第29卷第2期 上海师范大学学报(自然科学版) Vol. 29, No. 2
2000年6月 J. of Shanghai Teachers Univ. (Natural Sciences) Jun. 2000

差分迭代圆生成算法

王耀明[1]，施慧君[1]，董建萍[1]，王仲国[2]

(1. 上海师范大学 理工信息学院，上海 200234； 2. 上海市奉贤县电视大学，上海 201400)

摘 要：在光栅象素的圆生成算法中，中点圆算法以其计算量少、精度高而广泛使用于各种绘图软件中，但它仅在一个坐标上的迭代使其对斜率有较高敏感度，因而限制了它的应用范围．论文介绍的二阶差分迭代算法，使用了两个坐标上同时迭代的方法，使它无任何的斜率敏感度，因而扩大了它的应用范围，并且它的计算量和精度与中点圆算法相当．

关键词：中点；差分迭代；光栅；取样象素

中图分类号：TP391.41 **文献标识码**：A **文章编号**：1000-5137(2000)02-0030-06

圆是图形中经常使用的一个基本元素，并且往往是显示椭圆曲线的基础，生成圆的方法主要有两种．第一种是线段近似法(或称多边形近似)[1,2]，即把圆周分成许多小圆弧，每个小圆弧用一线段近似代替，所以这种方法也称为折线生成法．第二种称为光栅象素圆生成算法，它是比较象素与圆的距离而取样象素位置生成圆周的方法．在这种方法中最常用的是Bresenham圆算法[3]和中点圆算法[3,4]．下面简要介绍中点圆算法的原理，并指出其中缺点，然后着重讨论二阶差分迭代圆弧生成算法．

1 光栅象素分割及曲线显示

光栅显示器的屏幕可按水平行和垂直列分割成不重叠的许多小方块，这些小方块称为象素，象素的坐标由小方块的中心位置所决定．在屏幕上显示一条曲线实际上就是取样一些象素，其亮度等与其他象素不一样．取样象素一般应注意以下几个方面的问题：

(1) 取样象素与曲线之间距离应保持最小；(2) 在选择取样象素的过程中往往采取迭代计算方法，迭代方法的一个缺点是误差的传递．一般可把取样象素的中点与曲线之间的偏离保存起来，在下次迭代中使用，从而减少或消除误差传递．

收稿日期：1999-11-04

作者简介：王耀明(1945-)，男，上海师范大学理工信息学院副教授．

2　中点圆算法

对于给定半径 r 和圆屏幕中心，可先给出计算中心在原点圆的象素位置算法，然后通过平移将每个计算出的位置移到其适当的屏幕位置上. 利用图的对称性、仅需计算到的第一象限中圆弧段(其曲线斜率从0到－1)就行. 其他7段圆弧按图1所示设定.

2.1　中点算法原理

设一个圆的笛卡尔坐标方程为 $x^2+y^2=r^2$，定义一个圆函数

$$f_c(x,y)=x^2+y^2-r^2, \tag{1}$$

对屏幕上任意点(x,y)与圆的相对位置可用圆函数的符号来检测

$$f_c(x,y)\begin{cases}<0, & (x,y)\text{ 位于圆边界内.}\\ =0, & (x,y)\text{ 位于圆边界上.}\\ >0, & (x,y)\text{ 位于圆边界外.}\end{cases} \tag{2}$$

对于设定的从 $x=0$到 $x=y$ 的第一象限圆弧段(斜率从0变到－1)则可选择正 x 方向取单位步长变化[4]. 由于斜率在0～－1之间变化、那么当选定了(x_k,y_k)象素后，下一个象素要么是(x_{k+1},y_k)或是(x_{k+1},y_k-1)(见图2). 定义一个决策参数 P_k、其值为圆函数在这两个象素的中点$(x_{k+1},y_k-\frac{1}{2})$处的值，即

$$P_k=f_c(x_{k+1},y_k-\frac{1}{2})=(x_k+1)+(y_k-\frac{1}{2})^2-r^2. \tag{3}$$

若 $P_k<0$，说明中点$(x_{k+1},y_k-\frac{1}{2})$在圆边界内，则取$(x_{k+1},y_k-\frac{1}{2})$，若 $P_k\geqslant 0$，说明中点在圆边界上或圆边界外，则取(x_{k+1},y_k-1).

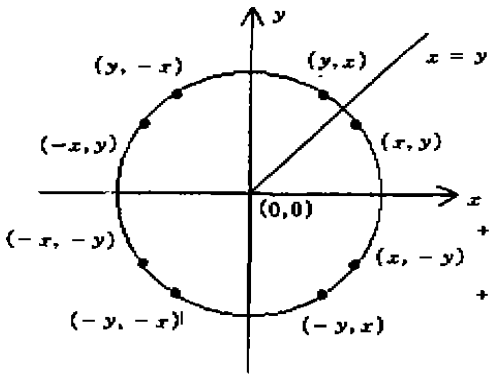

图 1

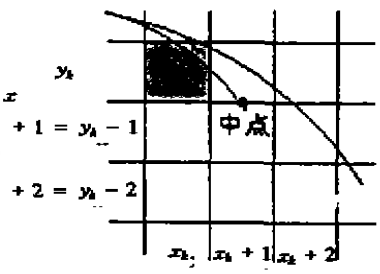

图 2

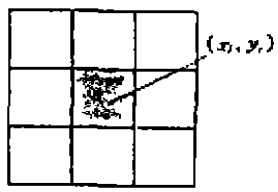

图 3

迭代公式：

(1) 若 $P_k<0$时，选象素为(x_{k+1},y_k)、则下一个决策参数：

$$P_{k+1}=f_c(x_{k+1}+1,y_k-\frac{1}{2})=P_k+2x_{k+1}+1. \tag{4}$$

(2) 当 $P_k \geqslant 0$时,选象素为(x_{k+1}, y_{k+1}),则下一个决策参数:

$$P_{k+1} = f_c(x_{k+1} + 1, y_k - \frac{1}{2}) = P_k + 2x_{k+1} - y_{k+1} + 1 . \tag{5}$$

2.2 优缺点

(1) 计算量少:在迭代主循环中仅有移位、加、减及判断 P_k 正负符号的运算.故方便硬件实现.

(2) 误差小:整个算法中,取样象素与圆周之间距离在象素大小的1/2之内.

(3) 在画圆弧时需判断曲线斜率变化,其绝对值小于1时,采用 x 轴增量,反之采用 y 轴增量 .

(4) 该算法可推广到画椭圆曲线中,但在主循环中出现乘法运算,运算量相应增加,较难硬件实现.

3 差分迭代算法

差分迭代算法使用圆参数方程的二阶泰勒级数近似式而产生的迭代计算公式.它具有计算量少(主循环中无乘法运算),精度高,对曲线斜率值不敏感等优点[5],而且把它推广的椭圆算法中也同样不需要乘法算法.

3.1 原 理

设圆心在点的圆参数方程及其一阶微商为:

$$\begin{cases} x(t) = r\cos t \\ y(t) = r\sin t \end{cases}, \tag{6}$$

$$\begin{cases} x' = -y \\ y' = x \end{cases}, \tag{7}$$

以初始点开始,以步长 Δt 增加来迭代求解.设第 i 点由第 $i-1$点增加 Δx_{i-1}和 Δy_{i-1}而得,即:

$$\begin{cases} \Delta x_{i-1} = x_i - x_{i-1} \\ \Delta y_{i-1} = y_i - y_{i-1} \end{cases}, \tag{8}$$

对第 $i+1$点,我们取式(6)的泰勒级数前三项来近似:

$$\begin{aligned} x_{i+1} &= x_i + x_i'\Delta t + \frac{1}{2}x_i''\Delta t^2, \\ y_{i+1} &= y_i + y_i'\Delta t + \frac{1}{2}y_i''\Delta t^2. \end{aligned} \tag{9}$$

由于

$$x_i'' = \frac{1}{\Delta t}(x_i' - x_{i-1}') = \frac{1}{\Delta t}(-y_i + y_{i-1}) = \frac{1}{\Delta t}(-\Delta y_{i-1}),$$

$$y_i'' = \frac{1}{\Delta t}(y_i' - y_{i-1}') = \frac{1}{\Delta t}(x_i - x_{i-1}) = \frac{1}{\Delta t}\Delta y_{i-1}.$$

则由(9)式就可得:

$$\begin{aligned} x_{i+1} &= x_i + [-2(y_{i+1} + \Delta y_{i-1})]\frac{\Delta t}{2} = x_i + \Delta x_i, \\ y_{i-1} &= y_i + (2x_i + \Delta x_{i-1})\frac{\Delta t}{2} = y_i + \Delta y_i. \end{aligned} \tag{10}$$

其中

$$\begin{aligned} \Delta x_i &= -(2y_i + \Delta y_{i-1})\frac{\Delta t}{2}, \\ \Delta y_i &= -(2x_i + \Delta x_{i-1})\frac{\Delta t}{2}. \end{aligned} \tag{11}$$

第(10)和(11)式就是迭代计算中用到的公式.

3.2 具体实现方法

3.2.1 降低误差

由上所述，为了降低偏离所引起的误差、应该把偏离保存起来、在下次计算中使用. 这反映在(11)式中. 式中 Δx_i(或 Δy_i)可分成两个部分，一个是整数部分(可用 $\mathrm{d}x$ 和 $\mathrm{d}y$ 表示)、反映了未来新的取样象素离原来点(x_i, y_i)的距离. 另一个是小数部分(可用 r_x 和 r_y 表示)，它反映了所选取样象素与曲线位置之间的偏离. 如果计算过程中限定小数部分不超过1/2，则偏离(即误差)在象素分段大小的1/2之内. 由上面分析(10)和(11)迭代式可改写为

$$\begin{cases} x_{i+1} = x_i + \mathrm{d}x_i \\ y_{i+1} = y_i + \mathrm{d}y_i, \end{cases} \tag{12}$$

$$\begin{aligned} \Delta x_i &= \mathrm{d}x_i + r_{x_i} = -(2y_i + \underbrace{\mathrm{d}y_i - 1 + r_{y_{i-1}}}_{\Delta y_{i-1}})\frac{\Delta t}{2}, \\ \Delta y_i &= \mathrm{d}y_i + r_{y_i} = -(2x_i + \underbrace{\mathrm{d}x_i - 1 + r_{x_{i-1}}}_{\Delta x_{i-1}})\frac{\Delta t}{2}. \end{aligned} \tag{13}$$

3.2.2 步长的选取

为了达到用移位来代替乘法运算，可选择 $\Delta t = 2^{-m}$，则 $\frac{\Delta t}{2} = 2^{-m+1}$. 在(11)式，$2x_i$ 和 $2y_i$ 显然是整数，假设 $\Delta x_{i-1}, \Delta y_{i-1}$是整数，那么产生的余数(小数)显然是由 $\frac{\Delta t}{2}$ 引起的. 即余数部分的二进制位数为$(m+1)$位.

为了保证曲线的连续性，则下个象素的坐标(x_{i+1}, y_{i+1})应当与原象素坐标(x_i, y_i)相邻. 在8方向前进形式中[4]，可有8个不同的象素供选择(如图3所示). 从(12)式中可知，$|\mathrm{d}x|$，$|\mathrm{d}y|$只有两种选择，要么是全1(4个对角线上象素)，要么一个为1、另一个为0(上、下、左、右4个象素). 即 $|\mathrm{d}x|, |\mathrm{d}y| \leqslant 1$，同时我们知道，圆形上的任意点坐标$(x_i, y_i)$均有 $|x_i| \leqslant r$，$|y_i| \leqslant r$，其中 r 是以象素数表示的圆半径.

由上面分析并从(11)、(13)式可以推出，当在点$(0, r)$时

$$|\mathrm{d}x_i| = [|\Delta x_i|] = [|2y_i + \Delta y_{i+1}|\frac{\Delta t}{2}] = |2r|\frac{\Delta t}{2} \leqslant 1$$

$$\Rightarrow \Delta t \leqslant \frac{1}{r} \Rightarrow 2^{-m} \leqslant \frac{1}{r} \Rightarrow m = [\log_2 r] + 1.$$

3.2.3 迭代次数

利用圆的对称性,在主循环中仅需对1/8圆周进行计算(见图1).

由(2)中可知$2^m>r$,圆的1/8周长为$\frac{1}{8}\times 2\pi r=\frac{1}{4}\pi r<\frac{1}{4}\pi\times 2^m=\pi\times r^{m-2}$,所以迭代次数为$N=[2^{m-2}\pi+0.5]$.

3.2.4 算法步骤

①初始条件

$dx=dy=0$, $r_x=r_y=2^m$, (相当于0.5个象素分段长度,r_x,r_y有$m+2$位),

$m=[\log_2 r]+1$, $x=r$,$y=0$, (r为象素数的半径长度).

②计算Δx_i,Δy_i并分别存放在r_x,r_y,dx,dy中.

$r_x=r_x+dy+2y$, $r_y=r_y+dx+2x$,

$dx=r_x$算术右移$(m+1)$位 (取dx_i), $dy=r_y$算术右移$(m+1)$位 (取dy_i),

$r_x=r_x$ and $(2^{m+1}-1)$ (取r_{x_i}), $r_y=r_y$ and $(2^{m+1}-1)$ (取r_{y_i}),

③计算x_{i+1},y_{i+1},并分别存放于x,y中.

$$x=x-dx,\qquad y=y+dy.$$

④确定其它7个8分圆中的对称点.

⑤计算出相应8个象素位置并显示.

⑥重复②~⑤步直至$y\geqslant x$.

4 两种算法的比较与实例

除了上面所述两种算法的特点外,下面对几个实际例子(对不同的半径r)通过计算它们各自运行时间和相对圆形$x^2+y^2=r^2$的误差列表比较它们之间的差异.对每种算法所选取的取样的象素坐标(x_i,y_i)由公式$x^2+y^2=r^2$计算r_i值,则取样象素与圆曲线之间的距离为$\delta_i=|r_i-r|$,其平均误差$\delta=\frac{1}{N}\sum_{i=0}^{N}\delta_i$.

运行时间t是在同一软、硬件的环境下计时,重复次数为100次.

表1

半径r(象素数)	中点圆算法		差分迭代圆算法	
	$t(s)$	δ	$t(s)$	δ
2	0	0.292893	0	0.236068
20	0.109980	0.181675	0.109890	0.220566
50	0.164835	0.211442	0.219780	0.253515
70	0.219780	0.204563	0.494505	0.211247
100	0.329670	0.214528	0.439560	0.240558
150	0.439560	0.211905	0.934066	0.228506
180	0.549451	0.221440	0.934066	0.234787

5 小 结

从上面的分析和实验结果看出,二阶差分迭代算法在精度上和中点圆算法相同,而其运算量约为中点算法的2倍.这是由于它同时进行两个坐标上的迭代的结果.但二阶差分迭代算法无任何的斜率敏感性,所以,它能推广到更一般圆锥曲线的生成算法中而无需大量增加计算量.

参考文献:

[1] MAXWELL P C, BAKER P W. The generation of polygons representing circle, ellipsis and hyperbolas [J]. Compute, Graph, Image Process, 1979, 10(2).

[2] BRESENHAM J E. A Linear Algorithm for Incremental Digital Display of Circular Arcs[J]. CACM, 1977, 20(2).

[3] MU KUNDAN. An Ellipse-Drawing Algorithm for faster Display[M], In Fundamental Algorithm for Computer Graphics Spring-Verlag, Berlin, 1985.

[4] DONALD HEARM, PAULINE M. Computer Graphics[M]. Baker Prentice Hall 1996.

[5] DIETER W FELLNEN. Robust Rendering of General Ellipses and Elliptical Arcs[J]. ACM Trons on Graphics, 1993, 12(3).

[6] DOUGLAS MCILROY M. Getting Raster Ellipses Right[J]. ACM Trans on Graphics, 1992. 11(3).

A Difference Iteration Algorithm for Generating a Circle

WANG Yao-ming[1], SHI Hui-jun[1], DONG Jian-ping[1], WANG Zhong-guo[2]

(1. College of Science and Information Technology, Shanghai Teachers University, Shanghai, 200234, China.
2. TV University of Fengxian county in Shanghai, Shanghai, 201400, China)

Abstract: In Circle-Generating Algorithms for raster-pixel, the central-point algorithm is widely used in various drawing software, because of its fast calculation and high precision. But it iterates only on one axis, which causes its high slope sensitivity and restricts its applications. We discuss an algorithm of two-order difference iteration, which uses two-axis iterations and has no slope sensitivity. It has wider range of application. Its calculation and precision are as fine as those of the central-point algorithm.

Key words: central-point; difference-iteration; raster; sample-pixel