基于像素链排序的直线绘制算法 *

朱晓林[a]，蔡 勇[a]，张建生[b]
（西南科技大学 a. 计算机科学与技术学院；b. 制造科学与工程学院，四川 绵阳 621010）

摘 要：针对直线生成算法在直线斜率大于 0.5 时的低效率问题，提出一种基于像素链排序的直线绘制算法。将直线看做是由许多条平行像素链或对角像素链拼接而成，利用逆向生成直线的类 Bresenham 算法求得各像素链的长度，通过 Bresenham 算法生成相应直线的位移码对各像素链进行排序，一次判断生成一条像素链。仿真实验表明，基于像素链排序的直线绘制算法生成的直线与 Bresenham 算法生成的直线精度一致，且计算量显著减少。该算法只有加法和乘法两种整数运算，适合硬件实现，其绘制速度是 Bresenham 算法的 4 倍。

关键词：计算机图形学；直线绘制算法；Bresenham 算法；逆向生成直线；像素链；排序；整数运算
中图分类号：TP391.41 **文献标志码**：A **文章编号**：1001-3695(2012)03-1175-03
doi:10.3969/j.issn.1001-3695.2012.03.104

Line drawing algorithm based on sorting pixel chains

ZHU Xiao-lin[a], CAI Yong[a], ZHANG Jian-sheng[b]
(*a. School of Computer Science & Technology, b. School of Manufacturing Science & Engineering, Southwest University of Science & Technology, Mianyang Sichuan* 621010, *China*)

Abstract: In order to increase the low efficiency of the line drawing algorithm when the slope of the line is at 0.5 ~ 1, this paper proposed a line drawing algorithm based on sorting pixel chains. It treated a straight line as an aggregation of several horizontal pixel chains or diagonal ones. The attribute of a line and an algorithm of line drawing in a reverse direction, which was similar to the Bresenham algorithm, were used to calculate the length of all the pixel chains. The algorithm sorted the pixel chains by the codes generated by Bresenham algorithm. It generated one pixel chains by one judgment. Simulation results show that the accuracy of straight line generated by new algorithm is as same as that generated by the Bresenham algorithm, and the computation is greatly reduced. The new algorithm only has two integer arithmetic, addition and multiplication, so it is suitable for hardware implementation.

Key words: computer graphics; line drawing algorithm; Bresenham algorithm; line generation in reverse direction; pixel chains; sorting; integer arithmetic

0 引言

直线是最基本的图形元素，是构成复杂图形的基础。目前人们所采用的显示器多为光栅扫描显示器，所以对直线绘制算法的研究也主要集中在选择距离实际直线最近的光栅点的整数算法上。最著名的直线绘制方法是 20 世纪 60 年代中期出现的 Bresenham 算法[1]中，所有的运算都是整数运算，绘制一个点的运算量是一次整数加法运算和一次符号判断。Bresenham 算法非常适合于硬件实现。21 世纪初，出现了一次有效的特征判断自适应地确定生成的像素数目，该算法[2]一次判断可生成相应像素行上的所有像素点，其效率明显取决于像素行上像素点的数目。对于斜率在[0.5,1]之间的直线，在大部分像素行上只有一个像素，由于代码的复杂性，其绘制速度并不比 Bresenham 算法快。文献[3,4]提出了基于对角行程的直线生成算法，将斜率在(0.5,1]之间的直线分为两类进行判别，提高了直线生成的效率，但增加了算法的冗余。文献[5]通过对 Bresenham 算法的分析，提出了逆向生成直线的类 Bresenham 算法，将求斜率在(0.5,1]之间直线的位移码转换为求斜率在[0,0.5]之间直线的位移码的算法，提高直线绘制速度，同样增加了算法的冗余。本文将直线看做是由许多条平行像素链或对角像素链拼接而成，利用文献[5]提出的逆向生成直线的类 Bresenham 算法求得各像素链的长度，通过 Bresenham 算法生成的相应直线的位移码将各像素链进行排序，一次判断生成一条像素链。该算法只有加法和乘法两种整数运算，适合硬件实现。

1 算法理论基础

在数学上，理想的直线是由无限个点组成的集合，考虑到光栅图形输出设备的应用，绘制直线的任务就等价于确定二维像素网格中像素点的位置。不失一般性，本文仅讨论斜率在[0,1]之间的直线。为方便讨论，设直线的起点为(0,0)，终点为(x_0,y_0)，$x_0 \geqslant 0$，$y_0 \geqslant 0$；而对一般的情况，则利用平移变换容易求得。将斜率在[0,1]之间的直线进一步分为两大类：斜率在 0 ~ 0.5 和在 0.5 ~ 1 之间的直线。记 $dx = x_0$，$dy = y_0$（第一类直线）或 $dy = x_0 - y_0$（第二类直线）决策参数为 D，初始决策参数为 $D_0 = 2dy - dx$，决策参数增量分别为 $incrD_1 = 2dy - 2dx$

收稿日期：2011-07-11；**修回日期**：2011-08-15 **基金项目**：国家自然科学基金资助项目(10576027)

作者简介：朱晓林(1984-)，男，山东济宁人，硕士研究生，主要研究方向为计算机图形学、计算可视化(linmy24@sina.com)；蔡勇(1962-)，男，教授，博士，主要研究方向为计算机图形图像处理、虚拟现实技术；张建生(1980-)，男，讲师，硕士，主要研究方向为逆向工程、计算可视化.

和incr$D_2 = 2\mathrm{d}y$。

Bresenham 算法生成的直线在除起始和终止两像素行外，其他各像素行的像素点数个数 e 满足

$$Q \leqslant e \leqslant Q+1 \tag{1}$$

其中：Q 为（x_0/y_0）的下取整（“/”表示除法）。文献[2]已证明。

在利用 Bresenham 算法生成直线时，主要关注的是纵向坐标 y 的变化。图 1 中，在直线生成过程中横向坐标 x 的增量为 1；纵向坐标要么为 0，要么为 1，这些值构成直线的位移码。一串长度为 len 的“0”位移码，也称为长度为 len 的水平像素链，水平像素链用 mode = 0 表示；一串长度为 len 的“1”位移码表示长度为 len 的对角像素链，对角像素链用 mode = 1 表示。这类编码可对直线进行精确描述，为探索直线之间的关系提供了一种方法。

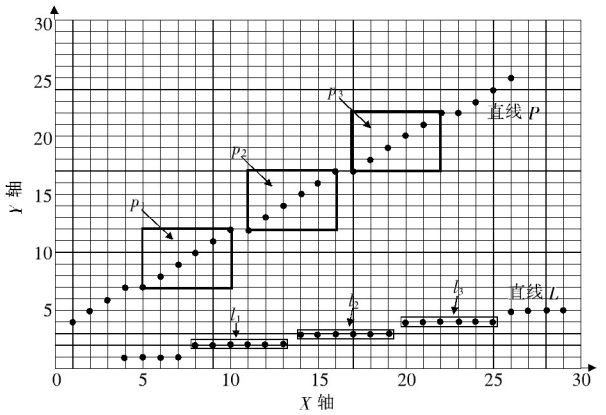


图 1　Bresenham 算法生成直线图例

2　像素链长度计算

在应用 Bresenham 算法生成直线时，生成方向是正向的，即 x 由小向大生成直线。根据 Bresenham 算法，可以推导出逆向类 Bresenham 直线生成算法[5]，即 x 由大向小生成直线。为了叙述方便，该算法暂称为逆向算法。

文献[5]算法的实现证明了，通过一次有效的特征判断自适应地确定生成的像素数目的直线生成算法，也适用于逆向算法。

由第一部分可知，直线可由多条像素链拼接而成，对于正方向生成的第一条像素链（逆方向最后一条），长度用 first_line 表示，最后一条像素链（逆方向第一条）长度用 last_line 表示。

引理 1　对于第一类直线（斜率为 0 ~ 0.5），有

$$\text{first\_line} \leqslant \text{last\_line} \leqslant \text{first\_line}+1 \tag{2}$$

对于第二类直线（斜率为 0.5 ~ 1），有

$$\text{first\_line}-1 \leqslant \text{last\_line} \leqslant \text{first\_line} \tag{3}$$

证明　对于第一类直线，第一条像素链最后一个像素点的决策参数为

$$D = 2\mathrm{d}y - \mathrm{d}x + 2\mathrm{d}y \times s(\text{first\_line}-1) \geqslant 0 \tag{4}$$

由其逆向算法，最后一条像素链最后一个像素点的决策参数为

$$D' = 2\mathrm{d}y - \mathrm{d}x + 2\mathrm{d}y \times (\text{last\_line}-1) > 0 \tag{5}$$

若式(4)中等号成立，比较式(4)和(5)得：

$$\text{last\_line} = \text{first\_line}+1 \tag{6}$$

若式(4)中等号不成立，可得

$$\text{last\_line} = \text{first\_line} \tag{7}$$

即式(2)成立。同理可得式(3)成立，引理得证。

根据引理 1，下面给出如何利用 first_line 判断 last_line。设 first_line = n，对于第一类直线，若 $2\mathrm{d}y \times n - \mathrm{d}x \leqslant 0$，

$$\text{last\_line} = n+1 \tag{8}$$

否则

$$\text{last\_line} = n \tag{9}$$

对于第二类直线，若 $2\mathrm{d}y \times (n-1) - \mathrm{d}x \geqslant 0$，

$$\text{last\_line} = n-1 \tag{10}$$

否则

$$\text{last\_line} = n \tag{11}$$

下面可求得 Q 值。

由式(1)及 $Q = \mathrm{d}x/\mathrm{d}y$ 可知：

$$Q \times \mathrm{d}y \leqslant \mathrm{d}x < (Q+1) \times \mathrm{d}y \tag{12}$$

对于第一类直线有

$$(\text{last\_line}-1) \times 2\mathrm{d}y \leqslant \mathrm{d}x < \text{last\_line} \times 2\mathrm{d}y \tag{13}$$

对比式(12)和(13)可得

$$Q = 2 \times \text{last\_line} - 2 \text{ 或者 } Q = 2 \times \text{last\_line} - 1$$

若$(2 \times \text{last_line}-1) \times \mathrm{d}y \leqslant \mathrm{d}x$，

$$Q = 2 \times \text{last\_line} - 1 \tag{14}$$

否则

$$Q = 2 \times \text{last\_line} - 2 \tag{15}$$

对于第二类直线有

$$(\text{first\_line}-1) \times 2\mathrm{d}y \leqslant \mathrm{d}x < \text{first\_line} \times 2\mathrm{d}y \tag{16}$$

对比式(12)和(16)可得

$$Q = 2 \times \text{first\_line} - 2 \text{ 或者 } Q = 2 \times \text{first\_line} - 1$$

若$(2 \times \text{first_line}-1) \times \mathrm{d}y \leqslant \mathrm{d}x$，

$$Q = 2 \times \text{first\_line} - 1 \tag{17}$$

否则

$$Q = 2 \times \text{first\_line} - 2 \tag{18}$$

由以上分析可知，已知第一条像素链长度可求最后一条像素链长度，进而求得 Q 值。一条直线共有 $\mathrm{d}x+1$ 个像素点，$\mathrm{d}y+1$ 条像素链，除去第一条和最后一条的情况下像素点个数为

$$Q_{\text{remain}} = \mathrm{d}x + 1 - \text{first\_line} - \text{last\_line} \tag{19}$$

像素链条数为 $\mathrm{d}y-1$，则像素链长度为 $Q+1$ 的条数为

$$Q_{\text{num\_long}} = \mathrm{d}x + 1 - \text{first\_line} - \text{last\_line} - (\mathrm{d}y-1) \times Q \tag{20}$$

像素链长度为 Q 的条数为

$$Q_{\text{num\_short}} = \mathrm{d}y - 1 - Q_{\text{num\_long}} \tag{21}$$

例 1　求第二类直线所有像素链的长度，目标直线起始点为(0,0)，终止点为(30,23)。

变量初始化，$\mathrm{d}x = 30$，$\mathrm{d}y = 7$，mode = 1；Bresenham 算法求得第一条像素链长度为 first_line = 3；利用表 3 求得 last_line = 3，从而求得 $Q = 4$；由式(16)和(17)得到 $Q_{\text{num\_long}} = 1$，$Q_{\text{num\_short}} = 5$，得到各像素链长度。该直线由两条长度为 3 的像素链、五条长度为 4 的像素链和一条长度为 5 的像素链拼接而成。

综上所述，已知直线的第一条像素链长度，可通过两次判断和两次计算求得各像素链的长度。

3　算法的实现

在已知所有像素链长度的情况下，可通过将其排序的方法生成直线。直线的第一条和最后一条像素链位置已确定，余下的像素链只有 $Q_{\text{num\_short}}$ 个长度为 Q 的像素链和 $Q_{\text{num\_long}}$ 个长度

为 $Q+1$ 的像素链。将长度为 Q 的像素链编码为“0”,长度为 $Q+1$ 的像素链编码为“1”,问题转换为对 Q_{num_short} 个“0”和 Q_{num_long} 个“1”如何编码的问题。

在直线生成过程中横向坐标 x 的增量为 1,纵向坐标要么为 0,要么为 1,这些值构成直线的位移码,故可以用 Bresenham 算法对上述编码排序。当位移码为 0 时,生成长度为 Q 的像素链;当位移码为 1 时,生成长度为 $Q+1$ 的像素链。该算法具体步骤如下:

a)根据直线类型初始化变量 dx、dy、mode。

b)若像素链条数 $dy+1\leqslant 3$,当 $dy=0$ 时,直线共有一条像素链,直接生成;当 $dy=1$ 时,直线有两条像素链,利用该直线类型算法求得 first_line,从而求得 last_line = dx + 1 − first_line;当 $dy=2$ 时,直线有三条像素链,利用该直线类型算法求得 first_line,进而判断得到 last_line,从而生成直线。

c)当像素链条数 $dy+1>3$,可通过两次判断和两次计算求得各像素链的长度及 Q_{num_long}、Q_{num_short};然后利用 Bresenham 算法生成起点为(0,0)终点为($Q_{num\_short}+Q_{num\_long}$,$Q_{num\_long}$)直线的位移码串,利用生成的位移码与直线像素链长度编码之间的关系生成直线。

例 2　像素链排序算法绘制直线,目标直线起始点为(0,0),终止点为(30,11)。

由第 2 章可知与该直线有关的参数为

$$first\_line=2,\ last\_line=2,\ Q=3$$

$$Q_{num\_long}=3,\ Q_{num\_short}=5$$

Bresenham 算法求得起点为(0,0)终点为($Q_{num\_short}+Q_{num\_long}$,$Q_{num\_long}$)直线的位移码串为“01010010”。利用该位移码串生成直线如图 2 所示。

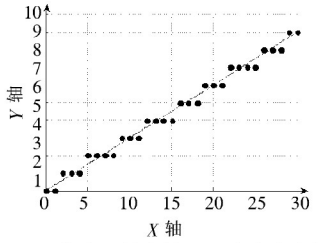
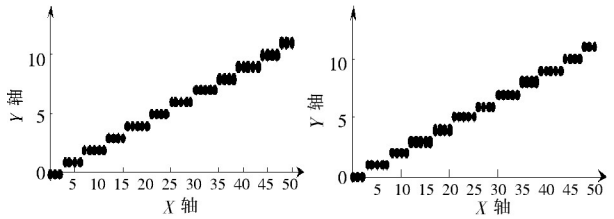


图 2　像素链排序算法生成直线实例

4　仿真结果与分析

基于像素链排序算法生成的直线与 Bresenham 算法生成的直线精度相当。图 3 是像素链排序与 Bresenham 算法生成的第一类直线的精度比较,直线终止点坐标为(50,11);图 4 是像素链排序与 Bresenham 算法生成的第二类直线的精度比较,直线终止点坐标为(50,31)。

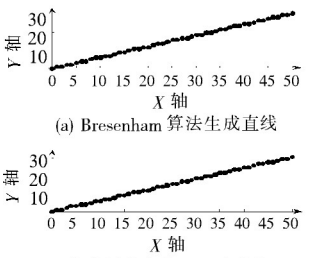


(a) Bresenham 算法生成直线　　(b) 像素链排序算法生成直线

图 3　像素链排序算法与 Bresenham 算法精度比较 1

由图 3 和 4 中生成两类直线的比较可知,本文算法与 Bresenham 算法生成直线均为不对称直线。图 3 和 4 中两种算法生成的直线不同点个数为 3 和 2,极端情况下两种算法生成的直线不同点个数不超过 $1+Q_{num\_short}/2$ 或 $1+Q_{num\_long}/2$,故可认为两种算法精度一致。

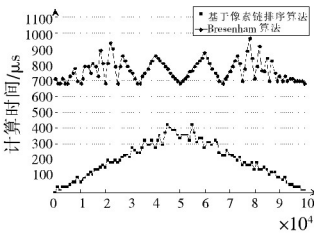
图 5 和表 1 为本文算法与 Bresenham 算法的效率比较。为方便比较,所计算的目标直线起始点均为(0,0),其终止点横坐标为 100 000,终止点纵坐标为图 5 中 x 轴坐标。图中纵坐标为计算像素链位置时间,单位为 μs,不包括像素点点亮时间,均为 Intel® Core™ 2 Quad CPU Q9400 2.66 GHz,3.25 GB 内存的计算机上用 MATLAB 7.0 计算所得。



(a) Bresenham 算法生成直线

(b) 像素链排序算法生成直线

图 4　像素链排序算法与 Bresenham 算法精度比较 2



目标直线终止点的 y 坐标

图 5　基于像素链排序算法与 Bresenham 算法效率比较

表 1　本文算法与 Bresenham 效率比较

目标直线	Bresenham/μs	本文算法/μs
(100000,9)	781	47
(100000,10008)	750	78
(100000,20007)	750	156
(100000,30006)	859	235
(100000,40005)	953	297
(100000,50004)	750	360
(100000,60003)	954	296
(100000,70002)	859	219
(100000,80001)	766	141
(100000,90000)	750	78
(100000,99999)	750	218
平均值	811	193

由图 5 和表 1 的仿真结果可知,基于像素链排序算法的效率约为 Bresenham 算法的 4 倍,显著提高了直线生成的速度。

5　结束语

根据直线的性质求得了直线各像素链的长度,利用 Bresenham 算法生成的位移码对各像素链进行排序,提高了直线绘制算法的速度。文中提出的像素链排序算法可应用于编码、密码学等其他领域,为信号编码提供了新的思路。

参考文献:

[1] BRESENHAM J E. Algorithm for computer control of digital plotter [J]. IBM Systems Journal,1965,4(1):25-30.

[2] 贾银亮. Bresenham 直线生成算法的改进[J]. 中国图象图形学报,2008,13(1):158-161.

[3] 叶晓彤,邓云. 基于对角行程的直线生成算法研究[J]. 计算机应用,2008,28(9):2270-2273.

[4] NIU Lian-qiang, FENG Hai-wen. A line segments approximation algorithm of grating lines[C]//Proc of International Forum on Computer Science-Technology and Applications. Washington DC: IEEE Computer Society,2009:34-37.

[5] 朱晓林,蔡勇. 基于像素链的直线绘制算法[J]. 计算机应用,2011,31(4):1057-1061.