基于查找表的像素处理器新算法*

郝建新 谢剑斌 蔡宣平 孙茂印

(国防科技大学电子技术系 长沙 410073)

摘 要 本文在基本的像素处理算法的基础上, 提出了一种基于查找表的快速平滑插值算法. 该算法不仅运算量小、精度较高, 而且易于硬件实现, 适合于高速显示处理系统中.

关键词 像素处理, 查找表, 高速显示处理

分类号 TN 941. 1

The New Algorithm of Pixel-Processing Based on Look-up Table

Hao Jianxin Xie Jianbin Cai Xuanping Sun Maoyin

(Department of Electronic Technology, NUDT, Changsha, 410073)

Abstract On the basis of fundamental pixel-processing algorithm, this paper presents a fast smooth interpolation algorithm based on look-up table. The algorithm is of not only higher precision with fewer operation, but also apt to be implemented by means of hardware, so it is suitable for application to be applied in high-speed graphics processing system.

Key words pixel-processing, look-up table, high-speed display system

在三维图形处理系统中, 快速平滑的像素处理是真实感图形绘制技术和高速显示技术的一个关键课题[1].

真实感图形绘制过程中, 逼真度和显示速度是互相矛盾的, 真实感图形绘制要求采用较复杂的像素处理算法; 而高速显示要求采用较简单的像素处理算法[2]. 在虚拟现实等高速图形处理系统中, 为了实现高速显示真实感图形, 作者采用了一种基于查找表的快速平滑像素处理算法.

1 像素处理算法的基本原理

在进行像素处理的过程中, 涉及到光照明模型、明暗处理方法等概念.

1 1 光照明模型

为了计算屏幕上相应景物可见点的颜色, 需建立一个能计算物体表面在空间给定方向上光亮度的光照明模型. 实用的 Phong光照明模型可表示如下:

$$I = K_a \cdot I_{pa} + \sum [K_d \cdot I_{pd} \cdot \cos i + K_s \cdot I_{ps} \cdot \cos^n \theta]$$

1 2 明暗处理方法

传统的明暗处理有两类方法: Gouraud算法和 Phong算法.

Gouraud算法是对多边形所有顶点离散的光亮度采样作双线性插值以获得连续的光亮度函数. 而 Phong算法是对离散的法向量采样作双线性插值来构造连续的法向量函数, 并将它代入光亮度计算公式, 即得到光亮度插值公式.

Gouraud算法计算量很小, 处理速度较快, 广泛用于实时图形生成[3].

* 1998年 5月 12日收稿
国防科技预研基金项目
第一作者: 郝建新, 男, 1956年, 副教授

1 3 像素处理算法

基于三角形面片的高速图形处理系统的流程如图 1所示。

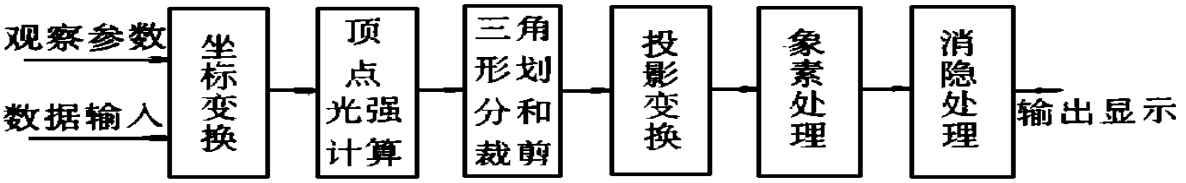

图 1 基于三角形面片的高速图形处理系统的流程

从图 1可以看出，像素处理部分完成如下的功能：在屏坐标系下，由三角形三顶点的信息 (x, y, z, c) 计算出三角形内部所有点的信息 (x, y, z, c)。

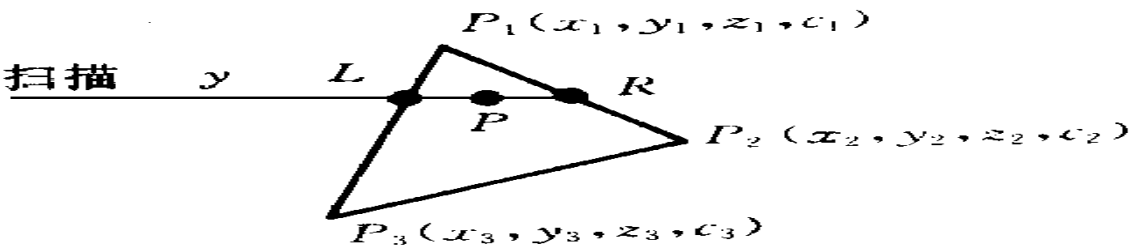

图 2 三角形像素插值模型

基本的像素处理算法为：如图 2所示，已知三角形的三个顶点 P_1、P_2、P_3，每个顶点有坐标 (x, y, z) 和灰度信息 c4个参数。由双线性插值算法可求得 P 点的坐标，即 L 由 P_1和 P_3线性插值求得，R 由 P_1和 P_2线性插值求得，P 由 L 和 R 线性插值求得。其实现流程如下所示：

step1: 三角形顶点信息的输入；

step2: 求 P_1P_3、P_1P_2、P_2P_3在 x, z, c方向上关于 y 的变化率；

step3: 求 L、R 在 z, c方向上关于 x 的变化率；

step4: 计算 LR 端点像素的信息；

step5: 计算 LR 扫描线上像素的信息；

step6: 三角形内所有像素信息的输出。

2 基于查找表的快速平滑像素处理算法

2 1 算法概述

综合考虑图形显示的速度和逼真两方面因素，故该算法的光照明采用 Phong模型，明暗处理采用 Gouraud算法，选择三角形面片作为像素处理模型，算法流程采用流水线结构，倒数值通过查找表获得，加/减法、比较运算和乘法等采用并行硬件来实现。对每个三角形有：

- 只需耗时较少的加法、减法、比较和乘法运算；
- 没有耗时较多的除法或求倒数运算；
- 利用查找表法可以保证灰度的高精度；
- 不会重复扫描任何像素。

2 1 算法的数学原理

如图 2，由 P_1、P_2、P_3求 L (L_x, L_y, L_z, L_c) 和 R (R_x, R_y, R_z, R_c) 的递推公式为：

$$L_{xn+1} = L_{xn} + \mathrm{d}x_L; \quad L_{zn+1} = L_{zn} + \mathrm{d}z_L; \quad L_{cn+1} = L_{cn} + \mathrm{d}c_L \tag{1}$$

$$R_{xn+1} = R_{xn} + \mathrm{d}x_R; \quad R_{zn+1} = R_{zn} + \mathrm{d}z_R; \quad R_{cn+1} = R_{cn} + \mathrm{d}c_R \tag{2}$$

$$L_{yn+1} = L_{yn} + 1; \quad R_{yn+1} = R_{yn} + 1 \tag{3}$$

其中: $dx_L = (x_3 - x_1)^* (1/(y_3 - y_1))$; $dx_{R\_1} = (x_2 - x_1)^* (1/(y_2 - y_1))$;

$dx_{R\_2} = (x_3 - x_2)^* (1/(y_3 - y_2))$ (4)

$dz_L = (z_3 - z_1)^* (1/(y_3 - y_1))$; $dz_{R\_1} = (z_2 - z_1)^* (1/(y_2 - y_1))$;

$dz_{R\_2} = (z_3 - z_2)^* (1/(y_3 - y_2))$ (5)

$dc_L = (c_3 - c_1)^* (1/(y_3 - y_1))$; $dc_{R\_1} = (c_2 - c_1)^* (1/(y_2 - y_1))$;

$dc_{R\_2} = (c_3 - c_2)^* (1/(y_3 - y_2))$ (6)

要注意的是: dx_R、dz_R、dc_R 各有两种情况。

求 P 的递推公式为

$$P_{xm+1} = P_{xm} + 1$$

$$P_{zm+1} = P_{zm} + dz$$

$$P_{cm+1} = P_{cm} + dc \quad (7)$$

其中: $dz = (R_z - L_z)^* (1/(R_x - L_x))$

$= (dz_R - dz_L)^* (1/(dx_R - dx_L))$ (8)

$dc = (R_c - L_c)^* (1/(R_x - L_x))$

$= (dc_R - dc_L)^* (1/(dx_R - dx_L))$ (9)

即在同一个三角形面片内, dz、dc 对所有扫描线都是一致的。这样, 每个三角形的像素处理需要 4次求倒数、11次乘法和 $(15 + 12n + 2n)$ 次加法, 其中 n 和 m 分别为三角形纵向和横向的平均像素个数。假设屏幕分辨率为 512× 512, 数据点阵取的是 128× 128, 故 $n = m = 4$。这样, 每个三角形的计算量包括 4次求倒数、11次乘法和 71次加法, 约需 338个操作, 整个 128× 128数据点阵共有 32768个三角形, 共需 11075584次操作。若要求在 40ms内完成, 处理速度需达到每秒 276, 889, 600次操作, 约 280MPPS。

2 3 算法的流程

该算法的处理过程可以分为 5个功能块, 其数据流是单向的; 同时 5个功能块又相对独立, 于是可以采用流水线操作, 并且整个流程由 5级流水组成。

step1: 数据输入接口;

step2: 求 dx、dz、dc;

step3: 求 dz、dc;

step4: 计算 LR 端点、计算 LR 扫描线;

step5: 数据输出接口。

2 4 实现算法的硬件结构

下面对算法流程的各级流水结构和功能作简要介绍。

(1) 第一级流水

数据输入接口是像素处理器和主机的数据接口, 负责接收主机送来的三角形顶点信息 $P_1(x_1, y_1, z_1, c_1)$、$P_2(x_2, y_2, z_2, c_2)$、$P_3(x_3, y_3, z_3, c_3)$。其中 P_1、P_2、P_3 是按 y 值从小到大排列的, 而且这级流水需要 7clk (即 1× 6+ 1 clk)。

(2) 第二级流水

计算三角形三边的 x、z、c 关于 y 的变化率 dx、dz、dc。对于一般三角形 $P_1P_2P_3$, dx、dz、dc 的计算如式 (4) (5) (6) 所示。对于特殊三角形, 若 $y_1 = y_3$ 时, 则取 $dx_2 = 0$ $dz_2 = 0$ $dc_2 = 0$; 若 $y_2 = y_3$ 时, 则取 $dx_3 = 0$ $dz_3 = 0$ $dc_3 = 0$, 即计算过程中不用到这些值。这级流水需要 7clk (即 1+ 2+ 4 clk)。

(3) 第三级流水

计算 dz、dc, 即扫描线 LR 在 z、c 方向上关于 x 的变化率, 这级流水需要 7clk (即 1+ 2+ 4 clk)。

(4) 第四级流水

第四级流水包括求 LR 点和扫描线两部分。这级流水需要 1+ n clk。

求 LR 点完成扫描线端点信息 (x, y, z, c) 的计算，其起点为点 P_1，终点为点 P_3。对线 P_1P_3 有：在 y 方向上的步进量为单位 1，在 x 方向上的步进量为 dx_L，在 z 方向上的步进量为 dz_L，在 c 方向上的步进量为 $d\alpha$。对线 P_1P_2 有：在 y 方向上的步进量为单位 1，在 x 方向上的步进量为 $dx_R - 1$，在 z 方向上的步进量为 $dz_R - 1$，在 c 方向上的步进量为 $d\alpha_R - 1$。对线 P_2P_3 有：在 y 方向上的步进量为单位 1，在 x 方向上的步进量为 $dx_R - 2$，在 z 方向上的步进量为 $dz_R - 2$，在 c 方向上的步进量为 $d\alpha_R - 2$。求 R 的过程中，x、z、c 方向上的步进量在点 P_2 要变化：在 $y_R < y_2$ 时，采用线 P_1P_2 方向的步进量；在 $y_R \geqslant y_2$ 时，采用线 P_2P_3 方向的步进量。

求 LR 扫描线完成线 LR 上所有点的 z、c 计算，其起点的选择是以 z 值为标准来考虑的，之所以这样选择，是因为考虑到 z 的计算涉及到 24位数 (16位整数和 8位小数) 的累加，而 x 的计算仅涉及到 9位数的累加，并且步进量为单位+ 1或 - 1，也就是说，x 的计算仅涉及到 9位数的可控加减法，判断加还是减需要时间；c 的计算仅涉及到 16位数 (8位整数和 8位小数) 的累加。同时，其起点选择为 L、R 两点中 z 值较小的一个，这样可使 z 的计算由 24位加法器来完成，否则 z 的计算要由 24位减法器来实现，显然后者比前者不仅硬件开销大，而且速度要慢些。按照作者的设计，计算 z 和 x、c 的时间便可以匹配，于是为提高模块的运算速度打下了基础。

(5) 第五级流水

数据输出接口负责像素处理器计算所得的像素数据送出，因为现有存储器速度上的限制，并考虑到性能价格比因素，这一部分需要采用特殊设计的存储器控制系统 (多体并行存储系统)。这级流水需要 7clk (即 1+ 2+ 4 clk)。

2.6 算法的硬件实现

采用 FPGA 技术和 EPLD 技术来实现该算法。FPGA (现场可编程逻辑门阵列) 和 CPLD (复杂可编程逻辑器件) 各有优缺点[4]：

(1) FPGA 和 CPLD 在资源方面比传统器件和 PLD 都要丰富[5]；

(2) FPGA 基于 RAM 工艺，不要求对 FPGA 母片编程，其编程数据写到 EPROM 中，上电时通过引导读入 FPGA 中的 RAM，从而使 FPGA 具有所需要的构造。这个过程需要几十到几百毫秒。而 CPLD 基于 EPROM 或 E^2PROM 工艺，无上电引导过程，需要对芯片本身编程[6]；

(3) FPGA 适合于寄存器密集型设计，但延时很难估计，速度较慢。而 CPLD 的延时固定，速度较快。

实现该算法的硬件原理框图如图 3所示。

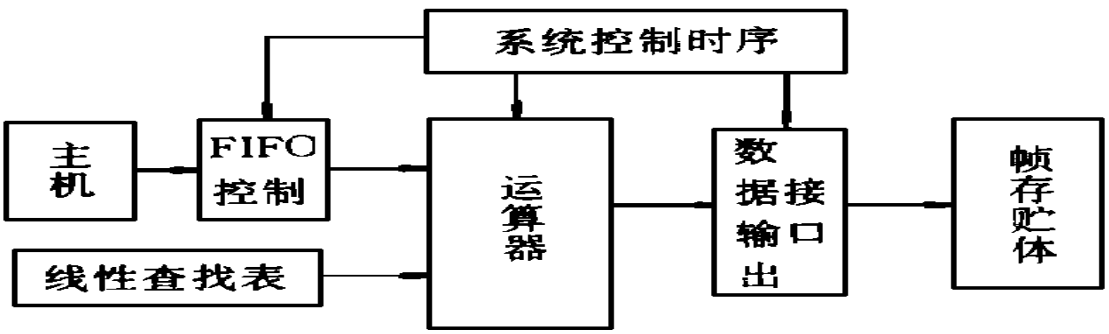

图 3 基于查找表的像素处理器硬件原理框图

最后，运算器、数据输出接口部分采用一片 FPGA 器件来实现，FIFO 控制、系统控制时序部分采用一片 CPLD 器件来实现[7]。

2.7 查找表的设计

在分辨率为 512× 512、256级灰度、2^{16}级深度信息的高分辨率灰度图形显示系统中，要求坐标 x

y的位数各取 9位，深度信息 z的位数取为 16位，灰度信息 c的位数取为 8位。此时，查找表的设计有如下特点:

(1) 采用存储倒数型查找表，即插值变化率为分母的倒数值和分子的乘积，其中分母的倒数值由查找表获得。

(2) 仅仅存储分母的倒数值，可大大减少对存储容量的要求。所需的存储空间可估算如下: 1/(D1ty) 所对应的查找表存储容量约为 512W (2^9字)，1/(D1tx) 所对应的查找表存储容量约为 512W (2^9字);

(3) 考虑到查找表存储容量很小，所以共安排多个线性查找表，以字为单位进行寻址。这样，如图 4所示，像素处理器需要约 2KW 的查找表存贮空间。

3 结束语

经过在微机上仿真运算，发现该算法不仅运算量较小、精度较高，而且速度很快，若像素处理器的系统时钟取为 20MHZ以上，就可达到实时显示的效果，故本算法非常适合应用于三维实时图形处理系统中。

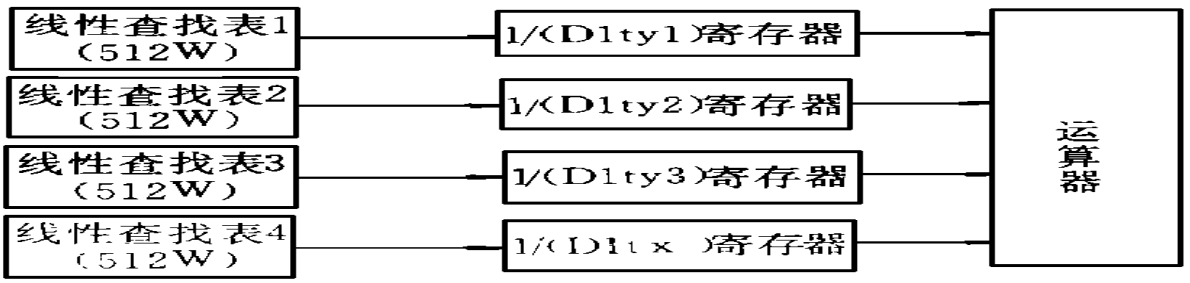

图 4 线性查找表的布置

基中，D1ty1表示 ($y_{p1} - y_{p2}$)，D1ty2表示 ($y_{p1} - y_{p3}$)，D1ty3表示 ($y_{p2} - y_{p3}$)，D1tx表示 ($x_p - x_L$)。

参考文献

1 陈振初，蔡宣平. 计算机图形显示原理 (软件). 国防科技大学出版社，1991

2 唐荣锡等. 计算机图形学教程. 北京：科学出版社，1994

3 石教英等. 高度真实感三维图形的生成. 第一届计算机图形学及其应用会议，1985

4 DATA BOOK，(XILINX 公司). 1995

5 DATA BOOK，(XILINX 公司). 1996

6 DATA BOOK，(ATMEL公司). 1995

7 郝建新，谢剑斌. 几个有关 FPGA 逻辑设计问题的浅析. 微型计算机，1997，3