一种支持多线宽直线反走样算法

骆朝亮
(中国地质大学 信息工程学院,湖北 武汉 430074)

摘 要:直线绘制中出现的锯齿现象称为走样,消除走样的方法称为反走样。文中通过对直线走样产生的原因进行理论上的分析,总结了现有的反走样技术。通过对经典的DDA直线绘制算法和Wu直线反走样绘制算法的研究,在二者结合的基础上,给出了一种任意宽度和复杂背景色下的直线反走样快速绘制算法:对于直线 $f(x)=mx+b, 0\leqslant m\leqslant 1$, x 轴上每移动一个像素单位,根据直线所需绘制的宽度,在 y 轴上进行跨度像素着色,填充的色深值取决于该像素到对应直线边缘线的距离、原有背景色深和当前直线绘制色深。对算法进行了去浮点优化,给出了复杂度分析和实验结果,实践证明,该算法有很好的执行效率和反走样效果。

关键词:直线绘制;反走样;DDA;快速
中图分类号:TP391.72 **文献标识码**:A **文章编号**:1673-629X(2010)09-0102-04

Rapid Algorithm for Anti-aliasing of Arbitrary Width Line Drawing

LUO Chao-liang
(Faculty of Information Engineering, China University of Geosciences, Wuhan 430074, China)

Abstract: Theoretically analyzes anti-aliasing problem. A solution based on the classical DDA (digital differential analyzer) algorithm and Wu anti-aliasing algorithm is proposed to address the issue. The solution offers an efficient line drawing algorithm which is applicable to arbitrary line width and complex background color: for line $f(x)=mx+b, 0\leqslant m\leqslant 1$. Specifically the algorithm makes pixel by pixel step wise move along $x(y)$ axis while filling span pixels along line direction of $y(x)$ based on line width. The filling color is dependent on three factors including current pixel's distance from the line's centerline, original background color and current line drawing color. The algorithm is further optimized with floating-point and complexity analysis, experiment result and application of the new algorithm are presented.

Key words: line drawing; anti-aliasing; DDA; rapid

0 引 言

直线的生成是计算机图形学中一个基本内容。在目前的光栅显示器中,每个像素用一对整数型坐标及其色深值来表示。对于一条直线 $f(x)=mx+b$,按斜率可以分为两类:当 $0\leqslant m\leqslant 1$,直线以 x 轴方向为主;当 $m>1$ 时,直线以 y 轴方向为主。当直线以 x 轴方向为主时,每在 x 方向上的坐标移动1个单位像素,就在 y 轴上计算出与该直线相距最近的像素点,并且填充该像素点,这一系列的像素点组合成最终所看到的直线,这就是直线绘制算法的基本原理,对于以 y 轴方向为主的情况,反之亦然。其中最常用的直线绘制算法有DDA算法和Bresenham算法,这种算法在绘制过程中对于每一个离散的像素点,要么填色,要么不填色,导致所绘制直线的边缘会出现锯齿现象,这种用离散量表示连续量所造成的失真效果,称为走样(见图1),消除这种走样的方法称为反走样[1-15]。文中通过对经典的DDA直线绘制算法[12]和Wu直线反走样绘制算法[13]的研究,在二者结合的基础上,给出了一种任意宽度和复杂背景色下的直线反走样快速绘制算法,取得了很好的效果。

图1 未反走样直线光栅化绘制效果

1 反走样技术

1.1 提高屏幕分辨率

在实际情况中,如果屏幕的分辨率足够高,那么每个像素的尺寸就会很小,所绘制出的直线的锯齿也会

收稿日期:2009-12-30;**修回日期**:2010-03-25
基金项目:国家自然科学基金(40771165)
作者简介:骆朝亮(1986-),男,湖北应城人,硕士研究生,研究方向为嵌入式GIS与GPS等。

很细,甚至可以达到人眼不易察觉的程度。该方法的优点在于不需要特殊的直线绘制算法,缺点对失真的降低是有限的,而且受技术和设备的限制,硬件成本比较高[1,3,12]。

1.2 全屏后处理法

对整幅图像全部绘制完毕后,在显示之前先做边缘检测,然后对边缘像素点做平滑。该方法的优点是整体效果很好,缺点是由于一般需要对整幅图像做边缘检测,加上做平滑处理也需要时间,因此在计算量和存储量上需要花费比较大的代价,实用性受到限制,特别是针对嵌入式设备。

1.3 直线反走样绘制

在直线的绘制过程中,对直线边缘处的像素点的色深值采用特定的计算方法动态计算,从而达到平滑过渡的效果。著名的有 Wu 直线反走样算法[13],和一些在 Wu 算法基础上的改进算法[1~16]。

但这些算法都存在某些方面的问题:

(1)不能良好支持任意线宽的直线绘制。一般直线反走样算法只支持固定像素的宽度的直线绘制,而对于任意宽度的直线反走样绘制,效果则不够理想[2~5,7,11,13,15]。

(2)算法效率不够高,不能很好适应嵌入式设备的运行要求。由于嵌入式移动终端的计算能力和内存空间的限制,当屏幕所需绘制直线数目较多,会出现绘制不及时的情况[1,6,8~10,14,16]。

(3)不支持复杂背景色下的直线反走样绘制。当背景色并非为单一色,而是复杂背景色的情况下,在背景色变化的过程中不能平滑地过渡[1,4~6,13,15]。

文中在 DDA 算法和 Wu 算法的基础上,给出了改进的快速直线反走样算法,并对算法进行了消除浮点计算优化,提高了算法的效率,实现了对任意线宽和多种背景色下的直线反走样绘制的支持。经实验证明,本算法具有良好的反走样效果和高效的执行效率。

2 算法描述

将直线看做信号 $s(x,y)$,其中的 x 与 y 分别为水平坐标和垂直坐标。其作用域为屏幕区域 $D(m,n)$,其中的 m 与 n 分别表示屏幕的水平和垂直像素个数,$(x,y) \in D$。对其可进行二维傅里叶变换至频域的信号 $S(u,v)$[6]。

对于屏幕的分辨率为 $m \times n$ 的情况,可以将光栅显示的屏幕看作二维连续区域的 D 的采样,其采样频率为屏幕的像素分辨率。

2.1 DDA 直线绘制算法

对于直线 $f(x) = mx + b$,采用通常的 DDA 直线绘制算法进行绘制:在以 x 轴方向为基准的情况下(y 轴同理),x 轴方向每移动一个像素单位,可以精确计算出 y 轴方向的坐标 $f(x)$,然后在 y 轴方向选取离 $f(x)$ 最近的像素点作为选取点填充(如图 2 中的小圆点),因此对于屏幕内的像素点采样后的结果只有两种:1(选取点) 和 0(非选取点)。则滤波函数可以表示为:

$$H(u,v) = \begin{cases} 1, (u,v) \in P \\ 0, (u,v) \notin P \end{cases}$$ (P 为绘制直线的选取点集)

在理想状况下,直线的边缘是无限光滑的,即其频谱为无穷大,而现实中屏幕总是具有一定的分辨率 (m,n),即有一定的采样频率,这样必然会产生信号失真,造成大量的边缘信息丢失,如图 2 所示,离散的像素点集无论如何也完整地保留原有直线的所有信息。

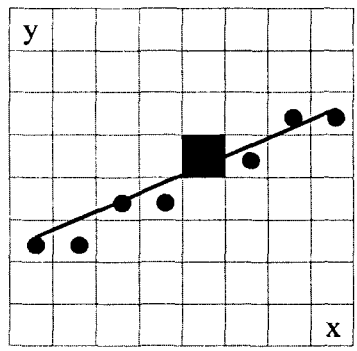

图 2　DDA 直线绘制算法

2.2 Wu 直线反走样算法

目前的显示器绝大部分都可以支持多位色深,这也就意味着像素点可以具有不同颜色值,对于一个采样点,采样得到的结果不止只有 0 和 1 两种,可以用不同的颜色深度表示该采样点到直线的程度。此时的滤波函数可变为:

$$H(u,v) = \begin{cases} d, & d \in [0, \text{ColorDepth}](u,v) \in P \\ 0, & (u,v) \notin P \end{cases}$$

其中,ColrDepth 指像素所能表示的最大色深值。

如图 3 所示,以 x 轴方向为基准,直线在 y 轴上跨越两个像素,按照 DDA 直线绘制算法,上方的像素点离直线的精确 y 坐标最近,只有该像素被完全填黑。若采用 Wu 直线反走样算法,则可以对相关联的上下像素点,根据其对直线的不同程度,给予不同的色深值来表示采样结果,从而精确地表示直线。

2.3 文中改进算法

在 Wu 直线反走样算法中,关键在于像素点颜色深度 d 的计算,现给出文中的计算算法:

设所绘制的直线为 $f(x) = mx + b$,其中 m 的取

值为 $0 \leqslant m \leqslant 1$(这里只讨论小于或等于1的情况,大于1时可将 x 轴和 y 轴互换处理)。

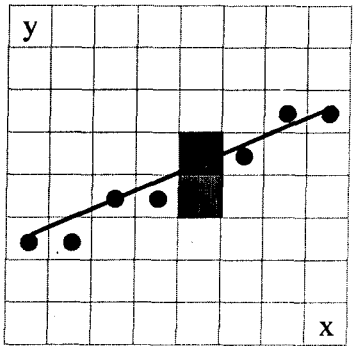

图3 反走样直线绘制算法

总体算法流程为:

(1)计算直线沿 y 轴方向上的跨度 W_y。对于宽度为 W 的直线,其在 x 轴方向每移动一个像素单位时,在 y 轴方向则需要填充相应的像素跨度 W_y,通过如下公式计算:

$$W_y = W * \sqrt{1 + m^2}$$

(2)边缘像素点填色。如图4所示,根据直线方程,可以计算出直线的上边缘线到其所跨相邻两像素中心点的距离 $d1, d2$。同理亦可以计算出下边缘线到其所跨相邻两像素的中点的距离 $d3, d4$。由于显示器的像素点是不可以分割的,直线 y 轴方向的宽度 W_y 应为整数,根据关于中心线上下对称性可知:$d1 = d3, d2 = d4$。所以在实际中只需要计算出 $d1, d2$ 即可。

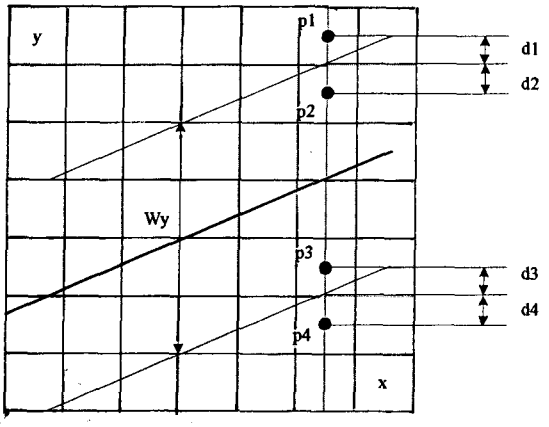

图4 边缘像素点填充的计算

设像素点 $P1$、$P2$、$P3$、$P4$ 原始的背景色深值分别为 C_1, C_2, C_3, C_4,所直线绘制色深值为 C_p,则对边缘点 $P1$、$P2$、$P3$、$P4$ 所应填充的色深值 C_1', C_2', C_3', C_4' 进行计算:

$$\begin{cases} C_1' = (d1 * C_1 + d2 * C_p)/(d1 + d2) \\ C_2' = \mathrm{Min}[(0.5 + d2), 1] * C_p \\ C_3' = \mathrm{Min}[(0.5 + d3), 1] * C_p \\ C_4' = (d3 * C_p + d4 * C_4)/(d3 + d4) \end{cases}$$

通过对背景色的实时性获取以计算边缘填充色,确保当背景色深值发生变化时,亦能具有良好的反走样效果。

(3)中间像素点填色。对于上下边缘线直线内部的像素点,由于这类像素点所处位置处于直线内部,如图4中处于 $P2$ 和 $P3$ 中间的像素点,边缘线不穿过其所在像素点区域,直接填充为直线绘制色深值即可。

(4)直线端点处理。当绘制直线的宽度大于1时,需要对直线两端线的端点进行特殊处理,和边缘点的处理方式类似,此时可以把直线的端线看作边缘线,计算出端线两侧的像素中心点到直线端线的距离,如果像素点仅是端点,直接计算相应的距离值 $d1, d2$,类似于边缘点的处理方式计算出相应的填充色深值;如果像素点既是直线端点又是边缘点,则分别计算两种情况下的色深值,最后取其二者平均值做为填充色深值,保证直线端点处的平滑过渡。

3 算法优化

在上述算法中,由于直线的斜率 m 的取值为 $0 \leqslant m \leqslant 1$,即 x 方向每前进一个像素,则 y 方向前进 m 个像素,在上述计算中 $d1, d2, d3, d4$ 均为小数,现采用近似计算消去算法中的浮点计算,以提高算法的效率。

将每个像素点扩大 N 倍(实验中 N 取值为20),在运用文中算法绘制直线的过程,x 轴方向每次前进 N 个像素,y 轴方向每次前进 $m * N$ 个像素,则原有的浮点计算则可以化成如下的整数计算:

$$\begin{cases} C_1' = (d1 * C_1 + d2 * C_p)/(N) \\ C_2' = \mathrm{Min}[(N/2 + d2), N] * C_p/(N) \\ C_3' = \mathrm{Min}[(N/2 + d2), N] * C_p/(N) \\ C_4' = (d3 * C_p + d4 * C_4)/(N) \end{cases}$$

当 x 轴方向每前进一个像素单位,y 轴方向需要进行跨度为 W_y 像素填充,在这个过程中,仅上下边缘像素点需要动态地计算其填充色深值,而对于中间点可以直接填充为直线绘制色,这个过程为4次整数运算,端点处的计算因为涉及区域很小可以忽略不计。这样,优化后本算法的算法复杂度为 $O(N)$ 整数运算。

4 实验结果

文中算法在拥有 ARM9 芯片的多普达 P300 的设备上进行了测试,实验结果如图5,6所示:图5是在多背景色下的显示实例,图6是图5的局部放大图。可以看到,此算法有着很好的抗锯齿效果,在背景色深值变化的衔接处,因为采用了实时获取背景色深值来计算边缘点像素填充色,过渡非常平滑。针对 Wu 直线

反走样算法只能适用于单像素宽的直线绘制的不足,文中算法可以适用于任何宽度的直线,并且都能取得很好的显示效果,同时不仅对单一背景色,对复杂背景色也都能获得平滑的绘制效果,这是其他算法不能做到的。

在实验中也对算法的效率做了相关的测试,在显示分辨率为 240×320 屏幕上,随机产生 100 条直线并进行反走样绘制,基本整体无延迟刷屏效果出现,非常适合于嵌入式设备上的直线反走样绘制。

图 5　多背景色下整体效果

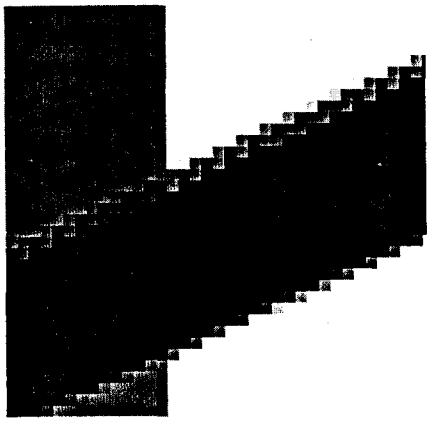

图 6　多背景色下局部放大效果

5　结束语

文中在 DDA 直线光栅算法和 Wu 反走样算法的基础上进行了扩展,给出了一种快速的可以支持任意宽度的直线反走样绘制算法,并且能适应复杂背景颜色下的绘制,并优化去掉算法中的浮点运算,提高了算法的执行效率,经过实验证明,非常适合于嵌入式设备上的美观性反走样直线的绘制。

参考文献:

[1] 杭后俊,付　勇.一种基于加权区域采样的直线反走样生成算法[J].计算机技术与发展,2009,19(6):138－142.

[2] 杜晨辉,经亚枝.全罗盘画面反走样算法的研究和实现[J].南京航空航天大学学报,2002,34(4):391－393.

[3] 张　波,张焕春,经亚枝.罗盘刻度线反走样快速绘制算法的改进研究[J].计算机辅助设计与图形学学报,2003,15(1):71－75.

[4] 江　修,张焕春,经亚枝.三像素宽反走样直线的绘制算法研究[J].南京航空航天大学学报,2003,35(2):148－151.

[5] 韩　丽,唐　棣.一个快速有效的直线反走样算法[J].小型微型计算机系统,2005,26(3):509－511.

[6] 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.

[7] 谢　莹,许荣斌,赵宏坤.基于嵌入式图形系统的改进 Bresenham 反走样算法[J].计算机技术与发展,2006,16(11):100－103.

[8] 王宇平.光栅显示图形中直线反走样技术算法的改进[J].哈尔滨理工大学学报,2008,13(3):51－54.

[9] 杨爱良,杨　睿,熊智勇.反走样技术在计算机图形仿真中的运用[J].计算机仿真,2005,22(4):124－126.

[10] 郝　鑫,何援军,柳　伟.一种基于灰度的三角形反走样算法[J].自动化技术与计算机技术,2008,25(5):143－146.

[11] 沈　强,张　波,陈淑珍,等.计算机图形学反走样技术及实现[J].武汉大学学报,1997,43(1):113－118.

[12] 孙家光,杨长贵.计算机图形学[M].北京:清华大学出版社,1995.

[13] Wu X. An efficient anti－aliasing technique[J]. Computer Graphics,1991,25(4):143－152.

[14] Diakopoulos N A,Stephenson P D. Anti－Aliased Lines Using Run－Masks[J]. Computer Graphics, 2005, 24(2): 165－172.

[15] Liu Yong Kui. An All－integer Algorithm for Drawing Anti－aliased Straight[J]. Computer Graphics, 1994, 13(4): 219－222.

[16] Thomas D, Netravali A N, Fox D S. Anti－aliased Ray Tracing with Covers[J]. Computer Graphics, 1989, 8(4): 325－336.

(上接第 101 页)

[6] Morgan T D. Recovering deleted data from the windows registry[J]. Digital Investigation, 2008(5): 33－41.

[7] 崔蓓蓓,陶　亮.并行格型结构实现基于 DCT 的 2D 实值离散 Gabor 变换[J].计算机技术与发展,2009,19(2):105－108.

[8] 王　硕,尤　枫,赵恒永.网页树型结构快速加载大数据量数据的实现[J].计算机工程与应用,2008,44(27):166－171.

[9] 杨祖龙,吴国平.递归算法在树型视图中的应用[J].计算机工程,2002,11(3):139－141.

[10] 赵亚萍.基于 Visual C#.NET 的空间缓冲区分析开发[J].计算机技术与发展,2009,19(12):29－35.

[11] 杨泽明,许榕生,刘宝旭.文件删除的恢复与反恢复[J].信息网络安全,2002(4):38－41.

[12] 涂彦晖,戴士剑.数据安全与编程技术[M].北京:清华大学出版社,2005:30－50.