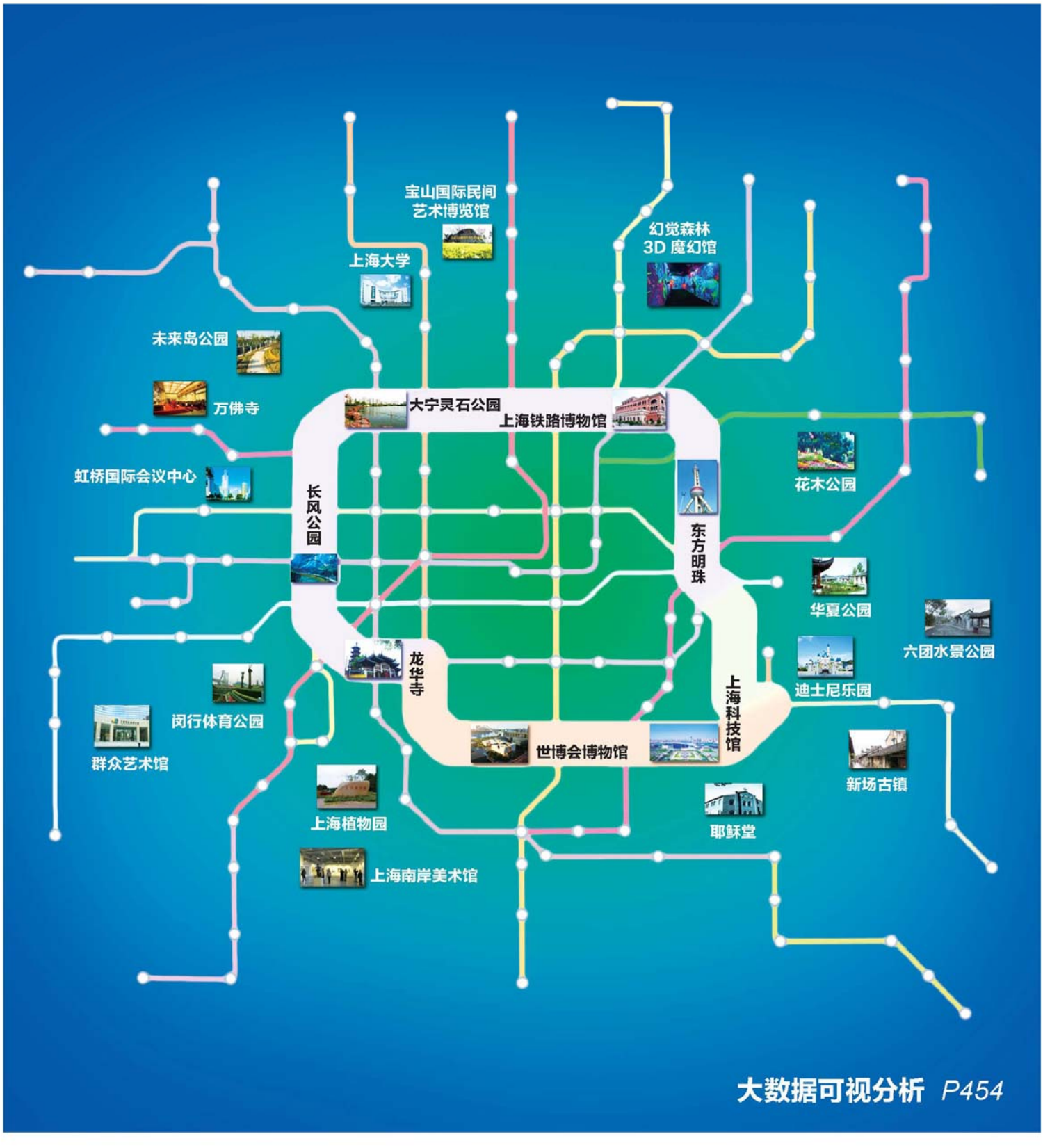
JOURNAL OF IMAGE AND GRAPHICS

主办：中国科学院遥感与数字地球研究所
中国图象图形学学会
北京应用物理与计算数学研究所

# 中国图象图形学报

2015
04
VOL.20

ISSN1006-8961
CN11-3758/TB




大数据可视分析 *P454*

# 中国图象图形学报

刊名题字：宋健 | 月刊（1996年创刊）

**主管单位** 中国科学院
**主办单位** 中国科学院遥感与数字地球研究所
中国图象图形学学会
北京应用物理与计算数学研究所

**主　　编** 李小文
**编辑出版** 《中国图象图形学报》编辑出版委员会
**邮政信箱** 北京9718信箱
**邮　　编** 100101
**电子信箱** jig@radi.ac.cn
**电　　话** 010-64807995
**网　　址** www.cjig.cn

**广告经营许可证** 京朝工商广字第0361号
**总 发 行** 北京报刊发行局
**订　　购** 全国各地邮局
**海外发行** 中国国际图书贸易集团有限公司
（邮政信箱：北京399信箱　邮编：100048）
**印刷装订** 北京科信印刷有限公司




第20卷第4期（总第228期）
2015年4月16日


中国精品科技期刊
中国国际影响力优秀学术期刊
中国科技核心期刊




# Journal of Image and Graphics

Title inscription: Song Jian | Monthly, Started in 1996

**Superintended by** Chinese Academy of Sciences
**Sponsored by** Institute of Remote Sensing and Digital Earth, CAS
China Society of Image and Graphics
Institute of Applied Physics and Computational Mathematics

**Editor-in-Chief** LI Xiaowen
**Editor, Publisher** Editorial and Publishing Board of Journal of Image and Graphics
**P.O.Box** 9718, Beijing, P.R.China
**Zip code** 100101
**E-mail** jig@radi.ac.cn
**Telephone** 010-64807995
**Website** www.cjig.cn

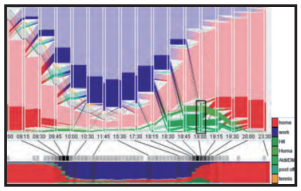
**Distributed by** Beijing Bureau for Distribution of Newspapers and Journals
**Domestic** All Local Post Offices in China
**Overseas** China International Book Trading Corporation
(P.O.Box 399, Beijing 100048,P.R.China))
**Printed by** Beijing Kexin Printing Co., Ltd.

CN 11-3758/TB
ISSN 1006-8961
CODEN ZTTXFZ

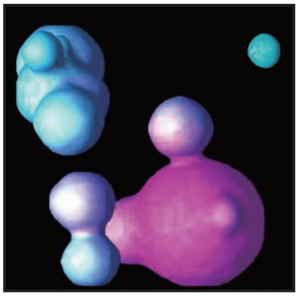
**国外发行代号** M1406
**国内邮发代号** 82-831
**国内定价** 50.00元

2015年4月 第20卷 第4期（总第228期）
中国图象图形学报
Zhongguo Tuxiang Tuxing Xuebao

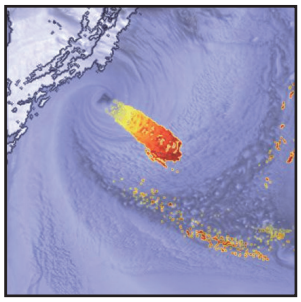
# 目次



城市道路交通数据可视分析综述(第0454页)



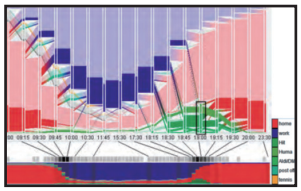
可视数据清洗综述(第0468页)



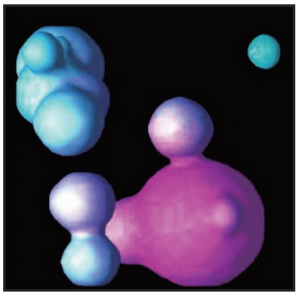
非对称高斯函数的时变体数据特征跟踪及可视化(第0576页)

Volume 20, Number 4
Published April 16, 2015

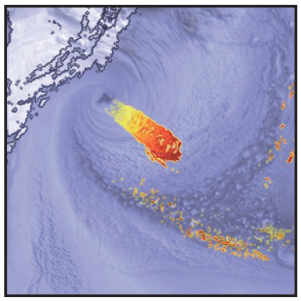
# CONTENTS
## JOURNAL OF IMAGE AND GRAPHICS



Visual analytics of urban road transportation data: a survey (P0454)



Survey of visualization data cleaning(P0468)



Time-varying volume visualization and feature tracking based on asymmetric Gaussian function(P0576)

**中图法分类号**：TP301.6　　**文献标识码**：A　　**文章编号**：1006-8961(2015)04-0527-06

**论文引用格式**：Fu H,Xie Y F. High efficiency triangle raster algorithm in GPU based on TBR architecture[J]. Journal of Image and Graphics,2015,20(4):0527-0532.[符鹤,谢永芳. TBR 架构 GPU 中三角形高效光栅化[J]. 中国图象图形学报,2015,20(4):0527-0532.][DOI:10.11834/jig.20150409]

# TBR 架构 GPU 中三角形高效光栅化

符鹤，谢永芳

中南大学信息科学与工程学院，长沙　410083

**摘　要**：**目的**　在基于分块渲染(TBR)架构的 GPU 中，三角形光栅化的速度对芯片的性能影响很大，采用传统的光栅化方法会产生大量多余的像素，无法发挥 TBR 架构的优势。**方法**　提出了一种该架构下的高效三角形光栅化算法，该算法充分利用了分块渲染的特点，通过预处理计算出三角形在每一个块内的绘制参数，得出三角形与块边界的位置关系，并将其随三角形的分块信息一起写入存储器，在光栅化阶段采用了 Bresenham 算法，利用生成的三角形边得到在每一个块内的扫描水平线，进而生成水平线上的每一个像素。**结果**　经过理论分析，该算法的光栅化效率可以达到 83% 以上，甚至接近 100%，在 FPGA 原型验证系统上对该算法进行了功能和性能的验证。**结论**　提出的三角形光栅化算法，能够适应 TBR 的架构，实际测试像素填充率与频率高一倍的 ATI M9 相当，因此该算法能够达到较高的光栅化效率。

**关键词**：三角形；光栅化；分块渲染；原型验证

# High efficiency triangle raster algorithm in GPU based on TBR architecture

Fu He, Xie Yongfang

*School of Information Science and Engineering, Central South University, Changsha* 410083, *China*

**Abstract**: **Objective** In GPUs based on TBR architecture, the efficiency of triangle raster greatly influences chip performance. Traditional algorithms generate severalredundant pixels and lose the advantages of TBR architecture. **Method** A high-efficiency triangle raster algorithm is proposed in this paper. The features of TBR architecture are fully utilized in this algorithm. The draw parameters of each tile are calculated in the pre-processing stage. The position relationship between the triangle and the tile boundary is obtained. All data are committed to memory. In the raster stage, the Bresenham algorithm is used to obtain all the horizontal scan lines in every tile untilall pixels of the scan lines are eventually generated. **Result** After theoretical analysis, the raster efficiency of the raster operation can reach 83% and even approach 100%, depending on the triangle shape. The function and performance of this algorithm are evaluated with a FPGA proto type verification system. **Conclusion** To adapt to the TBR architecture, a new triangle raster algorithm is proposed in this paper. The pixel fill rate is equivalent to ATI M9, which has twice the clock rate. Therefore, the algorithm can achieve high raster efficiency.

**Key words**: triangle; raster; tile based rendering; prototype verification

**收稿日期**:2014-10-29;**修回日期**:2014-12-18

**基金项目**:国家高技术研究发展计划(863)基金项目(2013AA040301);国家自然科学基金创新研究群体科学基金项目(61321003)

**第一作者简介**:符鹤(1978— ),男,中南大学信息科学与工程学院计算机应用专业博士研究生,主要研究方向为计算机图形学,大规模集成电路设计。E-mail: ehuf@foxmail.com

**Supported by**: National High Technology Research and Development Program of China (2013AA040301)

## 0 引 言

GPU(图形处理器)现已成为计算机系统中的一个重要组成部分,虽然目前 GPU 的一种重要应用是通用计算,但其最基本的功能仍然是图形生成,GPU 从诞生至今经历了固定流水线结构、顶点和像素可编程结构、统一渲染结构等发展阶段,无论哪个阶段,光栅化都是 GPU 中不可或缺的一个功能单元。光栅化即是将基本图元(点、线段、三角形)转化成一个个像素的过程(最终显示在光栅显示器上),作为图形处理流水线的一个重要步骤,其效率的高低直接影响到 GPU 做图形加速的性能。

随着智能手机等移动设备的普及,嵌入式 GPU 也进入了快速发展的时期,与通用 GPU 不同的是,嵌入式 GPU 大多采用分块渲染(TBR)的架构来实现,通过将绘图区划分成小块,按块为单位生成其中的像素,以此降低对存储器的频繁访问。

基于 TBR 架构研究中的三角形光栅化,提出了一种高效的边界扫描光栅化算法,并对其性能做了验证和对比。

GPU 中 3 维绘图的硬件基本架构有两种:一种是直接渲染(IMR),一种是 TBR。IMR 的架构如图 1 所示。

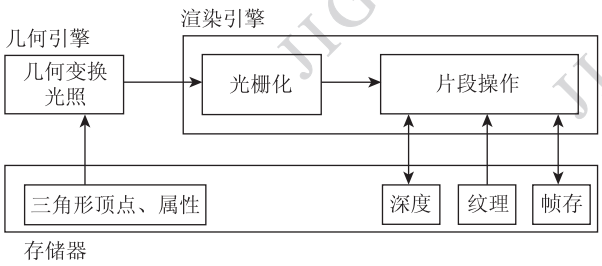



图 1 IMR 架构示意图

Fig. 1 Schematic diagram of IMR architecture

针对以上架构提出的相关硬件实现有很多,其中有针对几何运算加速而采用的可编程处理器(vertex shader),也有针对片段操作的可编程处理器(pixel shader),而整个硬件设计除了考虑到需要提供足够的浮点运算能力之外,另一个需要考虑的就是存储器的带宽。许多研究表明[1-2],后端像素级的渲染系统瓶颈主要在存储器的带宽,因为渲染系统对纹理的读取以及深度缓存、帧存读写的需求非常高,而且由于大量物体表面的互相遮盖,造成许多无效的像素重复读写造成浪费。

为了有效地去除不必要的计算和存储器带宽,[1,3-4]提出了基于分块的绘图架构,该方法利用改变绘图的顺序,将同一个块的图元全部收集在一起绘制,并利用片上存储器来避免多余的存取。

分块式绘图架构将屏幕显示范围切割成数个固定大小并没有重叠的区块,称之为 Tile。在绘图过程中,将所有绘制的三角形,按照其涵盖的范围,分至各个 Tile 的缓冲区中等待绘制,当收集完所有需要绘制的三角形之后,再依照 Tile 的顺序,绘制出各个 Tile 中的三角形,可以在硬件上使用一个刚好是 Tile 大小的片上存储器,将过程中所需要的图像及深度缓存区置于存储器中,绘制过程中只需要使用片上存储器,减少了大量无用的存储器读写,特别是对低功率存储器或嵌入式系统上具有非常大的优势,本文的研究就是基于该架构开展的。TBR 的架构如图 2 所示。

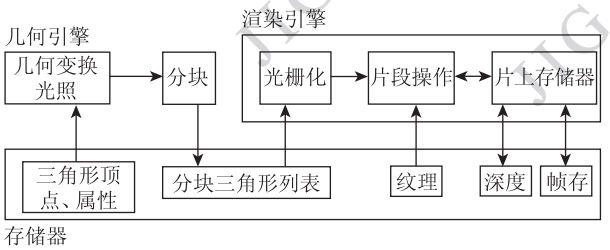



图 2 TBR 架构示意图

Fig. 2 Schematic diagram of TBR architecture

由顶点描述的图元转换成对应到屏幕的像素的过程称为光栅化。在使用顶点来描述一个图元时,能获得该图元的顶点坐标以及顶点的颜色等属性,但在输出显示装置上却只能表示 2 维、离散的坐标点,所以就必须经由光栅化操作把顶点所描述的范围转换成屏幕上所对应的像素位置,如图 3 所示。

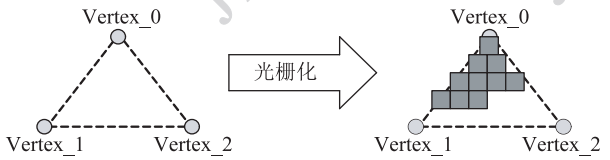



图 3 三角形光栅化示意图

Fig. 3 Schematic diagram of triangle raster

基于 TBR 架构,可以有多种扫描转换的方法,主要有块内顺序扫描[5]、边缘扫描等[5]。不同的转

换方法对应的效率也不同。顺序扫描是在包含三角形的块内顺序来回扫描，测试完一个像素后，将该位置代表的像素加上移动的 $x$，$y$ 位移量得到下一个像素的值再继续判断，该算法实现简单，但缺点也很明显，一般三角形在块内只占一部分，如此扫描会产生大量无效的像素，效率非常低；边缘扫描算法是先找出三角形的边缘，再依扫描线一一填满的扫描转换，该算法并不会浪费无用的时间去处理三角形以外的像素，整体效率会比顺序扫描高出很多。而本文针对 TBR 的架构提出了一种高效的三角形光栅化方法也是属于一种边缘扫描算法。

# 1　TBR 架构下的三角形光栅化

三角形高效光栅化算法包含了三角形分块、三角形预处理和光栅化 3 个部分。

## 1.1　三角形分块

在 TBR 架构中，一个三角形的绘制会被分割到许多块中进行，而一个三角形究竟会跨越几个块，被称为三角形的覆盖因子（overlap factor），如何判断三角形涵盖到哪些块十分重要，有很多种判断方法[1,3-4]，最常见的做法还是边界盒测试（bound box test），可以利用一个点对三角形的三边直线方程式的值来测试一个区域是否与三角形相交[2]。

假设一个三角形的 3 个顶点依顺时针顺序为 $A$、$B$、$C$，有一个宽为 $w$ 的正方形区块，左下角的坐标为 $(x_{LL},y_{LL})$，以此点代表该区块对于三角形 $ABC$ 三边的关系式可以整理为

$$\begin{cases} E_{\overline{AB}}(x_{LL},y_{LL}) \geqslant (w-1)\times \\ \qquad \{\min\mid 0, dx_{\overline{AB}}, -dy_{\overline{AB}}, dx_{\overline{AB}}-dy_{\overline{AB}}\} \\ E_{\overline{BC}}(x_{LL},y_{LL}) \geqslant (w-1)\times \\ \qquad \{\min\mid 0, dx_{\overline{BC}}, -dy_{\overline{BC}}, dx_{\overline{BC}}-dy_{\overline{BC}}\} \\ E_{\overline{CA}}(x_{LL},y_{LL}) \geqslant (w-1)\times \\ \qquad \{\min\mid 0, dx_{\overline{CA}}, -dy_{\overline{CA}}, dx_{\overline{CA}}-dy_{\overline{CA}}\} \end{cases} \tag{1}$$

式中

1）$dx_{\overline{AB}}=x_B-x_A$ 为三角形 $AB$ 边的 $x$ 方向坐标差，$dy_{\overline{AB}}=y_B-y_A$ 为三角形 $AB$ 边的 $y$ 方向坐标差，其余 $dx_{\overline{BC}}$、$dy_{\overline{BC}}$、$dx_{\overline{CA}}$、$dy_{\overline{CA}}$ 同理。

2）$E_{\overline{AB}}(x,y)=(x-x_A)\times(y_B-y_A)-(y-y_A)\times(x_B-x_A)$　　(2)

称为三角形 $AB$ 边的边缘函数，其余 $E_{\overline{BC}}$、$E_{\overline{CA}}$ 同理。

之所以取 $(w-1)$ 是因为屏幕上像素点的坐标是以左下角坐标表示的，所以只需要判断到 $(w-1)$ 即可判断该正方形与三角形的位置关系。

当式（1）3 个公式均成立时，表示该正方形区块至少有部分或是全部落在三角形之内，反之只要有一个不成立，则表示该区块完全落在三角形某个边之外，两者没有交集。

## 1.2　三角形预处理

若三角形与某块有相交，称之为三角形属于某一块。为了在光栅化阶段能够快速实现光栅化，需要在完成分块之后计算出三角形在块内的扫描线，目的是减少光栅化阶段生成不属于当前块的无效扫描线，最终在光栅化阶段需要生成的是水平扫描线，而水平扫描线的起始点和终止点可以在采用 Bresenham 算法生成边界点之后根据其与块左右边界的位置关系进行裁剪，在预处理阶段处理左右边界会大大增加光栅化算法复杂度，所以三角形在块内的左右边界在此阶段不进行处理。而上下边界则决定了使用 Bresenham 算法时的端点坐标，因此需要重点关注块的上下边界，三角形与块上下边界的位置关系有如图 4 所示的 8 种（图中的顶点 $B$ 表示三角形在 $Y$ 方向上的最低点，顶点 $M$ 表示次高点，顶点 $T$ 表示最高点，粗实线包围区域的上下边界即为需要确定的范围）。

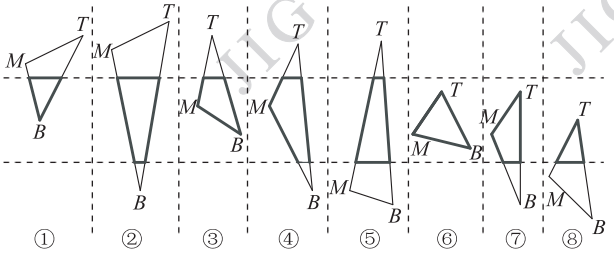



图 4　三角形与所属块的位置关系

Fig. 4　The position relation of triangle and tile belonged

分别计算三角形三条边与块的上下边界交点坐标。首先计算 3 条边的斜率倒数

$$k=(x_1-x_0)/(y_1-y_0) \tag{3}$$

式中，$(x_0,y_0)$ 和 $(x_1,y_1)$ 为线段的两个端点坐标。

然后计算 3 条边与边界的交点，即

$$x=(y-y_0)\times k+x_0 \tag{4}$$

式中，$y$ 根据计算的边界不同分别取下边界和上边界，计算结果转化成整数（Bresenham 算法是整数算法）。

最后根据三角形与所属块的位置关系得到各有效交点的坐标，如对图 4 中的情形④得到的坐标为三角形 *BM* 边、*BT* 边分别与块的下边界交点，三角形的顶点 *M*，三角形 *BM* 边、*BT* 边分别与块的上边界交点，其余情形类似得到。这些交点信息也需要与分块信息一起写入分块存储器。

### 1.3 三角形光栅化

从存储器读出三角形的分块信息，进行光栅化操作，采用 Bresenham 算法[6-7]生成一条主要边（主要边是指从顶点 *B* 到顶点 *T* 的边界，包括在其上的交点，如图 4 所示）、两条次要边（次要边是指从顶点 *B* 到顶点 *M* 和从顶点 *M* 到 *T* 的边界包括在其上的交点，如图 4 所示），在使用 Bresenham 算法生成线段时，对算法进行修改，只有当垂直方向的坐标发生变化时才会输出有效数据，每一次变化输出水平线的一个起始点，如图 5 所示。

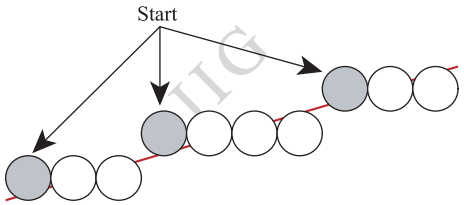



图 5 Bresenham 算法生成线段时输出水平线起始点

Fig. 5 Start point of the horizontal line generated by Bresenham algorithm

光栅化实现步骤为：

1）线段端点分配，分配主要边至 Bresenham 线段生成单元 $U_A$，分配一条次要边至 Bresenham 线段生成单元 $U_B$（需要两个并行的 Bresenham 单元，分别命名为 $U_A$ 和 $U_B$）；

2）$U_B$ 单元线段生成完毕后，再分配第二条次要边至该单元；

3）两个线段生成单元 $U_A$ 和 $U_B$ 输出的结果分别写入各自的 FIFO；

4）从两个 FIFO 中同时读出水平线起始点；

5）比较两个起始点，较小者作为水平线的左起点，较大者作为水平线的右终点；

6）利用 Tile 的左右边界对水平线进行裁剪；

7）得到的水平线利用循环生成其每一个像素，并计算像素属性。

以上实现步骤可以用图 6 描述。

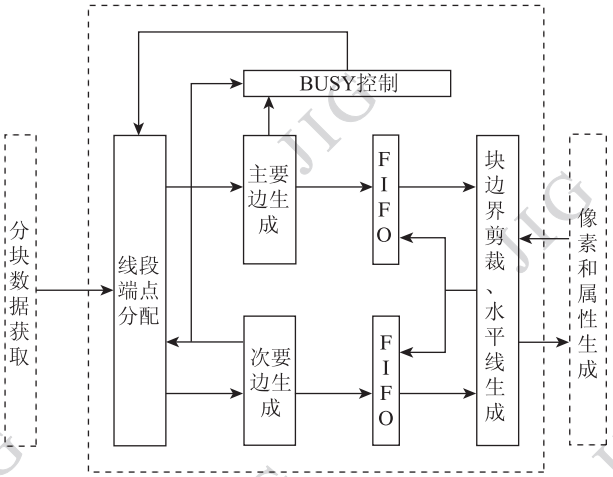



图 6 三角形光栅化实现结构

Fig. 6 Implementation structure of triangle raster

## 2 性能分析

### 2.1 理论性能

光栅化最主要的工作就是产生位于三角形内部的像素，所以平均每个时钟周期能够产生多少个有效的像素输出是最主要的性能指标。

为了不失一般性，设三角形的 3 个顶点坐标从高到低（以 $y$ 坐标降序排列）依次为 $(x_A, y_A)$、$(x_B, y_B)$、$(x_C, y_C)$，高为 $h$（单位：像素），该三角形包含的像素个数 $Pixel_{count}$ 近似为

$$Pixel_{count} = d_{BC} \times h/2 \tag{5}$$

式中，$d_{BC}$ 表示 BC 边的长度。根据 Bresenham 算法，生成 3 条线段的周期数 $C_{AB}$、$C_{AC}$、$C_{BC}$ 分别为

$$\begin{cases} C_{AB} = \begin{cases} |x_A - x_B| & |\Delta x| \geqslant |\Delta y| \\ |y_A - y_B| & |\Delta x| < |\Delta y| \end{cases} \\ C_{AC} = \begin{cases} |x_A - x_C| & |\Delta x| \geqslant |\Delta y| \\ |y_A - y_C| & |\Delta x| < |\Delta y| \end{cases} \\ C_{BC} = \begin{cases} |x_B - x_C| & |\Delta x| \geqslant |\Delta y| \\ |y_B - y_C| & |\Delta x| < |\Delta y| \end{cases} \end{cases} \tag{6}$$

那么生成三角形的扫描线花费的时间 $C_T$ 为

$$C_T = \max\{C_{BC} + C_{AB}, C_{AC}\} \tag{7}$$

之后每个周期产生一个像素，所以产生该三角形所有的像素总共需要花费的周期个数为

$$Cycle = Pixel_{count} + C_T \tag{8}$$

每周期产生的有效像素个数为

$$\frac{Pixel_{\text{count}}}{Cycle} = \frac{d_{BC} \times h}{d_{BC} \times h + 2 \times C_T} = \frac{1}{1 + \dfrac{2 \times C_T}{d_{BC} \times h}} \tag{9}$$

从式(7)(9)可以看出,影响效率的因素为三角形的面积和顶点的坐标差,按照 GPU 通用的评价标准,一般选取三角形的大小为 10×10 像素,不妨取 10×10 的直角三角形,将($C_T = 20, d_{BC} = 10, h = 10$)代入式(9)可以计算出三角形光栅化的效率可以达到 71%。

进一步分析,以上的计算是基于一种极端情况,三角形中有一条水平边和一条垂直边,利用 Bresenham 算法生成第 1 条水平线最长时间为 10(周期个数),如果考虑到产生第 1 条水平线后,FIFO 非空,后面的像素生成模块就可以取出数据处理,那么式(9)中的 $C_T = 10$,可以计算出三角形光栅化的效率为 83%。若第 1 条生成的线段非水平线,产生第 1 条水平线的时间将会变短,也即 $C_T < 10$,那么得到的光栅化效率大于 83%;若第 1 个周期即能生成有效水平线,那么光栅化效率接近 100%。

## 2.2　实际性能

为了测试本文算法的三角形实际处理能力,在 FPGA(field programmable gate array)原型验证系统上实际测试其像素填充率[8]和顶点通过率。

顶点通过率和像素填充率计算方法如下:

设在 $ss$ 内绘制了 $t$ 个三角形,每个三角形的面积为 $area$,由于 1 个三角形有 3 个顶点,则

$$\text{顶点通过率} = t \times 3/s$$

$$\text{像素填充率} = t \times area/s$$

为便于计算,按正三角形进行计算,如图 7 所示。设三角形外接圆的半径为 $r$。则 $\overline{OA} = r$, $\overline{AB} = (\sqrt{3}/2)r$,三角形边长 $\overline{AC} = 2\ \overline{AB} = \sqrt{3}r$。三角形的高 $\overline{BD} = \overline{OB} + \overline{AB} = r + \frac{1}{2}r = \frac{3}{2}r$,于是,三角形的面积为

$$area = \frac{1}{2} \times \overline{AC} \times \overline{BD} = \frac{1}{2} \times \sqrt{3}r \times \frac{3}{2}r = \frac{3\sqrt{3}}{4}r^2$$

因此

$$\text{像素填充率} = t \times area/s = t \times \frac{3\sqrt{3}}{4}r^2 \times \frac{1}{s} \tag{10}$$

为了测试顶点通过率,将三角形面积设为很小,使像素的渲染不占处理时间,所有的时间都用来处理顶点(即三角形面积最小时得到的顶点通过率为准确值);反之,将三角形面积设为很大,则顶点通过基本不占时间,所有时间都用来生成像素(即三角形面积最大时得到的像素填充率为准确值),是用来测试像素填充率。

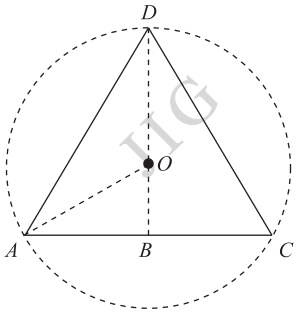



图 7　三角形及其外接圆

Fig. 7　Triangle and it's circumcircle

利用 C 语言编写测试统计程序,采用本文算法实现三角形渲染的 FPGA 原型验证系统(主频 100 MHz)与 ATI(array technology industry)M9(主频 200 MHz)、ATI M72(主频 450 MHz)进行比较。比较结果如表 1 和图 8 所示。

**表 1　实测性能比较**

**Table 1　The experimental performance comparison**

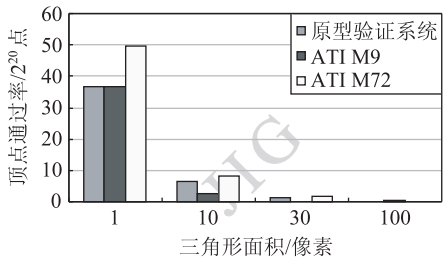
| 三角形面积/像素 | 测试项 | 原型验证系统 | ATI M9 | ATI M72 |
|---|---|---|---|---|
| 1 | Vts/($2^{20}$点/s) | 36.885 | 36.585 | 49.606 |
| | Pixel/($2^{20}$像素/s) | 0 | 0 | 0 |
| 10 | Vts/($2^{20}$点/s) | 6.270 | 2.727 | 8.280 |
| | Pixel/($2^{20}$像素/s) | 203.634 | 88.568 | 268.893 |
| 30 | Vts/($2^{20}$点/s) | 1.080 | 0.032 | 1.770 |
| | Pixel/($2^{20}$像素/s) | 315.657 | 104.384 | 517.470 |
| 100 | Vts/($2^{20}$点/s) | 0.116 | 0.351 | 0.210 |
| | Pixel/($2^{20}$像素/s) | 377.129 | 102.653 | 681.975 |

1)测试项中 Vts 代表顶点通过率;Pixel 代表像素填充率。

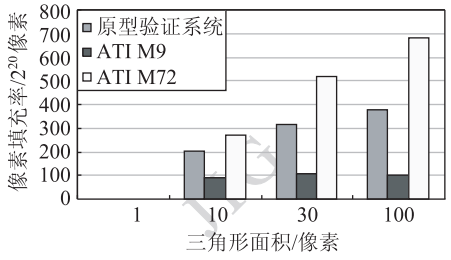
2)测试环境说明。原型验证系统(包含 4 条光栅流水线)和 ATI M9 采用的 CPU 均为 MPC8270;ATI M72 采用的 CPU 为 MPC8640,操作系统均为 VxWorks。

3)三角形大小为 1 时,Pixel 指标为 0,是因为测试程序在此情况下为了使得像素生成模块无任何压力而集中于顶点变换,没有渲染三角形的点。

从上述的测试结果可以得出,本文算法在像素




(a) 顶点通过率比较




(b) 像素填充率比较

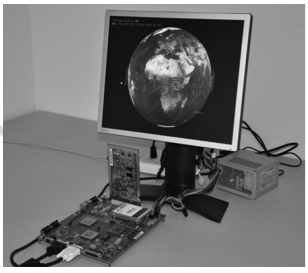
图 8　顶点通过率及像素填充率比较

Fig. 8　Comparison of vertex pass rate and pixel fill rate

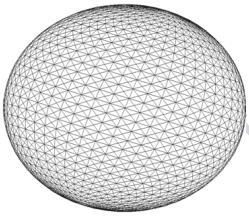
生成率上指标上,可以达到与频率是其两倍的 ATI M9 相当的水平。

## 3　算法验证

将该算法和实现结构采用 Verilog 语言描述出来,并在 FPGA 原型验证系统(如图 9 所示)上对该结构进行了验证,验证环境为:CPU 采用 Freescale PowerPC 8270;操作系统采用 VxWorks 6.0;FPGA 芯片采用 Altera 公司的 Stratix III 系列 EP3SL340F1517I4;内核频率为 100 MHz,DDR 控制器频率为 133 MHz。



(a) 原型验证系统实物图



(b) 组成旋转地球的三角形网格

图 9　FPGA 原型验证系统

Fig. 9　The FPGA prototype

为了验证该算法在实际系统中光栅化的效率,在原型验证板上运行一个由 3 072 个三角形组成的地球,同时经过了深度测试和纹理贴图。系统在实现时采用了四条流水线,运行的旋转地球在 1 024 × 768 分辨率下最快运行速度可以达到 80 帧/s。

## 4　结　论

TBR 的架构可以在有限的存储器带宽下获得较好的性能,它采用片上存储器来大幅减少像素、深度等信息的频繁访问外部存储器。为了配合该架构,进一步提升像素填充率,提出了一种基于边界扫描的三角形光栅化算法,在三角形分块之后通过计算三角形与块边界的位置关系确定扫描边界,在光栅化过程中避免了冗余像素的产生,提高了光栅化效率。经过理论分析,该算法的光栅化效率可以达到接近 100% 的效率。在 FPGA 原型系统上进行了实际测试,其像素生成率能够达到频率比其高一倍的 ATI M9 的水平。

### 参考文献(References)

[1] Juurlink B H H, Antochi I, Crisu D, et al. GRAAL: a framework for low-power 3D graphics accelerators[C]// Proceedings of IEEE Computer Graphics and Applications. Washington, USA: IEEE Computer Society, 2008: 63-73.

[2] Antochi I. Suitability of tile-based rendering for low-power 3D graphics accelerators[D]. Holland: Delft University of Technology, 2007.

[3] Crisu D. Hardware algorithms for tile-based Real-Time rendering [D]. Holland: Delft University of Technology, 2012.

[4] Antochi I, Juurlink B, Vassiliadis S, et al. Scene management models and overlap tests for tile-based rending[C] // The EUROMICRO Symposium on Digital System Design. Rennes, FRANCE: IEEE Computer Society, 2004: 424-431.

[5] Crisu D, Cotofana S D, Vassiliadis S, et al. 3D graphics tile-based systolic scan-conversion[C] // Proceedings of The 38th Asilomar Conference on Signals, Systems and Computers. Pacific Grove, California, USA: IEEE Computer Society, 2004: 517-521.

[6] Chen Y Y, Zhang R Z, Li J H. Computer Graphics Practical Technology [M]. 3rd ed. Beijing: Tsinghua University Press, 2012. [陈元琰,张睿哲,李建华. 计算机图形学实用技术[M]. 3 版. 北京:清华大学出版社,2012.]

[7] The MESA 3D Graphics Library[EB/OL]. [2014-10-19]. http://www.mesa3d.org.

[8] Akenine-Möller T, Haines E, et al. Real-Time Rendering [M]. 5th ed. Boca Raton, FL, USA: A K Peters/CRC Press, 2008: 702-702.