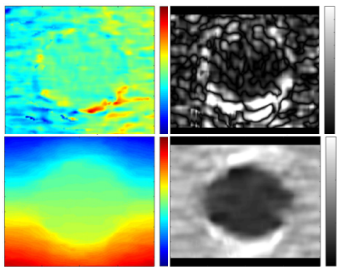
Opto-Electronic Engineering
光 电 工 程

Article

2019 年，第 46 卷，第 6 期

DOI: 10.12086/oee.2019.180437

基于和表的互相关计算方法在超声弹性成像中的性能分析



彭　博[1*]，罗莎莎[1]，杨　烽[1]，姜劲枫[2]

[1]西南石油大学计算机科学学院，四川 成都 610500；
[2]密歇根理工大学，密歇根州 霍顿 49931，美国

摘要：互相关计算方法的性能对超声弹性成像运动的计算效率起着决定性的作用。在串行计算环境下，基于和表的快速互相关算法在维持计算精度的同时可以获得更快的计算效率。然而，在并行计算环境下，尤其是 GPU 平台上，基于和表的快速互相关算法的实现以及性能还没有相关的报道。在本研究中，以二维超声弹性成像的运动追踪应用为目标，基于和表的快速互相关算法(ST-NCC)在 GPU 平台上得以实现，并且从计算效率及计算精度和传统的互相关算法进行了详细比较。初步结果显示，虽然基于和表的快速互相关算法(ST-NCC)在串行计算环境下获得了较好的计算效率，但是在 GPU 环境下，两种方法的计算效率没有较大的差距。

关键词：散斑追踪；块匹配；归一化互相关；和表；超声弹性成像

中图分类号：TB872　　**文献标志码：**A

引用格式：彭博，罗莎莎，杨烽，等. 基于和表的互相关计算方法在超声弹性成像中的性能分析[J]. 光电工程，2019，**46**(6): 180437

Performance analysis of a sum-table-based method for computing cross-correlation in GPU-accelerated ultrasound strain elastography

Peng Bo[1*], Luo Shasha[1], Yang Feng[1], Jiang Jinfeng[2]

[1]School of Computer Science, Southwest Petroleum University, Chengdu, Sichuan 610500, China;
[2]Department of Biomedical Engineering, Michigan Technological University, Houghton, Michigan 49931, USA

Abstract: The calculation of correlation is critically important for ultrasound strain elastography. The sum-table based method for the calculation of the normalized correlation coefficient (ST-NCC) can greatly improve computational efficiency under an environment of serial computing. Its implementation and performance are yet to be investigated when given a parallel computing platform, particularly, under a GPU environment. In this study, the published ST-NCC method was implemented into GPU and its performance was evaluated for speckle tracking. Particularly, the performance of the ST-NCC method was compared to the classic method of computing NCC using simulated ultrasound data. Our preliminary results indicated that, under the GPU platform, the implemented ST-NCC method did not further improve the computational efficiency, as compared to the classic NCC method implemented into the same GPU platform.

收稿日期：2018-05-17；　收到修改稿日期：2018-12-25
基金项目：四川省苗子工程重点项目(2018RZ0093)；四川省南充市校科技战略合作项目(NC17SY4020)
作者简介：彭博(1980-)，男，博士，副教授，主要从事医学超声信号与图像分析的研究。E-mail：bopeng@swpu.edu.cn

Keywords: speckle tracking; block-matching; normalized cross-correlation; sum-table; ultrasound strain elastography

Citation: Peng B, Luo S S, Yang F, *et al*. Performance analysis of a sum-table-based method for computing cross-correlation in GPU-accelerated ultrasound strain elastography[J]. *Opto-Electronic Engineering*, 2019, **46**(6): 180437

1 引 言

超声散斑运动追踪在许多超声成像技术中都具有非常重要的作用，如弹性成像、多普勒血流成像等。在弹性成像中，基于相关性的时域超声散斑运动追踪方法具有：实现简单，具有较好的空间解析度和较好的位移精度等优点。因此，超声散斑运动追踪被广泛用于软组织的运动估计[1-2]。块匹配方法是超声散斑运动追踪最简单也是最广泛使用的一种方法[3-4]。块匹配算法计算过程中，相似度评价方法扮演着非常重要的角色，是关系到计算精度和计算效率的最重要部分。不同的相似度评价方法对运动追踪的准确度和计算效率具有较大影响，常用的相似度评价方法有取最小值的绝对误差和(SAD)、均方误差和(SSD)和取最大值的归一化互相关(NCC)等。归一化互相关(NCC)已经被证明是块匹配方法中最好的相似性匹配准则之一[5]，由于归一化互相关具有较高的准确度且实现简单，因此它被广泛应用于各种超声散斑运动追踪算法中。

标准的归一化互相关算法在超声散斑运动追踪中具有计算准确度高和实现简单的优点，但缺点是计算量大，不易用于实时成像。因此，如果某些软组织的运动是可预测的，在运动追踪中可以通过搜索预测来减少互相关方法的计算量[1, 6]。但是，当搜索预测不适用的时候(比如，心脏的运动)[7-8]，全搜索成为唯一选择。Lewis 提出的基于和表的归一化互相关算法[9]有效地解决了这一问题，但其分子部分(标准互相关)采用的是基于 FFT 的频域方法，依然具有一定的计算量。后来 Luo 等对分子部分也采用和表计算替代较耗时的 FFT 方法[5]，他们的方法能够获得几乎完全相同的估计精度和更快的计算效率。

伴随着 GPU 硬件技术的发展，基于 GPU 的并行计算在准静态弹性成像[10-12]，剪切波弹性成像[13]，三维弹性成像[3]、彩色多普勒[14]和超声 CT[15]等超声技术领域中取得了丰富的研究成果。然而据查阅文献，在 GPU 并行计算环境下，标准的归一化互相关算法和基于和表的归一化互相关算法在超声散斑运动追踪中的研究还未见报道。在本文中，以 GPU 为并行加速平台，对和表的归一化互相关算法的效率以及计算精度进行了系统性的评估。同时，为全面比较和分析，本文中也实现了这两种方法的 CPU 串行计算程序并作为性能比较的重要参考。

2 方 法

2.1 超声散斑运动追踪方法

超声散斑运动追踪方法遵循块匹配运动估计框架。首先在压缩前射频(radio frequency，RF)回波数据(也就是参考帧)确定估计位置和追踪互相关窗口大小，然后确定好搜索区域，最后在压缩后 RF 回波数据(也就是目标帧)中通过计算归一化互相关系数(或其它相似性匹配方法)来寻找最相似的数据块，其对应的搜索位置就是参考窗口的位移。在实际计算中，通常会给定一个预先定义好的搜索区域，以减少不必要的计算开销。值得指出的是，相似性匹配方法在超声散斑运动追踪方法或块匹配方法中扮演着非常重要的角色。图 1 简单描述了基于传统归一化互相关算法(方法一)和基于和表的归一化互相关算法(方法二)的超声散斑运动追踪过程的计算流程。可以看出，方法二

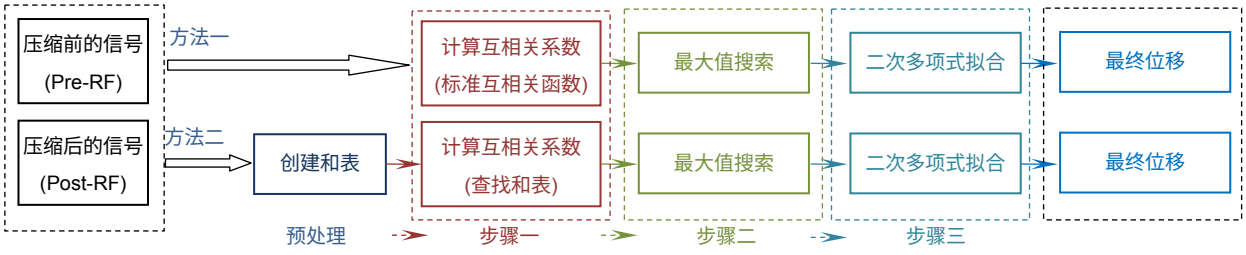


图 1 超声散斑运动追踪流程示意图

Fig. 1 Diagram of ultrasonic speckle motion tracking process

需要在计算之前创建和表，这样在步骤一中的互相关系数计算过程可以直接用查表法来代替方法一中所采用的标准归一化互相关计算公式。后面的步骤二和步骤三对于这两种方法计算过程完全一样，步骤二负责互相关系数矩阵的最大值搜索，它可以确定每一个估计位置的整数位移。而步骤三是将通过最大值位置周围 3×3(包含最大值本身)的互相关系数矩阵进行二次多项式拟合获得位移的偏移量[11]。通过这两步，可以获得最终的具有子采样精度的轴向位移和横向位移。

2.1.1 归一化互相关算法

假设压缩前后的两帧 RF 数据分别为参考信号 f 和目标信号 g。采用标准的归一化互相关算法对计算参考信号 f 中的某一参考窗口与目标信号 g 中的目标窗口之间的互相关系数的计算方法定义如下为

$$R_{\mathrm{ncc}}(m,n,\tau_x,\tau_y)=\frac{\sum_{m=u}^{U}\sum_{n=v}^{V}f(m,n)\cdot g(m+\tau_x,n+\tau_y)}{\sqrt{\sum_{m=u}^{U}\sum_{n=v}^{V}f^2(m,n)\cdot\sum_{m=u}^{U}\sum_{n=v}^{V}g^2(m+\tau_x,n+\tau_y)}},\quad(1)$$

式中：$U=u+W_x-1$，$V=v+W_y-1$，$\tau_1\le\tau_x\le\tau_2$，$\tau_3\le\tau_y\le\tau_4$，(m,n) 是参考窗口的左上点，用于确定参考窗口的位置，W_x 和 W_y 是参考窗口的横向和轴向大小。τ_x 和 τ_y 代表参考窗口在目标信号上的某一搜索位置。$[\tau_1,\tau_2]$ 和 $[\tau_3,\tau_4]$ 分别表示参考窗口在目标信号中的横向和轴向搜索范围。由式(1)计算一个互相关系数需要进行 $(3W_x\cdot W_y+1)$ 次乘法和 $3(W_xW_y-1)$ 次加法+1 次乘法、1 次除法和 1 次开方。

2.1.2 基于和表的归一化互相关算法

基于和表归一化互相关算法最早由 Lewis 提出[9]。算法的理论依据就来源于(标准归一化互相关算法计算全搜索各个搜索位置时，对应的目标窗口具有非常高的重叠度，导致大多数计算都是冗余。)这一事实，该方法的提出大幅提高了归一化互相关算法的计算效率。后来 Luo 等首次将基于和表方法用于超声信号的运动估计中[5]。式(1)分为三部分，$\sum_{m=u}^{U}\sum_{n=v}^{V}f(m,n)\cdot g(m+\tau_x,n+\tau_y)$ 表示参考窗中信号与目标窗中信号标准的互相关；$\sum_{m=u}^{U}\sum_{n=v}^{V}f^2(m,n)$ 表示参考窗中信号总能量；$\sum_{m=u}^{U}\sum_{n=v}^{V}g^2(m+\tau_x,n+\tau_y)$ 表示目标窗口中总能量。这三部分和表按下式构建：

$$s_f^2(u,v)=\sum_{m=0}^{u}\sum_{n=0}^{v}f^2(m,n),$$

$$s_g^2(u,v)=\sum_{m=0}^{u}\sum_{n=0}^{v}g^2(m,n),$$

$$s_{f,g}(u,v,\tau_x,\tau_y)=\sum_{m=0}^{u}\sum_{n=0}^{v}f(m,n)\cdot g(m+\tau_x,n+\tau_y)。\quad(2)$$

基于和表的归一化互相关算法将式(1)的计算转换成了对应和表的查表计算。

$$\sum_{m=u}^{U}\sum_{n=v}^{V}f^2(m,n)=s_f^2(U,V)-s_f^2(u-1,V)-s_f^2(U,v-1)+s_f^2(u-1,v-1)\quad,\quad(3)$$

$$\sum_{m=u}^{U}\sum_{n=v}^{V}g^2(m+\tau_x,n+\tau_y)=s_g^2(U+\tau_x,V+\tau_y)-s_g^2(u-1+\tau_x,V+\tau_y)-s_g^2(U+\tau_x,v-1+\tau_y)+s_g^2(u-1+\tau_x,v-1+\tau_y)\quad,(4)$$

$$\sum_{m=u}^{U}\sum_{n=v}^{V}f(m,n)\cdot g(m+\tau_x,n+\tau_y)=s_{f,g}(U,V,\tau_x,\tau_y)-s_{f,g}(u-1,V,\tau_x,\tau_y)-s_{f,g}(U,v-1,\tau_x,\tau_y)+s_{f,g}(u-1,v-1,\tau_x,\tau_y)\quad。\quad(5)$$

根据式(3)、式(4)、式(5)，式(1)的计算被简化成了 3×(1 次加法+2 次减法)+1 次乘法、1 次除法和 1 次开方运算。由于超声散斑运动追踪方法需要在整个信号的多个估计位置进行计算，这样节省的计算量是非常可观的。同时，基于和表的方法可以不受追踪互相关窗口大小的影响，所以从算法复杂度上可以直接分析出基于和表的归一化互相关算法相较于传统方式有较大的性能提升。

2.2 超声散斑运动追踪的 GPU 并行实现

2.2.1 CUDA 介绍

CUDA(Compute unified device architecture)是 NVIDIA 推出的通用并行计算架构。该架构使 GPU 能够以并行的方式解决复杂的计算问题。在 CUDA 中，KERNEL 函数是被定义为执行多线程并行计算的函数，DEVICE 函数是被定义为 GPU 上由 KERNEL 函数调用的单线程函数。同时，根据 CUDA 的内存结构，片外存储器(如全局内存、纹理存储器)相对于片上存储器(如寄存器、共享内存、常量存储器)具有较高的访问延迟，在程序设计时应尽量利用片上存储器提高访存效率。

2.2.2 超声散斑位移追踪并行实现框架

超声散斑位移追踪的并行实现过程通过以下几个策略来完成。第一，在互相关函数的计算中，轴向的位移估计点数量 M 及横向超声扫描线的数量 N，这两个参数决定了总的位移估计点数量为 $M\times N$。因为每一个位移估计点的计算过程都是无数据依赖的，在 CUDA 编程结构中，可以通过前面介绍的 KERNEL 函数启动 $M\times N$ 线程来实现并行计算。进一步如果考虑在每一个估计点的位移搜索计算过程中需要在轴向 A 个位置和横向 B 个搜索位置计算互相关系数值，这些搜索位置的互相关系数的计算过程也是不需要互相通信的。相应的步骤一(见图 1)的 KERNEL 函数可以启

动$(M\times N)\times(A\times B)$个 CUDA 线程满足算法并行执行的需求。而每一个搜索位置的互相关系数(标准归一化互相关算法)计算过程由一个 DEVICE 函数完成。而后续的步骤二和三(见图 1)对应的 KERNEL 函数启动的线程则与总的位移估计点数量($M\times N$)一致。每一个 CUDA 线程执行都会调用 1 个对应的 DEVICE 函数对其所对应的互相关系数矩阵进行最大值搜索及完成二次多项式拟合的计算任务。在实现过程中，为保证内存访问的一致性，即相邻的线程访问相近的内存区域，尽量满足合并内存事务而采用一维的线程结构。第二，RF 数据的存储到纹理内存，NVIDIA GPU 的纹理(TEXTURE)内存技术通过硬件加速块数据的读取过程，避免了跨行读取所造成的延迟。最后，一些在程序中不变的参数(如轴向与横向搜索范围，互相关计算核尺寸)存储到 GPU 常量内存中，以实现快速访问。

2.2.3 归一化互相关匹配方法并行实现

为了提高标准归一化互相关算法在计算时的访存效率。考虑到在每一个位移估计点搜索时的互相关系数计算所需要的数据都存在大量的数据重复读取的情况。为了避免频繁地从 GPU 全局内存读取数据，本文在 CUDA 编程时按列将 RF 回波数据加载到片上存储器中以提高访存效率。具体的三种实现方式如下：

1) GPU 版本 1：步骤一中，将计算所需的部分参考回波帧 RF 数据加载到共享存储器，而目标回波帧中 RF 数据保留在 TEXTURE 中。

2) GPU 版本 2：步骤一中，将计算所需的部分目标回波帧 RF 数据加载到共享存储器，而参考回波帧中 RF 数据保留在 TEXTURE 中。

3) GPU 版本 3：步骤一中，将参考回波帧和目标回波帧所需要的部分 RF 数据均加载到共享存储器。

2.2.4 和表并行计算实现

基于和表的二维归一化互相关算法相对于标准归一化互相关算法多了一个预处理步骤，即是和表计算。整个步骤需要计算 1 个参考信号和表、1 个目标信号和表、$(\tau_2-\tau_1+1)\times(\tau_4-\tau_3+1)$个参考信号和目标信号在搜索位置$(\tau_x,\tau_y)$的和表计算。本文中采用由 Sengupta 和 Harris 等提出 GPU 并行扫描算法[16]实现这些和表的快速创建。该算法基于 Blelloch 提出的算法[17]，并将平衡树的思想应用到算法中，通过在输入数据上构建一个平衡的二叉树，并将其扫描到根中并计算前缀和。

图 2 简单演示了二维和表的创建过程，二维和表构建时主要分为以下两个阶段：第一阶段沿 X 轴方向逐行使用并行扫描算法构建横向和表，第二阶段在横向和表的基础上沿 Y 轴方向逐列使用并行扫描算法构建竖向和表。在这两个阶段的计算时会按行和列调用并行扫描算法[16]两次。假如参考信号帧的尺寸为$(R_{\text{Rows}}\times C_{\text{Cols}})$，相应的 KERNEL 函数需要启动$(R_{\text{Rows}}\times C_{\text{Cols}})$个 CUDA 线程满足算法并行执行的需求。计算$(\tau_2-\tau_1+1)\times(\tau_4-\tau_3+1)$个参考信号和目标信号在搜索位置$(\tau_x,\tau_y)$的和表计算则需要启动$(\tau_2-\tau_1+1)\times(\tau_4-\tau_3+1)\times(R_{\text{Rows}}\times C_{\text{Cols}})$个 CUDA 线程。

2.2.5 实验验证

实验所需的超声 RF 回波信号通过计算机模拟产生。计算机模拟所采用的数字体模尺寸为(40 mm×40 mm)，体模内部的中心位置包含一个半径为 5 mm 的圆形包容物，其硬度为背景的 5 倍。利用有限元软件(ANSYS version 12.0, ANSYS, Inc., PA, USA)对这个数字体模在轴向实施了 1%的压缩。压缩产生的模拟位移被用来产生模拟超声测量位移，然后通过 FIELD II 超声模拟软件[18]使用一个 192 阵元的线性阵列探头，64 个活动阵元对这个计算机体模进行模拟扫描。模拟所采用的中心频率为 5 MHz 和采样频率为 40 MHz。聚焦深度设置为 20 mm，采用动态接收聚焦模型生成超声 RF 回波信号长度为 2078，共 150 条 RF 回波信号。

超声弹性成像的应变来源于对位移计算差分，即横向与轴向应变为$\varepsilon_y=\partial(t\mathrm{d}y)/\partial y$和$\varepsilon_x=\partial(t\mathrm{d}x)/\partial x$，其中，$t\mathrm{d}x$和$t\mathrm{d}y$分别代表计算得到的横向与轴向位移，在本研究中，局部轴向应变和横向应变的计算通过一个低通数字差分滤波器实现[19]。

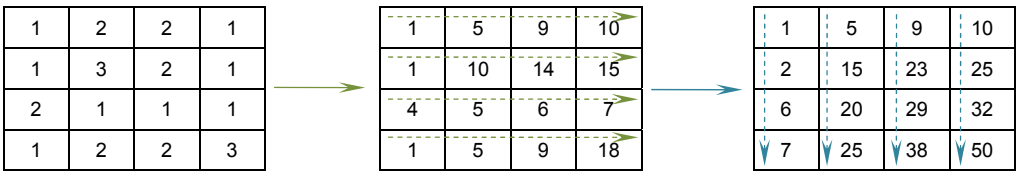


图 2 参考窗口和表的并行扫描构建

Fig. 2 An illustration of the parallel scan method for calculating the sum-table

2.2.6 算法实现与数据分析

算法实现采用的 GPU 硬件为 NVIDIA GeForce GTX TITAN X (NVIDIA Corp., Santa Clara, CA, USA)。该 GPU 卡有 80 个流多处理器(stream multiprocessors)，共计 5120 个 CUDA 计算核心，12 GB Memory。该 GPU 卡被安装在一台操作系统为 Ubuntu 16.04，计算 CPU 为 Xeon E3-1220 V2 CPU @3.1 GHz，主机内存为 16 GB 的桌面工作站。软件平台为 Matlab(Mathworks Inc., MA, USA)，CPU 算法实现采用 ANSI C，CUDA9.0 被用作编程框架来实现所有的 GPU 算法。所有实现的算法通过 MEX 接口在 Matlab 中调用。

为评估这两种方法，即基于标准归一化互相关算法的超声弹性成像运动追踪算法(方法一)和基于和表的归一化互相关算法的超声弹性成像运动追踪算法(方法二)的性能，本文实现了方法一的 CPU 版本和 GPU 版本 1、GPU 版本 2 及 GPU 版本 3；方法二的 CPU 版本和 GPU 版本。本文主要从以下方面进行对比：评估两种方法在 CPU 串行计算条件下的运动追踪位移差值是否一致；评估两种方法各自的 CPU 版本与对应的 GPU 版本的位移差值是否差异极小。评估两种方法在 CPU 串行计算条件下和 GPU 并行计算条件下在散斑追踪主要参数影响下的计算效率差异。

3 结 果

图 3 演示了对采用标准归一化互相关算法的超声弹性成像运动追踪方法(方法一)使用三种片上优化策略对计算时间的影响。实验中 RF 信号帧的压缩比为 1%，对应的搜索范围为[41×7，0.4 mm×1.4 mm]，位移追踪估计点尺寸为[100×100]。从实验结果来看，片上优化策略对于方法一计算效率的影响较大。随着互相关追踪窗口逐渐增大，观察三种优化策略对应的曲线可以明显得出结论，即方法一 GPU 版本 3(将计算所需的参考信号和目标信号都装入共享内存时)算法的计算时间最快，且随着互相关追踪窗口尺寸的增加，优势更加明显。在剩余两种优化策略中，方法一 GPU 版本 2，即将目标回波帧的计算部分存储在共享内存时比将参考回波帧的计算部分单独存储在共享内存中快。这是因为对任意一个位移估计点计算归一化互相关系数时，都需要频繁地去访问目标回波帧中对应于搜索位置 (τ_x,τ_y) 的数据，但该部分数据具有较大的重叠，冗余度非常高。方法一 GPU 版本 2 和版本 3 把这部分数据均存储于片上共享内存，大大提高了访存效率，因此减少了计算时间。(因此后续实验中方法一的 GPU 实现默认采用的是方法一的 GPU 版本 3)。

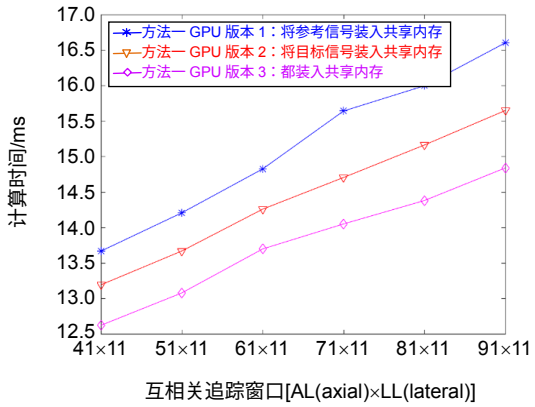


图 3 不同片上优化策略下算法的计算时间对比

Fig. 3 Comparison of computation time for method 1 under different optimization strategies

图 4 演示了两种方法的 CPU 串行实现和 GPU 并行实现所得运动追踪精度的差异。为了保证算法的可比性，两种算法在 CPU 串行计算条件下和 GPU 并行计算条件下的互相关追踪窗口大小为[61×11]、计算兴趣区为[100×100]个位移估计点、搜索范围为[11×7]均保持一致。图 4 中所显示的位移差值均为绝对值，单位为 mm。由于测得的位移差值的范围较大(数量级的差距)，为演示方便，本文采用 log10(位移差值)进行图示。图 4(a)和 4(d)为通过方法一的 CPU 串行实现计算得到横向与轴向位移，图 4(b)和 4(e)为对应的横向弹性图和轴向弹性图；图 4(c)和 4(f)演示了方法一与方法二的 CPU 串行实现之间运动追踪位移差，从图中可以看出它们之间的横向和轴向位移差范围分别在 [10^{-15} mm, 10^{-9} mm]和[10^{-17} mm, 10^{-14} mm]之间，这说明两种方法的 CPU 串行实现得到的位移追踪结果几乎完全一致，误差极小；图 4(g)和 4(i)为方法一的 CPU 串行实现与 GPU 并行实现之间的横向和轴向位移差，其误差范围分别为[10^{-7} mm, 10^{-5} mm]和[10^{-9} mm, 10^{-7} mm]，这是因为 GPU 实现时采用的数据类型为单精度，导致位移差范围相比于 CPU 的双精度实现有较大程度的增加；图 4(h)和 4(j)为方法一与方法二的 GPU 并行实现之间的横向和轴向位移差，其误差范围分别为[10^{-7} mm, 10^{-4} mm]和[10^{-8} mm, 10^{-5} mm]，这里位移差进一步增大主要是来自于两个方面，第一是两个方法本身具有一定的误差，第二是 GPU 实现采用单精度带来的误差，虽然误差范围相对于前面 CPU 串行实现增大，但这种误差范围对超声弹性成像的运动追踪精度几乎没有影响，不会影响到最终超声弹性成像的质量。

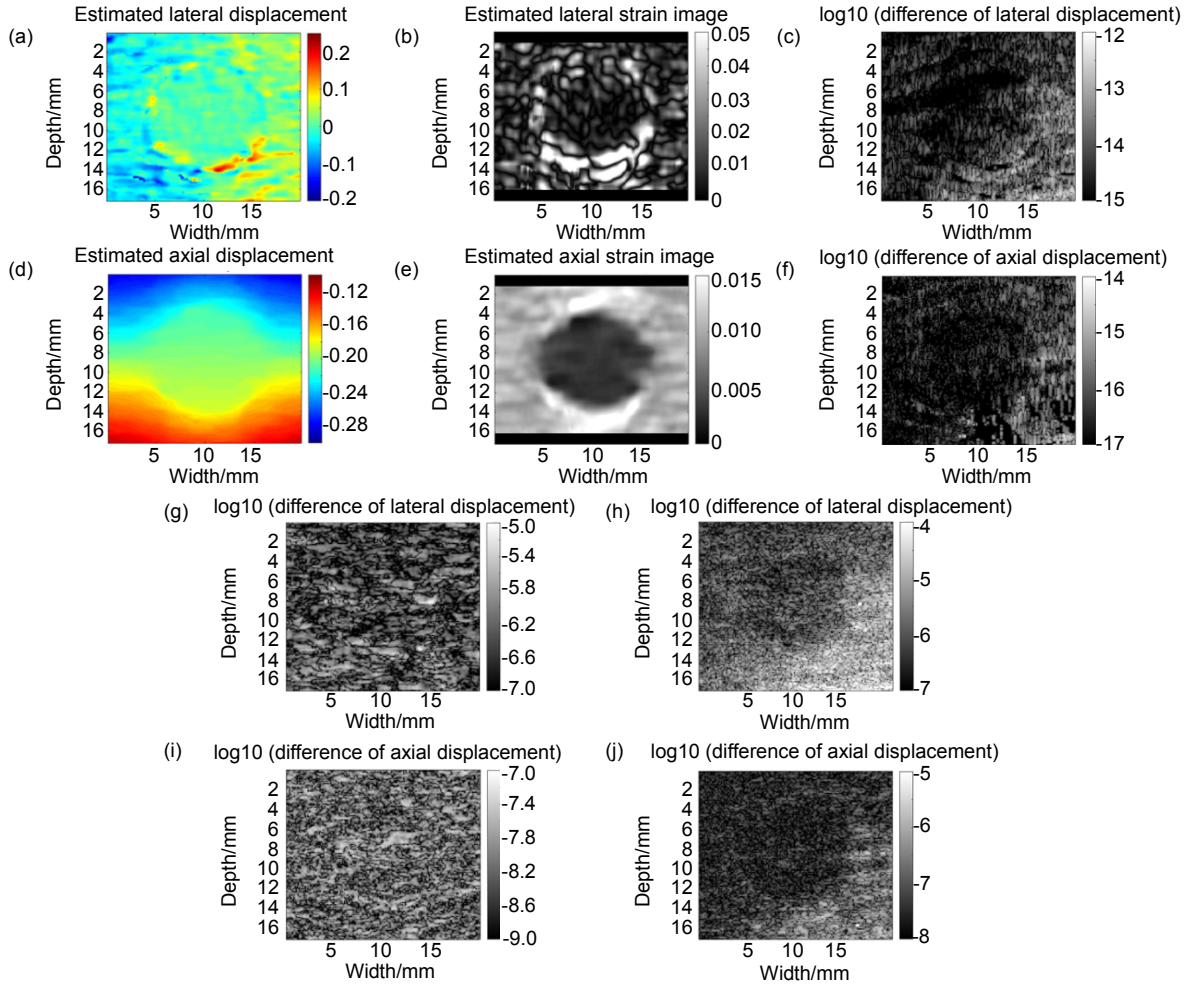


图 4 两种方法的运动追踪性能比较。(a)和(d)为用方法一的 CPU 串行实现计算得到横向和轴向位移；(b)和(e)为由(a)和(d)计算得到的对应横向和轴向应变；(c)和(f)为两种方法的 CPU 串行实现之间的横向和轴向位移差；(g)和(i)为方法一的 CPU 串行实现与 GPU 并行实现之间的横向和轴向位移差；(h)和(j)为两种方法的 GPU 并行实现之间的横向和轴向位移差

Fig. 4 A comparative performance analysis of CPU and GPU implementations. (a) and (d) are lateral and axial displacements obtained using CPU implementation of method 1; (b) and (e) are corresponding strain images; (c) and (f) are difference images of lateral and axial displacements between two methods on CPU; (g) and (i) are difference images of lateral and axial displacements between the CPU and GPU implementations of method 1; (h) and (j) are the difference images of displacement between GPU implementation of method 1 and method 2

图 5 演示了互相关追踪窗口尺寸变化对两种方法计算时间的影响，实验中 RF 信号帧的压缩比为 1%，所对应的搜索范围为[21×7]，在搜索范围相同的情况下，分别统计 CPU 串行计算条件下和 GPU 并行计算条件下两种算法各自所耗费的时间。通过对比图 5(a)和 5(b)可以发现：1) 无论是 CPU 还是 GPU 并行计算条件下基于和表的二维互相关算法的计算时间几乎与互相关窗口大小无关，而传统算法的计算时间受互相关窗口大小的影响较大。其中窗口越大，传统算法的计算时间就越长。2) 在 CPU 串行计算条件下，方法二具有明显的计算优势，最多获得了约 28 倍多的加速比。3) 在 GPU 并行计算条件下，方法一和方法二计算时间相差最多不超过 5 ms，但总体上，方法二的计算效率优于方法一。

图 6 演示了针对不同轴向压缩比的 RF 信号需要调整轴向搜索范围的计算时间对比结果。具有不同轴向压缩比的 RF 信号帧需要调整与之对应的轴向搜索范围，横向搜索范围基本保持不变，分别统计两种方法在 CPU 串行计算和 GPU 并行计算条件下的计算时间。从图 6(a)和 6(b)的比较可以得出以下三点：1) 在 CPU 串行计算条件下和 GPU 并行计算条件下，搜索范围发生改变时，基于和表的互相关方法和标准互相关方法的计算时间随着搜索范围的增大，它们的计算时间呈现出一种近似于线性增长的趋势。2) 在 CPU

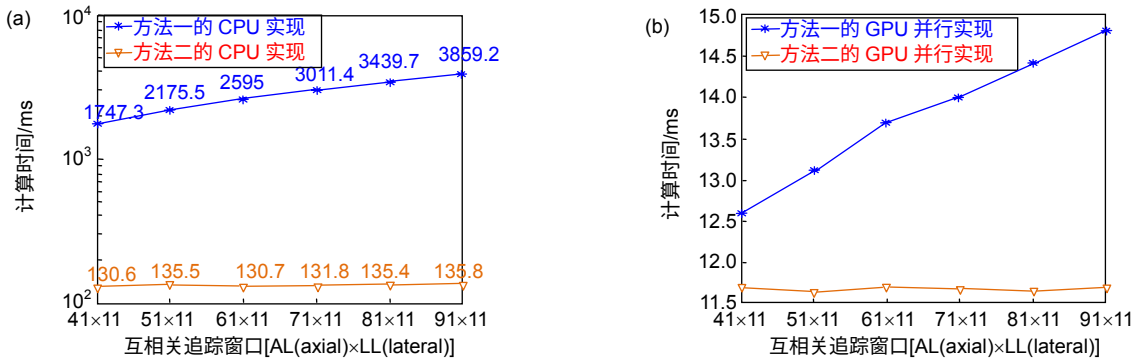


图 5 不同互相关追踪窗口两种算法计算时间。(a) 两种方法 CPU 实现随互相关追踪窗口变化时的计算时间；(b) 两种方法 GPU 实现随互相关追踪窗口变化时的计算时间

Fig. 5 Comparison of computation time of two method under different cross-correlation tracking windows. (a) Computation time of CPU implementation of the two methods; (b) Computation time of GPU implementation of the two methods

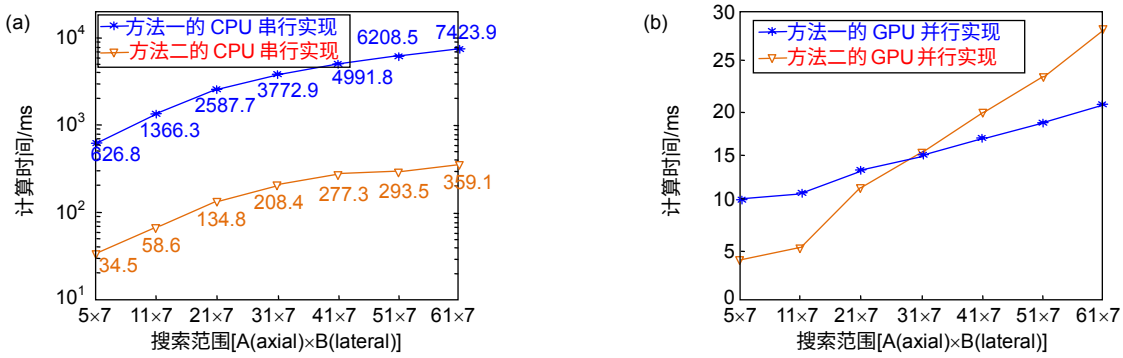


图 6 不同搜索范围条件下两种算法计算时间。(a) 两种方法 CPU 实现随搜索范围变化时的计算时间；(b) 两种方法 GPU 实现随搜索范围变化时的计算时间

Fig. 6 Comparison of computation time of two methods under different search ranges. (a) Computation time of CPU implementation of two methods; (b) Computation time of GPU implementation of two methods

串行计算条件下，与标准的归一化互相关算法相比，基于和表的归一化互相关算法可获得近 20 倍的加速比，在 GPU 并行计算条件下，当搜索范围较小时，该方法具备一定的速度优势，但如图 6 所示，随着搜索范围的变大，尤其是当搜索范围大于[31×7]时，由于和表算法所需构建的和表总数量增加，基于和表的互相关算法的计算效率开始低于标准方法。

4 讨 论

本文较全面地比较与分析了两种方法在 CPU 串行计算环境和 GPU 并行计算环境下的运动追踪准确度和计算效率。通过模拟的超声信号进行相关实现和实验发现，CPU 计算条件下，和表方法具有明显的计算优势；两种方法的 GPU 并行实现在保证不影响弹性成像准确度的前提下，都能获得较好的加速比。需要注意的是，相对于 CPU 串行计算环境，基于和表方法在 GPU 并行计算环境中与传统方法的计算效率并没有较大差距。同时也发现，和表方法还受超声信号尺寸、搜索范围等限制。如和表的创建要占用一定的计算时间，当搜索范围增大时，创建的和表增多，所占用的计算时间在整个计算时间的比重会有较大的上升。因此，当采用基于和表方法时，用一种多压缩追踪策略[19]减少大位移运动的搜索范围是一种合理的选择，从而避免出现和表计算耗时过多的情况。

标准的归一化互相关算法的 GPU 实现主要是通过充分利用片上内存方法提高访存效率，并且开发出了对应的三个版本，三个版本的实验结果也验证了片上内存在传输效率上与片外存储器的差距。值得注意的是，当前 GPU 的片内存储器的容量是有限制的，随着 GPU 硬件技术的发展，片内存储器的容量可能还会继续增加，这必将进一步提升传统方法 GPU 实现的性能。同时，当前的 GPU 实现也可以被进一步优化，在多指令多数据(multiple instruction multiple data，MIMD)模式下，当一条指令在等待装载数据时，另一条数据同时执行计算任务。因此，MIMD 模式可进一步提高 GPU 实现的计算效率。

在本文中，从原始的超声 RF 信号(2078×150)中选择的计算感兴趣区域尺寸为(1024×128)，可以满足对病灶区域成像计算需求。同时也满足并行扫描算法对于信号尺寸需要为 2 的 n 次幂的要求。虽然信号尺寸为非 2 的 n 次幂的计算也可以通过分块实现，但由于其需要汇总数据，效率会进一步降低。

实施这项研究目的是基于当前越来越流行的 GPU 并行高性能计算。虽然和表方法在 CPU 计算条件下具有明显的优势，但是在超声弹性成像的运动追踪这一应用，系统性地量化比较两种方法在 CPU 和 GPU 并行计算条件下计算效率的研究依然未见报道。本文的工作，有助于在设计 GPU 加速的超声散斑运动算法时选择合适的方案从而产生最理想的计算效率。

5 结 论

本文的主要工作与贡献是在以超声弹性成像中的运动追踪为应用目标的情况下，以超声弹性成像中的运动追踪为应用背景，较全面的比较了传统的归一化互相关算法和基于和表的归一化互相关算法在 CPU 串行计算环境和 GPU 并行计算环境下的计算性能和效率。研究发现：1) 对比两种方法的 CPU 串行实现，它们的 GPU 并行实现都获得了较高的加速比，同时保证了位移误差在可接受的范围(横向和轴向位移平均误差的数量级为 10^{-7} mm 和 10^{-9} mm)；2) 两种方法的 GPU 并行实现计算效率差距不明显，同时基于和表的归一化互相关算法会受到超声 RF 信号尺寸、搜索范围的影响，而传统方法的 GPU 并行不会受到两种因素的影响，具有较高的灵活性。通过本文的工作，尤其是在当前基于 GPU 硬件技术的快速发展的条件下，为基于归一化互相关算法的超声弹性成像方法提供了可参考的数据。进一步的研究将致力于如何充分利用两种算法的优缺点，如基于和表的归一化互相关算法不受互相关窗口尺寸的影响，标准的归一化互相关算法可以使用预测搜索等，从而使超声弹性成像的计算效率获得进一步的提升。

参考文献

[1] Jiang J, Hall T J. A parallelizable real-time motion tracking algorithm with applications to ultrasonic strain imaging[J]. *Physics in Medicine & Biology*, 2007, **52**(13): 3773–3790.

[2] Chen L J, Treece G M, Lindop J E, *et al*. A quality-guided displacement tracking algorithm for ultrasonic elasticity imaging[J]. *Medical Image Analysis*, 2009, **13**(2): 286–296.

[3] Peng B, Wang Y Q, Hall T J, *et al*. A GPU-accelerated 3-D coupled subsample estimation algorithm for volumetric breast strain elastography[J]. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 2017, **64**(4): 694–705.

[4] Zhou Y J, Zheng Y P. A motion estimation refinement framework for real-time tissue axial strain estimation with freehand ultrasound[J]. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 2010, **57**(9): 1943–1951.

[5] Luo J W, Konofagou E E. A fast normalized cross-correlation calculation method for motion estimation[J]. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 2010, **57**(6): 1347–1357.

[6] Zhu Y N, Hall T J. A modified block matching method for real-time freehand strain imaging[J]. *Ultrasonic Imaging*, 2002, **24**(3): 161–176.

[7] D'Hooge J, Bijnens B, Thoen J, *et al*. Echocardiographic strain and strain-rate imaging: a new tool to study regional myocardial function[J]. *IEEE Transactions on Medical Imaging*, 2002, **21**(9): 1022–1030.

[8] Konofagou E E, D'Hooge J, Ophir J. Myocardial elastography--a feasibility study in vivo[J]. *Ultrasound in Medicine & Biology*, 2002, **28**(4): 475–482.

[9] Lewis J P. Fast template matching[J]. *Proceeding of Vision Interface*, 1995, **32**(4): 351–361.

[10] Yang X, Deka S, Righetti R. A hybrid CPU-GPGPU approach for real-time elastography[J]. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 2011, **58**(12): 2631–2645.

[11] Peng B, Huang L. GPU-accelerated sub-sample displacement estimation method for real-time ultrasound elastography[J]. *Opto-Electronic Engineering*, 2016, **43**(6): 83–88.
彭博, 黄丽. GPU 加速的高精度位移估计方法及超声弹性成像应用[J]. 光电工程, 2016, **43**(6): 83–88.

[12] Peng B, Chen Y, Liu D Q. Investigation of GPU-based ultrasound elastography [J]. *Opto-Electronic Engineering*, 2013, **40**(5): 97–105.
彭博, 谌勇, 刘东权. 基于 GPU 的超声弹性成像并行实现研究[J]. 光电工程, 2013, **40**(5): 97–105.

[13] Rosenzweig S, Palmeri M, Nightingale K. GPU-based real-time small displacement estimation with ultrasound[J]. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 2011, **58**(2): 399–405.

[14] Chang L W, Hsu K H, Li P C. GPU-based color Doppler ultrasound processing[C]//*2009 IEEE International Ultrasonics Symposium*. Rome, Italy, 2009.

[15] Sun X, Wang S S, Song J J, *et al*. Toward parallel optimal computation of ultrasound computed tomography using GPU[J]. *Proceedings of SPIE*, 2018, **10580**: 105800R.

[16] Sengupta S, Harris M, Garland M, *et al*. Efficient parallel scan algorithms for GPUs[M]//Kurzak J, Bader D A, Dongarra J. *Scientific Computing with Multicore and Accelerators*. Boca Raton: Taylor & Francis, 2008.

[17] Blelloch G E. Scans as primitive parallel operations[J]. *IEEE Transactions on Computers*, 2002, **38**(11): 1526–1538.

[18] Jensen J A. Field: A program for simulating ultrasound systems[J]. *Medical & Biological Engineering & Computing*, 1996, **34**(1): 351–352.

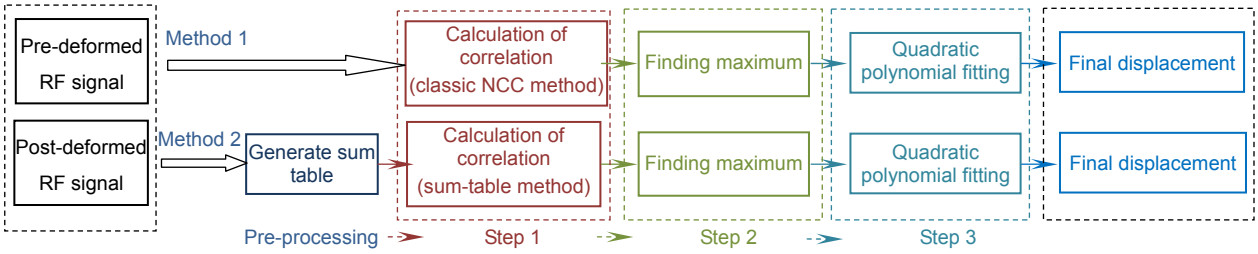
[19] Luo J W, Bai J, He P, *et al*. Axial strain calculation using a low-pass digital differentiator in ultrasound elastography[J]. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 2004, **51**(9): 1119–1127.

[20] Du H N, Liu J, Pellot-Barakat C, *et al*. Optimizing multicompression approaches to elasticity imaging[J]. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 2006, **53**(1): 90–99.

Performance analysis of a sum-table-based method for computing cross-correlation in GPU-accelerated ultrasound strain elastography

Peng Bo[1*], Luo Shasha[1], Yang Feng[1], Jiang Jinfeng[2]

[1]School of Computer Science, Southwest Petroleum University, Chengdu, Sichuan 610500, China;
[2]Department of Biomedical Engineering, Michigan Technological University, Houghton, Michigan 49931, USA



A schematic diagram showing steps involving in speckle tracking using radiofrequency (RF) echo ultrasound data

Overview: In our ultrasound strain elastography system, a modified block-matching algorithm is adopted to assess tissue motion. Then, local strains are assessed and used as surrogates of tissue elasticity. The calculation of correlation under the framework of the block-matching algorithm is a critical step and very computationally intensive. Because the correlation calculation is largely independent, graphics processing units (GPUs) have been utilized to improve computational efficiency through massive parallel programming. It is known in the literature that the sum-table based method can greatly reduce the computing burden when the calculation of the normalized correlation coefficient is needed in a serial computing environment. The sum-table based method is abbreviated as ST-NCC below. However, the performance of ST-NCC is yet to be investigated given a parallel computing platform, particularly, in a GPU environment. Consequently, our objective of this study is to investigate the performance of the ST-NCC method for the above-mentioned GPU-accelerated ultrasound strain elastography. More specifically, a published ST-NCC method by Luo et al. and the conventional NCC method were both programmed using CUDA (Version 9.0, NVIDIA Inc., CA, USA) and tested on an NVIDIA GeForce GTX TITAN X card. During the CUDA implementation, in order to achieve the best computational efficiency, two basic CUDA programming strategies were employed to improve computational efficiency for all CUDA implementation. First, in order to increase the memory bandwidth of GPUs, TEXTURE (memory) access was used for storing 2-D RF signals prior to the calculation of cross correlation. Second, programming variables that require frequent access (e.g., axial and lateral search ranges) were locked in read-only memory for rapid access. In terms of advanced CUDA programming strategies, on the one hand, a classic parallel scan method was adopted to generate those sum-table data for the ST-NCC method. On the other hand, a few different on-ship memory optimization strategies were used to implement the classic NCC method and they were compared against each other. Only the computationally most efficient implementation was used to compare with the above-mentioned GPU-accelerated ST-NCC method. Finally, performance assessments were conducted using simulated ultrasound data. Ultrasound data simulations involve both finite element modeling and acoustic simulations. Both displacement tracking accuracy and computational efficiency were evaluated during the performance assessments. Based on data investigated, we found that, under the GPU platform, the implemented ST-NCC method did not further improve the computational efficiency, as compared to the classic NCC method implemented into the same GPU platform. Comparable displacement tracking accuracy was obtained by both methods.

Citation: Peng B, Luo S S, Yang F, *et al*. Performance analysis of a sum-table-based method for computing cross-correlation in GPU-accelerated ultrasound strain elastography[J]. *Opto-Electronic Engineering*, 2019, **46**(6): 180437

Supported by Scientific Innovation Program of Sichuan Province (Major Engineering Project: 2018RZ0093) and Nanchong Scientific Council (Strategic Cooperation Program Between University and City: NC17SY4020)
* E-mail: bopeng@swpu.edu.cn