第 8 卷 第 6 期
2019 年 11 月
集 成 技 术
JOURNAL OF INTEGRATION TECHNOLOGY
Vol. 8 No. 6
Nov. 2019

引文格式：
蔡访, 丘志浪, 苏适, 等. 基于图形处理器加速 Wave-CAIPI 重建的改进共轭梯度法 [J]. 集成技术, 2019, 8(6): 11-20.
Cai F, Qiu ZL, Su S, et al. A graphics processing unit-based modified conjugate gradient method for accelerating wave-CAIPI reconstruction [J]. Journal of Integration Technology, 2019, 8(6): 11-20.

基于图形处理器加速 Wave-CAIPI 重建的改进共轭梯度法

蔡 访[1,2] 丘志浪[1,2] 苏 适[1] 朱燕杰[1] 王海峰[1] 梁 栋[1]

[1](中国科学院深圳先进技术研究院 深圳 518055)
[2](中国科学院大学 北京 100049)

摘 要 Wave-CAIPI 是一种利用多通道线圈和 k 空间螺旋轨迹采样来加速磁共振成像的新 3D 成像方法。然而，Wave-CAIPI 采集的 3D 数据对于重建计算是巨大的。为了加速重建过程，该文使用基于图形处理器改进的共轭梯度算法实现了 Wave-CAIPI 重建，减少了重建时间。水模数据集和体内人脑数据集的实验表明，基于图形处理器的 Wave-CAIPI 重建可以获得与传统基于中央处理器的 Wave-CAIPI 重建类似的图像结果，且重建效率显著提升。

关键词 图形处理器；Wave-CAIPI；磁共振成像；共轭梯度法
中图分类号 R 455.2 TP 391 **文献标志码** A doi: 10.12146/j.issn.2095-3135.20190618001

A Graphics Processing Unit-Based Modified Conjugate Gradient Method for Accelerating Wave-CAIPI Reconstruction

CAI Fang[1,2] QIU Zhilang[1,2] SU Shi[1] ZHU Yanjie[1] WANG Haifeng[1] LIANG Dong[1]

[1](*Shenzhen Institutes of Advanced Technology*, *Chinese Academy of Sciences*, *Shenzhen* 518055, *China*)
[2](*University of Chinese Academy of Sciences*, *Beijing* 100049, *China*)

Abstract Wave-CAIPI is a novel 3D imaging method with multiple-channel coils and corkscrew trajectories in k-space to speed up magnetic resonance imaging acquisition. However, the 3D data acquisitions of Wave-CAIPI are usually time consuming. In order to accelerate the reconstruction procedure, we realized a Wave-CAIPI reconstruction method using a modified GPU-based conjugate gradient algorithm to reduce time cost of the image

收稿日期：2019-06-18 **修回日期**：2019-08-21
基金项目：国家自然科学基金项目(61871373、81729003)；广东省自然科学基金项目(2018A0303130132)；深圳市发改委双链项目([2018]256)；深圳三名工程项目(SZSM201812005)
作者简介：蔡访(共同第一作者)，硕士研究生，研究方向为磁共振重建计算、GPU 并行计算、硬件加速；丘志浪(共同第一作者)，博士研究生，研究方向为磁共振成像方法与序列；苏适，硕士，助理研究员，研究方向为磁共振成像序列；朱燕杰，博士，副研究员，研究方向为心脏磁共振成像技术与快速重建算法；王海峰(通讯作者)，博士，副研究员，硕士研究生导师，研究方向为磁共振成像与信号处理，E-mail：hf.wang1@siat.ac.cn；梁栋(通讯作者)，博士，研究员，博士研究生导师，研究方向为图像分析、人工智能，E-mail：dong.liang@siat.ac.cn。

reconstructions. The experiments of phantom and *in vivo* human brain show that the proposed GPU-based Wave-CAIPI reconstruction can achieve similar imaging results with less time cost, comparing to the conventional CPU-based Wave-CAIPI reconstruction.

Keywords graphics processing unit; Wave-CAIPI; magnetic resonance imaging; conjugate gradient

1 引 言

磁共振成像(Magnetic Resonance Imaging，MRI)是现代医学中一项必不可少的技术，具有对比度高、多参数成像、任意方向断层成像、对人体无电离辐射伤害等优点[1]。为了减少磁共振成像扫描时间，研究者们提出了许多加速磁共振成像技术。并行磁共振成像[2-5]是一类可以减少扫描时间的图像重建方法。其中，敏感度编码(Sensitivity Encoding，SENSE)[4]和GRAPPA[5]等是目前在临床实践中最常用的并行成像技术[6]。Wave-CAIPI[7]是近年来新提出的一种基于多通道线圈与 k 空间螺旋轨迹来加速采样的并行磁共振重建方法。不同于传统的并行磁共振成像，Wave-CAIPI 结合并扩展了 2D-CAIPI[8]与 BPE(Bunched Phase Encoding)[9]，具有较高的成像图像质量及较快的采集速度等优点。

由于并行磁共振成像从多个接收线圈中获取补充信息来重建 k 空间欠采样数据，大部分并行磁共振成像还需要额外的线圈敏感度分布信息来消除伪影。对计算机来说，临床应用中的 MRI 重建计算变得越来越复杂。医生和研究者需要一个快速的磁共振重建处理过程，以便能更快地看到病人图像。因此，一些研究者将注意力投入到图形处理器(Graphics Processing Unit，GPU)加速计算。自 2005 年以来，不断有关于 GPU、MRI 和重建的成果发表。2007 年 NVIDIA 公司的 CUDA (Compute Unified Device Architecture)发布后[10]，基于 GPU 加速的 MRI 重建变得更加适用。Hansen 等[11]专注于 SENSE[4]重建问题，对每个混叠点的数据进行并行重建，并利用乔里斯基分解法对系统矩阵进行分解来求逆，完成了笛卡尔 SENSE 和 *k-t* SENSE 的 GPU 实现。Inam 等[12]基于大规模并行 GPU 架构提出了一种加速自校正 GRAPPA[5]网格化算符的方法。大多数经典 MRI 重建的 GPU 加速算法得到了很好的研究，并已在 GPU 上实现[13]。2018 年以来，部分研究者专注于某些非均匀采样或非笛卡尔采样的磁共振重建计算，并利用基于 GPU 的非均匀快速傅里叶变换算法来实现加速计算[14-15]。部分研究者专注于 SENSE 重建中的编码矩阵，如 Qazi 等[16]利用雅克比奇异值分解来求解系统矩阵的逆；Ullah 等[17]使用 QR 分解来求解系统矩阵的逆，并将整个矩阵分解计算分散到 GPU 的多个线程上来完成加速。然而，对于 Wave-CAIPI 这样的新 MRI 方法，其重建计算过程相比 SENSE 等经典并行 MRI 方法更加复杂，尚未有关于 GPU 加速实现的研究报道。

本文基于 NVIDIA CUDA 平台，实现了一种基于 GPU 改进的共轭梯度(Conjugate Gradient，CG)法[18]对 Wave-CAIPI 进行加速重建。本文对两个图像数据进行重建：一个是在中国上海联影 3T 磁共振扫描仪上采集的三维水模数据集，一个在德国西门子 7T 磁共振扫描仪上采集的三维人脑数据集。重建过程中需要包括对应的多个线圈敏感度分布信息及 Wave-CAIPI 采集过程中额外施加的正弦波浪(wave)梯度信息来重建出最终的图像。由于整个重建过程中涉及大量的矩阵操作，且不同线圈采集的数据在重建计算中行为相似，故利用重建中潜在的并行性能实现对重

建计算的加速。本文将以重建 3 层图像为例，先在中央处理器(Central Processing Unit，CPU)上使用共轭梯度最小二乘(Conjugate Gradient Least Squares，CGLS)算法[18]重建 Wave-CAIPI；然后，基于 CUDA，在 GPU 上实现并行的 Wave-CAIPI 重建改进的 CG 算法。本文通过对比重建计算的不同实现，验证了所提出的方法能有效地对重建计算过程进行加速。本文所提出的方法具有以下优点：(1)GPU 上实现的算法所重建出的图像质量与 CPU 上实现的算法重建出的图像质量相差不大；(2)由于采集数据之间潜在的无关性，GPU 加速重建的算法很容易推广到重建整个三维图像或其他欠采样倍数不同、数据量更大的图像数据上。

本文内容组织如下：第 2 小节阐述本文采用的方法原理；第 3 小节分别在 CPU、GPU 上实现本文提出的方法；第 4 小节对所提方法的性能作进一步地分析实验；第 5 小节讨论与分析；第 6 小节总结。

2　方　法

2.1　Wave-CAIPI 重建

对于相位编码和选层编码的直线三维磁共振成像，其采样轨迹如图 1 所示。在 k 空间固定 k_y 和 k_z 处接收到的信号可用 k 空间记号表示为：

$$s(t)=\int_{x,y,z} m(x,y,z)e^{-i2\pi[k_x(t)x+k_y y+k_z z]}\mathrm{d}x\mathrm{d}y\mathrm{d}z \quad (1)$$

其中，$m(x, y, z)$是潜在的图像。在每个读出行期间，Wave-CAIPI 在梯度的 y 轴、z 轴添加额外的正弦波浪梯度 g_y、g_z(g_y、g_z 相位差为 π/2)，其采样轨迹如图 2 所示，则接收到的信号为：

$$s(t)=\int_{x,y,z} m(x,y,z)e^{-i2\pi[k_x(t)x+k_y y+k_z z]}\exp\{-i\gamma\int_0^t[g_y(\tau)y+g_z(\tau)z]\mathrm{d}\tau\}\mathrm{d}x\mathrm{d}y\mathrm{d}z \quad (2)$$

上式经推导离散化后，可简写为：

$$wave[x,y,z]=F_x^{-1}Psf[k,y,z](F_x m[x,y,z]) \quad (3)$$

其中，$wave[x,y,z]$是施加波浪梯度后得到的图像；F_x 是沿 x 轴作离散傅里叶变换；F_x^{-1} 是沿 x 轴作离散傅里叶逆变换；k 是读出行上的数据点在 k 空间的坐标；$Psf[k,y,z]$点扩散函数(Point Spread Function，PSF)简单解释了波浪梯度的效应：潜在图像 $m(x, y, z)$的每个读出行与相对应的 PSF 进行卷积生成 wave 图像。

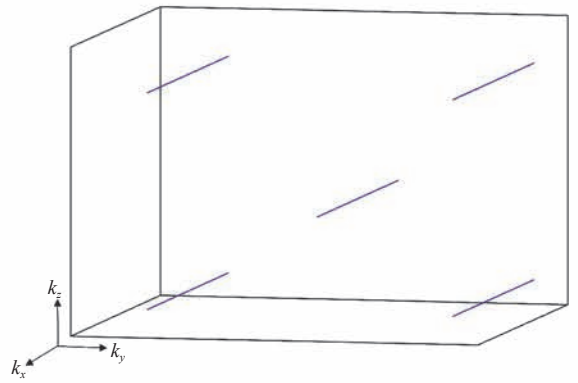


图 1　三维笛卡尔成像的 k 空间采样轨迹

Fig. 1　The k-space trajectory for rectilinear 3D imaging

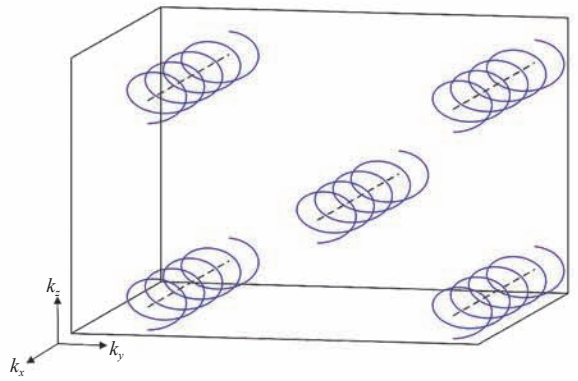


图 2　Wave-CAIPI 成像的 k 空间采样轨迹

Fig. 2　The k-space trajectory for Wave-CAIPI imaging

对于 R 倍加速的相位编码和选层编码，来自 R 个空间位置的图像读出行混叠在一起，利用线圈敏感度分布提供的空间编码信息可解混叠。这种方法可以拓展到 wave 采样模式引起的混叠。为简单起见，考虑只在相位编码(k_y)方向进行 2 倍加速的 Wave-CAIPI，该情况下相隔半个视距的图像位置会彼此混叠。用 $wave[y_1]$和 $wave[y_2]$分别表示将要混叠在一起的 y_1 和 y_2 处测到的信号，则潜在图像 $m[y_1]$和 $m[y_2]$与图像读

出行的关系为：$wave[y_1]=F^{-1}Psf[y_1]Fm[y_1]$，$wave[y_2]=F^{-1}Psf[y_2]Fm[y_2]$。所获取数据与未知图像的关系的前向模型为：

$$\left\{F^{-1}Psf[y_1]F \quad F^{-1}Psf[y_2]F\right\}\begin{Bmatrix}m[y_1]\\m[y_2]\end{Bmatrix}=wave \tag{4}$$

其中，$wave=wave[y_1]+wave[y_2]$，为欠采样导致的混叠图像。

对于有 n 个接收线圈的并行 Wave-CAIPI 成像，每个线圈的敏感度分布为 C_i，则该系统变为：

$$\begin{Bmatrix}F^{-1}Psf[y_1]FC_1[y_1] & F^{-1}Psf[y_2]FC_1[y_2]\\ \vdots & \vdots\\ F^{-1}Psf[y_1]FC_n[y_1] & F^{-1}Psf[y_2]FC_n[y_2]\end{Bmatrix}\begin{Bmatrix}m[y_1]\\m[y_2]\end{Bmatrix}=\begin{bmatrix}wave_1\\ \vdots\\ wave_n\end{bmatrix} \tag{5}$$

公式(5)的解从减半的视距线圈图像 $wave_i$ 中重建出潜在的图像行。本文在此处给出 Wave-CAIPI 重建的主要原理，公式(1)～(5)及其推导过程在 Bilgic 等[7]工作中有详细描述。

公式(5)中的编码矩阵一般不直接显式地出现在求解过程中。本文利用共轭梯度法求解公式(5)时，将编码矩阵看作一个函数 $A(\cdot)$，其实现算法(算法 1)如表 1 所示。该算法描述了 Wave-CAIPI 的重建问题。

2.2 共轭梯度法

CG 算法由 Hestenes 和 Stiefel[19]提出，是一种迭代地求解对称正定线性方程组的算法。CG 算法在求解 l 维方程组中，持续地找出和消去 l 个正交的误差成分。通过使用两两正交的余项确定的方向，可以降低算法的复杂度。CG 算法因其存储量少、快速收敛、二次终止性等优点，现已广泛地应用于实际问题中。传统的 CG 算法(算法 2)如表 2 所示。

Wave-CAIPI 的接收线圈数量一般比欠采样倍数要多，其编码矩阵是一个“长高”矩阵，

表 1 Wave-CAIPI 编码矩阵的函数实现算法

Table 1 Function implementation algorithm of Wave-CAIPI coding matrix

算法 1 Wave-CAIPI 编码矩阵的函数实现算法

输入：向量 $\boldsymbol{x}$、变体参数 *flag*、结构参数 *p*

输出：向量 $\boldsymbol{y}$

（1）若 *flag*=1，则：

① 将向量 $\boldsymbol{x}$ 重复 *p.n* 倍，并转化成大小为 $p.il\times p.Ry\times p.Rz\times p.n$ 的矩阵 $\boldsymbol{I}$

② $\boldsymbol{I}\leftarrow \boldsymbol{I}.*(\boldsymbol{p.rcv})$

③ 沿 $\boldsymbol{I}$ 的第一个维度分别在两侧填充 $(p.pl-p.il)/2$ 个 0

④ $\boldsymbol{I}\leftarrow ffts\{fft[ffts(\boldsymbol{I})]\}$

⑤ $\boldsymbol{I}\leftarrow \boldsymbol{I}.*(\boldsymbol{p.psfs})$

⑥ $\boldsymbol{I}\leftarrow ffts\{ifft[ffts(\boldsymbol{I})]\}$

⑦ 依次沿 $\boldsymbol{I}$ 的第二维和第三维求和，并将求和的结果转化为向量 $\boldsymbol{y}$ 输出

（2）若 *flag*=0，则：

① 将向量 $\boldsymbol{x}$ 转化成大小为 $p.pl\times1\times1\times p.n$ 的矩阵

② $\boldsymbol{I}\leftarrow ffts\{fft[ffts(\boldsymbol{I})]\}$

③ 将 $\boldsymbol{I}$ 沿第二维和第三维重复填充，变成大小为 $p.pl\times p.Ry\times p.Rz\times p.n$ 的矩阵

④ $\boldsymbol{I}\leftarrow \boldsymbol{I}.*conj(\boldsymbol{p.psfs})$

⑤ $\boldsymbol{I}\leftarrow ffts\{ifft[ffts(\boldsymbol{I})]\}$

⑥ 沿 $\boldsymbol{I}$ 的第一个维度裁剪出 *p.il* 长度的中间部分

⑦ $\boldsymbol{I}\leftarrow \boldsymbol{I}.*conj(\boldsymbol{p.rcv})$

⑧ 沿 $\boldsymbol{I}$ 的第四个维度求和，并将求和的结果转化为向量 $\boldsymbol{y}$ 输出

注：*p.n* 为接收线圈数量；*p.il* 为要重建图像的第一维长度；*p.pl* 为 Wave-CAIPI 采集数据的第一维长度；*p.Ry* 为相位编码方向的欠采样倍数；*p.Rz* 为选层编码方向的欠采样倍数；$\boldsymbol{p.rcv}$ 为线圈敏感度分布矩阵；$\boldsymbol{p.psfs}$ 为波浪梯度效应矩阵；*conj*()为共轭函数；*fft*()为沿矩阵的第一个维度作快速傅里叶变换；*ifft*()为沿矩阵的第一个维度作快速傅里叶逆变换；*ffts*()为沿矩阵的一个维度将矩阵中的元素顺移半个第一维长度的距离；.* 表示矩阵与矩阵之间对应元素相乘

无法直接用 CG 算法进行求解。Saunders[18]实现的 CGLS 算法，是一种可用来求解非对称线性方程组和最小二乘问题的 CG 算法。本文基于此，在 GPU 上实现一种改进的 CG 算法，用来加速 Wave-CAIPI 重建。本文实现的改进 CG 算法(算法 3)处理单读出行数据的原理如表 3 所示。

表 3 中的步骤(3)～(13)全部在 GPU 上完成

表 2　传统 CG 算法

Table 2　Traditional CG algorithm

算法 2　传统 CG 算法

输入：矩阵 $\boldsymbol{A}$、向量 $\boldsymbol{b}$

输出：向量 $\boldsymbol{x}$

(1)　$\boldsymbol{x}_0 \leftarrow$ 初始估计；$k \leftarrow 0$

(2)　$\boldsymbol{r}_0 \leftarrow \boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}_0$；$\boldsymbol{d}_0 \leftarrow \boldsymbol{r}_0$

(3)　$\alpha_k \leftarrow \dfrac{\boldsymbol{r}_k^{\mathrm{T}}\boldsymbol{r}_k}{\boldsymbol{d}_k^{\mathrm{T}}\boldsymbol{A}\boldsymbol{d}_k}$

(4)　$\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k+\alpha_k\boldsymbol{d}_k$

(5)　$\boldsymbol{r}_{k+1} \leftarrow \boldsymbol{r}_k-\alpha_k\boldsymbol{A}\boldsymbol{d}_k$

(6)　$\beta_k \leftarrow \dfrac{\boldsymbol{r}_{k+1}^{\mathrm{T}}\boldsymbol{r}_{k+1}}{\boldsymbol{r}_k^{\mathrm{T}}\boldsymbol{r}_k}$

(7)　$\boldsymbol{d}_{k+1} \leftarrow \boldsymbol{r}_{k+1}+\beta_k\boldsymbol{d}_k$

(8)　$k \leftarrow k+1$

(9)　若 $\boldsymbol{r}_k=0$ 或 $k=mx$，结束；否则转到第(3)步

注：mx 为设定的最大迭代次数；$\boldsymbol{r}_k$ 为近似解 $\boldsymbol{x}_k$ 的余项；$\boldsymbol{d}_k$ 表示用于更新 $\boldsymbol{x}_k$ 得到改进的 $\boldsymbol{x}_{k+1}$ 时所使用的新搜索方向；$\boldsymbol{x}_k$ 更新系数 $\boldsymbol{\alpha}_k$ 使新余项 $\boldsymbol{r}_{k+1}$ 与方向 $\boldsymbol{d}_k$ 正交；$\boldsymbol{d}_k$ 更新系数 $\boldsymbol{\beta}_k$ 使所有方向 $\boldsymbol{d}$ 两两共轭；上标 T 表示转置

表 3　改进的 CG 算法

Table 3　Modified CG algorithm

算法 3　改进的 CG 算法

输入：Wave-CAIPI 编码矩阵对应的函数 $A(\cdot)$、向量 $\boldsymbol{b}$

输出：向量 $\boldsymbol{x}$

(1)　为向量 $\boldsymbol{b}$、$\boldsymbol{x}$、$\boldsymbol{s}$、$\boldsymbol{r}$、$\boldsymbol{p}$、$\boldsymbol{q}$ 及函数参数分配 GPU 内存

(2)　将 CPU 内存中的向量 $\boldsymbol{b}$ 复制到 GPU 对应的内存上

(3)　$\boldsymbol{s} \leftarrow A(\boldsymbol{b},0)$；$\boldsymbol{x} \leftarrow 0$；$\boldsymbol{r} \leftarrow \boldsymbol{b}$

(4)　$k \leftarrow 0$；$\boldsymbol{p} \leftarrow \boldsymbol{s}$；$ns0 \leftarrow \|\boldsymbol{s}\|$；$\gamma \leftarrow ns0^2$

(5)　$\boldsymbol{q} \leftarrow A(\boldsymbol{p},1)$

(6)　$\alpha \leftarrow \gamma / (\|\boldsymbol{q}\|^2+\|\boldsymbol{p}\|^2)$

(7)　$\boldsymbol{x} \leftarrow \boldsymbol{x}+\alpha\boldsymbol{p}$；$\boldsymbol{r} \leftarrow \boldsymbol{r}-\alpha\boldsymbol{q}$

(8)　$\boldsymbol{s} \leftarrow A(\boldsymbol{r},0)$

(9)　$\gamma_1 \leftarrow \gamma$

(10)　$\gamma \leftarrow \|\boldsymbol{s}\|^2$

(11)　$\boldsymbol{p} \leftarrow \boldsymbol{s}+\dfrac{\gamma}{\gamma_1}\boldsymbol{p}$

(12)　$k \leftarrow k+1$

(13)　若 $k>mit$ 或 $\|\boldsymbol{s}\| \leqslant ns0*tol$ 或 $\|\boldsymbol{x}\|*tol \geqslant 1$，则转到第(14)步，否则转到第(5)步

(14)　将 GPU 内存中的 $\boldsymbol{x}$ 复制到 CPU 内存中

注：mit 为设定的最大迭代次数，在本文中设定为 200；tol 为设定的允许误差，在本文中水模重建实验设定为 1×10^{-4}，全脑重建实验设定为 1×10^{-3}；$\boldsymbol{\gamma}$ 表示 $\boldsymbol{s}$ 的二范数，用于更新 $\boldsymbol{p}$；$ns0$ 表示 $\boldsymbol{s}$ 初始的二范数

计算，以完成加速。但这些步骤仅描述了单读出行数据对应的行为。对于多读出行数据，上述改进的 CG 算法中相应的标量、向量、矩阵将额外增加一个维度。对应的矩阵向量操作，将是一个多数据的操作。Wave-CAIPI 的编码矩阵隐式地表达成函数形式后，改进的 CG 算法能对其重建问题进行迭代求解。

3　加速 Wave-CAIPI 重建

3.1　数据与材料

水模成像数据采集自中国上海联影 3T 磁共振扫描仪（型号：uMR790），扫描参数如下：视野为 192 mm×192 mm×192 mm；像素大小为 1 mm×1 mm×1 mm；翻转角为 9°；带宽为 70 赫兹/像素；接收线圈为 32 通道联影头线圈；重复时间为 26 ms；回波时间为 13 ms；wave 相对场强为 1；7 个正弦波浪周期/读出，使用 k_y 和 k_z 方向都是 3 倍加速的 Wave-CAIPI 采样采集梯度回波数据。由于波浪梯度效应在 k_x 方向上进行了 2 倍过采样，经预处理后得到的数据矩阵大小为 384×66×64×32，该数据矩阵的维度分别为 k_x、k_y、k_z 和线圈维度。

健康个体的全脑成像数据来自 Bilgic 等[7]公开的数据，在德国西门子 7T 磁共振扫描仪上扫描，扫描参数如下：视野为 240 mm×240 mm×120 mm；像素大小为 1 mm×1 mm×1 mm；重复时间为 40 ms；回波时间为 20 ms；带宽为 70 赫兹/像素；接收线圈为 32 通道西门子头线圈；最大波浪梯度场幅值为 6 mT/m；最大转换速率为 50 mT/(m·ms)；7 个正弦波浪周期/读出，使用 k_y 和 k_z 方向都是 3 倍加速的 Wave-CAIPI 采样采集梯度回波数据，每次采集 2.3 min。另外，在重复时间 12 ms 的条件下，采集低分辨率（3 mm 各向同性的像素大小）梯度回波数据，以此估计线圈敏感度分布。由于波浪梯度效应在 k_x 方向进行了

6 倍过采样，经预处理后得到的数据矩阵大小为 1 160×80×40×31，该数据矩阵与水模数据矩阵各维度的含义相同。

水模成像和全脑成像的多线圈三维数据都沿 k_z 方向相互独立，k_z 方向上的各数据重建过程相同。因本文只选取 k_z 方向上的一份数据来进行加速重建，并不是对整个三维数据进行重建，故在本文的实验中显存充足。具体地，本文选取水模成像数据中的 384×66×1×32 数据和全脑成像数据中的 1 160×80×1×31 数据来分别进行重建。两份数据最终重建出来的结果图像分别为 3 幅 192×198 的水模图像和 3 幅 240×240 的脑图像。

本文实验使用的计算工作站指标是 CPU 为：Intel Core i7-7700 CPU @ 3.60 GHz，安装内存为 16 GB；GPU 为 NVIDIA GeForce GTX 1080 显卡，1.73 GHz，全局内存为 8 GB。另外，实验使用的软件是 Visual Studio 2015 和 CUDA 8.0，操作系统为 Windows 7，并使用 C＋＋语言完成重建。

3.2 重建计算

为了在 CPU 上进行重建，需每次取 D_x×1×1×D_c 的数据及其对应的线圈敏感度分布数据和 PSF 数据来进行重建，共循环 D_y 次以完成整个数据重建。其中，D_x 为 k_x 维度的数据长度；D_c 为线圈维度的数据长度；D_y 为 k_y 维度的数据长度。重建过程中，数据类型选择单精度浮点复数类型，使用 CGLS 算法迭代求解 Wave-CAIPI 重建，使用 FFTW3 库来完成快速傅里叶变换。

为了使用 GPU 对重建过程进行加速，需利用 GPU 的多核、多线程等特点。本文基于 CUDA 平台，通过调用 GPU 核函数及 CUDA 库函数，实现重建计算的并行化。重建过程中，数据类型选择单精度浮点复数类型，并使用 2.2 小节中改进的 CG 算法迭代求解 Wave-CAIPI 重建。整个重建过程的控制流仍在 CPU 上执行，而数据的计算则在 GPU 上执行。首先，对于大小为 D_x×1×1×D_c 的数据，其重建中的简单计算（如改进 CG 算法中的 $\boldsymbol{r} \leftarrow \boldsymbol{b}$ 等），可通过编写核函数，并在一个网格（grid）的多个线程上并行执行核函数来完成加速，其他计算（如二范数的计算）可通过调用 cublas 库函数来完成加速。其次，对于多个数据的快速傅里叶变换，调用 cufft 库中的 cuffPlanMany 及 cufftExecC2C 来实现。可以观察到，其他 D_y－1 个读出行的数据与上述数据的操作一致，只是重建收敛的速度不同。基于此，为了进一步提高并行度，将程序调整为多数据流来执行。此时，如本文 2.2 小节中所述，算法中相应的参数增加一个维度。相应的各种数据操作为：对于编写的核函数，启动时需增加相应的线程数；对于 cublas 库函数，利用动态并行特性，通过使用静态库，就可先启动 GPU 端 D_y 个线程，再从 GPU 端调用 cublas 库函数；对于 cufft，因 Windows 平台上没有 cufft 静态库，故无法参照 cublas 库的处理方式。本文将 cufftPlanMany 输入参数中的傅里叶变换信号个数设置成原来的 D_y 倍来实现多数据的处理。为了控制 D_y 个读出行不同的收敛行为，本文设置一个大小为 D_y 的标志数组，用来标志对应的读出行是否完成重建。当标志重建完成时，该读出行的重建计算结果不再写入结果内存。基于 GPU 的重建流程如图 3 所示。

利用基于 CPU 和 GPU 的方法重建出的图像及重建时间如图 4、5 所示。由于本文所使用的数据集在 k_z 方向上 3 倍欠采样，从 k_z 方向上选取的一份混叠数据最终重建出的是 3 幅图像。因此，图 4、5 为重建结果是 3 幅图像，且这 3 幅图像在整个三维图像中是等间距的。从图 4 和图 5 可以看到，本文提出的基于 GPU 的方法重建的图像质量与基于 CPU 的方法重建的图像质量相差无几，但基于 GPU 的重建方法在水模图像数据和脑图像数据上分别减少了 90.2% 、88.3% 的

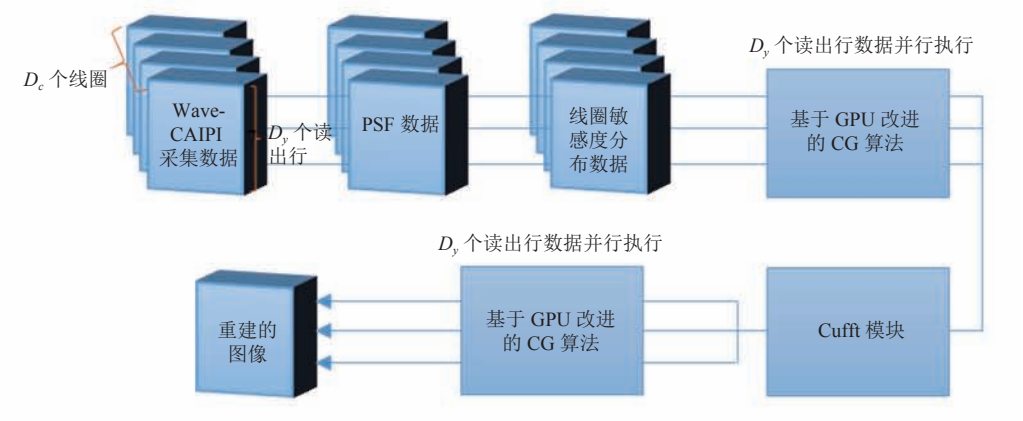
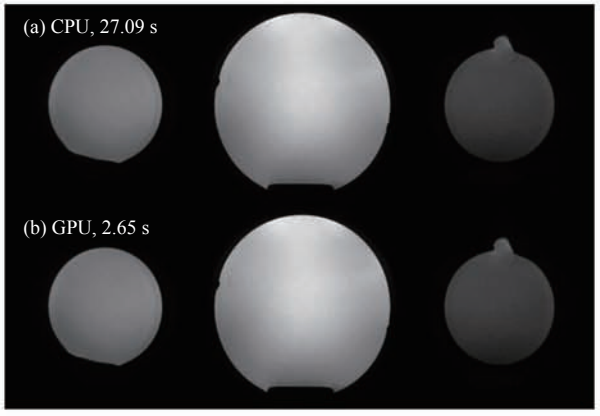


图 3　基于 GPU 的 Wave-CAIPI 并行重建流程

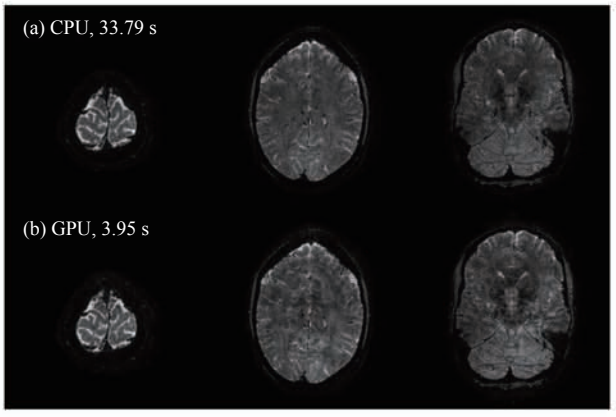
Fig. 3　GPU based parallel reconstruction process for Wave-CAIPI



注：(a) 为基于 CPU 重建的结果，重建时间为 27.09 s；(b) 为基于 GPU 重建的结果，重建时间为 2.65 s

图 4　重建出的水模图像

Fig. 4　Reconstructed phantom images



注：(a) 为基于 CPU 重建的结果，重建时间为 33.79 s；(b) 为基于 GPU 重建的结果，重建时间为 3.95 s

图 5　重建出的脑图像

Fig. 5　Reconstructed brain images

重建计算时间。

4　性能分析

本文提出的基于 GPU 的 Wave-CAIPI 重建方法显著减少了重建时间。然而，考虑到基于 GPU 的方法对于 D_y 个读出行数据在理想情况下是设计成并行计算的，且是对于单个读出行数据的执行，因此该方法也进行了某些计算步骤的并行化。即使由于 GPU 与 CPU 间的内存复制、内核启动、同步以及 GPU 相对较低的时钟频率等因素导致的计算性能下降，该方法相对于基于 CPU 的方法加速倍数也未到达预期。因此，本文在脑图像数据上研究了所提出的基于 GPU 的重建计算方法所需时间与一次输入的读出行数量之间的关系。本文提出的基于 GPU 的重建计算方法只需设定好参数，即可实现指定读出行数量的重建计算。如图 6 所示，纵坐标表示重建计算所需的总时间；横坐标表示一次输入的读出行数，即程序每次循环中计算的数据量。从图 6 可以看出，总重建计算时间随着每次输入的读出行数量的增加而减少。虽然该趋势与预期相符，但计算时间减少的幅度低于预期。特别是当一次输入的读出行数量分别为 20、40、80 时，重建计算的

时间几乎不变。

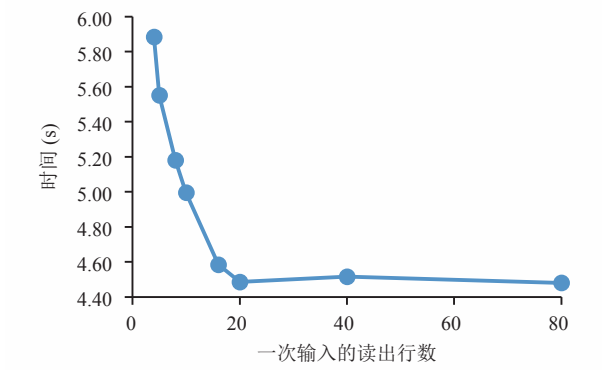


图 6 脑图像重建时间与一次输入的读出行数的关系

Fig. 6 Relationship between reconstruction time of brain images and the number of input readout lines

进一步分析程序发现，内核调用和动态并行的表现是已知的，而对于 cufft 库函数处理多个数据的快速傅里叶变换的行为未知。基于此，本文研究了在整个程序输入为 D_y 个读出行数据时，一次输入到 cufft 模块的读出行数量与重建总时间的关系。如图 7 所示，纵坐标为脑图像数据的重建计算所需时间；横坐标为 cufft 模块的循环次数 (j)，即每次输入 cufft 模块的读出行数量为 D_y/j。从图 7 可以看出，总重建时间先随着 cufft 模块循环次数的增加而减小，在循环次数为 20 时减到最小，此后又随着循环次数的增加而缓慢增加。Cufft 模块对计算性能的影响明显与推测不符。事实上，虽然 cufftPlanMany 提供了批量执行快速傅里叶变换的功能，但数据批量执行的过程不是完全并行化的。本文在研究 cufft 模块循环次数 j 对总重建计算时间的影响时，先调用 cufftPlanMany 对 $80/j$ 个读出行的数据量生成一个计划，生成计划的过程中会分配计算所需的中间缓存；然后，使用这个相同的计划在不同的数据上循环 j 次，并调用 cufftExecC2C 来完成所有数据的快速傅里叶变换。不同数据量生成计划的开销和批量执行傅里叶变换的并行度之间互相影响，导致了如图 7 所示的结果。因此，CUDA 平台中 cufft 模块的多数据快速傅里叶变换的行为是阻碍本文提出的基于 GPU 重建方法的性能进一步提升的关键因素。

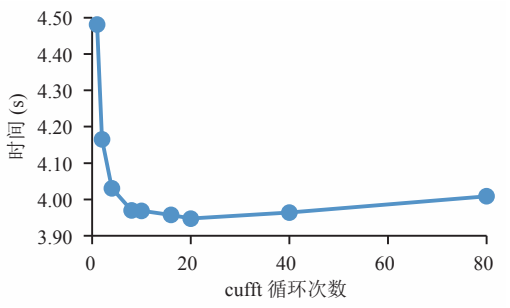


图 7 重建时间与 cufft 循环次数的关系

Fig. 7 Relationship between reconstruction time and number of cufft cycles

5 讨论与分析

本文提出了一种基于 GPU 改进的 CG 算法来完成 Wave-CAIPI 的重建。在该算法流程中，并行地对所有读出行数据进行迭代重建计算。从实验结果可以看出，本文提出的方法能显著减少重建计算的时间；相对于 CPU，GPU 实现在 9 倍欠采样的 Wave-CAIPI 数据上的加速倍数为 8～10 倍。

由于基于 GPU 改进的 CG 算法的并行计算对象是读出行，因此很容易将该算法扩充成一次处理更多读出行的实现，以便完成对多片 k 空间数据进行并行重建。考虑到 GPU 的内存及其他资源有限，甚至可以将该算法扩展成与 GPU 资源相对应的读出行数据量进行并行计算，即一次完成计算的数据量并不是整数片的 k 空间数据，该设计能更好地利用 GPU 资源。在基于上述扩展之外，由于采集到的 k 空间数据的无关性及该算法数据流的特性，该算法能被设计成多 GPU 实现，有进一步加速的潜力。此外，本文提出的方法数据处理流程还能推广到与 Wave-CAIPI 比较接近的磁共振成像的重建计算中去，但需对其中编码矩阵对应的函数 $A(\cdot)$进行相应的修改。

在已相对成熟的经典并行 MRI 方法的 GPU 实现的重建计算中，Hansen 等[11]在 256×256（16 个接收线圈、4 倍欠采样）数据上的 GPU 实现相较于 CPU 的加速倍数为 23.8 倍，重建时间为 26.1 ms。但 Qazi 等[16]在 256×256（30 个接收线圈、8 倍欠采样）人体心脏数据上的 GPU 实现相较于 CPU 的加速倍数为 10.53 倍，重建时间为 1.73 s；Ullah 等[17]在 448×224（12 个接收线圈、6 倍欠采样）数据上的 GPU 实现相较于 CPU 的加速倍数为 31.97 倍，重建时间为 61.09 ms。由于不同研究中的实验数据和条件不同，加速倍数不能直接用来比较，因此可通过重建时间来评估重建方法是否适用于实时医疗。我们以重建一个完整三维数据所需时间在 1 min 以内为可接受的时间，上述 Hansen 等[11]和 Ullah 等[17]介绍的 GPU 实现在各自实验使用的数据规模下都符合该条件。然而，本文使用的迭代方法中计算最小粒度是混叠的读出行数据，而不是混叠的像素数据，因此并行度不如上述 3 种方法。另外，本文方法中 cufft 表现出较低的并行度，以及 Wave-CAIPI 相较于经典并行 MRI 方法（SENSE 等）有着更为复杂的编码矩阵和重建计算过程，本文所提方法的 GPU 重建计算时间在文中使用的数据上比上述 3 种方法长，尚不能满足临床医学的实时性要求。对于较低的 cufft 并行度，可考虑设计一种更复杂的数据流控制过程，在对应的读出行重建完成时及时移除该读出行数据，以减少需要进行快速傅里叶变换的数据量，或利用 Linux 平台上的 cufft 静态库，通过动态并行技术来提高 cufft 的并行度，进一步实现加速。在未来研究中，将考虑把这些技术应用到本文提出的方法上，以便能更加接近实时性这一目标。

6 结　论

随着并行成像技术的普及和磁共振成像采集技术的发展，重建计算越来越复杂。本文提出了一种基于 GPU 改进的 CG 算法来完成对 Wave-CAIPI 这一新并行磁共振成像方法的重建。水模数据和人脑数据的重建实验结果表明，本文提出的基于 GPU 的方法重建图像质量与基于 CPU 的方法重建出的图像质量相差不大，但能显著减少重建计算时间，并且该方法很容易扩展到数据量更大的重建计算任务上。最后，本文还分析了影响该方法计算性能的因素。在未来的研究中，将尝试在 Linux 平台上实现该方法，并利用动态并行技术来提高该方法 cufft 模块的并行度，以进一步提高重建算法计算性能。

参 考 文 献

[1] 赵喜平. 磁共振成像 [M]. 北京: 科学出版社, 2004: 1-60.

[2] Hamilton J, Franson D, Seiberlich N. Recent advances in parallel imaging for MRI [J]. Progress in Nuclear Magnetic Resonance Spectroscopy, 2017, 101: 71-95.

[3] Sodickson DK, Manning WJ. Simultaneous acquisition of spatial harmonics (SMASH): fast imaging with radiofrequency coil arrays [J]. Magnetic Resonance in Medicine, 1997, 38(4): 591-603.

[4] Pruessmann KP, Weiger M, Scheidegger MB, et al. SENSE: sensitivity encoding for fast MRI [J]. Magnetic Resonance in Medicine, 1999, 42(5): 952-962.

[5] Griswold MA, Jakob PM, Heidemann RM, et al. Generalized auto calibrating partially parallel acquisitions (GRAPPA) [J]. Magnetic Resonance in Medicine, 2002, 47(6): 1202-1210.

[6] 郑海荣, 吴垠, 贺强, 等. 基于高场磁共振的快速高分辨成像 [J]. 生命科学仪器, 2018, 16(Z1): 29-44, 54.

[7] Bilgic B, Gagoski BA, Cauley SF, et al. Wave-CAIPI for highly accelerated 3D imaging [J]. Magnetic Resonance in Medicine, 2015, 73(6): 2152-2162.

[8] Breuer FA, Blaimer M, Mueller MF, et al. Controlled aliasing in volumetric parallel imaging (2D CAIPIRINHA) [J]. Magnetic Resonance in Medicine, 2006, 55(3): 549-556.

[9] Moriguchi H, Duerk JL. Bunched phase encoding (BPE): a new fast data acquisition method in MRI [J]. Magnetic Resonance in Medicine, 2006, 55(3): 633-648.

[10] Eklund A, Dufort P, Forsberg D, et al. Medical image processing on the GPU-past, present and future [J]. Medical Image Analysis, 2013, 17(8): 1073-1094.

[11] Hansen MS, Atkinson D, Sorensen TS. Cartesian SENSE and *k*-*t* SENSE reconstruction using commodity graphics hardware [J]. Magnetic Resonance in Medicine, 2008, 59(3): 463-468.

[12] Inam O, Qureshi M, Malik SA, et al. GPU-accelerated self-calibrating GRAPPA operator gridding for rapid reconstruction of non-cartesian MRI data [J]. Applied Magnetic Resonance, 2017, 48(10): 1055-1074.

[13] Wang H, Peng H, Chang Y, et al. A survey of GPU-based acceleration techniques in MRI reconstructions [J]. Quantitative Imaging in Medicine and Surgery, 2018, 8(2): 196-208.

[14] Baron CA, Dwork N, Pauly JM, et al. Rapid compressed sensing reconstruction of 3D non-cartesian MRI [J]. Magnetic Resonance in Medicine, 2018, 79(5): 2685-2692.

[15] Smith DS, Sengupta S, Smith SA, et al. Trajectory optimized NUFFT: faster non-cartesian MRI reconstruction through prior knowledge and parallel architectures [J]. Magnetic Resonance in Medicine, 2019, 81(3): 2064-2071.

[16] Qazi SA, Nasir S, Saeed A, et al. Optimizing image reconstruction in SENSE using GPU [J]. Applied Magnetic Resonance, 2018, 49(2): 151-164.

[17] Ullah I, Nisar H, Raza H, et al. QR-decomposition based SENSE reconstruction using parallel architecture [J]. Computers in Biology and Medicine, 2018, 95: 1-12.

[18] Saunders M. CGLS: CG method for $Ax=b$ and least squares [EB/OL]. [2019-05-24]. http://web.stanford.edu/group/SOL/software/cgls/.

[19] Hestenes MR, Stiefel E. Methods of conjugate gradients for solving linear systems [J]. Research of the National Bureau of Standards, 1952, 49(6): 409-436.