第3卷 第12期
1998年12月

中国图象图形学报
Journal of Image and Graphics

Vol. 3, No. 12
Dec. 1998

# 图形生成填充方式研究

张胜修 倪雅玲 邓方林
(西安高科技研究所，西安 710025)

**摘 要** 填充是图形生成中关键性的一步，其效率直接影响到整个图形生成系统的性能。本文首先讨论了几种常用的填充算法的局限性，然后提出了一种更有效的凸多边形填充算法，并详细说明了该算法在抗混叠。

**关键词** 图形生成，填充，并行

## 1 引 言

光栅扫描显示设备要求，在显示实体物体时，要给被显示物体所占区域的每个象素赋予一定的显示值，我们称这个过程为填充。多边形填充是几何图形生成系统中重要的步骤之一，它是根据边或顶点的简单描述生成实区域的过程，也可称为实区域填充或轮廓线填充。现在有许多种填充轮廓线的方法，一般可以把它们分成两大类：扫描变换和种子填充。

以上两类算法，由于具有较好的流水线处理等特性，在许多几何图形生成系统中广泛采用。但同时也可以看出，有序边表算法需要较大内存建立和维持边表，且要对它进行排序；而边标志算法在填充时，要对帧缓存进行频繁访问，而从目前来看，高速处理器和低速存储形成了突出矛盾，因此它们存在的缺点，影响了算法的使用效率。另外，对抗混叠、矩形填充的适应性等方面也存在着一定缺陷。凸多边形填充算法试图克服上述缺点，进一步提高图形生成效率。

## 2 填充方式

凸多边形是图元中的重要一类，由于许多复杂的几何形体都可分解成简单的凸多边形，因此在一些高性能图形系统中均采用简单凸多边形作为图形描述基元。

为了建立凸多边形内点判别定理，首先给出在$E^2$空间中直线的半平面方程的概念，并由此导出相应的引理。如图1所示，在2D 空间中，给定有向直线$f(x,y)=0$，则在该直线上所有点$(x_i,y_i)$必满足$f(x_i,y_i)=0(i=1,2\cdots)$，而不在该直线上的点$(x_j,y_j)$必有$f(x_j,y_j)\neq 0(j=1,2\cdots)$。换言之，该直线将2D空间分为2个半平面点集$f(x,y)\geq 0$与$(x,y)\leq 0$(它们通常称为“直线的半平面方程”)。根据直线的单调特性可得如下引理。

**引理**：若任意两点$B,D$(如图1所示)在同一半平面内，当且仅当

$$\mathrm{Sign}[f(B)]=\mathrm{Sign}[f(D)] \tag{1}$$

该引理根据直线的半平面性质和单调性，不难用反证法加以证明。并由引理可直接给出凸多边形内点判断定理。

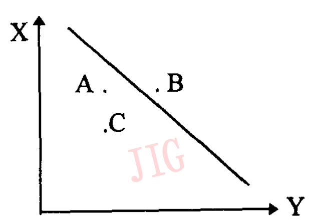



图1

**定理**：假定一凸多边形由$f_i(x,y)=0$ $(i=1,2,\cdots,n)$边界方程构成。(如图2所示)且确定$A$为凸多边

收稿日期：1997-11-10；收到修改稿日期：1998-04-06

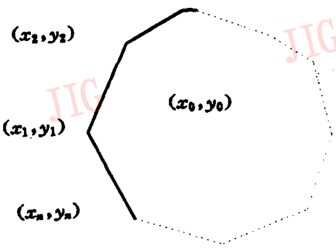



图2

形的内点，则任意点 B 为凸多边形内点的充要条件是：

$$\mathrm{Sign}[f_i(A)]=\mathrm{Sign}[f_i(B)] \quad (i=1,2,\cdots,n) \tag{2}$$

**凸多边形填充算法：**

设：质心坐标$(x_0,y_0)$，已知各顶点的坐标$(x_i,y_i)$　$(i=1,2,\cdots,n)$，则：

$$x_0=\frac{x_1+x_2+\cdots+x_n}{n}$$
$$y_0=\frac{y_1+y_2+\cdots+y_n}{n} \tag{3}$$

$n$ 边的直线方程为：

$$\begin{cases}\dfrac{y_1-y_2}{x_1-x_2}(x-x_1)=y-y_1\\[2mm]\dfrac{y_2-y_3}{x_2-x_3}(x-x_2)=y-y_2\\[2mm]\cdots\quad\cdots\quad\cdots\\[2mm]\dfrac{y_n-y_1}{x_n-x_1}(x-x_n)=y-y_n\end{cases} \tag{4}$$

令：

$$K=\begin{bmatrix}-(y_1-y_2) & (x_1-x_2) & -(x_1-x_2)y_1+(y_1-y_2)x_1\\ -(y_2-y_3) & (x_2-x_3) & -(x_2-x_3)y_2+(y_2-y_3)x_2\\ \cdots & & \cdots\quad\cdots\\ -(y_n-y_1) & (x_n-x_1) & -(x_n-x_1)y_n+(y_n-y_1)x_n\end{bmatrix} \tag{5}$$

方程组(4)可简化为：

$$\begin{cases}k_{11}x+k_{12}y+k_{13}=0\\ k_{21}x+k_{22}y+k_{23}=0\\ \cdots\quad\cdots\quad\cdots\\ k_{n1}x+k_{n2}y+k_{n3}=0\end{cases} \tag{6}$$

$$K\begin{bmatrix}x\\y\\1\end{bmatrix}=0 \tag{7}$$

内点的标志值

$$S=(s_1,s_2,\cdots,s_n)^{\mathrm{T}} \tag{8}$$

令：$x=x_0$　　　$y=y_0$

$$S=\mathrm{sign}K\begin{bmatrix}x\\y\\1\end{bmatrix} \tag{9}$$

判断某点$(x,y)$是否在凸多边形内，也就是说 $\mathrm{sign}K\begin{bmatrix}x\\y\\1\end{bmatrix}$ 是否等于 $S$。

**算法：**

(1)计算凸多边形内点标志值。

(2)判断所扫描到的象素是否满足标志。

由算法推导可看出凸多边形算法有以下特点：

(1)不需要有很大的数据区。

(2)凸多边形的边界是通过由中心点的标准符号组内定的。不需要预先在帧存中写入边界。

(3)对于凸多边形填充，建议采取三角形或凸四边形为宜。这是因为它们具有简单的几何形式，标志符号组分别为3位和4位，且便于计算。如果设填充象素为 $M$，每增加一条边所增加计算时间为 $Q$，那么增加 $N$ 条边，其计算时间增加为 $M\times Q\times N$。因此 $N$ 取太大反而会降低算法的工作效率；再则，任何一个凸多边形都可分割为三角形或凸四边形，因此边数太多也是不必要的。

(4)该算法中多为浮点计算，但并不会降低算法的效率。在过去的微处理器和图形算法硬件实现中，浮点运算，特别是浮点乘、除与定点运算相比，所用时间前者比后者相差几倍甚至几十倍。且前者硬件实现复杂；但 RISC 高速处理器中的浮点计算与定点计算周期基本相同。因此 RISC 技术对图形算法设计影响还是非常大的。

(5)使用浮点计算也可提高图形的绘制效果。

**计算量：**

(1)以三角形为例判断内点象素的一般计算量为：6次浮点乘，6次浮点加和3次判断。

(2)如果采用递推方式，比如简单地按横向递增 $x$ 或纵向递增 $y$ 一个单位，则计算量为：3次浮点加，3次判断。

该算法采用了内定边界标志位的方法，克服了有序边表算法需要较大存储量以建立维持边表和边标志算法对帧缓存频繁访问等缺点，而较充分地利用 RISC 技术，且在这个计算中保持较高精度的浮

点运算，使该算法具有如后面所述的抗混叠等特性。在i860处理器上软件实现时，该算法与有序边表算法执行速度几乎相同。

## 3 抗混叠

在一般三维图形生成系统中，多边形的各矢量点，以及坐标和投影都是以浮点操作进行的，但在以后的屏幕二维剪裁等操作中，均按定点屏幕坐标进行运算，因此可能由于舍入误差造成相邻三角形边界处产生间隙。

由于上述填充方法始终是按浮点进行的，并且已知填充象素点到三角形边界的距离，因此可在设计中设置某一精度以消除间隙。并且该填充方法能进一步消除锯齿现象。

在图形生成系统中，物体表面锯齿效应是常见的一种现象。一般采用根据象素点占用多边形内面积多少来确定象素颜色值的方法来消除。

设平面内任意三角形，其规正三边方程为：

$$\begin{cases} f_1(x,y)=\sin\alpha_1 y+\cos\alpha_1 x+d_1=0 \\ f_2(x,y)=\sin\alpha_2 y+\cos\alpha_2 x+d_2=0 \\ f_3(x,y)=\sin\alpha_3 y+\cos\alpha_3 x+d_3=0 \end{cases} \tag{10}$$

规正三边方程的计算在三角形填充初始阶段完成。

三角形内任意一点到三边的距离为：

$$\begin{cases} l_1=|f_1(x_0,y_0)|=|\sin\alpha_1 y_0+\cos\alpha_1 x_0+d_1| \\ l_2=|f_2(x_0,y_0)|=|\sin\alpha_2 y_0+\cos\alpha_2 x_0+d_2| \\ l_3=|f_3(x_0,y_0)|=|\sin\alpha_3 y_0+\cos\alpha_3 x_0+d_3| \end{cases} \tag{11}$$

在消除锯齿中，采用求面积的方法。

三角边界线段与被填充的象素矩形，一般有三种关系形式。(如图3所示)

其中，$(x_0,y_0)$为一内点，$f_i$和 $l_i$，$\alpha_i$，$(i=1,2,3)$分别为三角形边界方程，内点到该边界方程的距离及该边界方程与横坐标的夹角，均为已知。

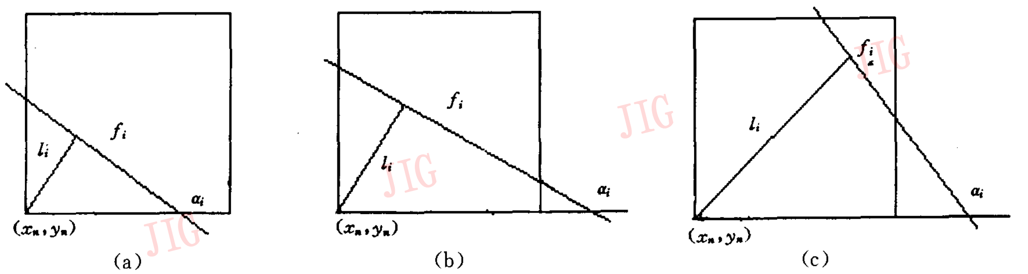



图3

在情况(a)中，根据边界直线所占面积 $S$ 为：

$$S=\left|\frac{-\sin\alpha_i y_0-d_i}{\cos\alpha_i}-x_0\right|\left|\frac{-\cos\alpha_i x_0-d_i}{\sin\alpha_i}-y_0\right|=$$

$$\left|\frac{-\sin\alpha_i y_0-d_i-\cos\alpha_i}{\cos\alpha_i}\right|\left|\frac{-\cos\alpha_i x_0-d_i-\sin\alpha_i y_0}{\sin\alpha_i}\right|=$$

$$\frac{l_i^2}{|\cos\alpha_i\sin\alpha_i|} \tag{12}$$

在情况(b)中，面积 $S$ 为：

$$S=\left|\frac{l_i^2}{\cos\alpha_i\sin\alpha_i}\right|-\frac{(l_i')^2}{|\cos\alpha_i\sin\alpha_i|}=$$

$$\frac{1}{|\cos\alpha_i\sin\alpha_i|}[l_i^2-(l_i')^2] \tag{13}$$

其中，$l_i'$ 为$(x_0+1,y_0)$点到 $f_i$ 的距离。

在情况(c)中，面积 $S$ 为：

$$S=\frac{1}{|\cos\alpha_i\sin\alpha_i|}[l_i^2-(l_i')^2-(l_i'')^2] \tag{14}$$

其中，$l_i'$ 和 $l_i''$ 分别为$(x_0+1,y_0)$和$(x_0,y_0+1)$两点到 $f_i$ 的距离。

由上面3个面积方程计算式可以看出：

(1)$\left|\dfrac{1}{\cos\alpha_i\sin\alpha_i}\right|$在三角形抗混叠填充中不变，因此可在初始化中设定。

(2)面积 $S$ 的计算，与自身象素点及相邻象素点到边界方程的距离平方有关，可以看出填充的判别式与距离方程组计算相同，因此增加抗混叠功能后，填充计算量没有增加很多。

**计算量估计：**

按式(14)最大工作量计算，一般象素的计算量为3次浮点加，4次浮点乘。

如果采用递推形式，计算量为3次浮点加，2次浮点乘。

由于该算法的灵活性，因此它可在可变矩形填充和解决三角形频繁调用等问题上有较好的适应性，篇幅所限不再赘述。

## 参 考 文 献


1 Nader Gharachorloo, Satish Gupta, Robert F. Sproull Ivan, E. Sutherland. A Charaterization of Ten Rasterization Techniques. Computer Graphics(ACM), 1989,23(3):355～368.

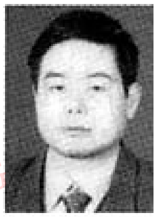
2 罗杰斯. D F 计算机图形学的算法基础. 北京:科学出版社,1987.

3 Juan Pineda. A Parallel Algorithm for Polygon Rasterization. Computer Graphics (ACM)1988,22(4):17～20.

4 Douglas Voorhies. Reduced-Complexity Graphics. IEEE Computer Graphics & Application,1989,63～70.

5 D. T. Harper III. A Multiaccess Frame Buffer Architecture. IEEE TRAN. ON Computer,1994,43(55):618～622.


**倪雅玲**　讲师,主要研究方向:图形图象处理、多媒体技术。



**张胜修**　博士后,主要研究方向:计算机视景生成、多媒体技术。

**邓方林**　教授、博导,主要研究方向:系统仿真,并行处理。

## Reseach of Filling in the Graphic Generation


Zhang Shengxiu, Ni Yaling, Deng Fanglin

(*Xian Research Inst. of Hi-Tech, Xian* 　710025)



**Abstract**　Filling is an essential part in the graphic generation. Its effecency decides the performance of the whole system. At first this dissertation discusses the limities of usual filling algorithms,finally presnets a new more effective algorithm of filling convex polygon,and explains in detail the algorithm applied in antialiasing.



**Keywords**　Graphic generation, Filling, Parallel