对称式八步直线生成算法

欧阳开翠[1] 曾令华[2] 谭 渊[1] 白宝钢[2]
(温州大学瓯江学院 温州 325035)[1] (温州大学计算机科学与工程学院 温州 325035)[2]

摘 要 分析了直线生成模式与直线斜率之间的关系,提出了一种八步增量算法。该算法一次能画四个像素,结合直线的对称性,在一次循环中可以画八个像素。该算法只用到了整数加法运算、减法运算和左移位运算,大大降低了硬件实现的复杂度,同时有效地提高了速度,易于硬件实现。

关键词 Bresenham算法,直线生成扫描转换算法,八步法,对称

Eight-step Line-generating Algorithm Based on Symmetry

OUYANG Kai-Cui[1] ZENG Ling-Hua[2] TANG Yuan[1] BAI Bao-Gang[2]
(Qujiang College, Wenzhou University, Wenzhou 325035)[1]
(School of Computer Science & Engineering, Wenzhou University, Wenzhou 325035)[2]

Abstract In this paper, we present a new line drawing algorithm named eight-step incremental generation of lines by analyzing the relationship between generation models of line and the linear slope. The algorithm draws four pixels on the same time, and it can draw eight pixels as well considering the symmetry of line generation. First, the algorithm gives the standard of the pixel choosing that the most closed to the straight line, and then convert to the integral variable form. As only addition, subtraction and left shift operation of integer are used in this algorithm, the speed has been improved greatly, and it is easy to be implemented by hardware.

Keywords Bresenham algorithm, Line scan conversion algorithm, Eight-step generating algorithm, Symmetry

1 引言

直线是光栅图形学中最基本的元素之一,直线生成算法的效率和质量直接影响着图形应用系统的效率和质量。为了提高直线的绘制速度,直到最近几年,仍然有大量的研究工作出现。这些工作集中在如何一次生成位于直线上尽可能多的像素[2~11]。

目前被广泛使用的直线生成是1965年出现的Bresenham算法[1],它是一种增量算法,只需要根据误差项的符号,即可得到所求像素。文[2]提出了一个对Bresenham算法进行改进而来的双步增量直线生成算法,在最好的情况下,对误差的一次判断操作可确定两个像素的走法,但在最差情况下,需要进行三次判断才可以确定两个像素,其速度比Bresenham算法大约提高了24%。文[6]提出一个对称的快速直线生成算法,该算法基于线段的中心对称性,使用Bresenham算法从线段的两端同时向中心生成直线,每次可确定两个像素点,其速度相对于Bresenham算法提高了33%左右。文[13]结合文[2]与文[6]的算法,一次可以生成四个像素点。在绘制相同数量直线段的条件下,该算法所用时间仅仅略微超出Bresenham算法所用时间的五分之二。文[11]给出一种依据直线斜率,快速生成直线的并行Bresenham直线算法。这种算法在直线段接近于坐标轴时,有着很高的直线生成效率。但随着斜率的增大,其效率会显著下降。

以上各种直线生成算法的特点都是一次生成尽可能的像素的数目。但是,若增加每次生成像素的数量,则会由于pattern数量过多,造成对误差值判断操作次数增加,致使直线生成的效率反而会下降。

本文依据斜率把直线段分为六种类型,由于受斜率的约束,使得每类直线可以使用的生成模式的数量要比直接仿照文[2]方法生成直线时的模式数目少很多,因而在不明显增加对误差值判断次数的基础上,每次可生成更多的像素。同时,再利用文[6]的对称算法,一次可生成多达八个像素,速度相对Bresenham算法提高了62%左右。

为方便讨论,本文假设线段的起点为$S(xs,ys)$,终点为$E(xe,ye)$,$\Delta x=xe-xs>0$,$\Delta y=ye-ys\geqslant 0$,$\Delta x\geqslant\Delta y$,斜率$m=\Delta y/\Delta x$,且$m\in[0,1]$。对于其它情况,在不降低算法效率的情况下,只要对算法做适当变换即可。

2 对称八步直线生成算法

2.1 生成模式

根据Bresenham直线生成算法的基本思想,生成的下一个像素与当前像素之间存在两种关系——沿轴向或沿斜向。如图1所示,当前像素点为P,则下一个像素点只能选择A(轴向)或者B(斜向)。

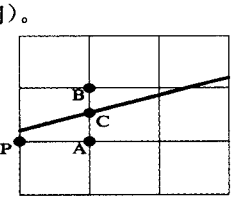

图1 A、B为待选像素点,C为理论计算点

欧阳开翠 讲师,主要研究方向:CAGD、CAD等;曾令华 讲师,主要研究方向:CAGD、CAD、网络与信息系统安全、中间件技术等;谭 渊 讲师,主要研究方向:现代教育技术、CAD;白宝钢 硕士,副教授,主要研究方向:CAGD、CAD、计算机动画等。

定义 1 在生成直线时，每一种由沿轴向移动和斜向移动构成的组合形式称为一个生成模式(pattern)。

由 n 个像素构成的生成模式可以用 $p_0 p_1 \cdots p_{n-1}$ 表示。其中：

$$\begin{cases} p_i=0, \text{若第 } i \text{ 像素沿轴向选择} \\ p_i=1, \text{若第 } i \text{ 像素沿斜向选择} \end{cases} \quad i=0,1,\cdots,n-1$$

例如：若一次生成四个像素点，则其对应的 16 种生成模式可分别表示如下：

0000,0001,0010,0011,0100,0101,0110,0111

1000,1001,1010,1011,1100,1101,1110,1111

由于生成模式中为 1 的位，表示了沿斜向的移动，因而生成模式中 1 的个数即是在选择该生成模式生成直线时，生成模式中的最末像素沿 Y 方向相对于起始点偏移的像素个数 $deltay$。

$$deltay=\sum_{i=0}^{n-1} p_i$$

2.2 斜率与生成模式之间的关系

设直线的生成模式由 n 个像素构成，有一递增数列 $\{m_i \mid m_i=b/a,\ 0\leqslant b<a\leqslant n \text{ 且 } a,b\in Z\}$，则可以使用该数列，依据斜率对直线进行划分。

设直线斜率为 m，且满足条件

$m_i<m\leqslant m_{i+1}$。其中 $m_i=b_0/a_0$，$m_{i+1}=b_1/a_1$。

这表明与该直线对应的任意连续的 a_0 个像素，沿斜向的个数不少于 b_0 个。同样，任意连续的 a_1 个像素中，最多只能有 b_1 个沿斜向。因此，在使用 n 个像素构成的生成模式来生成该直线时，可供选择的生成模式必须满足以下条件：

(1)任意连续的 a_0 个像素中，至少包含 b_0 个斜向像素，连续的轴向像素的个数不超过 (a_0-b_0)；

(2)任意连续的 a_1 个像素中，最多包含 b_1 个斜向像素，且连续的斜向像素的个数不超过 b_1 个。

(3)在将多个直线生成模式进行组合连接时，也必须遵循条件(1)和(2)。

若使用 4 个像素的生成模式来生成直线，即 $n=4$，按照上述方法，可以将直线依据其斜率划分为六类，那么可供选择的生成模式集、每一种被选取的生成模式后可以连接的生成模式与这些直线的斜率范围的关系如表 1 所示。从表中我们发现：

(1)直线的斜率不同，其可选模式构成的模式集也不同；

(2)每类直线可供选择的生成模式集中均包含五种生成模式；

(3)实际上可用于生成直线的四像素的生成模式只有 14 种，生成模式 0011 和 1100 是不会出现在任何直线的可选生成模式集之中的。

(4)大部分模式后允许连接的模式数目少于五个，据此可以简化判定树，从而可以加快程序的执行。

表 1 直线斜率范围与其可选模式集和可连接模式之间的关系

直线类别	斜率范围	模式编号	0	1	2	3	4
0	m≤1/4	可选模式集	0000	0001	0010	0100	1000
		可连接模式	0,1,2,3,4	0,1	0,1,2	0,1,2,3	0,1,2,3,4
1	1/4<m≤1/3	可选模式集	0001	0010	0100	1000	1001
		可连接模式	0,1	1,2	2,3,4	3,4	0,1
2	1/3<m≤1/2	可选模式集	0010	0100	0101	1001	1010
		可连接模式	1,2,3,4	3,4	0,1,2	0,1,2	1,2,3,4
3	1/2<m≤2/3	可选模式集	0101	0110	1010	1011	1101
		可连接模式	0,1,2,3	2,3,4	2,3,4	0,1	0,1,2,3
4	2/3<m≤3/4	可选模式集	0110	0111	1011	1101	1110
		可连接模式	3,4	0,1	0,1,2	2,3	3,4
5	3/4<m≤1	可选模式集	0111	1011	1101	1110	1111
		可连接模式	0,1,2,3,4	1,2,3,4	2,3,4	3,4	0,1,2,3,4

2.3 误差计算及生成模式的选择

直线的斜率决定了生成该直线时可供选择的生成模式集，而从生成模式集中具体选择哪一个生成模式，则要由误差的大小来判定。

例如，使用 4 像素的生成模式来生成直线，则被选择的生成模式 $p_0\ p_1\ p_2\ p_3$ 需要满足如下的约束条件：

$$\begin{cases} p_0+p_1+p_2+p_3+0.5>y_{i+4}-y_{ir}\geqslant p_0+p_1+p_2+0.5 \\ p_0+p_1+p_2+0.5>y_{i+3}-y_{ir}\geqslant p_0+p_1+0.5 \\ p_0+p_1+0.5>y_{i+2}-y_{ir}\geqslant p_0+0.5 \\ p_0+0.5>y_{i+1}-y_{ir} \end{cases}$$

由此可计算得到如表 2 所示的选取直线生成模式的判定表。该表指出了在每一类直线中被选择模式与误差取值之间的关系，其中直线和模式的类型编号见表 1。

在选择某一生成模式生成直线后，可以仿照 Bresenham 算法重新计算出误差：

$$\varepsilon(x_{i+4})=y_{i+4}-y_{ir}-0.5$$

$$\varepsilon(x_{i+8})=y_{i+8}-y_{i+4,r}-0.5=y_{i+4}-y_{i+4,r}+4m-0.5=\varepsilon(x_{i+4})+4m-deltay \quad (1)$$

2.4 基于对称技术的八步直线生成算法

线段具有中点对称性。如图 2 所示，EM 与 SM 关于中点 M 对称。当计算出线段 SM 上点 A 时，根据直线的对称性，马上就可以得到与之对称的 B 点，即 B 点是不用计算的，因而可以把计算量缩减为原来的一半[6]。

当我们使用 4 像素生成模式来生成直线时，结合直线的中点对称性，分别从线段的两端沿直线向中点方向对称生成直线，一次计算即可生成八个像素点，从而可以极大地提高直线生成算法的效率。

定义 2 从始端向中点方向生成直线所使用的直线生成模式称为始端生成模式。从线段终端向中点方向生成直线所使用的直线生成模式称为终端生成模式。

由于直线的中心对称性，其始端生成模式和终端生成模式也相互对称。只要把从始点 S 向中点 M 方向生成直线时

使用的始端生成模式旋转半周，即可得到沿终端 E 向中点 M 方向生成直线对应的终端生成模式。

表 2 直线生成模式判定表

直线类型	模式 0	模式 1	模式 2	模式 3	模式 4
0'	$\varepsilon(x_{i+4})<0$	$0\leqslant\varepsilon(x_{i+4})<m$	$m\leqslant\varepsilon(x_{i+4})<2m$	$2m\leqslant\varepsilon(x_{i+4})<3m$	$3m\leqslant\varepsilon(x_{i+4})<1$
1	$0\leqslant\varepsilon(x_{i+4})<m$	$m\leqslant\varepsilon(x_{i+4})<2m$	$2m\leqslant\varepsilon(x_{i+4})<3m$	$3m\leqslant\varepsilon(x_{i+4})<1$	$1\leqslant\varepsilon(x_{i+4})<m+1$
2	$m\leqslant\varepsilon(x_{i+4})<2m$	$2m\leqslant\varepsilon(x_{i+4})<1$	$1\leqslant\varepsilon(x_{i+4})<3m$	$3m\leqslant\varepsilon(x_{i+4})<m+1$	$m+1\leqslant\varepsilon(x_{i+4})<2m+1$
3	$2m\leqslant\varepsilon(x_{i+4})<m+1$	$m+1\leqslant\varepsilon(x_{i+4})<3m$	$3m\leqslant\varepsilon(x_{i+4})<2$	$2\leqslant\varepsilon(x_{i+4})<2m+1$	$2m+1\leqslant\varepsilon(x_{i+4})<m+2$
4	$m+1\leqslant\varepsilon(x_{i+4})<2$	$2\leqslant\varepsilon(x_{i+4})<3m$	$3m\leqslant\varepsilon(x_{i+4})<2m+1$	$2m+1\leqslant\varepsilon(x_{i+4})<m+2$	$m+2\leqslant\varepsilon(x_{i+4})<3$
5	$2\leqslant\varepsilon(x_{i+4})<3m$	$3m\leqslant\varepsilon(x_{i+4})<2m+1$	$2m+1\leqslant\varepsilon(x_{i+4})<m+2$	$m+2\leqslant\varepsilon(x_{i+4})<3$	$3\leqslant\varepsilon(x_{i+4})$

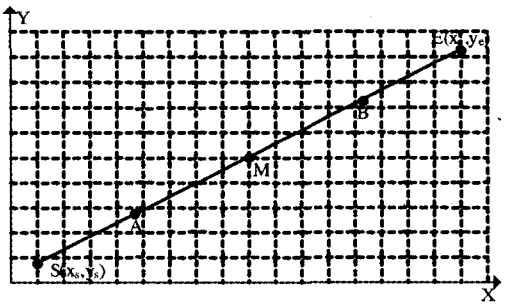

图 2 M 是线段 SE 的中点，A、B 关于 M 对称

算法设计如下：

根据起点坐标(xs,ys)和终点坐标(xe,ye)，计算直线的斜率 m；
根据 m 和表 1，装入对应的始端和终端生成模式集及每种生成模式对应的 Y 坐标方向的位移量 $deltay$；
根据 m 和表 2，计算得到相应的判定树；
设置 $x1=xs$，$y1=ys$，$x2=xe$，$y2=ye$；
计算初始误差：$e=4m-0.5$，迭代次数 $C=(xe-xs)/8$；
在($x1$,$y1$)和($x2$,$y2$)处画点；
while $C>0$
依据 e 的取值和判定树，选择对应的始端生成模式 $p_0p_1p_2p_3$ 和终端生成模式 $p_0p_1p_2p_3$，得到 Y 坐标方向的位移量 $deltay$；
从($x1+1$,$y1+1$)开始，使用始端生成模式绘出四个像素；
从($x2-1$,$y2-1$) 开始，使用终端生成模式绘出四个像素；
依据式(1)修改误差 e，
$y1=y1+deltay$；
$y2=y2-deltay$；
$x1=x1+4$，$x2=x2-4$；$C=C-1$；
LOOP
if $x1+1!=x2$
从 $x1+1$ 开始，使用 Bresenham 算法画出剩余的点；
算法结束.

优化措施：

(1)为避免除法计算，在计算 e、m 时，分别对它们都乘以 $2*(xe-xs)$，从而使得算法中对误差 e 的计算变为整数计算。

(2)设计判定树时，依据表 1 中模式之间的连接关系，可以减少判断的次数。在最优情况下，只需要一次判断，即可点亮 8 个像素。

3 结果分析

从实验结果看，在直线的起点和终点坐标相同时，本文生成的直线与其它算法所生成的直线完全相同。

不失一般性，在 $\Delta x\geqslant\Delta y\geqslant 0$ 条件下来比较各算法效率。设直线斜率在[0,1]之间等概率分布，为方便讨论，对本文给出的对称八步算法的判定树未作优化，则每种算法的运算量如表 3 所示。从表 3 可看出，不论是更新误差的次数还是对误差进行判断的次数，本文提出的对称八步算法都远比其它直线生成算法的效率高得多。

表 4 分别给出了几种直线生成算法在绘制相同数目的直线时所用时间，单位为毫秒。通过重复绘制起点为(0,0)，长度为 500 个像素，斜率为$\{m\mid 0\leqslant m<1$，且 $m_{i+1}-m_i=0.001$，$i\in Z\}$的 1000 条直线来达到所需绘制直线的数目。在每种情况下，每个算法各运行 5 次，取其中最短的运行时间来作为该算法的运行时间。此外，由于各种算法绘制相同直线时所生成的像素数相同，所以表 4 中未包括画点所用时间，以便能更清楚地比较各种算法画线的相对速度。程序运行环境为：Pentium 4-M CPU 1.8GHz，内存 368MB，操作系统 Microsoft Windows XP Professional 2002 Service Pack 2，Microsoft Visual C++ 6.0。

表 3 几种不同直线生成算法运算量比较

比较项目	Bresenham	双步法[2]	对称法[6]	四步法[13]	八步法(本文)
更新误差所用加减运算次数	Δx	$\Delta x/2$	$\Delta x/2$	$\Delta x/4$	$\Delta x/8$
对误差判断总次数	Δx	Δx	$\Delta x/2$	$\Delta x/2$	$3\Delta x/10$
平均画一点对误差判断次数	1	1	0.5	0.5	0.3

从表 4 的结果可看出，本文给出对称八步直线生成算法，由于可以在一次误差的更新操作中画出八个像素点，使得它的直线生成速度远超过其它现有的直线生成算法。

表 4 算法运行时间比较(ms)

直线数目(万)	Bresenham[1]	双步法[2]	对称法[6]	四步法[13]	八步法(本文)
10	300	230	200	140	110
20	611	460	401	280	230
30	912	691	601	420	350
40	1222	921	811	560	470
50	1513	1141	1011	701	580
60	1833	1382	1211	841	701
70	2113	1612	1412	981	821
80	2443	1843	1612	1112	931
90	2734	2073	1812	1262	1041
100	3034	2303	2012	1402	1161

结束语 现有的大部分直线生成算法，较少对直线本身的特性进行考虑，因而直线生成的效率不太高。对于部分考虑直线特性的直线生成算法，也只是当斜率处于某些特殊范围时，才会有较好的生成效率。

本文依据直线斜率对直线进行分类，由于受到直线斜率的约束，每类直线可使用的生成模式集不但小于直线所有可能的生成模式构成的集合。而且，生成模式在连接时，允许的

连接关系也少于所有可能的组合关系。这样，在生成直线时，可以减少大量的判断操作，再结合直线的对称性，可以极大地提高直线生成效率。另外，运用本文提出的方法，可以在一次循环中点亮更多的像素。但是，随着一次生成的像素数量的增长，使用的模式数量也会同步增长，会导致判断树的层次也将加深。因而，直线生成的效率是否会随着一次可生成的像素数量的增多而同步增长，以及究竟每次生成的像素数量为多少时可以使直线的生成效率达到最高等问题，都还有待于人们去探索。

参 考 文 献

1 Bresenham J E. Algorithms for computer control of a digital plotter[J]. IBM Systems Journal, 1965, 4(1):25～30

2 Wu A, Rokne J G. Double-step incremental generation of lines and circles [J]. Computer Vision Graphics and Image Processing, 1987, 37(3): 331～344

3 Bao P, Rokne J. Quadruple-step line generation[J]. Computer & Graphics, 1989, 13(4): 461～469

4 Wyvil B. Symmetric double-step line algorithm[M]. Glassner Andrew S. Graphics Gems Ⅰ. Boston: Academic Press, 1990. 101～104

5 Rokne G, Rao Y. Double-step incremental linear interpolation. ACM Transaction on Graphics, 1992, 11(2): 183～192

6 刘勇奎. 一个对称的快速直线生成算法[J]. 微计算机应用, 1993, 14(2): 42～51

7 Graham P, Sitharama, Lyengar S. Double-and triple-step incremental linear interpolation[J]. IEEE Computer Graphics and Applications, 1994, 20(5): 49～53

8 Hearn D, Baker M P. Computer Graphics[M]. 2nd ed. Beijing: Tsinghua University Press, 1998

9 郑宏珍，赵辉. 改进的 Bresenham 直线生成算法[J]. 中国图象图形学报，1999，4(7)：606～609

10 程锦，陆国栋，谭建荣. 基于直线特性的直线生成集成算法[J]. 中国图象图形学报，2001，6A(4)：392～395

11 孙岩，唐棣. 并行的 Bresenham 直线生成算法. 计算机工程与应用，2001(21)：136～140

12 柳士俊，邓北胜，徐怀刚. 对称扫描四步增量画线算法[J]. 中国图象图形学报，2002，7A(10)：1054～1057

13 林笠，Chen Rong. 基于 Bresenham 算法的四步画直线算法[J]. 暨南大学学报(自然科学版)，2003，24(5)：19～21

(上接第 239 页)

关的三种粗关系：非凸区域与凸精确区域的粗糙关系、非凸区域与凸粗糙区域的粗糙关系、非凸区域与非凸区域的粗关系。由于由非凸区域转换来的粗糙集较为特殊，为了能够讨论更为一般的情况，我们采用文[9,10]中的表示粗糙集的方法来进行讨论，如图 8 所示。这样，我们这里给出的 3 种关系可以由“蛋-黄”模型来表达，所以本文就不再详细叙述这些关系。

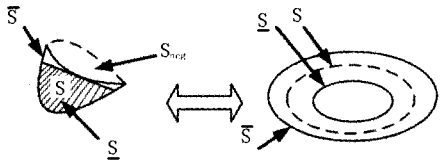

图 8 非凸区域等价关系

结论 图像中的目标空间特征，尤其是关系特征，对目标识别具有非常重要的作用。在实际工程中，提取出来的目标往往是非凸的，现在大量的文献都在讨论凸目标的空间特性，本文从这点出发给出了一种非凸区域转化为凸区域的粗糙近似算法，把一个非凸精确区域转变成一个粗糙区域，从而可以利用已经取得的各种空间拓扑空间关系来讨论非凸区域之间的拓扑关系，这对基于图像的空间定性推理(QSR)进行了推广，使其不但能够对含有凸区域的图像进行语义推理，而且能够对含有非凸区域的图像也能进行推理。

参 考 文 献

1 Randell D A, Cohn A G. Modeling topological and metrical properties in physical processes. In: Proc. 1th Int Conf. on the Principles of Knowledge Representation and Reasoning. Morgan Kaufmann, Los Altos, 1989. 55～66

2 Randell D A, Cui Z, Cohn A G. A spatial logic based on regions and connection. In: Proc. 3rd Int Conf. on Knowledge Representation and Reasoning. Morgan Kaufinann, Sanmateo, 1992. 165～176

3 Cui Z, Cohn A G, Randell D A. Qualitative simulation based on a logical formalism of space and time. In: Proc. ofAAAI-92. AAAI Press, Menlo Park, California, 1992. 679～684

4 Bennett B. Spatial reasoning with propositional logics. In: Proc. of KR-94. Morgan Kaufmann, 1994. 51～62

5 Bennett B, Cohn A G. Mufti-dimensional multi-modal logics as a framework for spatio-temporal reasoning. In: Proc. of the 'Hot Topics in Spatio-Temporal Reasoning' workshop, IJCAI-99. Stockholm, 1999

6 Clarke B L. A calculus of individuals based on 'connection'. Notre Dame Journal of Formal Logic, 1981, 23(3):204～218

7 Egenhofer M J, Franzosa R. Point-set topological spatial relations. Int J of Geographical Information Systems, 1991, 5 (2): 161～174

8 Egenhofer M J, Herring J. Categorizing binary topological relationships between regions, lines and points in geographic database:[Technical Report]. Department of Surveying Engineering, University of maine, 1991

9 王树良，李德仁，史中文，等. 地学粗空间的理论与应用[J]. 武汉大学学报(信息科学版)，2002，27(3)：274～282

10 王新洲，史中文，王树良，等. 模糊空间信息处理[M]. 武汉大学出版社，2003

11 欧阳继红. 时空推理中一些问题的研究[D]:[吉林大学博士学位论文]. 2005

12 邓方安. 关于非凸集的粗糙近似[J]. 苏州科技学院(自然科学版)，2003，20(3)：7～8

13 Cohn A G, Gotts N M. The “Egg-Yolk” representation Of Regions with Indeterminate Boundaries. In Geographic objects with indeterminate boundaries. P Burrpugh, A Frank. GISDATAII, European Science Foundation. Chapter12(1995)

14 Theresa Beaubouef, Fredrick Petry. Vagueness in Spatial Data: Rough Set and Egg-Yolk Approach. Lecture notes in Artificial Intelligence, 2001, 2070: 367～373

15 胡毓达，孟志清. 凸分析与非光滑分析[M]. 上海：上海科学技术出版社，2000