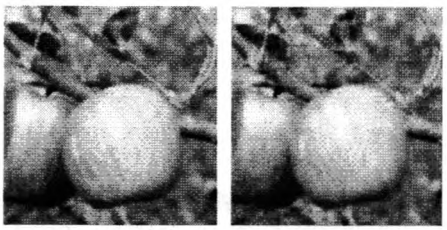
（b）左边为实例 2 原始图，右边为实例 2 重构图



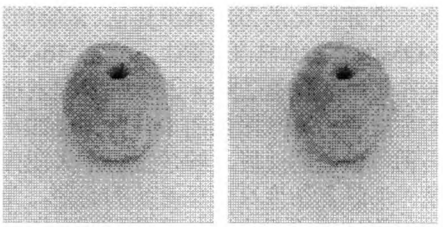
（c）左边为实例 3 原始图，右边为实例 3 重构图



（d）左边为实例 4 原始图，右边为实例 4 重构图



（e）左边为实例 5 原始图，右边为实例 5 重构图



（f）左边为实例 6 原始图，右边为实例 6 重构图

图 2 采样率 $m/n=0.5$ 时，苹果图像重构效果

采用图 2 所示的 6 幅不同内容的 256×256 BMP 格式的苹果灰度图像进行测试，变换矩阵采用多贝西小波变换矩阵，测量矩阵均采用 128×256（$n=256, m=128$）的随机高斯测量矩阵，即采样率 $m/n=0.5$ 的情况下进行采样（采样矩阵大小为 128×128，其采样的数据量为原图的 1/4），采用 2D-OMP 重构算法分别在 CPU 和 GPU 上进行重构测试，其执行时间如表 1 所示（其执行时间均为多次平均时间）。

表 1 苹果图像重构在 CPU 和 GPU 上执行效率对比

图像	迭代次数	CPU 执行时间（ms）	GPU 执行时间（ms）	加速比	PSNR
实例 1	1494	84954	4919	17.2	31.32
实例 2	1555	88101	5262	16.7	30.77
实例 3	1750	105924	6660	15.9	29.29
实例 4	628	26847	920	29.1	36.81
实例 5	1680	99370	6063	16.3	29.08
实例 6	512	22175	634	34.9	38.63

试验结果表明，不同内容的苹果图像达到所给定的误差所需迭代的次数 k 不同（当 R 的均方误差 MSE<0.03 时停止迭代），在室外自然条件下的苹果图像（实例 1，实例 2，实例 3，实例 5）背景较复杂，因此达到给定误差需要迭代的次数多，并且重构图像的峰值信噪比（Peak signal-noise ration, PSNR）较低，而在室内人为环境下的苹果图像（实例 4，实例 6），背景简单，则需要迭代的次数少，且重构图像的峰值信噪比较大，重构图像效果好。由于加速比与图像的内容无关，仅与图像的尺寸、迭代次数及采样率有关。实验中苹果图像的大小及采样率相同，因此加速比只和迭代次数相关。表 1 的结果表明，并行 2D-OMP 算法对于图 2 不同的苹果图像在 GPU 上可获得约 16-35 倍的加速。

4 结论

本文将压缩感知应用到苹果图像的采集上以减少传统方法采集图像的数据量，并提出一个并行快速的苹果重构方法。在 GPU 通用计算平台上利用 CUDA 技术设计一个并行化的 2D-OMP 算法。试验结果表明，并行 2D-OMP 重构算法在 GPU 上获得 16-35 倍的加速比，能在数秒内恢复 256×256 的苹果灰度图像，且需要的数据量为原来图像的 1/4。并行化的重构算法大大提高图像恢复的速度，解决目前苹果图像使用压缩感知技术面临复杂度高而耗时长的问题，为苹果生长过程远程监控系统采集图像及苹果质量快速检测与分类等应用提供条件。

参考文献

[1] Alexander Hornberg. Handbook of Machine Vision [M]. Published by Wiley-VCH. 2006.

[2] 司永胜，乔军，刘刚，等. 基于机器视觉的苹果识别和形状特征提取[J]. 农业机械学报，2009，40（8）：161-165.

[3] 戴琼海，付长军，季向阳. 压缩感知研究[J]. 计算机学报，2011，34（3）：425-434.

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

[5] 邵文泽，韦志辉. 压缩感知基本理论：回顾与展望[J]. 中国图象图形学报. 2012，17（1）：1-12.

[6] 蔡骋，张明，朱俊平. 基于压缩感知理论的杂草种子分类识别[J]. 中国科学：信息科学，2010，S1（40）：160-172.

[7] 杨小青. 基于压缩传感的水果分级系统的研究[D]. 西安：陕西科技大学. 硕士学位论文. 2012.

[8] Tze-Yui Ho, Ping-Man Lam, Chi-Sing Leung. Parallelization of cellular neural networks on GPU[J]. Pattern Recognition. 2008, 41(8): 2684-2692.

[9] 吴恩华. 图形处理器用于通用计算的技术、现状及其挑战[J]. 软件学报，2004，15（10）：1493-1504.

[10] J. Sanders, E. Kandrot. CUDA by Example: An Introduction to General-Purpose GPU Programming, 1st ed. Addison-Wesley Professional. 2010.

[11] 吴恩华，柳有权. 基于图形处理器(GPU)的通用计算[J]. 计算机辅助设计与图形学学报，2004，16(5)：601-612.

[12] 李树涛，魏丹. 压缩传感综述[J]. 自动化学报. 2009，35（11）：1369-1377.

[13] FANG Yong, WU JiaJi, HUANG BorMin. 2D sparse signal recovery via 2D orthogonal matching pursuit[J]. Science China: information science, 2012, 55(4): 889-897.

[14] David B. Kirk, Wen-mei W. Hwu, Programming Massively Parallel Processors: A Hands-on Approach [M]. Published by Elsevier. 2010.

[15] V. Boyer, D.El Baz, M.Elkihel. Solving knapsack problems on GPU[J]. Computers & Operations Research, 2012, 39: 42-4