基于 Fermi 架构的超声图像自动增益补偿并行算法

何兴无[1,2],张 霞[2]

1. 成都师范学院网络与信息管理中心,成都 611130
2. 成都农业科技职业学院电子信息分院,成都 611130

摘要 在医学超声成像系统中由于超声波在人体组织内传播会发生衰减,需要对超声图像进行有效的增益补偿,使超声图像的显示效果更好。但大多数自动增益补偿算法在处理时涉及大量的复杂计算,成为临床实时成像系统中的一大性能提升瓶颈,为此提出了一种基于高性能并行计算平台 Fermi 架构图形处理单元(GPU)的自动增益补偿并行处理算法。本算法主要的处理流程有数据预处理、区域类型检测、组织强度计算、二次曲面拟合以及自适应增益补偿等部分,核心的并行算法设计包括了粗粒度的并行均值滤波、局部方差系数的并行计算、优化的矩阵转置并行实现以及基于 LU 分解的粗粒度的矩阵求逆的并行实现等方面。数据测试结果显示,与基于 CPU 的实现相比,采用 Fermi 架构的 GPU 处理不仅可以得到完全一致和较好的增益补偿效果,而且可以取得较大的加速效果,满足实时系统需求,对 512×261 的图像数据能够达到 427 帧/s 的高帧率,速度提高了大约 267 倍。

关键词 高性能并行计算;深度增益补偿;超声成像;图像并行处理算法

中图分类号 TP391.41　　**文献标识码** A　　**doi** 10.3981/j.issn.1000-7857.2012.31.009

A Parallel Algorithm of Automatic Time Gain Compensation for Ultrasound Imaging Based on Fermi Architecture

HE Xingwu[1,2], ZHANG Xia[2]

1. *Network & Information Management Center, Chengdu Normal University, Chengdu 611130, China*
2. *Department of Electronics and Information, Chengdu Vocational College of Agricultural Science and Technology, Chengdu 611130, China*

Abstract Due to the acoustic attenuation in the human body, an efficient gain compensation on the ultrasound image is necessary for a better imaging quality in a medical ultrasound imaging system. The traditional manual adjustment method suffers some drawbacks, such as the difficulties in adjusting a special region, so it is very important to implement the Automatic Time Gain Compensation (ATGC) algorithm in the clinical ultrasound imaging system. Because of the massive computation involved in this ATGC technique, this problem becomes the bottleneck for a clinical real-time imaging system. In this paper, a new parallel algorithm of ATGC based on Fermi GPU (graphics processing unit) is presented. The main procedures of this algorithm include the pre-processing, the speckle detection, the tissue intensity computation, the 2-D surface fitting and the adaptive gain compensation. The key parallel algorithm includes a parallel box filter with coarse-grained, parallel local variance coefficient computation, the optimized parallel matrix transposition, the parallel matrix inversion based on the LU factorization in a coarse-grained parallel way. Test results not only show that the output of the graphics processing unit (GPU) is definitely the same as that of the CPU, but also demonstrate an obvious speedup by using the GPU, that is, with 427 frames per second for the image size (512×261) , 267 times faster than the CPU implementation.

Keywords high performance parallel processing; depth gain compensation; ultrasound imaging; parallel algorithm for image processing

收稿日期: 2012-05-30;修回日期: 2012-07-05

作者简介: 何兴无,副教授,研究方向为超声图像,电子信箱:hexw981@126.com

0　引言

民众对医疗条件日益增长的要求极大地推动了医学技术的快速发展,而作为临床诊断和治疗手段的医学影像学已经成为医学技术发展最快的领域之一。医学成像技术使得临床医生对病人体内病变部位的观察更直接,更清晰,诊断准确率也越来越高。国家"十二五"规划纲要把建设农村医疗卫生服务体系作为社会主义新农村建设的重点工程,而彩超在医疗仪器采购中占20%的份额,国内广阔的市场促进了彩超行业的迅猛发展。医学超声诊断技术因为其易用性、无侵害性、实时性及廉价性等特点和优势,在现在的医学诊断学中有着难以取代的作用[1]。然而在医疗超声检测系统上,回波信号在通过组织时会发生极大的衰减[2],导致传感器接收的脉冲回波信号动态范围过大而不适合直接显示。因此,针对超声波在组织中的衰减采用增益补偿的方式来优化超声图像是必需的。这种优化可以使整个图像中的软组织亮度保持均匀性,并且抑制热噪声。更为重要的是,这里提出的自适应超声图像时间增益补偿算法,使原先用户通过传统调整时间增益补偿(TGC)滑块无法调节到的区域也能得到优化。自动TGC技术消除了手动调节TGC滑块的需要,减少了处理扫描的时间开销,增加了吞吐量。因此,自适应时间增益补偿技术在临床上具有重大的应用价值。

有关自适应超声图像时间增益补偿算法的研究工作已经开展多年。He[3]和Parker[4]的研究表明,基于时域的衰减估计比基于频域[5]得到的结果的方差更小。文献[6]提出了一种适合在宽频带系统中使用的基于最小二乘拟合的衰减估计算法,但同样因为巨大的计算代价而难以满足实时处理要求。本文所研究自动增益补偿算法是通过计算局部变异系数(coefficient of variation)识别出图像上的斑点和软组织区域[7],对强反射区域和暗区域(如血管)施以不同的增益补偿。这个算法目前还无法应用在普通的超声系统上,这是因为在系统高帧率工作模式时需要60—80帧/s才能保证处理的数据不丢失,让医生可以做回放检测。经过分析,找到这个算法的2个核心计算瓶颈:局部变异系数计算和二次曲面拟合。

2006年,英伟达公司将CUDA技术引入GPU通用计算领域,作为一种高性能并行计算的利器,CUDA技术为解决数字信号与图像处理方面的计算瓶颈问题带来了新的契机。目前CUDA 2.0以上的设备已经出现了采用Fermi并行处理架构的GPU,提高了GPU处理能力,在显存缓存机制和计算精度等方面使GPU更适合于数据的并行计算[8]。在各行业如航天航空、地理信息分析等应用领域中,GPU都已经得到了成功的应用,但在超声成像系统中的实际应用还方兴未艾,如利用CUDA处理超声B模式成像[9]和超声血流成像[10]等。

CUDA程序语言从语法的角度上看仅是C/C++超集,但GPU程序执行的方式与传统的CPU不同,GPU程序一般需要将数据从主机端传送到设备端,然后进行GPU多线程核函数的执行配置,启动核函数进行计算,最后直接显示或将计算结果传回到主机端。本文提出一种在通用计算机环境中基于CUDA的并行处理平台,并利用其新一代的并行体系架构Fermi实现超声图像自动增益补偿的并行处理算法。

1　Fermi架构下的超声图像自动增益补偿并行处理算法

本文所研究的自动增益补偿算法的核心思想是依据图像中的组织类型计算局部平均亮度并施以不同的增益。如果对所有区域施加相同的增益调节,就会使强反射区补偿过少,暗区域补偿过多(如血管)。通过使用局部方差系数来识别出软组织区域,就可以使软组织区域的补偿提升多于斑点噪声。在强度域中,完全的斑点噪声区域的像素强度值分布服从瑞利分布模型,这样在对数压缩后方差系数对于斑点噪声来说就是一个常数,可用于软组织在强反射体中的识别。图1显示了自动增益算法的处理流程,主要包含预处理、斑点/组织区域检测、组织强度计算、二维曲面拟合和增益计算补偿等步骤。

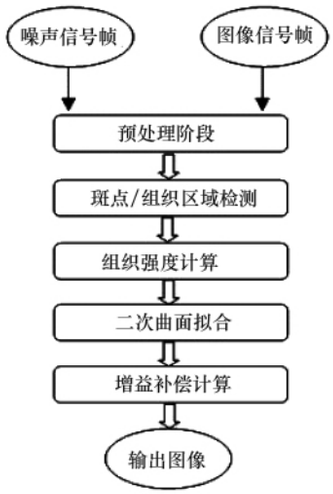


图1　超声图像自动增益补偿算法的处理流程

Fig. 1　Processing flow chart of automatic time gain compensation algorithm

1.1　预处理阶段

由于声学噪声和混杂信号服从相同的统计规律,方差本身并不足以成为组织类型判决的依据,所以需要一个低信噪比的背景噪声数据帧,也就是一个以热噪声帧为主的数据帧。使用热噪声帧的目的在于减少采用局部方差系数作为判决软组织区域依据的误差。同时,对数据帧和噪声数据帧施以均值滤波可以减少声学噪音信号的影响。预处理步骤中主要对输入的B模式数据帧和噪声数据帧做一个均值滤波操作,这个操作结果用于斑点噪声检测时比较信噪比。在并行算法设计中,主要有下面2种并行思想:细粒度和粗粒度。均值滤波采用细粒度的并行方式,与串行处理方式一样,将1

帧图像进行细分,每个线程负责完成 1 个像素点二维均值滤波输出。很明显,每个点的处理需要 n^2-1 步。而如果采用粗粒度的方式,分别沿着 x 和 y 轴完成 2 次一维的平均计算从而实现二维均值滤波,就可以启动 2 个核函数,让线程负责处理 1 行或者 1 列。采用这种并行处理方式就只需要 $n-1$ 步了,显然后者大大减少了冗余计算。在 GPU 上实现的具体做法是使用 2 个核函数。

第 1 个核函数沿 x 方向计算滤波窗口尺寸的和,采用 1 个线程扫描 1 行的方式,算法实现伪代码为

```
__global__void Sum(float *in,float *out,int w,int h,int r,
    float scale){//r 为窗口半径,scale 为窗口长度的倒数
    //将线程索引号与所需要处理的行号建立映射
    int r=blockIdx.x *blockDim.x+threadIdx.x;
    //进行横向的求和计算,采用的是加上下一个位置数
据,减去上一个的方式来消除冗余计算,输出为每一点的求和值。求和的伪代码如下:
    For 线程对应的这一行每个像素点。
    {
        1,加上下一个像素点,减去头一个像素点
        2,表示的是当前像素点与左右相邻两个点的和
        3,写入 out
    }
}
```

第 2 个核函数以同样的方式沿 y 方向计算各种窗口尺寸的和,并求得对应的均值。

1.2　斑点/组织区域检测

由于计算平均组织强度是针对每个模块内的所有像素,在进行组织强度计算之前必须判决出整幅图像上哪些区域是斑点,哪些是强反射体。此时,采用的是局部变异系数,与方差相比,根据局部变异系数进行区域类型检测的结果更稳定,如下:

$$D(I_{ij})=\frac{E(I_{ij}^2)-E(I_{ij})^2}{E(I_{ij})} \tag{1}$$

其中,$E(I_{ij})$为图像上以(i,j)为中心的局部窗口中的像素均值。

当得到每个像素点的局部变异系数之后,利用预定义阈值就可以识别出斑点噪声、亮反射体和混杂噪声。同时,通过比较 B 模式图像与噪声数据帧,可以将低信噪比区域排除掉,信噪比公式为

$$SNR=20\lg\frac{I}{N} \tag{2}$$

其中,I 和 N 分别表示 B 模式图像和噪声数据帧。一般地,噪声像素点至少比 B 模式图像像素点低 3—6dB。

最后通过方差系数与 B 模式图像的操作,再经过适当的抽取,就可以得到 1 个标记了斑点的模板。

这个处理环节主要有二个计算步骤:一是局部变异系数的计算,二是斑点模板的生成。前者是本算法的核心计算瓶颈之一。从式(1)可以看到,这个计算主要需要计算局部窗口像素均值和像素平方和均值。因此可以借鉴并行均值滤波的设计思路,分 x 方向和 y 方向 2 次完成这一计算过程,只是在计算像素均值的同时,也计算像素平方和的均值即可。

对斑点模板的生成计算,像素点之间的处理是无关的,因此可以直接并行化。在线程结构设计方面,为了避免寻址时的求商求模运算,线程块和线程网格均采用二维结构。对 Fermi 架构的 GPU,在线程块的第 1 个维度设置为 32,即 2 个半 warp 块,对应于图像的横向方向。第 2 个维度是经验测定值,目的在于让活动线程数尽可能地高。因为图像大小一般没有超过 GPU 承载的最大并行计算负荷,所以可以让线程布满整个图像。

1.3　组织强度计算

经过组织/斑点区域检测,就建立了 1 个已经标记好斑点的模板。利用这个模板就可以得到每个图像小网格中斑点噪声像素点的个数,并通过式(3)计算强度,即

$$u_{ij}=\begin{cases}S_{ij} & c>G_s/2\\ E(\boldsymbol{U}) & c=0\\ 1 & \text{else}\end{cases} \tag{3}$$

其中,S_{ij} 为每个网格的像素均值;G_s 为网格大小;c 为网格中斑点噪声像素点的个数;$E(\boldsymbol{U})$ 为当前网格的周围网格的强度。这一个计算步骤可描述如下。

首先,将 B 模式图像分割为许多个小的网格,例如 6×6,计算所有被标记过像素点网格的平均强度值;然后,检查网格的强度是否为 0,如果是则用周围四连通域的网格均值代替。需要注意的是,一些区域可能没有足够满足组织模板的像素点,可用周围区域的加权平均值代替。

组织强度计算有 2 个步骤,一个是网格均值计算,即用 1 个网格的均值代替原先的像素值,并且完成抽取;另一个是组织强度的精化计算。第 1 个步骤的计算过程与均值滤波基本一致,不同之处仅在于这个地方不是逐点求均值而是间隔求均值,也就是抽取。因此在这里,粗粒度的并行方式并不能减少冗余计算,采用的是二次处理细粒度的方式,同样是启动 2 次 GPU 在 x 和 y 方向进行处理,不同的是单个线程负责处理 1 个像素点而不是 1 行或 1 列。

第 2 个步骤是在得到了各个网格基本组织强度的基础上进行精化处理。这里考虑到抽取后的网格数并不多,因此 GPU 设备可以使用细粒度的方式进行并行处理,也就是让 1 个线程负责 1 个网格的精化处理。

1.4　二次曲面拟合

对于已经得到的强度值需要通过 1 个 2 阶的曲面拟合还原回原始数据大小。通过拟合可以减少一定的自由度,从而保证了增益纠正结果是平滑的,使产生不同回声的区域之间的对比度损失减小。二次曲面拟合公式如下

$$\text{fit2}\boldsymbol{D}_{xy}=\sum_{i=0}^{R}\sum_{j=0}^{S}c_{ij}x^iy^j \tag{4}$$

其中,fit2D_{xy}为拟合平面点(x,y)的值;c_{ij}是系数矩阵$\boldsymbol{C}$的第(i,j)元素;R和S分别为x和y的最高次阶数,本文取R=5,S=3。系数矩阵$\boldsymbol{C}$可表示为

$$\boldsymbol{C}=(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B})^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{U}\boldsymbol{G}(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G})^{-1} \quad (5)$$

其中,$\boldsymbol{U}$是第3步已经计算得到的强度矩阵;b_{ij}是矩阵$\boldsymbol{B}$的第(i,j)元素,$b_{ij}=(r+i\times n)^j$,有$0<i<F'_1$,$0<j<C_1$,且F'_1是采样帧的长度,C_1是系数矩阵$\boldsymbol{C}$的列数;g_{pq}是矩阵$\boldsymbol{G}$的第(p,g)元素,$g_{pq}=(r+p\times n)^q$,有$0<p<F'_w$,$0<q<C_r$,且F'_w是采样帧的宽度,C_r是系数矩阵$\boldsymbol{C}$的行数;n是窗口的长度;r是窗口半径。

二次曲面拟合是本算法的另外一个主要的计算瓶颈,所涉及的运算包括矩阵转置、求逆和相乘等。对矩阵乘法,本文通过调用CUBLAS库中矩阵相乘库函数完成[11]。

对矩阵求逆这一运算,本文采用LU分解法,其设计思路是由1个线程负责解1行的线性方程。对一次矩阵求逆,LU分解的结果对所有线程是一致的,也就是说,1次分解,全部线程共用,因此让每个线程块各分解1次,然后通过共享存储器共享。

对矩阵转置,在CUDA上最直接的一种并行矩阵转置方法是让每个线程在输入数据上读取1个位置的数据,然后将这个数据放到转置后的位置。这个做法的问题是如果读取时线程按行读取,就满足合并访问,但写到输入的位置时为按列写,无法实现合并访问。同理,若读取时按列读取,那么尽管写入可以合并,但读取就不合并,因此这种方式不能取得较好的带宽利用率。本文的解决方法是采用共享存储器作为中介体,先将数据从全局存储器进行合并读取,再写入共享存储器,最后再由共享存储器读取出来,即非合并读取,这样可以合并地写入保存结果的全局存储器。

1.5 增益补偿计算

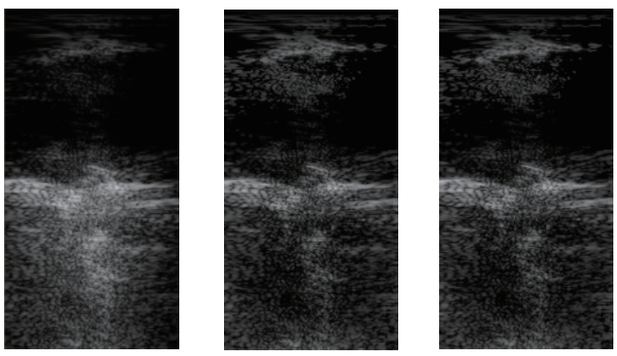
经过前面的步骤已经得到了数据帧中由平滑表面模仿的斑点噪声强度分布。增益修正是通过计算目标软组织强度和拟合值的差异来完成的。但为了使所得到的热噪声在观察时强度适中,局部的增益应仅基于噪声数据帧。局部的噪声平均强度计算以及局部增益的确定都会使无处不在的噪声处于1个预定的灰度水平。在暗区域,斑点噪声的强度需要被增强;而在过度明亮的区域就需要降低。对处于抽取后的网格中的所有点,给予1个最后的增益估计,而最小的增益值(由组织和噪声计算得到的)是可接受的。图像中存在的最大像素值与预先通过自动增益控制得到的总体平均组织水平之间的差异可以用来估计最佳的动态范围。计算这个差异使图像上存在的软组织平均水平外的信号的整个范围被映射到显示屏上并适宜用灰度值填充。一般情况下,软组织映射到的灰度级64是全部灰度区间的25%,并且最大像素值可达255。通过使用1.2节得到的模板识别斑点噪声区域,可以避免在噪声区域出现过量伪增益。增益补偿的目的是使所有区域的斑点噪声有相似的平均亮度。就心脏图像而言,因为近场区域存在空洞,所以图像偏暗;远场指示组织,所以偏亮。因此,增益补偿后,远场的斑点噪声区域强度应当降低,同时保证组织强度不变。本文所使用的自动增益补偿算法可以识别这种情况来自适应调整增益补偿,使远场区域的斑点噪声和组织结构都非常均匀。

在这一步骤的并行实现时,增益计算与补偿的过程在像素点之间是计算无关的,因此可以并行化。主要的处理内容包括完成对平滑曲面的增益计算与补偿。这里需要通过阈值来判决增益补偿的具体实现方式,即比较远场和近场的亮度。这个步骤对所有线程是共享的,为了减少冗余操作,这里的设计是让每个线程块仅计算1次,得到标识位,通过共享存储器让块内所有线程共享。在线程结构设计方面,与前面直接并行化的方式一样。

2 试验结果与分析

本文试验平台为2.81GHz的AMD Althlon (tm) II X2 240,2GB DDR2 RAM,操作系统为Windows XP。GPU为NVIDIA Geforce GTX 560 Ti,显存为1GB,核心频率为1.645GHz,14个多处理器,使用4.0版本的CUDA toolkit及对应的开发包。编程环境为Visual Studio 2010。

为了测试本文提出的并行实现算法的处理效果和运行效率,将由数字超声扫描器在超声系统iMago C21上采集得到的人体数据作为试验数据。图2(a)显示的是由7.5MHz线阵扫描器采集得到的人体心脏的超声图像,图2(b)和图2(c)分别是本文所使用的增益补偿算法的CPU和GPU处理结果。图3(a)显示的是由3.5MHz凸阵扫描器采集得到的人体肝脏组织超声图像,图3(b)和图3(c)分别是增益补偿后CPU和GPU的处理结果。通过对比可知,CPU和GPU的处理结果完全保持一致,误差为0。

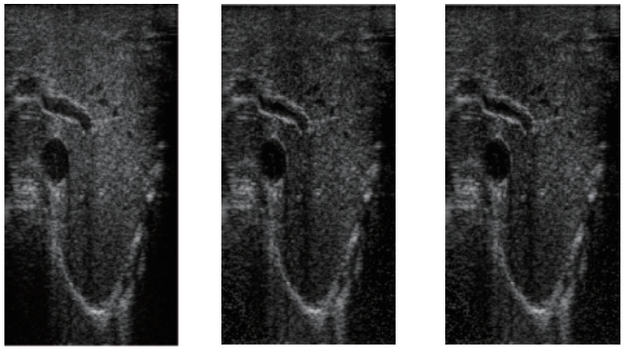


(a) 原图像 (a) Original image　(b) CPU处理结果 (b) Output of CPU　(c) GPU处理结果 (c) Output of GPU

图2 人体心脏的超声图像

Fig. 2 Ultrasound cardiac images

表1给出了本文所研究的自动增益补偿算法CPU和GPU处理的性能比较。程序的运行时间是图像数据进入显存后进行增益补偿处理的全部运行时间。在实际系统中图像信



(a) 原图像　(b) CPU 处理结果　(c) GPU 处理结果
(a) Original image　(b) Output of CPU　(c) Output of GPU

图 3　人体肝脏的超声图像
Fig. 3　Ultrasound liver images

表 1　自动增益补偿算法 CPU 和 GPU 处理性能比较
Table 1　Comparisons of processing performance between automatic time gain compensation algorithm for CPU and GPU

数据规模	CPU 运行时间/ms	GPU 运行时间/ms	加速比
128×261	159.85	0.76	210.33
256×261	311.56	1.28	243.41
512×261	624.76	2.34	266.99

号在本模块处理前已经进入到显存空间;同样,图像被处理后不需要写回主存,可以直接送至下一处理模块,或通过图形学绘制管线直接进行显示,因此这里不包括数据传输的时间。从表 1 可以明显看出,与传统的 CPU 串行处理相比,本文所提出的基于 Fermi 架构 GPU 并行处理算法的速度提高了大约 267 倍。加速比也说明了数据计算规模越密集,GPU 并行加速效果越明显。

3　结论

本文提出了一种基于 Fermi 架构 GPU 的超声图像自动增益补偿并行实现方法,试验结果显示,通过该方法得到的图像增益补偿效果和通用 CPU 处理的结果完全一致,而在时间性能方面达到了实时系统的处理要求,得到大约 267 倍的加速效果。在这样的 GPU 处理效率下,本文研究的自动增益补偿算法不仅可以得到比较好的图像质量,也可以满足临床超声检测系统的实时处理要求。同时为了获得更优的性能,本文在并行算法设计中主要的设计手段包括了尽可能的高并行化、合理高效的线程结构、减少冗余计算和最大化存储器利用率等方面。

参考文献(References)

[1] 李治安. 临床超声影像学[M]. 北京: 人民卫生出版社, 2003: 45-60.
Li Zhi'an. Clinical ultrasound imaging [M]. Beijing: People's Medical Publishing House, 2003: 45-60.

[2] Checkovich P. Time-gain control sharpens ultrasound [J]. *Design News*, 1998, 15(1): 103-104.

[3] He P. Acoustic attenuation estimation for soft tissue from ultrasound echo envelope peaks[J]. *IEEE Trans Ultra Ferroelec Freq Control*, 1989, 36(2): 197-203.

[4] Parker K J. Attenuation measurement uncertainties caused by speckle statistics[J]. *J Acoust Soc Amer*, 1986, 80(3): 727-734.

[5] Kuc R. Bounds on estimating the acoustic attenuation of small tissue region from reflected ultrasound signals [J]. *Proc IEEE*, 1985, 73(7): 1159-1168.

[6] Li X Y, Liu D C. Estimation of local attenuation and its application to rationalized gain control [C]//Proceedings of International Conference on Bioinformatics and Biomedical Engineering. Wuhan: IEEE, 2007: 1521-1524.

[7] Tang M W, Luo F, Liu D C. Automatic time gain compensation in ultrasound imaging system[C]//Proceedings of International Conference on Bioinformatics and Biomedical Engineering. Beijing: IEEE, 2009: 732-735.

[8] NVIDIA Corporation. CUDA programming guide, version 4.0 [M/OL]. [2012 -04 -16]. http://developer.download.nvidia.com/compute/DevZone/docs/html/c/doc/CUDA_C_Programming-Guide.pdf.

[9] 夏春兰, 石丹, 刘东权. 基于 CUDA 的超声 B 模式成像[J]. 计算机应用研究, 2011, 28(6): 2011-2015.
Xia Chunlan, Shi Dan, Liu Dongquan. *Application Research of Computers*, 2011, 28(6): 2011-2015.

[10] 范正娟, 石丹, 刘东权. 基于 CUDA 的超声彩色血流成像 [J]. 计算机应用, 2011, 31(3): 856-859.
Fan Zhengjuan, Shi Dan, Liu Dongquan. *Journal of Computer Applications*, 2011, 31(3): 856-859.

[11] NVIDIA Corporation. CUBLAS library, version 4.0 [M/OL]. [2012-02]. http://developer.download.nvidia.com/compute/DevZone/docs/html/CUDAilibraries/doc/CUBLAS_Library.pdf.

(责任编辑　孙秀云,马骁骁)