计算机图形学

第五章 反走样和裁剪方法

颜波

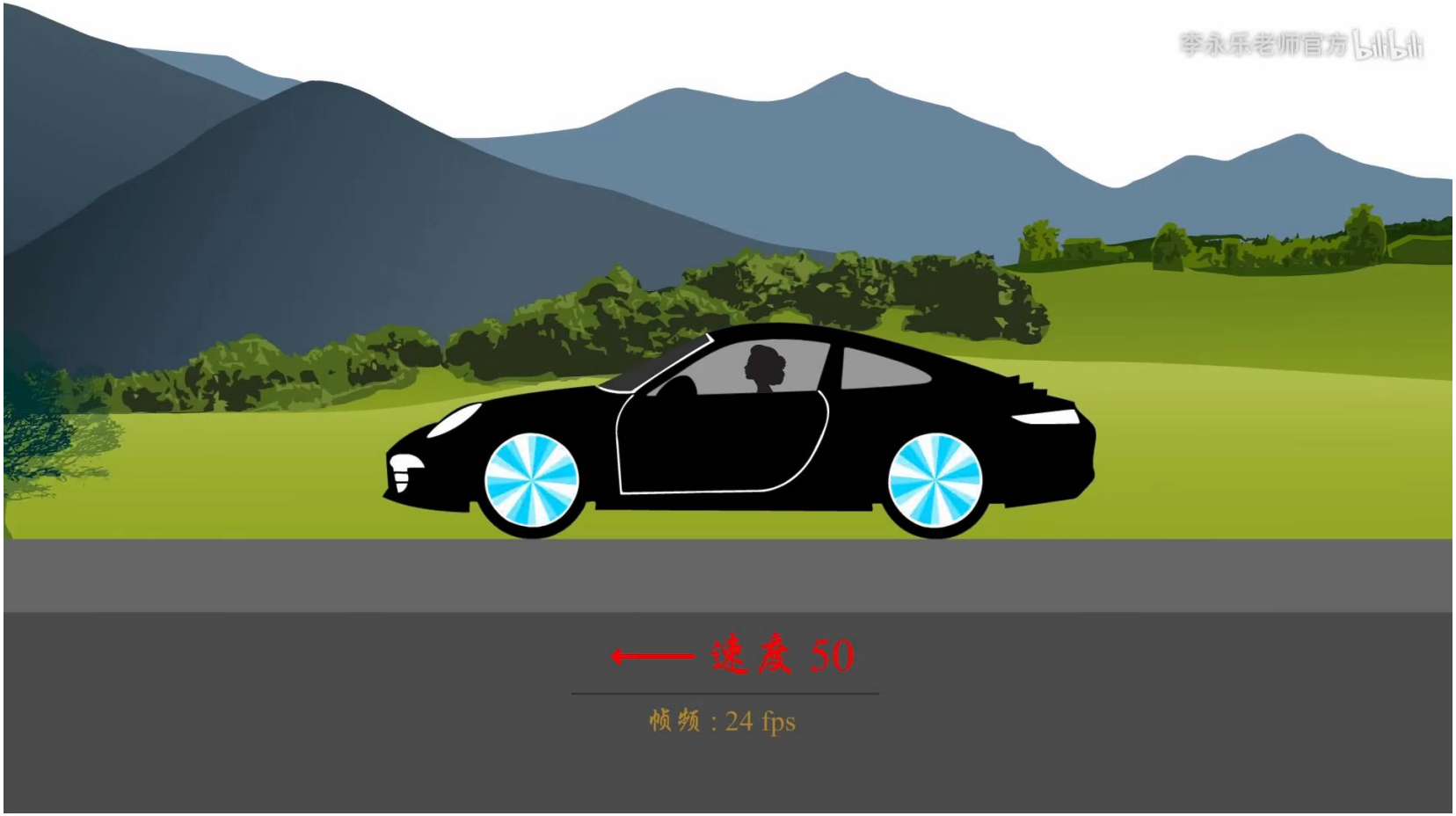
复旦大学计算机科学技术学院
byan@fudan.edu.cn

本章概述

- 抽样定理和反走样方法
- 裁剪算法

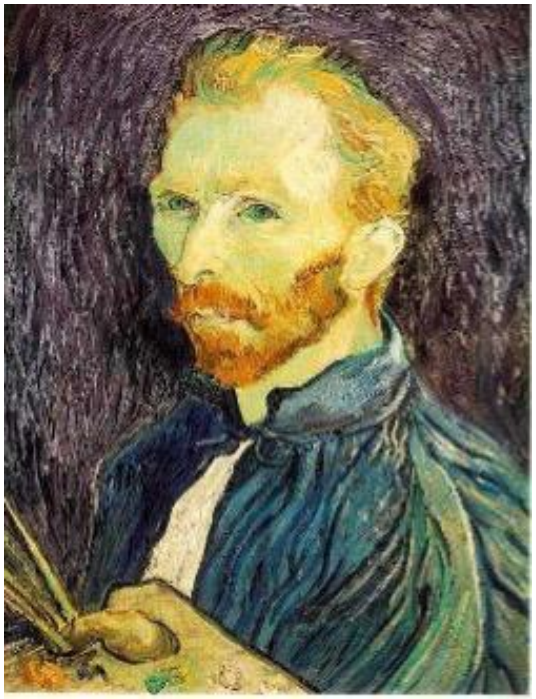
抽样定理和反走样方法

视频中的混叠



https://www.bilibili.com/video/BV12W41197Xi/?spm_id_from=333.337.search-card.all.click

图像下采样



1/2

删除
偶数行、偶数列

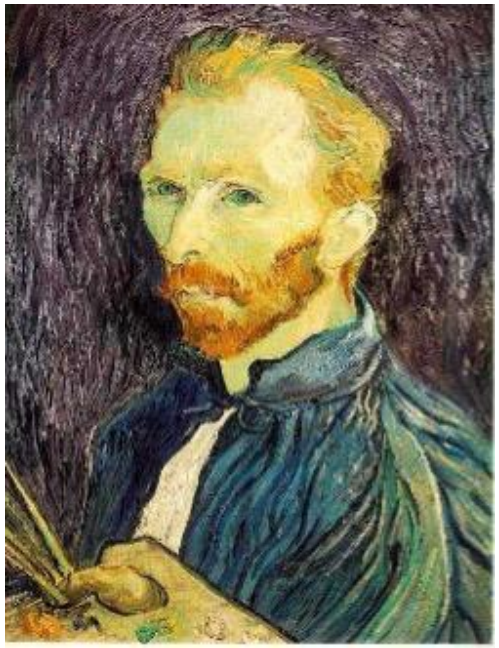
1/4

删除
偶数行、偶数列

1/8

有什么问题?

图像下采样



1/2

1/4

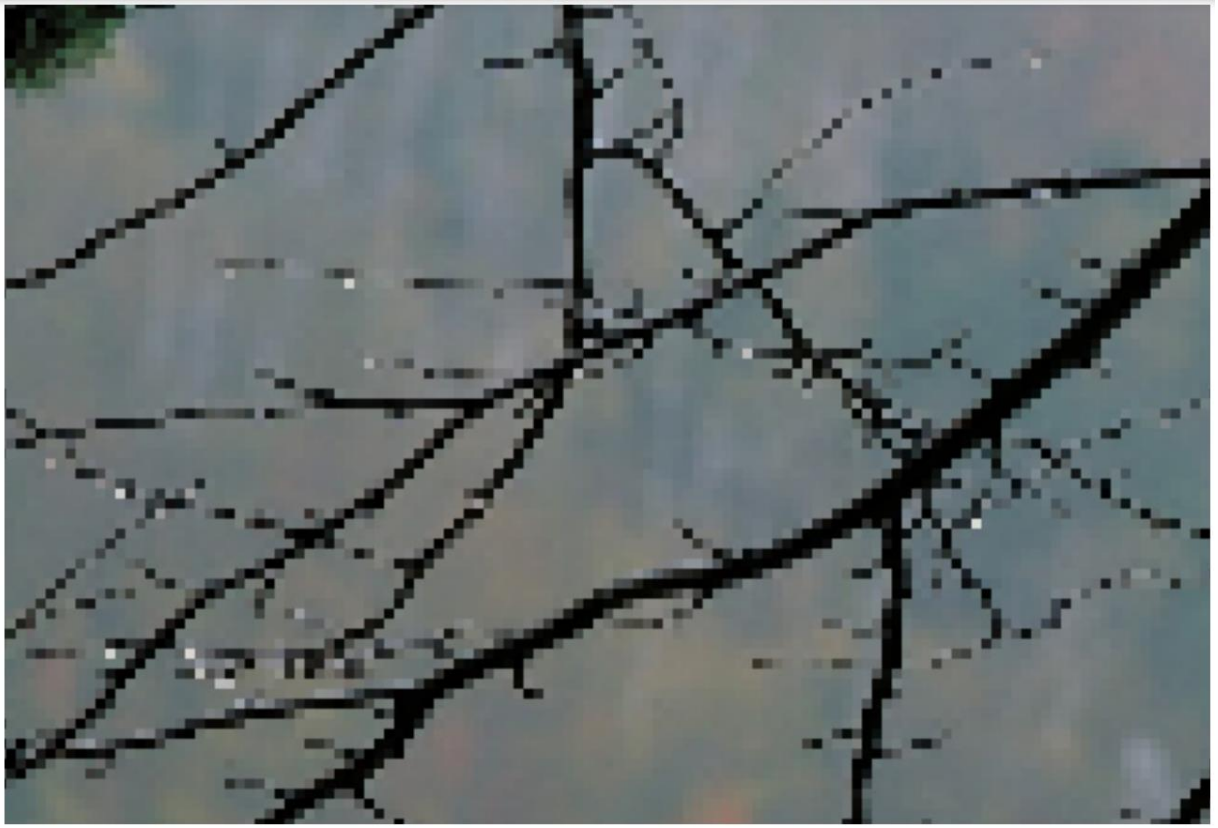
1/8

为什么1/8图像的像素混叠、锯齿现象严重?

[Philip Greenspun]

1000 pixel width

by dropping pixels



模拟视频数字化模型

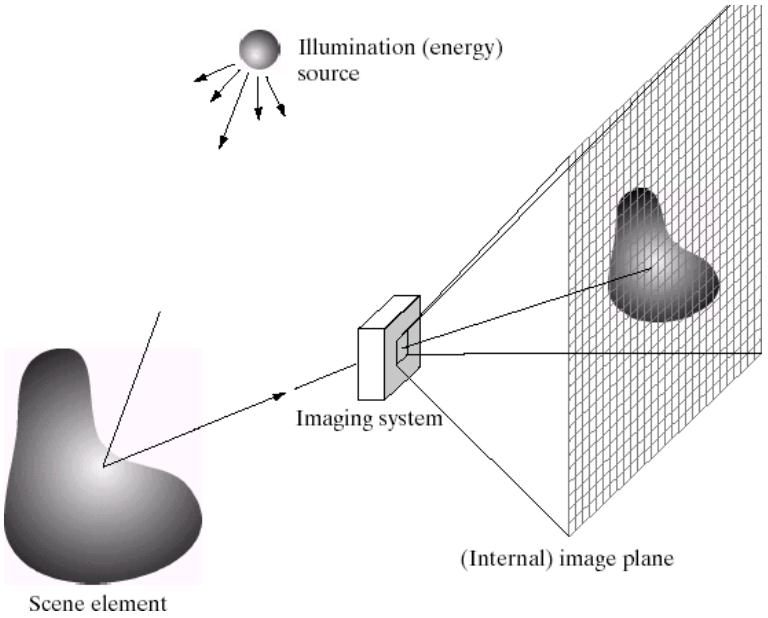
- 模拟视频数字化的必要性：
  - 计算机只能处理数字信号；
  - 数字传输和存储的需要

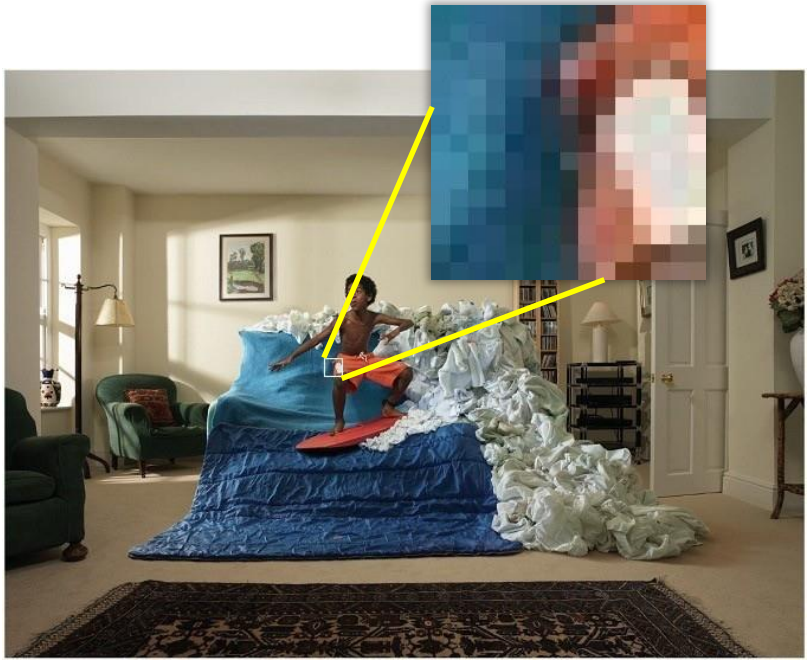
- 数字化过程的方案：
  - 数字摄像机直接对连续场景数字化数字视频信号；
  - 模拟摄像机得到的连续模拟视频信号进行取样和量化；

模拟视频信号数字化模型

模拟视频数字化模型

- 图像是对连续世界的离散采样结果





抽样定理

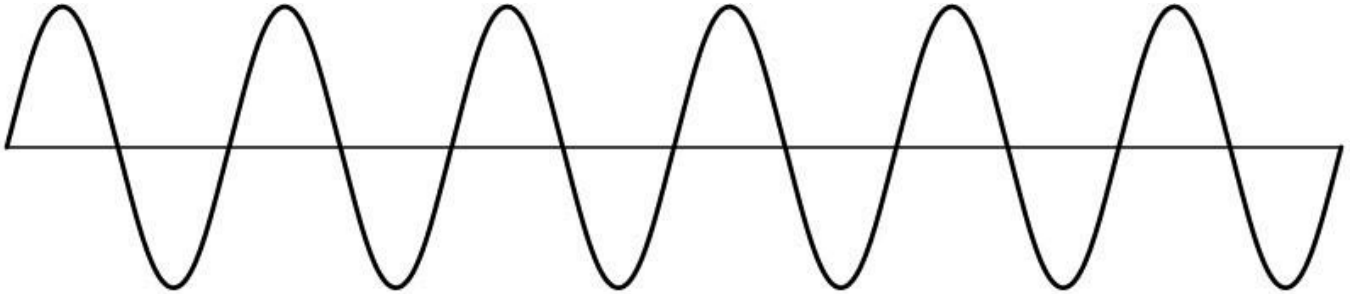
- 抽样是把时间上连续的模拟信号变成一系列时间上离散的抽样值的过程。能否由此样值序列重建原信号，是抽样定理要回答的问题。

- 抽样定理是，如果对一个频带有限的时间连续的模拟信号抽样，当抽样速率达到一定数值时，那么根据它的抽样值就能重建原信号。
  - 即：若要传输模拟信号，不一定要传输模拟信号本身，只需传输按抽样定理得到的抽样值即可。因此，抽样定理是模拟信号数字化的理论依据。

- 根据用来抽样的脉冲序列是等间隔的还是非等间隔的，
  - 均匀抽样定理
  - 非均匀抽样；

抽样定理

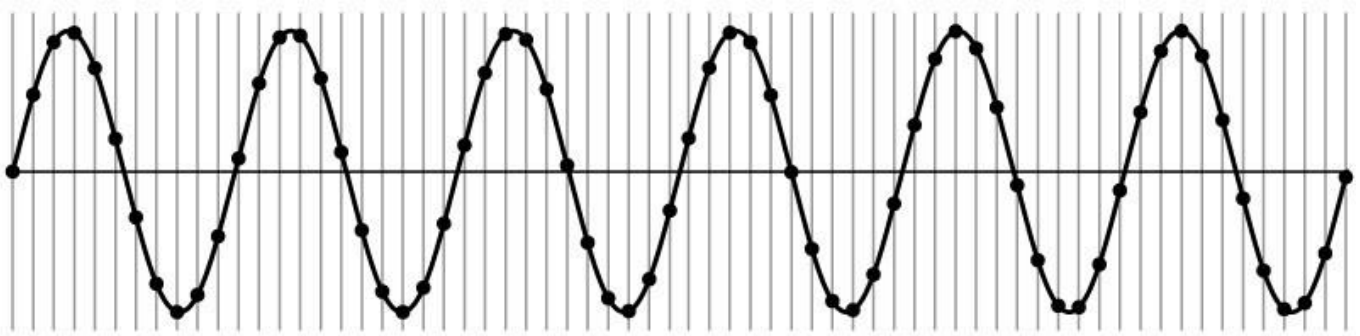
简单例子：正弦信号



如何将它离散化?

抽样定理

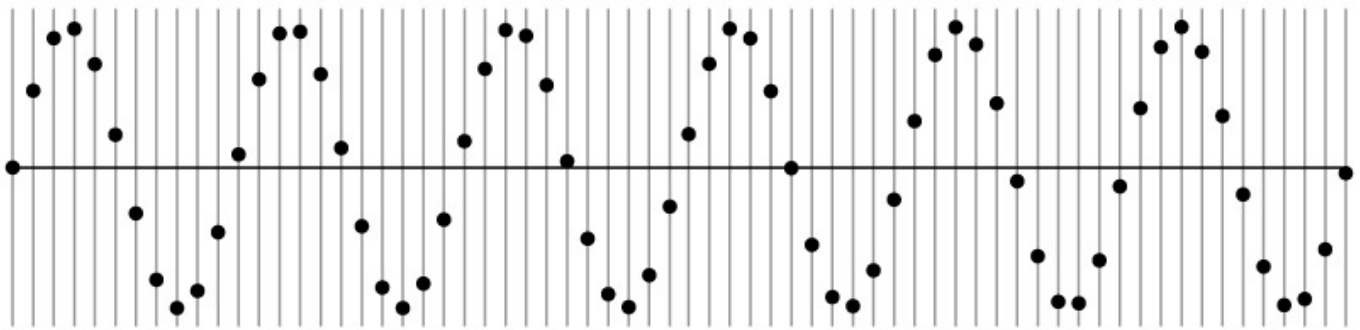
简单例子：正弦信号



等间隔采样

抽样定理

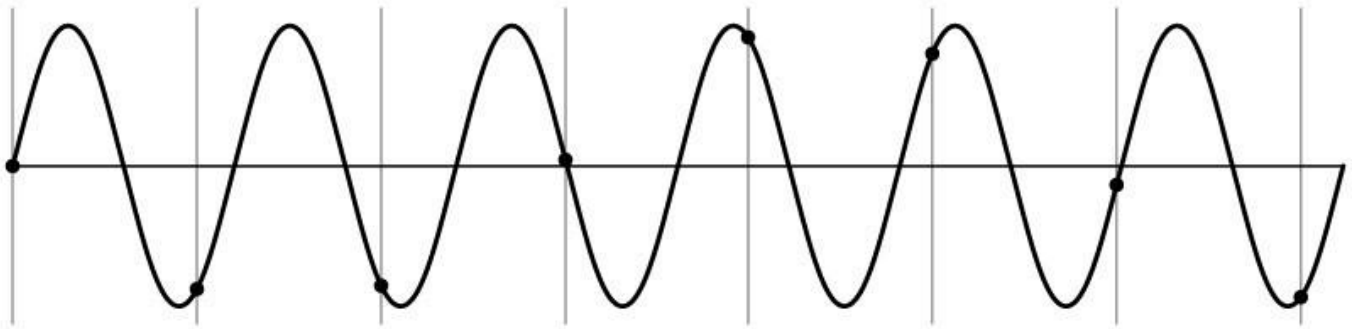
简单例子：正弦信号



采样多少点?

欠采样

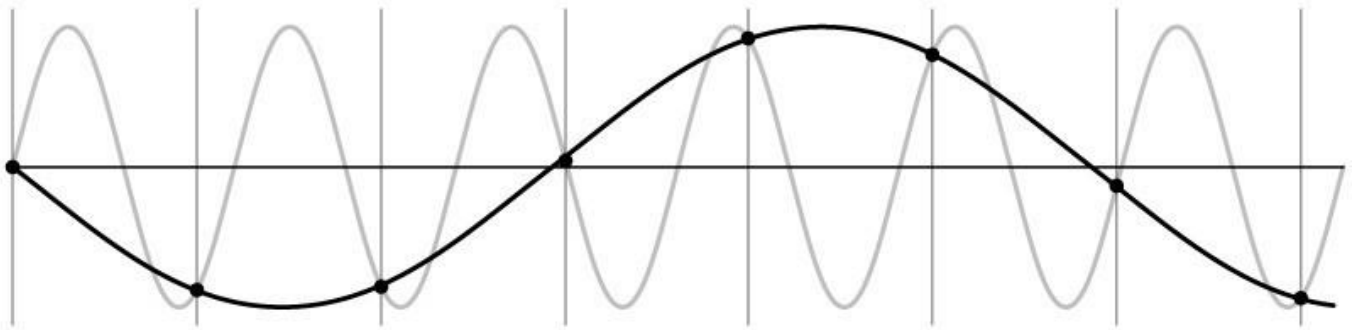
简单例子：正弦信号



信息丢失严重！

欠采样

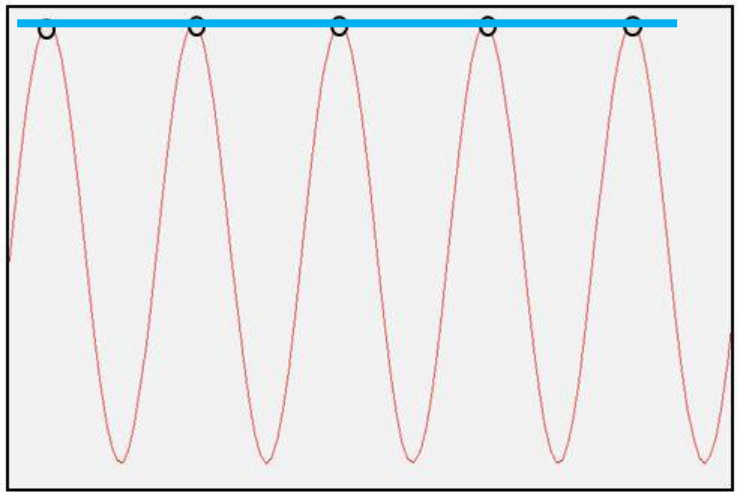
简单例子：正弦信号



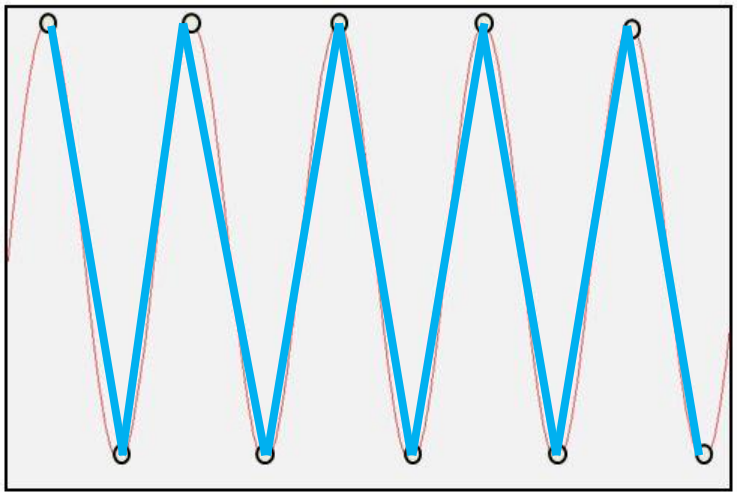
混叠：原本的高频信号被采样成低频信号

欠采样

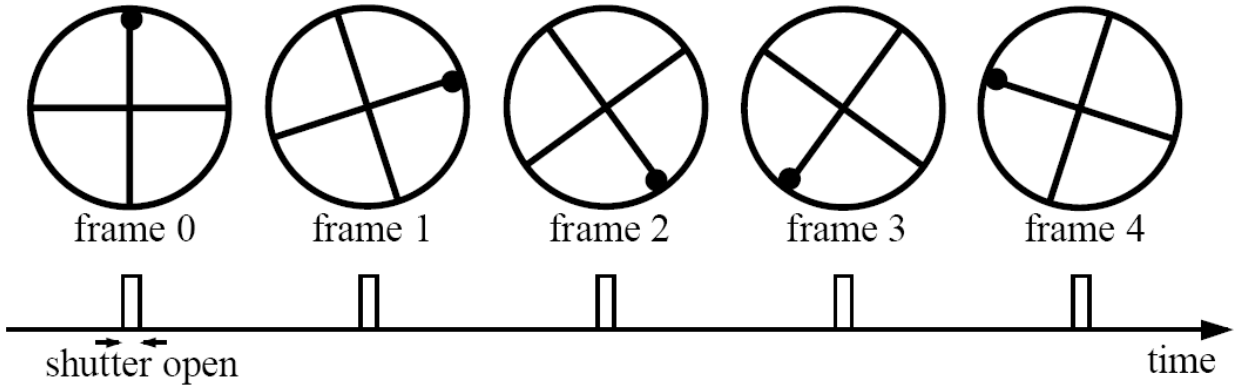
采样频率= 信号频率

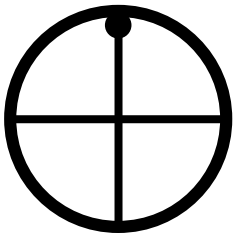


采样频率= 2*信号频率

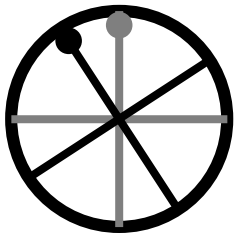


视频中的混叠





静止



逆时针旋转

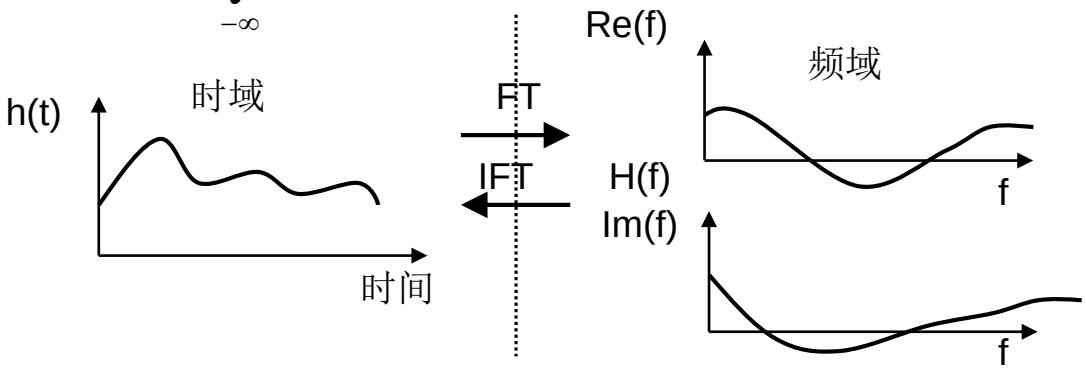
信号分析

- 信号
  - 时域信号
    - 时间变量的曲线
    - 示波器上的电信号
      - 随时间变化的电压值
      - 只给出直流分量和均方根值
  - 频域信号
    - 频谱分析仪

- 每一个时域信号都可以看作是无限多个正弦信号之和，其中每个正弦信号都有各自的幅度、相位和频率。

傅立叶变换

- 傅立叶变换可以得到时域信号的频谱：
  - 正变换 $H(f)=\int_{-\infty}^{+\infty}h(t)e^{-j2\pi ft}dt$
  - 逆变换 $h(t)=\int_{-\infty}^{+\infty}H(f)e^{j2\pi ft}df$

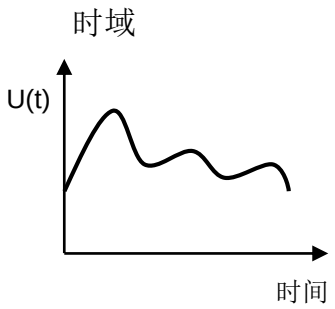


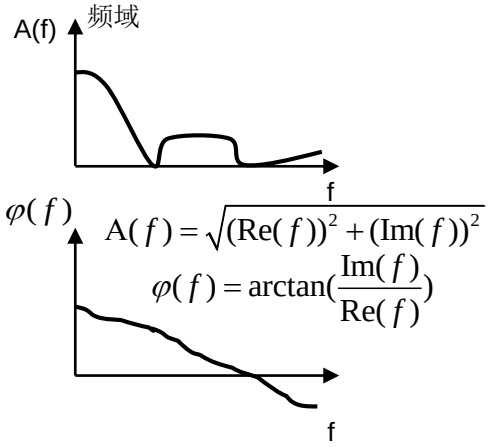
傅立叶变换

- 傅立叶变换的积分从负无穷到正无穷，要求已知信号的全部时域值，而且是确知信号。
- 傅立叶变换的结果是复数，实部是余弦分量的幅度，虚部是正弦分量的幅度，可以得到频谱上任一点的实部和虚部。

傅立叶变换

- 幅度和相位特性：
  - 幅频特性 $A(f)=\sqrt{(\mathrm{Re}(f)^2+\mathrm{Im}(f)^2)}$
  - 相频特性 $\varphi(f)=\arctan(\frac{\mathrm{Im}(f)}{\mathrm{Re}(f)})$
- 对实部和虚部应用毕达哥拉斯原理，可以计算出幅度和相位
  - 实部对称性 $\mathrm{Re}(-f)=\mathrm{Re}(f)$
  - 虚部反对称性 $\mathrm{Im}(-f)=-\mathrm{Im}(f)$





幅度和相位特性

傅立叶变换

- 傅立叶变换存在的条件：

$$\int_{\infty}^{+\infty} | x(t) | dt < \infty$$

（角频率-- the radian frequency ）

$$\Omega = 2\pi f \qquad T = \frac{1}{f}$$

离散信号与系统

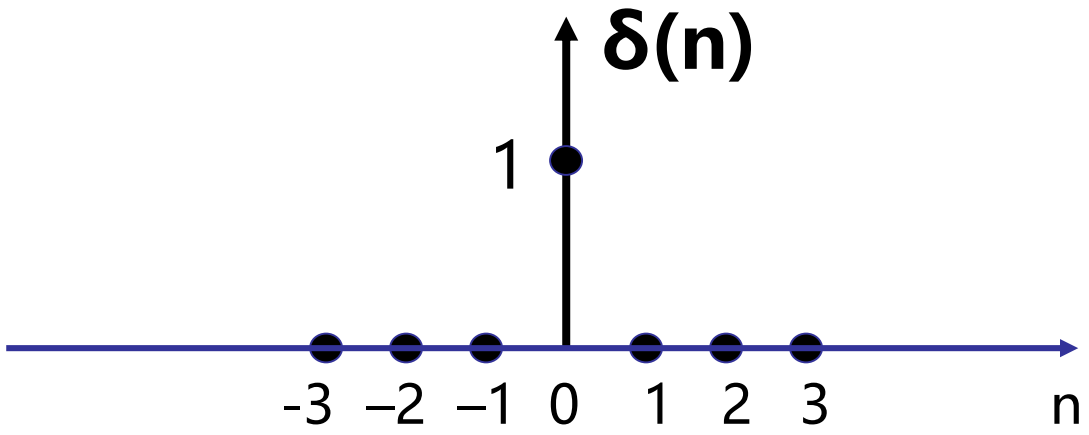
- 信号（Signals） $x(n) = x_a(t)\big|_{t=nT}$

- 信号类型（Signal Type）

δ(n) ----the unit impulse sequence

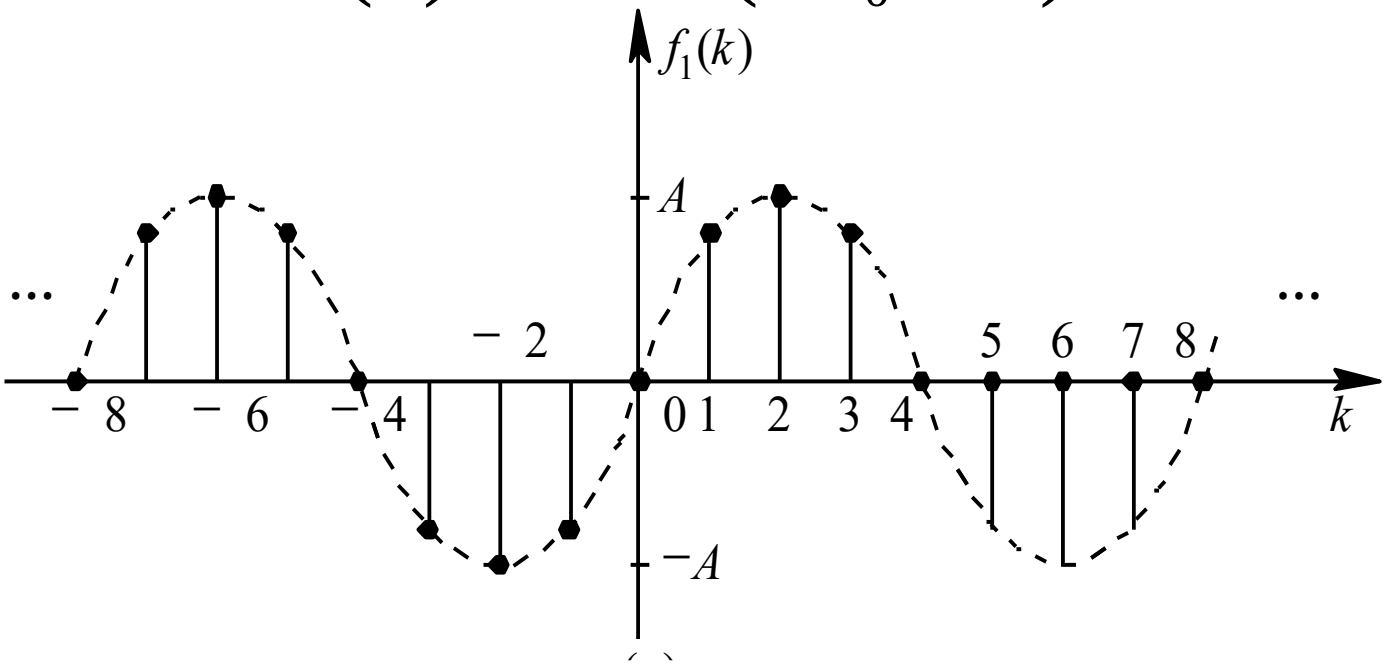
（单位冲激序列或单位脉冲序列）

$$\delta(n)=\begin{cases}1, n=0\\ 0, n\neq 0\end{cases}$$



- sin(ωn) --- the sinusoidal sequence （正弦序列）

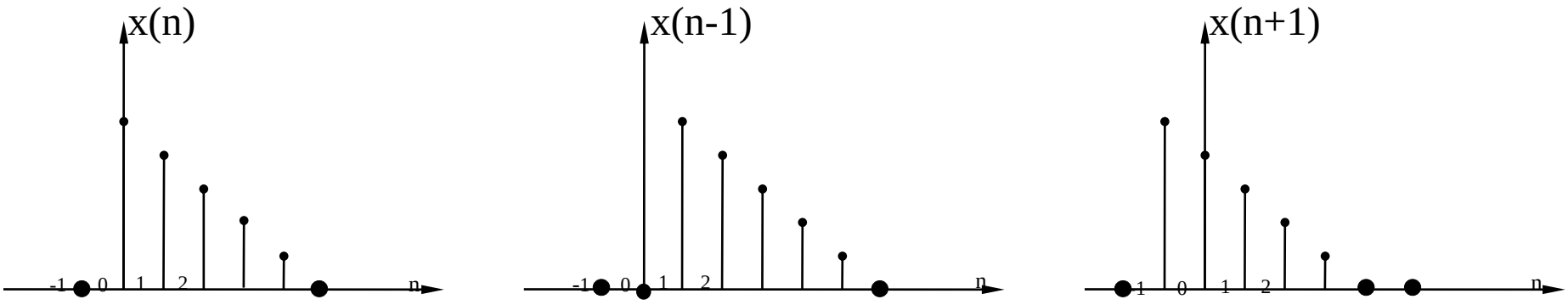
$$x(n) = A\sin(n\omega_0 + \phi)$$



x(n)=x(n+kN)---- the periodic sequence
（周期序列）

operation of sequence (序列的运算)

- shift(移位)

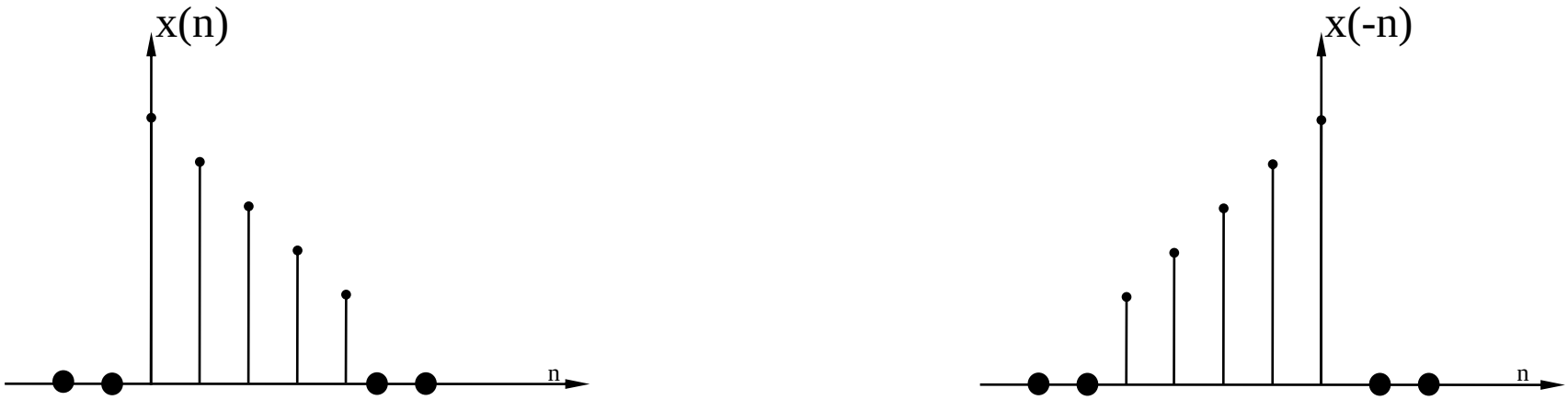


序列的移位

operation of sequence (序列的运算)

- turnover(翻褶)

  序列的翻褶又称为转置或反折，若序列为x(n)，则x(-n)就是以n=0为对称轴将序列x(n)加以翻褶



序列的翻褶

operation of sequence (序列的运算)

- sum of sequence(序列的和)

  序列x(n)与序列y(n)之和是指两个序列同序号的数值逐项对应相加而构成一个新的序列z(n)。

  example：已知x(n)和y(n)，求x(n)+ y(n)。

$$x(n)=\begin{cases}3^n & n\geq 0\\ 0 & n<0\end{cases}\qquad y(n)=\begin{cases}\left(\frac{1}{2}\right)^n & n\geq -1\\ n+1 & n<-1\end{cases}$$

解：

$$z(n)=x(n)+y(n)=\begin{cases}n+1 & n<-1\\ (1/2)^n & -1\leq n<0\\ 3^n+(1/2)^n & n\geq 0\end{cases}$$

operation of sequence (序列的运算)

- product of sequence(序列的积)

  序列x(n)与序列y(n)相乘是指两个序列同序号的数值逐项对应相乘而构成的一个新序列z(n)，

  表示为z(n) =x(n) ·y(n)。

example：已知

$$x(n)=\begin{cases}\dfrac{1}{2}\left(\dfrac{1}{2}\right)^n, & n\geq -1\\ 0, & n<-1\end{cases}\qquad y(n)=\begin{cases}2^n, & n<0\\ n+1, & n\geq 0\end{cases}$$

求z(n) =x(n) ·y(n)

solution：

$$z(n)=x(n)\cdot y(n)=\begin{cases}0, & n<-1\\ \dfrac{1}{2}, & -1\leq n<0\\ \dfrac{1}{2}(n+1)\left(\dfrac{1}{2}\right)^n, & n\geq 0\end{cases}$$

operation of sequence (序列的运算)

- accumulation of sequence(序列的累加)

if a sequence is x(n), then the accumulated sequence of x(n) is y(n) defined as:

$$y(n) = \sum_{k=-\infty}^{n} x(k)$$

它表示y(n)在某个n点的值等于这个n点上的x(n)以及以前的所有n值上的x(n)值之和。

operation of sequence (序列的运算)

- Differential operation of sequence (序列的差分运算)

  forward difference $\Delta x(n) = x(n+1) - x(n)$

  backward difference $\nabla x(n) = \Delta x(n-1)$

由此得出

$$\nabla x(n) = x(n) - x(n-1)$$

operation of sequence (序列的运算)

- 序列的时间尺度（比例）变换：对序列x(n)，其比例变换序列为x(mn)或x(n/m)，其中m为正整数。
  - x(4n)是从x(n)中每隔4个值取1个值，相当于将x(n)的抽样间隔从T增加到4T。这种运算称为抽取，将x(4n)称为x(n)的抽取序列。
  - x(n/4)表示将x(n)的抽样间隔从T减少到T/4，将x(n/4)称为x(n)的插值序列。

operation of sequence (序列的运算)

- 序列的卷积和
  - 卷积积分是求连续线性时不变系统输出响应的主要方法。
  - 卷积和是求离散线性时不变系统输出响应的主要方法。
  - 设两个序列为x(n)和h(n)，x(n)和h(n)的卷积和定义为

$$y(n)=\sum_{m=-\infty}^{\infty}x(m)h(n-m)=x(n)*h(n)=h(n)*x(n)$$

序列卷积和的求解

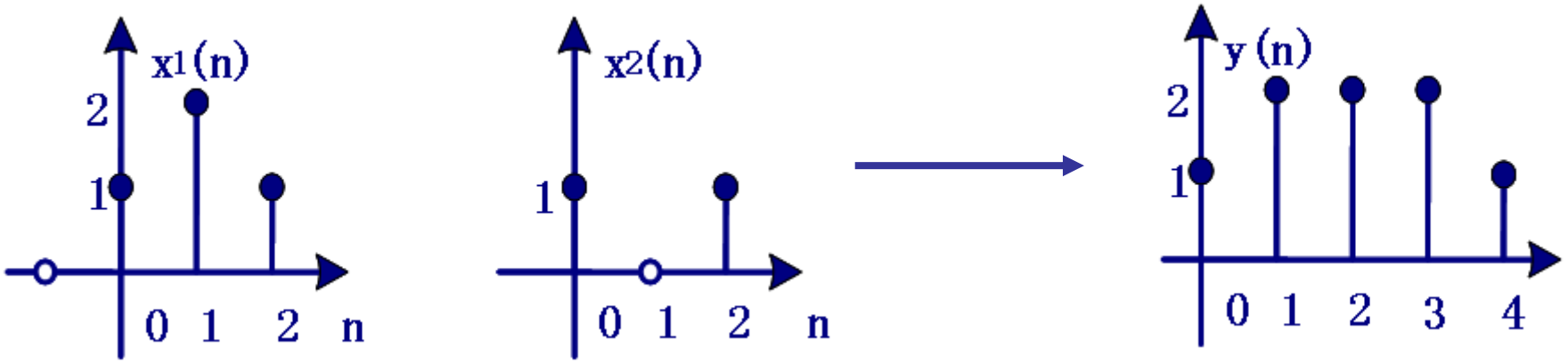
- 卷积和的图解法计算步骤如下：
  - 翻褶：先将x(n)和h(n)的变量置换为m，得到x(m)和h(m)，将h(m)以m=0的垂直轴为对称轴翻摺为h(-m)；
  - 移位：将h(-m)沿m轴平移n得到h(n-m)，当n>0时，右移n位，当n<0时，左移|n|位；
  - 相乘：对给定的某个n值，将h(n-m)和 x(m)相同m值的对应点相乘；
  - 相加：再将以上所有对应点的乘积累加，就可以得到给定的某n值时的y(n)。

序列卷积

例：求 $x_1(n)$和$x_2(n)$ 的线性卷积。

解：

$$y(n) = x_1(n) * x_2(n) = \sum_{m=-\infty}^{+\infty} x_1(m)x_2(n-m)$$



序列卷积

- 用单位抽样序列来表示任意序列

因为 $$\delta(n-m)=\begin{cases}1 & n=m\\ 0 & n\neq m\end{cases}$$

所以 $$x(n)\delta(n-m)=\begin{cases}x(m) & n=m\\ 0 & n\neq m\end{cases}$$

$$x(n)*\delta(n)=\ x(n)$$

- 由此可以得到序列的另一种表达形式，即任何序列都可以表示为单位抽样序列的加权移位和，即

$$x(n)=\sum_{m=-\infty}^{\infty}x(m)\delta(n-m)$$

序列的能量

- 序列x(n)的能量 E 定义为序列各抽样值的平方和，即：

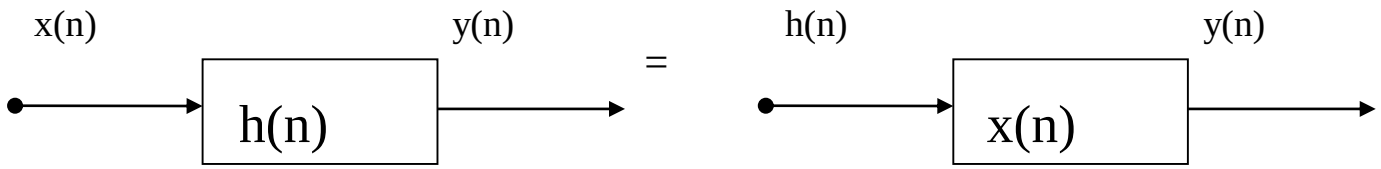
$$E = \sum_{n=-\infty}^{\infty} |x(n)|^2$$

卷积的性质

- 交换律
  - 卷积和与两卷积序列的次序无关，

$$y(n)=x(n)*h(n)=h(n)*x(n)$$

  - 将单位抽样响应$h(n)$改为输入，而将输入$x(n)$改作为系统单位抽样响应，则输出$y(n)$不变.

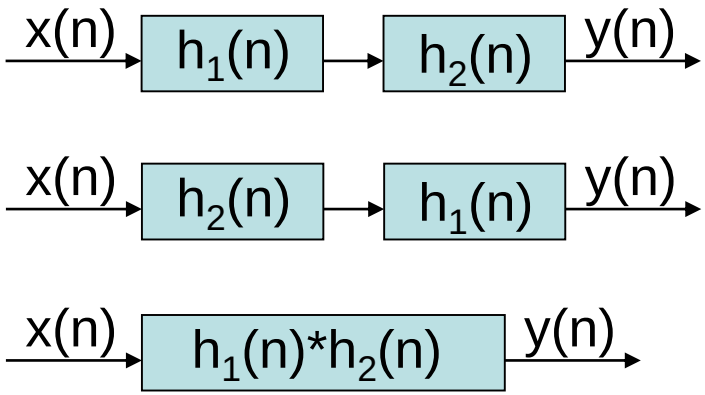


卷积的性质

- 结合律

$$x(n)*h1(n)*h2(n)=[x(n)*h1(n)]*h2(n)$$
$$=x(n)*[h1(n)*h2(n)]=\ [x(n)*h2(n)]\ *h1(n)$$

两个线性时不变系统级联后仍构成一个线性移不变系统，其单位抽样响应为两系统单位抽样响应的卷积和，且线性移不变系统的单位抽样响应与它们的级联次序无关。



卷积的性质

- 分配律

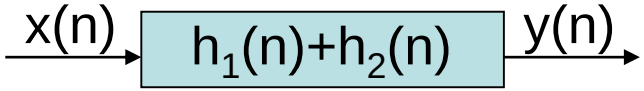
  x(n)* [h1(n) + h2(n)] =x(n)*h1(n) + x(n)* h2(n)

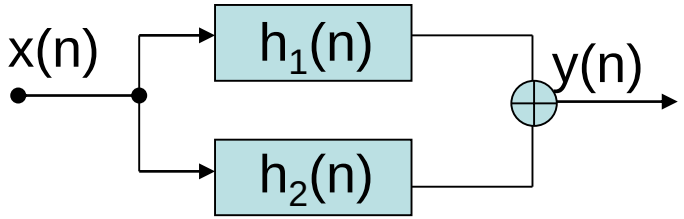
证明：

$$x(n)*[h_1(n)+h_2(n)]=\sum_{m=-\infty}^{\infty}x(m)\cdot[h_1(n-m)+h_2(n-m)]$$

$$=\sum_{m=-\infty}^{\infty}x(m)\cdot h_1(n-m)+\sum_{m=-\infty}^{\infty}x(m)\cdot h_2(n-m)$$

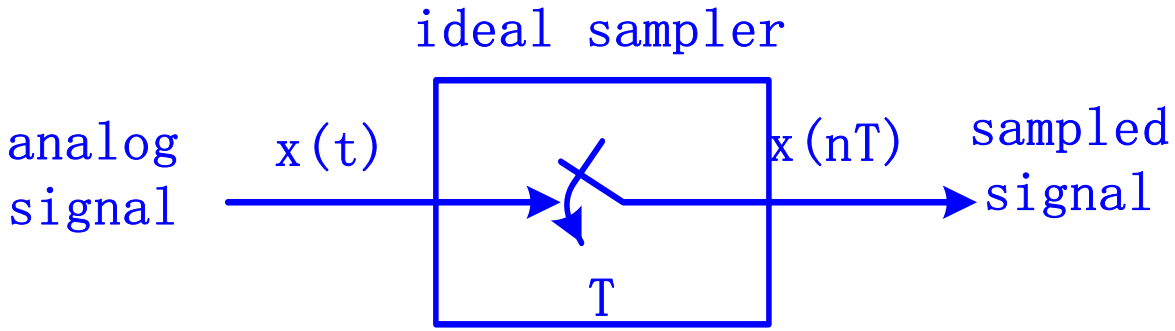
$$=x(n)*h_1(n)+x(n)*h_2(n)$$

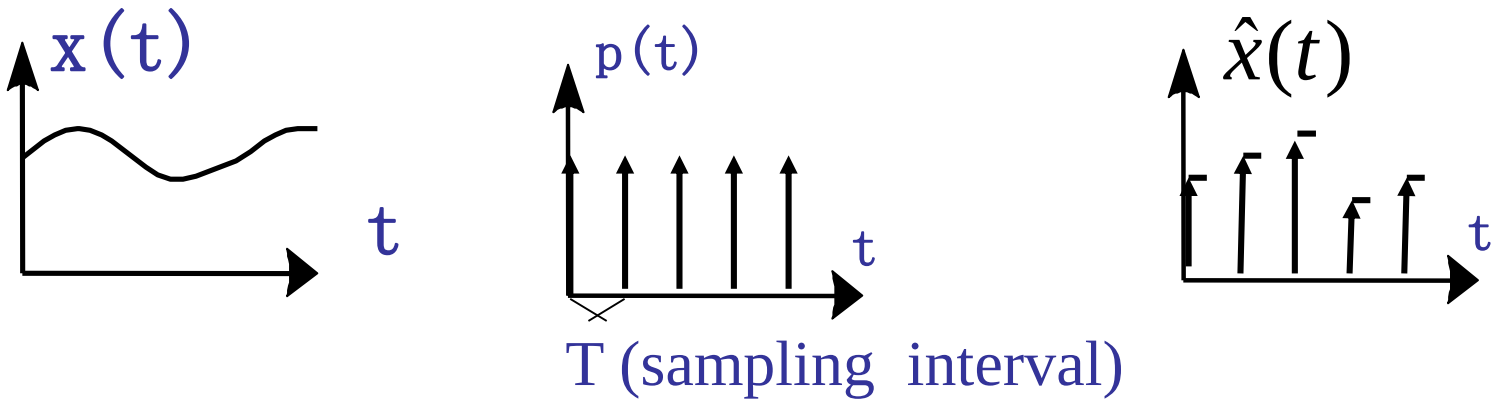




抽样定理

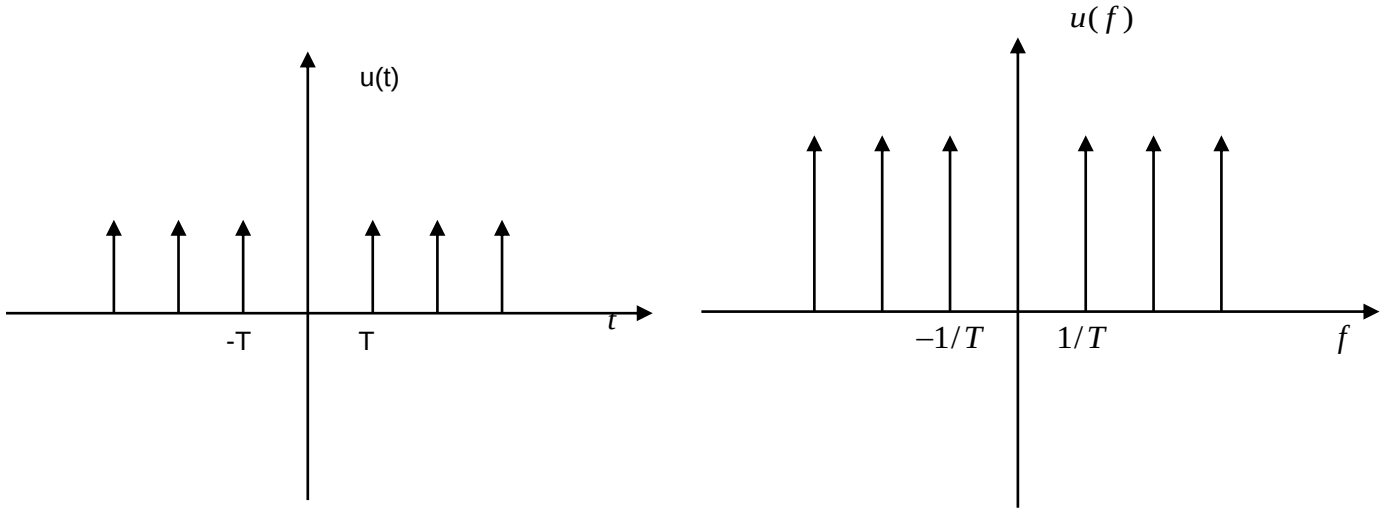
- 理想的抽样器:


ideal sampler
analog
signal
x(t)
x(nT)
sampled
signal
T


x(t)
t
p(t)
t
$\hat{x}(t)$
t
T (sampling interval)

时域信号到频域信号变换

- 周期为T的一序列Dirac脉冲的频谱是一序列周期为1/T的离散Dirac脉冲谱线



Dirac脉冲序列的傅立叶变换

- Dirac脉冲序列是分析信号采样的有用工具。
- 模拟信号的采样使用Dirac脉冲序列。

抽样定理

理想采样器P(t)

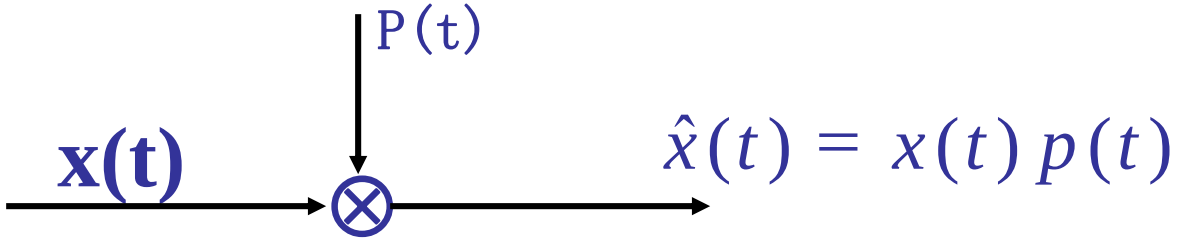
$$P(t) = \sum_{n=-\infty}^{+\infty} \delta(t - nT)$$

$$P(t) = \frac{1}{T} \sum_{k=-\infty}^{+\infty} e^{jk\Omega_s t}$$

δ(t)----impulse function，delta function
T----the sampling interval （采样间隔）

抽样定理

Sampled Signals(采样信号)： $\hat{x}(t)$



$$\hat{x}(t) = x(t)\,p(t)$$

$$= \sum_{n=-\infty}^{+\infty} x(t)\,\delta(t - nT)$$

$$= \sum_{n=-\infty}^{+\infty} x(nT)\,\delta(t - nT)$$

周期性频谱

discrete-time signal ⇔ periodic spectrum
(离散时间信号) (周期性频谱)

$$x(t) \Leftrightarrow X(j\Omega) = \int_{-\infty}^{+\infty} x(t) e^{-j\Omega t} dt$$

$$P(t)_{t=kT} = \frac{1}{T} \sum_{k=-\infty}^{+\infty} e^{jk\Omega_s t}$$

经过采样的信号可表示为:

$$\hat{x}(t) = x(t)P(t) = \frac{1}{T} \sum_{k=-\infty}^{+\infty} x(t) e^{jk\Omega_s t}$$

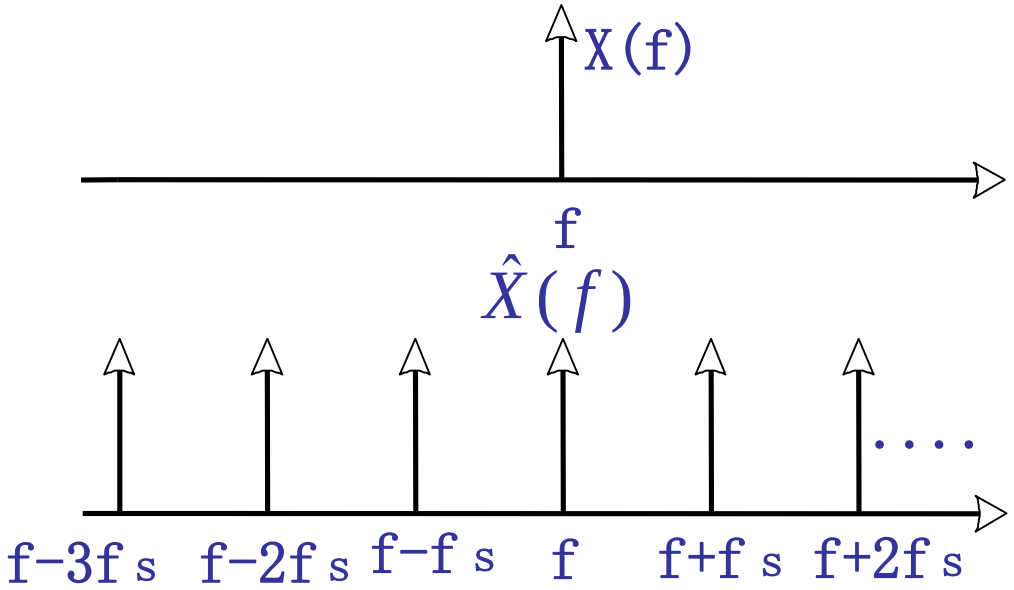
抽样定理

采样信号的频谱为：

$$\hat{X}(j\Omega) = \frac{1}{T}\sum_{k=-\infty}^{+\infty} X(j\Omega - jk\Omega s)$$

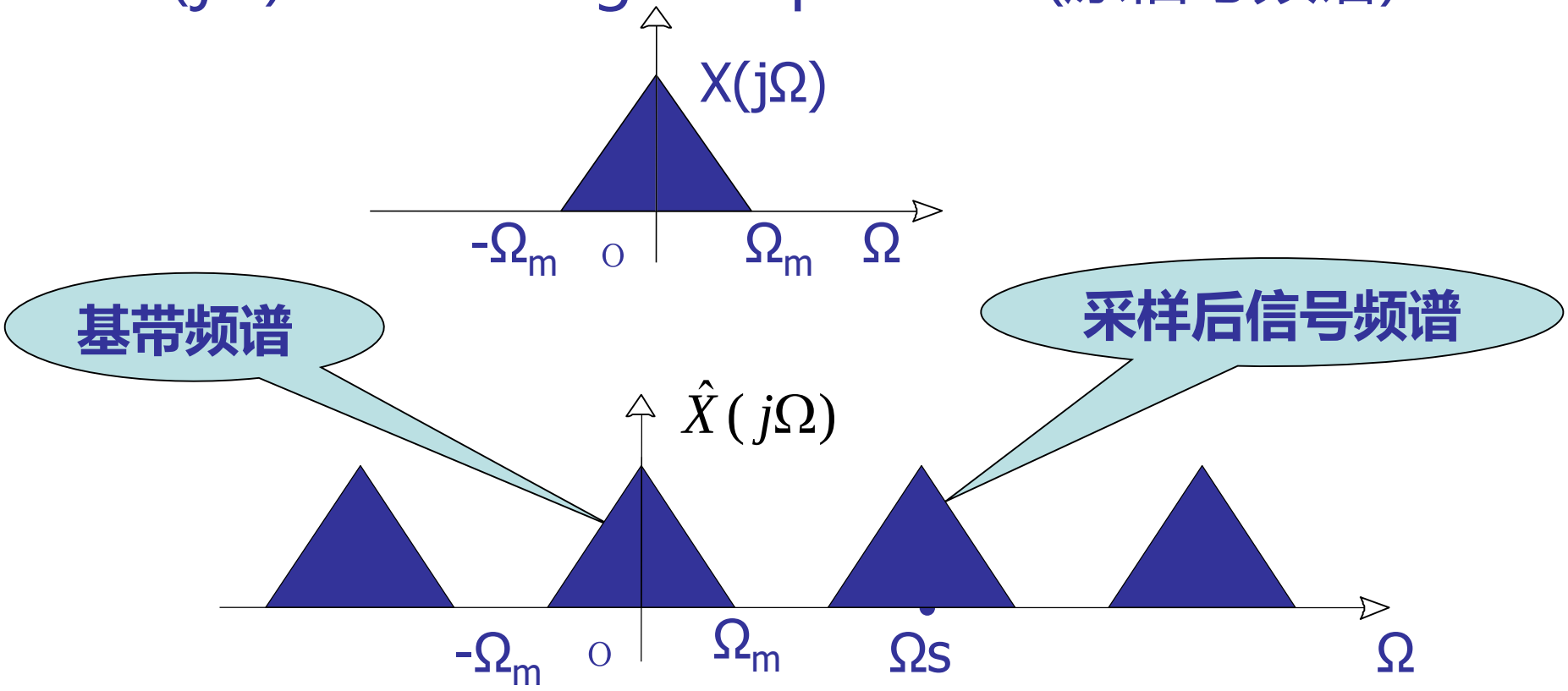
采样后信号的频谱是原模拟信号频谱以采样角频率Ωs 周期延拓的结果

Example1:



Example 2:

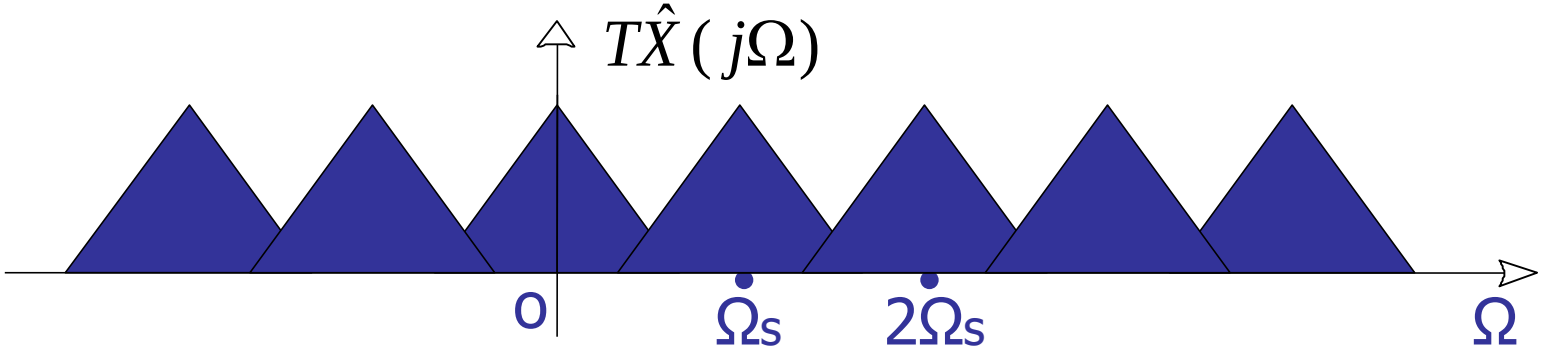
- X(jΩ)----the original spectrum(原信号频谱)



The periodic extension （周期性延拓） of the X(jΩ)

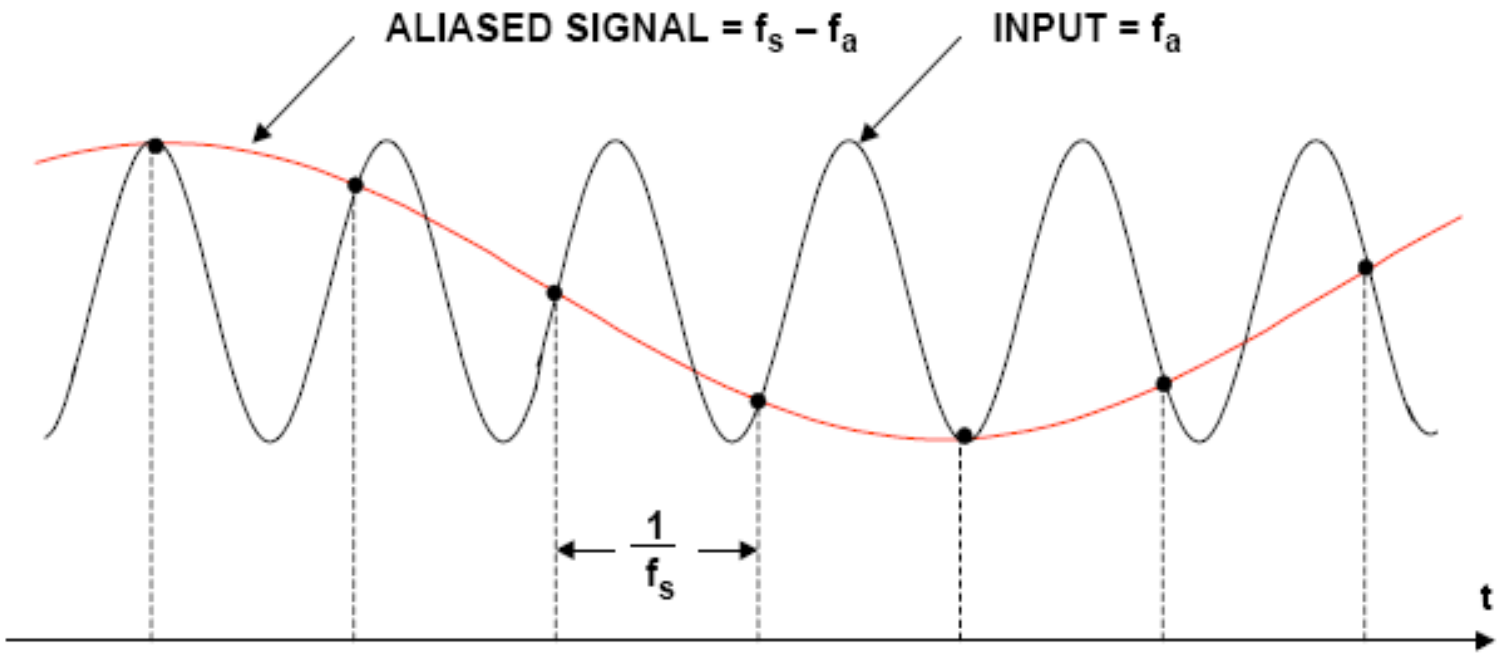
Example

- A folding （折叠）or aliasing （混叠）



信号混叠

ALIASING IN THE TIME DOMAIN



NOTE: f_a IS SLIGHTLY LESS THAN f_s

抽样定理

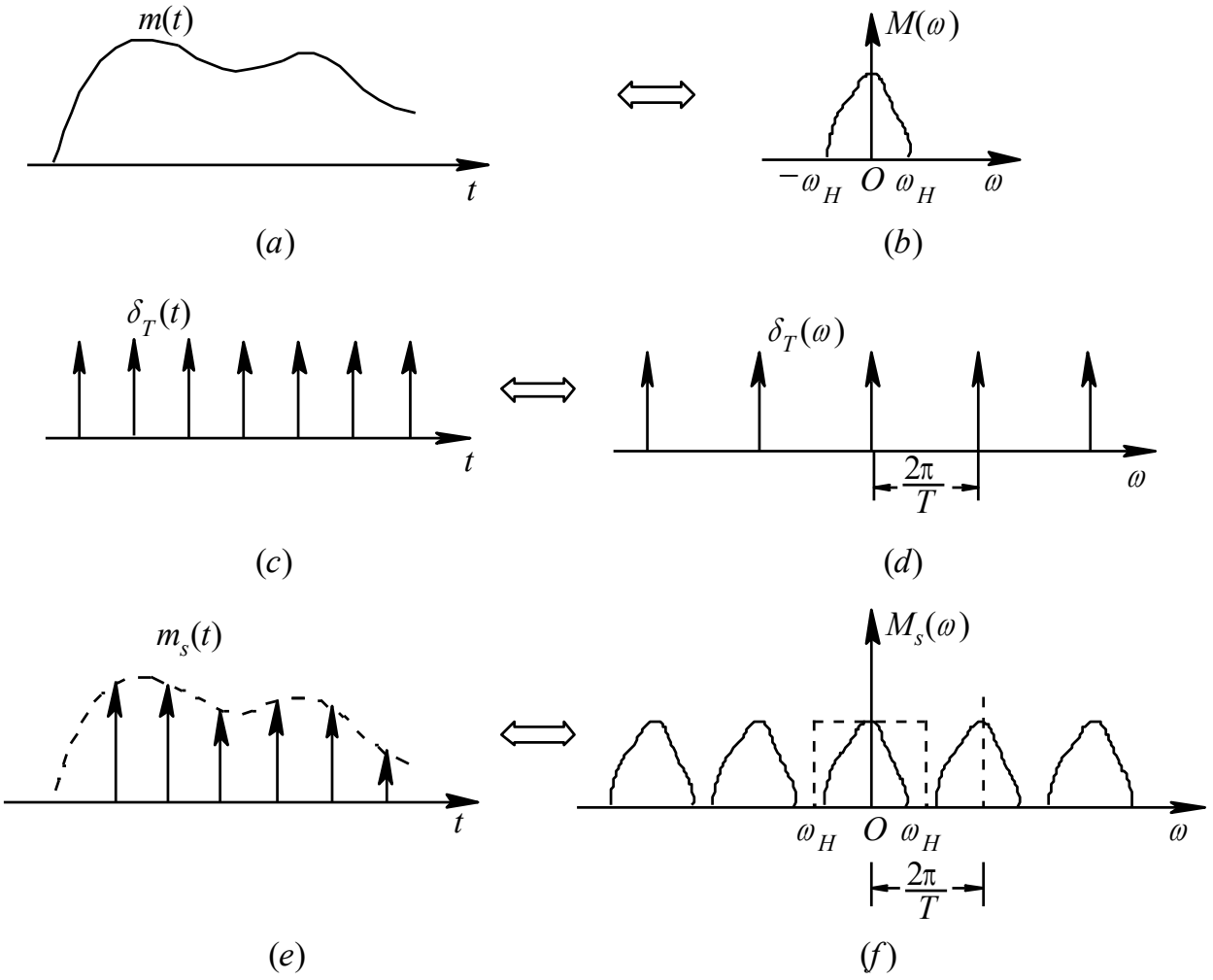
- 两个条件:
  - Bandlimited signal(带限信号)
  - fs≥2fmax　　or　　$T \leq \frac{1}{2f_{\max}}$

fs：采样频率　　T: 采样周期
fmax：信号最高频率

奈奎斯特抽样定理

- 奈奎斯特抽样定理：一个频带限制在(0，f_H）内的时间连续信号m(t)，如果以Ts≤1/(2f_H)秒的间隔对它进行等间隔（均匀）抽样，则m(t)将被所得到的抽样值完全确定。
- 此定理告诉我们：
  - 若m(t)的频谱在某一角频率ω_H以上为零，则m(t)中的全部信息完全包含在其间隔不大于1/(2f_H)秒的均匀抽样序列里。
  - 或者说，抽样速率f_s（每秒内的抽样点数）应不小于2f_H，若抽样速率$f_s < 2f_H$，则会产生失真，这种失真叫混叠失真。
- 下面我们从频域角度来证明这个定理。

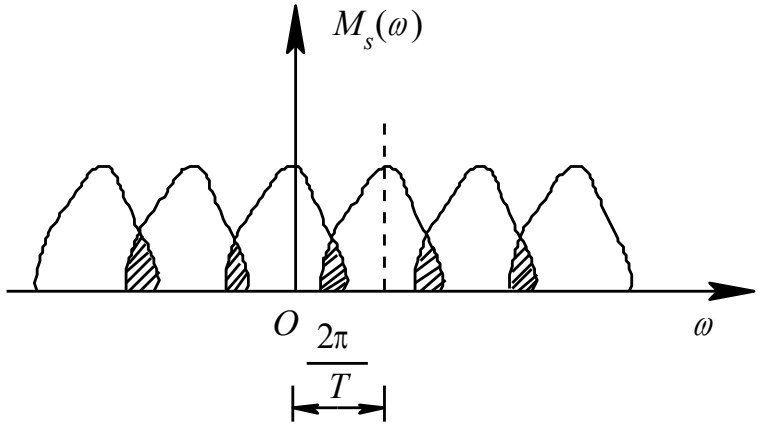
低通抽样定理的证明



抽样过程的时间函数及对应频谱图

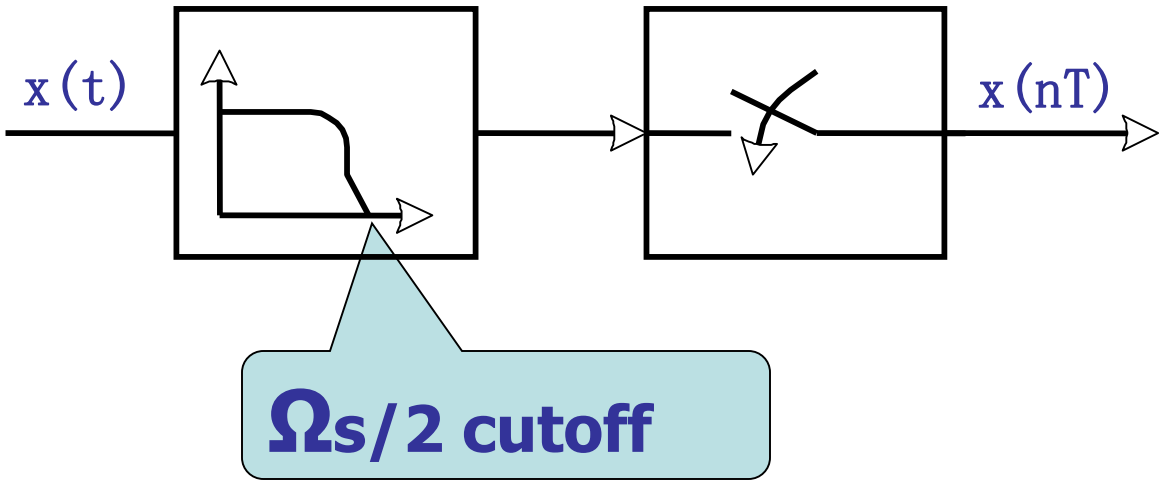
低通抽样定理的证明

- 如果ωs < 2ωH，即抽样间隔Ts > 1/(2fH)，则抽样后信号的频谱在相邻的周期内发生混叠，此时不可能无失真地重建原信号。



- 因此必须要求满足Ts≤1/(2fH)，m(t)才能被ms(t)完全确定，这就证明了抽样定理。

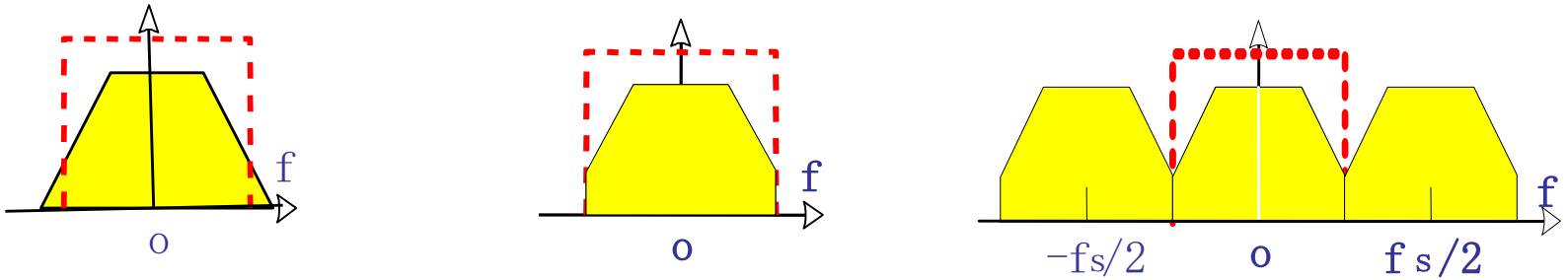
反混叠前置滤波器

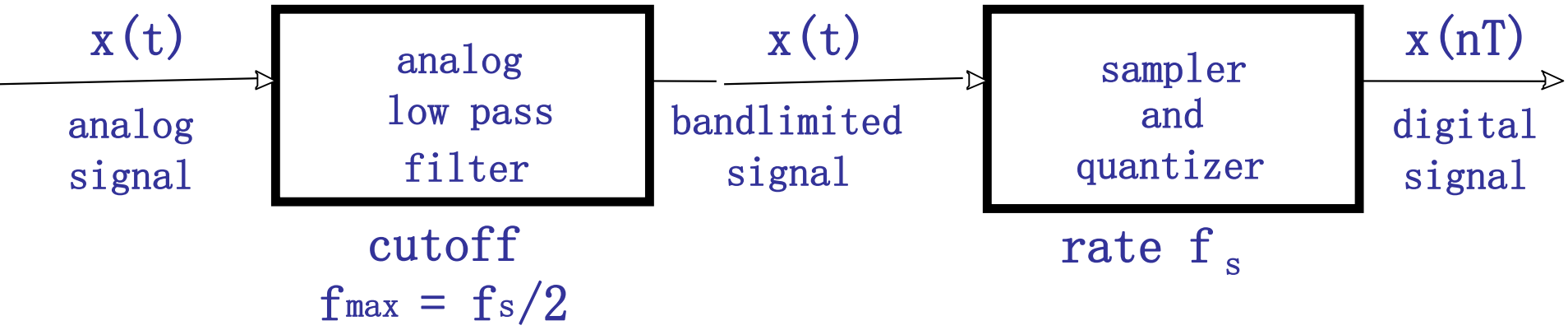


Cutoff frequency----截止频率

反混叠前置滤波器

An example





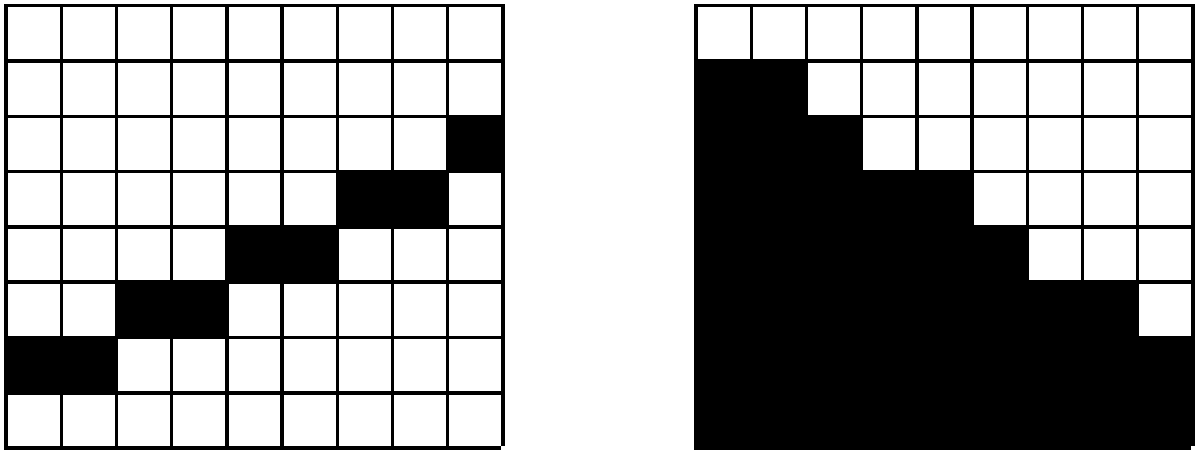
反走样

- 用离散量表示连续量引起的失真现象称之为**走样**(aliasing) 。

- **光栅图形的走样现象**
  - 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。

走样现象举例

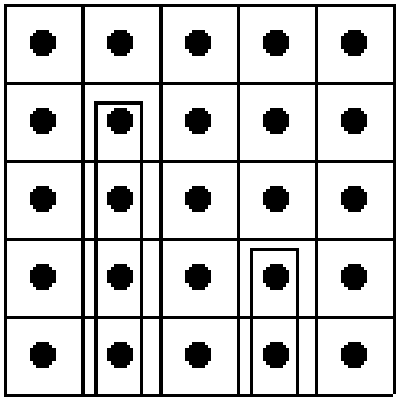
- 不光滑(阶梯状）的图形边界



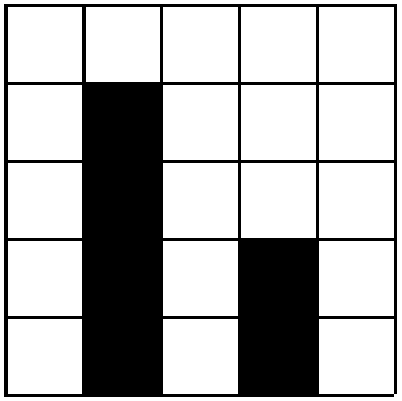
例子：PaintBrush

走样现象举例

- 图形细节失真



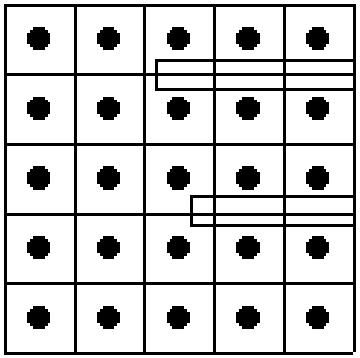
(a)



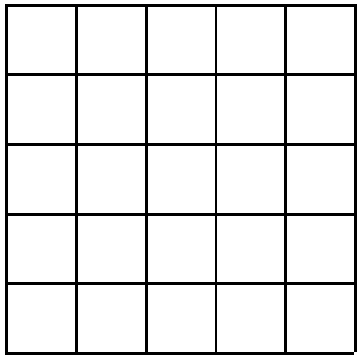
(b)

走样现象举例

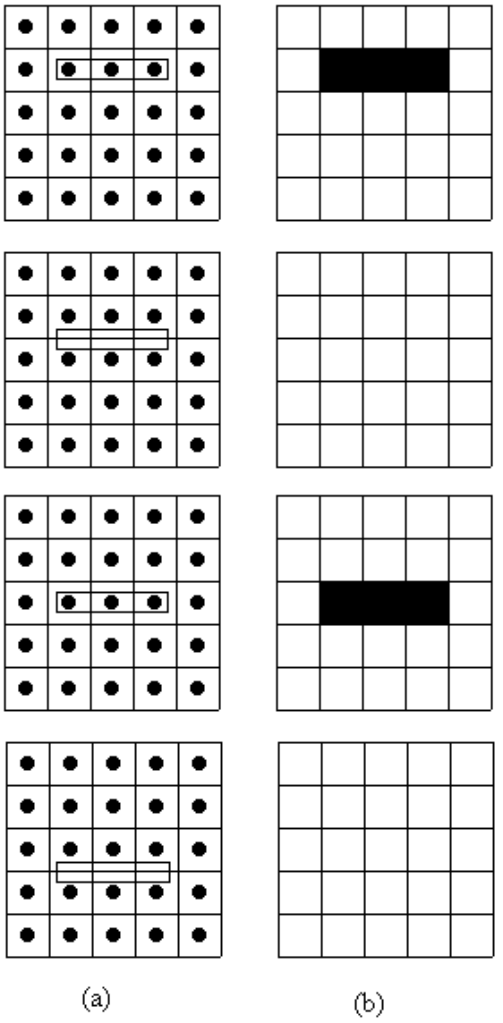
- **狭小图形的遗失与动态图形的闪烁**



(a)



(b)



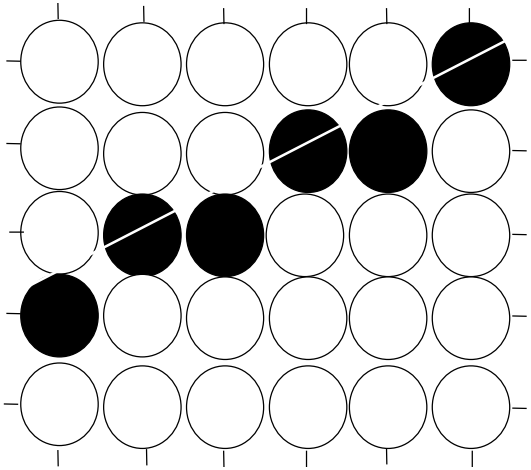
(a) (b)

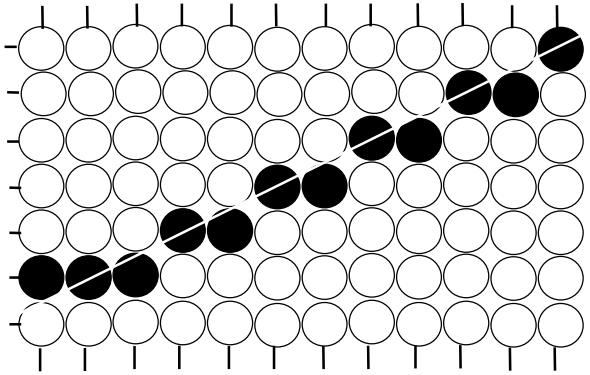
反走样概念及方法

- 用于减少或消除走样现象的技术称为**反走样**(antialiasing)
  - 提高分辨率
  - 简单区域取样
  - 加权区域取样

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。





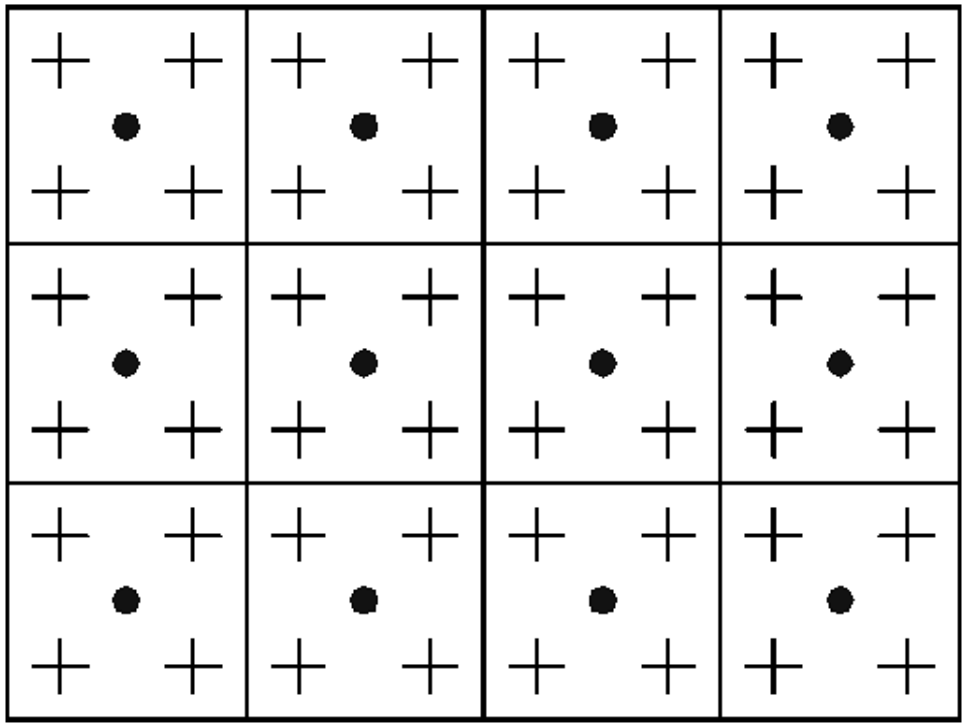
提高分辨率

- 方法简单，但代价非常大。
  - 显示器的水平、竖直分辨率各提高一倍，则显示器的点距减少一倍，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。
- 而且它也只能减轻而不能消除锯齿问题

- 另一种方法（软件方法）
  - 用较高的分辨率的显示模式下计算，（对各自子像素计算，再求（非）加权平均的颜色值），
  - 在较低的分辨率模式下显示。
  - 只能减轻而不能消除锯齿问题。

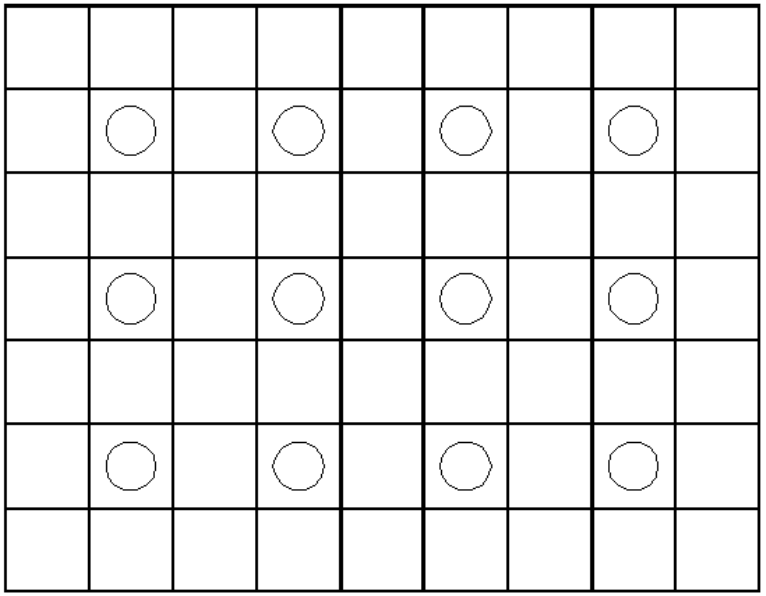
软件方法1

- 把每个像素分为四个子像素，扫描转换算法求得各子像素的灰度值，然后对四像素的灰度值简单平均，作为该像素的灰度值。



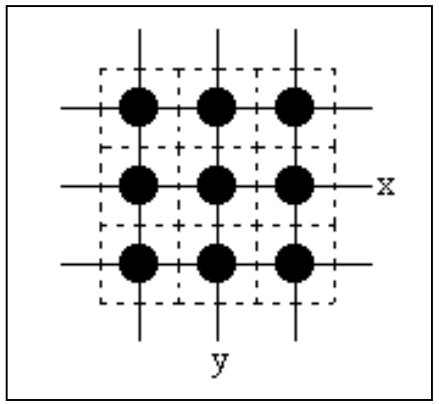
软件方法2

- 设分辨率为$m \times n$,把显示窗口分为$(2m+1)\times(2n+1)$个子像素，对每个子像素进行灰度值计算，
- 然后根据权值表所规定的权值，对位于像素中心及四周的九个子像素加权平均，作为显示像素的颜色。
- 设$m=3, n=4$



简单区域取样

- 方法由来
  - 两点假设
    - ① 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；
    - ② 直线段是数学上抽象直线段，它的宽度为0。
  - 现实
    - 像素的面积不为0；
    - 直线段的宽度至少为1个像素；



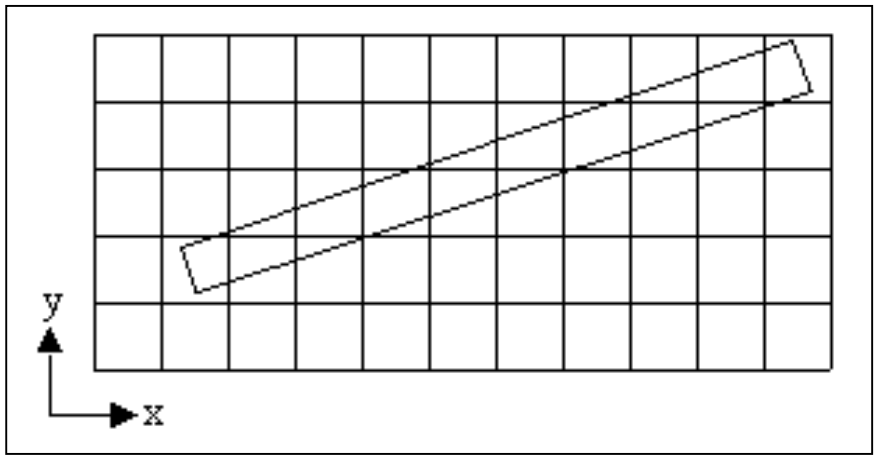
  - 假设与现实的矛盾是导致混淆出现的原因之一

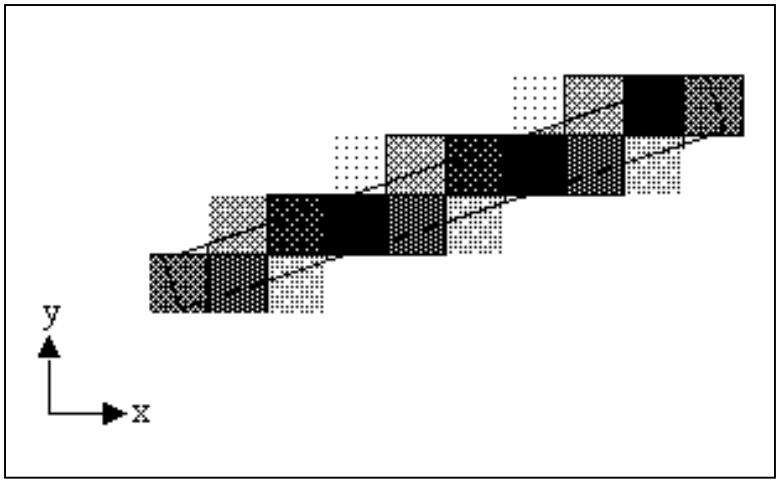
简单区域取样

解决方法：改变直线段模型，由此产生算法

- 方法步骤：

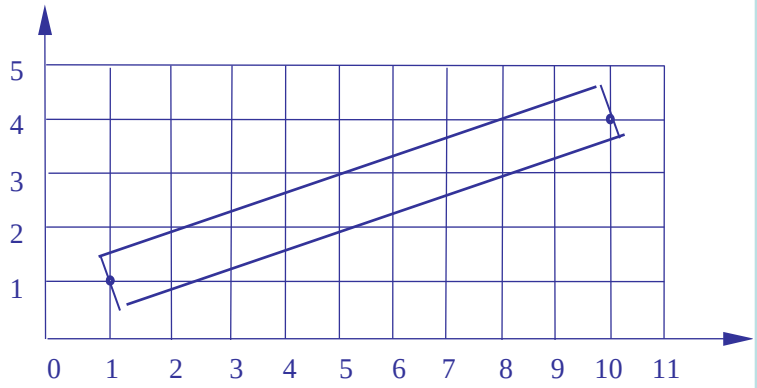
① 将直线段看作具有一定宽度的狭长矩形；
② 当直线段与某象素有交时，求出两者相交区域的面积；
③ 根据相交区域的面积，确定该象素的亮度值



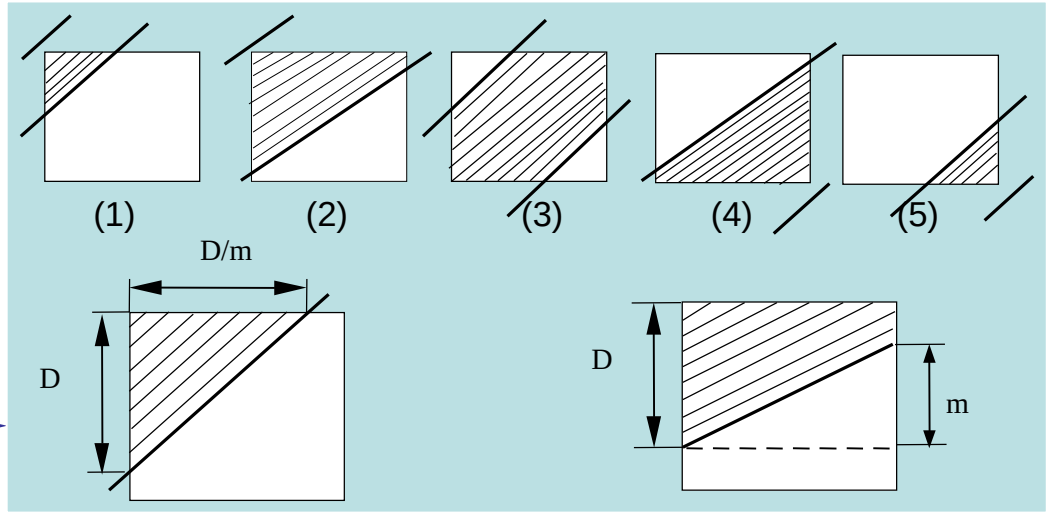


简单区域取样

- 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。

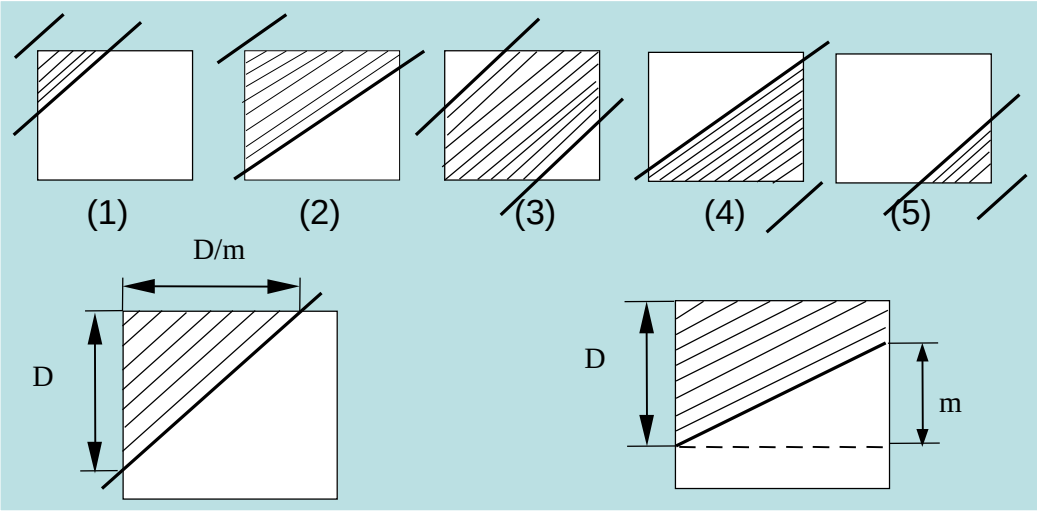


有宽度的线条轮廓



象素相交的五种情况及用于计算面积的量

简单区域取样

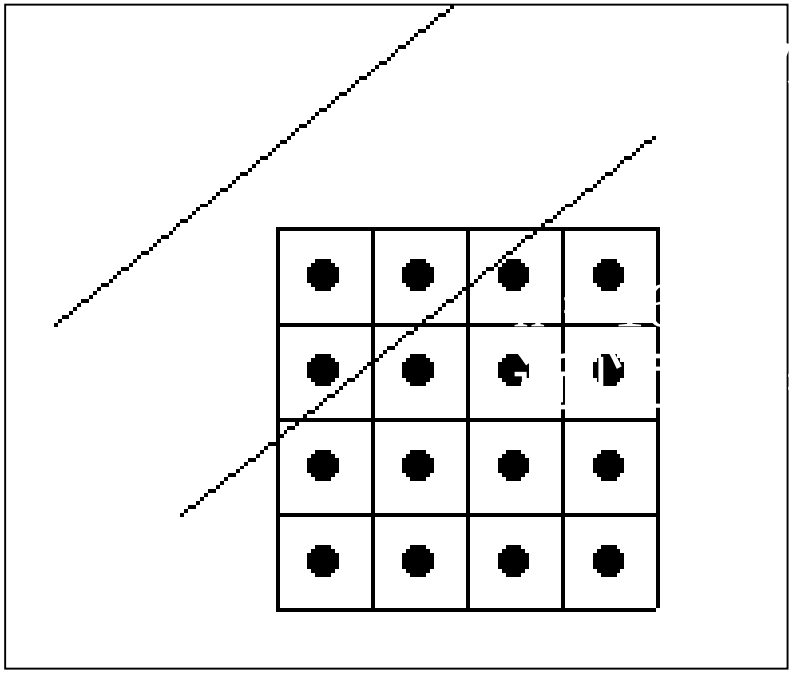


- 面积计算
  - 情况(1)(5)阴影面积为：$D^2/2m$;
  - 情况(2)(4)阴影面积为：D - m/2;　(D+D-m)/2
  - 情况(3) 阴影面积为：1 - D^2/m
- 为了简化计算可以采用离散的方法

简单区域取样

求相交区域的近似面积的离散计算方法

1. 将屏幕象素分割成n个更小的子象素；
2. 计算中心点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



简单区域取样

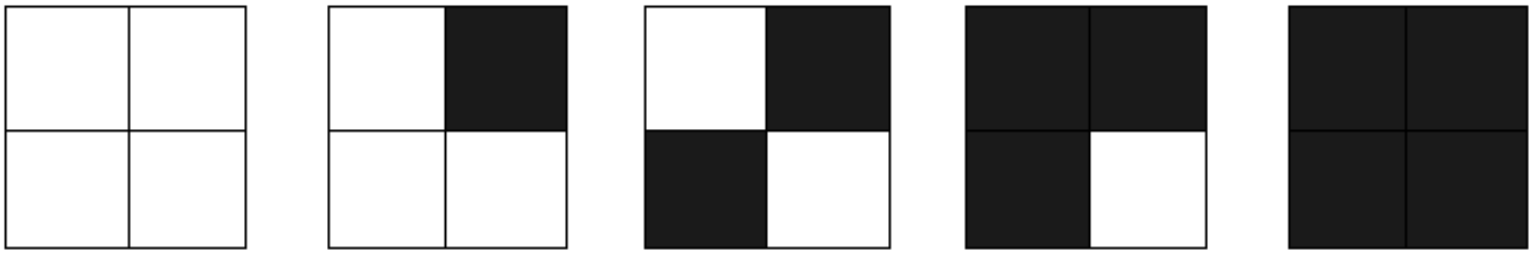
- 简单区域取样采用的是一个盒式滤波器，它是一个二维加权函数，以w表示。
  - w =1　若在当前像素所代表的正方形上
  - w =0　其它区域上
- 直线条经过该像素时，该像素的灰度值可以通过在像素与直线条的相交区域上对w求积分获得。

简单区域取样

- 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。

半色调技术

- 简单区域取样和加权区域取样技术的前提是多级灰度，利用多级灰度来提高视觉分辨率。但是，若只有两级灰度呢？能否使用上述技术呢？
- 对于给定的分辨率，通过将几个像素组合成一个单元来获得多级灰度。
- 例：在一个显示器中将四个像素组成一个单元，可产生5种光强。



半色调技术

- 可用如下矩阵来表示：

$$\begin{pmatrix} 3 & 1 \\ 2 & 4 \end{pmatrix}$$

它表示黑色像素填入2×2个位置中的次序，每一级灰度再添上一个黑色像素就得到下一级灰度。

注意：

1. 要尽量避免连成一条直线的花样。
2. 花样是可以选择的。

- 单元也可以是长方形，如

$$\begin{pmatrix} 4 & 1 & 5 \\ 6 & 3 & 2 \end{pmatrix}$$

半色调技术

- 一般来说，对于两级灰度显示器可能构成的灰度数等于单元中像素个数加1
- 所以单元越大，灰度级别越高
- 它是以牺牲空间分辨率为代价的。

半色调技术

- 例：灰度级别=4,每个单元=2*2

0	0
0	0

1	0
0	0

1	0
0	1

1	1
0	1

1	1
1	1

2	1
1	1

2	1
1	2

2	2
1	2

2	2
2	2

3	2
2	2

3	2
2	3

3	3
2	3

3	3
3	3

- 若有m级灰度，$n\times n$个像素组成一个单元，则灰度级别数为$n\times n\times(m-1)+1$

裁剪算法

二维裁剪

直线段裁剪

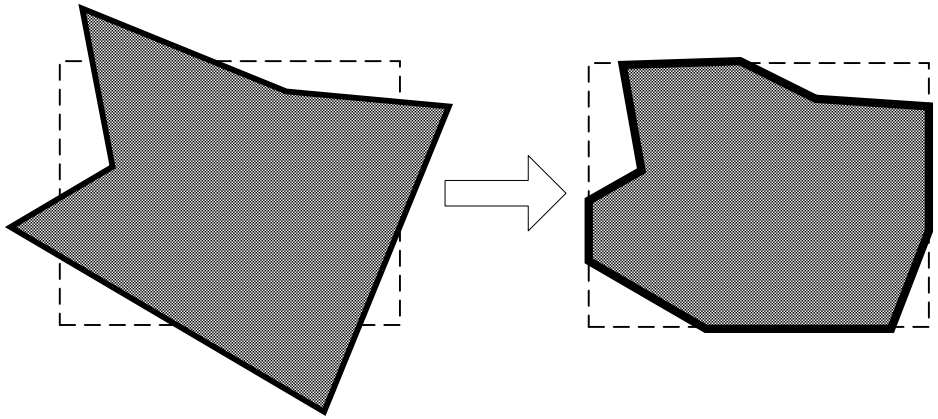
① 直接求交算法
② Cohen-Sutherland算法
③ 中点分割算法
④ 参数化裁剪算法
⑤ Liang-Barskey算法

多边形裁剪

① Sutlerland_Hodgman算法
② Weiler-Athenton算法

裁剪

- 裁剪：确定图形中哪些部分落在显示区之内，哪些落在显示区之外，以便只显示落在显示区内的那部分图形。这个选择过程称为**裁剪**。

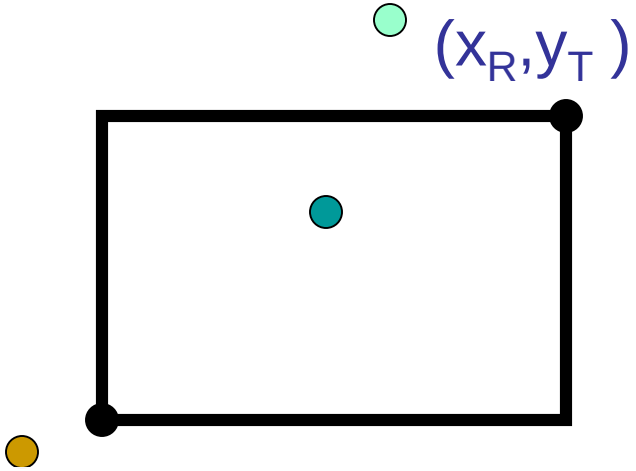


图形裁剪算法，直接影响图形系统的效率。

点的裁剪

- 图形裁剪中最基本的问题。
- 假设窗口的左下角坐标为(x_L,y_B),右上角坐标为(x_R,y_T)，对于给定点P(x,y)，则P点在窗口内的条件是要满足下列不等式：

  $x_L <= x <= x_R$ 且$y_B <= y <= y_T$，否则P点就在窗口外。



- **问题：对于任意多边形窗口，如何判别?**

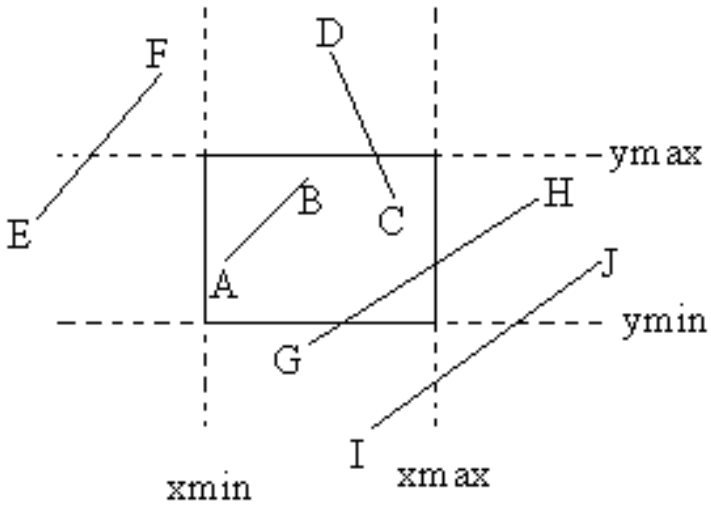
直线段裁剪

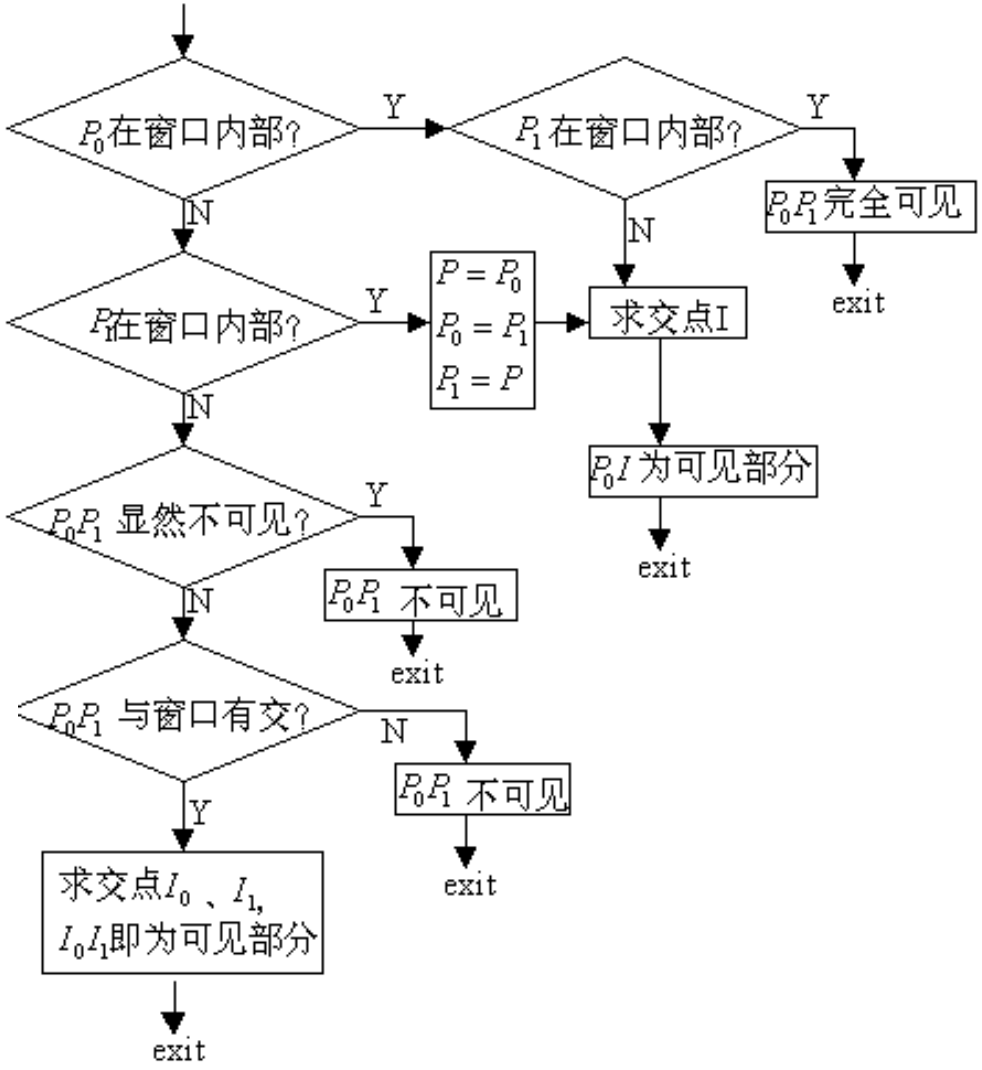
- 直线段裁剪算法是复杂图形裁剪的基础。复杂的曲线可以通过折线段来近似，从而裁剪问题也可以化为直线段的裁剪问题。

  - 直接求交算法
  - Cohen-Sutherland算法
  - 中点算法
  - 梁友栋 - barskey算法
  - 参数化裁剪算法

直接求交算法

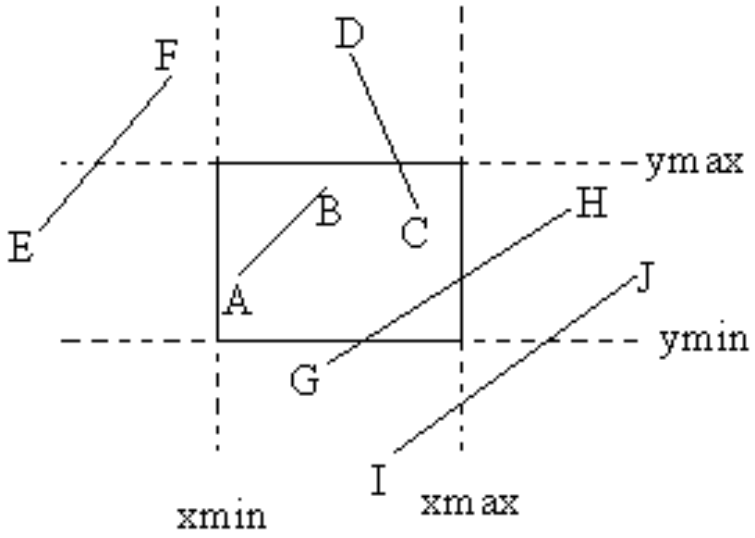
直线与窗口边都写成参数形式，求参数值。





直线段裁剪

- 裁剪线段与窗口的关系：
  ① 线段完全可见；
  ② 显然不可见；
  ③ 其它



- 提高裁剪效率：
  - 快速判断情形(1)(2)，
  - 对于情形(3)，设法减少求交次数和每次求交时所需的计算量。

Cohen-Sutherland裁剪

- 基本思想：

对于每条线段P_1P_2分为三种情况处理：

① 若P_1P_2完全在窗口内，则显示该线段P_1P_2。

② 若P_1P_2明显在窗口外，则丢弃该线段。

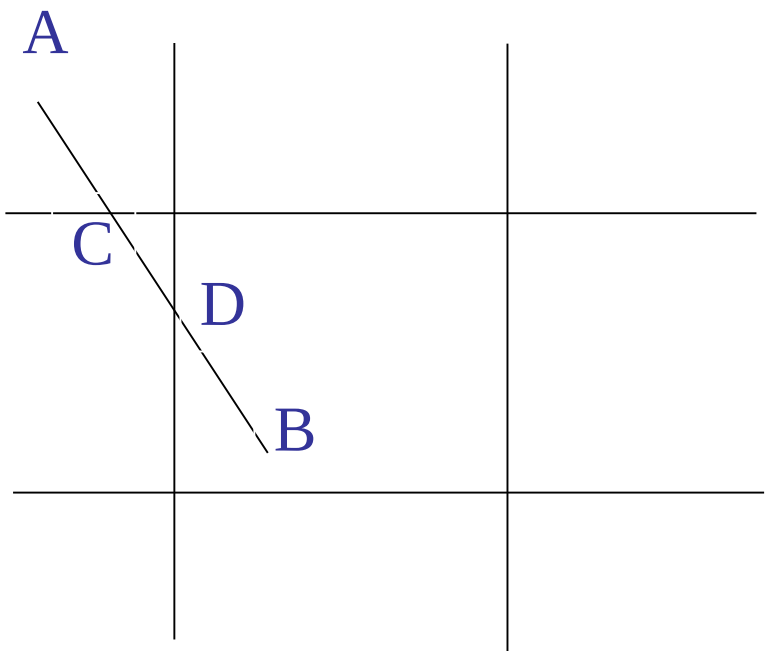
③ 若线段不满足（1）或（2）的条件，则在交点处把线段分为两段。其中一段完全在窗口外，可弃之。然后对另一段重复上述处理。

- 为快速判断，采用如下编码方法：

Cohen-Sutherland裁剪

实现方法：

1001	1000	1010
0001	0000	0010
0101	0100	0110



- 将窗口边线两边沿长，得到九个区域，每一个区域都用一个四位二进制数标识，
- 直线的端点都按其所处区域赋予相应的区域码，用来标识出端点相对于裁剪矩形边界的位置。

Cohen-Sutherland裁剪

- 将区域码的各位从右到左编号，则坐标区域与各位的关系为：

上	下	右	左
X	X	X	X

1001	1000	1010
0001	0000	0010
0101	0100	0110

  - 任何位赋值为1，代表端点落在相应的位置上，否则该位为0。
  - 若端点在剪取矩形内，区域码为0000。如果端点落在矩形的左下角，则区域码为0101。

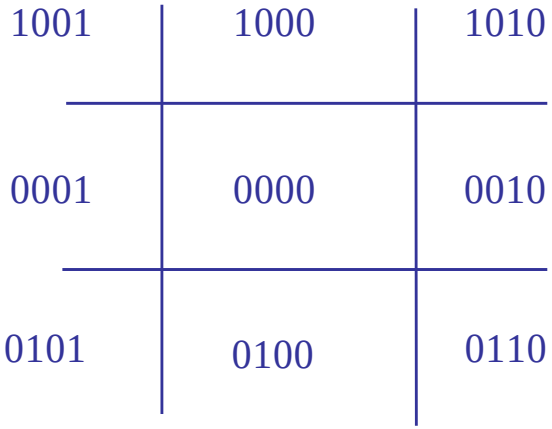
Cohen-Sutherland算法

- 一旦给定所有的线段端点的区域码，就可以快速判断哪条直线完全在剪取窗口内，哪条直线完全在窗口外。所以得到一个规律：

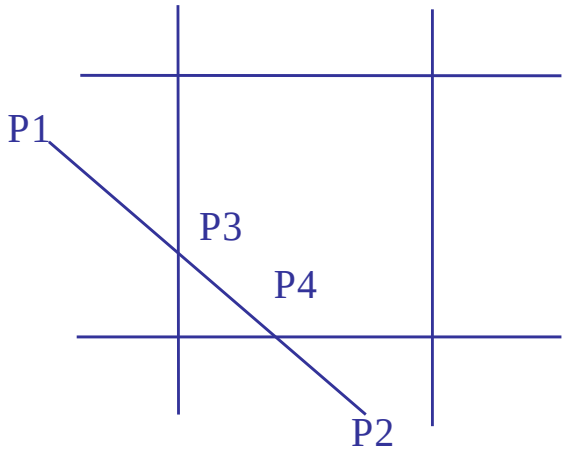
1001	1000	1010
0001	0000	0010
0101	0100	0110

Cohen-Sutherland裁剪

- 若P_1P_2完全在窗口内：code1=0，且code2=0，则“取”
- 若P_1P_2明显在窗口外：code1&code2≠0，则“弃”
- 在交点处把线段分为两段。其中一段完全在窗口外，可弃之。然后对另一段重复上述处理。



编码



线段裁剪

Cohen-Sutherland裁剪

如何判定应该与窗口的哪条边求交呢?

- 计算线段P1(x1,y1)P2(x2,y2)与窗口边界的交点

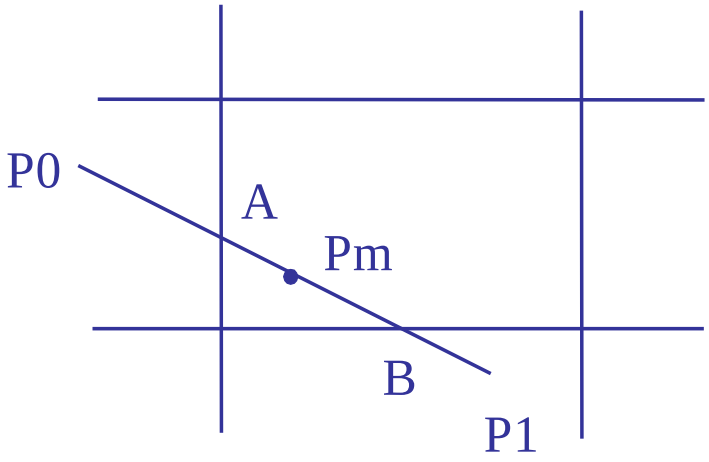
```
if(LEFT&code !=0)
    {       x=XL; y=y1+(y2-y1)*(XL-x1)/(x2-x1);}
else if(RIGHT&code !=0)
    {       x=XR; y=y1+(y2-y1)*(XR-x1)/(x2-x1);}
else if(BOTTOM&code !=0)
    {       y=YB; x=x1+(x2-x1)*(YB-y1)/(y2-y1);}
else if(TOP & code !=0)
    {       y=YT; x=x1+(x2-x1)*(YT-y1)/(y2-y1);}
```

Cohen-Sutherland直线裁剪算法小结

- 本算法的优点在于简单，易于实现。
- 可以简单的描述为将直线在窗口左边的部分删去，按左，右，下，上的顺序依次进行，处理之后，剩余部分就是可见的了。
- 在这个算法中求交点是很重要的，决定了算法的速度。另外，本算法对于其他形状的窗口未必同样有效。
- 特点：用编码方法可快速判断线段的完全可见和显然不可见。

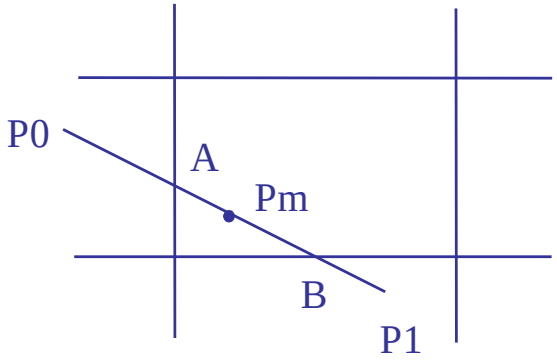
中点分割裁剪算法

- 基本思想：从P_0点出发找出离P_0最近的可见点，和从P_1点出发找出离P_1最近的可见点。这两个可见点的连线就是原线段的可见部分。
- 与Cohen-Sutherland算法一样首先对线段端点进行编码，并把线段与窗口的关系分为三种情况，对前两种情况，进行一样的处理；对于第三种情况，用中点分割的方法求出线段与窗口的交点。
- A、B分别为距P_0 、P_1最近的可见点，Pm为P_0P_1中点。

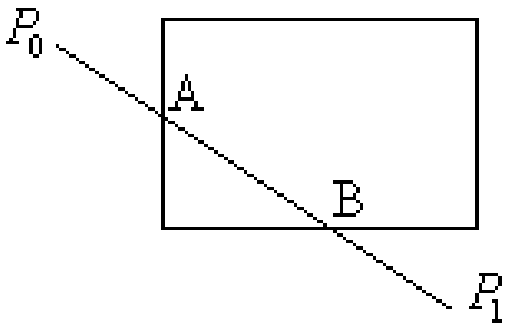


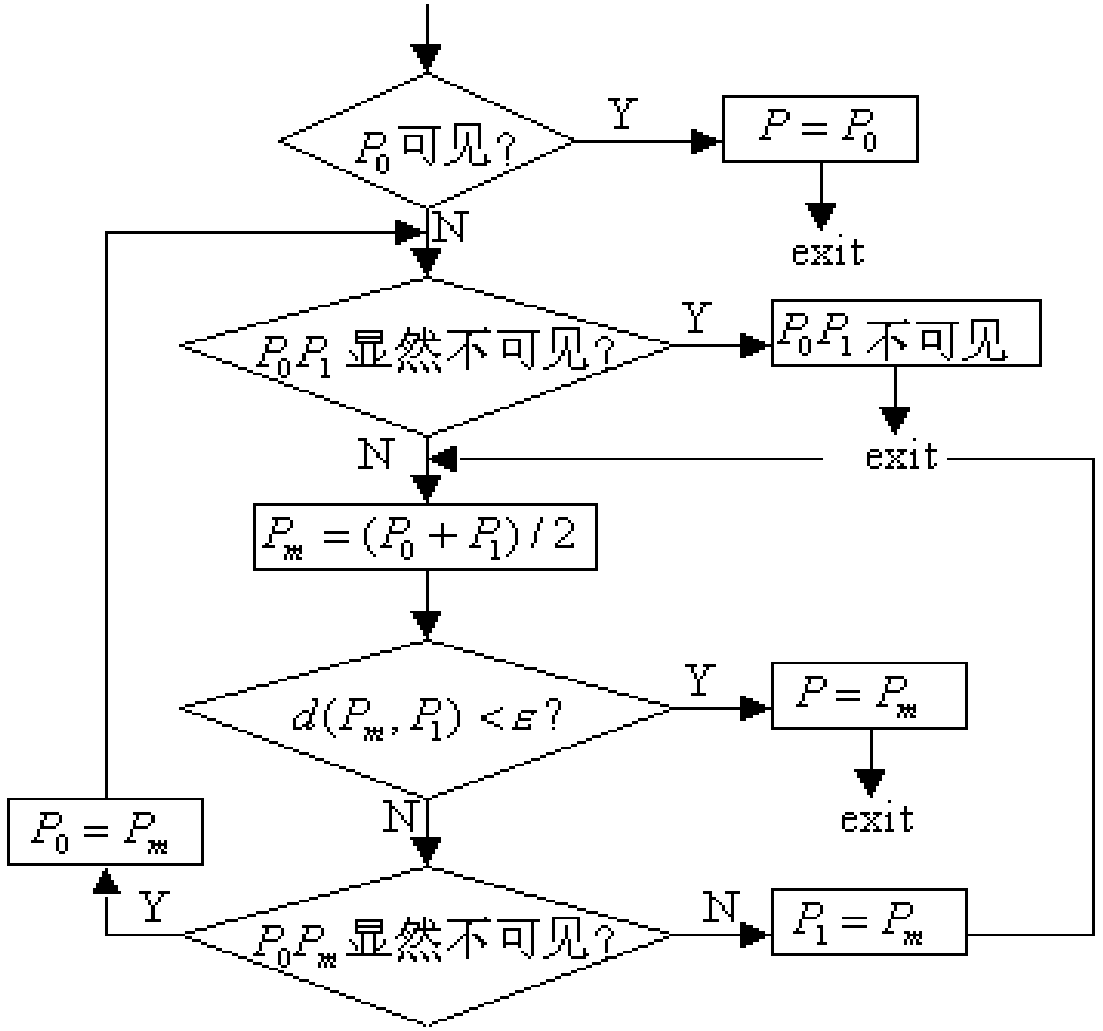
中点分割算法-求线段与窗口的交点

- 从P0出发找距离P0最近可见点采用中点分割方法
  - ① 先求出P0P1的中点Pm,
  - ② 若P0Pm不是显然不可见的，并且P0P1在窗口中有可见部分，则距P0最近的可见点一定落在P0Pm上，所以用P0Pm代替P0P1；
  - ③ 否则取PmP1代替P0P1。
  - ④ 再对新的P0P1求中点Pm。重复上述过程，直到PmP1长度小于给定的控制常数为止，此时Pm收敛于交点。
- 从P1出发找距离P1最近可见点采用上面类似方法。



中点分割裁剪算法

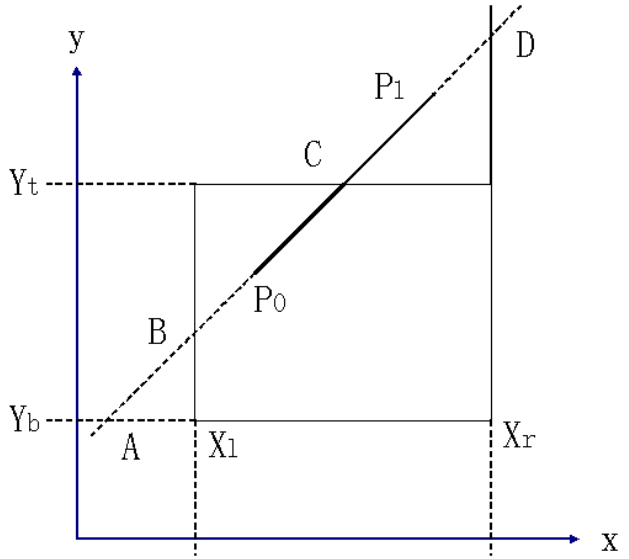

P_0
A
B
P_1


P_0可见?
Y
$P = P_0$
exit
N
P_0P_1显然不可见?
Y
P_0P_1不可见
exit
N
$P_m = (P_0 + P_1)/2$
$d(P_m, P_1) < \varepsilon$?
Y
$P = P_m$
exit
N
$P_0 = P_m$
Y
P_0P_m显然不可见?
N
$P_1 = P_m$

中点分割裁剪算法

- 对分辩率为2^N*2^N的显示器，上述二分过程最多进行N次。
- 主要过程只用到加法和除法运算，适合硬件实现，它可以用左右移位来代替乘除法，这样就大大加快了速度。

梁友栋-Barsky算法



- 设要裁剪的线段是P_0P_1，P_0P_1和窗口边界交于A,B,C,D四点
- 算法的基本思想是：
  ① 从A,B和P_0三点中找出最靠近P_1的点，图中要找的点是P_0。
  ② 从C,D和P_1中找出最靠近P_0的点。图中要找的点是C点。
  ③ 那么P_0C就是P_0P_1线段上的可见部分。

梁友栋-Barsky算法

线段的参数表示

$x=x_0+t\triangle x$

$y=y_0+t\triangle y$ $\quad 0<=t<=1$

$\triangle x=x_1-x_0 \quad \triangle y=y_1-y_0$

窗口边界的四条边分为两类：始边和终边。

$$\begin{cases} 若\Delta x \geq 0 \Rightarrow x = x_L 为始边，\ x = x_R 为终边。 \\ 若\Delta x < 0 \Rightarrow x = x_L 为终边，\ x = x_R 为始边。 \end{cases}$$

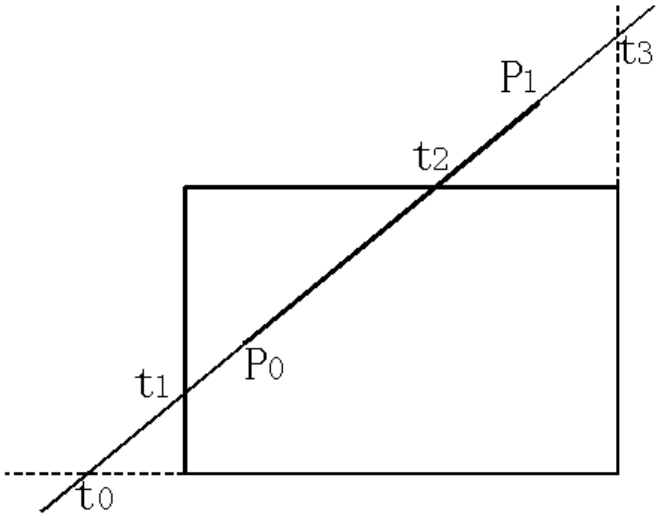
$$\begin{cases} 若\Delta y \geq 0 \Rightarrow y = y_B 为始边，\ y = y_T 为终边。 \\ 若\Delta y < 0 \Rightarrow y = y_B 为终边，\ y = y_T 为始边。 \end{cases}$$

梁友栋-Barsky算法：交点计算

① 求出P_0P_1与两条始边的交点参数t_0, t_1，
令$t_l=\max(t_0, t_1, 0)$，则t_L即为三者中离P_1最近的点的参数

② 求出P_0P_1与两条终边的交点参数t_2, t_3,
令$t_u=\min(t_2, t_3, 1)$，则t_U即为三者中离P_0最近的点的参数

③ 若$t_u > t_l$，则可见线段区间$[t_l, t_u]$

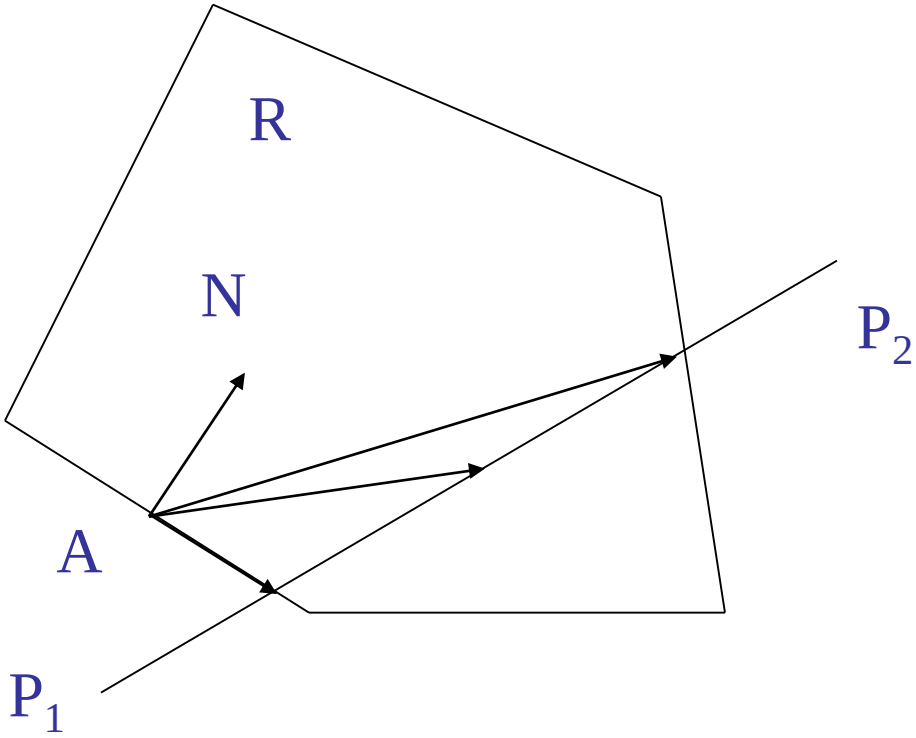


参数化算法(Cyrus-Beck)

- 考虑凸多边形区域R和直线段P_1P_2

$$P(t)=(P_2-P_1)*t+P_1$$

- 设A是区域R的边界上一点，N是区域边界在A点的内法线向量



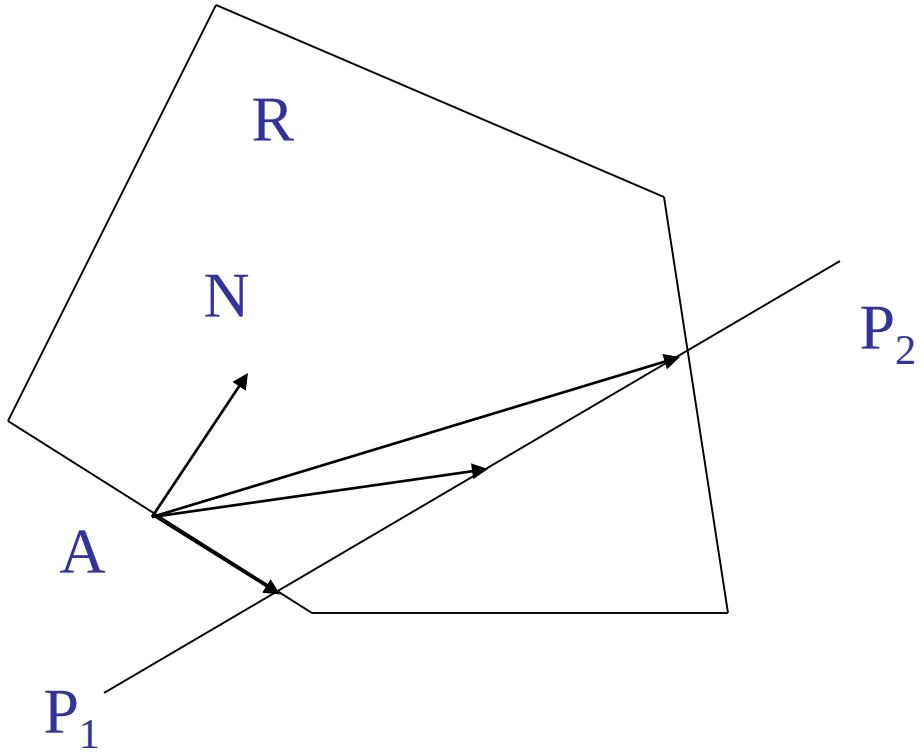
参数化算法(Cyrus-Beck)

则对于线段P_1P_2上任一点P(t)

① N ·(P(t)-A)<0 -> **边界**外侧

② N ·(P(t)-A)>0 -> **边界**内侧

③ N ·(P(t)-A)=0 -> 边界或其延长线上



参数化算法(Cyrus-Beck)

- 凸多边形的性质：点P(t)在凸多边形内的充要条件是，对于凸多边形**所有边界**上**任意一点**A和该点处内法向N，都有

$$N \cdot (P(t)-A) > 0$$

参数化算法(Cyrus-Beck)

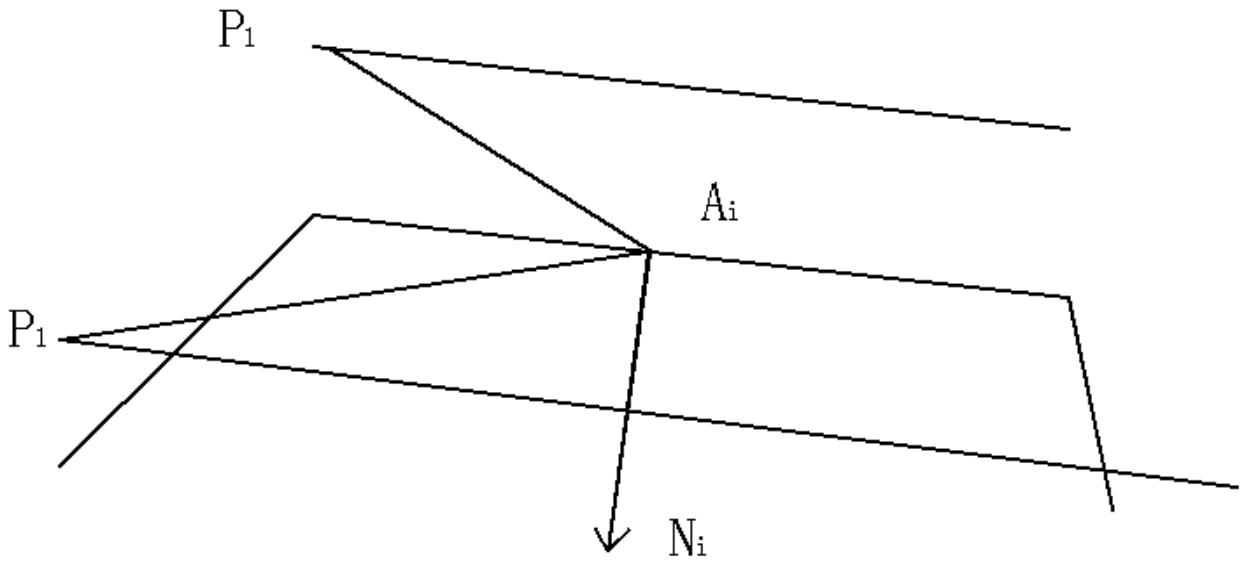
- K+1条边的多边形，可见线段参数区间的解:
  $N_i \cdot (p(t)-A_i) >= 0$, i=0,…,k, $0 \le t \le 1$.
  即: $N_i \cdot (P_1-A_i) + N_i \cdot (P_2-P_1)\, t >= 0$ (1)式
- 可得:

$$\begin{cases} N_i \bullet (P_2 - P_1) > 0 \Rightarrow t \ge -\dfrac{N_i \bullet (P_1 - A_i)}{N_i \bullet (P_2 - P_1)} \\ N_i \bullet (P_2 - P_1) < 0 \Rightarrow t \le -\dfrac{N_i \bullet (P_1 - A_i)}{N_i \bullet (P_2 - P_1)} \\ N_i \bullet (P_2 - P_1) = 0 \Rightarrow N_i \perp (P_2 - P_1) \end{cases}$$

- 令$t_i = N_i \cdot (P_1-A_i)/[N_i \cdot (P_2-P_1)]$

参数化算法(Cyrus-Beck)

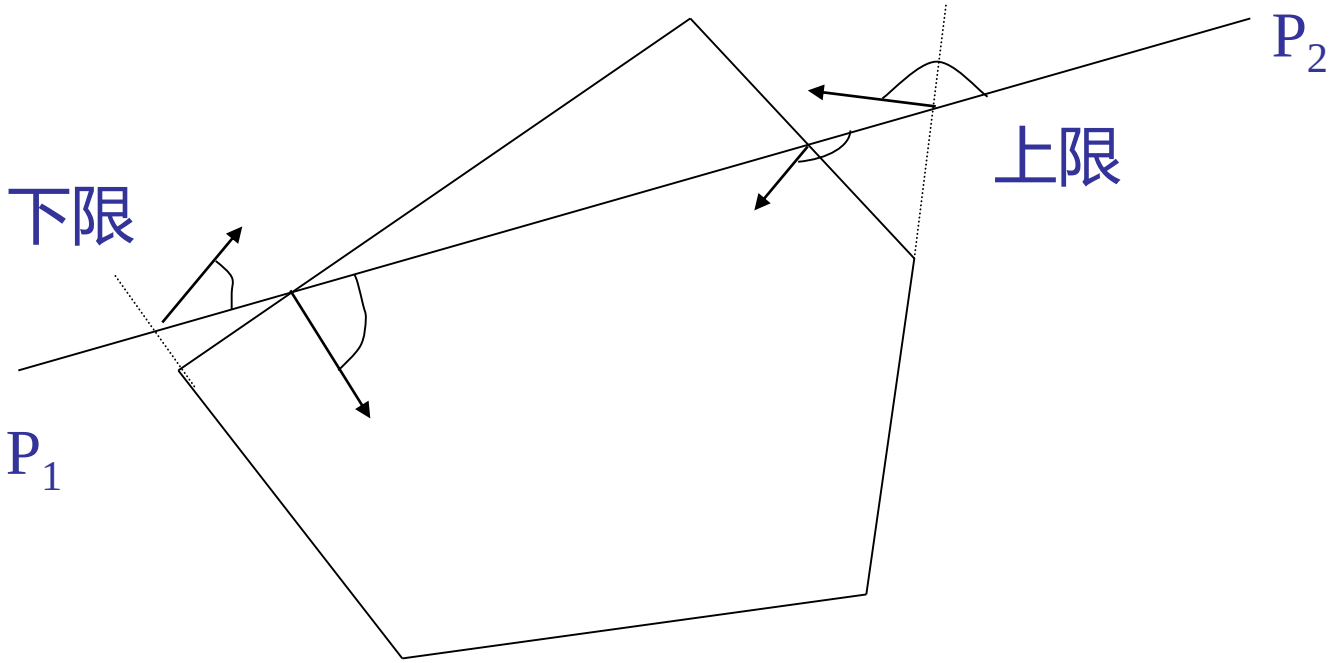
- $N_i \cdot (P_2-P_1) = 0$ -> 平行于对应边。
- 此时判断$N_i \cdot (P_1-A_i)$
  - 若$N_i \cdot (P_1-A_i) < 0$->$P_1 P_2$在多边形外侧->不可见，
  - 若$N_i \cdot (P_1-A_i) > 0$->$P_1 P_2$在多边形内侧->继续其它边的判断



参数化算法(Cyrus-Beck)

- 对于t值的选择：首先，要符合0≤t≤1；其次，对于凸窗口来说，每一个线段与其至多有两个交点，即有两个相应的t值。
- 所以我们可以把计算出的t值分成两组：
  - 一组为下限组，是分布在线段起点一侧的；
  - 一组为上限组，是分布在线段终点一侧的。
  - 只要找出下限组中的最大值及上限组中的最小值，就可确定线段了。
- 分组的依据是：
  - 如果$N_i \cdot (P_2-P_1) < 0$，则计算出的值属于上限组（距离终点近）
  - 如果$N_i \cdot (P_2-P_1) > 0$，则计算出的值属于下限组（距离起点近）

参数化算法的几何意义



- 上限组以$N_i \cdot (P_2-P_1) < 0$为特征，表示在该处沿P_1P_2方向前进将越来越远地离开多边形区域。
- 下限组以$Ni \cdot (P2-P1) > 0$为特征，表示在该处沿P1P2方向前进将接近或进入多边形内侧。

参数化算法(Cyrus-Beck)

因此，线段可见的交点参数：

① $t_l = \max\{0, \max\{t_i: N_i \cdot (P_2 - P_1) > 0\}\}$

② $t_u = \min\{1, \min\{t_i: N_i \cdot (P_2 - P_1) < 0\}\}$

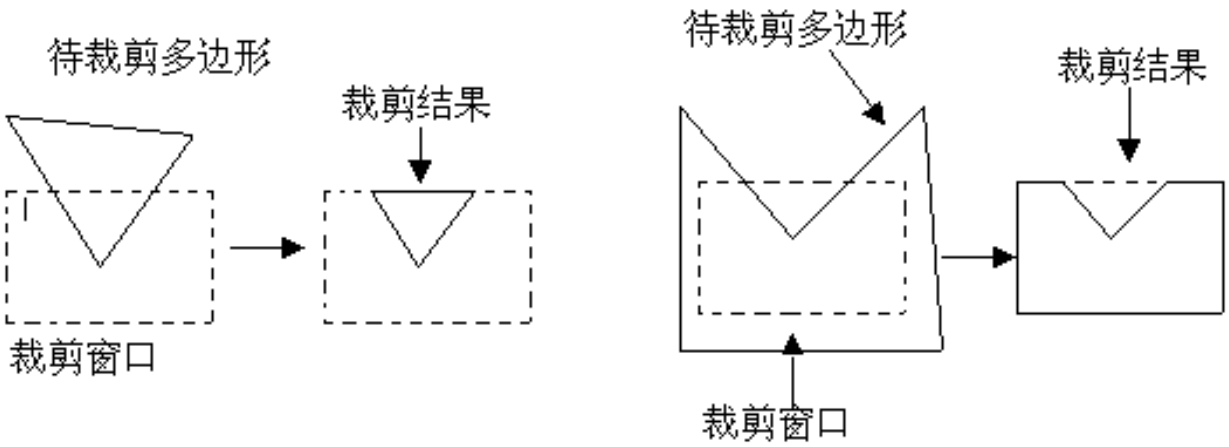
③ 若 $t_l <= t_u$, $[t_l, t_u]$是可见线段的交点参数区间，否则，线段不可见。

参数化算法

- 当凸多边形是矩形窗口且矩形的边与坐标轴平行时，该算法退化为Liang-Barsky算法。

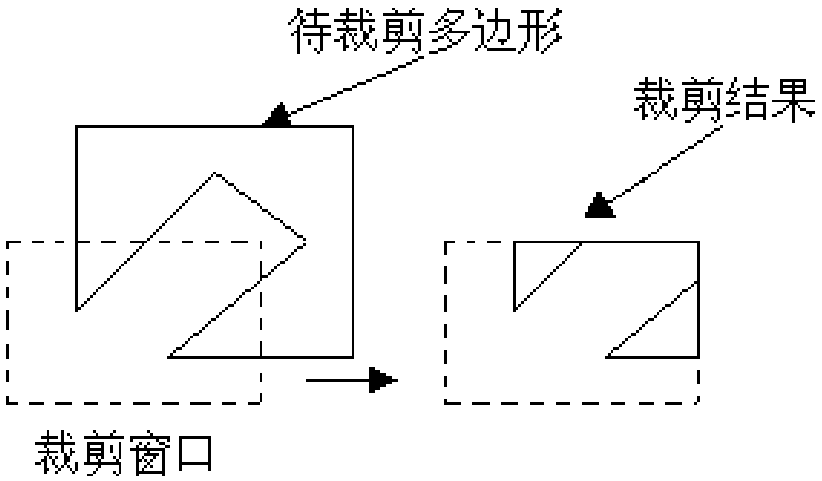
多边形裁剪

- **错觉**：直线段裁剪的组合？
- **新的问题**：1）边界不再封闭，需要用窗口边界的恰当部分来封闭它，如何确定其边界？



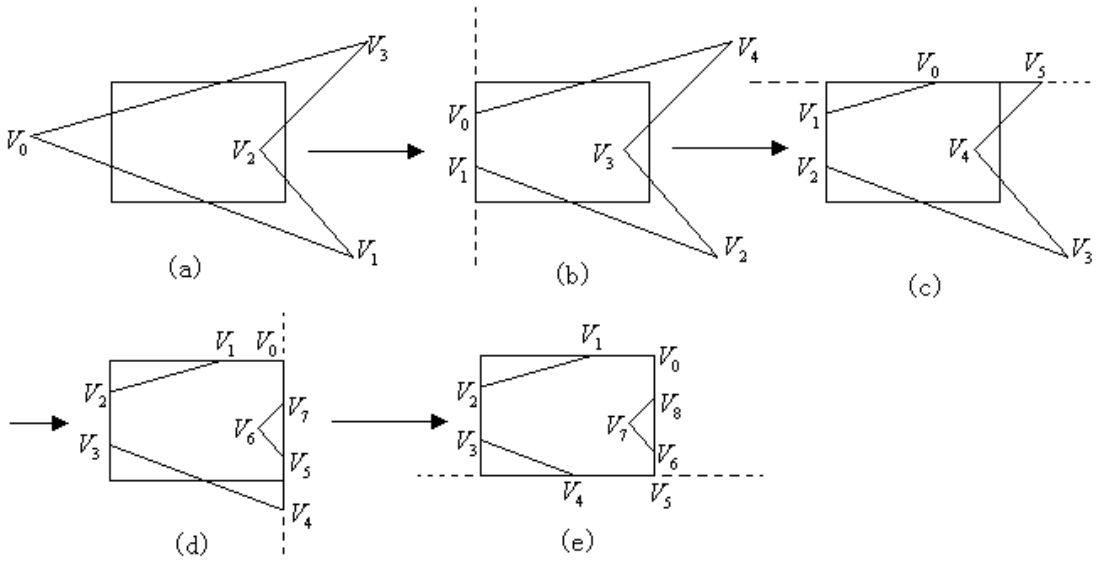
多边形裁剪

- 一个凹多边形可能被裁剪成几个小的多边形，如何确定这些小多边形的边界？



Sutherland-Hodgman算法

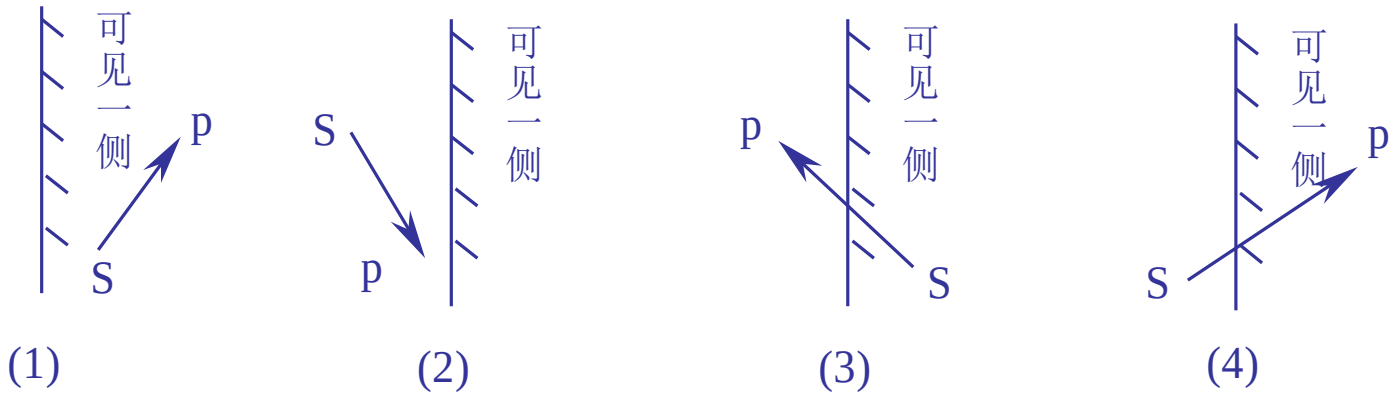
- **分割处理策略**：将多边形关于矩形窗口的裁剪分解为多边形关于窗口四边所在直线的裁剪。
- **流水线过程(左上右下)**：前边的结果是后边的输入。



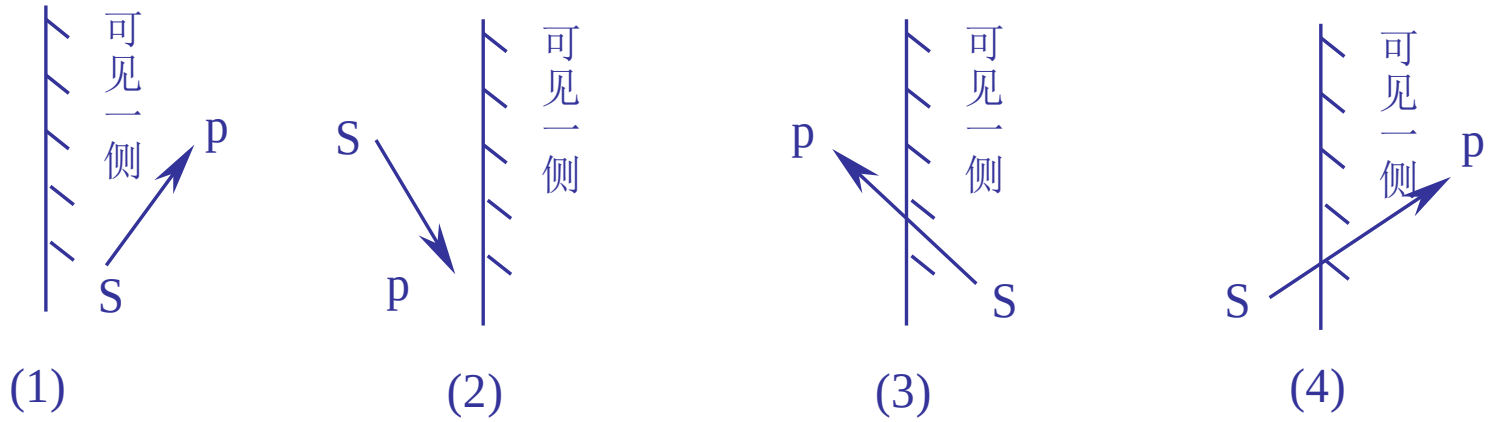
亦称逐边裁剪算法

Sutherland-Hodgman算法

- 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，该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

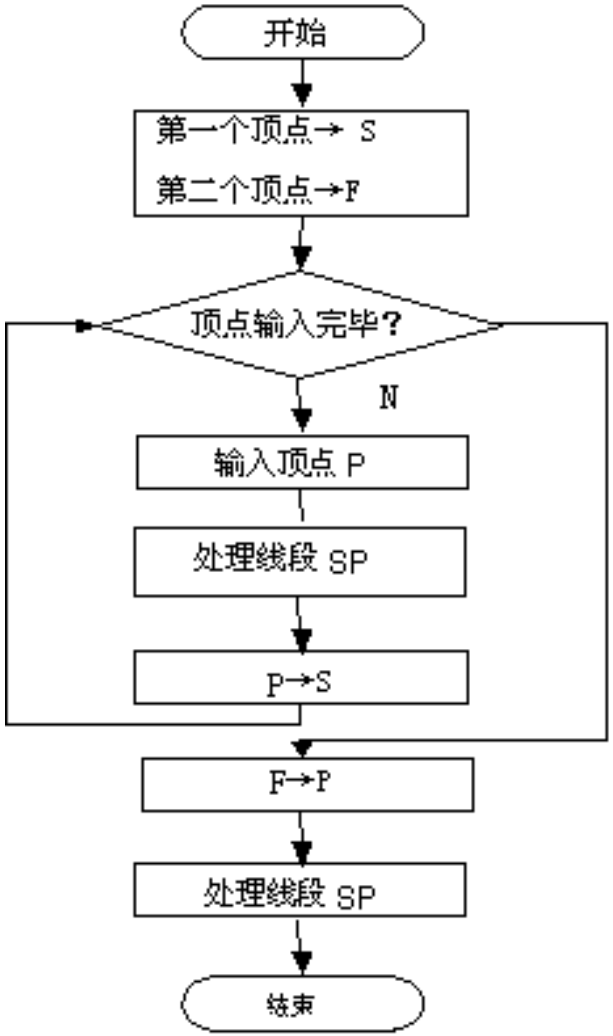


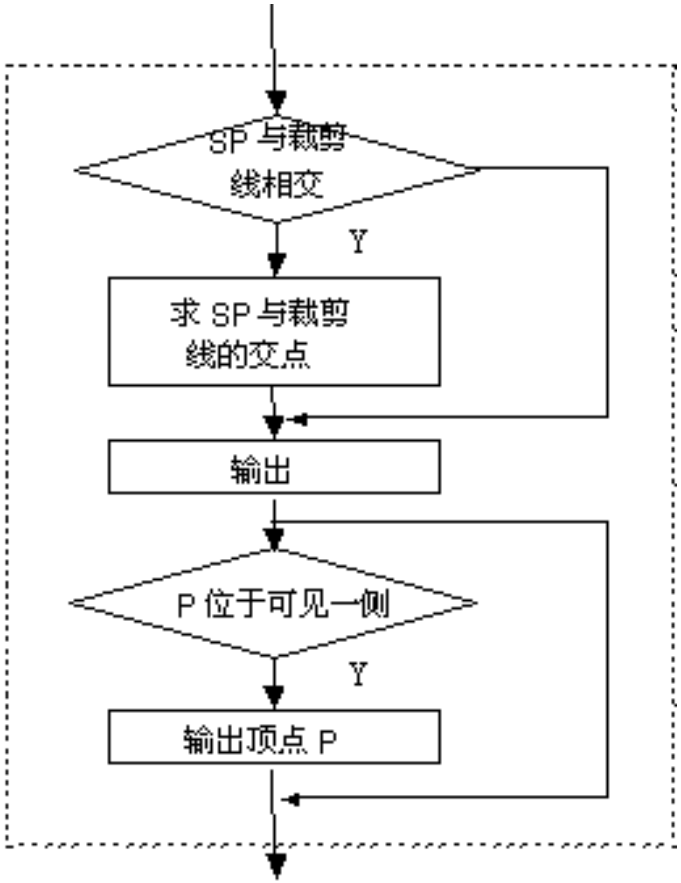
Sutherland-Hodgman算法



- 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和终点P

Sutherland-Hodgman算法框图





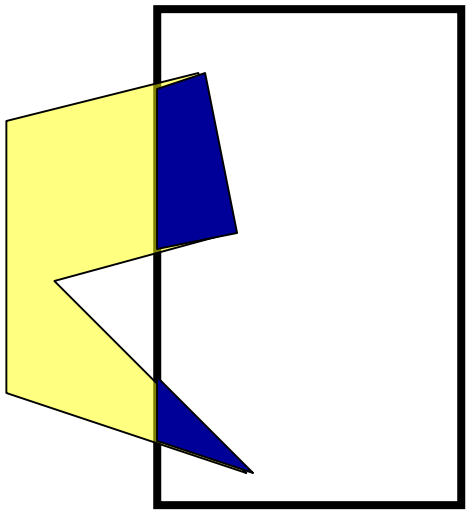
Sutherland-Hodgman算法

- 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，算法框图同上，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。

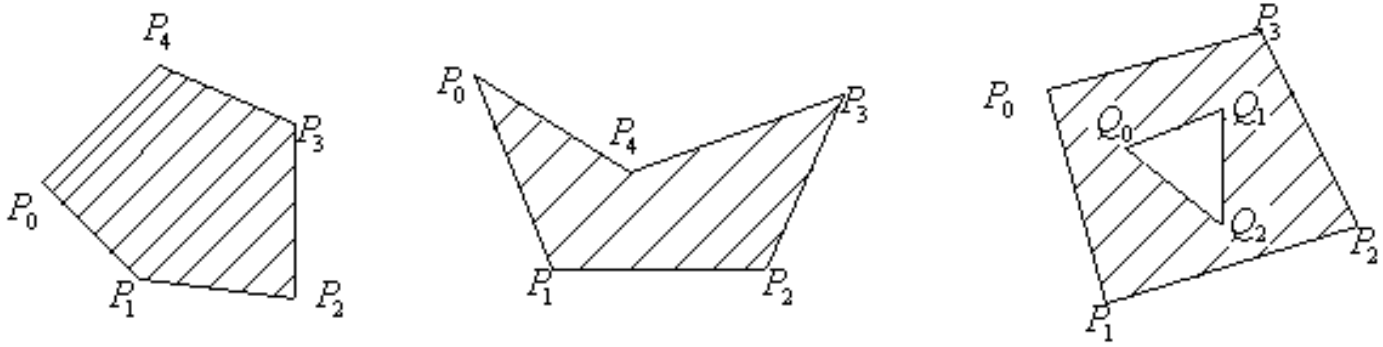
Sutherland-Hodgeman算法

- 对凸多边形应用本算法可以得到正确的结果，但是对凹多边形的裁剪将如图所示显示出一条多余的直线。
- 这种情况在裁剪后的多边形有两个或者多个分离部分的时候出现。因为只有一个输出顶点表，所以表中最后一个顶点总是连着第一个顶点。
- 解决这个问题有多种方法，
  ① 一是把凹多边形分割成若干个凸多边形，然后分别处理各个凸多边形。
  ② 二是修改本算法，沿着任何一个裁剪窗口边检查顶点表，正确的连接顶点对。
  ③ 再有就是Weiler-Atherton算法。



Weiler-Athenton算法

- 裁剪窗口为任意多边形（凸、凹、带内环）的情况：



  - 主多边形：被裁剪多边形，记为A
  - 裁剪多边形：裁剪窗口，记为B

Weiler-Athenton算法

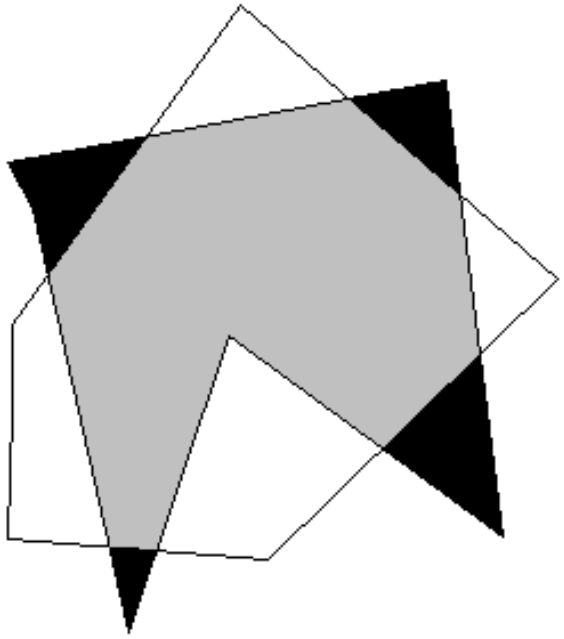
多边形顶点的排列顺序（使多边形区域位于有向边的左侧）

外环：逆时针；内环：顺时针

●主多边形和裁剪多边形把二维平面分成两部分。

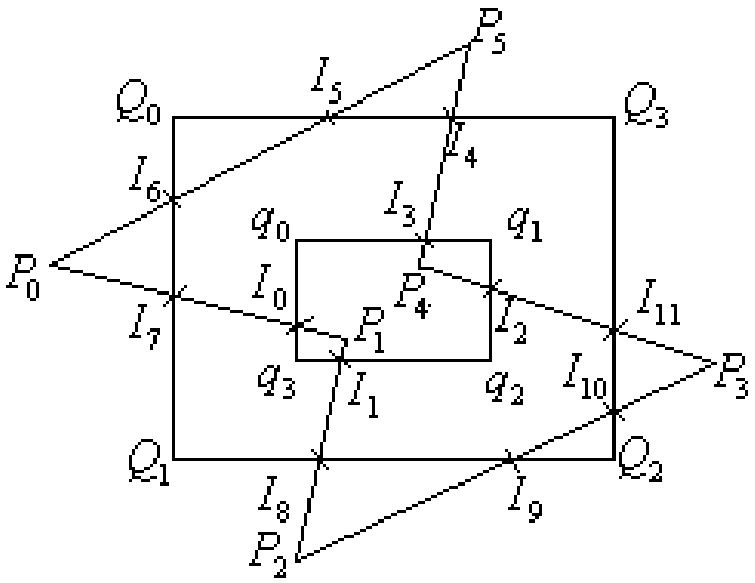
① 裁剪结果区域的边界由A的部分边界和B的部分边界两部分构成，

② 并且在交点处边界发生交替，即由A的边界转至B的边界，或由B的边界转至A的边界



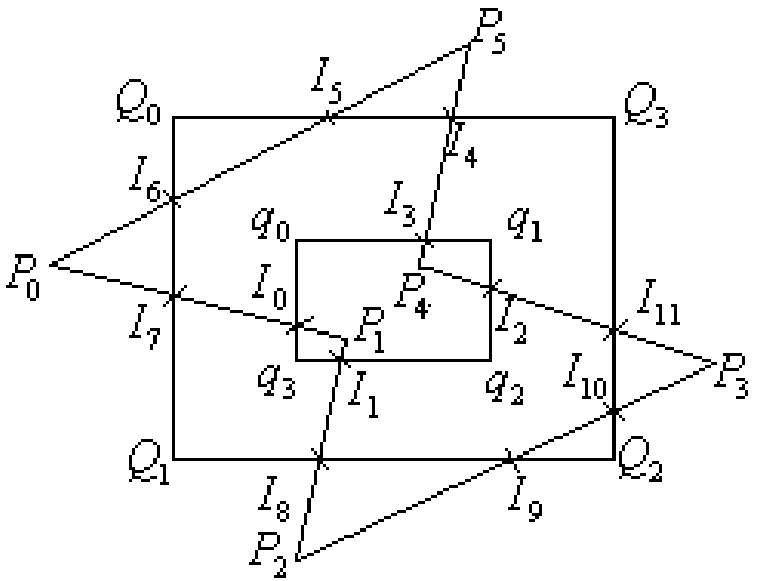
Weiler-Athenton算法

- 如果主多边形与裁剪多边形有交点，则交点成对出现，它们被分为如下两类：
  - 进点：主多边形边界由此进入裁剪多边形内
    如，I1,I3, I5, I7, I9, I11
  - 出点：主多边形边界由此离开裁剪多边形区域.
    如， I0,I2, I4, I6, I8, I10

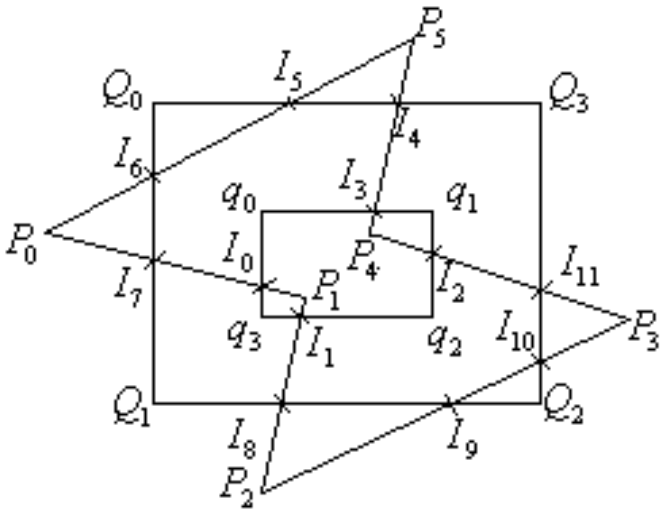


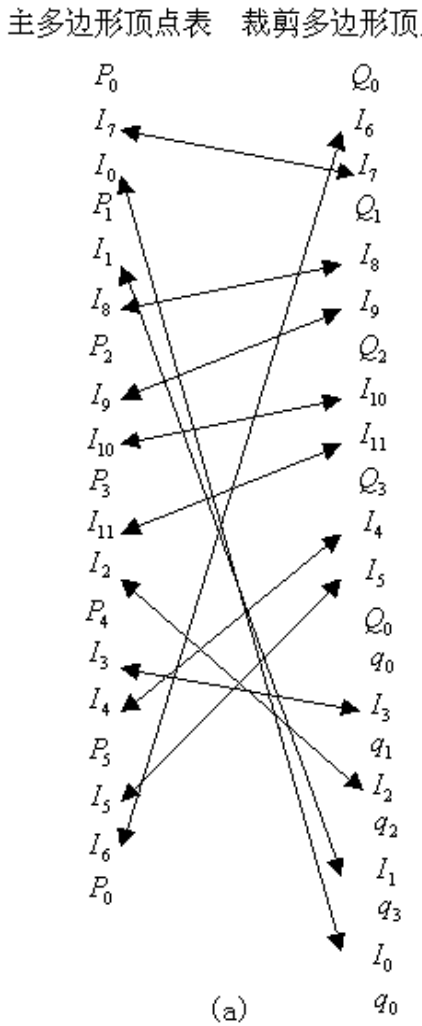
Weiler-Athenton算法

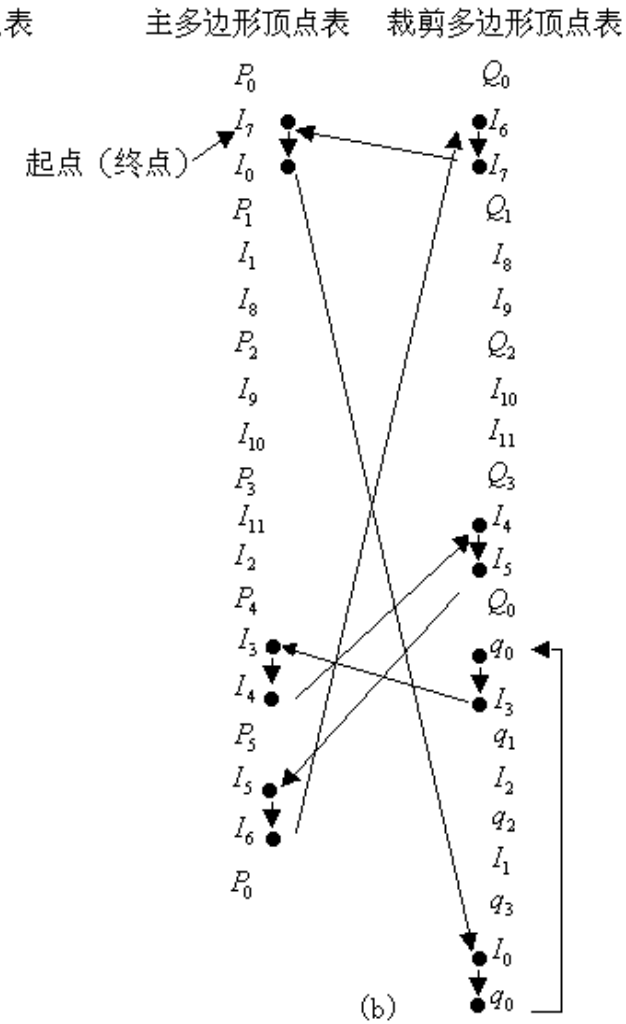
1. 建立主多边形和裁剪多边的顶点表。
2. 求主多边形和裁剪多边形的交点，并将这些交点按顺序插入两多边形的顶点表中。在两多边形顶点表中的相同交点间建立双向指针 。
3. 裁剪： 如果存在没有被跟踪过的交点，执行以下步骤：



Weiler-Athenton算法







Weiler-Athenton算法

① 建立空的裁剪结果多边形的顶点表.

② 选取任一没有被跟踪过的交点为始点，将其输出到结果多边形顶点表中.

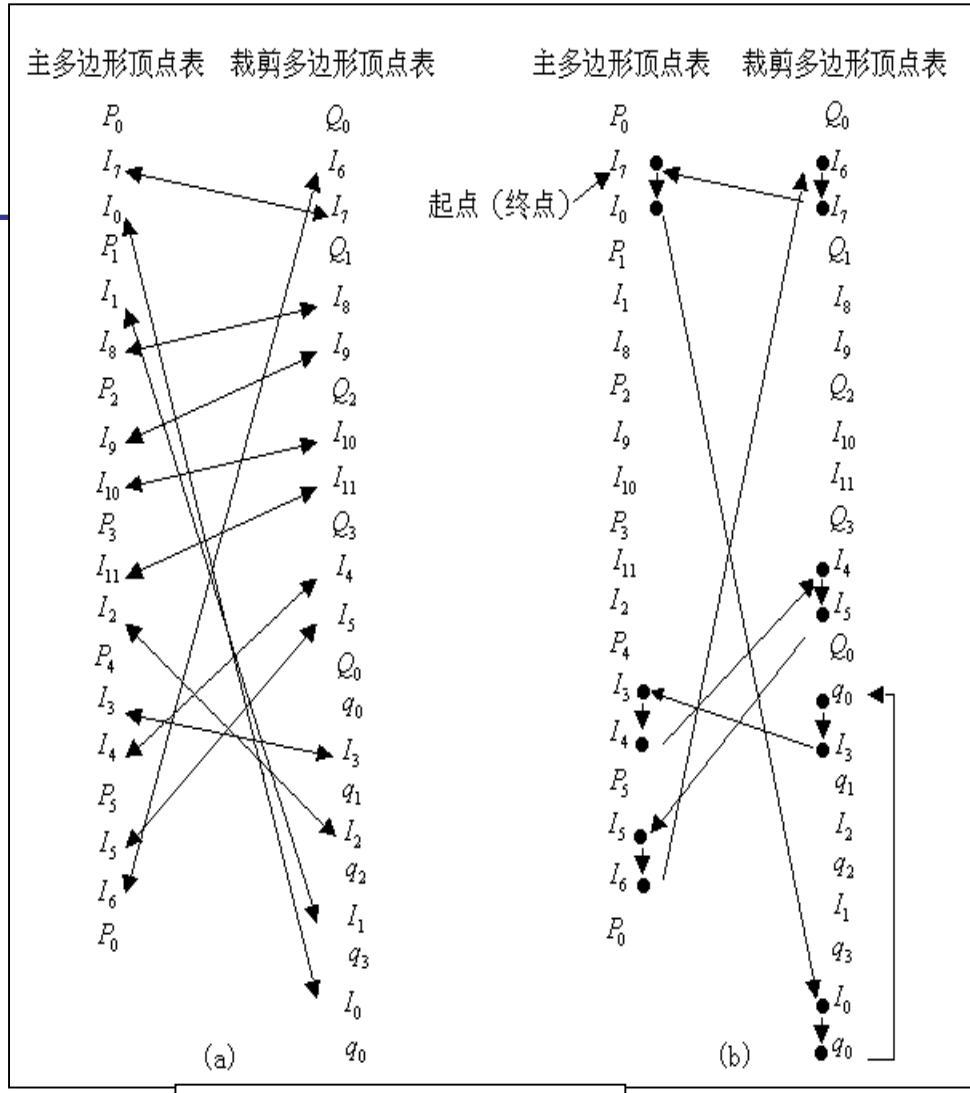
③ 如果该交点为进点，跟踪主多边形边边界；否则跟踪裁剪多边形边界.

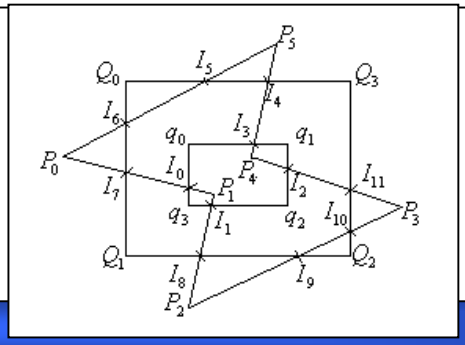
④ 跟踪多边形边界，每遇到多边形顶点，将其输出到结果多边形顶点表中，直至遇到新的交点.

⑤ 将该交点输出到结果多边形顶点表中，并通过连接该交点的双向指针改变跟踪方向（如果上一步跟踪的是主多边形边界，现在改为跟踪裁剪多边形边界；如果上一步跟踪裁剪多边形边界，现在改为跟踪主多边形边界）.

⑥ 重复(4)、(5)直至回到起点

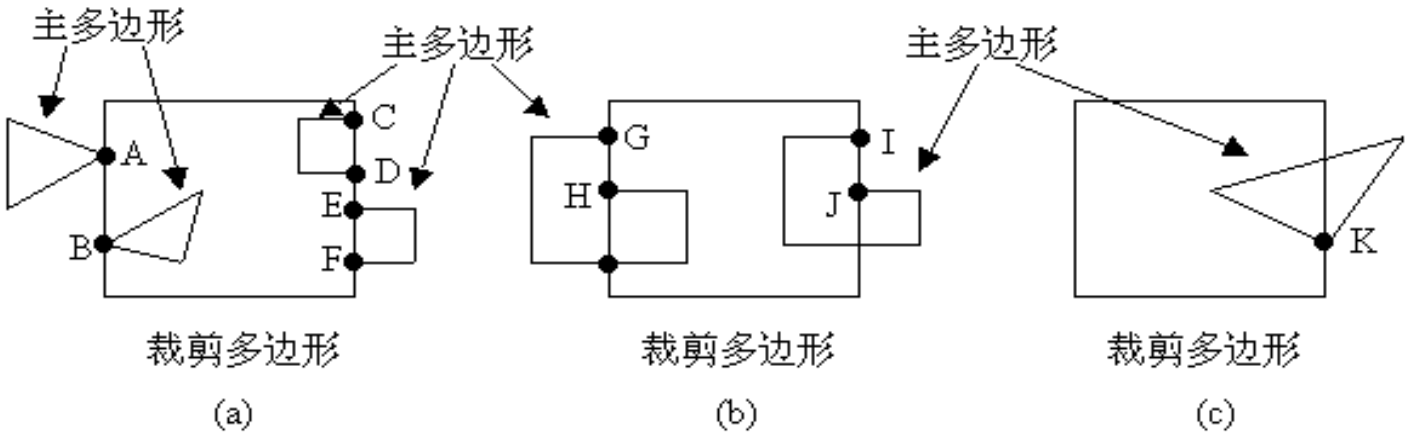
- 取I_7为起点，所得裁剪结果多边形$I_7I_0q_0I_3I_4I_5I_6I_7$。取I_8为起点，所得裁剪结果多边形为$I_8I_9I_{10}I_{11}I_2q_2I_1I_8$。





Weiler-Athenton算法

– 交点的奇异情况处理



① 与裁剪多边形的边重合的主多边形的边不参与求交点（CD, EF）；

② 对于顶点落在裁剪多边形的边上的主多边形的边，如果落在该裁剪边的内侧，将该顶点算作交点；而如果这条边落在该裁剪边的外侧，将该顶点不看作交点