

基于梯度塔形分解的 GPU 加速多源图像融合

潘晓辉¹, 张 涛²

(1 上海政法学院 现代教育技术中心, 上海 201701; 2 上海交通大学 计算机科学与工程系, 上海 200240)

摘 要: 为了解决多分辨率图像融合算法在单纯 CPU 上运行效率低的问题, 提出了一种用于在基于梯度塔形分解的图像融合算法中使用的新融合规则, 并将整个融合算法在图形处理器上进行了设计和高效的实现, 将计算密集的任务安排到图形处理器上执行. 实验验证, 融合图像保留了源图像中的显著特征. 并且, 与单纯 CPU 上的融合算法相比, 该系统在不同尺寸的图像融合中都获得加速. 展现了一种通过图形处理器提高多分辨率图像融合算法效率的方法.

关键词: 图像融合; 梯度塔形分解; 图形处理器; 融合规则

中图分类号: TP391.9

文献标识码: A

文章编号: 1000-7180(2011)12-0054-05

GPU-accelerated Image Fusion Based on Gradient Pyramid Decomposition

PAN Xiao-hui¹, ZHANG Tao²

(1 Modern Education Technology Center, Shanghai University of Political Science and Law, Shanghai 201701, China;
2 Computer Science and Engineering Department, Shanghai Jiaotong University, Shanghai 200240, China)

Abstract: To address the low-efficiency issue of the traditional multi-resolution image fusion algorithms executing on CPUs, a novel image fusion rule for gradient-pyramid-decomposition based image fusion algorithms is proposed along with the design and efficient implementation of the entire fusion algorithm on Graphics Processing Units (GPUs). The computation-intensive tasks are offloaded from the CPU to the GPU. In the experiments, the fused image preserves the salient features of the source images. Moreover, the GPU accelerated fusion system gains speedups on images of different dimensions versus the pure CPU implementation. Thus an approach to improve the efficiency of multi-resolution image fusion algorithms is presented.

Key words: image fusion; gradient pyramid decomposition; graphics processing unit; fusion rule

1 引言

近年来, 传感器件的发展很快, 图像的时空分辨率快速提高, 使得图像融合的数据量持续增长, 融合系统的效率难以保证^[1]. 需要更研究高效的融合算法, 或者设计更大容量的存储和强劲的计算设备. 图形处理器 (Graphics Processing Unit, GPU), 它具备强大而灵活, 可编程的多线程流处理器, 提供高效的计算能力以及很高的内存带宽^[2]. 图像处理器的强大能力和编程灵活性使得它们成为了图像处理以及通用计算的计算密集型任务的理想平台. 根据硬

件的性能和应用的模式, 基于 GPU 实现一般可比单纯 CPU 的实现十几倍甚至上百倍的加速. 在基于塔形分解的图像融合中, 融合算法在多幅图像的所有像素或者区域上迭代地进行处理, 因此这部分工作很适合分配到 GPU 上并行地处理. 本文提出了一种自适应式的基于区域标准差的融合准则, 并根据 NVIDIA 最新的 Fermi GPU 架构设计实现了基于塔形分解的图像融合系统. 实验证明, 此融合系统可以融合多源图像得到较好的结果图像, 比单纯 CPU 的实现有较大的加速比. 证明了新融合准则以及整个系统的有效性.

2 图形处理器架构

Fermi 是 NVIDIA 于 2010 年 3 月发布的最新一代的 GPU 架构. GeForce GTX480 是 GeForce 系列的最强大的 Fermi 架构图形处理器之一. 图 1 描述了 GTX480 图形处理器的架构. 它由 480 个 CUDA (Compute Unified Device Architecture, 计算统一设备架构) 处理器构成, 核心频率 1.4 GHz. 每个 CUDA 处理器具备一个全流水的整数算术逻辑单元(Integer Arithmetic Logic Unit, ALU)和一个浮点数处理单元(Floating Point Unit, FPU). 这 480 个 CUDA 处理器被均分到 15 个流多处理器(Streaming Multiprocessors, SMs)中, 因此每个流多处理器具备 32 个 CUDA 处理器. 每个流多处理器具备 2 套指令系统, 每套指令系统对应 16 个 CUDA 处理器. 在每个时钟周期, 这两套指令系统从就绪的指令中选出两条, 并分别发射给 16 个 CUDA 处理器, 或者 16 个存储(load/store)单元, 或者 4 个特殊功能单元(Special Function Units, SFUs), 即双发射^[3]. 所有流处理器上运行同一个程序, 称作 Kernel.

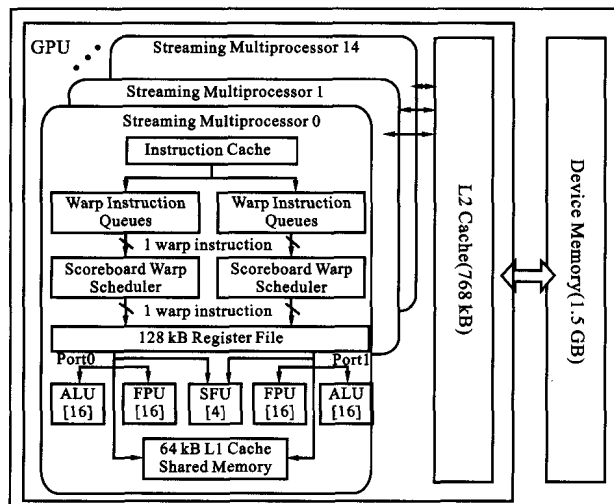


图 1 GTX480 架构

每个流多处理器具备 128 kB 的寄存器及 64 kB 的共享内存(shared memory)和 L1 缓存(cache). 除了每个流多处理器各自的 L1 缓存之外, 整个芯片还有 768 kB 的 L2 缓存. 此外, 还有 1.5 GB 的 GDDR5 的片外全局内存(也称设备内存), 可以 177.4 GB/s 的速度访问. 共享内存只能被其所在的流多处理器上的线程访问, 而全局内存可被 GPU 上运行的所有线程访问. 需要指出, 共享内存的访问延迟(约 38 时钟周期)比全局内存的访问延迟(约

440 时钟周期)小得多, 因此应当把经常要访问的数据存储到共享内存中.

NVIDIA 提供 CUDA (Compute Unified Device Architecture, 计算统一设备架构) 编程模型和软件环境. CUDA C 是对标准 C 语言的一个扩展. 在 CUDA 编程模型中, 一个程序的所有线程被分为多个块(Block), 每个块被以 Round-Robin 的方式调度到一个空闲的流多处理器上执行. 每个块又由多个线程束(Warp)组成, 每个线程束包含 32 个线程. 当被调度到流多处理器上执行的时候, 块中的线程束被称作活动线程束. 如果流多处理器上还可以容纳多个块, 那么这些块被称作活动块. 每个 GTX480 的流多处理器可以容纳最多 8 个活动块, 或者 48 个活动线程束, 或者 1 536 个并发线程. 更多的并发线程不仅能提高并行度, 而且能掩盖访存等的延迟. 然而, 并发线程数是受硬件资源, 特别是片上的寄存器和共享内存的限制.

3 基于梯度分解的图像融合算法

在多分辨率图像融合算法中, 基于塔形分解各种算法都得到了广泛使用^[4]. 其中, 基于梯度塔形分解的图像融合算法具有方向性^[5], 因此可以比其他的塔形分解方法保留更多诸如图像中的边缘等的信息. 因此我们的研究集中在这个算法上. 基于梯度分解的图像融合算法由下面五步构成:

(1) 用 Reduce 操作构建高斯金字塔(Gaussian Pyramid).

(2) 对高斯金字塔的每层图像施加 4 个方向的梯度算子, 得到梯度金字塔.

(3) 对源图像 A 及源图像 B 的梯度金字塔 GLA 及 GLB 的每层每方向的图像根据融合规则进行融合, 得到一个融合后的梯度金字塔 GLC.

(4) 将融合后的梯度金字塔 GLC 每层的 4 个方向的图像合并为 1 个, 得到拉普拉斯金字塔(Laplacian Pyramid)LC.

(5) 对拉普拉斯金字塔施加 EXPAND 操作和加操作, 重构最终的输出图像.

3.1 构造高斯金字塔

图像的高斯金字塔是一个图像序列, 每一幅图像即是对前驱图像施加低通滤波器的结果. 将源图像记作 G_0 , 即为高斯金字塔的第 0 级. 构造高斯金字塔的第一步即为将 G_0 和一个高斯 Kernel ω 进行卷积, 然后进行降采样. 降采样时, 每隔一行及一列进行, 所以新的图像将是原图像的宽度和高度的一半.

简单来说, 高斯金字塔的第 l 层如下构造:

$$G_l = \sum_{m=-2}^2 \sum_{n=-2}^2 \omega(m, n) G_{l-1}(2i+m, 2j+n) \quad (0 < l \leq N, 0 \leq i \leq C_l, 0 \leq j \leq R_l) \quad (1)$$

式中, N 代表金字塔的顶层, C_l 代表第 l 层的总列数, R_l 代表第 l 层的总行数, $\omega(m, n)$ 代表一个 5×5 的高斯核(kernel), 如下:

$$\omega(m, n) = \frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \quad (2)$$

这个卷积和降采样的操作也被称作是 REDUCE 操作, 即:

$$G_l = \text{Reduce}(G_{l-1}) \quad (3)$$

REDUCE 操作的逆操作是 EXPAND 操作, 是将 G_{l+1} 的图像按行和列复制扩展, 并卷积同一个高斯核 ω 以得到一个新的图像 G_l^* . 此图像的尺寸和 G_l 是一样的:

$$G_l^* = \text{Expand}(G_l) \quad (4)$$

EXPAND 操作的数学定义如下:

$$G_l^* = 4 \sum_{m=-2}^2 \sum_{n=-2}^2 \omega(m, n) G'_l\left(\frac{i+m}{2}, \frac{j+n}{2}\right) \quad (0 < l \leq N, 0 \leq i \leq C_l, 0 \leq j \leq R_l) \quad (5)$$

其中 $G'_l\left(\frac{i+m}{2}, \frac{j+n}{2}\right)$ 为

$$G'_l\left(\frac{i+m}{2}, \frac{j+n}{2}\right) = \begin{cases} G_l\left(\frac{i+m}{2}, \frac{j+n}{2}\right), & \text{当 } \frac{i+m}{2} = \frac{j+n}{2} = 0 \\ 0, & \text{其他} \end{cases} \quad (6)$$

3.2 构造梯度金字塔

用 4 个方向的梯度算子对图像的高斯金字塔的各层图像(除了顶层图像)进行滤波, 就得到图像的梯度金字塔分解:

$$GP_k = d_k * (G_l + \hat{\omega} * G_l) \quad (0 \leq l < N, k = 1, 2, 3, 4) \quad (7)$$

式中, GP_k 代表第 l 层第 k 方向的梯度金字塔图像, G_l 代表图像高斯金字塔的第 l 层的图像, d_k 代表梯度算子, $k = 1, 2, 3, 4$ 分别代表水平方向、45 度方向、垂直方向、135 度方向, $*$ 代表卷积操作, $\hat{\omega}$ 代表如下的 3×3 的 kernel 函数:

$$\hat{\omega} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad (8)$$

梯度算子 d_k 定义如下:

$$\begin{aligned} d_1 &= \begin{bmatrix} 1 & -1 \end{bmatrix} \\ d_2 &= \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \\ d_3 &= \begin{bmatrix} -1 \\ 1 \end{bmatrix} \\ d_4 &= \frac{1}{\sqrt{2}} \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \end{aligned} \quad (9)$$

3.3 融合规则

融合规则决定了分解后的金字塔的各层或者子图像如何融合到一起. 在多分辨率图像融合过程中, 融合规则的选择对于融合的质量至关重要, 它的选择直接影响着融合的效果. 在图像分解后, 低频部分集中了原图像的主要能量, 而高频部分反映了原图像的亮度突变特性, 即边缘、区域边界特性. 需要融合规则能够保留原图像的这些最显著信息. 目前广泛使用的融合规则主要有两点取最大值、基于区域能量的方法^[5]、基于统计排序滤波器等. 两点取最大值的融合规则简单, 但是容易丢失细节. 基于区域能量的方法效果较好, 但计算量偏大. 基于统计排序滤波器方法容易受噪声干扰. 在这里, 我们提出一种自适应的基于局部方差的融合规则.

在统计中, 方差用于度量一系列数值相互之间的偏离程度. 在数字图像中, 一个局部区域的方差可表示此区域中的像素值的变化和分散程度. 方差越大, 则说明变化越多, 越分散. 那么这个区域所包含的诸如边缘等的图像细节信息就越多. 因此, 方差可被用于制定融合规则. 我们提出的具体规则如下:

设图像 LA 和 LB 将融合为 LC, 那么在 LA 和 LB 上迭代的处理每个像素 $P_{LC}(x, y)$. 以此像素为中心选取矩形窗口如 $3 \times 3, 5 \times 5$ 等, 则所选窗口的方差计算如下:

$$\text{VAR} = \frac{1}{M \times N} \sum_{i=1}^M \sum_{j=1}^N (x_{i,j} - \bar{x})^2 \quad (10)$$

式中, M, N 分别代表此矩形窗口的行数和列数, $x_{i,j}$ 为此窗口中第 i 行, 第 j 列像素的灰度值, 而 $\bar{x}$ 为此窗口中所有像素的平均灰度值.

定义一个比例阈值 T (通常为 $1.0 \sim 5.0$). 定义 $\text{VAR}_{\max}$ 为 VAR_{LA} 和 VAR_{LB} 中的最大值, 而 $P_{\max}(x, y)$ 为具备这个最大值的窗口的中心像素的灰度值. 对应的, 定义 $\text{VAR}_{\min}$ 为 VAR_{LA} 和 VAR_{LB} 中的最小值, 而 $P_{\min}(x, y)$ 为具备这个最小值的窗口的中心像素的灰度值. 则我们提出的自适应的融合规则如下:

$$P_{LC}(x, y) = \begin{cases} P_{\max}(x, y), & \text{当 } R_{\text{var}} \geq T \\ R_{\max} P_{\max}(x, y) + R_{\min} P_{\min}(x, y), & \text{其他} \end{cases} \quad (11)$$

式中, R_{var} 为 $\text{VAR}_{\max}$ 对 $\text{VAR}_{\min}$ 的比例, $R_{\max}$ 为 $\frac{\text{VAR}_{\max}}{\text{VAR}_{\max} + \text{VAR}_{\min}}$, $R_{\min}$ 为 $\frac{\text{VAR}_{\min}}{\text{VAR}_{\max} + \text{VAR}_{\min}}$.

3.4 构造拉普拉斯金字塔

由融合后的梯度金字塔构造拉普拉斯金字塔的方式如下:

$$\text{LP}_l = [1 + \omega] * \sum_{k=1}^4 -\frac{1}{8} d_k * \text{GP}_k, 0 \leq l < N \quad (12)$$

3.5 图像重构

对拉普拉斯金字塔做 EXPAND 和加操作, 重构最后的图像:

$$\begin{cases} G_l = \text{LP}_l, & \text{当 } l = N \\ G_l = \text{LP}_l + \text{Expand}(G_{l+1}), & 0 \leq l < N \end{cases} \quad (13)$$

4 系统实现与优化

在 Intel Core i3 CPU 和 GTX480 GPU 的硬件上实现了这个图像融合系统. 在系统中, CPU 和 GPU 协同工作以融合图像. CPU 负责初始化和结束时资源回收的工作, 以及调动 GPU 进行计算. 计算密集的工作如图像分解、融合和重构都安排在 GPU 上执行以获得加速. GTX480 GPU 是高度并行化的设备, 可同时运行上万个硬件线程, 因此这个 GPU 加速的图像融合系统可比单纯 CPU 的系统快速很多.

4.1 系统实现

GPU 加速的图像融合系统有 5 个核(Kernel)函数将在 GPU 上执行: Gaussian Pyramid(), Gradient Pyramid(), Laplacian Pyramid(), Fuse(), Reconstruct(). 这 5 个函数对应着基于梯度分解图像融合的 5 个步骤. 整个图像融合系统执行的流程如图 2 所示.

系统开始运行时, CPU 首先完成初始化的工作如主机端(CPU)和设备端(GPU)的内存空间分配, 接着将源图像的数据从文件读入到主机端内存中的 1 维数组, 然后传输到设备端的内存中. 5 个 GPU 上运行的核函数被依次调用, 上一个函数的结果存放在设备端的内存中, 作为下一个函数的输入. 最后得到的输出图像的数据被从设备端内存传输回主机端内存, CPU 做资源回收的工作.

5 个 GPU 上运行的核函数被调用时, 每个函数都将被调用预先定义的金字塔的层数或者层数-1

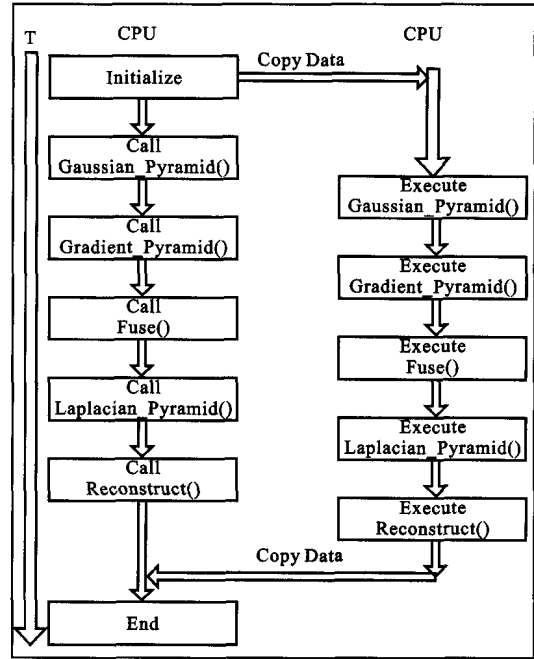


图 2 系统执行流程

次. 每次调用时根据当前要处理的图像的维度, GPU 上将产生对应数量的大量线程, 每个线程处理一个像素. 预设的金字塔的层数必须大于 3 层, 通常设为 7 层. 更大的层数会得到更好的图像质量, 但计算量也相应更大.

4.2 系统优化

在完成了系统的初始版本后, 我们进行了一些优化. 对系统效率提升最大的优化如下:

(1) 重新安排对 GPU 全局内存的访问模式

核函数在 GPU 上运行的时候, 需要大量读写 GPU 上的全局内存中存放的金字塔每层的数据. 通过安排对全局内存的访问模式, 使这些访问尽可能满足 GPU 合并访问的要求, 可以减少访存延迟.

(2) 调整判断语句以减少分支

GPU 的每个流多处理器是类似 SIMD 的结构, 只有在每个标量处理器都运行相同指令时才能取得最大效率. GPU 上运行的 5 个核函数代码中, 有一些条件判断, 在 GPU 上运行时会引起分支, 导致性能下降. 通过调整和精简了判断语句, 减少了分支.

(3) 设置共享内存和 L1 缓存的模式

GPU 上每个流多处理器上有 64 kB 的可快速访问的存储空间, 由共享内存和 L1 缓存共享, 可配置. 由于我们实现的 GPU 加速的图像融合系统的核代码更多的时候是从全局内存读数据, 而大容量的 L1 和 L2 缓存可以显著减少访问全局内存的延迟, 因此我们配置为 16 kB 的共享内存和 48 kB 的 L1 缓存.

5 图像融合实验及结果分析

图 3 及图 4 是同一个传感器在不同情况下取得的两幅图像. 图像 3 聚焦在右边的大时钟上, 左边的小时钟模糊. 图像 4 聚焦在左边的小时钟上, 右边的大时钟模糊. 目标是将这两幅图像融合在一起, 获得一幅两个时钟同时都清晰的图像.



图 3 右聚焦图像



图 4 左聚焦图像

经过图像融合系统处理后的结果图像如图 5 所示. 可以看到, 图中两个时钟都是清晰的. 两幅源图像中显著的图像细节被融合到了一起.

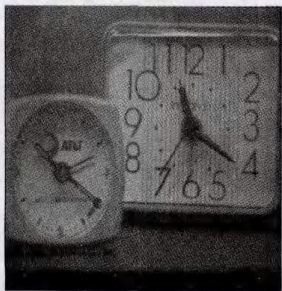


图 5 融合图像

实现了单纯 CPU 和利用 GPU 对大小钟表图像进行融合试验的系统, 试验测试了 512×512 , $1\,024 \times 1\,024$, $2\,048 \times 2\,048$ 这三种图像尺寸, 时间如表 1 所示.

表 1 CPU 和 GPU 实现的用时对比 ms

源图像尺寸	CPU	CPU+GPU	加速比
512×512	1 985.5	148.3	13.4
$1\,024 \times 1\,024$	7 970.4	317.1	25.1
$2\,048 \times 2\,048$	15 778.6	465.6	33.9

表 1 中, CPU 栏所列的时间是单纯 CPU 的系统处理不同尺寸图像所用的时间. CPU+GPU 栏所列的时间是 GPU 加速的系统处理不同尺寸图像所用的时间, 包括 CPU 消耗的时间和 GPU 消耗的时间. 实验显示在三种不同尺寸的图像上, GPU 加速的系统比单纯 CPU 的系统获得了从 13.4 倍到 33.9 倍的加速比.

实验显示加速比随着图像尺寸的增加而增大, 由于在源图像尺寸增大时, 金字塔每层的像素数都相应增多, 更能发挥 GPU 的并行度, 因此整个系统获得了更大的加速比.

6 结束语

结合最新的 GPU 的硬件结构和基于梯度分解的多源图像融合算法特点, 设计、实现并且优化了 GPU 加速的高效图像融合系统. 文中还提出了自适应式的基于方差的融合规则. 实验的融合图像比较有效的保留了源图像的细节信息, 相比单纯 CPU 图像融合系统的加速比证明了 GPU 加速图像融合系统的高效性和有效性.

参考文献:

- [1] 童盈, 张健. 基于 GPU 的快速二维沃尔什变换研究[J]. 微电子学与计算机, 2011, 28(1): 46—49.
- [2] Lindholm E, Nickolls J, Oberman S, et al. NVIDIA tesla: a unified graphics and computing architecture[J]. IEEE Micro, 2008, 28(2): 39—55.
- [3] NVIDIA. NVIDIA's next generation CUDA compute architecture: fermi, version 1.1[M]. Santa Clara: NVIDIA Corporation, 2010.
- [4] Smith M, Heather J. Review of image fusion technology in 2005[C]// Defense and Security Symposium 2005. Bellingham: SPIE, 2005, 5782: 29—45.
- [5] Burt P J. A gradient pyramid basis for pattern selective imagefusion[C]// Proceedings of the Society for Information Display Conference. San Jose: SIDPress, 1992: 467—470.
- [6] 陈浩, 王延杰. 基于小波变换的图像融合技术研究[J]. 微电子学与计算机, 2010, 27(5): 39—41.

作者简介:

潘晓辉 女, (1978—), 硕士, 讲师. 研究方向为高性能计算、图形图像处理.

张涛 男, (1978—), 博士研究生. 研究方向为并行与分布式计算.