

基于双线性插值的图像缩放在 GPU 上的实现

王 平, 全吉成, 赵柏宇

(空军航空大学 航空航天情报系, 吉林 长春 130022)

摘 要: 针对传统的在 CPU 上实现的基于双线性插值的图像缩放存在速度慢等问题, 利用 GPU 高性能并行计算优势, 实现了在 GPU 上基于双线性插值的快速缩放. 此算法将目标图像的每个像素分配给 GPU 中每个线程同时执行, 提高插值效率. 从实验结果可以看出, 此算法在放大图像时, 随着图像分辨率的增大, GPU 的插值速度相对 CPU 单线程和多线程的插值速度显著提高, 能很好达到实时缩放图像的效果.

关键词: GPU; 双线性插值; 并行算法; 图像缩放

中图分类号: TP391

文献标识码: A

文章编号: 1000—7180(2016)11—0129—04

Realization of Image Zooming in GPU Based on Bilinear Interpolation

WANG Ping, QUAN Ji-cheng, ZHAO Bo-yu

(Aerospace Intelligence Department, The Aeronautical University of The
China People Liberation Airforce, Changchun 130022, China)

Abstract: The traditional image scaling algorithm in CPU based on bilinear interpolation has to confront with the problem of slow scaling. In this paper, GPU as a burgeoning high performance computing technique, implemented fast zooming on the GPU based on bilinear interpolation. In this algorithm, each pixel of the target image is allocated to each thread in GPU, and the interpolation efficiency is improved. The experimental results show that compared with the CPU single thread and multi thread, this algorithm significantly improved the interpolation speed of GPU, with the increase of the image resolution, and can achieve the effect of real-time image zooming.

Key words: GPU; bilinear interpolation; parallel algorithm; image zooming

1 引言

图像的缩放技术是图像处理中的重要内容之一, 它指的是将图像的尺度进行放大或者缩小, 也就是转换图像的分辨率. 图像缩放技术在军事、遥感、图像显示、航天工程等领域有着广泛的应用. 例如当输入图像的分辨率与所需图像分辨率不匹配时, 就需要将原始图像进行缩放来满足所需图像的要求.

图像的缩放一般通过插值技术来实现, 常见的插值算法中包括最近邻域插值法、双线性插值法、双三次插值法. 其中最近邻域插值法是最简单的方法, 其优点是计算量小, 速度快, 但是缩放后的图像质量低, 会出现不连续的现象, 比如会出现马赛克和锯齿

现象^[1]. 双线性插值和三次插值算法按照缩放要求通过重采样都能得到较清晰的目标图像, 但是三次插值算法复杂度相对较高, 运算速度就会降低, 所以并没有被广泛采用. 而双线性插值既不会出现像素值不连续的情况, 而且运算速度快于三次插值算法^[2]. 因此本文采用双线性插值算法对图像进行缩放.

随着图像分辨率的增加, 所包含的数据量也随之增大. 将分辨率较大的图像进行缩放处理或者将图像进行大倍率的放大时都是非常耗时的计算过程, 很难达到实际应用中要求的实时性. 在单核计算机上, 文献[3-4]在提高图像放大的速度上, 有了很大的提高, 但是由于计算机核心主频的限制, 很难达

到实时处理的效果. 苗莎等在多核计算机上利用多线程技术设计了双线性插值并程序, 实现了图像放大的快速处理^[6], 但是对于数据量很大的图像进行缩放时, 由于内存的限制以及线程间的资源争夺, 此方法也难以达到实时处理.

图形处理器(Graphic Process Unit, GPU)具有单指令流多数据流(SIMD)的并行处理特性, 已经成为强大的并行处理单元. 本文实现了基于双线性插值图像在 GPU 上的快速缩放, 极大地缩减了计算时间, 达到了实时性的要求.

2 基于双线性插值的图像缩放

2.1 双线性插值原理

双线性插值是利用需要处理的原始图像点 p 周围的四个邻点 $(g(0,0), g(0,1), g(1,0), g(1,1))$ 的像素值在 x, y 两个方向上作线性内插如图 1 所示.

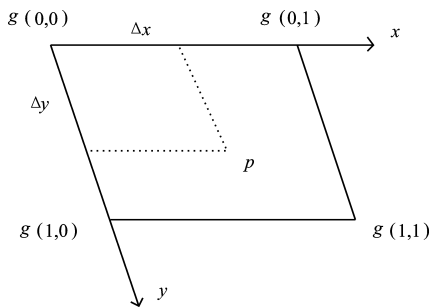


图 1 双线性插值法

该插值方法是用一个分段线性函数来近似表示四个邻点的灰度值对内插点灰度值的贡献大小, 分段线性函数为

$$w(t) = \begin{cases} 1 - |t|, & 0 \leq |t| \leq 1 \\ 0, & \text{其他} \end{cases} \quad (1)$$

假设图 1 中像素之间的间隔为 1, 且 p 点到像素点 $g(0,0)$ 间的距离在 x 和 y 方向的投影分别为 Δx 和 Δy , 则内插点 p 的灰度值为

$$D_p =$$

$$[\omega(\Delta x)\omega(1-\Delta x)] \begin{bmatrix} D_{00} & D_{01} \\ D_{10} & D_{11} \end{bmatrix} \begin{bmatrix} \omega(\Delta y) \\ \omega(1-\Delta y) \end{bmatrix} \quad (2)$$

(2) 式中, D_{ij} 为像素点 $g(i,j)$ 的灰度值.

2.2 基于双线性插值的图像缩放算法

假设原图像大小为 $\text{size} = l \times m$, 其中 l, m 分别为原图像的行数和列数. 若图像的缩放比例 t ($t > 0$), 则缩放后的图像大小为 $\text{size} = t \times l \times t \times$

m . 对于缩放后图像的某个像素点 $p(x,y)$ 通过 $p \times 1/t$ 可得到对应的原图像坐标 $p'(x_1, y_1)$, 其中 $x_1 = x/t, y_1 = y/t$, 由于 x_1, y_1 可能都不是整数, 所以并不存在这样的点, 但是可以找出与它相邻的四个点的灰度 g_0, g_1, g_2, g_3 , 使用双线性插值法就可以得到这个像素点 $p'(x_1, y_1)$ 的灰度, 也就是像素点 $p(x,y)$ 的灰度.

一个完整的基于双线性插值的图像缩放算法可描述如下:

(1) 通过原图像和比例因子得到新图像的大小, 并创建新图像.

(2) 由新图像的某个像素 (x,y) 映射到原图像 (x', y') 处, 利用双线性插值得到像素点 (x,y) 的值并写回新图像.

(3) 重复步骤(2) 直到新图像的所有像素的灰度值写完.

3 基于双线性插值图像缩放并行算法在 GPU 中的分析与设计

通过对基于双线性插值的图像缩放算法进行分析可以看出, 目标图像上的每一个像素都是经过和原图像坐标映射、灰度内插和赋值的, 计算量较大, 并且每个像素之间没有任何依赖关系, 耦合度高. 由于 GPU 专注于执行高度并行化的处理任务, 所以基于双线性插值的图像缩放算法非常适合在 GPU 上进行并行实现.

GPU 中的线程的组织包括 3 个层次: 线程网格(Grid)、线程块(Block)和独立线程(Thread)^[6]. 每个线程网格(Grid)由多个线程块(Block)组成, 而每个线程块(Block)由多个线程(Thread)组成. 在 GPU 中进行图像缩放并行化时, 将 GPU 线程网格映射到整个缩放后的目标图像上, 再将目标图像与 GPU 线程网格分块, 确保 GPU 线程块中每个线程对应目标图像分块中的每一个像素进行处理, 如图 2 所示.

经过 GPU 线程与目标图像像素映射之后, 双线性插值图像缩放在 GPU 中的并行过程如图 3 所示. 说明如下.

(1) 利用 GDAL 组件读取待缩放图像及其长度和宽度, 计算出缩放后图像的长度和宽度.

(2) 将待缩放图像数据拷贝到 GPU 全局存储器中, 设置 GPU 线程块大小, 根据目标影像大小确定线程块数量. 本文所选图像均为 256 的整数倍, 因此一个线程块中分配 256 个线程, 用插值缩放后的

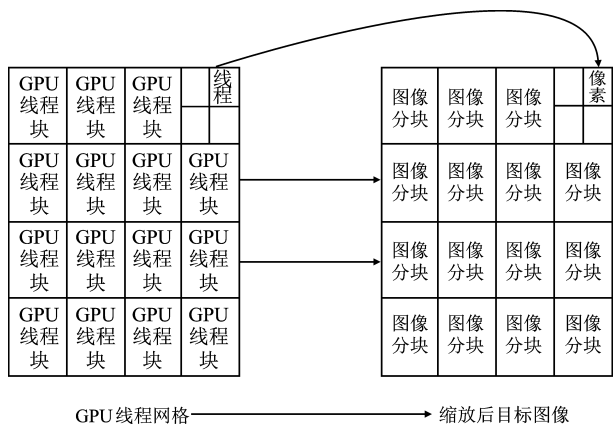


图 2 GPU 图像缩放并行化映射方式

图像的长与宽.

(3)根据目标图像每个像素点的索引值计算对应原图像的坐标,并进行双线性插值.

(4)将缩放后的图像传输到主机内存.

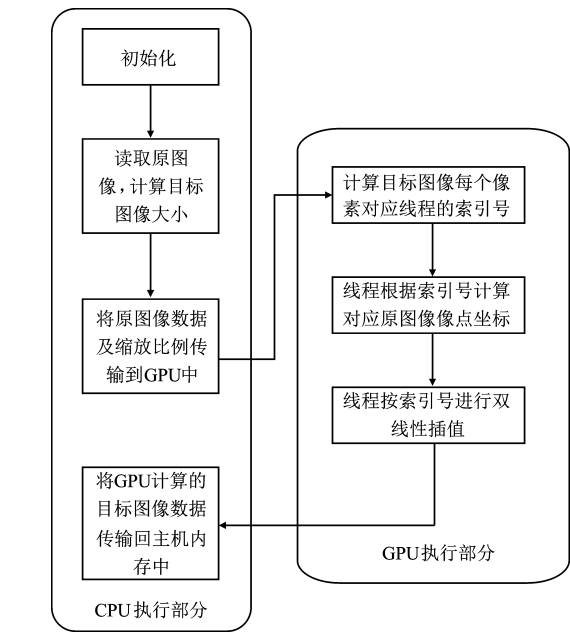


图 3 基于双线程插值图像缩放算法在 GPU 中的基本流程

4 实验结果与分析

4.1 实验环境

为了验证算法的有效性,用 C++实现本文的算法.具体的软硬件系统配置如表 1 所示.

4.2 实验过程

为了进行多组数据的对比实验,首先对原始图像进行预处理,获取分辨率分别为 256×256 , 512×512 , $1\,024\times1\,024$, $2\,048\times2\,048$, $4\,096\times4\,096$, $8\,192\times8\,192$, $10\,240\times10\,240$ 的图像.

表 1 试验环境软硬件参数

名称	内容
处理器	AMDAthlon (tm) II X4 640 3.00 GHz
内存	10.0 GB
硬盘	1.0 TB
操作系统	Windows Server 2008 R2 Standard
开发环境和语言	Microsoft Visual Studio 2010,C++
显卡	NVIDIA GTX580
CUDA 版本	7.0 版

运用双线性插值对 7 幅不同分辨率的图像进行放大 2 倍的插值对比实验,分别运行插值的 CPU 和 GPU 程序,记录双线性插值的处理时间.本文使用时间加速比 S_p 对算法进行评价,定义如下^[7]:

$$S_p = \frac{T_s}{T_P} \tag{3}$$

式(3)中, T_s 为 CPU 中串行算法的运行时间, T_P 为 GPU 中并行算法的运行时间, GPU 中的时间包括实验数据从主机内存到显卡内存的输入和输出时间.对实验数据进行了 10 次测试求取平均运行时间,数据如表 2 所示.

表 2 不同尺寸图像双线性插值在 CPU 和 GPU 中的运行时间

实验图像 大小(pixel)	平均运行时间/s		加速比/倍
	CPU 串 行算法	GPU 并 行算法	
256×256	0.062	0.001 605	38.63
512×512	0.271	0.005 505	49.23
1 024×1 024	2.387	0.031 183	76.55
2 048×2 048	8.982 6	0.084 689	106.07
4 096×4 096	37.699	0.315 850	119.36
5 120×5 120	62.744	0.509 055	123.26
8 192×8 192	155.108	1.233 329	125.76
9 216×9 216	201.738	1.582 770	127.46

为了验证在双线性插值图像缩放并行算法上 GPU 多线程与 CPU 多线程的性能优越性,将本文算法与文献[5]的算法(同时并行 6 个线程)进行对比,如表 3 所示.

另外,为了验证 GPU 中双线性插值图像缩放结果的正确性,将 GPU 和 CPU 计算缩放后的结果影像进行差值处理,计算出差值图像的灰度直方图,

如图 4 所示.

表 3 本文算法与文献[5]中算法试验结果比较

实验图像 大小(pixel)	平均运行时间/s		性能提升/倍
	本文并 行算法	文献[5]中 并行算法	
256×256	0.001 605	0.021	13.08
512×512	0.005 505	0.092	16.71
1 024×1 024	0.031 183	0.721	23.12
2 048×2 048	0.084 689	2.153	25.42
4 096×4 096	0.315 850	9.386	29.72
5 120×5 120	0.509 055	16.876	33.15
8 192×8 192	1.233 329	45.176	36.63
9 216×9 216	1.582 770	61.518	38.87

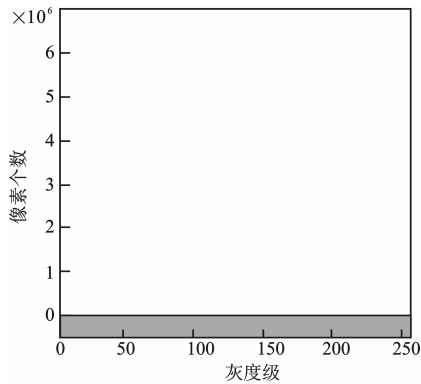


图 4 差值影像的灰度直方图

4.3 结果分析

由表 2 可以分析得出,GPU 并行的双线性插值算法处理速度远远大于传统的单线程双线性插值算法的处理速度.随着图像分辨率的增大,两者之间的加速比也相对增加,加速比甚至达到 127.46 倍,这说明分辨率越大,GPU 加速的双线性插值速度越明显.对于分辨率较小的图像,GPU 加速效果不明显,这是由于显存与主存之间的数据传输时间占用了算法运行时间的相当大的比例.图像分辨率的增加使双线性插值时间增多,数据传输时间占用的比例也随之降低,加速比也就越明显.

由表 3 可以看出,相对于文献[5]的算法,本文算法性能较好,且随着图像分辨率增加,性能提升比也越来越大.因为 GPU 中同时并行运行的线程可达几千个,远远大于 CPU 中的启动的线程,且 GPU 中线程之间切换开销很小,极大提高了处理速度,所以 GPU 加速的双线性插值并行算法速度远远大于 CPU 中的多线程加速的双线性并行算法.

本文使用的 NVIDIA GTX580 是 Fermi 系列

GPU,具有和 CPU 同样的双精度数据处理能力.由图 4 差值影像的灰度直方图可以看出,GPU 和 CPU 计算缩放后的影像不存在很大灰度差异,表明了 GPU 计算结果的正确性.

5 结束语

针对在 CPU 环境下,图像基于双线性插值缩放速度慢的问题,本文实现了图像双线性插值在 GPU 上的计算,充分利用了 GPU 的并行计算单元,极大地加快了图像缩放速度,相对于基于 CPU 单线程和多线程的双线性插值,有很大的计算高效的优点,基本达到了实时处理能力.

参考文献:

[1] Chen Liang, Gao Cheng-min. Fast discrete bilinear interpolation algorithm[J]. Computer Engineering and Design,2007 ,28(15) :3787-3790.

[2] Gribbon K T, Bailey D G. A novel approach to real-time bilinear interpolation[C]// Second IEEE International Workshop on Electronic Design, Test and Applications. New Zealand:IEEE, 2004 : 126-131.

[3] 胡敏,张佑生. Newton Thiele 插值方法在图像放大中的应用研究[J]. 计算机辅助设计与图形学学报,2003, 15(8):1004-1007.

[4] 陈建辉,许卫全,王博亮. 一种实时视频分辨率提升算法及其 VLSI 设计与实现[J]. 微电子学与计算机, 2005,22(1),148-151.

[5] 苗莎,郑晓薇. 双线性插值图像放大并行算法的设计与实现[J]. 微电子学与计算机,2011,11(28),182-184.

[6] 赖积保,孟圆,余涛. 一种基于 Dual-GPU 的三次卷积插值并行算法研究[J]. 计算机科学,2013,40(8), 24-27.

[7] 邹贤才,李建成,汪海洪,等. OpenMP 并行计算在卫星重力数据处理中的应用[J]. 测绘学报,2010,39(6): 636-641.

作者简介:

王 平 男,(1991-),硕士研究生. 研究方向为并行图像处理. E-mail:wpsdinws@163. com.

全吉成 男,(1960-),教授,博士研究生导师. 研究方向为信息系统与资源管理.

赵柏宇 男,(1991-),硕士研究生. 研究方向为粒子系统仿真.