(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 104899840 B
(45)授权公告日 2018.12.18

(21)申请号 201510324806.0

(22)申请日 2015.06.12

(65)同一申请的已公布的文献号
申请公布号 CN 104899840 A

(43)申请公布日 2015.09.09

(73)专利权人 天津大学
地址 300072 天津市南开区卫津路92号

(72)发明人 何凯 王新磊 王晓文 葛云峰

(74)专利代理机构 天津市北洋有限责任专利代理事务所 12201
代理人 李林娟

(51)Int.Cl.
G06T 5/00(2006.01)

(56)对比文件
CN 102073982 A,2011.05.25,
CN 103745447 A,2014.04.23,
CN 104050637 A,2014.09.17,
aipiano.“引导滤波的OpenCV实现”.《http://blog.csdn.net/aichipmunk/article/details/21163543》.2014,第1页第2-8段,第2页第1-4段,第3页代码第24-54行.

审查员 谢明远

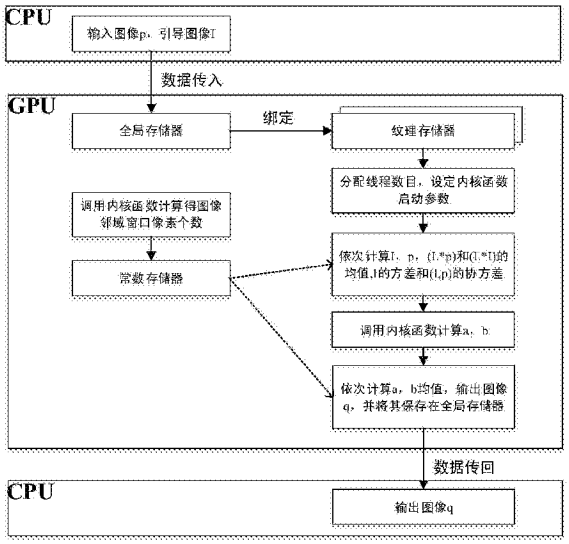
权利要求书2页 说明书8页 附图2页

(54)发明名称

一种基于CUDA的引导滤波加速优化方法

(57)摘要

本发明公开了一种基于CUDA的引导滤波加速优化方法,所述引导滤波加速优化方法包括以下步骤:将输入图像p和引导图像I由主机端内存读入全局存储器,通过构建第一内核函数,分别获取输入图像p、引导图像I、图像I*P、图像I*I在邻域窗口的图像邻域均值;构建第二内核函数依次求取图像(I,p)的协方差,引导图像I的方差,进而求取滤波关键参数a和b;调用第一内核函数获得参数a的邻域均值mean_a,参数b的邻域均值mean_b,进而获取最终滤波结果q,将结果保存到对应的全局存储器,传出到主机端内存。该方法利用GPU浮点计算能力、并行计算等方面的优势,在保证图像滤波效果的同时,有效提高了引导滤波算法的执行效率,快速实现了引导滤波算法。



CN 104899840 B

1.一种基于CUDA的引导滤波加速优化方法，其特征在于，所述引导滤波加速优化方法包括以下步骤：

将输入图像p和引导图像I由主机端内存读入全局存储器，通过构建第一内核函数，分别获取输入图像p、引导图像I、图像I*p、图像I*I在邻域窗口的图像邻域均值；

构建第二内核函数依次求取图像(I,p)的协方差，引导图像I的方差，进而求取滤波关键参数a和b；

调用第一内核函数求取参数a的邻域均值mean_a，参数b的邻域均值mean_b，进而获取最终滤波结果，将结果保存到对应的全局存储器，传出到主机端内存；

实现图像羽化功能时，图像分辨率为946×756，耗时62.3ms，加速度比为60.2；

实现图像平滑功能时，图像分辨率为942×659，耗时21.2ms，加速度比为11.0；

实现图像增强功能时，图像分辨率为1024×1024，耗时59.7ms，加速度比为16.2；

实现flash去噪功能时，图像分辨率为1024×1024，耗时89.7ms，加速度比为10.5；

所述通过构建第一内核函数，分别获取输入图像p、引导图像I、图像I*p、图像I*I在邻域窗口的图像邻域均值的步骤具体为：

将输入图像p、引导图像I、图像I*p、图像I*I在邻域窗口的图像邻域均值的计算分别转换为图像邻域窗口像素值的求和计算；

通过构建第一内核函数分别计算图像邻域窗口像素值的求和；

所述通过构建第一内核函数分别计算图像邻域窗口像素值的求和的步骤具体为：

采用积分图实现邻域窗口像素值求和，通过第一内核函数中4个核函数进行CUDA并行优化；

所述引导滤波加速优化方法还包括：

调用第一内核函数中4个核函数，获取邻域窗口像素个数N，并保存到常数存储器；

所述引导滤波加速优化方法还包括：

采用积分图来实现邻域窗口像素值求和，通过第一内核函数中4个kernel函数进行CUDA并行优化，其具体实现步骤如下：

(i)第1个kernel函数负责并行计算图像第i列从第1行到第j行的像素和，其启动参数为block维度为1024×1，grid维度为1×1；

每个线程通过循环调用完成图像中一列数据的计算，循环中采用寄存器保存中间数据，此时数据读取符合全局存储器合并访问；

(ii)第1个kernel函数产生的数据需要进行数据边界的处理，第2个kernel函数以行为单位处理数据边界问题，启动参数为block维度为16×16，grid维度为((图像宽度+dimBlock.x-1)/dimBlock.x)×((图像高度+dimBlock.y-1)/dimBlock.y)个block，其中dimBlock.x表示线程块在x轴的维度，dimBlock.y表示线程块在y轴的维度；

(iii)第3个kernel函数负责并行计算图像第j行从第1列到第i列的像素和；首先对kernel函数输入数据进行矩阵置换，然后调用第1个kernel函数进行计算，在数据存储时采用按列写存储方式；

(iv)第3个kernel函数产生的数据也需要进行数据边界的处理，第4个kernel函数以列为单位处理数据的边界问题，启动参数与第2个kernel函数相同，此时输出数据为第1个kernel函数输入图像的邻域窗口像素值和，并将其保存到对应的全局存储器；

依次类推，可依次求得引导图像I的邻域均值mean_I，图像I*p的邻域均值mean_Ip，图像I*I的邻域均值mean_II。

一种基于CUDA的引导滤波加速优化方法

技术领域

[0001] 本发明涉及计算机应用技术和图像处理领域，尤其涉及一种基于CUDA(统一计算设备架构)的引导滤波加速优化方法。

背景技术

[0002] 图像滤波是图像处理的重要手段，具有重要的意义和研究价值。由于成像系统、传输介质和记录设备等的不完善，数字图像在其形成、传输记录过程中往往会受到多种噪声的污染。而图像滤波，即在尽量保留图像细节特征的条件下对目标图像的噪声进行抑制，是图像预处理中不可缺少的操作，其处理效果的好坏将直接影响到后续图像处理和分析的有效性和可靠性。

[0003] 图像滤波方法可分为两种：一种是线性移不变滤波，其滤波核权值与输入图像的内容无关，代表为高斯滤波、均值滤波、拉普拉斯滤波等；另一种是线性移变滤波，代表为引导滤波，在滤波过程中需要利用原有图像所包含的的内容信息，称之为引导图信息。双边滤波核函数即考虑了图像模板内像素空间差值的信息，又考虑了像素值差值信息，其中引导图和输入图为同一幅图像，因此双边滤波器可认为是引导滤波的一种简单形式。联合双边滤波器的引导图和输入图不同，可以获得了更理想的滤波效果。但是双边滤波器和联合双边滤波器还存在一些明显的缺陷，如双边滤波器在细节增强和高动态范围图像压缩的应用中，都会出现一个很明显的边缘梯度翻转现象，所以滤波器本身的算法和构造上还有待进一步改进。引导滤波概念自2010年正式提出，其一方面具有双边滤波保边去噪的特点，又克服了伪影的影响，同时由于与拉普拉斯矩阵之间的密切关系，引导滤波在图像去噪，图像增强、HDR(高动态范围图像)压缩、flash/noflash去噪[1]、抠图、去雾和级联采样等领域得到了广泛的应用。该算法简单有效，但需要计算复杂的矩阵和求解大型线性方程组，导致了引导滤波算法耗费了大量的运算时间和空间，无法满足实际应用中的需要。

[0004] 总而言之，引导图像滤波算法计算量较大，很难在保证算法准确性的同时提高算法的执行效率。因此传统的基于CPU的架构在很难满足人们对算法准确性和实时处理的要求，只有采用图像处理器(GPU)去满足实际应用中的需求。

发明内容

[0005] 本发明提供了一种基于CUDA的引导滤波加速优化方法，本发明在保证图像滤波质量的同时，又提高了计算效率，降低了计算复杂度，详见下文描述：

[0006] 一种基于CUDA的引导滤波加速优化方法，所述引导滤波加速优化方法包括以下步骤：

[0007] 将输入图像p和引导图像I由主机端内存读入全局存储器，通过构建第一内核函数，分别获取输入图像p、引导图像I、图像I*P、图像I*I在邻域窗口的图像邻域均值；

[0008] 构建第二内核函数依次求取图像(I,p)的协方差，引导图像I的方差，进而求取滤波关键参数a和b；

[0009] 调用第一内核函数求取参数a的邻域均值mean_a，参数b的邻域均值mean_b，进而获取最终滤波结果，将结果保存到对应的全局存储器，传出到主机端内存。

[0010] 所述通过构建第一内核函数，分别获取输入图像p、引导图像I、图像I*P、图像I*I在邻域窗口的图像邻域均值的步骤具体为：

[0011] 将输入图像p、引导图像I、图像I*P、图像I*I在邻域窗口的图像邻域均值的计算分别转换为图像邻域窗口像素值的求和计算；

[0012] 通过构建第一内核函数分别计算图像邻域窗口像素值的求和。

[0013] 所述通过构建第一内核函数分别计算图像邻域窗口像素值的求和的步骤具体为：

[0014] 采用积分图实现邻域窗口像素值求和，通过第一内核函数中4个核函数进行CUDA并行优化。

[0015] 所述引导滤波加速优化方法还包括：调用第一内核函数中4个核函数，获取邻域窗口像素个数N，并保存到常数存储器。

[0016] 本发明提供的技术方案的有益效果是：

[0017] 本发明在深入研究引导滤波算法的基础上，基于CUDA编程实现引导滤波算法，在图像平滑、图像羽化、图像增强与图像flash去噪四个实例方面，与基于C程序和Matlab程序进行实验对比。本发明与现有技术相比的优点在于：

[0018] (1)思路新颖，利用CUDA架构进行引导滤波算法设计，突破串行编程的时间限制，具有较大创新意义。

[0019] (2)执行效率高，在一定程度上可达到实时处理。该方法利用GPU浮点计算能力、并行计算等方面的优势，在保证图像滤波效果的同时，有效提高了引导滤波算法的执行效率，快速实现了引导滤波算法。

[0020] (3)实现简单，硬件要求低，在C语言环境下完成对GPU并行架构的调用，代码编写容易，同时在消费级的GPU硬件上就可实现大规模数据的处理。

附图说明

[0021] 图1本发明引导滤波算法流程图；

[0022] 图2图像平滑效果对比图；

[0023] (a)输入图像，(b)C程序输出图像，(c)CUDA程序输出图像；

[0024] 图3图像羽化效果对比图；

[0025] (a)输入图像，(b)引导图像，(c)C程序输出图像，(d)CUDA程序输出图像；

[0026] 图4本发明图像增强效果对比图；

[0027] (a)输入图像，(b)C程序输出图像，(c)CUDA程序输出图像；

[0028] 图5本发明图像flash去噪效果对比图。

[0029] (a)输入图像，(b)引导图像，(c)C程序输出图像，(d)CUDA程序输出图像。

具体实施方式

[0030] 为使本发明的目的、技术方案和优点更加清楚，下面对本发明实施方式作进一步地详细描述。

[0031] 近十年来，计算机图形处理器(Graphics Pocessing Unit,GPU)由原本只是处理

计算机图形的专用设备发展成为高并行度、多线程、多核的处理器。目前主流GPU的运算能力早已超过主流通用CPU，从发展趋势上看将来差距会越来越大。统一计算设备架构是由NVIDIA(英伟达)公司推出的一种将GPU作为数据并行计算的软硬件架构体系，它是一个完整的GPGPU(通用图形处理器)解决方案。CUDA的出现降低了程序员使用GPU进行通用计算的开发难度。由于CUDA特殊的编程模型以及存储数据方法，使得大量而复杂的同类运算可以由线程同时处理，大大减少了程序的执行时间。

[0032] 为此，本发明提出一种基于CUDA的引导滤波加速优化方法，利用CUDA构建CPU和GPU协同工作环境，将CPU作为负责进行逻辑性强的事务处理和串行计算的主机，将GPU作为负责执行高度线程化并行处理的协处理器，利用CUDA并行编程实现图像邻域窗口像素值求和，进而获得图像邻域均值，同时利用寄存器和纹理存储器，优化算法步骤，获得引导滤波关键参数，进而实现了对算法的整体优化。本发明的技术方案如下：

[0033] 实施例1

[0034] 一种基于CUDA的引导滤波加速优化方法，参见图1，引导滤波加速优化方法包括以下步骤：

[0035] 101：将输入图像p和引导图像I由主机端内存读入全局存储器，通过构建第一内核函数，分别获取输入图像p、引导图像I、图像I*P、图像I*I在邻域窗口的图像邻域均值；

[0036] 102：构建第二内核函数依次求取图像(I,p)的协方差，引导图像I的方差，进而求取滤波关键参数a和b；

[0037] 103：调用第一内核函数求取参数a的邻域均值mean_a，参数b的邻域均值

[0038] mean_b，进而获取最终滤波结果，将结果保存到对应的全局存储器，传出到主机端内存。

[0039] 其中，通过构建第一内核函数，分别获取输入图像p、引导图像I、图像I*P、图像I*I在邻域窗口的图像邻域均值的步骤具体为：

[0040] 将输入图像p、引导图像I、图像I*P、图像I*I在邻域窗口的图像邻域均值的计算分别转换为图像邻域窗口像素值的求和计算；

[0041] 通过构建第一内核函数分别计算图像邻域窗口像素值的求和。

[0042] 进一步地，通过构建第一内核函数分别计算图像邻域窗口像素值的求和的步骤具体为：

[0043] 采用积分图实现邻域窗口像素值求和，通过第一内核函数中4个核函数进行CUDA并行优化。

[0044] 该引导滤波加速优化方法还包括：

[0045] 调用第一内核函数中4个核函数，获取邻域窗口像素个数N，并保存到常数存储器。

[0046] 本方法利用CUDA编程对引导滤波算法进行并行优化，在保证滤波结果效果的同时，又能大大提高引导滤波算法的执行效率，在一定程度上实现了引导滤波算法的实时处理。

[0047] 下面结合具体的计算公式、计算步骤对实施例1中的方法进行描述，详见下文描述：

[0048] 实施例2

[0049] 引导滤波算法是基于一个局部线性模型来实现的，在局部线性模型中，设输入图

像为p，引导图像为I，滤波输出图像为q，局部线性模型假设在以中心像素k的邻域窗口 ω_k 存在如下线性关系：

[0050] $$q_i = a_k I_i + b_k, \forall i \in \omega_k \quad (1)$$

[0051] 其中，ω_k 是以边长为r的方形窗口，a_k 和 b_k 是邻域窗口 ω_k 中的线性系数，I_i 为引导图像在邻域窗口 ω_k 中的像素值，q_i 为邻域窗口 ω_k 中的滤波输出。系数 a_k 和 b_k 可通过求取输入图像p和输出图像q的最小化差值来确定，即使得式(2)达到最小。

[0052] $$E(a_k, b_k) = \sum_{i \in \omega_k} [(a_k I_i + b_k - p_i)^2 + \varepsilon a_k^2] \quad (2)$$

[0053] 式(2)中 $E(a_k, b_k)$ 为邻域窗口 ω_k 中的代价函数的输出，p_i 为输入图像在邻域窗口 ω_k 中的像素值，ε为一个惩罚性的方差调整参数，其目的是防止 a_k 取值过大。线性回归求解上式可得：

[0054] $$a_k = \frac{\frac{1}{|\omega|}\sum_{i \in \omega_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \varepsilon} \quad (3)$$

[0055] $$b_k = \bar{p}_k - a_k \mu_k \quad (4)$$

[0056] 式中，μ_k 和 σ_k^2 分别是引导图像I在邻域窗口 ω_k 的均值和方差。$|\omega|$ 为邻域窗口 ω_k 中的像素个数，$\bar{p}_k$ 是输入图像p在邻域窗口 ω_k 中的均值。

[0057] 由于每一个像素点会包含在多个邻域窗口 ω_k 中，在不同邻域窗口 ω_k 中计算得到的 q_i 也不同，故而需要对 q_i 进行平均处理，通过计算所有窗口中的 a_k 和 b_k，滤波输出如式(5)。

[0058] $$q_i = \frac{1}{|\omega|}\sum_{k | i \in \omega_k} (a_k I_i + b_k) = \bar{a}_i I_i + \bar{b}_i$$

[0059] (5)

[0060] 其中，$\bar{a}_i = \frac{1}{|\omega|}\sum_{k \in \omega_k} a_k$，$\bar{b}_i = \frac{1}{|\omega|}\sum_{k \in \omega_k} b_k$，$\bar{a}_i$，$\bar{b}_i$ 分别为 a_k，b_k 在点i处的所有重叠邻域窗口的平均值。

[0061] 通过对式(3)、式(4)分析可知，μ_k、$\bar{p}_k$、$\frac{1}{|\omega|}\sum_{i \in \omega_k} I_i p_i$ 分别代表引导图像I、输出图像p、I×p在其邻域窗口 ω_k 中的均值，σ_k^2 是I在邻域窗口 ω_k 中的方差。方差与均值之间存在 $DX = E(X^2) - (EX)^2$ 的关系，可利用均值进行计算。故在引导滤波算法中，图像邻域均值需要反复计算，是整个算法中最耗时的部分，因此，如何快速求取图像在某点邻域窗口中的图像邻域均值，就成为实现引导滤波算法的一个关键，也是本发明CUDA优化的一个重点环节。

[0062] 本发明采用公式(6)构建第一内核函数，实现图像领域均值的计算。

[0063] mean_p=boxfilter(p,r)/N (6)

[0064] 其中，mean_p代表输入图像p的邻域均值，boxfilter(p,r)代表输入图像p在邻域窗口中像素值之和，N代表邻域窗口中像素个数，r代表邻域窗口边长。其中邻域窗口像素个

数N,可通过对与所求图像大小相同的全1矩阵求邻域窗口像素和得到。该计算步骤为本领域技术人员所公知,本发明实施例对此不做赘述。

[0065] 采用上述方法,可将图像邻域均值的计算转变为图像邻域窗口像素值的求和计算,便于进行CUDA并行处理。本发明采用积分图来实现邻域窗口像素值求和,通过第一内核函数中4个kernel函数进行CUDA并行优化,其具体实现步骤如下(假设所用数据已位于GPU显存中):

[0066] (i)第1个kernel函数负责并行计算图像第i列(1≤i≤图像宽度)从第1行到第j(1≤j≤图像高度)行的像素和,其启动参数为block维度为1024×1,grid维度为1×1。每个线程通过循环调用完成图像中一列数据的计算,循环中采用寄存器保存中间数据,此时数据读取符合全局存储器合并访问。

[0067] (ii)第1个kernel函数产生的数据需要进行数据边界的处理,第2个kernel函数以行为单位处理数据边界问题,启动参数为block维度为16×16,grid维度为((图像宽度+dimBlock.x-1)/dimBlock.x)×((图像高度+dimBlock.y-1)/dimBlock.y)个block。其中dimBlock.x表示线程块在x轴的维度,dimBlock.y表示线程块在y轴的维度。

[0068] (iii)第3个kernel函数负责并行计算图像第j行(1≤j≤图像高度)从第1列到第i列(1≤i≤图像宽度)的像素和。为消除数据读取时非合并访问的制约,采用首先对此kernel函数输入数据进行矩阵置换,然后调用第1个kernel函数进行计算,在数据存储时采用按列写存储方式。

[0069] (iv)第3个kernel函数产生的数据也需要进行数据边界的处理,第4个kernel函数以列为单位处理数据的边界问题,启动参数与第2个kernel函数相同,此时输出数据为第1个kernel函数输入图像的邻域窗口像素值和,并将其保存到对应的全局存储器。

[0070] 依次类推,可依次求得引导图像I的邻域均值mean_I,图像I*P的邻域均值mean_Ip,图像I*I的邻域均值mean_II。

[0071] 这里值得说明的是,CUDA的编程模型是CPU与GPU协同工作。传统的CPU架构受其硬件架构的影响不能有效的利用资源进行通用计算,而利用CUDA可以使GPU不仅能执行传统的图形计算,还能高效的执行通用计算。为了尽可能减少数据传输耗时,提高运算速度,本发明设定CPU内存和GPU显存之间只进行2次数据传输,即输入图像p和引导图像I由主机端内存传入设备端显存,以及输出图像q由设备端显存传入主机端内存,其具体步骤如下:

[0072] (i)利用CUDA构建CPU和GPU协同工作环境;

[0073] (ii)将输入图像p和引导图像I由主机内存读入设备显存的全局存储器,并绑定到纹理存储器。

[0074] (iii)分配线程数目,设定内核启动参数为每个block分配16×16,每个grid有((图像宽度+dimBlock.x-1)/dimBlock.x)×((图像高度+dimBlock.y-1)/dimBlock.y)个block,将图像进行棋盘划分。其中dimBlock.x表示线程块在x轴的维度,dimBlock.y表示线程块在y轴的维度。

[0075] (iv)调用第一内核函数(即包括4个核函数),通过对与所求图像大小相同的全1矩阵求邻域窗口像素和,得到邻域窗口像素个数N,并将其保存到常数存储器。

[0076] (v)调用第一内核函数及常数存储器中N依次求取输入图像p和引导图像I的邻域均值,图像I*P的邻域均值mean_Ip,图像I*I的邻域均值mean_II,并依次将结果保存在对应

的全局存储器。

[0077] (vi)构建第二内核函数依次求取图像(I,p)的协方差cov_Ip,图像I的方差var_I,进而构建内核函数求取滤波关键参数a和b。

[0078] 即,根据公式cov_Ip=mean_Ip-mean_I.*mean_p构建协方差内核函数求取图像(I,p)的协方差;

[0079] 根据公式var_I=mean_II-mean_I.*mean_I构建方差内核函数求取图像I的方差;

[0080] 根据公式a=cov_Ip./(var_I+ε)构建参数a内核函数求取滤波关键参数a;

[0081] 根据公式b=mean_p-a.*mean_I构建参数b内核函数求取滤波关键参数b。

[0082] (vii)调用第一内核函数对滤波关键参数a和b求邻域窗口均值,并求得最终滤波结果q,并将结果保存到对应的全局存储器。

[0083] 即,根据公式mean_a=boxfilter(a,r)/N,调用第一内核函数求得关键参数a的邻域均值;

[0084] 根据公式mean_b=boxfilter(b,r)/N,调用第一内核函数求得关键参数b的邻域均值;

[0085] 根据公式q=mean_a.*I+mean_b构建输出q内核函数求得最终滤波结果q,并将结果保存到对应的全局存储器。

[0086] (viii)将保存在设备显存的全局存储器中的滤波结果图像传出到主机内存。

[0087] 此外,引导滤波算法在实现图像羽化算法时,与上述流程不同:

[0088] (i)输入图像p和引导图像I的r、g、b分量数据由CPU的内存拷入到GPU的显存时,由于涉及多次数据传输,本发明采用CUDA流,这样数据赋值操作和核函数执行交叉进行时,可提高GPU资源的使用率;特别是当数据量较大时,CUDA流的优势比较明显;

[0089] (ii)在求解关键参数a时,本发明采用一个kernel函数来实现a中3个分量r、g、b的计算,其启动参数设置为block维度为16×16,block以二维地址排列。首先每个线程依次将全局存储器var_I_rr,var_I_rg,var_I_rb,var_I_gg,var_I_gb,var_I_b中的数据保存到寄存器,构建3×3的Sigma矩阵,并利用行列式计算公式求解,将结果存入寄存器;其次每个线程将Sigma矩阵求逆,以及cov_Ip矩阵与该逆矩阵相乘统一计算,以增加程序执行的计算密集度,充分利用GPU的计算性能。

[0090] 实施例3

[0091] 为使本发明的目的、技术方案和优点更加清楚,下面将结合具体的实例对本发明的技术方案做进一步详细描述。

[0092] 本发明实例采用windows 7操作系统,CPU为Intel Core i5-3470,主频为3.2GHz,系统内存为4GB;GPU为NVIDIA GeForce GTX660,其中包括了5个流多处理器(SMS),每个SMS含有192个CUDA核,板载全局内存为2048Mbytes,内存带宽为192比特,支持支持CUDA Compute Capability为3.0。同时本发明利用CUDA Toolkit自带的Visual Profile来分析各项数据,实现对程序性能的量化分析。

[0093] 为验证本方法的有效性,本发明实例对引导滤波算法在图像平滑、图像羽化、图像增强和flash去噪等4个应用领域进行CUDA并行优化,其滤波效果图和加速比表如下:

[0094] 实例1 图像平滑

[0095] 本次实例中,滤波半径r为16,滤波参数eps为0.04。输入图像p和引导图像I设为同

一幅图像，输出图像q为最终输出结果，实例1效果如图2所示。从图2中可以看出：输入图像p中的细节、突变、边缘和噪声都得到了一定程度的抑制，获得了比较理想的图像平滑效果。

[0096] 实例2 图像羽化

[0097] 本次实例中，滤波半径r为60，滤波参数eps为0.000001。输入图像p和引导图像I设为两幅不同的图像，输出图像q为最终输出结果，实例2效果如图3所示。从图3中可以看出：输出图像羽化效果明显，边缘部分实现了渐近变化，达到了自然衔接的效果。

[0098] 实例3 图像增强

[0099] 本次实例中，滤波半径r为16，滤波参数eps为0.01。输入图像p和引导图像I设为同一幅图像，输出图像q为最终输出结果，实例3效果如图4所示。由图4可以看出：输出图像的整体或局部特征都得到了明显的增强，有效提高了图像细节部分的辨识能力。

[0100] 实例4 flash去噪

[0101] 本次实例中，滤波半径r为8，滤波参数eps为0.0004。输入图像p和引导图像I设为两幅不同的图像，输出图像q为最终输出结果，实例4效果如图5所示。由图5可以看出：输出图像q着色去噪效果自然协调，得到了理想的处理效果。

[0102] 除此之外，从图2～5中可以看出，本发明在平滑、羽化、增强、flash去噪4个方面都与基于C程序的原算法效果基本相同，证明了本发明的准确性。为了比较本发明的加速效果，分别基于Matlab编程、基于C程序和CUDA编程实现引导滤波算法，并进行了实验对比。不同分辨率图像处理的时间消耗及加速比如表1所示：

[0103] 表1 基于不同程序编程实现引导滤波算法时间消耗(ms)及加速比

[0104]

实现功能	图像分辨率	Matlab程序	C程序	CUDA程序	C/CUDA加速比
图像平滑	276×276	25.1	24.2	8.2	2.9
	512×640	156.4	141.6	15.1	9.3
	942×659	241.2	231.9	21.2	11.0
图像羽化	556×568	4275.7	1125.2	37.1	30.4
	886×542	6489.2	2251.2	49.9	45.9
	946×756	9651.4	3736.1	62.3	60.2
图像增强	640×480	310.7	290.6	40.4	7.2
	800×960	834.2	741.1	53.1	13.9
	1024×1024	1120.1	971.7	59.7	16.2
flash去噪	512×512	267.8	251.2	54.3	4.6
	820×546	486.2	470.7	69.1	6.8
	1024×1024	1016.1	934.3	89.7	10.5

[0105] 从表1中可以看出，相比于基于Matlab程序和C程序实现引导滤波算法，本发明基于CUDA并行实现的时间消耗大大缩短；其中，图像羽化的加速效果尤为明显，可以实现60多倍的加速比；同时也可以看出，随着图像分辨率的不断增加，本发明的加速效果也越明显。

[0106] 参考文献：

[0107] [1]Petschnigg G,Szeliski R,Agrawala M,et al.Digital photography with

flash and no-flash image pairs[J].ACM transactions on graphics(TOG),2004,23(3):664-672.

[0108] 本领域技术人员可以理解附图只是一个优选实施例的示意图，上述本发明实施例序号仅仅为了描述，不代表实施例的优劣。

[0109] 以上所述仅为本发明的较佳实施例，并不用以限制本发明，凡在本发明的精神和原则之内，所作的任何修改、等同替换、改进等，均应包含在本发明的保护范围之内。

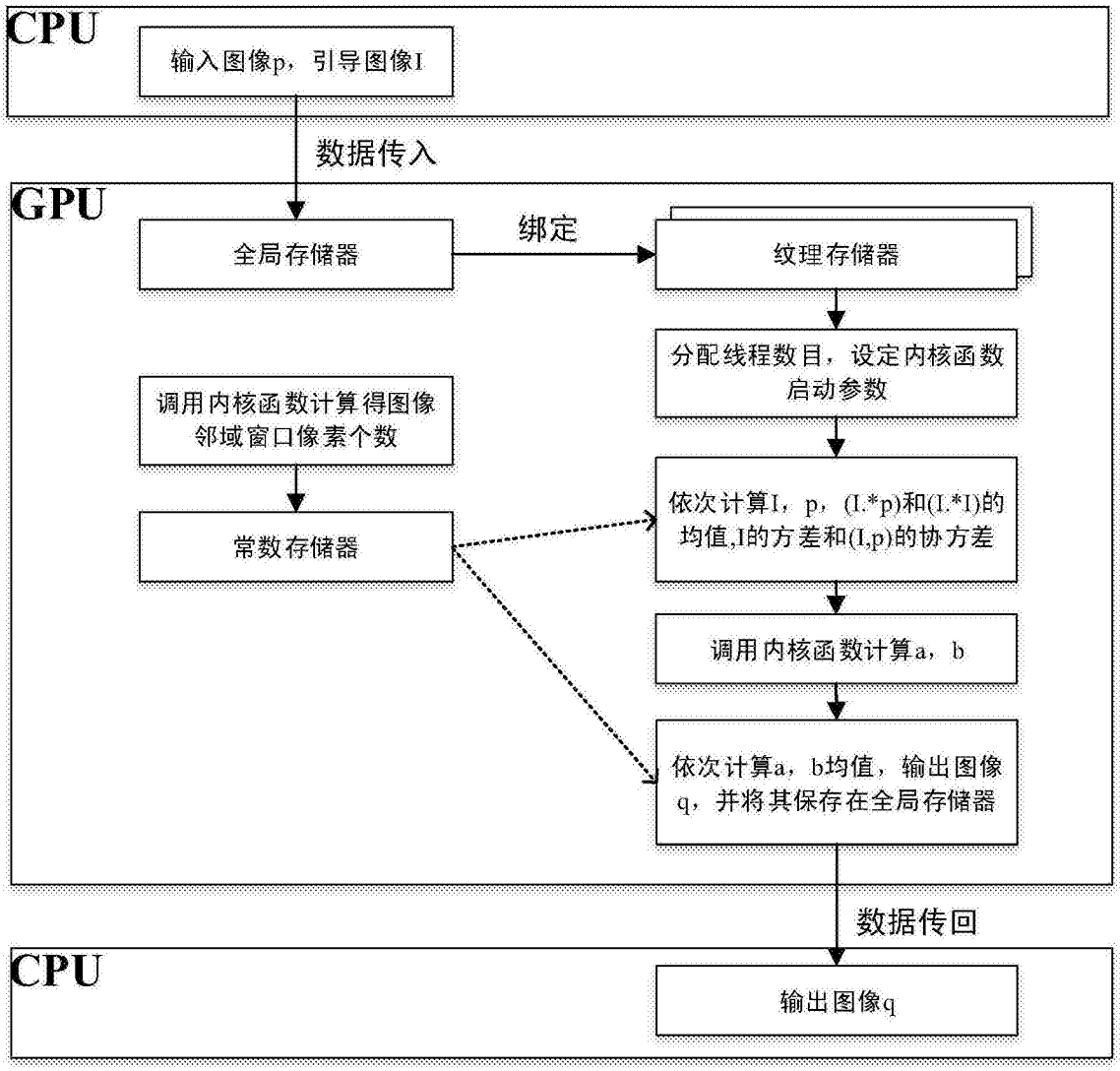

CPU
输入图像p，引导图像I
数据传入
GPU
全局存储器
绑定
纹理存储器
分配线程数目，设定内核函数启动参数
调用内核函数计算得图像邻域窗口像素个数
常数存储器
依次计算I，p，(I.*p)和(I.*I)的均值,I的方差和(I,p)的协方差
调用内核函数计算a，b
依次计算a，b均值，输出图像q，并将其保存在全局存储器
数据传回
CPU
输出图像q

图1

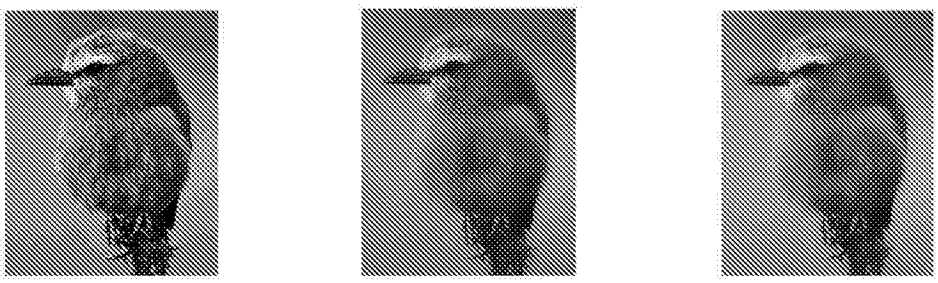

(a)
(b)
(c)

图2

图3

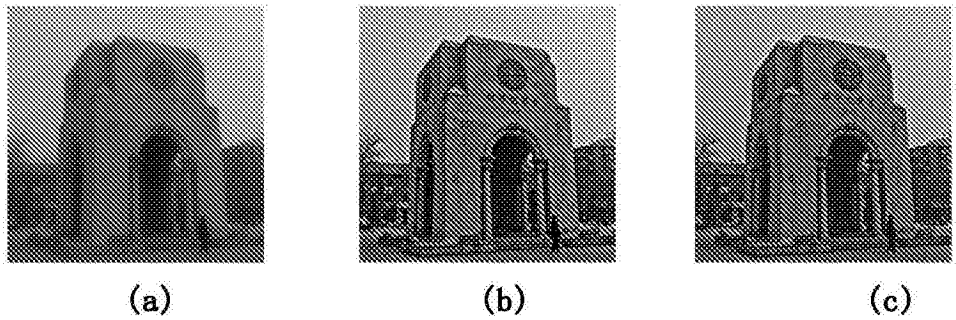


图4

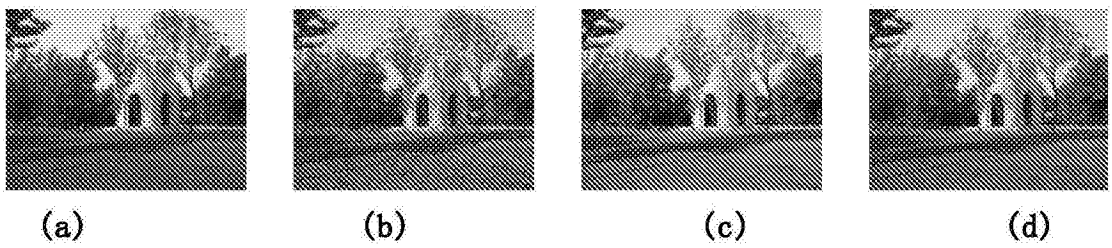


图5