第四章　混淆与反混淆方法

本章的主要内容

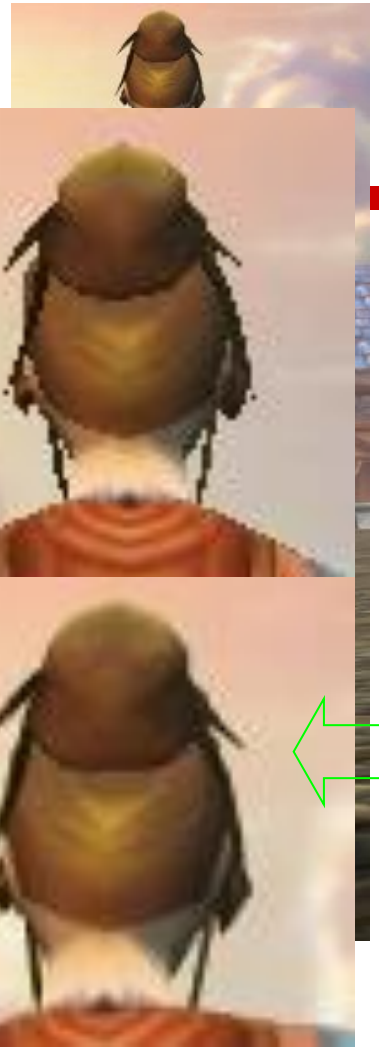
- 混淆现象
- 反混淆方法
  - 提高分辨率
  - 非加权区域采样
  - 加权区域采样

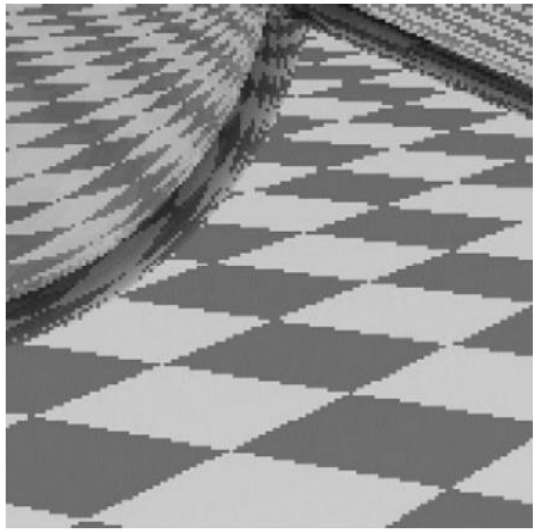
掌握要点

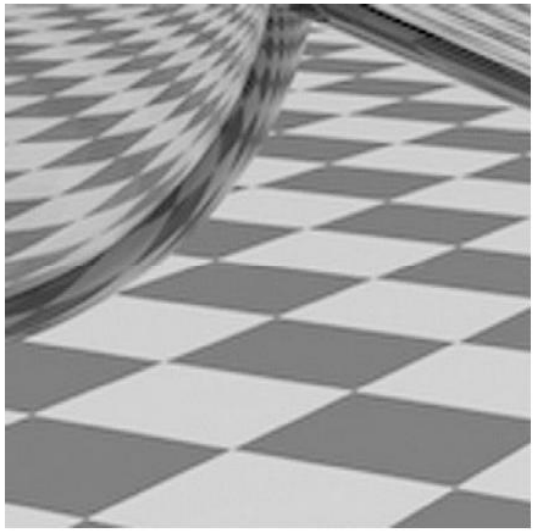
- 掌握在光栅图形绘制过程中常见的三种混淆现象，即图形边界呈阶梯状、图形细节失真、狭小图形遗失；
- 掌握常用的三种反混淆方法：提高分辨率方法、非加权区域采样方法、加权区域采样方法；

4.1 混淆现象

- 混淆： 在光栅显示器上显示图形时，直线段或图形边界或多或少会呈锯齿状。原因是图形信号是**连续**的，而在光栅显示系统中，用来表示图形的却是一个个**离散**的象素。这种用**离散量表示连续量**引起的失真现象称之为混淆(或走样:aliasing)
- 采样频率小于信号频率，导致采样后信号在频域发生了交错、重叠
- 用于减少或消除这种效果的技术称为反混淆(反走样: antialiasing)。



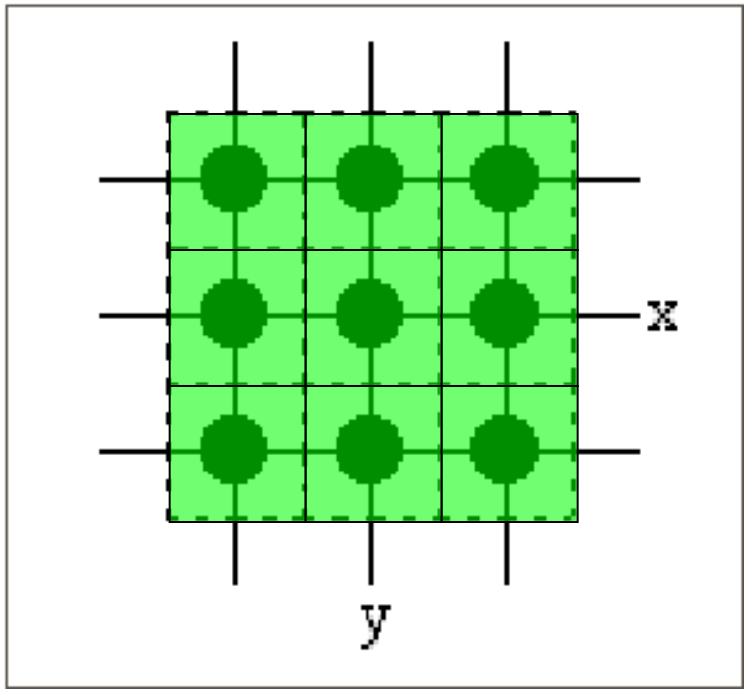




- 光栅图形的混淆现象
  - 阶梯状边界；
  - 图形细节失真；
  - 狭小图形遗失：动画序列中时隐时现，产生闪烁。
- 常用的反走样方法主要有：提高分辨率、区域采样和加权区域采样。

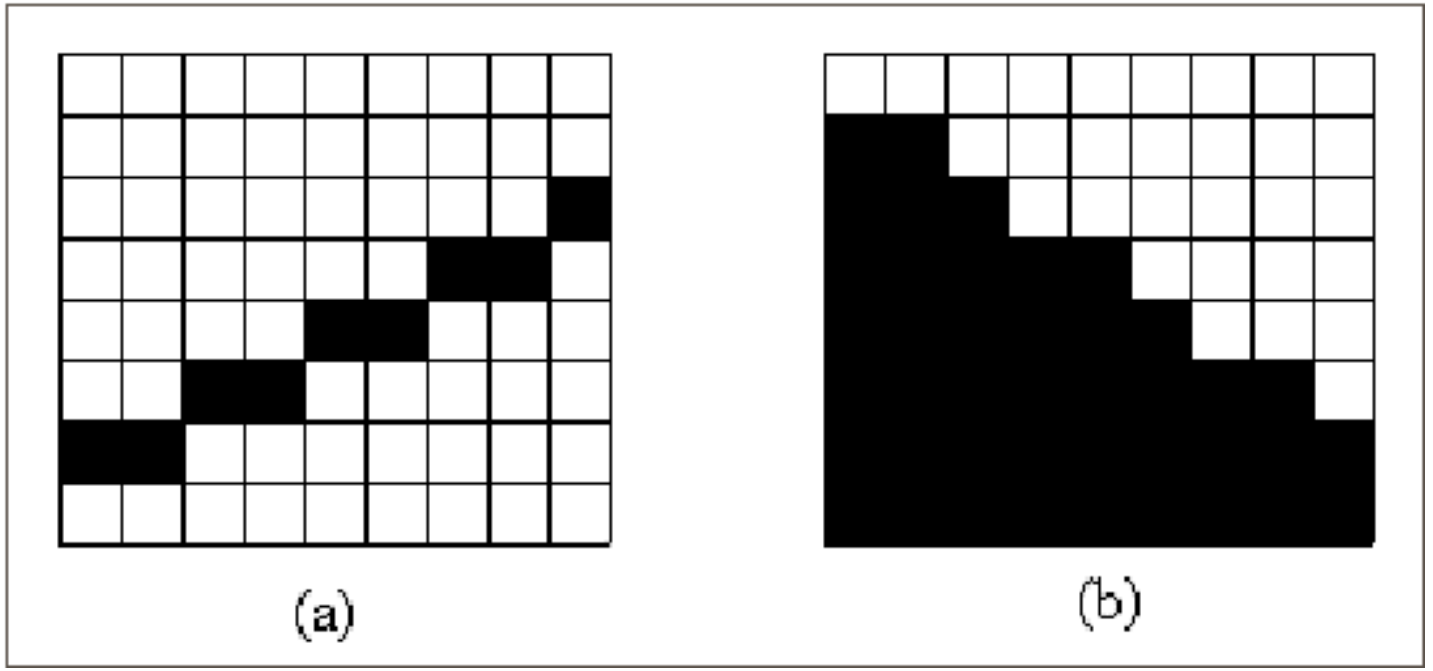
二维光栅图形的混淆现象

- 为了方便说明问题，这儿我们把像素看成中心为坐标点、边长为1的正方形，如图。



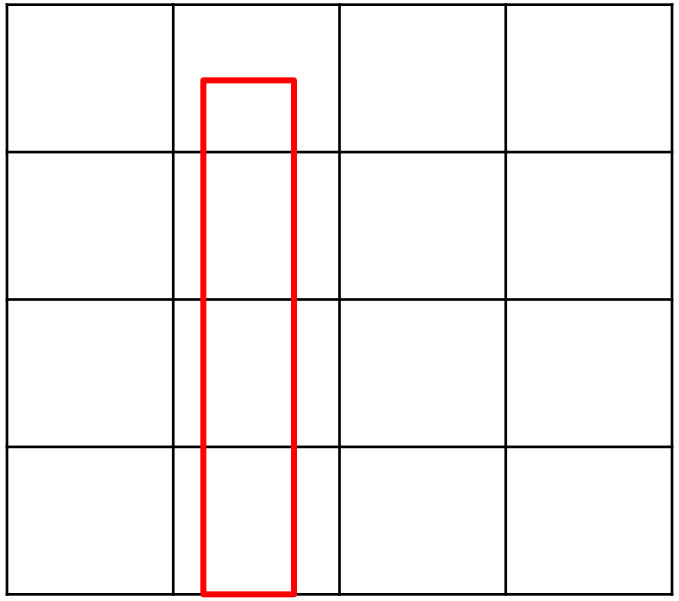
不光滑(阶梯状)的图形边界

直线　　填充多边形



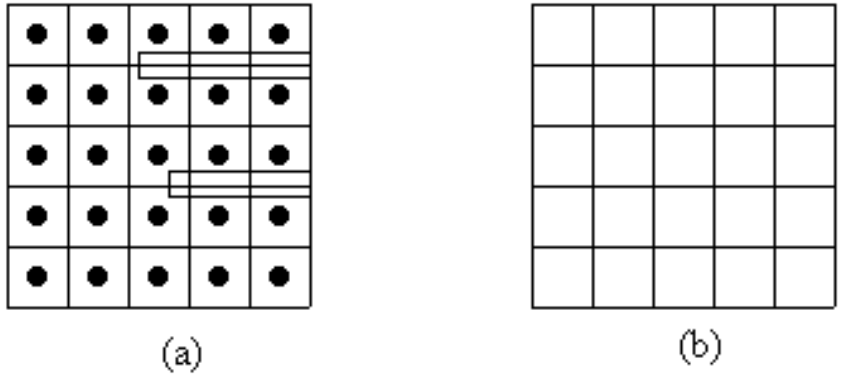
图形细节失真

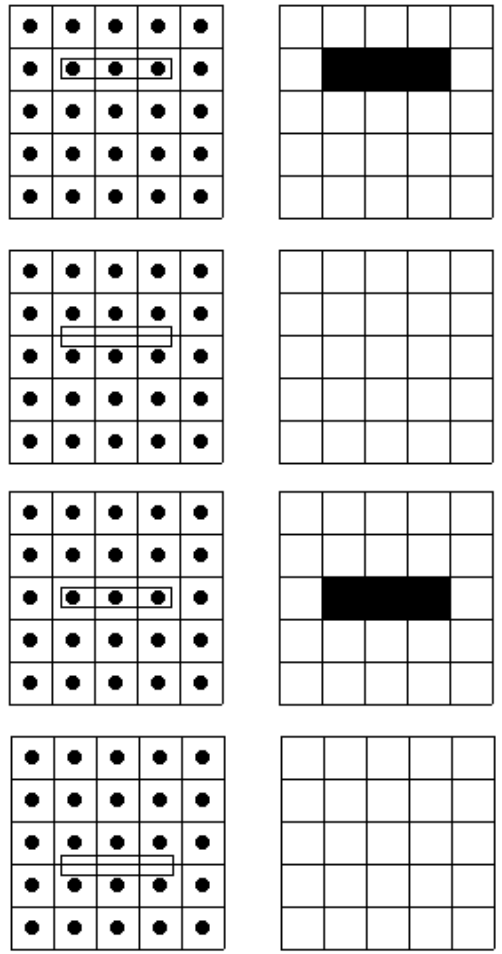
- 细长的矩形显示成为加宽的矩形，细节丢失。



狭小图形的遗失与动态图形的闪烁

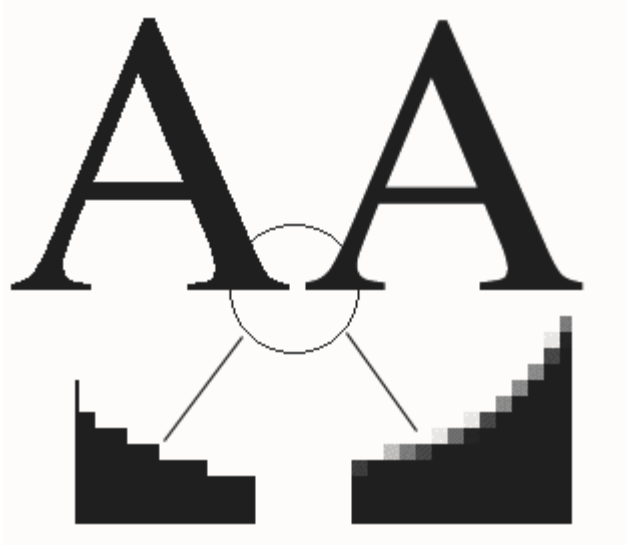
- 下图：狭小图形丢失
- 右图：细小矩形在移动过程中的闪烁





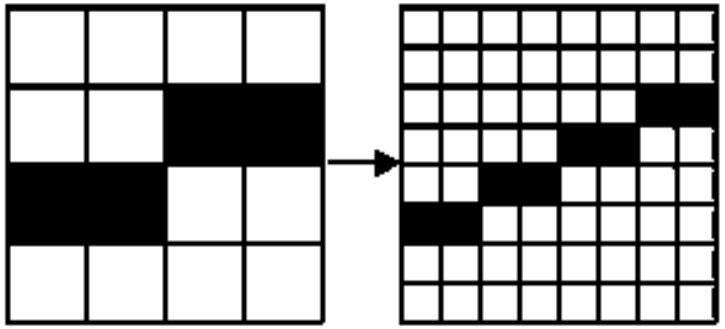
反混淆方法

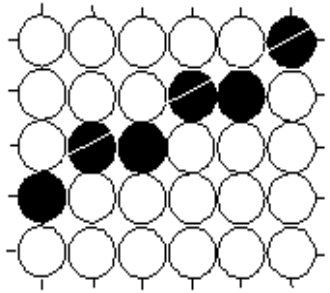
- 什么是反混淆
  - 在图形显示过程中，用于减少或消除混淆现象的方法
  - 三种方法
    - 提高分辨率方法
    - 非加权区域采样
    - 加权区域采样

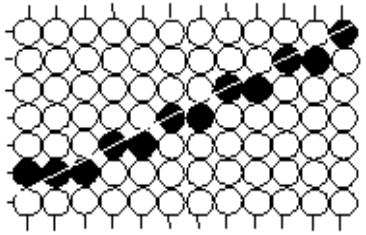


反混淆方法：提高分辨率

- 假如把显示器分辨率（水平、垂直）提高一倍，直线经过两倍的象素，线段上的阶梯锯齿也增加一倍，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。
- 帧缓存容量则增加到原来的4倍，而扫描转换同样大小的图元却要花4倍时间。
- 评价：方法简单，但代价非常大，不经济。

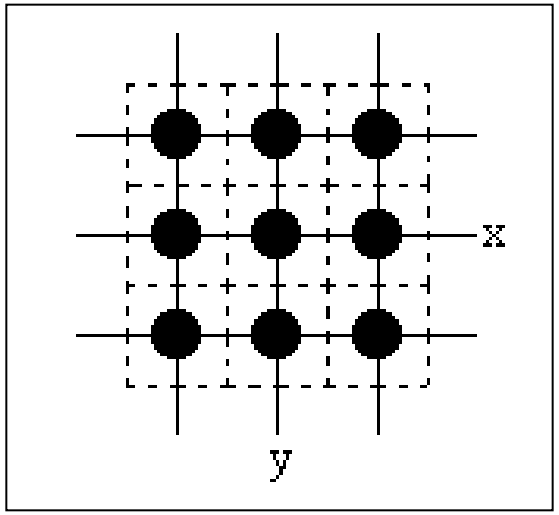






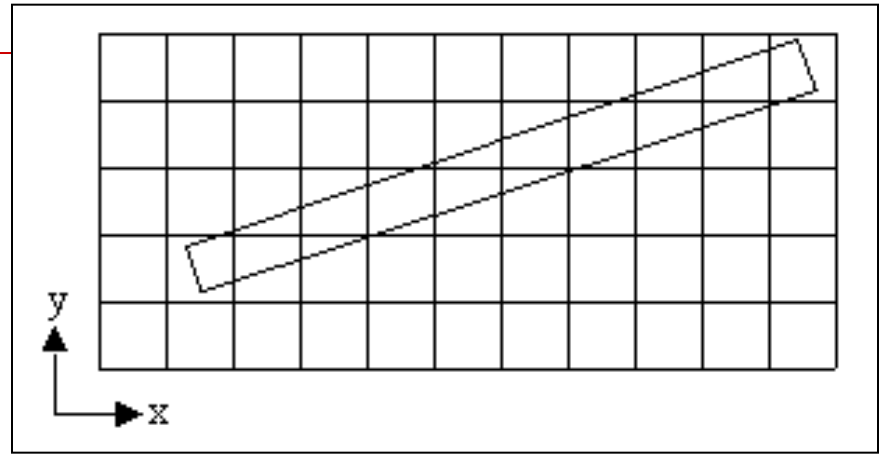
非加权区域采样方法

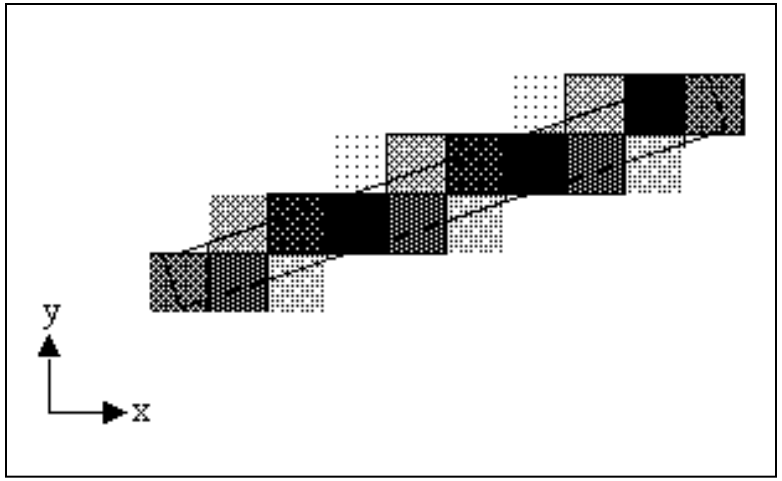
- 理想状态的两点假设
  - 象素是数学上抽象的点，它的面积为0，它的亮度由覆盖该点的图形的亮度所决定；
  - 直线段是数学上抽象直线段，它的宽度为0。
- 现实是残酷的……
  - 像素的面积不为0；
  - 直线段的宽度至少为1个像素；
- 假设与现实的矛盾是导致混淆出现的原因之一



非加权区域采样方法

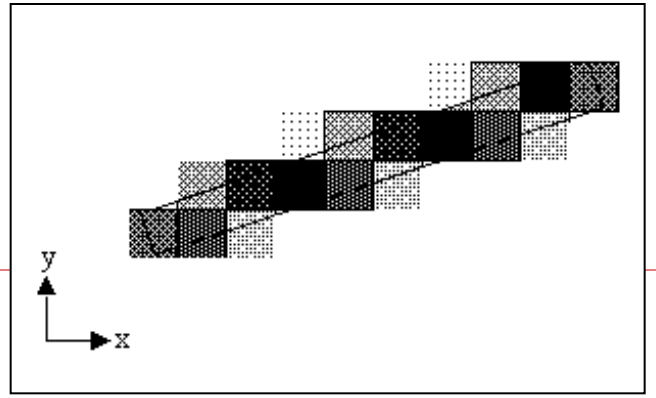
- 解决方法：改变直线段模型
- 方法步骤：
  - 将直线段看作具有一定宽度的狭长矩形；
  - 当直线段与某象素有交时，求出两者相交区域的面积；
  - 根据相交区域的面积，确定该象素的亮度值





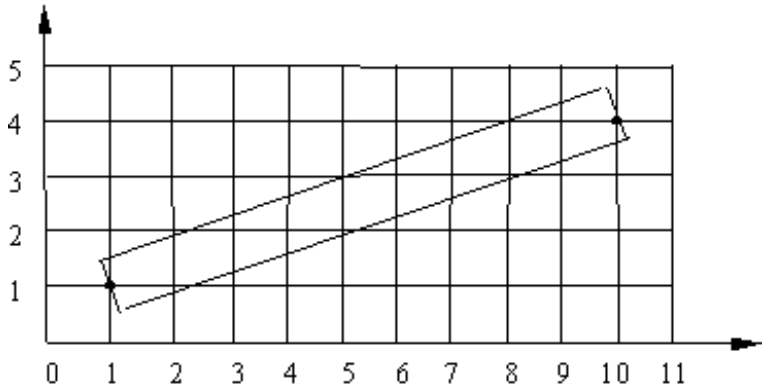
考虑：

- 直线段对一个像素亮度的贡献：
  - 与两者相交区域的面积成正比
  - 和像素中心点距直线段的距离成反比(因为像素中心点距直线段距离越远，相交区域的面积越小)；
- 当直线段和某个像素不相交时，它对该像素的亮度无影响；
- 相同面积的相交区域对像素的亮度贡献相同，而与这个相交区域落在像素内的位置无关。
- 关键：如何计算这个面积?

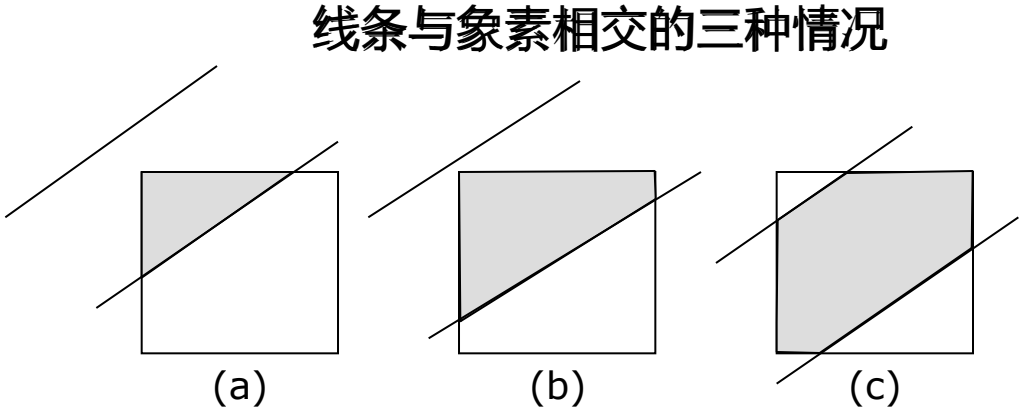


计算相交区域的面积

- 假设一条直线段的斜率为k(0≤k≤1)，且所画直线为一个象素单位，则直线段与象素相交有三种情况。



有宽度的线条轮廓

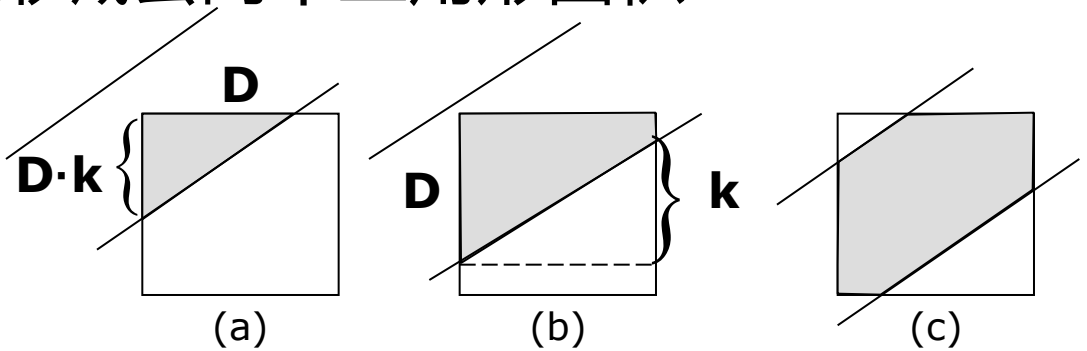


计算相交区域的面积

a) 面积=(k*D*D)/2

b) 面积=(D+D-k)*1/2=D-k/2

c) 转化为正方形减去两个三角形面积



- 像素实际显示的灰度值=所得面积(0-1间)*该像素的最大灰度值

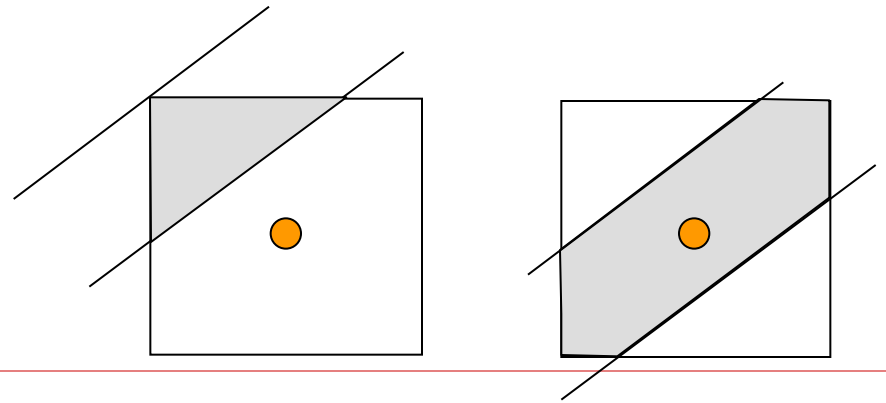
求相交区域的近似面积的离散计算方法

多重采样反走样（Multi-Sample Anti-Aliasing，MSAA）

1. 将屏幕象素分割成n个更小的子象素；
2. 计算中心落在直线段内的子象素的个数，记为k；
3. k/n为线段与象素相交区域面积的近似值

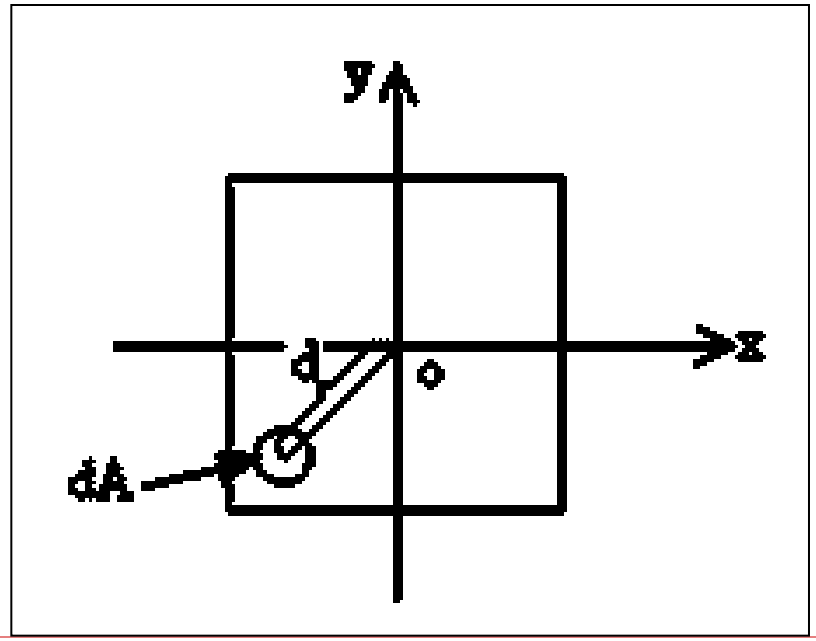
- 目的：简化计算

- 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。



加权区域采样方法

- 想法：改进非加权区域采样方法的第3条性质，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。



权函数W(x,y)

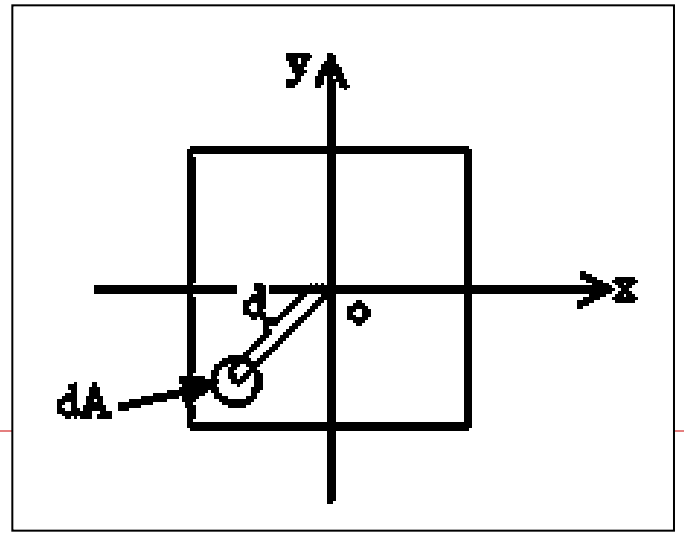
- 以象素A的中心为原点建立二维坐标系
- w(x,y)反应了微面积元dA对整个象素亮度的贡献大小，与d成反比。d越大，权越小。
- 位于(x,y)处的微面积元dA对像素的亮度的贡献为w(x,y)dA，有：

$$w(x,y) \propto \frac{1}{d}$$

$$\int_A w(x,y)dA = 1$$

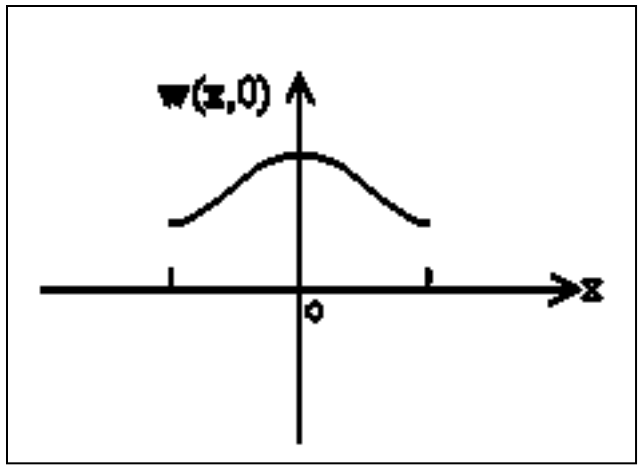
- 相交区域 A'对该象素的亮度贡献

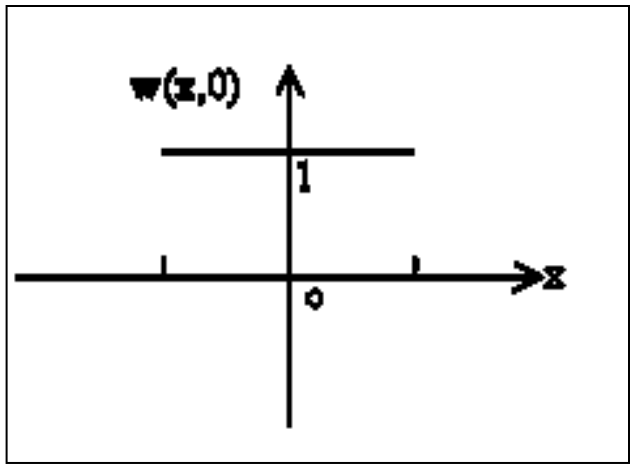
$$\int_{A'} w(x,y)dA$$



- 权函数可以取高斯函数(左)或恒等函数(右)：
  - 取恒等函数时,加权区域采样方法退化为非加权区域采样方法

$$w(x,y)\equiv 1 \qquad \int_{A'} w(x,y)dA = A'\text{的面积}$$





加权区域采样方法实现步骤

1. 求直线段与象素的相交区域 A'；
2. 计算的值 $\int_{A'} w(x,y)dA$ ；
3. 上面所得到的值介于0、1之间，用它乘象素的最大灰度值，即设该象素的显示灰度。

- 问题：计算量大

加权区域采样方法的离散计算方法

- 求积分的运算量是很大的。为此可采用离散计算方法。
- 首先将象素均匀分割成n个子象素。则每个象素的面积为1/n。计算每个子象素对原象素的贡献，并保存在一张二维的加权表中。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的值。该值乘以象素的最大灰度值作为该象素的显示灰度值。

1	2	1
2	4	2
1	2	1

离散计算方法的步骤

1. 将屏幕象素均匀分割成m个子象素 $\{A_i\}_{i=1}^{m}$ ，则每个子象素的面积为 $\int_{A_i} dA = \frac{1}{m}$，计算每个子象素对原象素亮度的贡献，记为$w_i$,将 $\{w_i\}_{i=1}^{m}$ 保存在一张加权表中;
2. 求出所有中心落于直线段内的子象素，记为 $\{A_i : i \in \Omega\}$ ，Ω为{1,2,...,n}的子集。
3. 计算所有这些子象素对原象素亮度贡献之和 $\sum_{i\in\Omega} w_i$ ，该值乘以象素的最大灰度值即为象素的显示灰度值。

加权表的取法

- 对n=9的情况：

$$\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

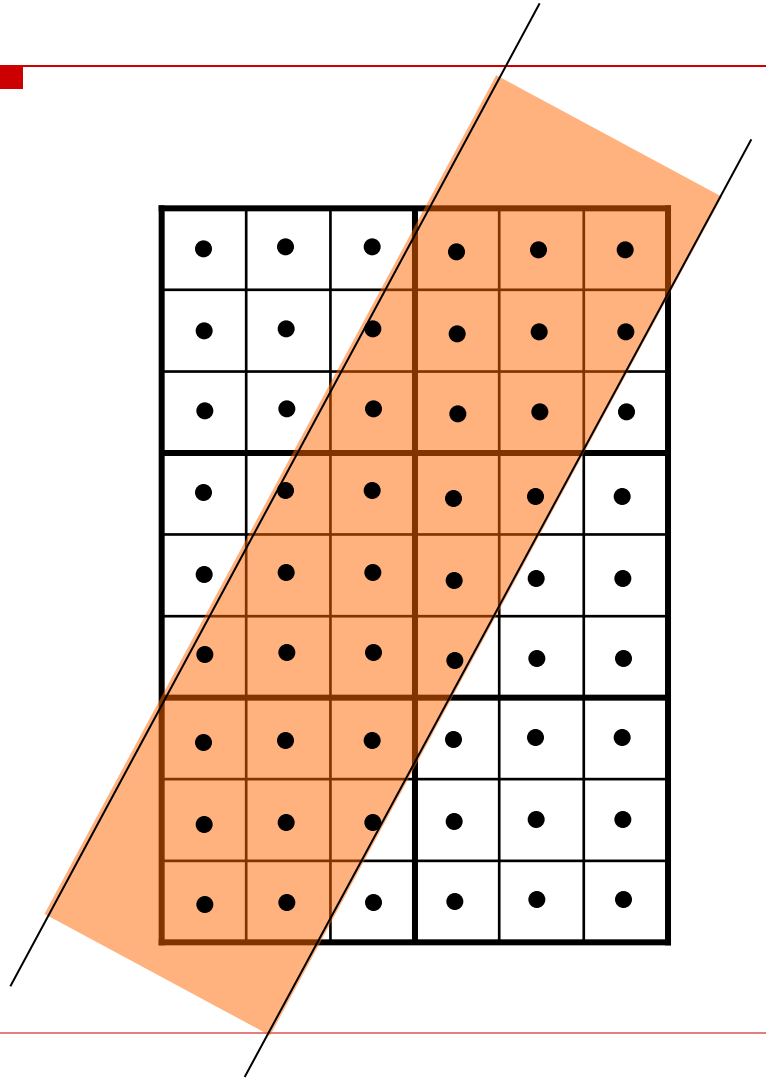
- 对n=25的情况：

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 & w_5 \\ w_6 & w_7 & w_8 & w_9 & w_{10} \\ w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{16} & w_{17} & w_{18} & w_{19} & w_{20} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \end{bmatrix} = \frac{1}{88}\begin{bmatrix} 1 & 2 & 4 & 2 & 1 \\ 2 & 5 & 6 & 5 & 2 \\ 4 & 6 & 8 & 6 & 4 \\ 2 & 5 & 6 & 5 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{bmatrix}$$

离散计算方法举例

- 已知：屏幕亮度级别为256级(0～255)。右侧有6个屏幕象素，每一个都被分为9个子象素；橙色的是需要绘制的直线。

  求：为绘制这条直线，每个象素的亮度应为多少?



其他反走样算法

- 时域反走样（Temporal antialiasing，TAA）
  - 使用前帧信息来提升图像质量的技术。每帧通过一个很微小的平移，在像素内得到不同的采样位置。当产生的图像数量越多，其均值得到的结果就越好。
  - 商业引擎最流行的几种反走样算法之一。
- 快速近似反走样（Fast Approximately -Aliasing，FXAA）
  - 先进行边缘检测，然后通过提取边缘像素周围的颜色信息，通过混合颜色信息来消除高对比所产生的锯齿，其实就是对图像边缘进行柔化。
- 深度学习超采样（Deep Learning Super Sampling，DLSS）
  - Nvidia公司，利用神经网络的重建能力进行图像处理。DLSS 背后使用的技术是 Recurrent CNN。

END