1000-9825/2002/13(12)2275-06 ©2002 Journal of Software 软 件 学 报 Vol.13, No.12

一种快速圆弧绘制算法*

程 锦, 陆国栋, 谭建荣

(浙江大学 CAD&CG 国家重点实验室,浙江 杭州 310027)
E-mail: chengjin_zju@sohu.com
http://www.cad.zju.edu.cn

摘要: 提出一种圆弧绘制算法.与传统的基于单个像素点的圆弧绘制算法不同,新算法每执行一次输出操作均可生成两个或多个像素点.该算法将圆弧离散轨迹看成是由一系列水平位移和对角位移构成,逐段找出并绘制这些位移,从而减少了圆弧绘制过程中所需的输出操作,有效地提高了圆弧绘制速度.实验结果表明,新算法与著名的 Bresenham 算法相比,圆弧绘制速度提高近一倍.进一步地,新算法可以推广到其他二次曲线的绘制中.

关 键 词: 圆弧绘制;Bresenham 算法;离散轨迹;水平位移;对角位移

中图法分类号: TP391 **文献标识码**: A

圆弧是组成图形最基本的元素之一.圆弧生成算法的优劣对整个图形系统的效率和质量有着直接而重要的影响.由于目前人们所采用的显示器多为光栅扫描显示器,所以对圆弧绘制算法的研究也主要集中在选择距离实际圆弧最近的光栅点的整数型算法上.假定圆弧的圆心在坐标原点,圆弧半径为整数,由于圆的对称性,我们只需研究第 2 个八分区中圆弧的绘制,而其他 7 个八分区的圆弧可由对称性获得.Bresenham 算法[1,2]和中点法[3,4]的每步 x 坐标增 1,y 坐标则根据决策参数的符号来决定是否减 1.通过引入决策参数,这两种算法在生成圆弧的过程中,除初始化时要进行乘法运算外,都只需进行简单的整数加减法运算和移位操作,因而具有较快的圆弧绘制速度.Bresenham 法和中点法每次循环只生成一个点,Wu 和 Rokne 提出的两步画圆法[5]则可使每次循环生成两个点.该法每次循环前进两步,利用非参数曲线 $f(x,y)=0$ 在光栅显示器上的离散轨迹所具有的性质,中间一点无须数学运算便可生成.两步法使得绘制圆弧所需的数学运算量减少了一半,但与此同时,生成圆弧所需的逻辑操作数目却大大增加了.实验结果表明,两步法并不能获得比单步法更快的绘制速度.

上述各种圆弧绘制算法每走一步生成一个或两个像素点,即所有像素点都是用画点命令逐个生成并显示的,我们称其为基于单个像素点的圆弧绘制算法.为了进一步提高圆弧绘制速度,Hsu,Chow 和 Liu 等人提出了一个基于水平直线段的圆弧扫描转换算法[6],以下我们将其简称为 HCL 法.该算法基于以下事实:一段位于第 2 个八分区的 45° 圆弧的离散轨迹是由一系列具有相同 y 坐标的相邻像素点所构成的水平直线段组成的.如图 1 所示,为一段位于第 2 个八分区的圆弧,其圆心在坐标原点,半径为 15 个像素单位,圆弧起点坐标为(0,15).HCL 法将该圆弧看成由 5 条水平小直线段 $L_i(i=1,2,\ldots,5)$构成,这些直线段所具有的像素点数目依次为 4、3、2、1、1,该法逐段找出这些水平直线段并用画线命令进行绘制.对于前 3 条水平直线段 L_1,L_2 和 L_3,由于它们所具有的像素点数目不小于 2,故一次画线可以生成至少两个点,确实能够减少输出操作所需的执行时间.但对于 L_4 和 L_5,由于这两条水平直线段已退化为单个像素点,HCL 法却仍采用画线命令进行绘制,其绘制速度显然不如直接用画点命令输出单个像素点.这种退化水平直线段的客观存在决定了该算法并不能使圆弧绘制速度得到明显提

* **收稿日期**: 2001-05-15; **修改日期**: 2001-06-26

基金项目: 国家自然科学基金资助项目(69878038);国家 863 高科技发展计划资助项目(863-511-9842-006);浙江省自然科学基金资助项目(696045)

作者简介: 程锦(1978－),女,浙江永康人,博士生,主要研究领域为计算机辅助设计,计算机图形学;陆国栋(1963－),男,浙江东阳人,博士,教授,主要研究领域为计算机图形学,计算机辅助设计;谭建荣(1954－),男,浙江湖州人,博士,教授,博士生导师,主要研究领域为产品信息建模,计算机辅助设计,计算机图形学.

高.尤其是对于那些半径较小的圆弧,由于其离散轨迹中已退化的水平直线段所占的比例较大,用 HCL 法进行绘制所需的时间往往会超出用 Bresenham 法等基于单个像素点的圆弧绘制算法进行绘制所需的时间.

本文通过对圆弧离散像素点的轨迹进行深入分析和研究,提出了一种快速圆弧绘制算法.该算法充分利用圆弧离散轨迹所具有的特性,先找出离散轨迹中各段水平位移与对角位移,然后用画线命令逐段进行绘制.新算法每执行一次输出操作都可以生成两个或两个以上的像素点,极大地减少了圆弧绘制过程中所需的输出操作,从而显著地提高了圆弧生成速度.

1 水平位移与对角位移

我们将如图 1 所示位于第 2 个八分区的圆弧在光栅显示器上的离散轨迹看成由两条水平直线段 $L_{hi}(i=1,2)$ 和两条斜率为-1 的对角直线段 $L_{dj}(j=1,2)$构成,其中 $L_{h1},L_{h2},L_{d1},L_{d2}$ 所包含的像素点数目分别为 4、2、2、3.因此,我们可以找出这些小直线段,并用画线命令进行绘制,从而每执行一次输出操作都可以生成至少两个像素点.由于在新算法绘制圆弧的过程中将水平方向和对角方向的小直线段作为基元,因此,我们将这两种直线段分别定义为水平位移和对角位移,记为 L_{hi} 和 L_{dj};并将其长度定义为各直线段所具有的像素点数目,分别记为 l_{hi} 和 l_{dj},则对于如图 1 所示的圆弧,我们有 $l_{h1}=4,l_{h2}=2,l_{d1}=2,l_{d2}=3$.容易看出,用新算法绘制圆弧的关键在于决定何时由绘制水平位移转为绘制对角位移,并逐段求出这些水平位移和对角位移及其相应的长度.

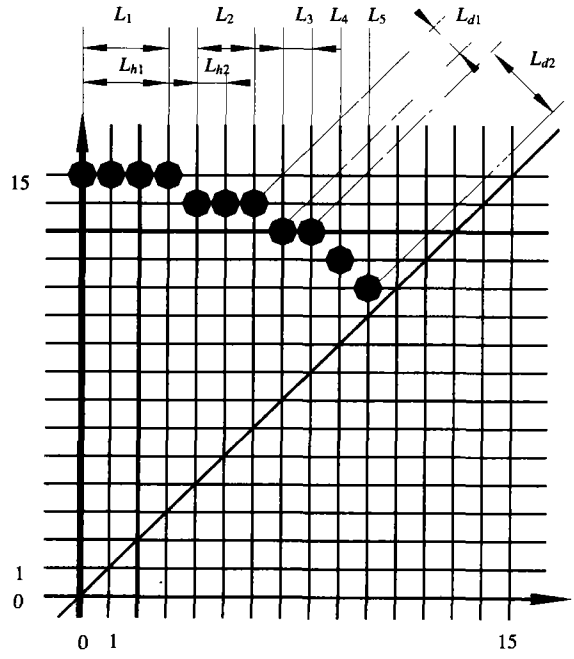

Fig.1 The one eighth of a circle with a radius of 15 pixel units

图 1 半径为 15 个像素单位的 1/8 圆弧

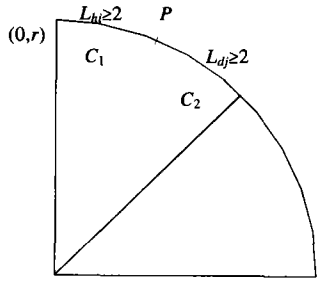

Fig.2 Point P divides one eighth of a circle into two segments

图 2 点 P 将 1/8 圆弧分为两段

引理 1. 设 $P(x,y)$为如图 2 所示圆弧上满足 $y=2x$ 的点,该点将图中上半段位于第 2 个八分区的圆弧分为 C_1 和 C_2 两段,则 C_1 段圆弧上所有点(不包含点 P)的斜率绝对值 abs(SLOPE C_1)<0.5,而 C_2 段圆弧上所有点(不包含点 P)的斜率绝对值 abs(SLOPE C_2)>0.5.

定理 1. 如果水平位移 L_{hi} 完全包含于 C_1,则该水平位移的长度 $l_{hi}\geq 2$.

证明:当 $i>1$ 时,设水平位移 L_{hi} 所对应的圆弧 A_{hi} 的起点和终点坐标分别为$(x_{i-1},y_{i-1}),(x_i,y_i)$,则水平位移 L_{hi} 的长度为

$$l_{hi}=[x_i]-[x_{i-1}].$$

由 Lagrange 定理,在开区间(x_{i-1},x_i)上至少存在一点 $x_0(x_{i-1}<x_0<x_i)$,使得等式 $y_i-y_{i-1}=f'(x_0)(x_i-x_{i-1})$ 成立.

由引理 1 可知,$|f'(x_0)|<1/2$.又由于$|y_i-y_{i-1}|=1$,因此可得

$$x_i-x_{i-1}=|x_i-x_{i-1}|=|y_i-y_{i-1}|/|f'(x_0)|>2|y_i-y_{i-1}|=2,$$

则
$$l_{hi}=[x_i]-[x_{i-1}]>(x_i-1)-x_{i-1}=(x_i-x_{i-1})-1>1.$$

又由于位移长度为整数,因此上式等价于 $l_{hi}\geq 2$. □

定理 2. 如果水平位移 L_{hi} 完全包含于 C_2,则该水平位移的长度 $l_{hi}\leq 2$.

证明:类似于定理 1 的证明.当 $i>1$ 时,设水平位移 L_{hi} 所对应的圆弧 A_{hi} 的起点和终点分别为$(x_{i-1},y_{i-1}),(x_i,y_i)$.同样,在开区间$(x_{i-1},x_i)$上至少存在一点 $x_0(x_{i-1}<x_0<x_i)$,使得等式 $y_i-y_{i-1}=f'(x_0)(x_i-x_{i-1})$ 成立.由引理 1 可知,此时 $|f'(x_0)|>1/2$.因此有

$$x_i-x_{i-1}=|x_i-x_{i-1}|=|y_i-y_{i-1}|/|f'(x_0)|<2|y_i-y_{i-1}|=2,$$

则
$$l_{hi}=[x_i]-[x_{i-1}]<x_i-(x_{i-1}-1)=(x_i-x_{i-1})+1<3.$$

又由于位移长度为整数,因此上式等价于 $l_{hi}\leq 2$. □

引理 2. Rokne 等人在文献[5,7]中指出,对于一个 2×2 的网格,位于第 2 个八分区的一段圆弧在其上的离散轨迹只能形成如图 3 所示的 4 种模式.并且,该段圆弧可分为两个子段,在每个子段中,模式 1 和模式 4 不可能同时出现.

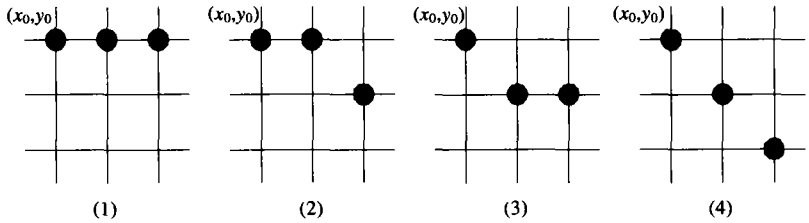

Fig.3 The four possible patterns of an arc's discrete loci in the second octant on 2×2 raster meshes

图 3 位于第 2 个八分区的圆弧在 2×2 栅格上的离散轨迹的 4 种可能出现的模式

由引理 2,我们可以得出如下推论:

推论 1. 设 A_i 是位于第 2 个八分区的一段圆弧上的任意一个子段,若 A_i 所对应的水平位移的长度 $l_{hi}\leq 2$,则必有 A_i 所对应的对角位移的长度 $l_{dj}\geq 2$.反之,若 A_i 所对应的水平位移的长度 $l_{hi}\geq 2$,则必有 A_i 所对应的对角位移的长度 $l_{dj}\leq 2$.

由定理 2 和推论 1,我们又可以得出:

推论 2. 如果对角位移 L_{dj} 完全包含于 C_2,则该对角位移的长度 $l_{dj}\geq 2$.

2 快速圆弧绘制算法

根据定理 1 和推论 2,可将位于第 2 个八分区的圆弧以直线 $y=2x$ 与圆弧的交点(即如图 2 所示的 P 点)为界分为两段(即如图 2 所示的 C_1 和 C_2)来绘制.对于 C_1 段圆弧,其离散轨迹由一系列长度至少为 2 的水平位移构成,故采用水平位移来绘制;对于 C_2 段圆弧,其离散轨迹由一系列长度至少为 2 的对角位移构成,故采用对角位移来绘制.下面介绍如何找出构成圆弧 C_1 和 C_2 离散轨迹的各段水平位移和对角位移.

2.1 求圆弧C_1中各段水平位移

对于 C_1 段圆弧,其上任意点的坐标(x,y)满足 $y>2x$.圆弧 C_1 起点坐标 $(x_0,y_0)=(0,r)$,当 $i\geq 1$ 时,第 i 段水平位移的 y 坐标为 $y_i=r-i+0.5$ (加 0.5 是为了考虑取整操作),最大 x 坐标为 x_i,则有

$$x_i^2=r^2-y_i^2=r^2-[r-(i-0.5)]^2, \tag{1}$$

$$x_{i+1}^2=r^2-y_{i+1}^2=r^2-[r-(i+0.5)]^2. \tag{2}$$

式(2)−式(1)得

$$x_{i+1}^2=x_i^2+2(r-i). \tag{3}$$

令 $X_i=[x_i], Y_i=[y_i]$,则第 i 段水平位移上所有像素点的序列为

$$(X_{i-1}+1, Y_i), (X_{i-1}+2, Y_i), \ldots, (X_i, Y_i). \tag{4}$$

当 $i=1, X_0=0, Y_0=r$ 时,第 1 段水平位移上所有像素点的 X 坐标满足

$$X^2 \le r^2-(r-0.5)^2 = r-1/4 < r. \tag{5}$$

由于 X 和 r 均为整数,故式(5)等价于

$$X^2 \le r-1. \tag{6}$$

为了避免费时的求平方和开方运算,用 S_i 表示 x_i^2,并令 $S_1=r-1$,则当 $i\ge 2$ 时,S_i 可由 $s_i=s_{i-1}+2(r-i)$ 求得.假定已求得 S_i, X_{i-1},且 $x=X_{i-1}$,引入变量 $E=S_i-X_{i-1}^2$,又由于 $(x+1)^2-x^2=x^2=2x+1$,则通过以下迭代即可得到各段水平位移的最大 x 坐标 X_i:

```
REPEAT{
    E←E−2x−1;
    x←x+1
}UNTIL (E<2X+1).
```

求出各 $X_i(i\ge 1)$ 后,根据式(4)即可绘制出圆弧 C_1 的各段水平位移,完整的算法参见第 2.3 节.其中变量 S_{inc} 和 S_{delta} 的引入是为了消除迭代过程中的乘法运算,以进一步提高圆弧生成速度.

2.2 求圆弧 C_2 中各段对角位移

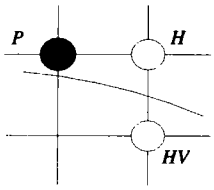

Fig.4 Point selection in Bresenham circle algorithm

图 4 Bresenham 法画圆时点的选择

对于 C_2 段圆弧,其上任意点坐标 (x,y) 满足 $y\le 2x$.设用水平位移绘制的最后一个像素点坐标为 (x',y'),用对角位移绘制的第 1 个像素点为 (x,y),则有 $x=x'+1, y=y'-1$.下面介绍如何求出构成 C_2 段圆弧的各段对角位移.

Bresenham 画圆法采用决策参数 d 来决定每步迭代的走向,在圆弧起点 $(x_0,y_0)=(0,r)$ 处决策参数的初值 $d=3-2r$.如图 4 所示,假定当前像素点为 P,该算法根据 d 的符号来决定下一个像素点是选择 H 还是 HV,同时更新决策参数 d:

若 $d<0$,则选 H 点,

$$d=d+\Delta H=d+4x+6; \tag{7}$$

若 $d\ge 0$,则选 HV 点,

$$d=d+\Delta HV=d+4(x-y)+10. \tag{8}$$

由于对角位移可看成由一个水平位移和一个垂直位移构成,故有下式成立:

$$\Delta HV=4(x-y)+10=(4x+6)+4(1-y)=\Delta H+\Delta V, \tag{9}$$

其中 $\Delta V=4(1-y)$.

绘制从 $(0,r)$ 到 (x,y) 的圆弧共经历了 x 段水平位移和 $(r-y)$ 段垂直位移,因此在点 (x,y) 处,决策参数 d 为

$$\begin{aligned} d &= (3-2r)+[4\cdot 0+6]+[4\cdot 1+6]+\ldots+[4(x-1)+6]+4(1-r)+4[1-(r-1)]+\ldots+4[1-(y+1)] \\ &= (3-2r)+4[1+2+\ldots+(x-1)]+6x+4(r-y)-4[r+(r-1)+\ldots+(y+1)] \\ &= 3-2r+2x(x-1)+6x+4(r-y)-2(r-y)(r+y+1) \\ &= 3-2r+4x+2x^2-2(r-y)(r+y-1). \end{aligned} \tag{10}$$

得到了绘制 C_2 段圆弧时决策参数的初值,我们就可以通过以下迭代找出各段对角位移:

```
REPEAT{
    x0←x;y0←y;
    REPEAT{
        d←d+4(x−y)+10;x←x+1;y←y−1;
    }UNTIL(d<0);
    IF (x<y) line (x0,y0,x,y);
```

$d \leftarrow d+4x+6; x \leftarrow x+1;$

}UNTIL($x \geq y$).

2.3 算法描述

根据上述思想,下面给出快速圆弧绘制算法的伪语言描述:

/*用水平位移绘制图 2 中 C_1 圆弧段*/

$x_0 \leftarrow 0; x \leftarrow x_0; y \leftarrow r; E \leftarrow -r-1; S_{inc} \leftarrow 2r; S_{delta} \leftarrow 1;$

WHILE($y > 2x$)DO{

$E \leftarrow E + S_{inc}; S_{inc} \leftarrow S_{inc} - 2;$

WHILE($E \geq S_{delta}$)DO{

$E \leftarrow E - S_{delta}; S_{delta} \leftarrow S_{delta} + 2; x \leftarrow x + 1;$

}

$line(x_0, y, x, y); y \leftarrow y-1; x_0 \leftarrow x+1;$

}

/*用对角位移绘制图 2 中 C_2 圆弧段*/

$x \leftarrow x+1; d \leftarrow 3-2r+4x+2x^2-2(r-y)(r+y-1);$

WHILE($x < y$)DO{

$x_0 \leftarrow x; y_0 \leftarrow y;$

WHILE($d \geq 0$)DO{

$d \leftarrow d+4(x-y)+10; x \leftarrow x+1; y \leftarrow y-1;$

}

IF($x < y$)line(x_0, y_0, x, y);

$d \leftarrow d+4x+6; x \leftarrow x+1;$

}

3 实验结果分析与比较

表 1 给出了分别用 Bresenham 法、中点法、HCL 法和本文提出的快速圆弧绘制算法绘制 9 种不同半径长度的 1/8 圆弧所用的时间(每种圆弧均绘制 1 000 次),括号中的数字为 Bresenham 法与各种算法绘制相同圆弧所需时间的比值.各种算法均用 Turbo C 实现,并在 P II 350 的 PC 机上进行实验.

Table 1 Run-Time comparison among the four algorithms

表 1 4 种算法运行时间对比

Radius (pixel units)①	Run-Time (ms)②			
	Bresenham algorithm③	Midpoint algorithm④	HCL algorithm⑤	New algorithm⑥
50	110	110 (1)	110 (1)	60 (1.83)
100	330	330 (1)	280 (1.18)	160 (2.06)
150	490	490 (1)	410 (1.20)	220 (2.23)
200	610	550 (1.11)	480 (1.27)	280 (2.18)
250	770	770 (1)	660 (1.17)	380 (2.03)
300	880	880 (1)	700 (1.26)	450 (1.96)
350	1040	1050 (0.99)	790 (1.32)	600 (1.73)
400	1260	1200 (1.05)	950 (1.33)	660 (1.91)
450	1320	1320 (1)	1320 (1.33)	710 (1.86)

①半径(像素单位),②运行时间(毫秒),③Bresenham 算法,④中点法,⑤HCL 算法,⑥新算法.

从表 1 可以看出,新算法与以往各种算法相比,能够提高绘制速度将近一倍.对于某些半径的圆弧,绘制速度甚至可以提高一倍以上.

4 结论与今后的工作

基于水平位移和对角位移,本文提出了一种快速圆弧绘制算法.该算法首先自适应地找出圆弧离散轨

迹中的各段水平位移和对角位移,然后用画线命令逐段进行绘制,每执行一次输出操作都可以生成多个像素点,从而减少了圆弧绘制过程中所需的输出操作,显著地提高了圆弧绘制速度.实验结果表明,与 Bresenham 法、中点法等经典的圆弧绘制算法相比,新算法可以提高圆弧绘制速度接近甚至超过一倍.

水平位移和对角位移在本文算法的推导和实现中起到了重要作用.圆弧是一种典型的二次曲线,而椭圆等其他二次曲线在光栅显示器上的离散轨迹也是由一系列水平位移和对角位移构成的.因此,今后工作的一个方向是将本文所提出的快速圆弧绘制算法推广到图形学中其他二次曲线的绘制中,相信可以取得较好的效果.

References:

[1] Bresenham, J.E. A linear algorithm for incremental digital display of circular arcs. Communications of the Association for Computing Machinery, 1977,20(2):100~106.

[2] Tang, Rong-xi, Wang, Jia-ye, Peng, Qun-sheng. Computer Graphics Course. Beijing: Science Press, 1990 (in Chinese).

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

[4] Hearn, D., Baker, M.P. Cai Shi-jie, Wu Chun-rong, Sun Zheng-xing, *et al.*, Trans. Computer Graphics. Beijing: Publishing House of Electronics Industry, 1998 (in Chinese).

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

[6] Hsu, S.Y., Chow, L.R., Liu, C.H. A new approach for the generation of circles. Computer Graphics Forum, 1993,12(2):105~109.

[7] Wu, X., Rokne, J. Double-Step generation of ellipses. IEEE Computer Graphics and Applications, 1989,9(3):56~69.

附中文参考文献:

[2] 唐荣锡,汪嘉业,彭群生.计算机图形学教程.北京:科学出版社,1996.65~68.

[4] Hearn, D., Baker, M.P.,著.蔡士杰,吴春镕,孙正兴,等,译.计算机图形学.北京:电子工业出版社,1998

A Fast Algorithm for the Drawing of Circles*

CHENG Jin, LU Guo-dong, TAN Jian-rong

(*State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou* 310027, *China*)

E-mail: chengjin_zju@263.net

http://www.cad.zju.edu.cn

Abstract: An algorithm for the drawing of circles is proposed in this paper. It differs greatly from the conventional algorithms that select one pixel per iteration because it can generate at least two pixels every time an I/O operation is executed. The algorithm is presented on the observation that the discrete loci of a circle are composed of a series of horizontal displacements and diagonal displacements. By locating and drawing these displacements one by one, the algorithm can effectively increase the speed of circle drawing due to the great reduction in the number of I/O operations. The experiment results prove that the circle-drawing speed of the algorithm can be almost doubled comparing with that of the Bresenham algorithm. Furthermore, the algorithm can be generalized to the production of other conics in computer graphics.

Key words: circle drawing; Bresenham algorithm; discrete loci; horizontal displacement; diagonal displacement

* Received May 15, 2001; accepted June 26, 2001

Supported by the National Natural Science Foundation of China under Grant No.69878038; the National High Technology Development 863 Program of China under Grant No.863-511-9842-006; the Natural Science Foundation of Zhejiang Province of China under Grant No.696045