文章编号:1001-2486(2011)03-0111-04

基于GPU的二维离散小波变换快速计算*

马伯宁,王晨昊,汤晓安,匡纲要
(国防科技大学 电子科学与工程学院,湖南 长沙 410073)

摘 要:通过分析小波变换的多相表示和GPU通用计算模型,结合现代GPU的多纹理和多渲染目标特性,提出了一种基于GPU与多相表示的二维离散小波变换计算方法,该方法使小波变换的计算形式完全适合GPU的SIMD结构,同时大幅减少了纹理访问次数,充分利用了GPU的矢量运算和二维寻址能力,实验表明该方法在处理速度上有大幅的提高。

关键词:二维离散小波变换;多相表示;图形处理单元
中图分类号:TP311 **文献标识码**:A

Fast 2D DWT Calculation Based on GPU

MA Bo-ning, WANG Chen-hao, TANG Xiao-an, KUANG Gang-yao
(College of Electronic Science and Engineering, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: By analyzing the DWT's polyphase representation and GPU's general-purpose computation model, this study presents a 2D DWT algorithm based on GPU and polyphase, which combines with modern GPU's multi-texture property and multiple render targets property. The computing form is well suited to GPU's SIMD architecture in the method presented. The number of texture visit is significantly reduced. The GPU's vector processing ability and 2D address accessing ability is well utilized. The experimental result shows that the processing speed is greatly improved.

Key words: 2D DWT; polyphase representation; GPU(Graph Process Unit)

由于小波分析具有多分辨率特性、良好的时频域局部特性,其在卫星遥感领域有着广泛的应用。但遥感数据量普遍很大,如一景影像常在百兆之上,且分辨率还在不断提高,使得利用通用CPU难以完成遥感数据的实时小波变换。本文主要目的是充分利用现代图形处理单元(GPU)的特性,实现持续大数据量遥感影像的快速小波变换。

1 已有研究基础

现代GPU都采用了单指令多数据流(SIMD)的结构,使得它在进行大数据量的密集计算时,处理速度已远超同价位的通用CPU,且随着GPU可编程能力的提高,一些通用的计算已可由GPU来实现。目前已有一些研究者提出利用GPU实现低成本快速的小波计算:文献[1-2]最早提出了利用GPU进行小波变换,它利用特定GPU上的卷积功能以及对图像的缩放偏移功能进行小波变换,但该方法限定在少数具有卷积功能的GPU。文献[3-4]对基于GPU的三维数据小波变换进行了研究,重点研究了适应GPU计算的三维数据组织。文献[5]利用当前GPU具有的可编程能力完成小波变换,重点针对小波变换不同子带计算式不统一及边界问题,利用地址表统一计算式,该方法适合任何小波及各种常用边界处理的情况;但它实际上是一种间接寻址的方式,增加了纹理数据访问的步骤,也破坏了纹理数据的顺序访问,大大增加了读取纹理数据的时间。

目前利用GPU进行小波变换的主要问题是计算式在不同子带或奇偶位置不统一,一般的方法是通过条件判断选择相应的计算式,这与GPU的SIMD结构不符,降低了计算效率。文献[5]虽利用地址表统一了计算式,却大大增加了读取纹理的时间。

2 相关基础理论

2.1 GPU通用计算模型

现代GPU都具有可编程能力,使得其在一定场合可实现通用计算。在文献[6-7]中,详细介

* 收稿日期:2010-10-12
基金项目:国家部级科研项目
作者简介:马伯宁(1975—),男,讲师,博士生。

绍了 GPU 通用计算模型。图 1 是使用片段着色器的通用计算模型:可编程片段着色器相当于 CPU,纹理缓存和离屏缓存相当于内存。在启动绘制命令后,片段着色器对每个片段进行如下操作:根据每个片段的纹理坐标读取相应的纹理数据,并对这些纹理数据进行处理,将处理后的结果根据片段的坐标自动写入离屏缓存。

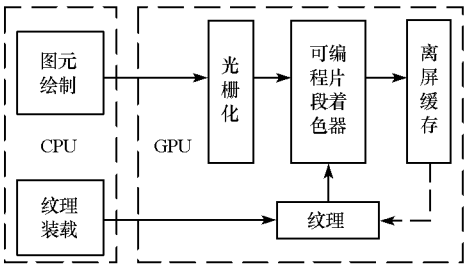

图 1 GPU 通用计算模型结构图

Fig.1 GPU's general-purpose computation model

由上述 GPU 计算过程可以看出,GPU 通用计算可看作是单指令多数据(SIMD)模型。

2.2 二维离散小波变换

针对离散小波变换的计算,目前主要有两种方法,一种是 Mallat 算法,另一种是提升算法。由于提升算法涉及奇偶数据的大量交织运算,不适合 GPU 的 SIMD 结构,因此基于 GPU 的小波变换主要利用 Mallat 算法。

对于 2D DWT,Mallat 算法的计算如式(1)所示。式中下标 $S = LL, HL, LH, HH$,分别对应 2D DWT 的四个子带,$H(z, w)$ 为分析滤波器;$C^J(z,w)$为待分解的数据,$C^{J-1}(z,w)$为经过一次分解后的结果数据;D 表示二维抽取算子,作用是抽取二维 Z 变换中的$\{z^{2m}w^{2n} \mid m\in Z, n\in Z\}$项,并将其转换为$\{z^m w^n \mid m\in Z, n\in Z\}$。

$$C_S^{J-1}(z,w) = D(C^J(z,w)\cdot H_S(z,w)) \quad (1)$$

对于二维小波逆变换,设综合滤波器的二维 Z 变换分别为 $F_{LL}(z,w)$、$F_{HL}(z,w)$、$F_{LH}(z,w)$、$F_{HH}(z,w)$,则二维重构公式如式(2)所示。

$$\begin{aligned} C^J(z,w) = {} & C_{LL}^{J-1}(z^2,w^2)F_{LL}(z,w) + \\ & C_{HL}^{J-1}(z^2,w^2)F_{HL}(z,w) + \\ & C_{LH}^{J-1}(z^2,w^2)F_{LH}(z,w) + \\ & C_{HH}^{J-1}(z^2,w^2)F_{HH}(z,w) \end{aligned} \quad (2)$$

在已有的基于 GPU 的小波变换算法中,为避免式(1)中显式的抽取操作,一般将式(1)写成式(3)的形式,表达式中 $h_S(u,v)$分别对应四个子带 $S = LL, HL, LH, HH$ 的滤波器。因此若在片段程序中计算整个数据的小波变换,需要判断当前点应该利用哪个子带的计算公式,这会导致片段程序中存在分支结构,而 GPU 执行分支语句时,执行效率会大大降低。对于逆变换也同样存在该问题。

$$c_S^J(m,n) = \sum_{u,v} h_S(u,v)c^J(2m-u, 2n-v) \quad (3)$$

3 基于 GPU 与多相表示的二维离散小波变换

3.1 小波变换的多相表示

为使小波变换更适合 GPU 的 SIMD 结构,我们利用多相表示改写了小波变换的计算式,使计算最大限度地适应 GPU 的结构。

记 $C(z,w)$、$H(z,w)$分别为二维数据与滤波器的二维 Z 变换,以下标 $P1$、$P2$、$P3$、$P4$ 表示二维数据的四相,即 $C_{P1}(z,w)$对应二维数据$\{c_{m,n} \mid m\in Z, n\in Z\}$的偶行偶列元素(即$\{c_{2m,2n}\}$),$C_{P2}(z,w)$对应偶行奇列元素,$C_{P3}(z,w)$对应奇行偶列元素,$C_{P4}(z,w)$对应奇行奇列元素。

则 $C(z,w)$的多相表示为

$$\begin{aligned} C(z,w) = {} & C_{P1}(z^2,w^2) + z^{-1}\cdot C_{P2}(z^2,w^2) + \\ & w^{-1}\cdot C_{P3}(z^2,w^2) + z^{-1}w^{-1}\cdot \\ & C_{P4}(z^2,w^2) \end{aligned}$$

同理 $H(z,w)$也可以写成多相表示。考虑到算子 D 的抽取作用,可将 $D(C(z,w)\cdot H(z,w))$表示为如下多相形式:

$$\begin{aligned} & D(C(z,w)\cdot H(z,w)) \\ = {} & C_{P1}(z,w)\cdot H_{P1}(z,w) + z^{-1}C_{P2}(z,w) \\ & \cdot H_{P2}(z,w) + w^{-1}C_{P3}(z,w)\cdot H_{P3}(z,w) \\ & + z^{-1}w^{-1}C_{P4}(z,w)\cdot H_{P4}(z,w) \end{aligned}$$

对式(1)采用上述多相形式表示,并写成矩阵形式有

$$\begin{pmatrix} C_{LL}^{J-1}(z,w) \\ C_{HL}^{J-1}(z,w) \\ C_{LH}^{J-1}(z,w) \\ C_{HH}^{J-1}(z,w) \end{pmatrix} = \begin{pmatrix} H_{LL,P1}(z,w) & H_{LL,P2}(z,w) & H_{LL,P3}(z,w) & H_{LL,P4}(z,w) \\ H_{HL,P1}(z,w) & H_{HL,P2}(z,w) & H_{HL,P3}(z,w) & H_{HL,P4}(z,w) \\ H_{LH,P1}(z,w) & H_{LH,P2}(z,w) & H_{LH,P3}(z,w) & H_{LH,P4}(z,w) \\ H_{HH,P1}(z,w) & H_{HH,P2}(z,w) & H_{HH,P3}(z,w) & H_{HH,P4}(z,w) \end{pmatrix} \begin{pmatrix} C_{P1}^J(z,w) \\ z^{-1}C_{P2}^J(z,w) \\ w^{-1}C_{P3}^J(z,w) \\ z^{-1}w^{-1}C_{P4}^J(z,w) \end{pmatrix} \quad (4)$$

同理,对于二维小波重构计算式(2)也可以利用多相表示,式(2)可改写为如下矩阵形式:

$$\begin{pmatrix} C_{P1}^{J}(z,w) \\ C_{P2}^{J}(z,w) \\ C_{P3}^{J}(z,w) \\ C_{P4}^{J}(z,w) \end{pmatrix} = \begin{pmatrix} F_{LL,P1}(z,w) & F_{HL,P1}(z,w) & F_{LH,P1}(z,w) & F_{HH,P1}(z,w) \\ F_{LL,P2}(z,w) & F_{HL,P2}(z,w) & F_{LH,P2}(z,w) & F_{HH,P2}(z,w) \\ F_{LL,P3}(z,w) & F_{HL,P3}(z,w) & F_{LH,P3}(z,w) & F_{HH,P3}(z,w) \\ F_{LL,P4}(z,w) & F_{HL,P4}(z,w) & F_{LH,P4}(z,w) & F_{HH,P4}(z,w) \end{pmatrix} \begin{pmatrix} C_{LL}^{J-1}(z,w) \\ C_{HL}^{J-1}(z,w) \\ C_{LH}^{J-1}(z,w) \\ C_{HH}^{J-1}(z,w) \end{pmatrix} \tag{5}$$

3.2　基于 GPU 与多相表示的二维离散小波变换结构

由式(4)和(5)可以看出,对每个分量计算都是通常的二维卷积,没有了显式的抽取内插操作,表达式中计算的输入、输出都是四元矢量,这些计算形式在现代 GPU 上都可得到有效的支持:多输入可使用 GPU 的多纹理功能,多输出可使用 GPU 的多渲染目标功能。因此式(4)和(5)已能很好地适应 GPU 的 SIMD 结构。

结合 2D DWT 的多相表示和 GPU 通用计算模型,对于小波的分解计算式(4)在 GPU 上的实现形式如图 2(a)所示。图中,利用 GPU 的多渲染目标功能将(4)式的 4 个结果分量分别存入 4 个离屏缓存,待处理数据与 4 个子带滤波器系数分别保存在纹理缓存中,可编程片段着色器根据(4)式的二维卷积关系进行编程。进行小波变换时,将待处理的二维数据(大小为 $M \times N$)读入纹理缓存,然后通过绘制一个大小为($M/2$)×($N/2$)的矩形启动 GPU 计算,绘制结束后,4 个离屏缓存中的结果即为小波变换后的 4 个子带。

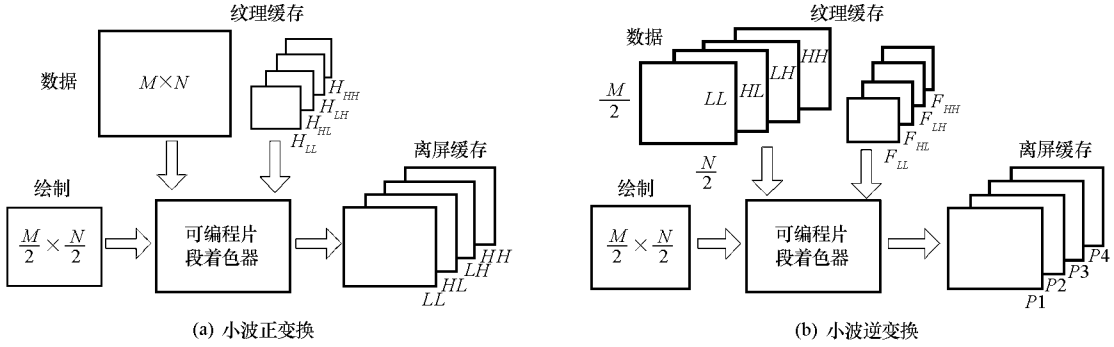

图 2　基于多相表示的二维离散小波变换在 GPU 中的实现结构图

Fig.2　The implementation architecture of 2D DWT based on polyphase and GPU

在图 2(a)中没有显式地将待处理数据组织成多相形式,而是直接将内存中数据读入纹理缓存,原因是为了减少重新组织内存数据的额外开销。因此在对片段着色器编程时需注意(4)式中多相形式数据与纹理缓存中数据的对应关系:$P1$ 相中的位置(x,y)在纹理数据中对应位置$(2x,2y)$,$P2$ 相中的位置(x,y)在纹理数据中对应位置为$(2x+1,2y)$,$P3$ 相、$P4$ 相类似。

小波的重构计算式(5)在 GPU 上的实现形式如图 2(b)所示。图中将重构后的四相数据分别存入 4 个离屏缓存,即每一个缓存保存(5)式中结果矢量的一个分量,待处理的 4 个子带数据与对应的滤波器系数分别保存在纹理缓存中,可编程片段着色器根据(5)式进行编程,最后绘制一个($M/2$)×($N/2$)大小的矩形启动 GPU 的计算,绘制结束后,4 个离屏缓存中的数据即为重构结果的四相表示。

3.3　GPU 特性的利用

由式(4)可以看出,对于结果矢量的各分量,其计算所需的源数据绝大部分相同,即计算一个分量所需的纹理数据大部分可以供其他三个分量使用,因此,在图 2(a)的片段着色器中处理一个片段时,可以先将结果矢量计算所需的所有数据读入寄存器中,然后各分量的计算可以共用这些数据,这是式(4)将小波分解结果表示为四元矢量产生的一个好处。相对于现有的算法,本方法可以使纹理数据的访问次数减少近 3/4,在 GPU 计算中,纹理访问比较费时,因此该方法会大大减少系统的处理时间。同时,GPU 为加快纹理数据的读取,都会有数据预取的功能,与 CPU 不同的是,GPU 预取是采用二维块的形式,即在读取某一点的纹理时,会将其二维邻域的纹理数据读入 cache。本文采用二维卷积的形式,因此其在纹理数据的读取上要快于两次一维卷积的方式。

进一步,考虑到对一景遥感影像处理时,通常将其分为多个影像块,因此可以利用 GPU 的四元数运算使多个影像块并行处理:将 4 块数据分别保存在纹理的 RGBA 分量中,经过图 2 所示的处理,可实现 4 个块的并行计算。已有文献对 GPU 四元数运算的利用都采用将一块数据保存在

RGBA 分量中,该方式虽然使读取纹理数据的次数减少,但会导致片段着色器获取数据时定位不统一,即对某个片段点除了确定其纹理坐标外,还需确定纹素中对应的颜色分量位置;另外,这种方式只能减少纹理访问的时间,无法利用 GPU 的矢量运算能力,而本文的方式不存在这些问题。

4 实验结果与分析

鉴于文献[5]的时间分析最为详细且时间最近,我们选用其作为性能比较的参考。文献[5]中 GPU 为 GeForce 7800 GTX;本文 GPU 选用 Geforce 6600,128M 显存,显卡核心频率 450MHz,变换使用 Daubechies9/7 小波。实验对小波变换处理时间进行统计,包括:数据组织时间、纹理装载时间、小波变换计算时间、结果回读至内存时间。统计时间未考虑 OpenGL 及 CG 初始化开销,这是由于大批量数据计算时,初始化时间会平摊至各块。

2D DWT 正变换的处理时间如表 1 所示,表中列出了针对不同尺寸图像本文方法 1 至 5 级处理的时间及文献[5]中 5 级处理的时间(已减掉了 OpenGL 及 CG 初始化时间)。

表 1 2D DWT 正变换处理时间(单位为 ms)

Tab.1 The time of 2D DWT

图像尺寸	256×256	512×512	1024×1024
1 级处理时间	18	45	128
2 级处理时间	19	47	142
3 级处理时间	20	50	146
4 级处理时间	20	50	147
5 级处理时间	21	50	148
文献[5]5 级处理时间	282	359	781

由于本文方法的二维小波逆变换结果以多时相形式存储,因此不适合在 GPU 中进行多级小波逆变换,表 2 中仅列出了单级小波逆变换的时间。

表 2 2D DWT 逆变换处理时间(单位为 ms)

Tab.2 The time of 2D IDWT

图像尺寸	256×256	512×512	1024×1024
1 级	25	62	162

由实验结果可以看出:

(1)本文方法在计算性能上较文献[5]有大幅提高(在 512×512 尺寸以下提高了 6 倍以上,在 1024×1024 尺寸提高了 4 倍)。随着图像尺寸增大,提高的幅度逐渐减少,这是由于随着图像尺寸增大,数据组织时间及数据在 GPU 与 CPU 之间传输时间所占的比重逐渐增大。

(2)在表 1 多级小波变换的时间上,90%以上时间花费在前两级,原因是 1 级以后的变换数据无需由内存传入显存,且后续各级处理的数据以 4 倍速递减,因此在性能评价上,前两级的处理时间已能很好地反应算法的性能。

5 结 论

本文提出了基于 GPU 与多相表示的二维小波变换方法,相对于已有基于 GPU 的小波变换方法,本文方法具有如下特点:

(1)通过小波变换的多相表示,消除了计算时的分支结构,使得小波变换计算形式完全适合 GPU 的 SIMD 结构。

(2)在小波分解计算过程中,通过结合多相表示和 GPU 的多目标输出,使得一个结果矢量的计算可以共用大部分纹理数据,从而使纹理数据的访问次数减少近 3/4;同时直接计算二维小波变换,对 GPU 二维寻址能力及纹理预取能力的利用要优于两次一维小波变换。

(3)针对多幅影像处理同时进行的情况,利用 GPU 对四元数运算的硬件支持,将 4 幅影像数据分别保存在纹理的 RGBA 分量上,可实现 4 幅影像数据的并行变换。

参 考 文 献:

[1] Hopf M, Ertl T. Hardware Based Wavelet Transformations[C]// Workshop on Vision, Modeling, and Visualization' 99, 1999: 317 - 328.

[2] Hopf M, Ertl T. Hardware Accelerated Wavelet Transformations [C]//Proceedings of EG/IEEE TCVG Symposium on Visualization, 2000.

[3] Garcia A, Shen H W. GPU-based 3D Wavelet Reconstruction with Tileboarding[J]. The Visual Computer, 2005,21(8 - 10):755 - 763.

[4] 罗月童,薛晔,刘晓平.基于 GPU 的多分辨率体数据重构和渲染[J].计算机辅助设计与图形学学报,2009,21(1)

[5] Wong T T, Leung C S, Heng P A, et al. Discrete Wavelet Transform on Consumer-level Graphics Hardware [J]. IEEE Transactions on Multimedia, 2007, 9(3): 668 - 673.

[6] Tor D, Trond R H, Jon M H. The GPU as A High Performance Computational Resource [C]//Proceedings of the 21st Spring Conference on Computer Graphics, 2005: 21 - 26.

[7] 吴恩华,柳有权.基于图形处理器(GPU)的通用计算.计算机辅助设计与图形学学报[J].2004,16(05): 601 - 612.