

# 基于 PAAG 的图形图像算法的并行实现

孙建<sup>1</sup>, 李涛<sup>2</sup>, 李雪丹<sup>2</sup>

(1. 西安邮电大学 计算机学院, 陕西 西安 710121;

2. 西安邮电大学 电子工程学院, 陕西 西安 710121)

**摘要:**为了解决当前的 CMOS 技术遇到“功耗墙”和“散热墙”等问题导致的很难通过提高主频来提升芯片性能的问题, 文中提出了一种新型多态同构阵列处理器—PAAG (Polymorphic Array Architecture for Graphics)。该阵列机在一个芯片上集成了多个处理器, 能够通过将各种高性能复杂的算法合理分解映射到该平台上实现并行计算。通过结合使用数据并行、操作并行的计算方法, 对固定渲染管线的图形算法以及由国际标准组织 Khronos 提出的计算视觉标准 OpenVX1.0 中的 Kernel 函数图像算法进行了深入分析, 并给出了基于这些算法在 PAAG 上的并行化设计。通过在 PAAG 硬件平台对应的仿真环境上进行各个算法的并行实现, 得到了算法在多个处理单元上的运行时钟, 由此计算出算法在多个处理单元上运行的加速比。实验结果表明, 文中的并行化设计方法在 PAAG 上能够实现对图形图像算法的线性加速, 与串行相比, 效率更高。

**关键词:**并行计算; 多态同构阵列机; 图形处理; 图像处理; OpenVX1.0

中图分类号: TP301.6

文献标识码: A

文章编号: 1673-629X(2015)11-0061-06

doi:10.3969/j.issn.1673-629X.2015.11.013

## Parallel Implementation of Graphics Rendering and Image Processing Algorithm Based on PAAG

SUN Jian<sup>1</sup>, LI Tao<sup>2</sup>, LI Xue-dan<sup>2</sup>

(1. School of Computer Science, Xi'an University of Posts and Telecommunications,  
Xi'an 710121, China;

2. School of Electronic Engineering, Xi'an University of Posts and Telecommunications,  
Xi'an 710121, China)

**Abstract:** In order to solve the problem that current CMOS technology has already met the "wall" of power and cooling which may cause the issue of improving the performance by improving the frequency of the chips, present a new polymorphic isomorphic array processor, called PAAG (Polymorphic Array Architecture for Graphics and image processing). This array integrates multiple processing elements on a chip, it can realize parallel computing of the high-performance and complex algorithms by dividing and mapping them to the platform. By combining the data-level and the operation-level parallel calculation methods, the algorithms of the fixed rendering pipeline and these of OpenVX1.0, a standard of computer vision, proposed by the international standard organization Khronos, are in-depth analyzed in this paper. And the parallel design of these algorithms are proposed based on PAAG. The soft simulation platform of PAAG can give the result number of the running clock of the parallel implementation of the algorithms. Then through calculating the speed-up ratio, can conclude that the parallel implementation of the design method and its experimental results show that the algorithms of both graphic and image can be parallel acceleration on PAAG platform, compared with the serial processing, this method can be more effective.

**Key words:** parallel computing; polymorphic array processor; graphics processing; image processing; OpenVX1.0

## 0 引言

近年来, 图形图像处理和计算视觉在算法和系统

结构上都有了长足的发展。在各类终端设备<sup>[1]</sup>, 大型游戏开发等方面, 人们对于更高质量的视觉感受追求

收稿日期: 2015-01-04

修回日期: 2015-04-10

网络出版时间: 2015-11-04

基金项目: 国家自然科学基金重点基金资助项目 (61136002)

作者简介: 孙建 (1988-), 男, 硕士研究生, 研究方向为计算机系统结构与 VLSI; 李涛, 博士, 教授, 研究方向为计算机体系结构、集成电路设计、计算机图形学硬件等。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20151104.0950.038.html>

使得图形图像处理器设计<sup>[2]</sup>和计算机视觉技术面临更大挑战。并行计算在提高计算处理速度,解决大数据量,高复杂性的算法问题方面扮演了重要角色。而随着半导体工艺技术的持续发展,传统的单纯依靠提高主频的方式来提高单核处理器(CPU)性能,从而实现复杂算法高效运行已不能满足现实要求。多线程对内核的并行处理器应运而生,作为执行并行程序的重要载体,众核图形处理器为复杂图形图像算法以及相关应用的高性能计算的并行处理提供了很好的平台。

计算视觉和增强现实已经得到了广泛应用,在各种移动器件上也都进行了实现,具有及其重要的地位。最近,继建立 OpenGL 图形标准之后,Khronos 组织提出了基于底层的计算视觉标准 OpenVX1.0。该标准能够支持多种现代硬件系统,如各种移动终端设备、嵌入式 SOC、DSP 子系统等。文中实现了固定图形渲染管线的并行处理,同时介绍了一种并行的 OpenVX 1.0<sup>[3]</sup>在自主研发的新型多态同构阵列众核处理器—PAAG 上的实现。OpenVX 是最新的底层计算视觉加速标准,据笔者所知,OpenVX 实现是该标准最早的并行实现。通过实现基于 PAAG 渲染架构的两种不同的并行方法,在详细实验数据基础上分析了 PAAG 实现经典统一渲染以及 OpenVX 视觉标准的并行优势与不足。该研究对于图形渲染算法及应用和高性能计算视觉实现都有独特的意义。

## 1 PAAG 多态同构并行阵列处理器

第一代阵列结构的超级计算机 ILLIAC IV<sup>[4]</sup>的诞生为并行计算领域提供了极为重要的基础并行计算理论,掀起了人类研究并行计算机的热潮,此后业界推出了各种各样的并行计算机与并行阵列机。以 NVIDIA 公司的主流图形处理器(Graphic Processing Unit, GPU)为例,它是由多核流处理器阵列组成,每个阵列包含数个流处理器。采用单指令多线程(Single Instruction Multiple Threads, SIMT)<sup>[5]</sup>计算结构,该结构适合数据并行计算,而在其他的并行计算模式下却少有用武之地。与其相比,文中提出的多态同构并行阵列处理器(Polymorphic Array Architecture for Graphic, PAAG)<sup>[6]</sup>是自主研发的一种适用于图形和图像处理<sup>[7]</sup>的并行高效的阵列机结构,能够适应多种并行处理模式,同时其兼有可编程器件的灵活性和接近专用集成电路(ASIC)的处理性能。

如图 1 所示,PAAG 阵列处理器是一个簇结构,它是由多个最基本、最核心的处理部件(Processing Elements, PE)组成的二维阵列。而 PE 又是由 ALU(算术逻辑单元)控制器、路由器、四个相邻共享存储,一个本地指令存储和本地数据存储构成的双发射处理单

元。图 1 中多个 PE 组成的二维阵列,称之为簇。它可以以层次结构和平面展开的方式组织。

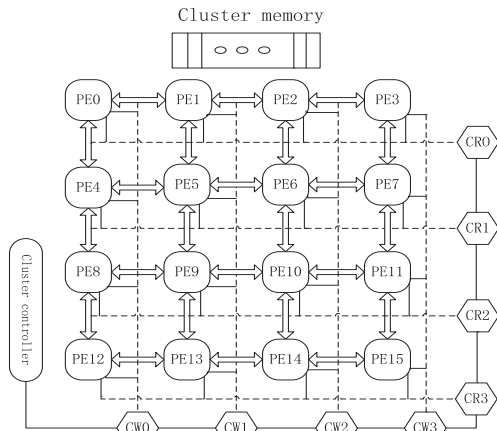


图 1 阵列处理器基本簇结构

基本簇单元(16 个 PE)由近邻的处理单元组成,通过簇控制器实现对簇中的行列控制器以及 PE 的控制。对于单个簇,每个 PE 通过共享存储,可以以阻塞或者非阻塞的方式实现与其四个不同方向 PE 的数据通信,PE 中的路由器可以通过路由选择使不相邻的 PE 经过多跳后实现数据通信,并且路由器允许数据多播,能够实现数据流的多目标扇出,提高通信效率。

PAAG 完整的阵列架构如图 2 所示。包括一个与 CPU 的接口,前段处理器(FEP),四个 F 簇(F Cluster0-3),四个 S 簇(S Cluster0-3),四组 POR 处理器,一个片上 Z 缓存(HyperZ),外部存储器,各种专用加速器 Accel(包括基本函数运算器、图形处理器、加速器等)和片上互联 interconnect。

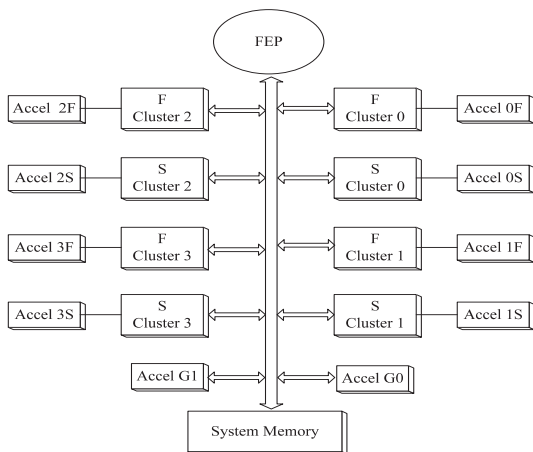


图 2 PAAG 阵列机完整架构图

处理器支持 SIMD(单指令多数据流)和 MIMD(多指令多数据流)并行编程模式。在 SIMD 模式下,ALU 执行来自行、列簇控制器的指令序列(外部指令),指令操作的数据来自本地数据存储或者邻接存储,控制器执行指令序列。在 MIMD 模式下,ALU 执行来自本地指令存储中的指令序列,指令操作的数据可以源自本地数据存储、邻接存储以及远程处理器。

通过结合 SIMD、MIMD 并行模式,PAAG 多态阵列处理器能够实现高复杂度的算法高效并行处理,同时特殊的通信方式为并行效率的提高提供了可靠保障。因此,PAAG 为图形渲染和 OpenVX 实现的并行处理提供了可靠的平台。

## 2 图形渲染管线及相关算法

本节介绍基于 PAAG 的固定图形渲染管线架构调整和优化后的实现。优化后的固定渲染管线由以下的功能模块组成:几何变换、定点染色、投影变换、剪裁、视窗变换和消隐、光栅化、像素染色以及片段操作<sup>[8]</sup>。以下是对整个流水线中重要功能模块的介绍以及相关算法的优化:

几何变换与投影变换:几何变换设置模型的位置和方向,实现图像的动画效果;投影变换是决定物体如何映射到屏幕;两种变换均通过矩阵运算来实现。

顶点染色:利用光照和材料的相互作用来模拟更加接近于现实的场景。文中采用经典 Phong 模型来完成对顶点的渲染。

剪裁与消隐:分为平面剪裁和视景体剪裁。剪裁模块实现了对所需场景外的图元剔除,保留了渲染场景内的图元。平面剪裁使用空间中的一个或者多个平面,将平面外的图元予以剪裁。文中可视体的剪裁在经典的 Sutherland-Hodgman (S-H) 算法<sup>[9]</sup>的基础上,对新产生的裁剪点进行属性差值计算,再对计算后产生的新点所构成的图元进行新一轮剪裁,如此循环直至图元的所有边都计算完成为止。

消隐主要的实现功能是在要处理的图元中针对那些无法看到的图元进行丢弃(剔除)操作,在消隐操作完成后,由于那些图元的剔除,后续的流水线模块中的计算量相应也较少了,这样一来图形处理效率便提高了。视窗变换则实现了将三维的物体显示到二维的视口平面。文中采用简化后的视窗变换公式对图元数据进行处理。公式如下:

$$\text{new\_}x = (x \times \frac{w}{2}) + (\frac{w}{2} + x_{\text{can}})$$

$$\text{new\_}y = (\frac{h}{2} \times y) + (\frac{h}{2} + y_{\text{can}})$$

$$\text{new\_}z = (\frac{f-n}{2}) \times z + (\frac{f+n}{2})$$

像素染色:在渲染流水线中,采用纹理映射的方法实现图形与现实场景的逼近,通常是将光栅化后生成的片段进行纹理贴图,从而增加了物体表面的细节。文中所设计的渲染管线中,像素染色将进行如下操作:光栅化、纹理的环绕方式、纹理滤波处理等。其中,光栅化就是把几何数据和像素数据转换为片段的过程,

每个片段对应于帧缓冲区中的像素值;纹理的环绕方式和滤波处理时需要考虑优先使用纹理对象来帮助控制纹理贴图潜在的缓存和定位问题。

片段操作:在 OpenGL 渲染管线中,经过像素染色处理所输出的片元要再进行一些特殊处理,这些处理决定了是否需要将此片元以及以怎样的方式将其写入帧缓冲区。片元操作所描述的就是片元在写入帧缓冲区之前可能对它进行的所有操作。如果片元在某个操作中未通过,后面的操作将不再进行,片元将会直接被剔除。在 OpenGL 标准中,片元操作包括:剪裁测试、Alpha 测试、模板测试、深度测试、混合和逻辑操作。

## 3 图像处理算法

文中根据国际标准组织 Khronos 最新提出的机器视觉计算标准 OpenVX1.0,完成了标准中各个图像算法在 PAAG 阵列机上的并行实现<sup>[10]</sup>。使用的图像是 256 \* 256 的灰度图像,图像的每个像素以 32 位的形式存储在 PAAG 阵列机的共享存储中。

对图像数据的处理,可以将其分为:像素级处理、特征级处理和对象级处理。OpenVX 多数 kernel 函数针对图像做像素级(或者小区域内)处理,像素级图像处理一般包含:点处理、局部处理、全局参数抽取处理、迭代处理和跟踪处理。限于篇幅,文中着重介绍 OpenVX 核函数中涉及点处理算法的像素累加(Accumulate)、图像局部处理中的中值滤波(Median Filter)、高斯滤波(Gaussian Filter)算法、Sobel3\*3 图像滤波算法;图像全局参数抽取处理中的直方图(Histogram)、迭代处理中的积分图像(Integral);跟踪处理的角点检测。算法具体描述如下:

像素累加即对输入图像和累加图像进行累加计算,公式为:

$$\text{accum}(x, y) = \text{accum}(x, y) + \text{input}(x, y)$$

中值滤波<sup>[11]</sup>是一种非线性平滑技术,将每一像素点的灰度值设置为该点某邻域窗口内的所有像素点灰度值的中值,文中邻域大小为 3 \* 3。

高斯滤波对已有的输入图像进行高斯卷积计算,使用的卷积矩阵为:

$$K_{\text{Gaussian}} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * \frac{1}{16}$$

Sobel3\*3 图像滤波算法通过 Sobel 卷积算子进行卷积运算求解一幅图像沿着 x 和 y 方向的梯度值。

直方图是对整幅图像中像素强度分布的图形表达方式,它统计了每一个强度值所具有的像素个数。

积分图像中的像素点是由输入像素点与原图像像素点上方和左侧像素点的值求和之后与左上角像素值



作差得到,公式如下所示:

$$\begin{aligned} \text{sum}(x,y) = & \text{in}(x,y) + \text{sum}(x-1,y) \\ & + \text{sum}(x,y-1) - \text{sum}(x-1,y-1) \end{aligned}$$

Harris Corners, Fast Corners 又叫做拐点,拐点是指水平、竖直两个方向上变化均较大的点,包括图像的极值点、线段的终点、曲线曲率最大的点或者某个方向上属性最大的点。

4 并行化设计

主要的并行计算方法有数据并行计算 (Data Level Parallel, DLP) 和操作并行计算 (Operation Level Parallel, OLP)<sup>[12]</sup>。数据并行计算以 SIMD 为基础,主要用于算法层的并行;操作并行主要是指 FPGA 和 ASIC 类的计算方法<sup>[13]</sup>,充分利用底层的操作并行度,而不是算法并行度。

4.1 图形并行渲染设计

PAAG 所提供的 PE 之间的阻塞通信模式为图形的固定渲染管线提供了灵活的支持。在图 1 中的二维簇中,PAAG 中的邻接通信共享存储可以将阵列中的 PE 以 Z 形串行的方式有机地组织在一起。文中将固定渲染管线中串行连接的子模块与簇中的 PE 进行了一对一的映射,并通过 PE 之间的邻接通信共享存储的阻塞模式来实现各个渲染子模块之间的握手协议。实验中,根据渲染管线中处理单元元素的不同,以光栅化为分界线,将渲染管线分为渲染前端和渲染后端。其中,渲染前端以基本图元点、线、三角形作为基本处理元素,渲染后端以片元作为基本处理元素。另外,在渲染管线的整个处理过程中,因为各个渲染子模块的负载有所不同,所以如何合理平衡各个模块在执行过程中的负载大小,以便提高整个流水管线的执行效率也是必须要考虑的问题。通过分析,在各个渲染子模块中,剪裁模块的负载相对最重,因为剪裁模块要实现多数图元的剪裁操作,在完成剪裁操作之后图元要重新计算节点属性并进行重新装配,因此负责剪裁操作的处理器在工作时其他的 PE 会处于闲置状态,这样会导致流水线效率降低;同样的瓶颈在顶点染色、像素操作和片段操作这些负载相对较重的模块中也存在。为了解决各个子模块执行过程中负载不均衡的问题,文中通过优化分块方法,将负载过重的子模块进行再分块,如图 3 所示。

如此分段处理的方法能够较好地实现负载均衡,增加了重负载部分的计算效率,很大程度上降低了瓶颈部分的负载,更好地实现了流水线中模块的操作并行。

当要对图形渲染管线实现的数据进行并行处理时,将阵列机中的多个 PE 组成一个阵列单元,每个单

元中的多个 PE 可以各自执行一段完整的渲染程序,每段渲染程序以独立的线程方式执行。图形渲染管线中的几何变换、顶点渲染以及像素染色等分别被当作一个独立执行的线程。PE 之间独立操作而互不影响,在一个单元中可同时处理多个顶点,每个阵列单元中的每个 PE 完成对一个顶点的所有渲染操作。当一个单元中仅仅有一个 PE 时,PAAG 中所进行的操作级并行即为线程级并行操作。

实验数据表明,受负载均衡影响,改进后统一渲染中的数据级并行处理效率要略优于粗粒度操作级并行,两种并行处理方式都能够使数据处理的速度得到显著提高。

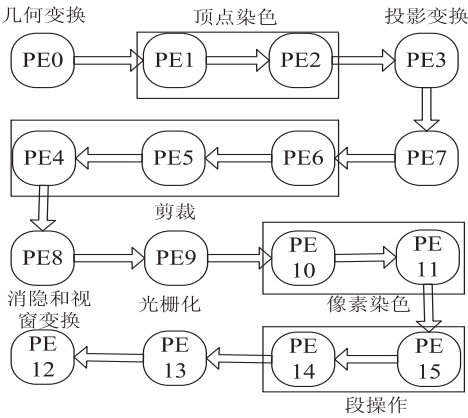


图 3 优化分块流水线

对于程序处理的结果,将程序在 PAAG 阵列机上的执行的时钟数作为数据分析对象,用加速比作为直接衡量 PAAG 并行执行效率的性能评估标准。加速比能够直观地体现在并行计算机上,尤其是并行阵列处理机器上执行并行算法以求解实际问题所能获得的效率提高。所谓加速比,狭义来讲,即是一个特定的算法或者应用,并行执行相对于串行执行的比值。公式如下:

$$S = \frac{T_p}{T_s}$$

其中,  $T_s$  为算法串行执行的时间;  $T_p$  为算法并行执行的时间;两者的比值即为加速比。实验中主要模块相关算法加速比如表 1 所示。

表 1 图形渲染管线主要模块相关算法加速比统计表

算法	顶点染色	剪裁	像素染色
$S_1$	1	1	1
$S_2$	1.83	1.9	1.85
$S_4$	3.69	3.81	3.70
$S_8$	7.27	7.31	7.29
$S_{16}$	14.89	14.10	14.56
$S_{32}$	29.78	30.36	30.43
$S_{64}$	60.92	61.36	62.16

表 1 中,  $S$  即为实验所得的加速比, 当下标为 1 的时候表示的是算法串行执行的加速比, 即对应于算法在一个 PE 上的执行结果。文中分别使用 2、4、8、16、32、64 个 PE 进行算法的并行处理, 并且分别计算出加速比统计于表中。

4.2 图像算法并行化设计

文中针对图像算法的并行化设计<sup>[12]</sup>同样使用数据并行和操作并行相结合的方法。为实现图像处理算法的数据并行化, 首先要进行的工作是对图像进行分块。针对像素级图像, 将较大的图像像素数据集分解为多个子像素数据集, 每个处理器对不同的像素数据子集执行相同的操作, 以达到并行的目的。在此基础上辅以操作并行方法<sup>[14]</sup>, 即将算法进行分解, 分解为若干个独立执行的部分, 同时分块处理后的数据通过不同的通信方式实现对图像流水线式处理, 从而得到最终的处理结果。如此可以以更少的 PE 来实现更高的 PE 利用率, 节约资源, 提高效率。

(1) 数据并行设计。

在数据分解的过程中, 需要考虑在对图像边界邻域像素进行处理时, 采用边界复制、边界补零等方法对边界进行处理。例如, 在局部处理、迭代处理和跟踪处理中针对邻域像素, 采用边界复制法对图像的边界像素进行处理, 即将相邻子块边界像素同时分配到相邻两个 PE 中。以此类推, 可以将图像的像素值分配到 4、8 个等更多的 PE 上。如此就实现了数据的分块处理, 这是实现并行计算的重要一步。当分配完的数据都加载到各个 PE 上, 并且加载指令完成后, 即可进行算法的计算部分, 以实现算法的数据并行化。

(2) 操作并行设计。

对于算法的操作并行, 以中值滤波和积分图像为

例, 详细叙述算法的优化和实现过程。

中值滤波算法的细节即依次求出每个邻域窗口的中值, 文中采用的是  $3 \times 3$  的窗口, 传统方法为每次对九个像素值进行比较排序, 取中间值作为新的像素值, 这种方法的时间复杂度高。文中将算法进行优化, 过程为将九个像素分成三个小组, 每个小组三个像素, 每次对同一组的三个像素值进行比较, 取第一组的最大值, 第二组的中间值和第三组的最小值, 再对这三个值进行排序, 排序后的中值即为九个像素的中值。这样每次只对三个像素值进行排序, 降低了程序执行的时间复杂度, 同时减少了内存消耗。

在高斯滤波算法中, 如图 1 所示, 在进行数据分块之后, 文中的操作并行的处理方法是高斯卷积核按列分别加载到 PE0、PE1、PE2 上, 执行计算时, 按行从图像块中取三个数据加载到对应的 PE 中, 进行乘积和加法操作。如此经拓展, 每次可以取九个像素值分别映射到九个 PE 上执行卷积操作。

计算积分图像时, 根据文中第 3 节中的公式可知, 任意一点  $(x, y)$  的积分像素值是指输入像素点与原图像像素点上方和左侧像素点的值求和之后与左上角像素值作差得到, 即每个新的像素值是由之前所计算出的多个像素值共同决定的。针对这一特征, 文中提出这样的并行方法, 使用阻塞模式。每个 PE 处理一行像素, 计算出的每个像素值传给处理下一行像素的 PE, 依次类推, 在此过程中, 每个 PE 均为阻塞模式, 当接收到上一个 PE 传来的数据时才进行相应的计算, 保证了算法的正确性, 同时也提高了算法并行的效率。

(3) 实验结果。

对于图像算法并行实现的实验结果, 同样使用并行化加速比作为衡量标准, 实验结果如表 2 所示。

表 2 图像相关算法加速比统计表

算法	Histogram	Accumulate	Integral Image	Media Filter	Harris Corners	Faster Corners
$S_1$	1	1	1	1	1	1
$S_2$	1.89	1.88	1.84	1.81	1.90	1.79
$S_4$	3.77	3.58	3.58	3.59	3.60	3.73
$S_8$	7.16	7.31	7.24	7.19	7.20	6.98
$S_{16}$	14.92	14.80	14.78	13.94	13.88	14.34
$S_{32}$	28.98	29.76	29.88	30.40	31.01	29.78
$S_{64}$	60.89	62.53	60.86	61.61	60.95	61.26

同样地, 在 OpenVX 核函数的并行实现中, 也采用多个不同的 PE 实现算法的并行处理。如表 2 所示, 多个 PE 执行后的加速比与所使用的 PE 的个数基本成正比, 加速效果明显。

从实验结果中可以看出, PAAG 阵列机对固定渲染管线和 OpenVX 核函数能基本实现线性加速, 大大

提高各个算法的计算效率。

5 结束语

文中通过对经典图形渲染管线的优化设计和基于 OpenVX 标准的图像算法的研究, 运用了数据并行和操作并行相结合的方法, 在自主研发的 PAAG 多态同

构阵列机上实现了算法的并行化设计。在统一图形渲染管线中,两种并行处理方式很好地提高了数据处理的速度;同样利用像素级图像处理数据量大、数据相关性小等特点,在阵列机上很好地实现了图像算法的并行化,并且获得了较高的加速比,实现了图像处理加速。通过文中的研究表明,PAAG 阵列机在并行处理图形图像算法及其相关应用,提高程序执行效率,降低功耗等方面具有独特的优势。在以后的研究工作中,将针对阵列机的整体设计、通信机制等方面作进一步优化,以追求其在高性能并行计算方面取得更多的突破。

#### 参考文献:

- [1] Tomas A M, Haines E, Hoffman N. Real-time rendering[M]. 2nd ed. 夏文宇, 胡艳祥, 译. 北京: 清华大学出版社, 2002.
  - [2] Macedonia M. The GPU enters computing's mainstream[J]. Computer, 2003, 36(10): 106-108.
  - [3] Rainey E, Gautam S. The OpenVX™ Specification Version 1.0 [S]. USA: Khronos Vision Working Group, 2014.
  - [4] Barnes G H, Brown R M, Kato M, et al. The ILLIAC IV computer[J]. IEEE Transactions on Computers, 1968, C-17(8): 746-757.
  - [5] Lindholm E, Nickolis J, Oberman S, et al. NVIDIA Tesla: a unified graphics and computing architecture[J]. IEEE Micro, 2008, 28(2): 39-55.
  - [6] Li T. PAAG: a polymorphic array architecture for graphics and image processing[C]//Proceedings of the 2012 fifth international symposium on parallel architectures, algorithms and programming. [s. l.]: IEEE Computer Society, 2012: 242-249.
  - [7] Moreno M C C, Auzinger T. General-purpose graphics processing units in service-oriented architectures [C]//Proc of IEEE 6th international conference on service-oriented computing and applications. Koloa, HI: IEEE, 2014: 260-267.
  - [8] Kato S, Lakshmanan K, Rajkumar R, et al. TimeGraph: GPU scheduling for real-time multi-tasking environments [C]//Proceedings of the USENIX annual technical conference. [s. l.]: USENIX, 2011.
  - [9] Angel E. Interactive computer graphics[M]. Beijing: Tsinghua University Press, 2006.
  - [10] Ao Qian, Chen Rongguan, Ning Ning, et al. High-definition image processing algorithm and digital platform design [C]//Proc of IEEE 12th international conference on computer and information technology. Chengdu: IEEE, 2013: 798-800.
  - [11] Dong Fuguo, Li Yiling. Study on fast algorithm of median filtering based on DC and ICS method [C]//Proc of international conference on wireless communications & signal processing. Nanjing: IEEE, 2009: 1-5.
  - [12] Trahanian P E, Venetsanopoulos A N. Color image enhancement through 3-D histogram equalization [C]//Proc of IAPR international conference on pattern recognition. The Hague: IEEE, 1992: 545-548.
  - [13] Chiuchisan I. A new FPGA-based real-time configurable system for medical image processing [C]//Proc of e-health and bioengineering conference. Iasi: IEEE, 2014: 1-4.
  - [14] Prajapati H B, Vij S K. Analytical study of parallel and distributed image processing [C]//Proc of international conference on image information processing. Himachal Pradesh: IEEE, 2011: 1-6.
- 
- (上接第 60 页)
- ules using support vector machines[J]. Journal of Systems and Software, 2008, 81(5): 649-660.
  - [6] Wang J, Shen B, Chen Y. Compressed C4.5 models for software defect prediction [C]//Proc of 2012 12th international conference on quality software. Washington D C: IEEE, 2012: 13-16.
  - [7] Wang T, Li W. Naive Bayes software defect prediction model [C]//Proc of 2010 international conference on computational intelligence and software engineering. [s. l.]: [s. n.], 2010: 1-4.
  - [8] Song Q, Jia Z, Shepperd M, et al. A general software defect-proneness prediction framework [J]. IEEE Transactions on Software Engineering, 2011, 37(3): 356-370.
  - [9] Sun Z, Song Q, Zhu X. Using coding-based ensemble learning to improve software defect prediction [J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 2012, 42(6): 1806-1817.
  - [10] Wang H, Khoshgoftaar T M, Seliya N. How many software metrics should be selected for defect prediction? [C]//Proc of FLAIRS. Palm Beach: [s. n.], 2011.
  - [11] 王 培, 金 聪, 葛贺贺. 面向软件缺陷预测的互信息属性选择方法[J]. 计算机应用, 2012, 32(6): 1738-1740.
  - [12] Lanubile F, Visaggio G. Evaluating predictive quality models derived from software measures: lessons learned [J]. Journal of Systems and Software, 1997, 38(3): 225-234.
  - [13] 缪林松. 基于代价敏感神经网络算法的软件缺陷预测 [J]. 电子科技, 2012, 25(6): 75-78.
  - [14] Belkin M, Niyogi P. Laplacian eigenmaps for dimensionality reduction and data representation [J]. Neural Computation, 2003, 15(6): 1373-1396.

基于PAAG的图形图像算法的并行实现

作者：[孙建](#), [李涛](#), [李雪丹](#), [SUN Jian](#), [LI Tao](#), [LI Xue-dan](#)

作者单位：[孙建, SUN Jian\(西安邮电大学 计算机学院, 陕西 西安, 710121\)](#), [李涛, 李雪丹, LI Tao, LI Xue-dan\(西安邮电大学 电子工程学院, 陕西 西安, 710121\)](#)

刊名：[计算机技术与发展](#) 

英文刊名：[Computer Technology and Development](#)

年, 卷(期)：2015, 25(11)

引用本文格式：[孙建](#). [李涛](#). [李雪丹](#). [SUN Jian](#). [LI Tao](#). [LI Xue-dan](#) [基于PAAG的图形图像算法的并行实现](#) [期刊论文]

-[计算机技术与发展](#) 2015(11)