中图法分类号: TP391 文献标志码: A 文章编号: 1006-8961(2012)01-0130-07
论文引用格式: 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

圆的整数反走样生成算法

贾银亮,张焕春,经亚枝
南京航空航天大学自动化学院,南京 210016

摘 要: 针对现有圆的反走样生成算法计算复杂,反走样效果一般的缺点,提出一种基于中点画圆法的整数反走样生成算法。该算法根据像素中心到理想圆弧的距离来分配灰度,生成 64 级灰度的反走样圆弧。通过省略二次项来简化计算,并用简单的计算修正省略带来的误差以保证精度。为了简化计算,提出相邻像素的灰度递推方法,利用整数移位、加法、比较来实现反走样。该算法结构简单,反走样效果较好,由于避免了浮点和除法运算,便于硬件实现。

关键词: 圆弧; 反走样; 整数运算; 灰度

Integral algorithm for circle anti-aliasing

Jia Yinliang, Zhang Huanchun, Jing Yazhi
College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

Abstract: Anti-aliasing for circles is complex and the existing algorithms are not working satisfactory. To improve the efficiency and the anti-aliasing effect, an integral algorithm is presented for circle anti-aliasing based on midpoint generating algorithm. The new algorithm finds the grayscales of each pixel according to the distance between the center of the pixel and the circle. A circle with a 64 levels grayscale can be drawn. The algorithm abandons the two-order epsilon and corrects the error by simple calculations. We built a method to forecast the grayscale change between neighboring pixels using integer shift, addition, and comparing without using floating-point and divisions, making it easy to implement the algorithm on hardware. The results show that the anti-aliasing effect and its efficiency have been improved.

Key words: circle; anti-aliasing; integer operation; grayscale

0 引 言

随着计算机的发展,关系显示效果的计算机图形技术得到了广泛的应用,各种图形算法越来越受到重视。圆是最基本的图形之一,在显示时经常需要大量绘制,由于圆的基础性和绘制的大量性,提高其生成算法的效率或显示效果对画面的显示质量的提升有直接影响,所以其生成算法的任何改进都显得非常重要。目前,圆生成得到了深入研究,提出了一些算法,其进一步改进的难度也很大。

光栅图形显示器是目前使用广泛的图形显示器,其缺陷在于只能用若干像素来表示连续的理想图形。例如理想的圆是连续的,而光栅显示器用一系列离散的像素来表示一个圆,这将导致所显示的圆呈现锯齿状,影响显示效果。这种用离散量表示连续量引起的失真现象称为走样。减轻或者除去走样现象的技术称为反走样[1]。

目前,直线反走样技术研究较深入,出现了一些成熟的算法。由于圆方程是二次的,其反走样计算比直线复杂得多,尚未有效率和效果都较好的圆反走样算法[2]。本文对圆反走样技术进行研究,提出

收稿日期: 2011-01-17; 修回日期: 2011-05-30

第一作者简介: 贾银亮(1979—),男,讲师,现为南京航空航天大学测控技术专业博士研究生,主要研究方向为计算机图形学、嵌入式系统等。E-mail: flybearnuaa@163.com

一种新的算法。该算法实现 64 级灰度的反走样,像素的灰度计算准确,效果较好,并且计算简单,只需要比较、移位和加法运算。

1　相关工作

1.1　圆的生成算法

圆被定义为到给定中心位置距离为 R 的点集。圆心位于原点的圆有 4 条对称轴 $x=0$,$y=0$,$x=y$ 和 $x=-y$。若已知圆弧上一点(x,y),可以得到其关于 4 条对称轴的其他 7 个点,这种性质称为圆的八方向对称性。因此,只要扫描转换 1/8 圆弧,就可以求出整个圆。对于圆心位于原点,半径为 R 的圆,本文只考虑从$(0,R)$到 $\left(\frac{R}{\sqrt{2}},\frac{R}{\sqrt{2}}\right)$ 的圆弧。

中点画圆法在同一列的两个像素之间的中点处给出一个评估函数值,并据此在这两个像素中选择更靠近理想圆弧的像素。如图 1 所示,如果像素 $P(x_P,y_P)$ 是刚被选择的像素,那么下一个像素就要在像素 E 和 S 之间选择,$M(x_P+1,y_P-0.5)$ 是它们之间的中点。

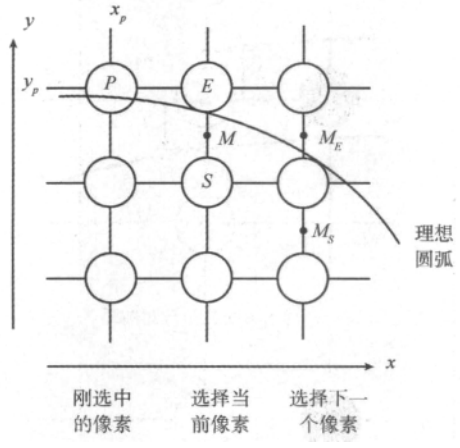

图 1　中点画圆法的像素

Fig. 1　Pixels of midpoint generating algorithm

设函数 $f(x,y)=x^2+y^2-R^2$。对于圆上的点,函数值是 0;对于圆内的点,函数值为负;而对于圆外的点,函数值为正。如果 M 在圆外,则 S 更靠近圆。相反,如果 M 在圆内,则 E 更靠近圆。

选择像素是根据判定变量 d 的值,即函数 $f(x,y)$ 在中点 M 处的值

$$d_{old}=f(x_P+1,y_P-0.5)=(x_P+1)^2+(y_P-0.5)^2-R^2$$

如果 $d_{old}<0$,就选择像素 E,而下一个中点是 $M_E(x_P+2,y_P-0.5)$,从而得到

$$d_{new}=f(x_P+2,y_P-0.5)=(x_P+2)^2+(y_P-0.5)^2-R^2$$

由于 $d_{new}=d_{old}+(2x_P+3)$,因此,增量 $\Delta_E=2x_P+3$。

如果 $d_{old}\geqslant 0$,就选择像素 S,而下一个中点是 $M_S(x_P+2,y_P-1.5)$,从而得到

$$d_{new}=f(x_P+2,y_P-3/2)=(x_P+2)^2+(y_P-3/2)^2-R^2$$

由于 $d_{new}=d_{old}+(2x_P-2y_P+5)$;可得到增量 $\Delta_S=2x_P-2y_P+5$。

Δ_E 和 Δ_S 每一步都要变化,直接将上一步所选择像素的 x 和 y 坐标值代入计算新的增量。由于计算公式是线性的,其直接计算的开销并不大。

此算法的每次循环都做同样的两步: 第 1 步根据上一次循环时所计算的循环变量 d 值的符号选择一个像素; 第 2 步根据所选择的像素计算增量 Δ,并用它计算新的判定变量 d。

圆的起始像素是$(0,R)$。由于下一个中点的位置是$(1,R-0.5)$,因此判定变量 d 的初值为 $1.25-R$。

Bresenham 算法与中点算法类似,其计算量是一样的。Wu 和 Rokne 提出的双步圆生成算法可以提高效率,但是该算法并不是选择所有到理想圆弧最近的像素。这些生成算法效率较高,但是没有反走样,显示效果较差[3]。

1.2　圆的反走样算法

常用的反走样技术有提高分辨率和区域采样。提高分辨率可以减少灰度跳变的幅度。在视觉上,显示出的图形比较平直光滑,但是存储器容量和扫描转换时间都将大大增加。正因为增加分辨率的代价非常大,所以这种方法使用较少。区域采样认为图形对象有一定宽度,根据图形对象在每个像素上的覆盖程度率来确定像素的灰度。由于区域采样的反走样效果较好且增加的计算量不大,所以区域采样是反走样的主流方法[4-6]。

Field 提出一种方形区域采样实现圆弧反走样的算法,该算法需要平方运算,计算量较大[7]。文献[8-10]讨论了几种圆的反走样算法,通过像素到理想圆弧的距离来分配灰度实现反走样,但是算法复杂,计算量大,甚至需要浮点运算。

Wu 和 Rokne 提出双点圆反走样算法,该算法只使用整数运算,计算量少,但是该算法在最小和最大灰度之间只能产生一个中间灰度级。当半径 R 为奇数时,生成的是一个类似角度平滑的四边形而

不是圆，且在某些固定角度有像素点丢失现象[11]。

文献[12-13]对此算法作了些改进，减少了累加误差值。当R为奇数时，生成的图形比较接近圆，其生成的圆弧比前者的效果要好一些。但是改进后的算法在某些固定角度仍有像素丢失，而且只能产生5个中间灰度级，反走样效果一般。

文献[14]通过对目标像素与理想圆弧位置的详细分析，以中点画圆法为基础建立了反走样算法。根据中点画圆法中的误差控制参数、候选像素与理想圆弧间的位置关系计算出精确距离，舍去作用较小的高阶量，利用查表或区间二分检索法计算像素的灰度。该算法号称仅使用整数基本运算，但计算中要么需要除法和取整运算；要么需要查表或检索，实际上计算效率并不高。在实现16级灰度反走样时，该算法的计算量尚可接受，进一步提高灰度级别会大幅提高计算量，更关键的是直接舍去高阶量在某些情况下误差较大，甚至改变灰度的符号导致选择错误的像素。

总之，现有算法有的中间灰度级太少，反走样效果一般，有的计算复杂，效率较低，且都不适于硬件实现[15]。本文提出一种新的算法，该算法实现64级灰度的圆反走样并能扩展到更多灰度级，像素的灰度计算准确，效果较好，且计算简单，只需要整数比较、移位和加法运算，有利于硬件实现。

2 新的圆反走样算法

设像素灰度 $g \in [1,64]$，即实现64级灰度的反走样，像素E的灰度为 g_E。中点画圆法在每一列上选择距离理想圆弧最近的一个像素赋以最大灰度，其他像素赋以最小灰度。为了减少像素灰度的跳变幅度，新算法在每一列上选择距离理想圆弧最近的两个像素。这两个像素的灰度和为最大灰度，根据这两个像素中心到理想圆弧的距离来分配灰度。

由于一个像素的面积很小，为简化计算，将每一列内部的圆弧看成直线段，如图2所示。以圆弧与像素中心连线的交点Q处的切线来代表这一段圆弧。从图2可见，两个像素中心到切线的距离比和两个像素中心沿y方向到达中点Q的长度比 $l_1:l_2$ 相同，本文将通过计算 $l_1:l_2$ 来得到像素的灰度。图2中，$l_1:l_2=2:3$，设最大灰度为 g_{max}，则图中上方像素的灰度为 $0.6\times g_{max}$，下方像素的灰度为 $0.4\times g_{max}$。

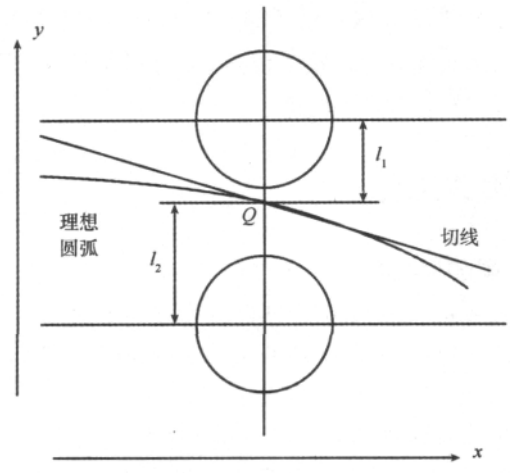

图2 根据距离分配灰度

Fig. 2 Assign grayscale according to the distance

在中点画圆法中，像素 $P(x_P,y_P)$ 是已确定的像素，在判定下一个像素时，若判定变量 $d<0$，如图3(a)，像素E是最接近理想直线的。先考虑理想

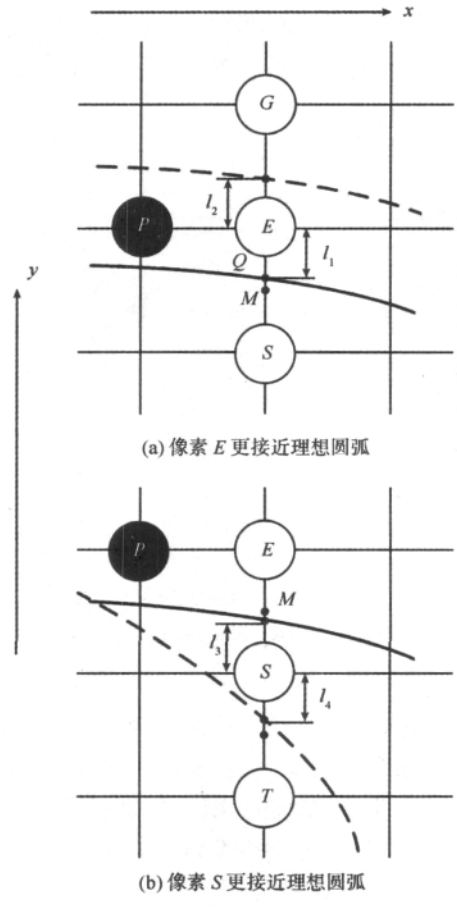

图3 圆弧的不同位置

Fig. 3 Position of circle

圆弧为图 3(a) 中实线的情况。由于交点 Q 在圆上,所以式(1) 成立。

$$(x_P + 1)^2 + (y_P - l_1)^2 - R^2 = 0 \quad (1)$$

而像素 E、S 的中点 M 的判定变量满足

$$d = (x_P + 1)^2 + (y_P - 0.5)^2 - R^2 \quad (2)$$

将式(2) 代入式(1)

$$l_1 = \frac{d - 0.125}{2y_p} + \frac{1}{2} + \frac{l_1^2 - 0.125}{2y_p}$$

显然,此时像素 E、S 距理想圆弧最近,像素 S 的灰度应为 $g_{max} \times l_1$,像素 E 的灰度应为 $g_{max} - g_S$。由于像素 E 最接近理想圆弧,所以 $l_1 \in [0, 0.5)$,则 $\frac{l_1^2 - 0.125}{2y_p} \in \left[-\frac{1}{16y_p}, \frac{1}{16y_p}\right)$。

像素最大灰度 $g_{max} = 64$,若 $y_P \geqslant 4$,则 $64 \times \frac{l_1^2 - 0.125}{2y_p} \in [-1, 1)$。在这种情况下,省略 $\frac{l_1^2 - 0.125}{2y_p}$ 带来的误差不超过 1 个灰度单位,可以用 $l_1 = \frac{d - 0.125}{2y_p} + \frac{1}{2}$ 来计算灰度以简化计算。

这里假设 $y_P \geqslant 4$,即要保证 $R \geqslant 7$。若 $R < 7$,像素的灰度计算误差将超过 ±1,但此时误差仍较小,而且由于半径较小时,各种算法生成的圆更接近多边形,所以灰度误差造成的影响有限,不会影响反走样效果。

此时

$$g_S = \frac{32d - 4}{y_P} + 32, \quad g_E = 64 - g_S$$

由判定变量的定义,$d = (y_P - 0.5)^2 - (y_P - l_1)^2$,因为 $l_1 \in [0, 0.5)$,所以 $d \in [0.25 - y_P, 0)$。

若理想圆弧为图 3(a) 中虚线的情况,则像素 E、G 最接近理想圆弧,有式(3) 成立。

$$(x_P + 1)^2 + (y_P + l_2)^2 - R^2 = 0 \quad (3)$$

将式(2) 代入式(3) ,得

$$l_2 = -\frac{d - 0.125}{2y_p} - \frac{1}{2} - \frac{l_2^2 - 0.125}{2y_p}$$

像素 E 最接近理想圆弧,所以 $l_2 \in (0, 0.5]$,则 $\frac{l_2^2 - 0.125}{2y_p} \in \left(-\frac{1}{16y_p}, \frac{1}{16y_p}\right]$,可以省略。此时 $g_G = -\frac{32d - 4}{y_p} - 32$,$g_E = 64 - g_G$。

由判定变量的定义,$d = (y_P - 0.5)^2 - (y_P + l_2)^2$,因为 $l_2 \in (0, 0.5]$,所以 $d \in [-2y_P, 0.25 - y_P)$。

若判定变量 $d \geqslant 0$,则如图 3(b) 所示。先考虑理想圆弧为图 3(b) 中实线的情况,像素 S、E 最接近理想圆弧,有式(4) 成立。

$$(x_P + 1)^2 + (y_S + l_3)^2 - R^2 = 0 \quad (4)$$

而像素 S、E 的中点 M 的判定变量满足

$$d = (x_P + 1)^2 + (y_S + 0.5)^2 - R^2 \quad (5)$$

将式(5) 代入式(4)

$$l_3 = -\frac{d - 0.125}{2y_S} + \frac{1}{2} - \frac{l_3^2 - 0.125}{2y_S}$$

由于像素 S 最接近理想圆弧,所以 $l_3 \in (0, 0.5]$,则 $\frac{l_3^2 - 0.125}{2y_S} \in \left(-\frac{1}{16y_S}, \frac{1}{16y_S}\right]$,可以省略。此时 $g_E = 32 - \frac{32d - 4}{y_S}$,$g_S = 64 - g_E$。

由判定变量的定义,$d = (y_P - 0.5)^2 - (y_S + l_3)^2$,因为 $l_3 \in (0, 0.5]$,$y_S = y_P - 1$,所以 $d \in [0, y_P - 0.75)$。

若理想圆弧为图 3(b) 中虚线的情况,则像素 S、T 最接近理想圆弧,有式(6) 成立。

$$(x_P + 1)^2 + (y_S - l_4)^2 - R^2 = 0 \quad (6)$$

将式(5) 代入式(6) ,得

$$l_4 = \frac{d - 0.125}{2y_S} - \frac{1}{2} + \frac{l_4^2 - 0.125}{2y_S}$$

由于像素 S 是最接近理想圆弧,所以 $l_4 \in [0, 0.5)$,则 $\frac{l_4^2 - 0.125}{2y_S} \in \left[-\frac{1}{16y_S}, \frac{1}{16y_S}\right)$,可以省略。此时 $g_T = \frac{32d - 4}{y_S} - 32$,$g_G = 64 - g_T$。

由判定变量的定义,$d = (y_P - 0.5)^2 - (y_S - l_4)^2$,因 $l_4 \in [0, 0.5)$,故 $d \in (y_P - 0.75, 2y_P - 2]$。

根据以上推导,圆的反走样算法可以这样设计: 当 $0.25 - y_P \leqslant d < 0$ 时,选择 $E(x_P + 1, y_P)$ 和 $S(x_P + 1, y_P - 1)$,灰度分别为 $g_S = \frac{32d - 4}{y_P} + 32$,$g_E = 64 - g_S$; 当 $d < 0.25 - y_P$ 时,选择 $E(x_P + 1, y_P)$ 和 $G(x_P + 1, y_P + 1)$,灰度分别为 $g_G = -\frac{32d - 4}{y_P} - 32$,$g_E = 64 - g_G$; 当 $0 \leqslant d < y_P - 0.75$ 时,选择 $E(x_P + 1, y_P)$ 和 $S(x_P + 1, y_P - 1)$,灰度分别为 $g_E = 32 - \frac{32d - 4}{y_S}$,$g_S = 64 - g_E$; 当 $d \geqslant y_P - 0.75$ 时,选择 $S(x_P + 1, y_P - 1)$ 和 $T(x_P + 1, y_P - 2)$,灰度分别为

$g_T = \frac{32d-4}{y_S} - 32$，$g_S = 64 - g_T$。

在计算像素灰度的同时，按照中点画圆法递推 d 的下一个值，而 d 的初值为 1.25 − R。圆弧第一列的两个像素(0,R)、(0,R−1)的灰度分别为 64 和 1。

3 圆反走样算法的简化

按照上面的推导，可以通过加、比较和乘除运算完成圆弧的反走样。但是小数乘除法比较复杂，不利于提高效率和硬件实现，下面考虑如何消除这些相对复杂的运算。

设判定变量 $c = 32d$。当 $8 - 32y_P \leqslant c < 0$ 时，选择 E 和 S；当 $c < 8 - 32y_P$ 时，选择 E 和 G；当 $0 \leqslant c < 32y_P - 24$ 时，选择 E 和 S；当 $c \geqslant 32y_P - 24$ 时，选择 S 和 T。

在计算像素灰度的同时，递推 c 的下一个值，而 c 的初值为 40 − 32R。

$$c_{new} = c_{old} + 32(2x_P + 3) \qquad c_{old} < 0$$

$$c_{new} = c_{old} + 32(2x_P - 2y_P + 5) \qquad c_{old} \geqslant 0$$

经过处理后算法变成了整数运算，接下来通过递推来消去除法运算。

圆弧上相邻像素的位置关系有图 4 中的 4 种，图中每列只画出了最接近理想圆弧的像素。像素 W、U 的灰度已计算好，现计算像素 V 的灰度。圆弧上相邻像素的灰度变化是有规律的，可以利用这种规律建立灰度变化量的递推公式，即通过像素 W、U 的灰度变化，求像素 U、V 的灰度变化，进而求出 V 的灰度。

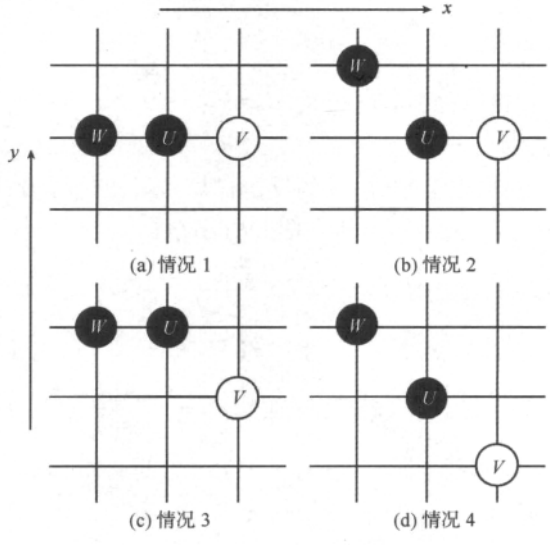

图4 相邻像素的不同位置

Fig. 4 Position of neighbouring pixels

如图 4(a)中的情况，求 V 的灰度需要求 $\frac{c_V - 4}{y_V}$。设 $a_V = \frac{c_V - 4}{y_V}$，$b_V = 2x_W + 3$。根据中点画圆算法，不难得到

$$a_V = \frac{c_U + 32b - 4}{y_U} = a_U + \frac{32b}{y_U}$$

由于 $\Delta b = 2$，有式(7)成立

$$\Delta a_V = \frac{32b}{y_U} = \Delta a_U + \frac{64}{y_U} \tag{7}$$

若相邻像素如图 4(b)中的情况，根据中点画圆算法

$$a_V = \frac{c_V - 4}{y_V} = \frac{c_U + 32(2x_W - 2y_W + 5) - 4}{y_U} = a_U + \frac{32b}{y_U} - 64$$

由于 $\Delta b = 2$，有式(8)成立

$$\Delta a_V = \Delta a_U + \frac{64}{y_U} - 64 \tag{8}$$

利用式(7)(8)，可以为灰度建立递推公式，利用 Δa_U，区分图 4(a)(b)的不同情况，算出灰度改变量的增量，得到 Δa_V，进而算出像素 V 的灰度值。以图 4(a)情况为例，设 α、β 是 $64/y_U$ 的商和余数；γ、δ 是 Δa 的商和余数；ε、η 是 a 的商和余数。β、δ 是对 y_U 的余数，所以 $\beta + \delta \in [0, 2y_U - 2]$，同理 $\eta + \delta \in [0, 2y_U - 2]$。在计算新的像素灰度时，先计算 Δa_V 的商和余数，即

$$\begin{aligned} \gamma_V &= \alpha + \gamma_U \\ \delta_V &= \beta + \delta_U \end{aligned} \qquad \beta + \delta_U < y_U$$

$$\begin{aligned} \gamma_V &= \alpha + \gamma_U + 1 \\ \delta_V &= \beta + \delta_U - y_U \end{aligned} \qquad \beta + \delta_U \geqslant y_U$$

接下来计算 a_V

$$\begin{cases} \varepsilon_V = \varepsilon_U + \gamma_V \\ \eta_V = \eta_U + \delta_V \end{cases} \qquad \eta_U + \delta_V < y_U$$

$$\begin{cases} \varepsilon_V = \varepsilon_U + \gamma_V + 1 \\ \eta_V = \eta_U + \delta_V - y_U \end{cases} \qquad \eta_U + \delta_V \geqslant y_U$$

得到 a_V 就可以算出像素 V 的灰度了。

若相邻像素如图 4(c)中的情况，同样需要求 a_V，$\frac{c_V - 4}{y_V + 1}$ 的商 ε 和余数 η 可以用式(7)得到，而 $\frac{c_V - 4}{y_V} = \varepsilon + \frac{\varepsilon + \eta}{y_V}$。设 $\frac{\varepsilon + \eta}{y_V}$ 的商和余数是 μ、ν。

由于像素灰度 $g \in [1,64]$,若为图 3(b) 中实线的情况,不难得到 $\mu < 6$。为了避免除法和取整,定义整形变量 k,先求 $k = (\varepsilon + \eta) - (y_V \ll 1) - y_V (\ll 1$ 指左移 1 位)。若 $k < 0$,再求 $k = (\varepsilon + \eta) - (y_V \ll 1)$,若此时 $k \geqslant 0$,则 $\mu = 2, \nu = k$,否则再判断 $(\varepsilon + \eta)$ 与 y_V 的关系,若此时 $(\varepsilon + \eta) < y_V$,则 $\mu = 0, \nu = \varepsilon + \eta$,否则 $\mu = 1, \nu = \varepsilon + \eta - y_V$; 若 $k \geqslant 0$,再求 $k = k - y_V$,若此时 $k < 0$,则 $\mu = 3, \nu = k + y_V$,否则再判断 k 与 y_V 的关系,若此时 $k < y_V$,则 $\mu = 4, \nu = k$,否则 $\mu = 5, \nu = k - y_V$。

图 3(b) 中虚线的情况与之类似,不再叙述。总之,可以通过二、三次比较求得 a_V,从而求出该像素的灰度。而 Δa_V 也可以用类似方法求出,从而为新像素行的灰度递推做好准备。$64/y_V$ 的商 α、余数 β 可以用下式计算:

$$\begin{cases} \alpha_{new} = \alpha_{old} \\ \beta_{new} = \alpha_{old} + \beta_{old} \end{cases} \quad \alpha_{old} + \beta_{old} < y_V$$

$$\begin{cases} \alpha_{new} = \alpha_{old} + 1 \\ \beta_{new} = \alpha_{old} + \beta_{old} - y_V \end{cases} \quad \alpha_{old} + \beta_{old} \geqslant y_V$$

若相邻像素的判定变量如图 4(d) 中的情况,可以用式(8) 得到 $\frac{c_V - 4}{y_V + 1}$ 的商 ε 和余数 η,再用图 4(c) 中的方法得到 $\frac{c_V - 4}{y_V}$。

圆弧第 1 列的两个像素 $(0,R)$、$(0,R-1)$ 的灰度分别为 64 和 1,Δa 的初值为 $\frac{32}{R}$,可以用图 4(c) 中的方法求得,$\frac{64}{y}$ 的初值 $\frac{64}{R}$ 可以一并求出。

按以上推导,可以为灰度建立递推公式,利用 Δa_U,区分图 4 的不同情况,算出灰度改变量的增量,得到 Δa_V,从而算出新的灰度值。

4　算法比较

本文提出的算法基于中点画圆法,每次循环可以求出一对像素的灰度。一次循环需要先求灰度变化量,再求像素的灰度,平均分别需要 3 次加法和 1 次比较。每个像素行的起始像素在计算时平均还需要额外的 5 次比较和一些相应计算,由于这些起始像素数量较少,所以总的计算量不大。

对本文算法进行了各种参数的 64 级灰度圆弧绘制,与利用浮点运算直接计算像素灰度比较,二者的误差不超过 1 个灰度级,反走样效果较好。相比之下,尽管文献 [12-13] 给出了反走样圆的双点算法,计算量较小,但分别只有 3 个和 7 个灰度级,达不到实际应用的程度。文献 [14] 虽然能得到多级像素灰度,但是灰度是粗略的估计值,甚至可能选择错误的像素,且计算量较大。本文的算法通过整数加、位移和比较,建立了相邻像素的灰度递推关系,计算比文献 [14] 简单而效果更好。具体反走样效果比较见图 5,图中自上而下分别是同一段 1/8 圆弧,在未反走样,用文献 [11] 算法和本文算法分别实现后的照片对比。显然本文算法生成的圆弧视觉效果较好。

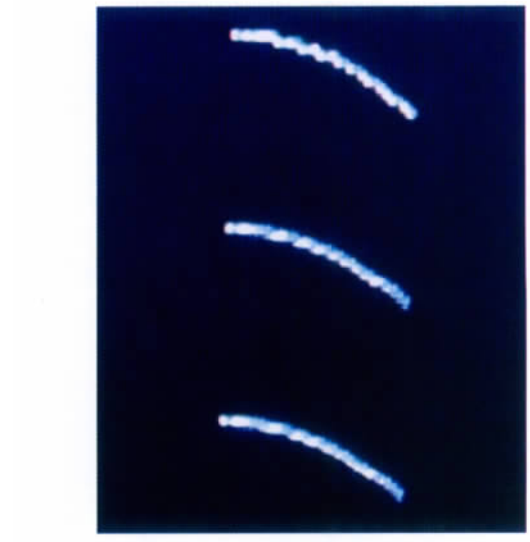

图 5　反走样效果比较

Fig. 5　Compare of the antialiasing effect

5　结　论

提出一种新的圆反走样算法。该算法建立了相邻像素之间坐标和灰度的递推公式,只使用整数移位、加法和比较来生成一条 64 级灰度的反走样圆弧,计算简单,并可以扩展到更多的灰度级。通过运行验证,与原有算法相比,新算法更快地生成反走样圆弧,提高了反走样效果,且新算法可以在诸如 FPGA 的硬件设备上运行。

参考文献(References)

[1]　James D F. Introduction to Computer Graphics [M]. Beijing:

China Machine Press,2004: 48-56. [James D F. 计算机图形学导论[M]. 北京: 机械工业出版社,2004: 48-56.]

[2] Hearn D,Baker M P. Computer Graphics [M]. Beijing: Publish House of Electronics Industry,2005: 70-75. [Hearn D,Baker M P. 计算机图形学[M]. 蔡士杰,宋继强,蔡敏,等译. 北京: 电子工业出版社,2005: 70-75.]

[3] Bresenham J. A linear algorithm for incremental digital display of circular arcs [J]. Communications of the ACM,1977,20(2): 100-106.

[4] Xu X L,Hong B. A sub-pixel regional sampling anti-aliasing algorithm based on integer coordinate [J]. Journal of Image and Graphics,2009,14(12): 2438-2442. [徐小良,洪波. 一种基于整数坐标的亚像素精度区域采样反走样算法[J]. 中国图象图形学报,2009,14(12): 2438-2442.]

[5] Kong L D. Research on area-weighted antialiasing algorithm [J]. Journal of Engineering Graphics,2009,30(4): 49-54. [孔令德. 基于面积加权反走样算法的研究[J]. 工程图学学报,2009,30(4): 49-54.]

[6] Rokita,P. Depth-based selective antialiasing [J]. Journal of Graphics Tools,2005,10(3): 19-26.

[7] Field D. Algorithms for drawing anti aliased circles and ellipses [J]. Computer Vision,Graphics,and Image Processing,1986,33(1): 1-15.

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

[9] Chang S L,Shantz M,Rocchetti R. Rendering cubic curves and surfaces with integer adaptive forward differencing [J]. Computer Graphics,1989,23(3): 157-166.

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

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

[12] Liu Y K,Shi J Y. Double-step circle drawing algorithm with and without grey scale [J]. Journal of Computer-Aided Design & Computer Graphics,2005,17(1): 34-41. [刘勇奎,石教英. 圆的像素级生成及反走样算法[J]. 计算机辅助设计与图形学学报,2005,17(1): 34-41.]

[13] Niu Y J,Tang D. Double-step anti-aliasing drawing algorithm of circle [J]. Computer Engineering and Applications,2010,46(23): 175-178. [牛玉静,唐棣. 双步圆的反走样生成算法[J]. 计算机工程与应用,2010,46(23): 175-178.]

[14] Niu L Q,Shao Z. Unified integral algorithm for anti-aliased lines and typical curves [J]. Journal of Computer-AidedDesign & ComputerGraphics,2010,22(8): 1293-1299. [牛连强,邵中. 直线与典型曲线的统一反走样整数生成算法[J]. 计算机辅助设计与图形学学报,2010,22(8): 1293-1299.]

[15] Chen W. A hardware accelerated anti-aliasing volume Splatting algorithm [J]. Journal of Computer-Aided Design & Computer Graphics,2005,17(4): 677-682. [陈为. 硬件加速反走样体Splatting算法[J]. 计算机辅助设计与图形学学报,2005,17(4): 677-682.]