一种基于整数坐标的亚像素精度区域采样反走样算法

徐小良 洪 波
(杭州电子科技大学计算机学院,杭州 310018)

摘 要 为了快速有效地消除图形显示中的边界锯齿,提出了一种基于整数坐标的亚像素精度区域采样反走样算法。该算法将坐标点的亚像素部分保存于整数的低位,整数部分存于高位,根据亚像素部分与整数部分的差值,利用位运算快速计算像素的覆盖面积,像素亮度等级可达到 2^n(n 为亚像素位数,通常情况 n 取 8)。与基于中点算法的区域采样算法相比,图像精度高,计算速度快;与过采样算法(采用 3×3 或 4×4 的过采样网格)相比,采样精度高,计算量不会随着采样精度的提高而变大,也不存在图像重建过程中由高分辨率数据向低分辨率转变时的信息丢失问题。实验结果表明,新算法生成的图像比较细腻,实时性比较高,优于一般的反走样算法。

关键词 反走样 过采样 区域采样 整数坐标 亚像素
中图法分类号: TP391.4 **文献标识码**: A **文章编号**: 1006-8961(2009)12-2438-05

A Sub-pixel Regional Sampling Anti-aliasing Algorithm Based on Integer Coordinate

XU Xiao-liang, HONG Bo
(*Computer College of Hangzhou Dianzi University*, *Hangzhou* 310018)

Abstract In order to eliminate the graphic aliasing more quickly and efficiently, we propose a sub-pixel level precision regional sampling anti-aliasing algorithm based on integer coordinate, putting sub-pixel information on the low bits and the integral parts on the high bits of plastic number. Calculating the cover area of each pixel with bit operation according to the difference between the sub-pixel part and the integer part. The Brightness level can reach 2^n (n is the length of sub-pixel, most of the time it is 8). Compared with mid-point based regional sample algorithm, the picture generated is more precise and fast. Compared with over sampling method(most of the time use 3×3 or 4×4 sample grid), it's more precise, when the sample bits arise, the calculation will be the same, and doesn't exist information losing problems when the data is conversed from high resolution to low resolution witch results aliasing. The result shows our algorithm can quickly generate high quality images better than general antialiasing algorithm.

Keywords anti-aliasing, over-sampling, regional sampling, integer coordinate, sub-pixel

1 引 言

将图形显示到光栅设备过程中,需要将物体的坐标点数字化为离散的整数像素坐标,由于显示设备的不连续性,在图形的边界会出现锯齿。解决走样的方法分为两类,一类是过采样,即将每个像素点看成更多的点,在高分辨率下计算,然后根据某种平均算法将采样信息平均到一个像素[1-4];另一类是区域采样,即将一个像素看成一块区域[5-9],像素的

基金项目:浙江省科技计划重点科研项目(2008C21082)
收稿日期:2008-12-15;**改回日期**:2009-03-02

第一作者简介:徐小良(1976~),男。副教授。2004 年于浙江大学获电子信息技术及仪器专业博士学位。主要研究领域为嵌入式系统、中间件技术、图形图像等。E-mail:xxl@hdu.edu.cn

光强与多边形占此块区域的面积成正比。

为了提高图形显示效果,国内外对反走样算法做了很多研究。文献[1]解决了过采样算法以像素中心为分界点造成的色彩激变问题,但图像精度有限;文献[2]提出了一种过采样网格的压缩算法,减少了采样频率增加引起的内存开销;文献[3]、[4]在过采样反走样处理中引入了信号处理领域的滤波方法,提出了两种滤波算法解决高频过采样数据降频时产生的噪声问题;文献[5]、[6]提出两种前向滤波(即区域采样)算法,一定程度提高了反走样的效果,但同时增加了处理的内存开销;文献[7]、[8]分别提出了两种针对特定应用的前向滤波算法,即针对对称直线的圆锥滤波算法,比传统圆锥滤波算法速度提高了一倍,但只能保证在较小的斜率下保持直线的平滑性;采用矩形滤波器的任意宽度直线的绘制方法,解决仪表表盘的绘制走样,但是这两种滤波选择通用性差。

针对上述过采样算法精度不高,采用滤波器的反走样算法存在滤波器选择通用性差且图像精度增加时处理内存开销增大等问题,研究提出一种基于整数坐标的高精度区域采样算法,利用位运算快速准确地计算出亚像素部分与整数部分的差值即像素的覆盖面积。

2　整数坐标区域采样算法描述

本算法将坐标值的小数部分存放于 32 位整数的低 n 位,整数部分存放于高 $32-n$ 位。其中 n 为亚像素位数,它决定了生成图像的细腻程度,像素点的亮度等级为 2^n。n 可以取 0 ~ 32 之间的任意值,本文默认为 8(可以得到 256 个亮度等级),相比于一般过采样算法的 16 个亮度等级已经足够精细,因截断小数部分而丢失的信息可以忽略不计。

对于二进制存储的数据,本算法直接将其小数部分读入整数的低位,整数部分存放于高位。对于以浮点数格式存储的坐标,需将浮点数乘以 2^n 进行放大,将整数部分结果截断后保存于 32 位整数内。

2.1　提取亚像素坐标

设 x 为存放坐标值的变量,x 的前 24 位表示坐标点的整数部分 $E(x)$,后 8 位表示坐标点放大后的小数部分即亚像素部分 $F(x)$,通过位运算提取坐标值的整数部分与亚像素部分如下:

$$\begin{aligned} E(x) &= x >> pixel\_shift \\ F(x) &= x\&pixel\_shift \\ pixel\_scale &= pixel\_shift << 1 \\ pixel\_mask &= pixel\_scale - 1 \end{aligned} \tag{1}$$

式中,$pixel\_shift = 8$。

将线段端点坐标(x_n, y_n) $(n = 1, 2)$代入式(1)得到坐标的整数部分(ex_n, ey_n)与亚像素部分(fx_n, fy_n)。

2.2　计算直线经过像素点的覆盖值

根据直线两个端点的坐标值,可分为 4 种情况:(1)两个端点的纵坐标整数值相同,处于水平线上。(2)两个端点的横坐标相同,处于垂直线上。(3)两点落在同一像素内。(4)两个端点的横、纵坐标值均不相同,为斜线段。

下面根据以上 4 种情况分别计算直线经过像素点的覆盖值:

(1)两点处于水平线

只需对处于同一水平线的一段线段解决走样问题,本文调用水平线光栅化子程序(见第 2.3 节)进行反走样处理。

(2)两点处于垂直线

为了更好地进行算法描述,采用 3 × 3 过采样网格来近似表示每个像素点,如图 1 所示,$p1$,$p2$ 点的 x 值相同,即整数部分与小数部分均相同,只需要渲染一段垂直线段。对于两点 x 值的整数部分相等,亚像素部分不相等的情况作为斜线段处理,在(4)中进行描述。

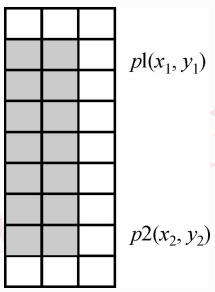

图 1　垂直线段的光栅化

Fig. 1　Rasterizer of vertical line segment

像素点的细分大小(即亚像素部分)为 $pixel\_scale$,其值为 2^n(n 为亚像素位数),第 1 个像素点的覆盖值计算方法为

$$\begin{aligned} width &= (x_1 - (ex << pixel\_shift)) << 1 \\ delta &= pixel\_scale - fy_1 \\ ared_{\text{first}} &= delta \times width \end{aligned} \tag{2}$$

式中，*delta* 表示第 1 个像素在 *y* 方向距离下一个整数坐标的距离。因为线的两边覆盖面积相等，取线段 *x* 方向宽度的两倍表示宽度值（*width*），两者相乘得到第 1 个像素的覆盖面积。图 1 中的灰色部分表示线段在每一像素的细分网格的覆盖部分，粗黑线表示直线段。

第 2 个像素至倒数第 2 个像素的 *y* 分量均为 *pixel_scale*。最后一个像素的 *y* 分量为线段终点的亚像素分量，两类像素点的覆盖值计算方法如下：

$$\begin{aligned} area_{\text{middle}} &= pixel\_scale \times width \\ area_{\text{last}} &= fx_2 \times width \end{aligned} \tag{3}$$

（3）两点位于一个像素点内

如果线段整数坐标相同，此时只需要计算一个点的覆盖值，计算方法如下：

$$\begin{aligned} width &= (pixel\_scale << 1 - fx_1 - fx_2) \\ area_{\text{point}} &= (fy_2 - fy_1) \times width \end{aligned} \tag{4}$$

式中，*width* 表示两个端点的 *x* 值距离下边界的长度之和，$area_{\text{point}}$ 等于直线与边界构成的梯形面积的 2 倍。图 2 中的灰色区域为填充面积的一半。

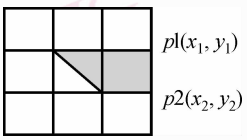

图 2 像素点内的光栅化

Fig. 2 Rasterizer inside a pixel point

（4）两点连线为斜线段

当两端点的 *x*，*y* 值均不相等时，渲染斜线段，将其作为一系列的水平线处理，分为 3 种情况进行分别处理，即第 1 行、最后一行和中间线段。

起始点光栅化，计算出第 1 个像素点的宽度值，同时计算出第 1 个整数坐标值。计算公式为

$$\begin{aligned} x_{\text{first}} &= x_1 + (pixel\_scale - fy_1) \times dx/dy \\ ey &= ey + 1 \\ x_1 &= x_{\text{first}} \end{aligned} \tag{5}$$

式中，*dx*，*dy* 表示直线在 *x*、*y* 方向上的距离差。

x_{first} 表示第 1 个整数坐标的值，调用水平线光栅化子程序 $f(ey, x_1, fy_1, x\_first, pixel\_scale)$。*ey* 指 *y* 的整数坐标，$fy_1$，$fy_2$ 分别为两点在 *y* 方向的亚像素精度，x_1 x_2 为水平方向的坐标值，最后设置 x_1 的值为 x_{first}。

对第 2 行到最后一行进行循环处理，x_{middle} 为每次 *x* 增加水平分量后的坐标值，*y* 每次增加 1，调用水平线渲染子程序 $f(ey, x_1, 0, x_{\text{middle}}, pixel\_scale)$，并将 x_1 的值设置为 x_{middle}，公式如下：

$$\begin{aligned} x_{\text{middle}} &= x_1 + pixel\_scale \times dx/dy \\ ey &= ey + 1 \\ x_1 &= x_{\text{middle}} \end{aligned} \tag{6}$$

最后调用 $f(ey, x_1, 0, x_2, fy_2)$ 函数，渲染结束。

2.3 水平线光栅化

水平线光栅化子程序计算直线在水平线上的某一线段内每个像素的覆盖值，涉及的 5 个参数分别是 *ey*，x_1，fy_1，x_2，fy_2。水平线渲染分为 3 种情况：（1）落在同一点，（2）两点位于一个像素，（3）在一系列水平像素内。图 3 中给出了基于 3×3 过采样网格的水平线光栅化示意图，渲染面积为灰色部分的 2 倍。

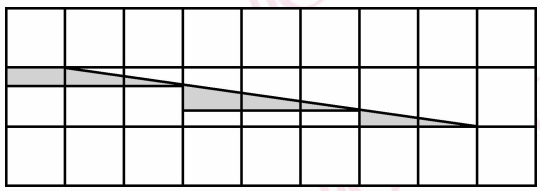

图 3 水平线光栅化示意图

Fig. 3 The schematic of a horizontal line's rasterization

当 $fy_1 = fy_2$ 时，线段两端点在同一点，保存 ex_2，*ey* 为当前点，无需计算覆盖值。

当 $ex_1 = ex_2$ 时，目标区落于同一像素内，像素覆盖值为 *y* 方向的差值与水平方向的边形成的梯形面积的 2 倍。计算公式如下：

$$area_{\text{point}} = (fy_2 - fy_1) \times (fx_1 + fx_2) \tag{7}$$

当 $fy_1 != fy_2$ && $ex_1 != ex_2$ 时，第 1 个像素点的 *y* 分量差值为 *delta*，覆盖面积为图中梯形面积的 2 倍，每次计算完当前点的覆盖值之后，更新 ex_1，fy_1，计算公式为

$$\begin{aligned} delta &= (pixel\_scale - fx_1) \times (fy_2 - fy_1)/dx \\ area_{\text{first}} &= delta \times (pixel\_scale + fx_1) \\ fy_1 &+= delta, ex_1 = ex + 1 \end{aligned} \tag{8}$$

处理第 2 个像素点到倒数第 2 像素点，fy_1 每次累加 *delta*，面积每次累加 $(fx_1 + first) \times delta$。$ex_1$ 每次递增 1，循环处理。每个像素点的覆盖值计算好之后，更新 ex_1，fy_1，计算公式如下：

$$\begin{aligned} delta &= pixel\_scale \times (fy_2 - fy_1)/dx \\ area_{\text{middle}} &= delta \times pixel\_scale \\ fy_1 &+= delta \\ ex_1 &= ex + 1 \end{aligned} \tag{9}$$

最后一个像素点的覆盖值为

$$area_{last} = (y_2 - y_1)fx_2 \tag{10}$$

2.4　像素覆盖面积的标准化

像素覆盖面积的标准化是指将像素的覆盖值转化为覆盖区域与未覆盖区域的百分比,因为步骤 1 到 3 中对每个像素点的覆盖面积放大了 2 倍,所以需要除以 2,否则会有部分像素点的覆盖值百分比大于 1。

3　算法性能分析

通过上述算法的描述与分析可见,对一条线段进行反走样处理,首先需要预处理,分离点的整数部分与亚像素(小数)部分,相对于一条直线上的所有像素来说,预处理所做的计算量显得微不足道。对一条含有为 n 个像素的线段进行反走样处理,新算法的运算次数为 $2n \sim 3n$,算法复杂度为 $O(n)$。与文献[9]提出的基于中点算法的区域采样算法计算量相当,但精度更高;与基于滤波算法(文献[3]、[4])改进的过采样算法以及基于前向滤波的区域采样算法(文献[5] ~ [8])相比,本算法不需要滤波方法中的查找表,内存开销低。

同时基于整数坐标的反走样算法对不同精度的反走样计算量是一样的。对于不同的显示设备,只需要适当调整反走样的参数,就能适应不同场合的不同需要,具有较好的通用性。

4　实验结果对比

由于亚像素精度的反走样算法能实现 256 (2^8) 级的图像精度,因此图像质量要比一般反走样算法精细很多。常规过采样算法大部分采用 9 个网格的细分,只能产生十个等级的亮度值,文献[1]提出的改进的过采样算法,也只能将亮度等级加倍,图像质量仍然不够精细,根据奈奎斯特理论,过采样算法得到的高分辨率数据向低分辨率转换时,会造成一定的图形信息丢失,采用不同的滤波器进行滤波试验结果相差很大。文献[9]提出的基于中点算法的区域采样算法,将决策参数转换为覆盖率,因为中点算法的不精确性,该类区域采样算法不能获得较高的精度值。

图 4 ~ 图 7 分别是未经反走样、两种常规反走样及亚像素精度的阿基米德螺旋线;图 8、图 9 分别是常规反走样及亚像素精度反走样的宽度一组一到十像素宽度的直线图。可以看出亚像素精度的反走样效果明显要细腻很多。

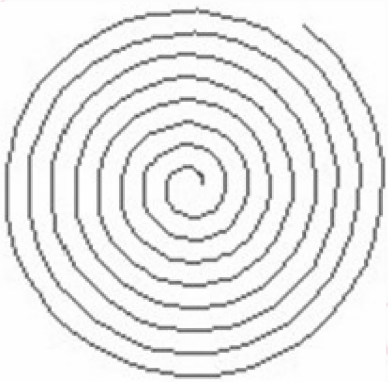

图 4　未经反走样处理的阿基米德螺旋线

Fig. 4　Archimedes spiral processed without anti-aliasing

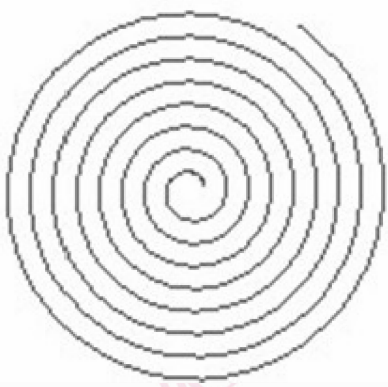

图 5　基于中点算法的区域采样反走样阿基米德螺旋线

Fig. 5　Archimedes spiral processed by mid-point anti-aliasing algorithm

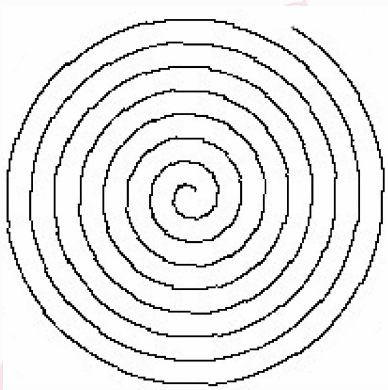

图 6　基于 4 × 4 过采样的反走样阿基米德螺旋线

Fig. 6　Archimedes spiral processed by 4 × 4 over-sampling anti-aliasing

图 7 基于整数坐标亚像素精度反走样阿基米德螺旋线

Fig. 7 Archimedes spiral processed by sub-pixel anti-aliasing algorithm based on integer coordinate

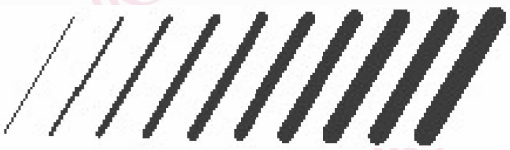

图 8 一般反走样直线图

Fig. 8 Conventional anti-aliasing Line

图 9 亚像素精度的反走样直线

Fig. 9 Sub-pixel precision anti-aliasing line

5 结 论

提出一种基于整数坐标的亚像素精度区域采样反走样算法,该算法将坐标点的亚像素部分保存于整数的低 n 位,整数部分存于高位,像素亮度等级可达到 2^n(一般 $n=8$)。实验结果表明,相比于常规的反走样算法,新算法图像精度高,运算速度快,不存在图像从高分辨率向低分辨率转换时导致的信息丢失问题,内存开销也不会随着采样精度的提高而增加。

参考文献(References)

1 Schilling A. A new simple and efficient antialiasing with subpixel masks[A]. In: Proceedings of SIGGRAPH Computer Graphics[C], New York, NY, USA, 1991, **25**(4):133-141.

2 Beaudoin P, Poulin P. Compressed multisampling for efficient hardware edge antialiasing [A]. In: Proceedings of Graphics Interface[C], London, Ontario, Canada, 2004:169-176.

3 Mitchell D P, Netravali A N. Reconstruction filters in computer graphics[A]. In: Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques [C], Atlanta, Georigia, USA: ACM, 1988:221-228.

4 Boult T E, Wolberg G. Local image-reconstruction and subpixel restoration algorithms[A]. In: Proceedings of Graphical Model and Image Processing[C], Orlando, FL, USA: Academic Press, 1993, **55**:63-77.

5 Guenter B, Tumblin J. Quadrature prefiltering for high quality antialiasing [A]. In: Proceedings of ACM Transactions on Graphics [C], New York, NY, USA: ACM Press, 1996, **15**(4):332-353.

6 Fabris A E, Forrest A R. Antialiasing of Curves by Discrete Pre-filtering[A]. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques[C], New York, NY, USA: ACM Press, 1997: 317-326.

7 Liu Tao, Gao Qing-wei, Zhao Guo-rong. A New antialiasing technique for drawing straight line based on cone filter [J]. Journal of Engineering Graphics, 2006, **27**(3):61-64. [刘涛,高青伟,赵国荣. 一种新的基于圆锥滤波的直线反走样生成技术[J]. 工程图学学报,2006, **27**(3):61-64.]

8 Li Zhen-xiao, He Yuan-jun. Arbitrary width line generation and antialiasing [J]. Engineering Journal of Wuhan University, 2006, **39**(4):130-133. [李震霄, 何援军. 任意宽度直线的绘制与反走样[J]. 武汉大学学报(工学版), 2006, **39**(4):130-133.]

9 Pitteway M L V, Watkinson D J. Bresenham's algorithm with Grey scale[A]. In: Proceedings of Communications of the ACM[C], New York, NY, USA: ACM Press, 1980, **23**(11): 625-626.