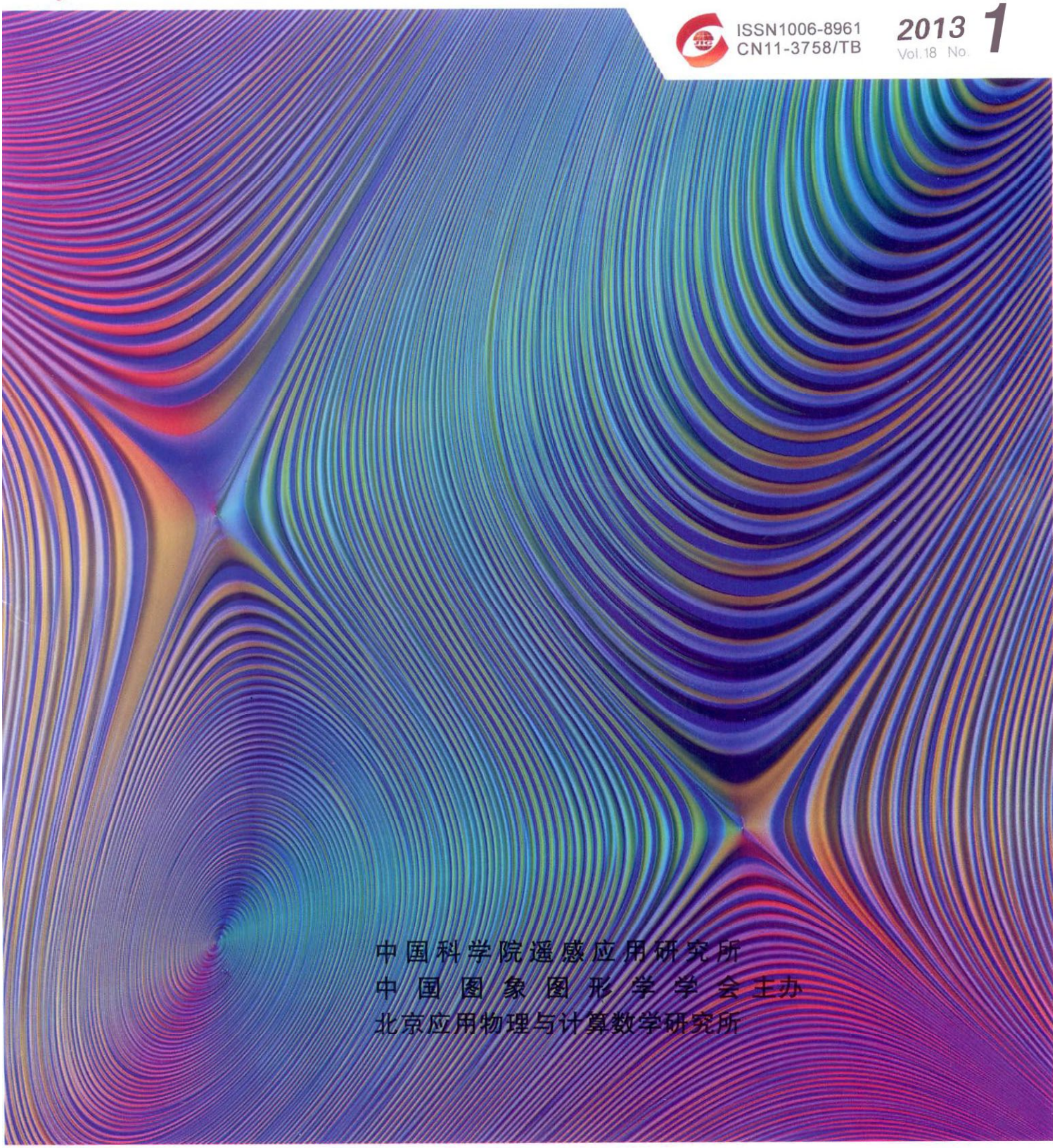
Journal of Image and Graphics

中国图象图形学报

ISSN1006-8961
CN11-3758/TB

2013 1
Vol.18 No.



中国科学院遥感应用研究所
中国图象图形学学会主办
北京应用物理与计算数学研究所

中国图象图形学报

Zhongguo Tuxiang Tuxing Xuebao

2013 年 1 月　第 18 卷　第 1 期(总第 201 期)

目　　次

综述

正面人脸图像合成方法综述 …… 赵林，高新波，田春娜(1)

SAR 图像道路网提取方法综述 …… 程江华，高贵，库锡树，孙即祥(11)

图像处理和编码

头部缺失的 JPEG 文件碎片恢复 …… 徐明，黄立，张海平，徐建，郑宁(24)

安全高效的可撤销指纹模板构造 …… 喻建平，张鹏，王瑶，杨懿竣(36)

基于残差的图像超分辨率重建 …… 陈华华，姜宝林，刘超，陈伟强，陆宇，张嵩(42)

旋转的 Wang Tiles 纹理合成算法 …… 王继东，庞明勇，赵瑞斌(49)

基于圆形约束快速水平集的原生质体细胞分割 …… 王晓飞，庞全(55)

图像分析和识别

采用压缩传感的鲁棒的视频指纹方案 …… 孙锐，李超，蒋飞云(62)

基于图像显著性的路面裂缝检测 …… 徐威，唐振民，吕建勇(69)

基于局部熵的主动轮廓模型 …… 潘改，高立群，赵爽(78)

基于算子的图像分解 …… 李峰，曾晓辉，陈盛霞，沈玉娟(86)

图像理解和计算机视觉

中值流辅助在线多示例目标跟踪 …… 王德建，张荣，尹东，张智瑞(93)

四叉树直方图的特殊方向关系表达 …… 张珂，王小捷，靳越(101)

计算机图形学

协同进化的近似规则纹理合成 …… 王相海，陶兢喆(107)

反走样直线的灰度循环生成算法 …… 牛连强，张胜男，钟玲(115)

地理信息技术

微博客蕴含交通信息的提取 …… 张恒才，陆锋，陈洁(123)

"计算机视觉前沿论坛"专栏征文通知 …… (130)

Journal of Image and Graphics

(**Monthly, Started in 1996**)

Vol. 18 No. 1 January 2013

Contents

Review

Review of frontal face image synthesis methods ····· Zhao Lin, Gao Xinbo, Tian Chunna(1)

Review of road network extraction from SAR images ····· Cheng Jianghua, Gao Gui, Ku Xishu, Sun Jixiang(11)

Image Processing and Coding

Rrecovery method for JPEG file fragments with missing headers ····· Xu Ming, Huang Li, Zhang Haiping, Xu Jiang, Zheng Ning(24)

Secure and efficient scheme to construct a cancelable fingerprint template ····· Yu Jianping, Zhang Peng, Wang Yao, Yang Yijun(36)

Image super-resolution reconstruction based on residual error ····· Chen Huahua, Jiang Baolin, Liu Chao, Chen Weiqiang, Lu Yu, Zhang Song(42)

Texture synthesis using rotational Wang Tiles ····· Wang Jidong, Pang Mingyong, Zhao Ruibin(49)

Protoplasm somatic cells segmentation based on circle dependent fast level-set segmentation ····· Wang Xiaofei, Pang Quan(55)

Image Analysis and Recognition

Robust video fingerprinting via compressed sensing ····· Sun rui, Li Chao, Jiang Feiyun(62)

Pavement crack detection based on image saliency ····· Xu Wei, Tang Zhenmin, Lv Jianyong(69)

Active contour model driven by local entropy energy ····· Pan Gai, Gao Liqun, Zhao Shuang(78)

Operator-based image decomposition ····· Li Feng, Zeng Xiaohui, Chen Shengxia, Shen Yujuan(86)

Image Understanding and Computer Vision

Median flow aided online multi-instance learning visual tracking ····· Wang Dejian, Zhang Rong, Yin Dong, Zhang Zhirui(93)

Expression of special directional relation based on quadtree histogram ····· Zhang Ke, Wang Xiaojie, Jin Yue(101)

Computer Graphics

Fast near-regular texture synthesis based on the concept of co-evolution ····· Wang Xianghai, Tao Jingzhe(107)

Integral algorithm for generating anti-aliased straight line controlled by gray iteration ····· Niu Lianqiang, Zhang Shengnan, Zhong Ling(115)

Geoinformatics

Extracting traffic information from massive micro-blog messages ····· Zhang Hengcai, Lu Feng, Chen Jie(123)

中图法分类号: TP391 **文献标识码**: A **文章编号**: 1006-8961(2013)01-0115-08
论文引用格式:牛连强,张胜男,钟玲. 反走样直线的灰度循环生成算法[J]. 中国图象图形学报,2013,18(1):115-122.

反走样直线的灰度循环生成算法

牛连强[1,2], 张胜男[1], 钟玲[2]
1. 沈阳工业大学信息科学与工程学院, 沈阳 110870; 2. 沈阳工业大学软件学院, 沈阳 110023

摘 要: 灰度转换是整数反走样直线和曲线绘制算法中较为耗时的操作。为了提高反走样直线的绘制效率,提出了一种直接利用灰度循环控制生成反走样直线的纯整数算法。该算法采用两点反走样模式,根据对灰度值的分解、每次步进时的像素及其灰度值直接利用灰度增量控制产生,从而避免了候选点与真实直线之间的距离计算,以及由距离到灰度的转换。分析表明,该算法每次步进仅需要 4 ～5 次整数基本运算,其效率较现有整数反走样直线生成算法有大幅度提高,且具有与基本直线生成算法相似的简单性。

关键词: 直线生成; 反走样; 光栅转换; 整数运算; 灰度循环控制;灰度转换

Integral algorithm for generating anti-aliased straight line controlled by gray iteration

Niu Lianqiang[1,2], Zhang Shengnan[1], Zhong Ling[2]
1. *School of Information Science and Engineering, Shenyang University of Technology, Shenyang* 110870, *China*;
2. *School of Software, Shenyang University of Technology, Shenyang* 110023, *China*

Abstract: Gray-conversion is a time-consuming operation for drawing anti-aliased straight lines or curves. In order to improve the efficiency of drawing anti-aliased straight lines, we propose a new anti-aliasing algorithm based on a two-pixel model. Different from other anti-aliasing algorithms, a technique which controls iterations by grey values directly rather than distances is proposed. Calculations about distances between pixels and real line and distance-grey conversion are abandoned, but accurate grey values rather than estimated ones can be obtained. The analyses prove that for one step only 4 ～5 integer operations are invoked in this algorithm, and the drawing speed is faster than other methods. Furthermore, this algorithm can be constructed with the simplicity similar to basic scan conversion algorithms.

Key words: straight line drawing; anti-aliased; scan-conversion; integer operations; grey-iteration control; grey-conversion

0 引 言

在光栅图形显示器上,直线由离散的像素点来近似描绘,由于采样不充分会导致显示时存在着“走样”问题[1]。反走样处理的目的就是要减少和消除图形的之字形边界,以提高图形显示的真实感,其方法因作用于图像边界以及独立的曲线而有所不同。其中,可以将有关直线(曲线)反走样绘制的研究结果归结为两大类,分别是增加分辨率方法和增加过渡颜色方法。通常,前者主要体现在硬件水平的提高,但它只能减轻而不能消除走样,且代价较高,后一种方法的代价小,质量高,一般可依赖软件实现,故被广泛采用。

收稿日期:2012-03-19;**修回日期**:2012-05-30
基金项目:辽宁省科技计划项目(2007410003);沈阳市科技计划项目(F12-168-9-00)
第一作者简介:牛连强(1965—),男,教授,沈阳工业大学电气工程专业在读博士研究生,主要研究方向为 CAD&CG、图像处理、可视化及仿真技术等。E-mail:niulq@sut.edu.cn

由于直线、圆等图形基元的生成算法对整个图形系统的效率和质量具有重要影响。因此,从 20 世纪 70 年代起,这些基元的光栅转换以及相应的反走样算法的研究就已开始,至今仍受到关注[2-7]。一种典型的反走样方法是采用矩形滤波,它通过区域采样来确定像素被直线覆盖的区域大小,但这种非加权模式不能达到最优的频率响应[8-12],采用圆锥滤波器则可以使效果得到改善[13]。为了提高反走样绘制速度,文献[14]分析了直线上像素的对称性质,并利用圆锥滤波器和预查表来快速查找像素中心与真实直线之间的距离以得到近似的像素亮度,利用同时点亮 3 个像素的方法实现直线的反走样绘制。文献[15]提出了将像素网格进行逻辑细分,得到比实际像素更小的子像素,再计算子像素的个数作为亮度依据的方法,有效地提高了反走样的效果。文献[16]提出了一种同时点亮两个离直线最近的像素,并按其与直线的近似距离分配以不同的亮度,从而达到反走样目的的方法(以下称为 Wu 算法)。这种加权方法需要计算的灰度以及点亮的像素少,速度快。

光栅直线的基本生成算法(如 Bresenham 算法或中点法)的重要长处之一在于其整数化,但目前尚不存在真正意义上的与直线的基本生成算法近似的反走样直线生成算法。Wu 算法可以利用定点小数提高绘制速度,但本质上仍是浮点型算法。文献[17]提出的整数型反走样算法是利用对像素与真实直线间距离的估计值实现的,不仅灰度转换的代价很大,灰度值也是近似的。文献[18]给出了一个多点式的整数反走样直线绘制方法,但该方法复杂,且仍依赖距离到灰度的转换,并需要较多的乘除法运算。文献[19]提出了一种简单的反走样整数绘制方法,并阐明了利用预查表得到像素灰度的思想,缺点是需要消耗较多的内存。

总体上,现存的反走样绘制算法都是以基本直线生成算法为基础的,因此,需要以像素与真实直线的距离或直线覆盖像素的面积大小为依据,再转换为对应的灰度,这就难免要使用除法,或处理浮点数,或逻辑细分像素进行近似计算,如图 1 所示。本文称现存方法为基于距离控制的生成算法。由于缺乏有效的整数反走样算法,加之需要引入复杂的概念,一般图形学书籍中都没有给出相应的算法描述。

提出一种仅依据直线的性质和灰度变化的周期性,直接通过灰度变化控制迭代过程、无需进行后期灰度转换的纯整数快速反走样直线生成算法。由于既消除了计算距离的过程,也不必再消耗由距离到灰度的转换时间,使得该算法几乎与直线的基本生成算法具有相同的效率和简单性。

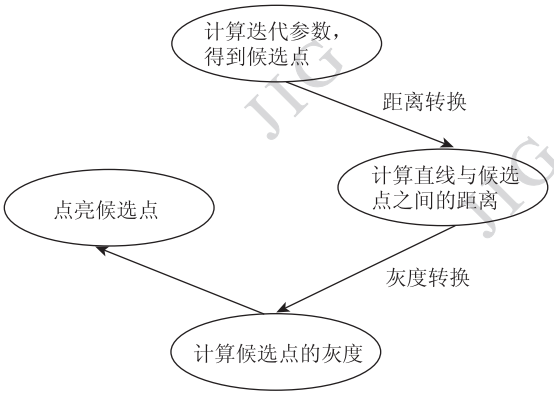


图 1 反走样直线绘制的一般过程

Fig. 1 Normal processing method for anti-aliased drawing

1 相关工作分析

考虑到直线的方向与 x 和 y 坐标的可对称调整性,记直线的斜率为 k,以下仅给出对第 1 个 1/8 象限($0<k<1$)内的光栅直线的讨论,其他情况可由对称性得到。约定在候选像素与真实值的误差为 0.5 时,取下方像素。直线的起点为(x_1, y_1),终点为(x_2, y_2),$\Delta x = x_2 - x_1$,$\Delta y = y_2 - y_1$。此外,不严格区分词汇亮度和灰度,本质上,亮度越大,其灰度数值越小。

这里主要回顾几种典型的反走样直线的整数绘制算法以及两点模式的灰度转换技术。

1.1 Wu 算法

1991 年,Wu 提出了一种十分简单有效的两点模式的反走样直线绘制方法[16],如图 2 所示。

在进行光栅转换时,直线上的真实点 P 由满亮度的像素 P_1 来近似,而产生反走样直线时则由离其最近的"跨骑"直线的两个像素 P_1、P_2(候选点)来表示,距离直线近的像素分配更大的亮度,但两像素的亮度 I_1 和 I_2 之和为一个满亮度 G,即 $I_1 + I_2 = G$。P_1 和 P_2 称为"Wu 像素(对)"。

由于两个相邻像素间的距离为 1,因此,图 2 中的距离 d_1 和 d_2 分别由 $\sum k$ 和 $1 - \sum k$ 确定,其中的直线斜率 k 是一个浮点数。于是,可将其转换为对

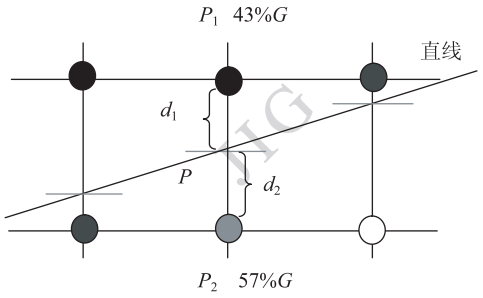


图 2　用 Wu 像素实现的反走样直线绘制

Fig. 2　Drawing anti-aliased line with Wu-pixels

应的亮度

$$I_1 = d_1 \times G$$
$$I_2 = d_2 \times G \tag{1}$$

为了加快亮度转换,可以将 d_1 和 d_2 用定点小数表示,从而将式(1)转化为位移运算。

1.2　Liu 算法

1994 年,文献[17]给出了一个较完整的整数反走样直线绘制方法。该方法先根据 Bresenham 直线生成算法的决策参数近似算出 Wu 像素的相对位置(距离)

$$d = \Delta x(d_1 - d_2) \tag{2}$$

然后,将 d 所在区间分成 $2n$ 份($2n$ 也是灰度级数),并根据 d 所处的区间再计算分配给 P_1 的近似灰度值。

首先,这种方法需要存储预置的灰度等级。其次,由于原算法中采用的顺序比较的运算量过大,可以考虑利用二分检索完成距离与灰度的匹配,但即便如此,对于 256 级灰度,一次灰度转换的平均时间消耗也约为 $3 \times 8 = 24$ 次基本运算。

1.3　其他算法

文献[18]于 1996 年提出了另一种反走样直线整数绘制方法(Chung-Tsai 算法),但该方法主要研究了如何依据直线特性进行多点绘制。在仅有一个中间灰度时,灰度可事先估计得到;对于多灰度级,灰度值要采用乘除法和取整运算来确定。该算法较复杂,迭代中需要采用乘除法运算,难说明其优势。

文献[19]提出了一个比较简单的反走样绘制方法,该方法在迭代时利用决策参数表示出了被选择像素与直线的距离 d 以及完整的距离范围 S,于是,灰度值可以由 $(d/S) \times G$ 计算出来。考虑到显示设备的尺寸都是有限的,可以事先保存一个预查表 Lookup_Table,表的第 k 行(如 $0 \leqslant k < 1\,024$)、第 n(如 $0 \leqslant n < k$)列存储对应灰度 $\frac{n}{k} \times G$。于是,灰度转换仅由一次对表的索引即可完成,其时间消耗主要是 2 次地址计算时间。因为要消耗近 512 kB 的内存,这通常是难以接受的。

2　基于灰度控制生成反走样直线

这里首先分析反走样直线的灰度变化规律,再构造直接采用灰度控制迭代过程的反走样直线生成算法。

2.1　像素的灰度变化与 Y 方向步进的关系

在对 $0 < k < 1$ 内的直线进行光栅转换时,X 方向为步进主轴,即如果第 i 步已绘制了点(x_i, y_i),下一步要确定应绘制的目标像素(x_{i+1}, y_{i+1}),其中,$x_{i+1} = x_i + 1$,需要解决的问题是在候选点(x_{i+1}, y_i)和$(x_{i+1}, y_i + 1)$中确定哪个像素应被选择,如图 3 所示。

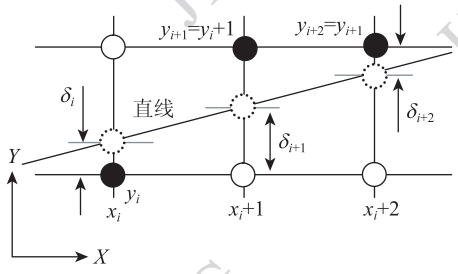


图 3　根据误差选择目标像素

Fig. 3　Selecting object pixels according to residuals

记 δ_{i+1} 为点(x_{i+1}, y_{i+1})到直线的距离(残差),$i \geqslant 0$。那么,如果 $\delta_{i+1} > 0.5$ 时,基本光栅直线生成算法将产生 Y 方向(包括 X 方向)的步进,否则,仅在 X 方向产生步进。

分析利用垂直方向的两个像素合成一个“灰度像素”的反走样直线的灰度变化可以发现,反走样直线上像素的灰度变化与前述的生成光栅直线相比,步进点有半个周期的错位,即二者在 Y 方向产生步进位置并不一致,如图 4 所示。

在生成光栅直线时,Y 方向是否产生步进由残差 δ_{i+1} 决定。因此,在 x_1 点处,如果 Y 方向残差超过 0.5,光栅直线将产生 Y 方向步进,到达点(x_1, y_2)而不是点(x_1, y_1)。但是,由于此时离直线最近的两个像素仍是点(x_1, y_1)和点(x_1, y_2),在绘制反走样直线时,并不需要沿 Y 方向步进,而应沿 X 方

向步进到点(x_1, y_1)。在 x_2 点处,点(x_2, y_2)与直线的距离小于 0.5,光栅直线不会产生 Y 方向步进,但点(x_1, y_1)已经远离直线,距离直线最近的两个像素是点(x_2, y_2)和点(x_2, y_3),因此,在绘制反走样直线时必须产生 Y 方向的步进。简言之,反走样绘制时应在完成一次灰度循环后才产生 Y 方向的步进。

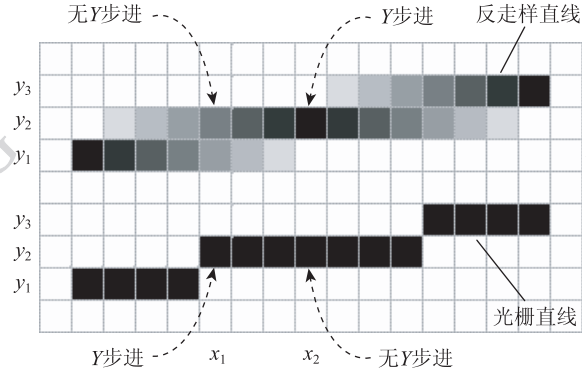


图 4 光栅直线与反走样直线步进点的差异

Fig. 4 Difference of stepping position in Y-axis between drawing a line and an anti-aliased line

2.2 算法构造与描述

在直线斜率 k 一定时,利用整数分解可以将一个 Wu 像素对的灰度计算出来。

记 G 为最大灰度级。因为起始点位于直线上,其灰度值为 0。对于光栅直线上的 x_i 点,其 y 坐标为 $k\times i$,相应的灰度值为

$$g_i = k\times i\times G$$

若记两点间的灰度变化为

$$\Delta G = k\times G$$

则有

$$g_i = g_{i-1} + \Delta G \tag{3}$$

由 2.1 节的讨论可知,若 $g_i \geqslant G$,则 Y 方向应发生步进。以下讨论将浮点数转换为整数的方法。

因为 $k = \Delta y/\Delta x$,有 $\Delta G = (\Delta y/\Delta x)\times G = (\Delta y\times G/\Delta x)$

令 Δg、Δr 分别表示 $\Delta y\times G$ 除以 Δx 的商和余数,即

$$\Delta g = \lfloor (\Delta y\times G)/\Delta x \rfloor$$
$$\Delta r = (\Delta y\times G) \mod \Delta x$$

则有

$$\Delta G = \Delta g + \Delta r/\Delta x \tag{4}$$

式(4)说明,每次步进时的灰度计算可以先累加 Δg,再用灰度余量(余数)的累加和 $\sum \Delta r$ 与 Δx 比较进行校正。如果 $\sum \Delta r \geqslant \Delta x$,灰度值增 1,并在 $\sum \Delta r$ 中去掉一个 Δx 部分,否则保持灰度值不变。

根据上述分析可以构造十分简洁高效的反走样算法。

算法 1 基于灰度控制的反走样直线生成算法

该算法中的% 表示取余数;所有变量均为整型量;x、y 为屏幕坐标。

```
Δx←x2 − x1,   Δy←y1 − y2;
x← x1,   y←y1;
G←255,   g←0;
ΔG←(G×Δy)/Δx,   Δr←(G×Δy)% Δx;
acc←0,   ras←G − ΔG;
while(x≤x2)
{
  Pixel(x, y, g);   Pixel(x, y − 1, G − g);
  if(g < ras)
    g←g + ΔG;
  else
  {  y←y − 1;
     g←g − ras;
  }
  acc←acc + Δr;
  if(acc≥Δx)
  {  g←g + 1;
     acc←acc − Δx;
  }
  x←x + 1;
}
```

其中,连续的 2 个函数“*Pixel*(x, y, g); *Pixel*(x, $y-1$, $G-g$);”的功能是分别用指定的灰度 g 和 $G-g$ 绘制像素点,从而构成一个 Wu 像素。

由于 $G = 2^n - 1$,其中的 n 为正整数,$G\times\Delta y = 2^n\times\Delta y - \Delta y$。因此,计算 $G\times\Delta y$ 仅需要一次位移和一次加法。此外,两个整数的商和余数本质上可利用一次除法得到。因此,算法中的初始化仅需要一次整数除法。

2.3 行程算法

算法 1 中的每次循环都需要对条件“$g < ras$”进行判定,消耗了较多的运算。事实上,很多研究者都对直线的快速绘制方法(如多点绘制和行程法等)进行了研究[4-6, 20-21]。注意到 2.2 节中所讨论的灰度计算与每次迭代时像素的残差无关,因此,可以将

算法 1 与任何一种快速算法相结合而得到相应的快速反走样绘制算法。

通常,光栅直线可认为由一系列连续的直线段所组成,而组成光栅直线的起始线段和最末线段的长度通常为中间线段长度的一半(也可能多 1 或少 1),这就给快速直线绘制带来了困难。

观察式(4)可知,在生成反走样直线时,相邻点每次灰度的步进量为 $\Delta G=(\Delta y\times G)/\Delta x$。
因此,达到满灰度 G 所需要的循环次数为 $G/\Delta G=\Delta x/\Delta y$。

若对 Δx 作整数分解

$$\Delta x=m\Delta y+\Delta s,0\leqslant\Delta s<\Delta y \tag{5}$$

则条件"$g\geqslant G$"只能在绘制 m 个像素后成立,而灰度循环长度可能是 m 或 $m+1$(像素)。

上述讨论说明,在直线斜率确定时,m 仅依赖斜率的大小,不会产生起始和结束时的特殊长度线段。于是,利用灰度控制循环可以构成更为简单的行程反走样生成算法。

算法 2　基于灰度控制的行程反走样直线生成算法

```
Δx←x2 − x1,   Δy←y1 − y2;
x← x1,   y←y1;
G←255,   g←0;
ΔG←(G×Δy)/Δx,   Δr←(G×Δy)% Δx;
acc←0;
m←Δx/Δy;
while(y > y2)
{
   for(xend←x + m; x < xend; x←x + 1)
   {
      Pixel(x, y, g);Pixel(x, y − 1, G − g);
      g←g + ΔG;
      acc←acc + Δr;
      if(acc ≥ Δx)
      {   acc←acc − Δx;
          g←g + 1;
      }
   }
   if(g < G)//处理第 m + 1 个像素
   {   Pixel(x, y, g);Pixel(x, y − 1, G − g);
       g←g + ΔG;
       acc←acc + Δr;
       if(acc≥Δx)
       {   acc←acc − Δx;
           g←g + 1;
       }
   }
   g←g − G;   y←y − 1;
}
Pixel(x, y, 0)。
```

与算法 1 相比,算法 2 需要增加一次除法,它对绘制稍长一些且 m 值较大时的直线更有优势。

本质上,算法 2 与一般的行程式光栅直线转换算法一致,都是将直线视为连续的线段,每个线段的长度为 m 或 $m+1$。因为不计算残差,反走样算法对直线的两端不需要特殊处理,故较一般的行程式光栅直线绘制算法更为简单。

2.4　消除对灰度误差的累加

2.3 节中的算法 2 利用行程方法消除了每次步进时对 g 的判断,仅在绘制一个线段(m 个像素)后产生一次判断。但这种改进还不能有效地消除每次步进时对 acc 的累加和判断。事实上,每次内层循环都包括对 acc 进行累加和判定,是否需要则由 Δx 与 Δr 的相对大小确定。为此,再作整数分解

$$\Delta x=p\times\Delta r+\Delta t \tag{6}$$

于是,可以仅在 p 次步进后作如下的一次性灰度累加

$$acc=acc+\Delta t\times G \tag{7}$$

其后,再对"$acc\geqslant\Delta x$"进行判定。这种方法可以减少每次步进时的基本运算次数,但需要在初始化中增加 2 次乘法(除法)。不过,由于反走样本身是一种近似,对于通常的 256 级灰度,在灰度误差较小时,反走样绘制的效果几乎没有任何变化,实际实验也验证了这一点。因此,可以在算法 2 的内层循环中只保留对 g 的累加,而将对 $acc\Delta r$ 的操作移到每次内层循环之后。

算法 3　改进的行程反走样直线生成算法

```
Δx←x2 − x1,   Δy←y1 − y2;
x← x1,   y←y1;
G←255,   g←0;
ΔG←(G×Δy)/Δx,   Δr←(G×Δy)% Δx;
acc←0;
m←Δx/Δy;
p←m×Δr,   Δc←p/Δx;
c←Δc×Δx,   p_c←p − c;
while(y > y2)
```

```
{
    for(x_end←x+m;  x<x_end; x←x+1)
    {   Pixel(x, y, g);Pixel(x, y-1, G-g);
        g←g+ΔG;
    }
    acc←acc+p_c;
    g←g+Δc;
    if(acc ≥ Δx)
    {   acc←acc-Δx;
        g←g+1;
    }
    if(g<G)
    {   Pixel(x, y, g);Pixel(x, y-1, G-g);
        g←g+ΔG;
        acc←acc+Δr;
        if(acc≥Δx)
        {   acc←acc-Δx;
            g←g+1;
        }
    }
    g←g-G;   y←y-1;
}
```

Pixel(x, y, 0)。

在每段绘制后,由于灰度误差的积累有可能达到一个新的 Δx,"$acc \geq \Delta x$"的判断是必需的。

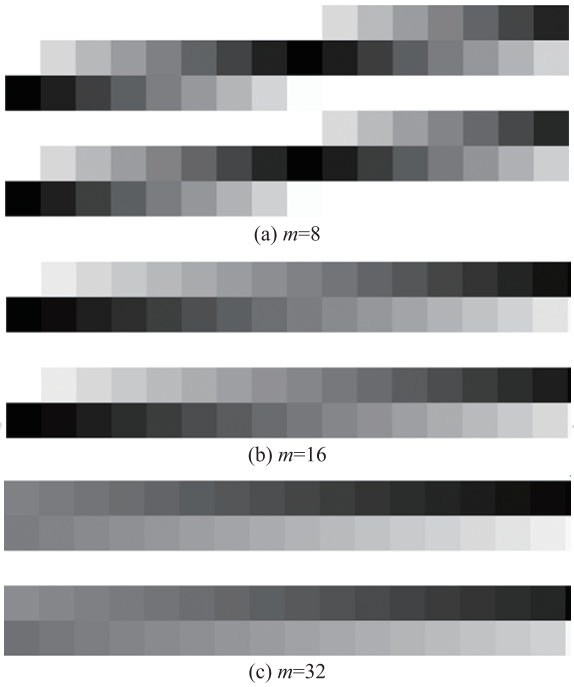
上述调整会产生一定的灰度误差,绘制的效果如图 5 所示。实际绘制不同斜率下的直线说明,当 $m \leq 16$ 时,算法 3 与算法 2 的绘制效果几乎完全相同。这是因为每个像素的灰度值较大,忽略误差校正至多使相邻间像素的灰度值差为 1,且能够维持其灰度的大小次序不变。在 $m \geq 255$ 的极端情况下,因为 $\Delta G = 0$,忽略灰度校正会使直线得不到反走样处理。在 $16 < m < 255$ 时,两种方法绘制的反走样直线有极为细小的视觉差异,m 越大差异也越大。因此,可以按如下方式对灰度值进行校正:

1)若 $m \leq 2$,直接利用算法 1 或算法 2 进行绘制;

2)若 $2 < m \leq 16$,利用算法 3 绘制;

3)若 $m > 16$,修改算法 3,将内层循环中一条线段的绘制分成[m/16]个子段进行绘制,每个子段绘制后就对灰度进行一次校正,只是累加的灰度误差为(m/16)×Δr。

应该说明的是,尽管算法 3 中计算 p、c 和 Δc 需要 3 次乘法(除法)运算,但由于其值很小,可以利用加法(减法)快速计算出这些值而避免除法。



(a) m=8

(b) m=16

(c) m=32

图 5 有无灰度误差校正时的效果对比

Fig. 5 Property comparision with or without gray correction

2.5 有关提高算法效率的进一步讨论

可以采取一些有效的措施进一步减少算法所消耗的运算量。

1)很多文献已经说明了光栅直线关于两端的对称性[4,15]。可以很容易将本文算法修改为每次步进从直线两端对称位置点亮两个像素,这会使算法消耗的运算量减少一半。

2)如果 $\Delta r = 0$,所有对 acc 的累加、判断以及对"$g < G$"的判断都是不需要的,组成直线的所有线段的长度都是 m,反走样直线可以只由 $g = g + \Delta G$ 更新灰度直接绘制出来。

3)算法 3 每次绘制 m 个像素的线段后进行条件"$g < G$"的判别,可以在连续绘制几条线段后再执行。这是因为,单独绘制一个具有 m 个像素的线段后的灰度误差为

$$m\Delta G = ((\Delta x - \Delta s)/\Delta x) \times ((\Delta y \times G)/\Delta x) = G - (\Delta s/\Delta x) \times G$$

因此,如果没有初始误差,条件"$g < G$"并不需要判别。若记

$$\Delta y = n\Delta s + \Delta w (0 \leq \Delta w < \Delta s)$$

则有

$$n \times m\Delta G = nG - \frac{\Delta y}{\Delta x} \times G + \frac{\Delta w}{\Delta x} \times G =$$

$$nG - \Delta G + (\Delta w/\Delta x) \times G$$

这说明,连续绘制 n 条长度为 m 的线段后,灰度累积减少量可能超过 ΔG,此时需要判别是否在第 n 条线段上可再继续绘制一个水平步进的像素。特别地,若 $\Delta w = 0$,此类直线可以不经中间判别直接绘制。同样,这种方法也需要增加 2 次乘法(除法)。

4)若 GCD(Δx, Δy)表示 Δx 和 Δy 的最大公约数,则光栅直线可以被划分为 GCD 段相同的模式[4],因此,只要记录第一段的模式,就可以直接复制到其他段。但是,这需要使用一个预先计算的 GCD 查找表,消耗部分内存。

3　算法分析及讨论

算法 1 利用直接对灰度的一次累加完全取代了距离计算,但需要附加一次加法和逻辑运算进行灰度校正。这是一种较精确地直接得到灰度值的方法。算法中的条件"$acc \geqslant \Delta x$"是否成立主要由$\lfloor \Delta x/\Delta r \rfloor$决定,从平均意义上说,每次步进需要 5 次基本运算(加法和比较)。算法 2 则需要 4 ～5 次基本运算,如果 m 和$\lfloor \Delta x/\Delta r \rfloor$较小,算法效率略低,反之,算法效率较高。此外,这些算法所消耗的计算量与灰度级别无关,不会因灰度级增大而增加额外的时间。

对于目前存在的 2 个主要的整数型反走样方法,即 Liu 算法及 Chung-Tsai 算法,首先,这两种方法的灰度值都是近似的,如果要由距离转换为灰度,每次步进都需要消耗大量的运算。例如,Liu 算法中每次灰度转换要对灰度区间进行划分,再逐个区间进行判定,平均需要"区间数/2 ×8"次运算,至少也需要平均约 3 ×8 次的运算量(对 256 级灰度)。其次,Chung-Tsai 方法的主要优势是能够在每次循环时利用拷贝得到多点灰度,但这种灰度拷贝并没有考虑灰度值的初始累积误差,实验表明,该方法对多灰度级反走样绘制并不能理想地工作。此外,该算法需要较多的初始化工作(乘除法),算法也非常复杂,且需要较多的附加运算,如比较、检索像素位置等,其总体运算量较大。因此,本文算法的简单性和效率远优于上述方法。

仅从运算次数上考虑,Wu 算法的每次灰度转换也需要约 5 次基本运算,但 Wu 算法的实现依赖一次浮点数除法和定点小数运算,而算法 1 和算法 2 分别仅需要 1 次和 2 次整数除法,所有运算都是整数运算,硬件实现更为简单。

在反走样质量上,新算法的灰度是精确值而非估计值,其效果与直接计算距离后再转换为灰度的浮点算法完全相同。图 6 是一些典型直线的绘制示例,其反走样效果良好。

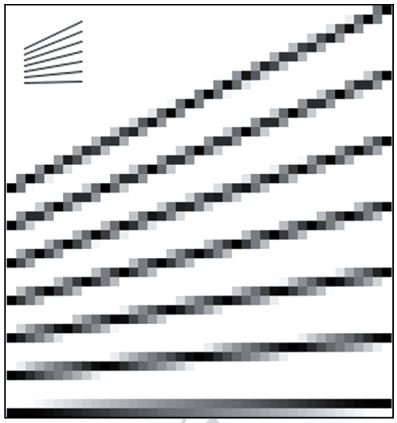


图 6　本文算法的反走样直线绘制示例

Fig. 6　Demonstration of anti-aliased line drawn by our algorithm

4　结　论

提出了一个基于 Wu 两点模式的纯整数反走样光栅直线的生成算法,其核心是一种利用灰度分解和迭代取代现存的距离误差迭代技术。由于该算法无须计算像素与直线之间的距离,而是直接利用像素灰度自身的循环特性控制迭代过程,并依据对余量的累加进行灰度校正,从而避免了耗时的灰度转换操作。然而,由于算法所得到的灰度值是精确的而非粗略的估计值,这使该算法的反走样质量不逊于任何其他两点模式的反走样算法,但其简单性与 Bresenham 基本生成算法接近,利用硬件或软件均很容易实现。此外,所提出的直接利用灰度控制迭代的方法对曲线绘制也有较好的借鉴意义。

参考文献(References)

[1] Crow F C. The aliasing problem in computer generated shaded images [J]. Communications of the ACM, 1977, 20(11): 799-805.

[2] Bresenham J E. Algorithm for computer control of a digital plotter

[J]. IBM System Journal, 1965, 4(1): 25-30.

[3] Foley J D, Dam A V, Feiner S K, et al. Computer Graphics: Principles and Practice [M]. Reading, MA: Addison-Wesley, 1990.

[4] Boyer V, Bourdin J J. Fast lines: a span by span method [J]. Computer Graphics Forum, 1999, 18(3): 377-384.

[5] Boyer V, Bourdin J. Auto-adaptive step straight-line algorithm [J]. IEEE Computer Graphics and Applications, 2000, 20(5): 67-69.

[6] Lin X H, Zhang T W. An adaptive multi-slice line drawing algorithm [J]. Journal of Computer-Aided Design & Computer Graphics, 2006, 18(8): 1136-1141. [蔺想红, 张田文. 自适应多基元直线绘制算法[J]. 计算机辅助设计与图形学学报, 2006, 18(8):1136-1141.]

[7] Jia Y L, Zhang H C, Jing Y Z. Integral algorithm for circle anti-aliasing [J]. Journal of Image and Graphics, 2012, 17(1): 130-136. [贾银亮,张焕春,经亚枝. 圆的整数反走样生成算法[J]. 中国图象图形学报, 2012, 17(1):130-136.]

[8] Leler W J. Human vision, anti-aliasing, and the cheap 4000 line display [J]. Computer Graphics, 1980, 14(3): 308-313.

[9] Field D. Two algorithms for drawing anti-aliased lines [C]// Proceedings of National Computer Graphics Association of Canada Conference-Graphics Interface'84. Toronto S: MacKay, 1984: 87-95.

[10] Fujimoto A, Iwata K. Jag-free images on raster displays [J]. IEEE Computer Graphics and Applications, 1983, 3(9): 26-34.

[11] Pitteway M L V, Watkinson D J. Bresenham's algorithm with grey scale [J]. Communications of the ACM, 1980, 23(11): 625-626.

[12] Waller M D, Ewins J P, White M, et al. Efficient coverage mask generation for antialiasing [J]. IEEE Computer Graphics and Applications, 2000, 20(6): 86-93.

[13] Wolberg G. Sampling, Reconstruction, and Antialiasing [M]. The Computer Science and Engineering Handbook. Boca Raton: CRC Press, 1997.

[14] Gupta S, Sproull R. Filtering edges for gray-scale displays [J]. Computer Graphics, 1981, 15(3): 1-5.

[15] Schilling A. A new simple and efficient antialiasing with subpixel masks [J]. Computer Graphics, 1991, 25(4): 133-141.

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

[17] Liu Y. An all-integer algorithm for drawing anti-aliased straight lines [J]. Computer Graphics Forum, 1994, 13(4): 219-221.

[18] Chung K L, Tsai Y H. On drawing anti-aliased straight lines [J]. Journal of Information Science and Engineering, 1999, 15(6): 833-844.

[19] Jacek Lebiedź. Simple algorithm for antialiased scan conversion of straight line segments. [EB/OL]. (2001-06-23) [2010-10-20]. http://wscg.zcu.cz/WSCG1999/wscg_99_program_short.htm.

[20] Gill G W. N-step incremental straight-line algorithms [J]. IEEE Computer Graphics and Applications, 1994, 14 (3): 66-72.

[21] Niu L Q, Shao Z. A fast straight line rasterization algorithm based on model decomposition [J]. Journal of Computer-Aided Design & Computer Graphics, 2010, 22(8): 1286-1292. [牛连强,邵中. 基于模式分解的快速直线生成算法[J]. 计算机辅助设计与图形学学报, 2010, 22(8): 1286-1292.]