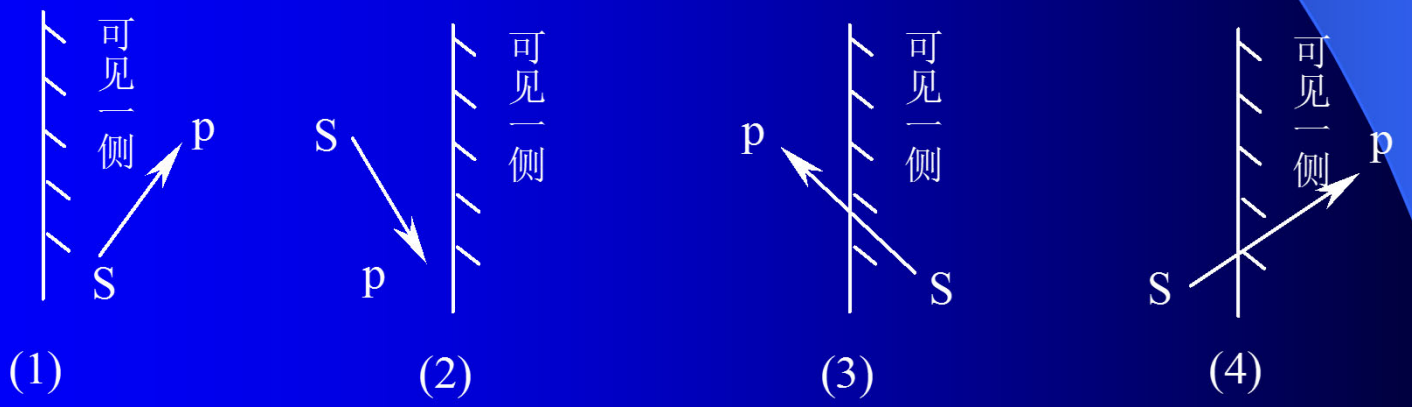
2.5.2 多边形裁剪

基本思想是一次用窗口的一条边裁剪多边形。考虑窗口的一条边以及延长线构成的裁剪线把平面分成两个部分：可见一侧；不可见一侧

多边形的各条边的两端点S、P。它们与裁剪线的位置关系只有四种



对于情况（1）仅输出顶点*P*；情况（2）输出0个顶点；

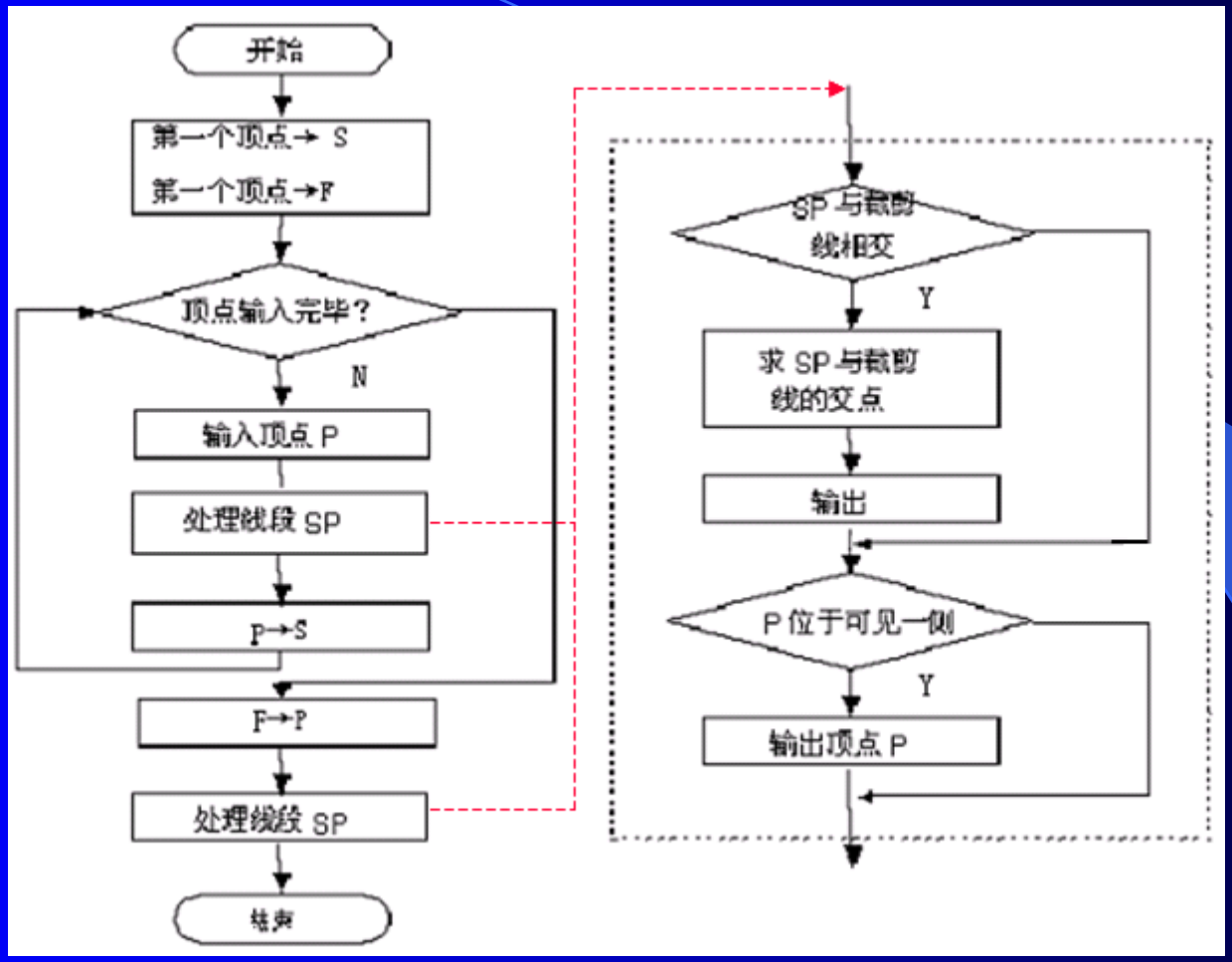
情况（3）输出线段*SP*与裁剪线的交点*I*;

情况（4）输出线段*SP*与裁剪线的交点*I*和*P*

- 上述算法仅用一条裁剪边对多边形进行裁剪，得到一个顶点序列，作为下一条裁剪边处理过程的输入。
- 对于每一条裁剪边，算法过程同上，只是判断点在窗口哪一侧以及求线段*SP*与裁剪边的交点算法应随之改变。

逐次多边形裁剪算法框图

多边形裁剪过程



处理线段*SP*过程子框图

2.5.3 字符裁剪

串精度: 将包围字串的外接矩形对窗口作裁剪

字符精度: 将包围字的外接矩形对窗口作裁剪

笔画\象素精度: 将笔划分解成直线段对窗口作裁剪

待裁剪字符串　　串精度裁剪　　字符精度裁剪　　象素精度裁剪

2.6 反走样 (1)

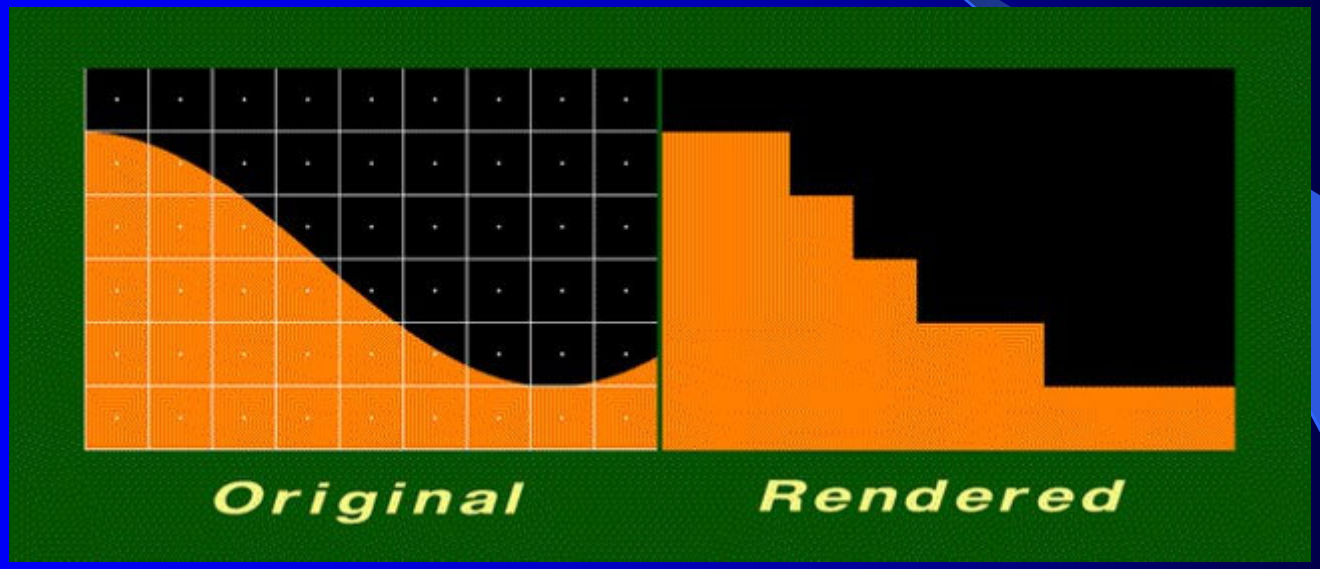
用离散量表示连续量引起的失真现象称之为**走样(aliasing)**，用于减少或消除这种效果的技术称为**反走样(antialiasing)**

2.6.1 提高分辨率

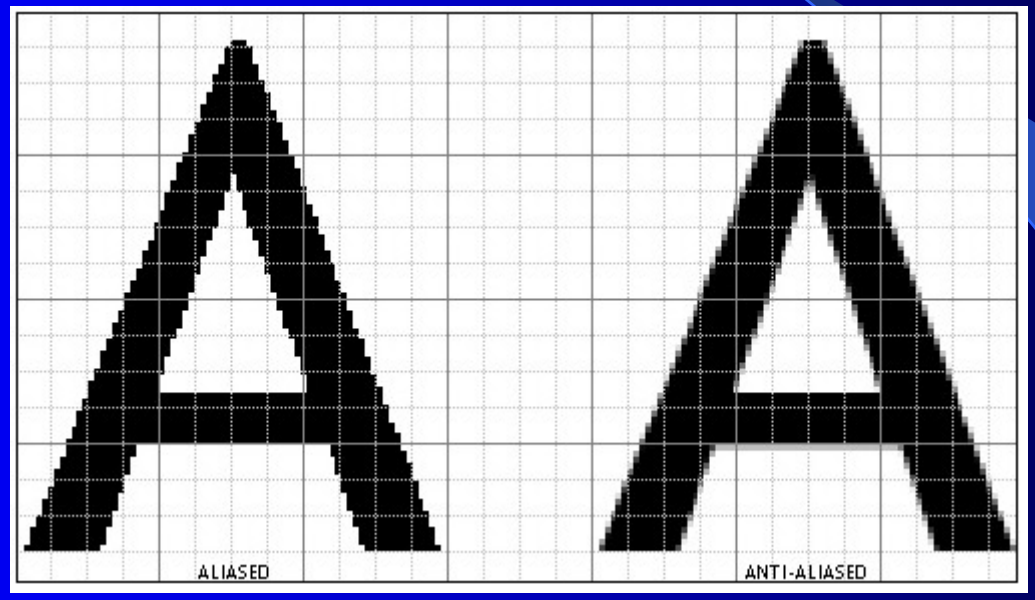
2.6.2 区域采样

2.6.3 加权区域取样

2.6 反走样 (2)



2.6 反走样 (3)



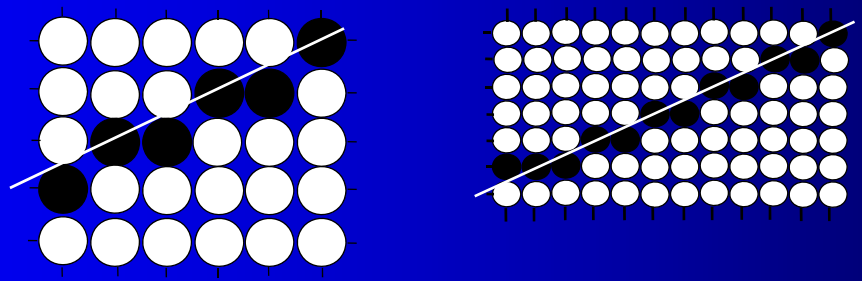
2.6 反走样 (4)

2.6.1 提高分辨率

把显示器分辨率提高一倍，

- 直线经过两倍的象素，锯齿也增加一倍，
- 但同时每个阶梯的宽度也减小了一倍，
- 所以显示出的直线段看起来就平直光滑了一些。

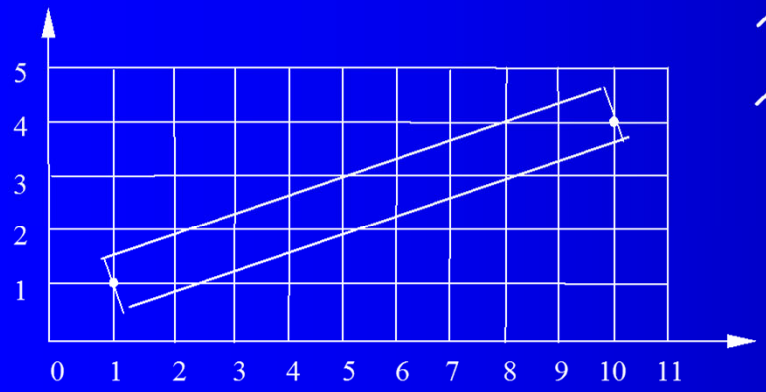
增加分辨率虽然简单，但是不经济的方法，而且它也只能减轻而不能消除锯齿问题



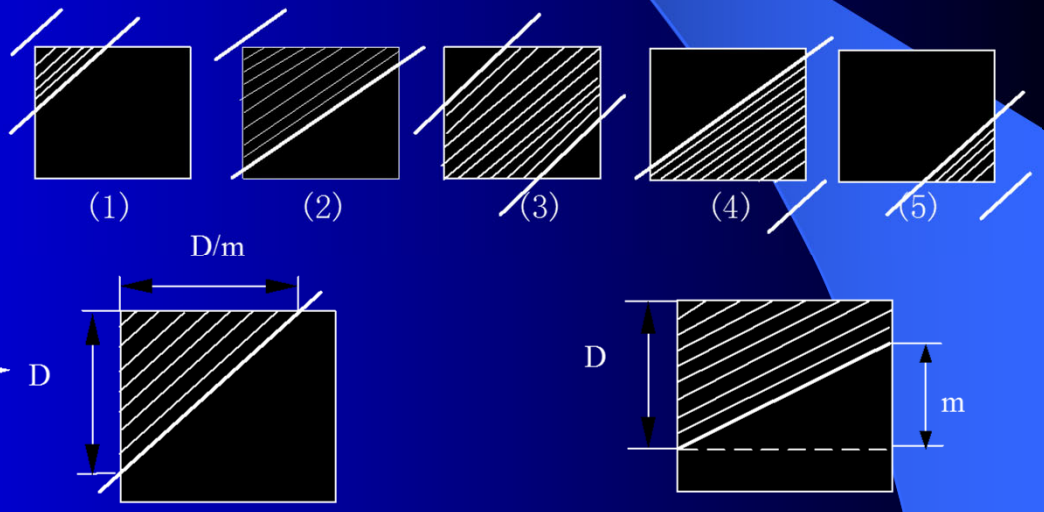
2.6.2 区域采样

基本思想:

– 每个象素是一个具有一定面积的小区域，将直线段看作具有一定宽度的狭长矩形。当直线段与象素有交时，求出两者相交区域的面积，然后根据相交区域面积的大小确定该象素的亮度值。



有宽度的线条轮廓



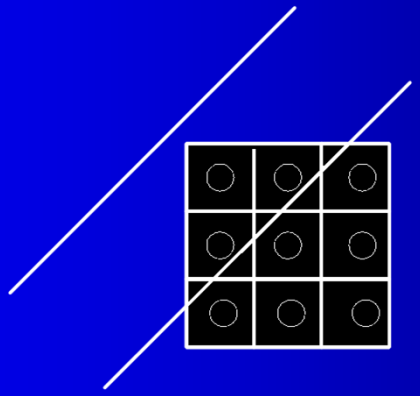
象素相交的五种情况及用于计算面积的量

面积计算

- 情况(1)、(5)阴影面积为：$D^2/(2m)$;
- 情况(2)、(4)阴影面积为：$D - m/2$;
- 情况(3)阴影面积为：$1 - D^2/m$

m是直线段的斜率

为了简化计算可以采用离散的方法



$n = 9, k = 3$近似面积为1/3

- 首先将屏幕象素均分成n个子象素，
- 然后计算中心点落在直线段内的子象素的个数k。
- 将屏幕该象素的亮度置为相交区域面积的近似值可k/n。

非加权区域采样方法有两个缺点：

- 象素的亮度与相交区域的面积成正比，而与相交区域落在象素内的位置无关，这仍然会导致锯齿效应。
- 直线条上沿理想直线方向的相邻两个象素有时会有较大的灰度差。

2.6.3 加权区域取样

基本思想:

- 使相交区域对象素亮度的贡献依赖于该区域与象素中心的距离
- 当直线经过该象素时，该象素的亮度F是在两者相交区域A'上对滤波器（函数w）进行积分的积分值。

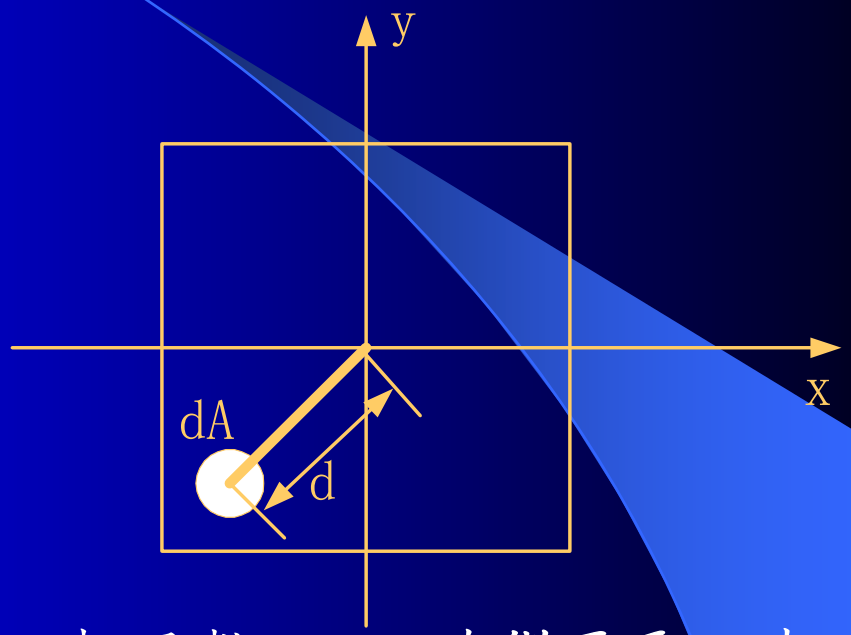
$$w(x,y)=\frac{1}{\sqrt{2\pi\sigma}}e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad F=\int_{A'} w(x,y)dA$$

滤波器函数w可以取高斯滤波器

可采用离散计算方法

– 如：我们将屏幕象素划分为 $n=3\times3$个子象素，加权表可以取作：

$$\begin{bmatrix} w1 & w2 & w3 \\ w4 & w5 & w6 \\ w7 & w8 & w9 \end{bmatrix} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$



权函数$w(x, y)$为微面元dA与象素中心距离d的函数

– 然后求出所有中心落于直线段内的子象素。

– 最后计算所有这些子象素对原象素亮度贡献之和 $\sum_{i\in\Omega} w_i$ 乘以象素的最大灰度值作为该象素的显示灰度值。