第五章
图形变换与裁剪

苏小红

计算机科学与技术学院

哈尔滨工业大学

本章内容

1 窗口视图变换
2 二维图形几何变换
3 三维图形几何变换
4 投影变换
5 二维线段裁剪
6 多边形的裁剪

三维图形显示的基本问题（1/5）

1. 在二维屏幕上如何显示三维物体？

- 显示器屏幕、绘图纸等是二维的
- 显示对象是三维的
- 解决方法
  - 投影
  - 三维显示设备

三维图形显示的基本问题（2/5）

2. 如何表示三维物体？

- 二维形体的表示----直线段, 折线, 曲线段, 多边形区域
- 二维形体的输入----简单（图形显示设备与形体的维数一致）

- 三维形体的表示----空间直线段、折线、曲线段、多边形、曲面片
- 三维形体的输入、运算、有效性保证----困难
- 解决方法----各种用于形体表示的理论、模型、方法

三维图形显示的基本问题（3/5）

3. 如何反映遮挡关系？

- 物体之间或物体的不同部分之间存在相互遮挡关系
- 遮挡关系是空间位置关系的重要组成部分
- 解决方法----消除隐藏面与隐藏线

三维图形显示的基本问题（4/5）

4. 如何产生真实感图形

- 何谓真实感图形？
  - 逼真的
  - 示意的
- 人们观察现实世界产生的真实感来源于
  - 空间位置关系----近大远小的透视关系和遮挡关系
  - 光线传播引起的物体表面颜色的自然分布
- 解决方法
  - 建立光照明模型
  - 采用真实感图形绘制方法

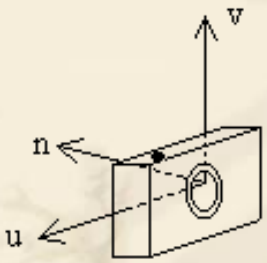
三维图形显示的基本问题（5/5）

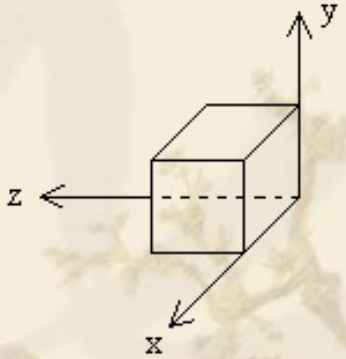
三维图形显示的基本研究内容

1. 投影变换
2. 三维形体的表示
3. 隐藏面消除算法
4. 光照明模型、真实感图形绘制方法

平面几何投影（1/15）

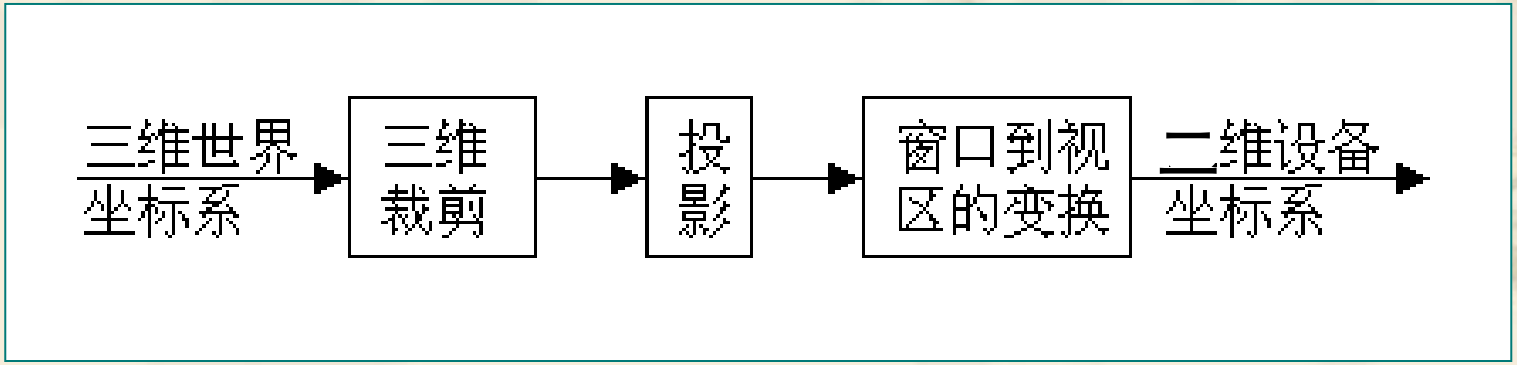
- 照像机模型与投影
  - 如何投影?
  - 生活中的类比——如何拍摄景物?
    - 拍摄过程
      - 选景，取景——裁剪
      - 对焦——参考点
      - 按快门——成像
    - 移动方式
      - 移动景物
      - 移动照相机
    - 两个坐标系





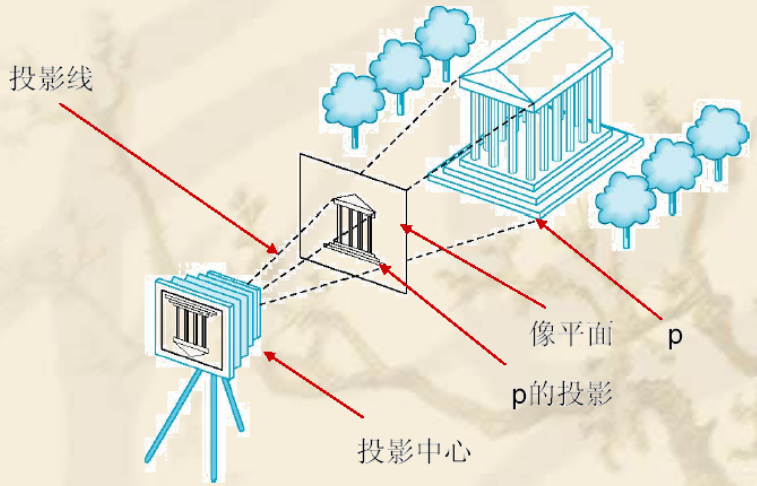
平面几何投影（2/15）

- 投影—照相机模型
  - 选定投影类型
  - 设置投影参数– 拍摄方向、距离等
  - 三维裁剪 –取景
  - 投影和显示 –成像
- 简单的三维图形显示流程图



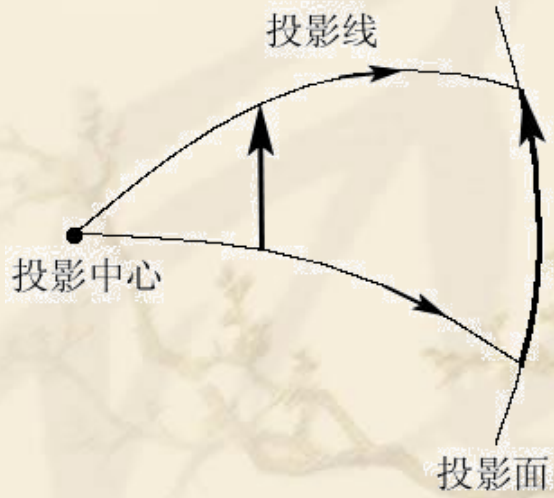
平面几何投影（3/15）

- 平面几何投影及其分类
  - 投影
    - 将n维的点变换成小于n维的点
    - 将3维的点变换成小于3维的点
  - 投影中心(COP:Center of Projection)
    - 视觉系统—观察点、视点
    - 电影放映机—光源
  - 投影面
    - 不经过投影中心
    - 平面--照相机底片
    - 曲面—球幕电影,视网膜



平面几何投影（4/15）

- 投影线
  - 从投影中心向物体上各点发出的射线
  - 直线—光线
  - 曲线—喷绘
- 平面几何投影
  - 投影面是平面
  - 投影线为直线
- 投影变换
  - 投影过程
  - 投影的数学表示

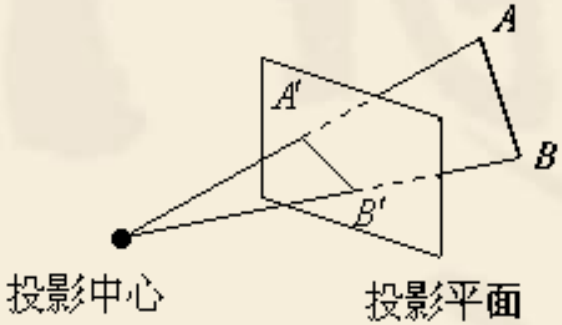


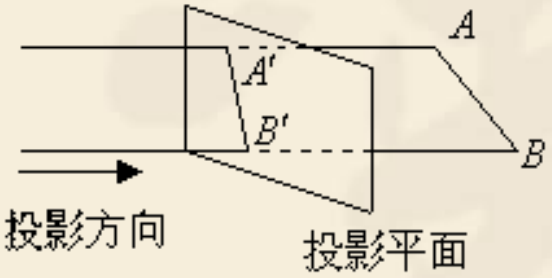
平面几何投影（5/15）

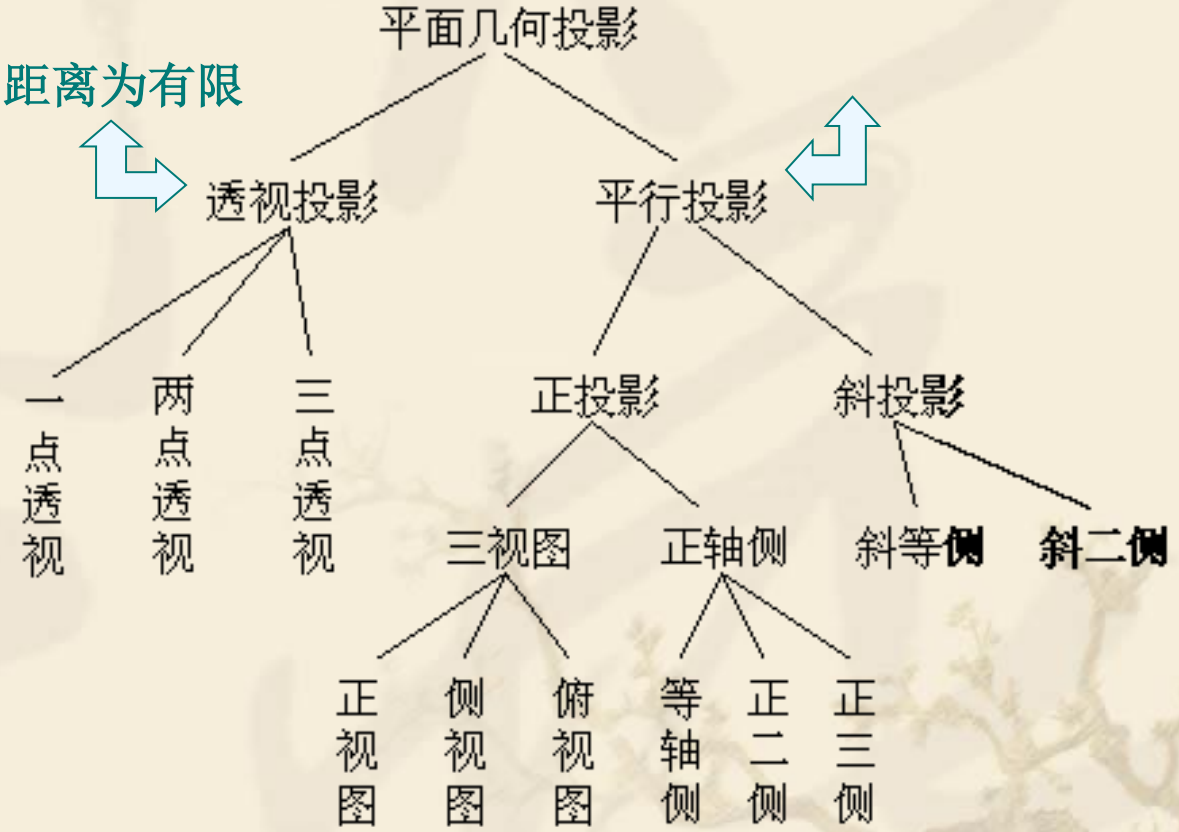
❖ 投影分类

投影中心与投影平面之间的距离为无限
是透视投影的极限状态

投影中心与投影平面之间的距离为有限

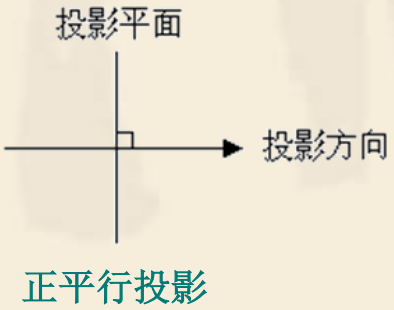




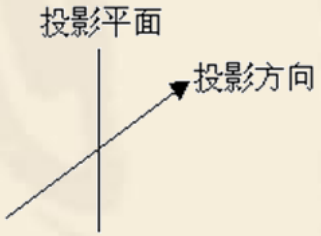


平面几何投影（6/15）

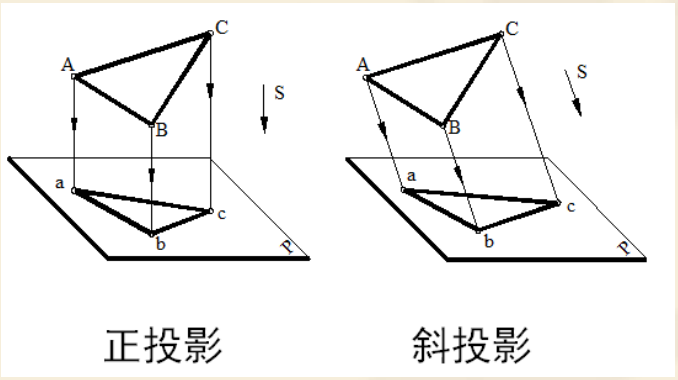
❖ 投影分类



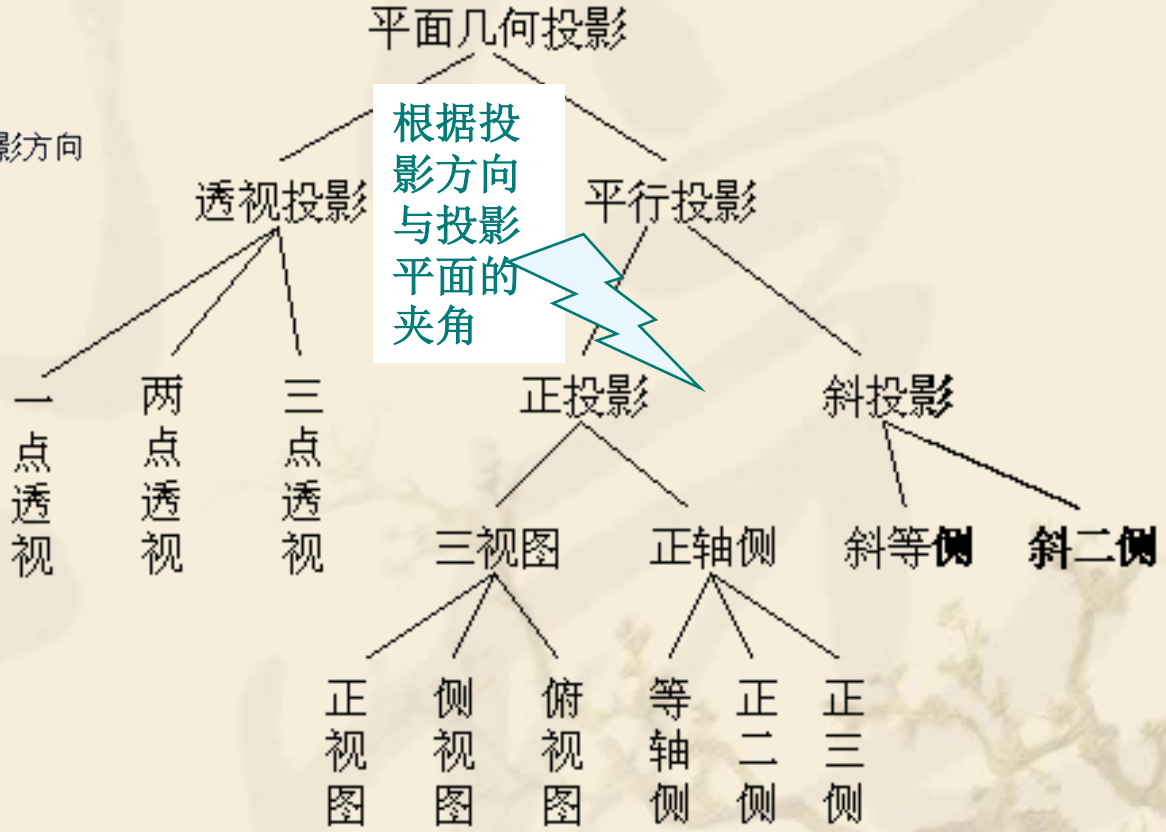
正平行投影



斜平行投影

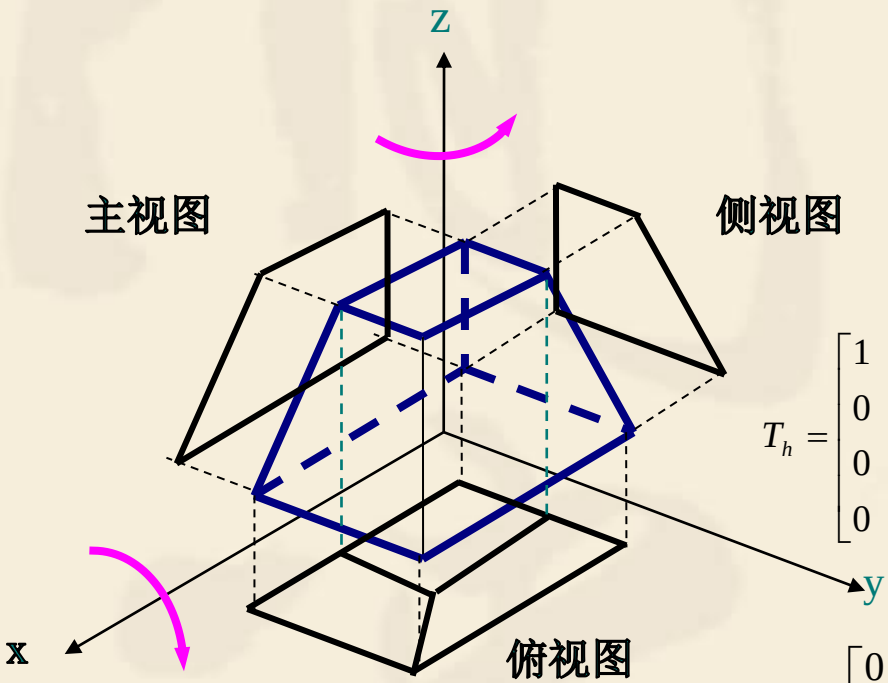


正投影　　斜投影

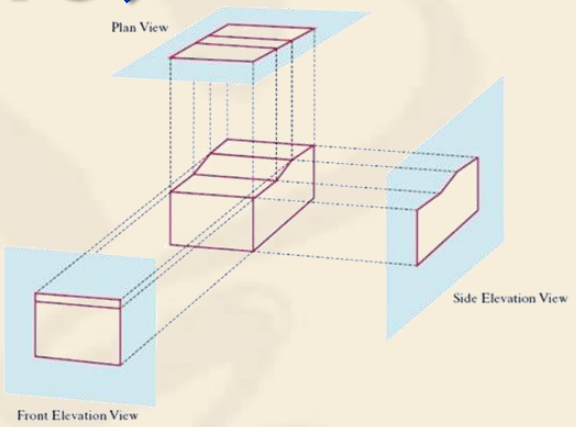


平面几何投影（7/15）

- 三视图：正视图、侧视图和俯视图



一个直角棱台的三视图



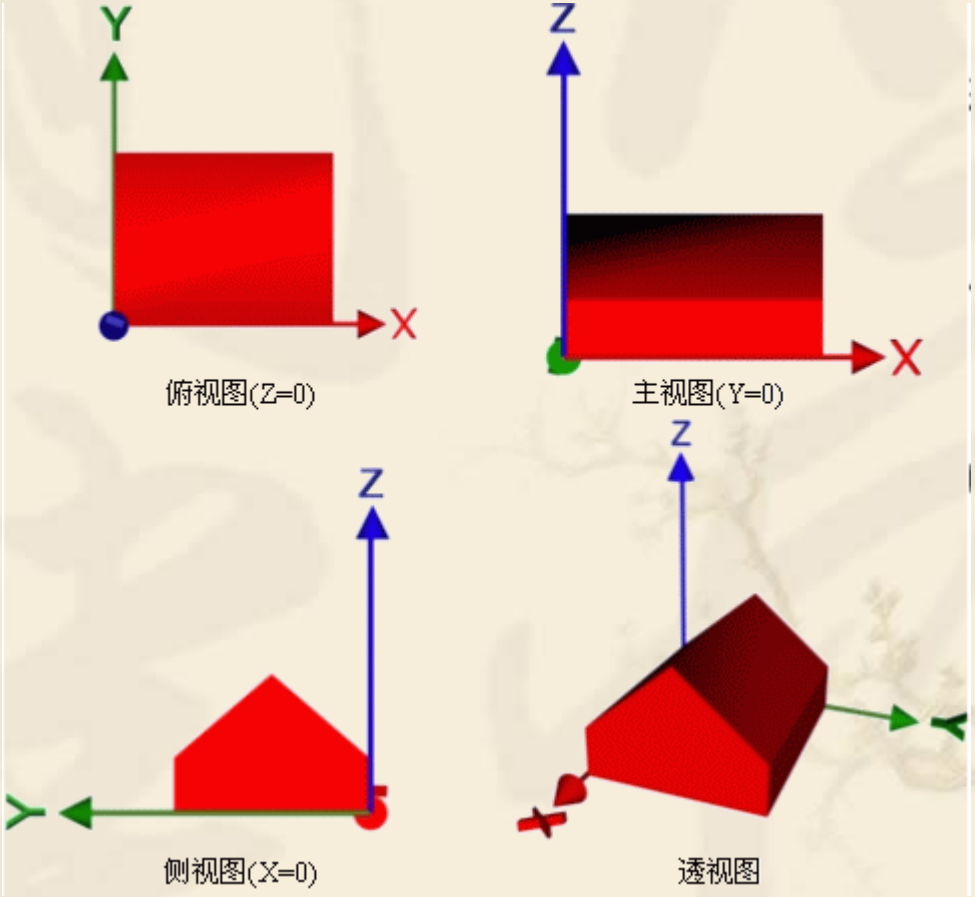
$$T_v = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_h = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(-90°) & \sin(-90°) & 0 \\ 0 & -\sin(-90°) & \cos(-90°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -z_p & 1 \end{bmatrix}$$

$$T_w = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 90° & \sin 90° & 0 & 0 \\ -\sin 90° & \cos 90° & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_L & 0 & 0 & 1 \end{bmatrix}$$

平面几何投影（8/15）

- 从三视图很难想象出实际物体的空间形状



平面几何投影（9/15）

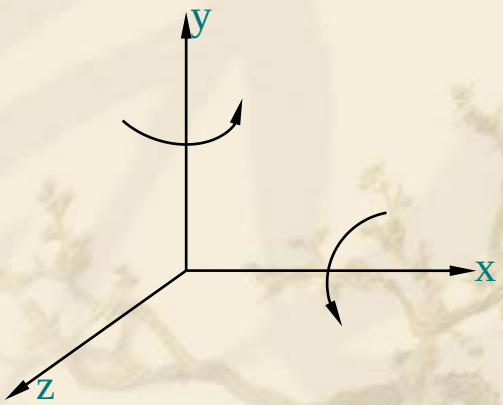
- 解决：
  - 投影平面不垂直于任何一个坐标轴——正轴测投影

$$R_{yx}=R_yR_x=\begin{bmatrix}\cos\beta & 0 & -\sin\beta & 0\\ 0 & 1 & 0 & 0\\ \sin\beta & 0 & \cos\beta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\alpha & \sin\alpha & 0\\ 0 & -\sin\alpha & \cos\alpha & 0\\ 0 & 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}\cos\beta & \sin\beta\sin\alpha & -\sin\beta\cos\alpha & 0\\ 0 & \cos\alpha & \sin\alpha & 0\\ \sin\beta & -\cos\beta\sin\alpha & \cos\beta\cos\alpha & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$T=R_{yx}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}\cos\beta & \sin\beta\sin\alpha & 0 & 0\\ 0 & \cos\alpha & 0 & 0\\ \sin\beta & -\cos\beta\sin\alpha & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

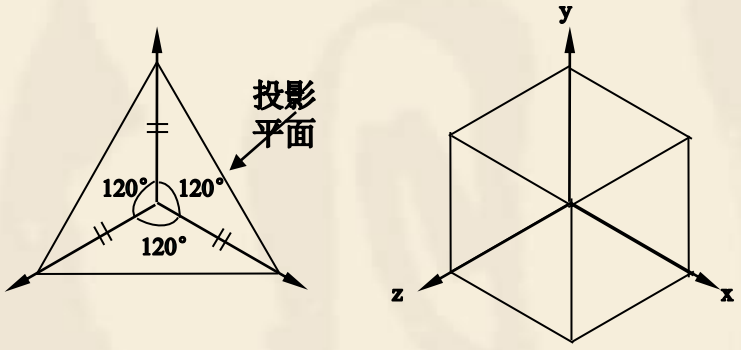
投影方程：

$$[x' \quad y' \quad z' \quad 1]=[x \quad y \quad z \quad 1]T$$



正轴测投影平面的定义

平面几何投影（10/15）



正方体的正等轴测投影

三个单位向量将投影成三个长度相等的平面向量，即三根坐标轴有相同的变形系数

$$T=\begin{bmatrix}\cos\beta & \sin\beta\sin\alpha & 0 & 0\\ 0 & \cos\alpha & 0 & 0\\ \sin\beta & -\cos\beta\sin\alpha & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$[x' \quad y' \quad z' \quad 1]_x=[1 \quad 0 \quad 0 \quad 1]T=[\cos\beta \quad \sin\beta\sin\alpha \quad 0 \quad 1].$$

这三个单位向量的长度分别变为：

$$\sqrt{\cos^2\beta+(\sin\beta\sin\alpha)^2}.$$

$$[x' \quad y' \quad z' \quad 1]_y=[0 \quad 1 \quad 0 \quad 1]T=[0 \quad \cos\alpha \quad 0 \quad 1].$$

$$\sqrt{\cos^2\alpha}.$$

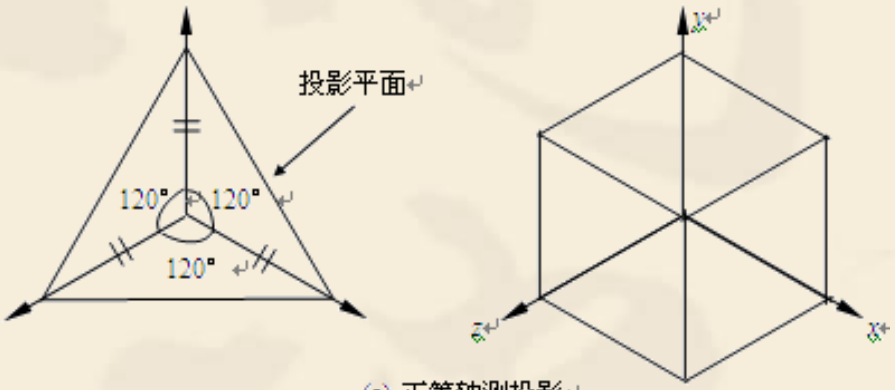
$$[x' \quad y' \quad z' \quad 1]_z=[0 \quad 0 \quad 1 \quad 1]T=[\sin\beta \quad -\cos\beta\sin\alpha \quad 0 \quad 1].$$

$$\sqrt{\sin^2\beta+(-\cos\beta\sin\alpha)^2}.$$

平面几何投影（11/15）

$$\sqrt{\cos^2\beta+(\sin\beta\sin\alpha)^2}=\sqrt{\cos^2\alpha}.$$

$$\sqrt{\sin^2\beta+(-\cos\beta\sin\alpha)^2}=\sqrt{\cos^2\alpha}.$$



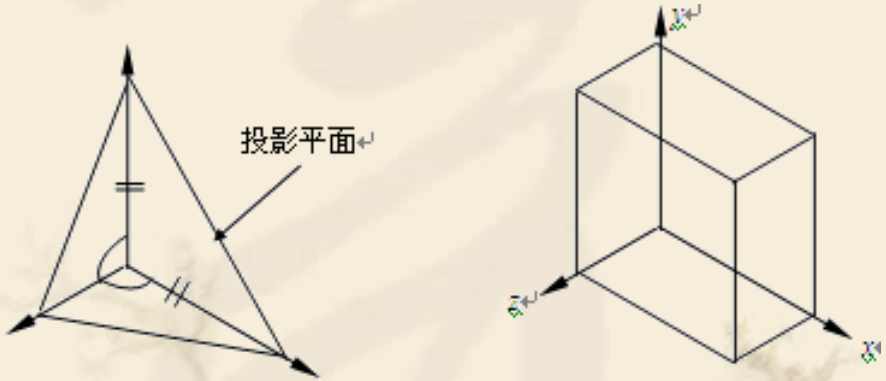
(a) 正等轴测投影

$$\alpha=35.264°$$

$$\beta=45°$$

$$\sqrt{\cos^2\beta+(\sin\beta\sin\alpha)^2}=\sqrt{\cos^2\alpha}.$$

$$\sin^2\beta=\frac{\sin^2\alpha}{1-\sin^2\alpha}$$



(b)正二轴测投影

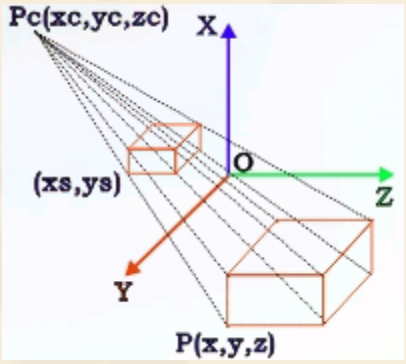
$$\alpha_1=\arcsin\sqrt{1/8}=20.705°$$

$$\alpha_2=90°\text{（无意义，舍去）}$$

$$\beta=\arcsin\sqrt{1/7}=22.208°$$

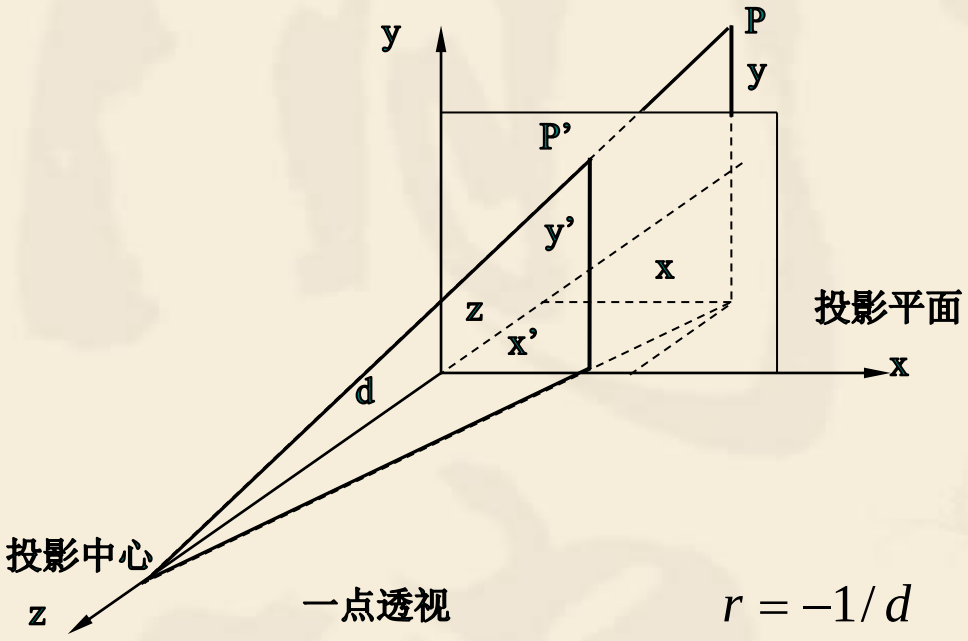
平面几何投影（12/15）

- 透视投影
  - 投影中心与投影平面之间的距离为有限
  - 参数：投影方向，距离
  - 例子：室内白炽灯的投影，视觉系统



- 特点：
  - 产生近大远小的视觉效果，由它产生的图形深度感强，看起来更加真实。

平面几何投影（13/15）



一点透视

- 透视投影投影方程

$$\frac{x'}{d}=\frac{x}{(|z|+d)}=\frac{x}{-z+d}$$

$$\frac{y'}{d}=\frac{y}{(|z|+d)}=\frac{y}{-z+d}$$

$$x'=\frac{x}{(-z/d)+1} \qquad y'=\frac{y}{(-z/d)+1}$$

$$r=-1/d$$

$$T=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&r\\0&0&0&1\end{bmatrix}$$

$$[x' \quad y' \quad z' \quad H]=[x \quad y \quad z \quad 1]T=[x \quad y \quad z \quad rz+1]$$

$$[x' \quad y' \quad z' \quad 1]=\left[\frac{x}{rz+1} \quad \frac{y}{rz+1} \quad \frac{z}{rz+1} \quad 1\right]$$

平面几何投影（14/15）

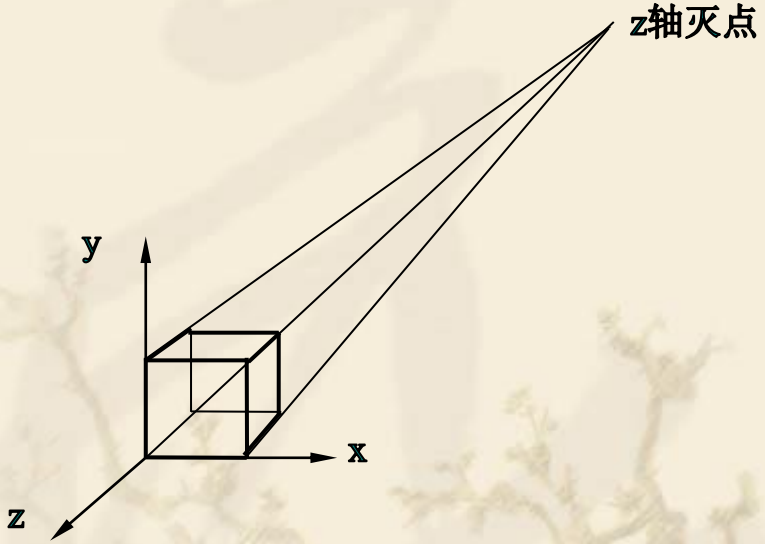
- 灭点：
  - 不平行于投影平面的平行线，经过透视投影之后相交于一点，称为灭点.

灭点的个数?　　灭点的位置?

$$[x \quad y \quad z \quad H] = [0 \quad 0 \quad 1 \quad 0] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix} = [0 \quad 0 \quad 1 \quad r]$$

无穷远点 → 灭点

$$[x' \quad y' \quad z' \quad 1] = [0 \quad 0 \quad 1/r \quad 1]$$



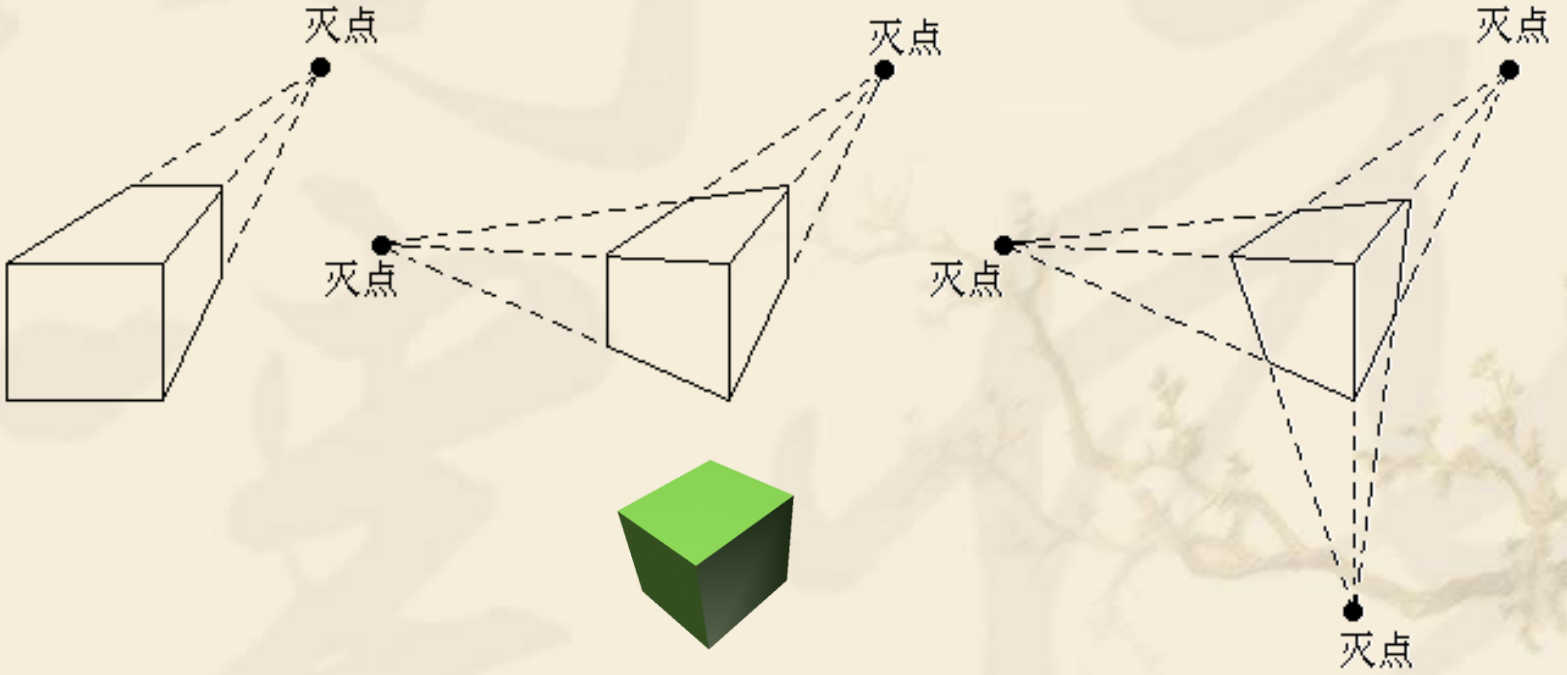
正方体的一点透视及其灭点

空间平行线可认为是相交于无穷远点，
灭点可以看成是无穷远点经透视投影后得到的点

平面几何投影（15/15）

- 主灭点:平行于坐标轴的平行线产生的灭点。
  - 一点透视
  - 两点透视
  - 三点透视

主灭点的个数由什么决定?



(a)一点透视　(b) 两点透视　(c)三点透视

5.5 直线段裁剪（Line clipping）

- 直接求交算法
- Cohen-Sutherland算法
- 中点分割裁剪算法
- 梁友栋-Barsky算法

5.6 多边形裁剪（Ploygon clipping）

- Sutherland_Hodgman算法
- Weiler-Atherton算法

本章内容

1 窗口视图变换
2 二维图形几何变换
3 三维图形几何变换
4 投影变换
5 二维线段裁剪
6 多边形的裁剪

5.5 二维线段裁剪

- 裁剪（clipping）的目的
  - 判断图形元素是否在裁剪窗口之内并找出其位于内部的部分
- 裁剪窗口
  - 矩形、圆形、一般多边形
- 被裁剪对象
  - 线段、多边形、曲线、字符
- 裁剪与覆盖的区别

5.5 二维线段裁剪

- 矩形裁剪窗口：

$$[x_{min}，\ x_{max}] \times [y_{min}，\ y_{max}]$$

- 待裁剪线段：

$$P_0(x_0, y_0)P_1(x_1, y_1)$$

5.5 二维线段裁剪

- 把直线当作点的集合，逐点裁剪
  - 点 (x，y) 在窗口内的充分必要条件：

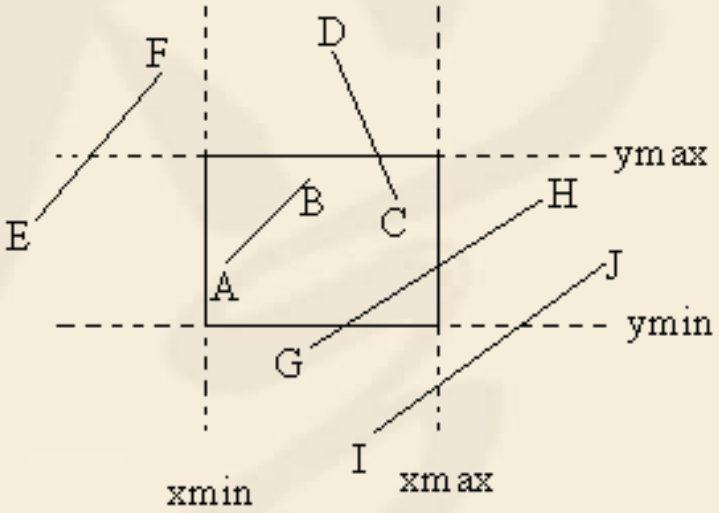
$$x_{\min} \le x \le x_{\max}$$

$$y_{\min} \le y \le y_{\max}$$

- 问题：没有把直线当作一个整体来裁剪，费时，精度不高
- 设计裁剪算法的核心问题
  - 提高裁剪效率

5.5 二维线段裁剪

- 待裁剪线段和窗口的关系
  1. 完全落在窗口内
  2. 完全落在窗口外
  3. 部分在内，部分在外



- 任何平面线段在凸多边形窗口进行裁剪后，落在窗口内的线段不会多于1条
- 裁剪处理的基础
  - 图元关于窗口内外关系的判别
  - 图元与窗口的求交

5.5 二维线段裁剪

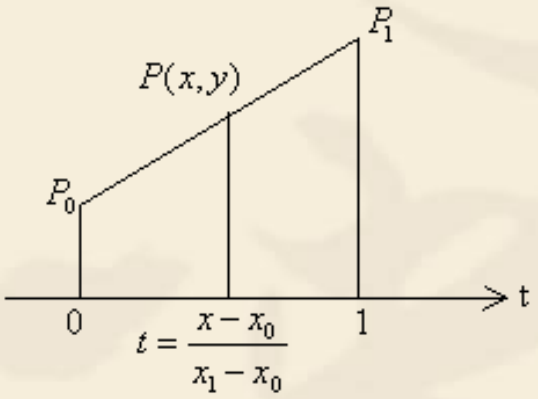
矩形窗口线段裁剪

- 直接求交算法
- Cohen-Sutherland算法
- 中点分割裁剪算法
- 梁友栋-Barsky算法

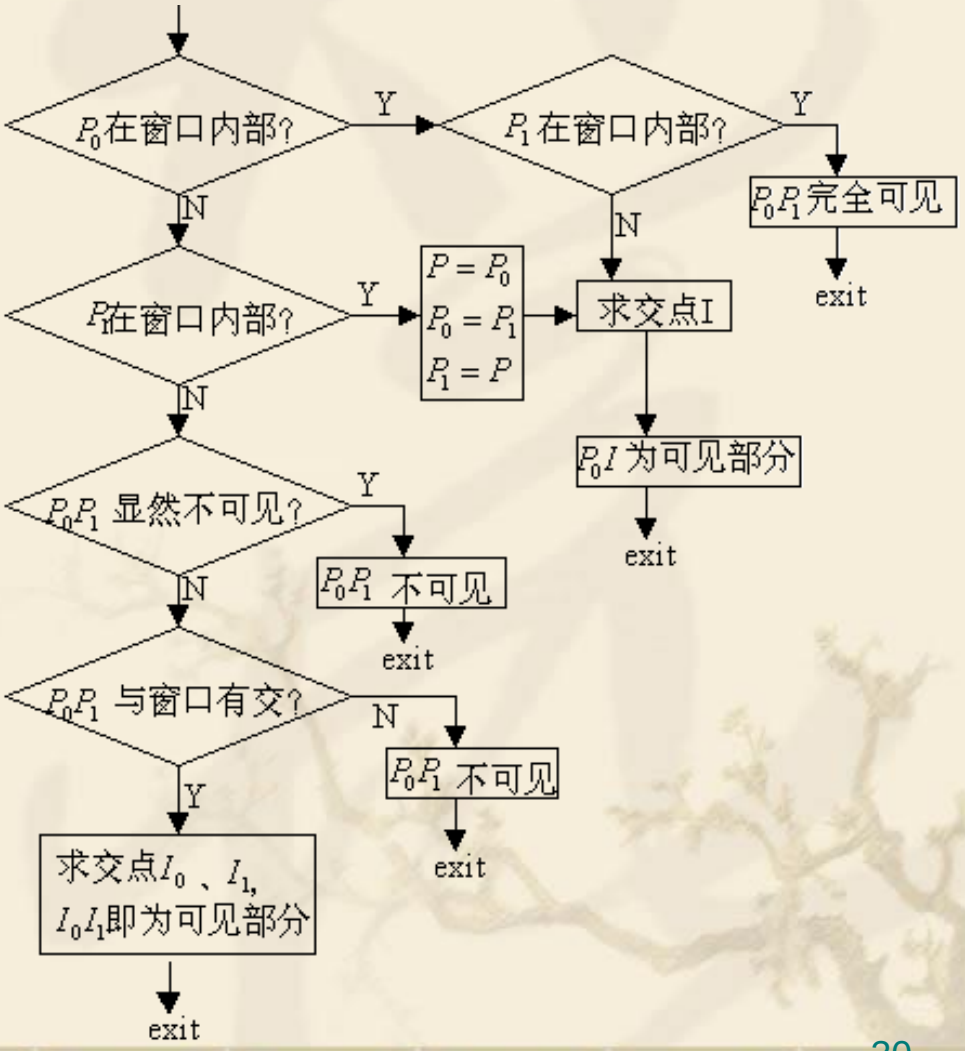
5.5 二维线段裁剪

直接求交算法

直线与窗口边都写成参数矢量方程，直线求交——求参数值



$$P = P(t) = P0 + t(P1 - P0) = (1-t)P0 + tP1 \qquad t \in [0,1]$$



5.5 二维线段裁剪

1 矩形窗口线段裁剪

- 直接求交算法
- Cohen-Sutherland算法
- 中点分割裁剪算法
- 梁友栋-Barsky算法

Cohen-Sutherland 算法（1/6）

- 待裁剪线段和窗口的关系

1. 完全落在窗口内
2. 完全落在窗口外
3. 部分在内，部分在外

为提高效率，该算法强调：

- 快速判断情形(1)(2)；
- 减少情形(3)的求交次数和求交所需的计算量。

Cohen-Sutherland 算法（2/6）

算法步骤：

1. 判别线段两端点是否都落在窗口内，如果是，则线段**完全可见**，转至第4步；
2. 判别线段是否为**显然不可见**，如果是，则裁剪结束，转至第4步　；
3. 求线段与窗口边延长线的交点，这个交点将线段分为两段，其中一段显然不可见，丢弃。对余下的另一段重新进行第1步处理，
4. 结束

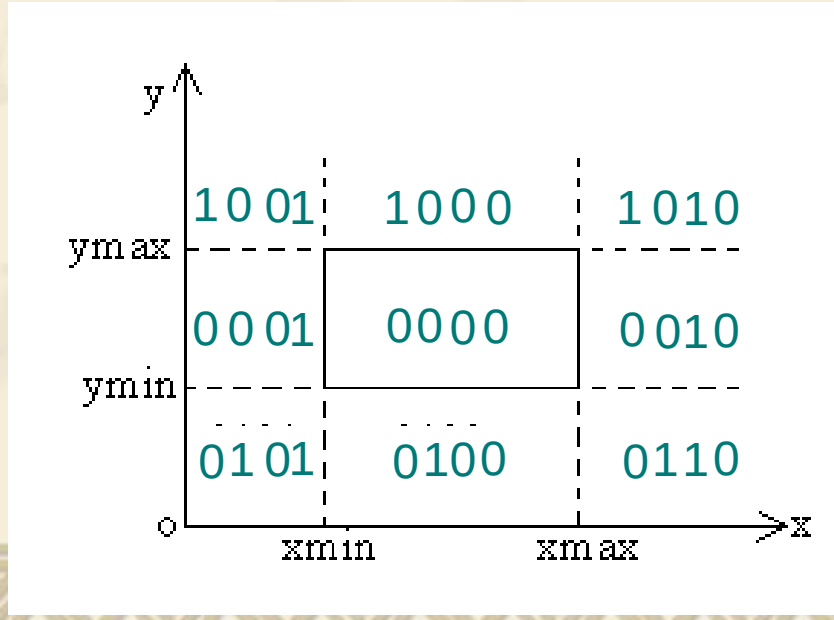
裁剪过程是递归的。

Cohen-Sutherland 算法（3/6）

- 关键问题：
  - 如何快速判别 完全可见 和 显然不可见 线段？
- 解决方法 —— 编码
  - 由窗口四条边所在直线把二维平面分成9个区域，每个区域赋予一个四位编码，$C_tC_bC_rC_l$

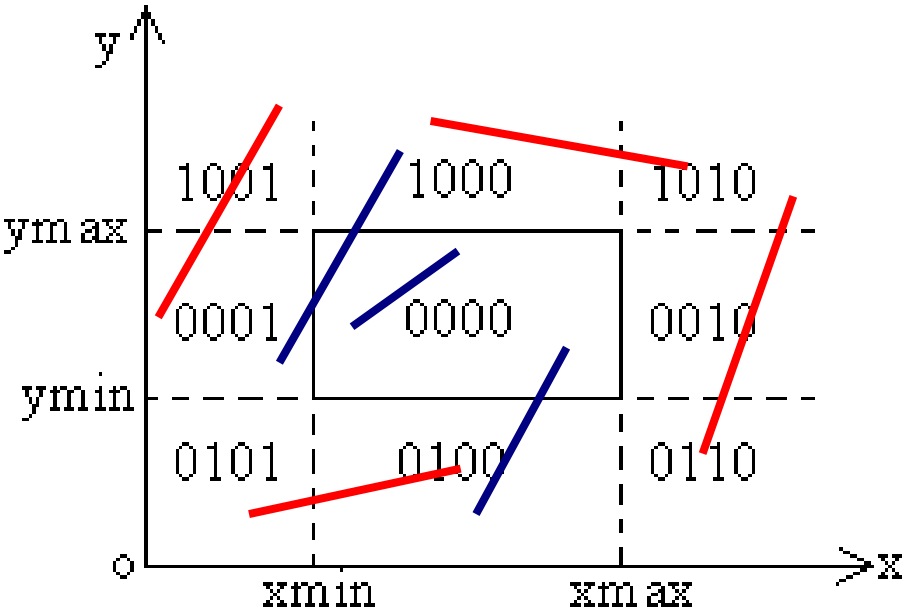
$$C_t=\begin{cases}1, & y>y_{max}\\ 0, & else\end{cases}\qquad C_b=\begin{cases}1, & y<y_{min}\\ 0, & else\end{cases}$$

$$C_r=\begin{cases}1, & x>x_{max}\\ 0, & else\end{cases}\qquad C_l=\begin{cases}1, & x<x_{min}\\ 0, & else\end{cases}$$



Cohen-Sutherland 算法（4/6）

- 所以也称为编码裁剪算法
- 端点编码：
  - 定义为它所在区域的编码
- 快速判断“显然不可见”
  - 线段两端点编码的按位与运算结果非零



Cohen-Sutherland 算法（5/6）

对于部分可见又部分不可见的线段，需要求交，
求交前先测试与窗口哪条边所在直线有交？

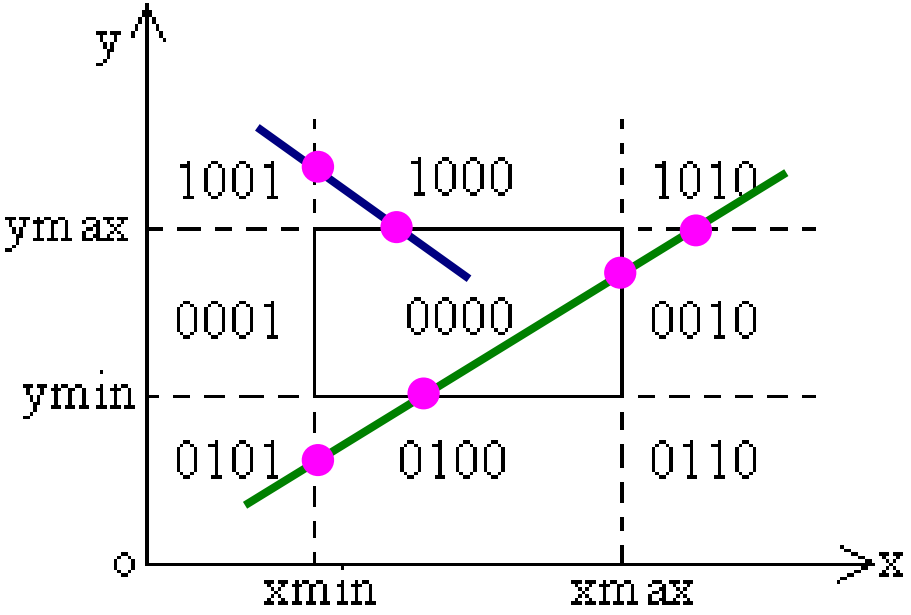
- 逐个端点判断其编码

  $C_lC_tC_rC_b$中各位是否为1，

  若是，则求交。

- 最坏情形：

  线段求交四次。



Cohen-Sutherland 算法（6/6）

1）特点：

用编码方法可快速判断线段———

完全可见 或 显然不可见。

2）特别适用两种场合：

① 大窗口场合；

② 窗口特别小的场合

（如：光标拾取图形时，光标看作小的裁剪窗口）

矩形窗口裁剪

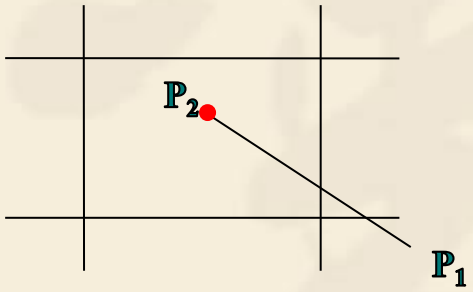
5.5 二维线段裁剪

1 矩形窗口线段裁剪

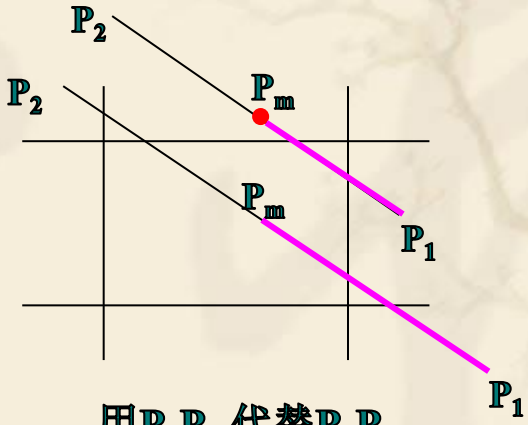
- 直接求交算法
- Cohen-Sutherland算法
- 中点分割裁剪算法
- 梁友栋-Barsky算法

中点分割裁剪算法（1/2）

- 基本思想：利用对分搜索思想，折半搜索交点
  - 不断地在中点处将线段一分为二，取中点Pm=(P1+P2)/2
  - 对每段线段重复Cohen-Sutherland算法的线段可见性测试
  - 直至找到每段线段与窗口边界线的交点
  - 或分割子段的长度充分小可视为一点为止



P2在窗口内，则是距P1最远的可见点



用P_1P_m代替P_1P_2

用P_mP_2代替P_1P_2

中点分割裁剪算法（2/2）

- 优点：
  - 算法原理和编码裁剪是一致的，不同之处在于用移位运算代替求交计算
  - 适合硬件实现

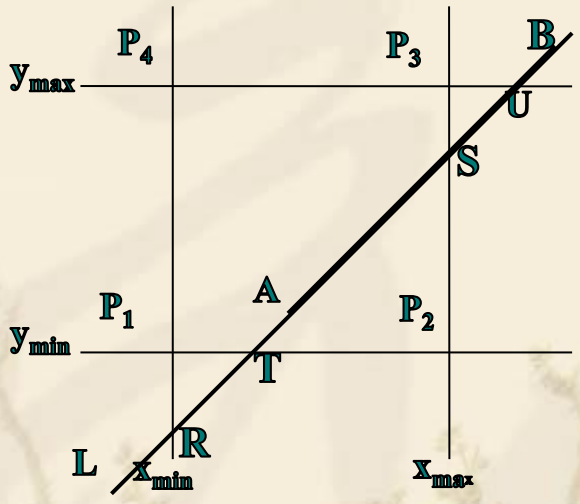
5.5 二维线段裁剪

1 矩形窗口线段裁剪

- 直接求交算法
- Cohen-Sutherland算法
- 中点分割裁剪算法
- 梁友栋-Barsky算法

Liang-Barsky裁剪算法（1/5）

- 基本思想：
  - 把二维裁剪转为一维裁剪问题，并向x（或y）方向投影以决定可见线段



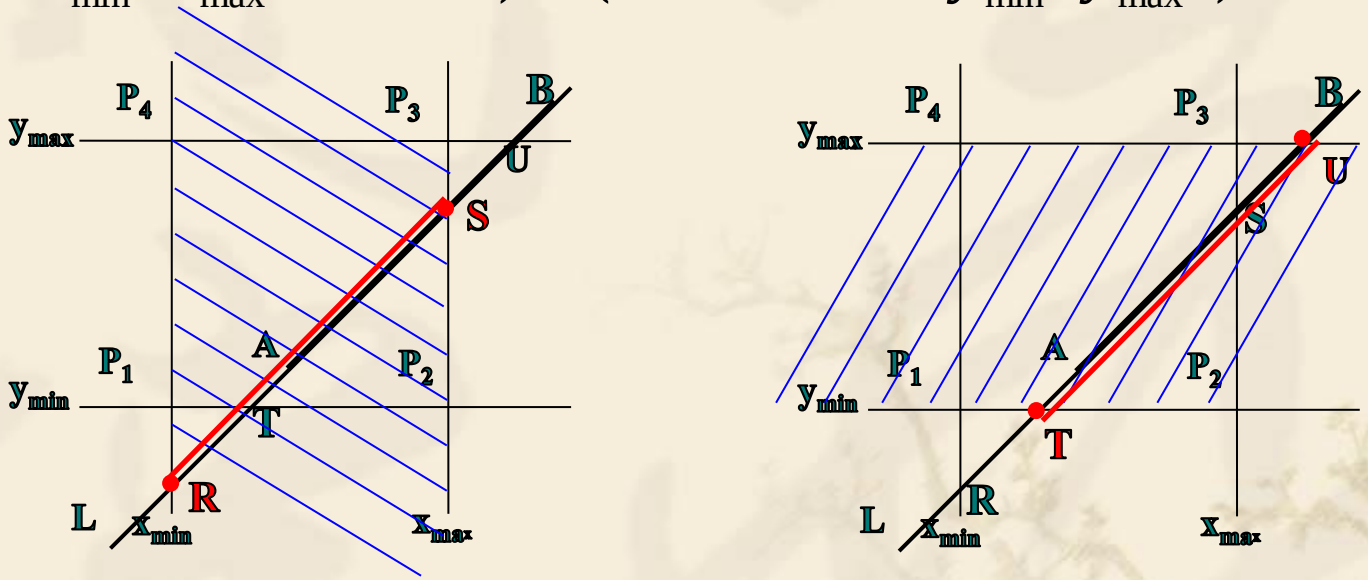
AS是一维窗口TS中的可见部分

Liang-Barsky裁剪算法（2/5）

- 直线L与区域的交：

$$Q = L \cap \square P_1P_2P_3P_4 = L \cap [x_{\min}, x_{\max}; -\infty, +\infty] \cap [-\infty, +\infty; y_{\min}, y_{\max}]$$

$$= (L \cap [x_{\min}, x_{\max}; -\infty, +\infty]) \cap (L \cap [-\infty, +\infty; y_{\min}, y_{\max}]) = RS \cap TU$$

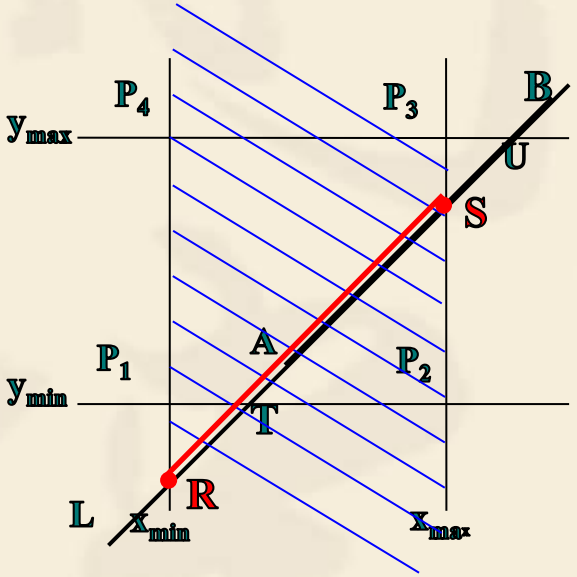


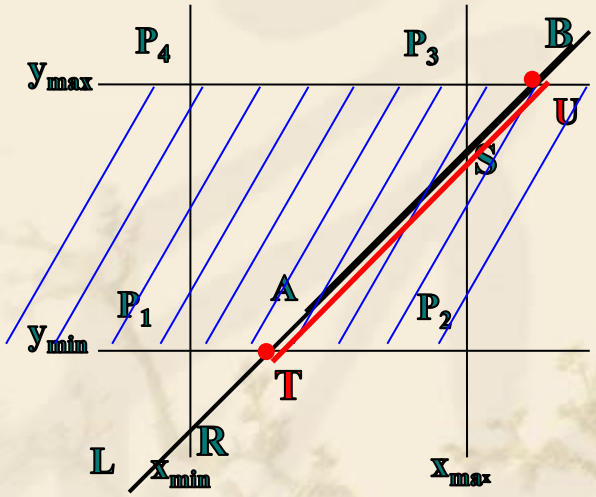
- 当Q为空集时，线段AB不可能在窗口中有可见线段。
- 当Q不为空集时，Q可看成是一个一维窗口

Liang-Barsky裁剪算法（3/5）

- 存在可见线段的充要条件
  - $AB \cap Q$ 即 $AB \cap RS \cap TU$ 不为空集 。

$$Q = RS \cap TU$$

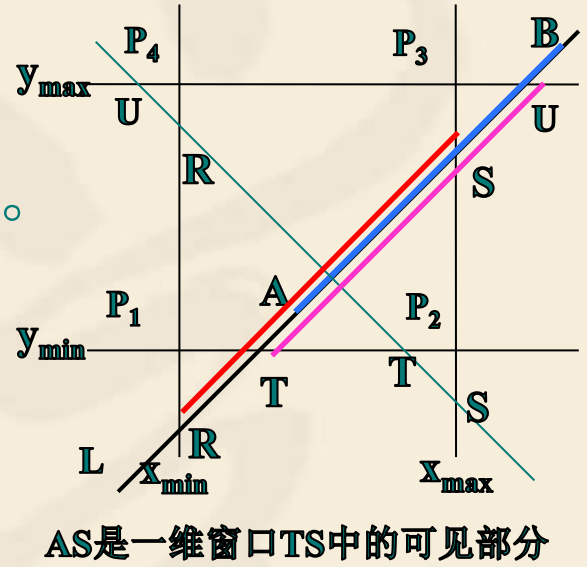




- 当Q为空集时，线段AB不可能在窗口中有可见线段。
- 当Q不为空集时，Q可看成是一个一维窗口

Liang-Barsky裁剪算法（4/5）

- 存在可见线段的充要条件
  - $AB \cap Q$ 即 $AB \cap RS \cap TU$ 不为空集。



AS是一维窗口TS中的可见部分

- *RS*,*AB*, *TU*三条线段的**交集**的端点坐标
- 等价于求三条线段的
  左端点的最大值，右端点的最小值
- 向x轴投影，得到可见线段端点的 x 坐标变化范围

$$\max[x_{\min}, \min(x_A, x_B), \min(x_T, x_U)] \le x \le \min[x_{\max}, \max(x_A, x_B), \max(x_T, x_U)]$$

左端点x坐标 $x_\alpha = \max[x_{\min}, \min(x_A, x_B), \min(x_T, x_U)]$

右端点x坐标 $x_\beta = \min[x_{\max}, \max(x_A, x_B), \max(x_T, x_U)]$

y坐标可由将x坐标代入直线方程计算得到

Liang-Barsky裁剪算法（5/5）

$$\max[x_{\min}, \min(x_A, x_B), \min(x_T, x_U)] \le x \le \min[x_{\max}, \max(x_A, x_B), \max(x_T, x_U)]$$

- AB有可见部分的充要条件也可表示为

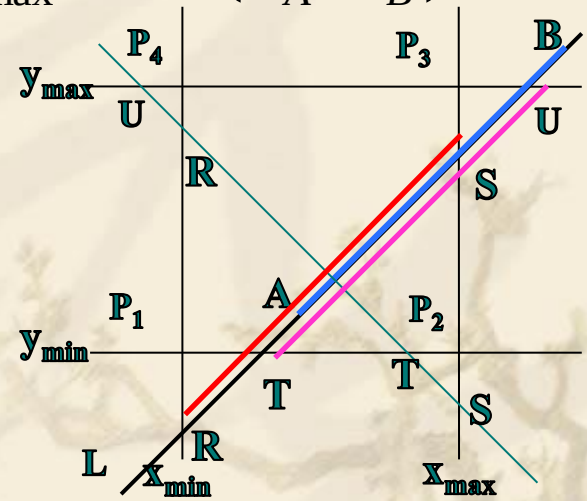
$$\max[x_{\min}, \min(x_A, x_B), \min(x_T, x_U)] \le \min[x_{\max}, \max(x_A, x_B), \max(x_T, x_U)]$$

令 $L = \max[x_{\min}, \min(x_A, x_B)]$　　$R = \min[x_{\max}, \max(x_A, x_B)]$

则有：$\max[L, \min(x_T, x_U)] \le \min[R, \max(x_T, x_U)]$

等价于判断
$$\begin{cases} L \le R \\ L \le \max(x_T, x_U) \\ \min(x_T, x_U) \le R \end{cases}$$

裁剪



AS是一维窗口TS中的可见部分

本章内容

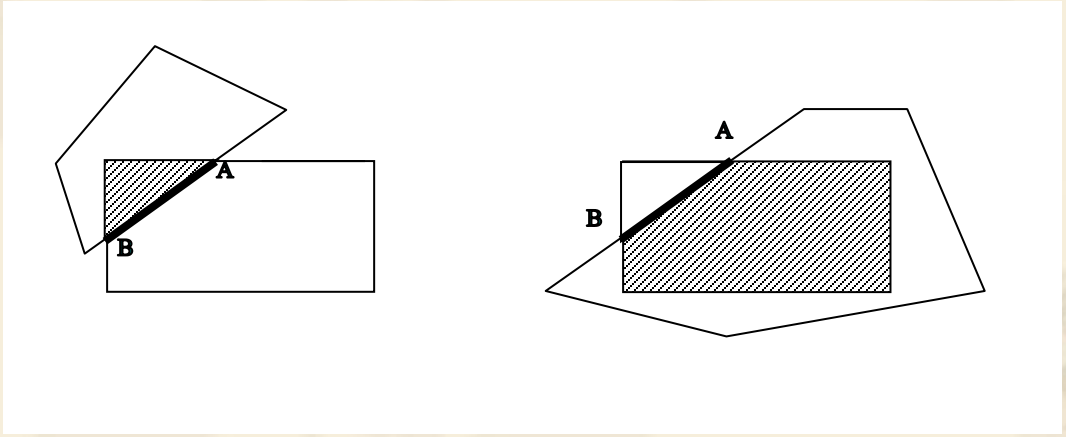
1 窗口视图变换
2 二维图形几何变换
3 三维图形几何变换
4 投影变换
5 二维线段裁剪
6 多边形的裁剪

本章内容

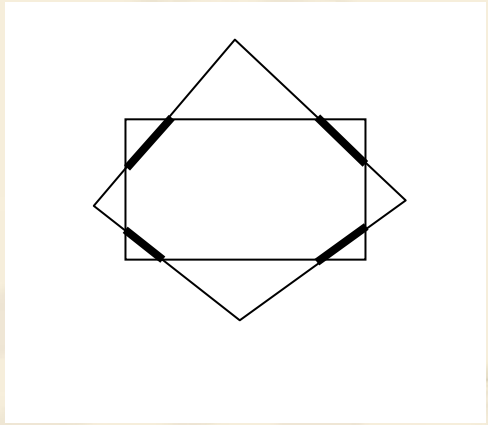
1 窗口视图变换
2 二维图形几何变换
3 三维图形几何变换
4 投影变换
5 二维线段裁剪
6 多边形的裁剪

5.6 多边形裁剪

- 错觉：多边形裁剪是直线段裁剪的组合？
- 新的问题：



因丢失顶点信息而去法确定裁剪区域

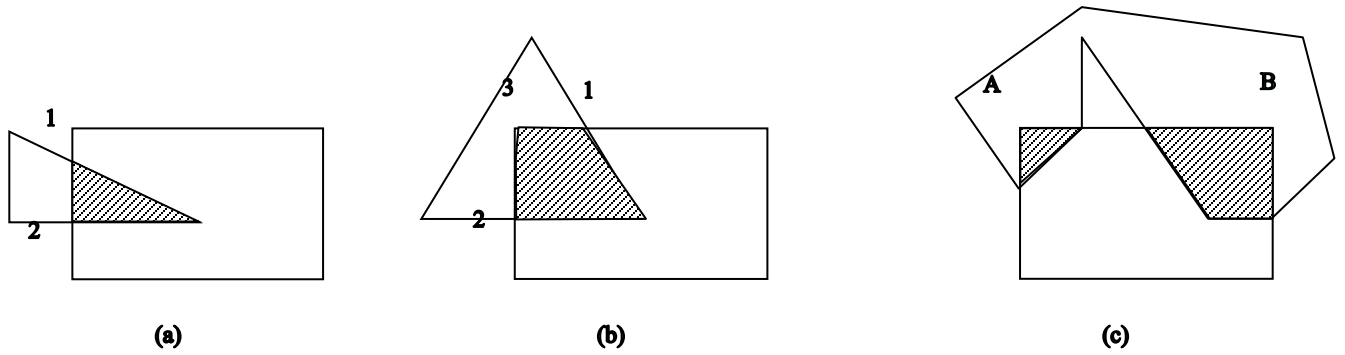


原来封闭的多边形变成了孤立的线段

边界不再封闭，需要用窗口边界的恰当部分来封闭它

5.6 多边形裁剪

- 新的问题：
- 裁剪结果有可能分裂为几个多边形



裁剪后的多边形顶点形成的几种情况

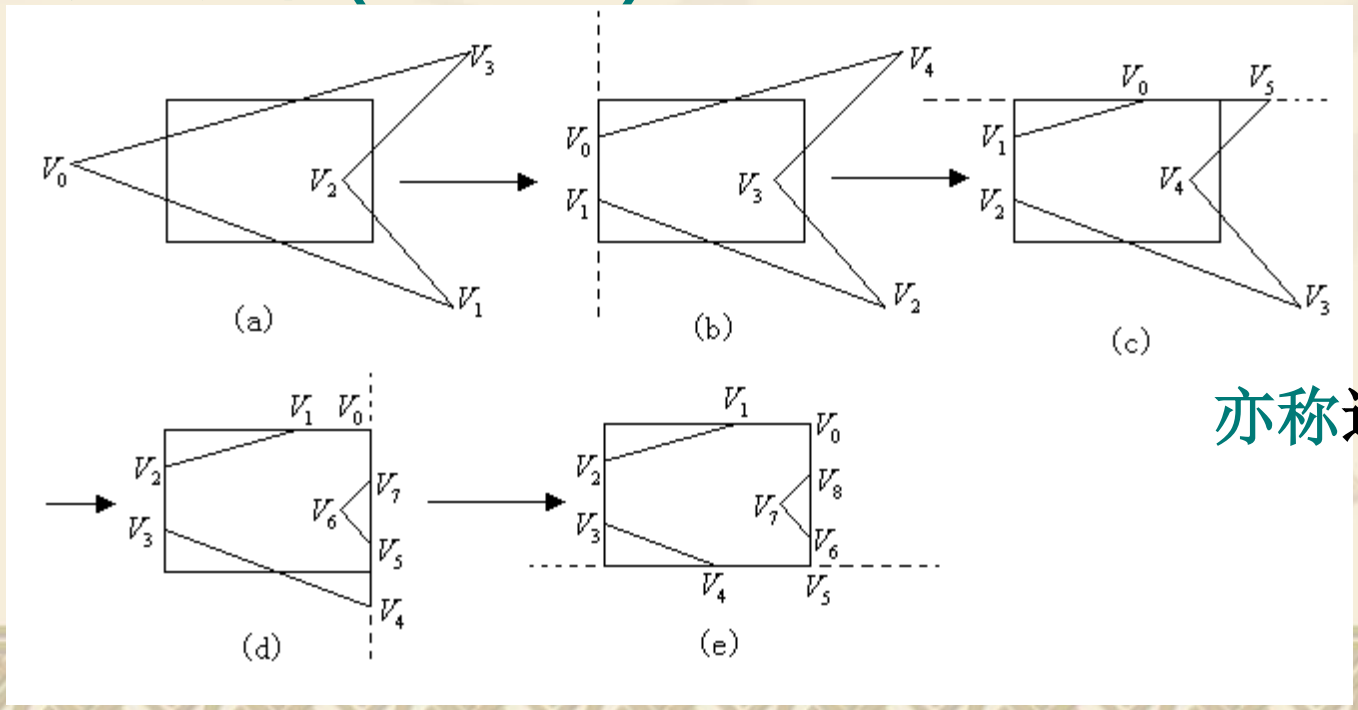
- 关键：
  - 不仅在于求出新的顶点，删去界外顶点
  - 还在于形成正确的顶点序列

5.6 多边形裁剪

- 常用算法
  - Sutherland_Hodgman算法
  - Weiler-Atherton算法

Sutherland-Hodgman算法（1/3）

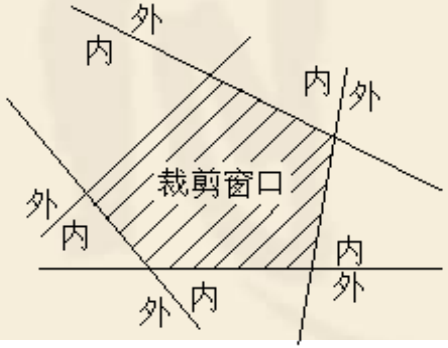
- 分割处理策略：
  - 将多边形关于矩形窗口的裁剪分解为多边形关于窗口四边所在直线的裁剪。
- 流水线过程(左上右下)：左边的结果是右边的开始。



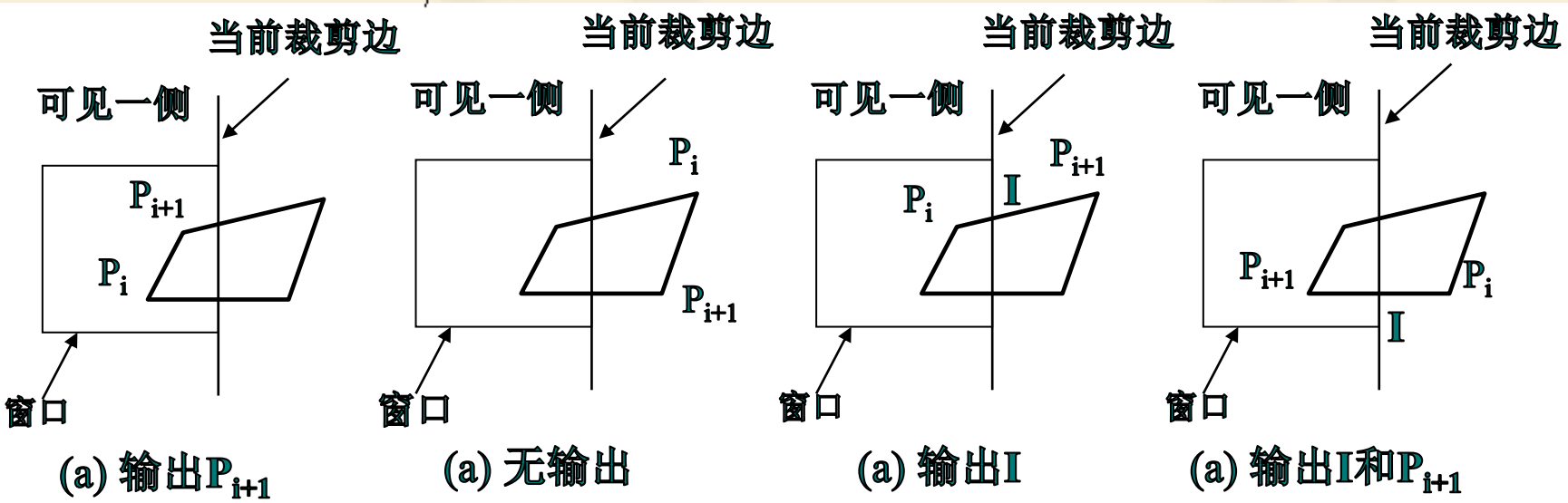
亦称逐边裁剪算法

Sutherland-Hodgman算法（2/3）

- 多边形的边与半空间的关系



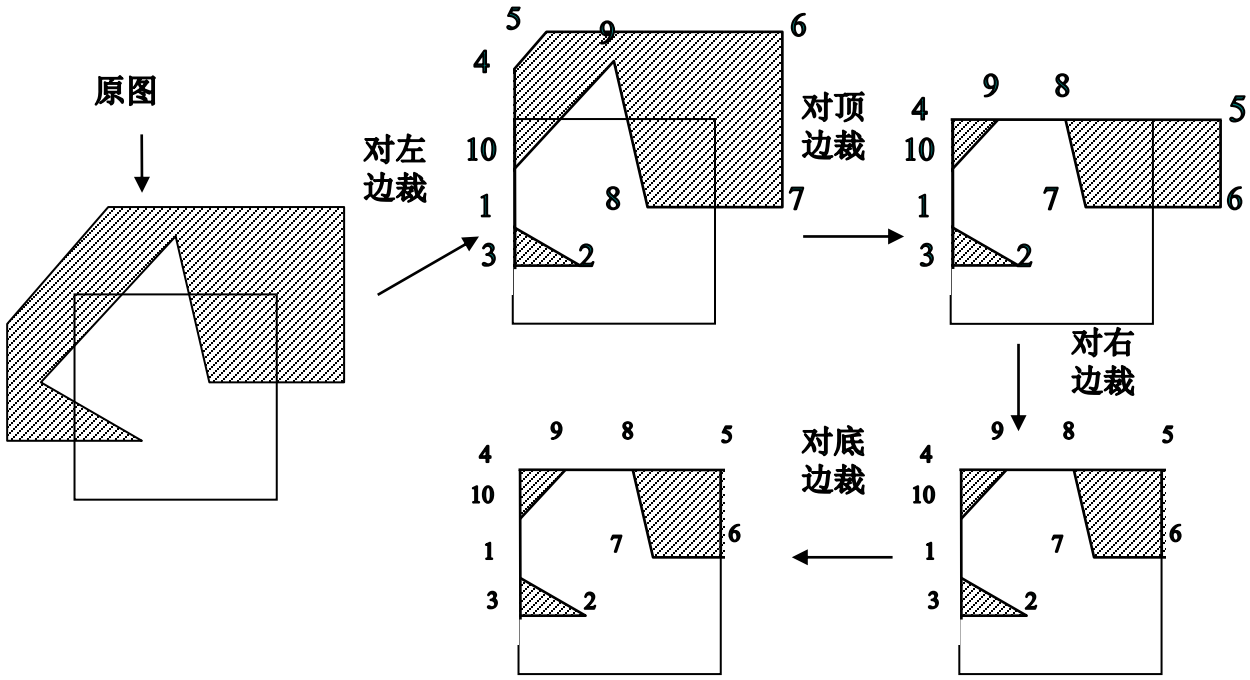
- 裁剪结果的顶点构成：
  - 裁剪边内侧的原顶点
  - 多边形的边与裁剪边的交点
  - 顶点顺序连接



线段与当前裁剪边的位置关系

Sutherland-Hodgman算法（3/3）

- 可推广到任意凸多边形裁剪窗口，那么凹多边形窗口怎么办？
- 逐边裁剪法对凹多边形裁剪时，裁剪后分裂为几个多边形，这几个多边形沿边框产生多余的线段？



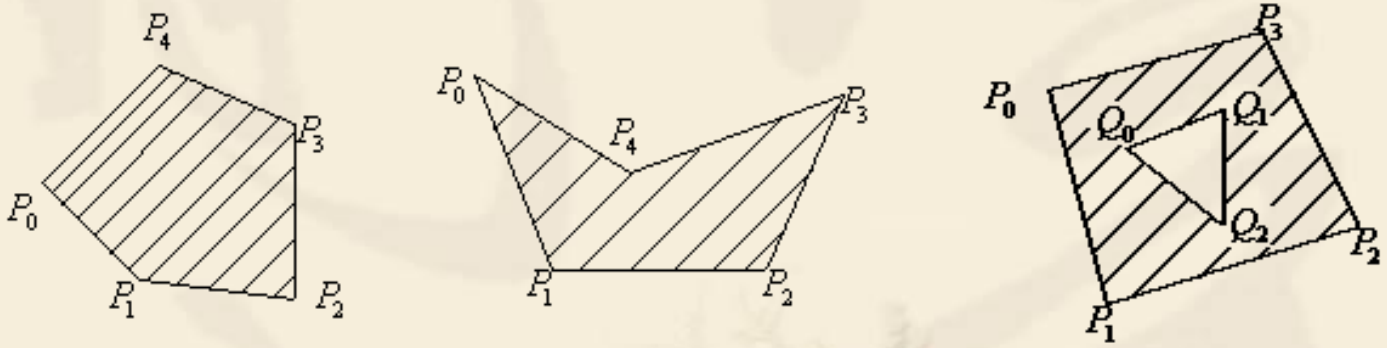
逐边裁剪法对凹多边形裁剪时可能出现的问题

5.6 多边形裁剪

- 常用算法
  - Sutherland_Hodgman算法
  - Weiler-Atherton算法

Weiler-Atherton算法（1/7）

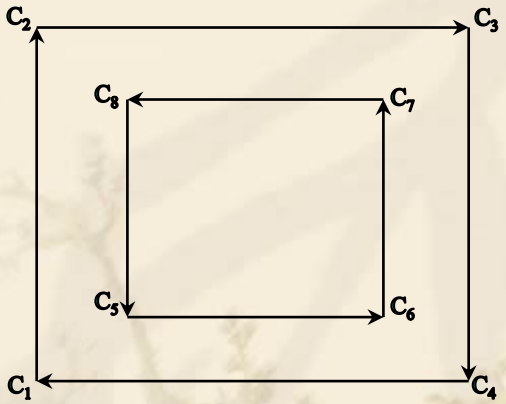
裁剪窗口为任意多边形（凸、凹、带内环）的情况：



- 主多边形：被裁剪多边形，记为SP
- 裁剪多边形：裁剪窗口，记为CP

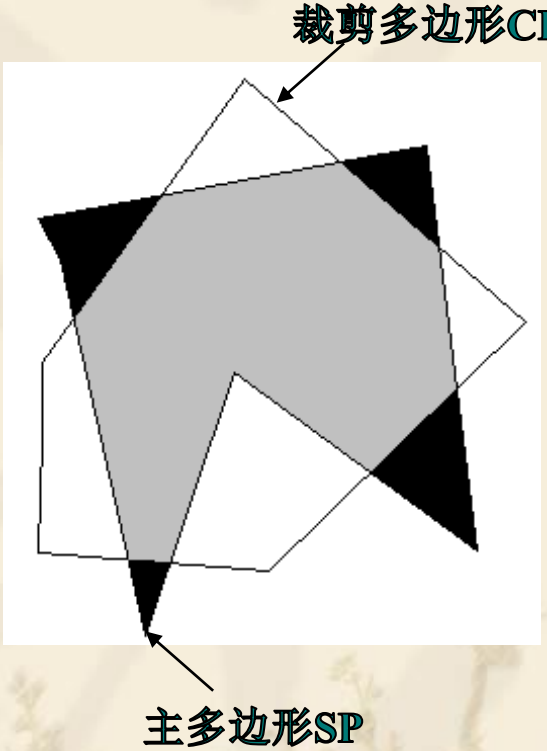
Weiler-Atherton算法（2/7）

- 约定：
  - SP与CP均用它们顶点的环形链表定义
  - 外边界取顺时针方向
  - 内边界取逆时针方向
  - 使得沿多边形的边走动，其右边为多边形的内部。



Weiler-Atherton算法（3/7）

- SP和CP把二维平面分成两部分。
- 内裁剪：SP∩CP
- 外裁剪：SP-CP



裁剪结果区域的边界由两部分构成：

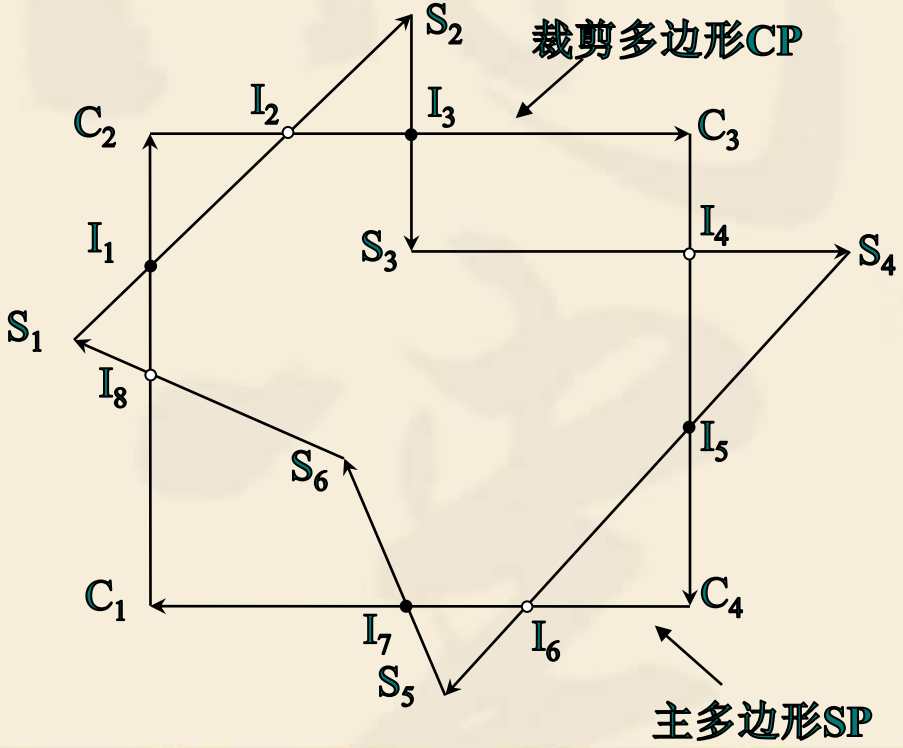
1. SP的部分边界
2. CP的部分边界

且在交点处，边界发生交替

即由SP边界转至CP边界，或由CP边界转至SP边界

Weiler-Atherton算法（4/7）

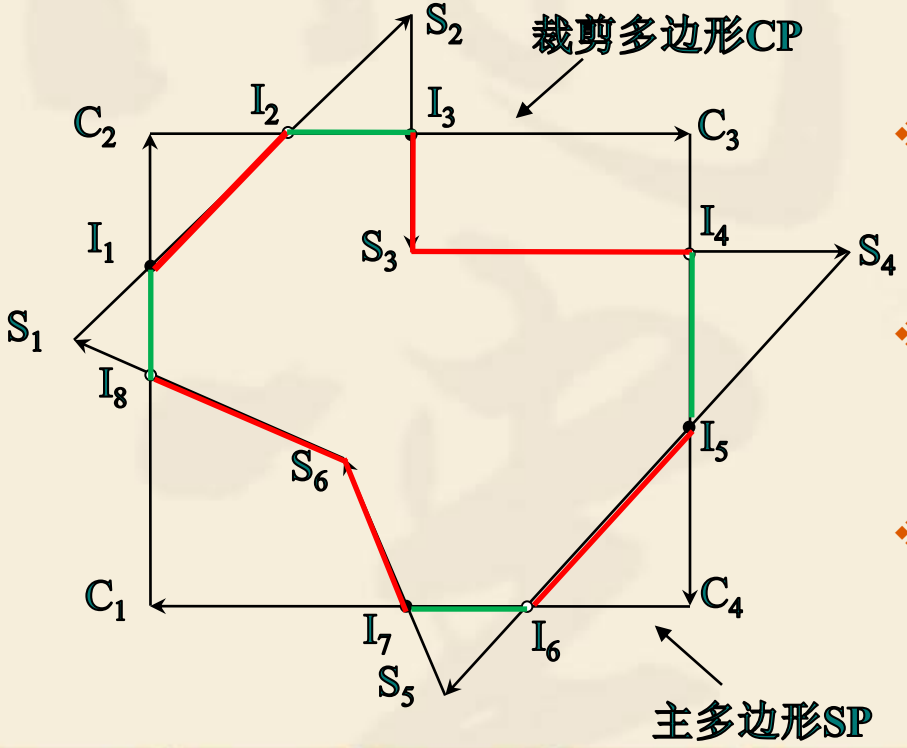
- 如果SP与CP有交点，则*交点成对出现，*它们被分为如下两类：



- ***进点***：SP边界由此进入CP
  如，I1,I3, I5, I7
- ***出点***：SP边界由此离开CP
  如，I2, I4, I6, I8

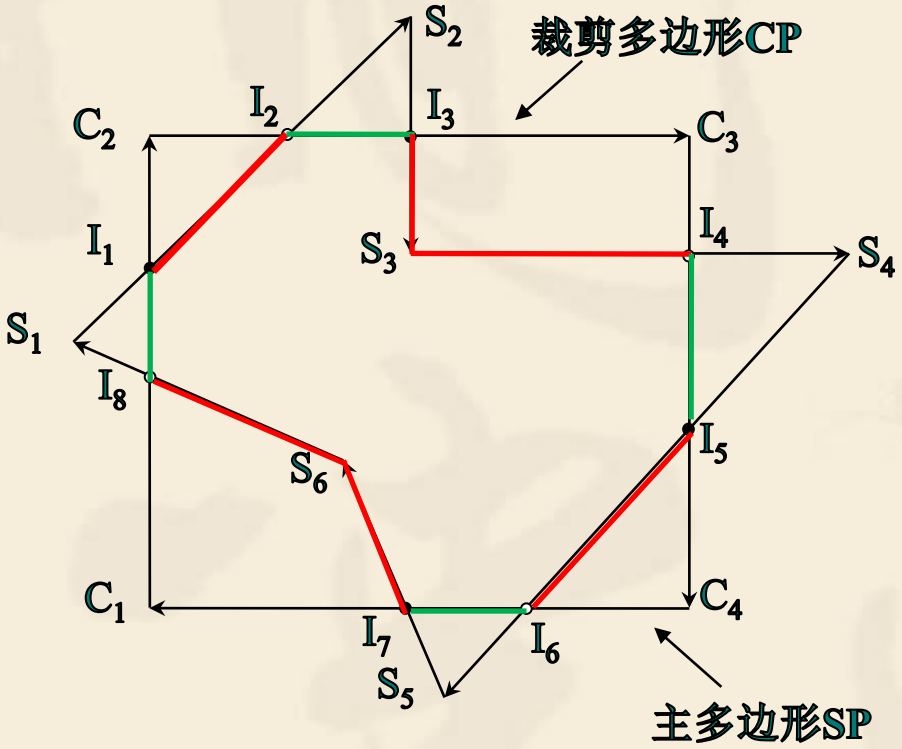
Weiler-Atherton算法（5/7）

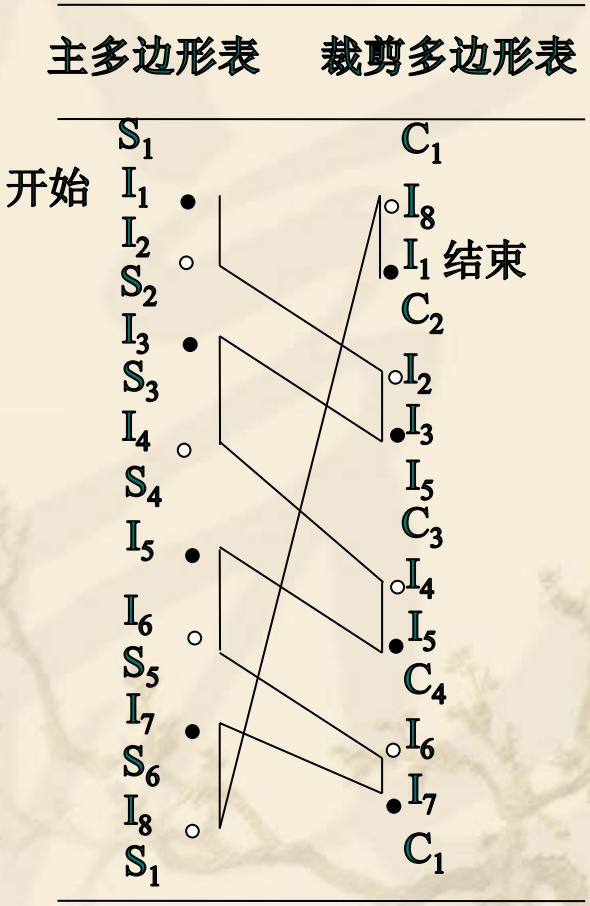
- 由任一个进点出发，沿SP的边，跟踪检测其与CP的交点（前交点），并判断该交点是进点还是出点。如此交替沿两个多边形的边行进。直至回到跟踪的起始点为止。



- 若是*进点*：沿SP边所示方向收集顶点序列。
- 若是*出点*：从此点开始，检测CP的边所示方向收集顶点序列。
- 如此交替沿两个多边形的边行进，直至回到跟踪起点为止。

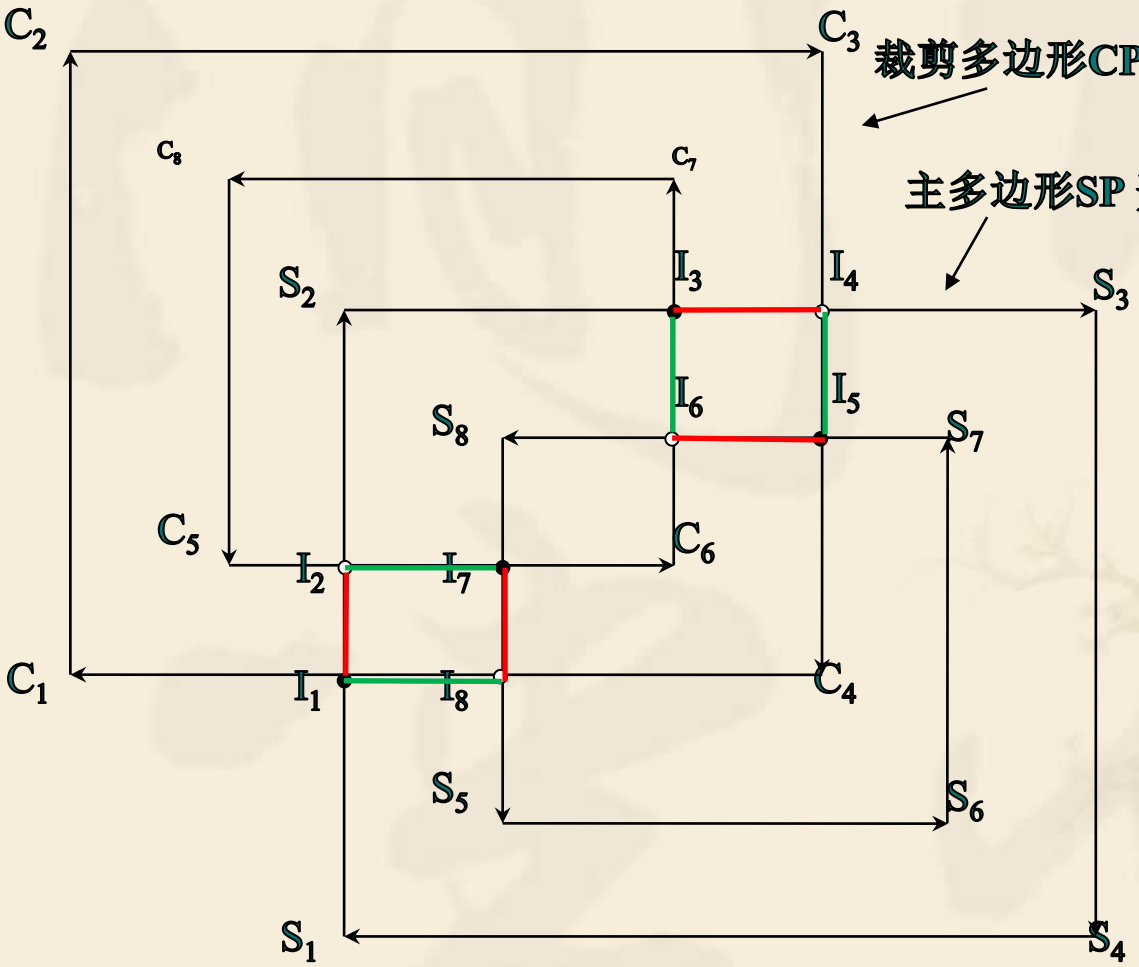
Weiler-Atherton算法（6/7）





算法裁剪后所生成的多边形为$I_1I_2I_3S_3I_4I_5I_6I_7S_6I_8\ I_1$

Weiler-Atherton算法（7/7）



主多边形表	裁剪多边形表
S_1	C_1
I_1	C_2
I_2	C_3
S_2	I_4
I_3	I_5
I_4	C_4
S_3	I_8
S_4	I_1 结束
S_1	C_1
S_5	C_5
S_6	I_2
S_7	I_7
I_5	C_6
I_6	I_6
S_8	I_3 结束
I_7	C_7
I_8	C_8
S_5	C_5

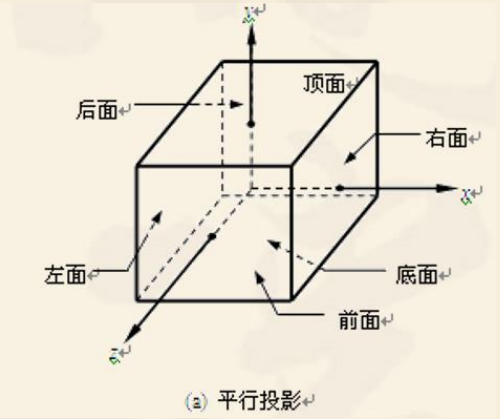
算法裁剪后所生成的多边形为$I_1I_2I_7I_8I_1$和$I_3I_4I_5I_6I_3$

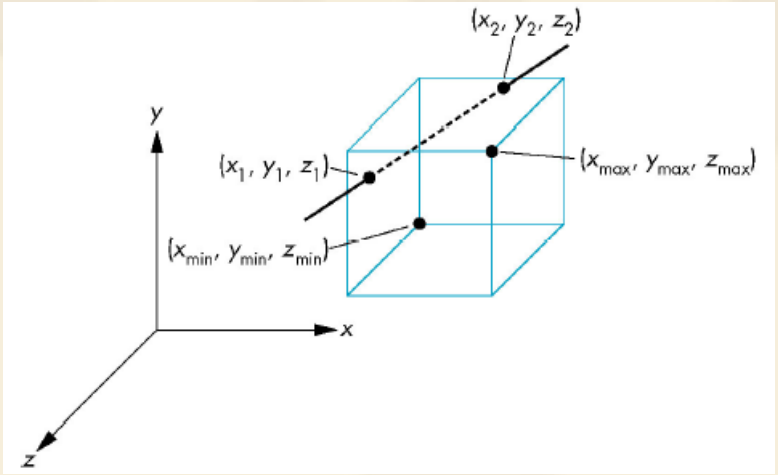
三维裁剪

- 三维裁剪
  - 裁剪对象：线裁剪、面裁剪
  - 裁剪窗口：
    - 平行投影：规范的立方体
    - 透视投影：视域四棱锥
- 裁剪算法
  - Sutherland-Cohen、梁友栋-Basky等都可以推广到三维情形

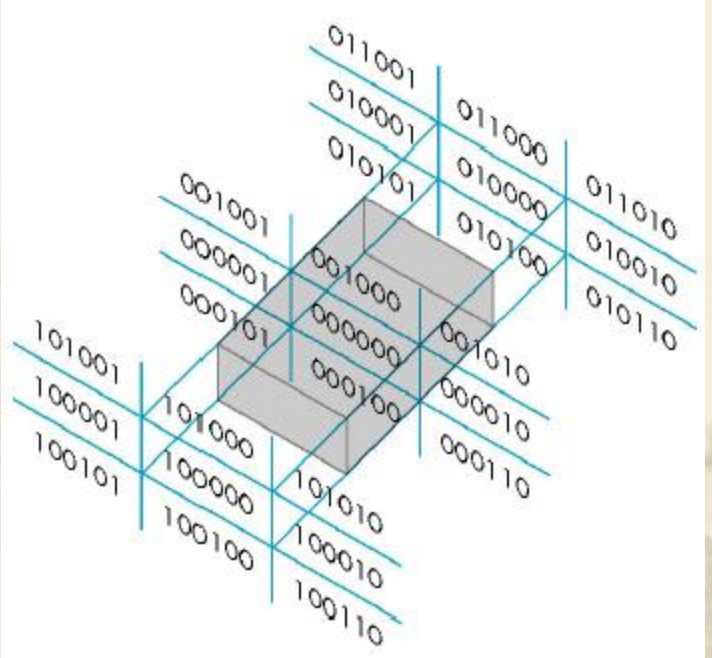
5.7 三维线段裁剪

- 平行投影中的三维裁剪
  - 三维空间中的Cohen-Sutherland 算法
    - 4位编码扩展为6位编码

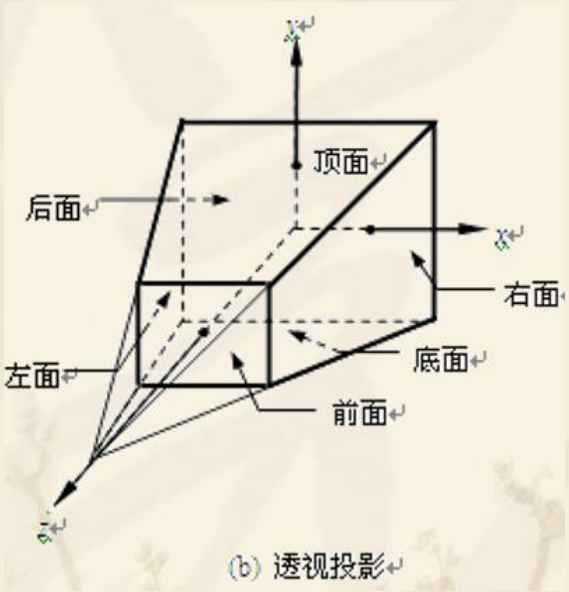




当线段端点位于裁剪体的左侧时，即 $x<-1$ 时，第 0 位置为 1，否则置为 0。
当线段端点位于裁剪体的右侧时，即 $x>1$ 时，第 1 位置为 1，否则置为 0。
当线段端点位于裁剪体的下方时，即 $y<-1$ 时，第 2 位置为 1，否则置为 0。
当线段端点位于裁剪体的上方时，即 $y>1$ 时，第 3 位置为 1，否则置为 0。
当线段端点位于裁剪体的前方时，即 $z>1$ 时，第 4 位置为 1，否则置为 0。
当线段端点位于裁剪体的后方时，即 $z<-1$ 时，第 5 位置为 1，否则置为 0。



5.7 三维线段裁剪



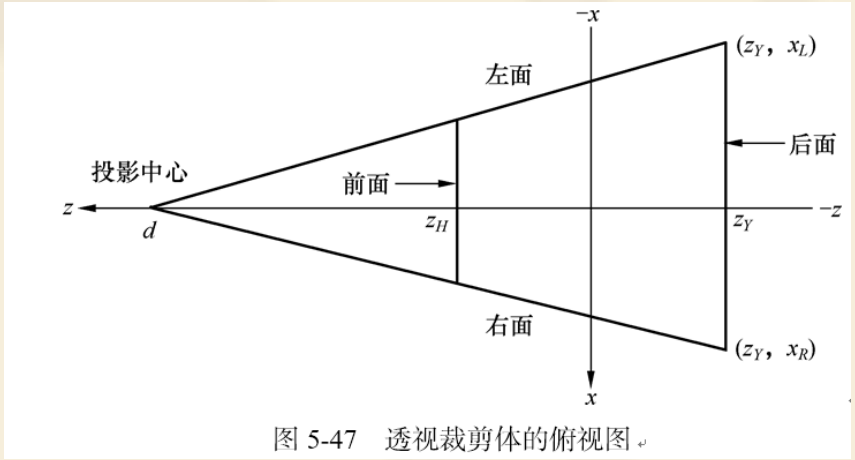
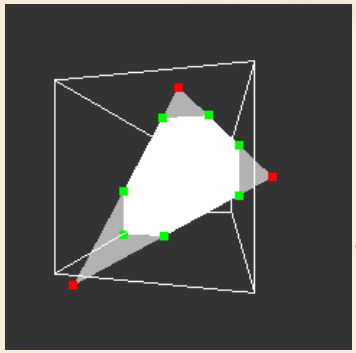
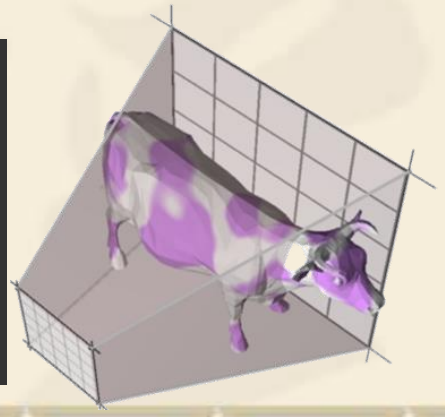


图 5-47　透视裁剪体的俯视图

以右侧面方程为例　$x = \dfrac{x_R}{z_Y - d}(z - d) = a_1 z + a_2$

$$x - a_1 z - a_2 = 0$$





先将点 $P(x, y, z)$ 的坐标代入右侧面方程式的左侧，可得判别函数如下：

若 $f_R = x - a_1 z - a_2 > 0$，表明点 P 位于该平面的右方。

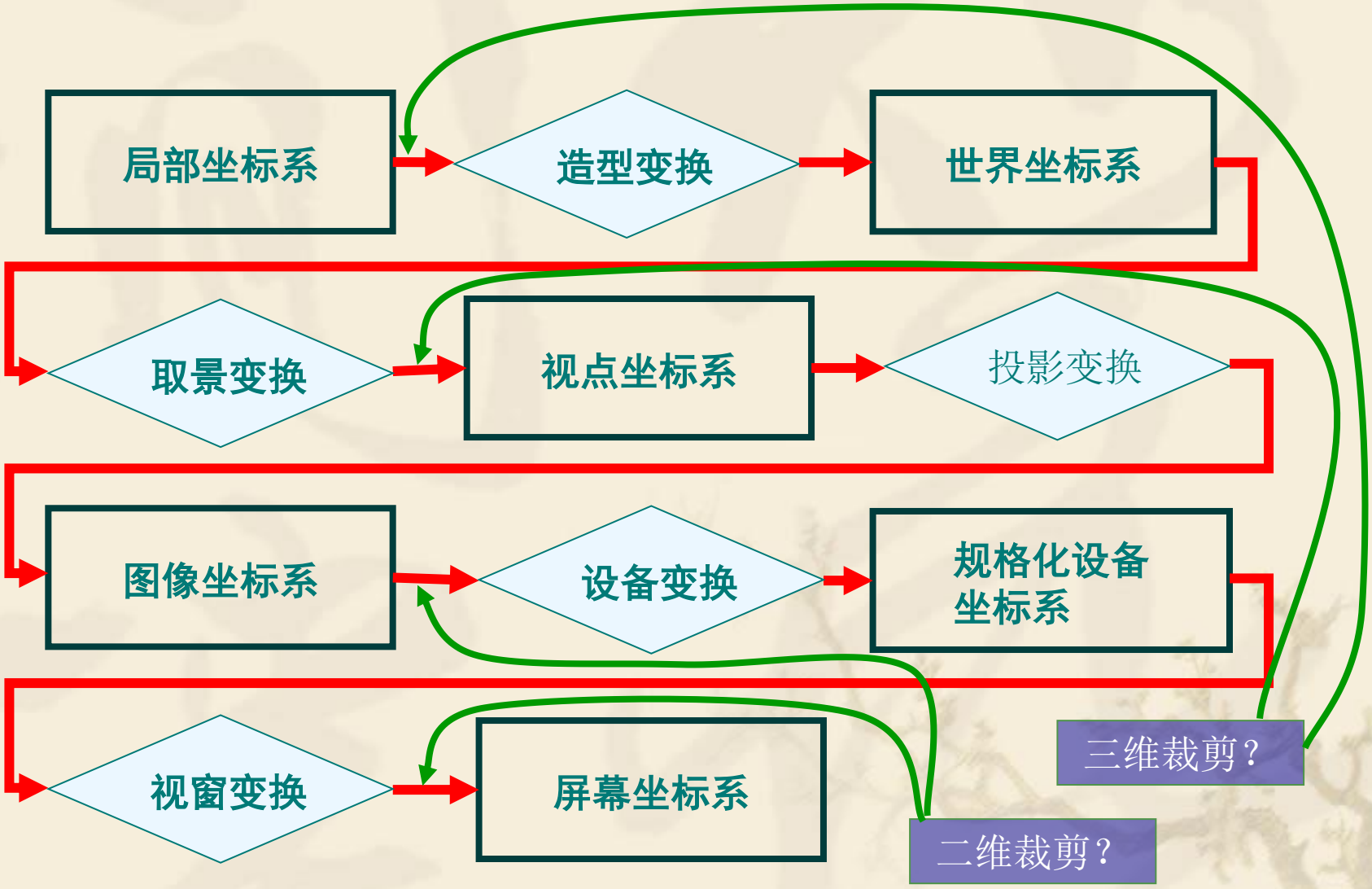
若 $f_R = x - a_1 z - a_2 = 0$，表明点 P 位于该平面上。

若 $f_R = x - a_1 z - a_2 < 0$，表明点 P 位于该平面的左方。

三维裁剪

- 何时裁剪？
  - 投影之前裁剪——三维裁剪
    - 优点：只对裁剪后可见的物体投影，提高了消隐效率
    - 缺点：三维裁剪相对复杂
  - 投影之后裁剪——二维裁剪
    - 优点：二维裁剪相对容易
    - 缺点：需要对所有的物体进行投影变换

三维变换流程图


局部坐标系
造型变换
世界坐标系
取景变换
视点坐标系
投影变换
图像坐标系
设备变换
规格化设备
坐标系
视窗变换
屏幕坐标系
三维裁剪？
二维裁剪？


Thank you for
your attention !
SuXiaoHong


Question?