二维裁剪

刘世光
天津大学计算机学院

主要内容

- 二维点裁剪
- 二维线裁剪

  （Cohen-Sutherland裁剪算法；Nicholl-Lee-Nicholl裁剪算法；梁友栋-Barsky裁剪算法）
- 非矩形裁剪窗口的线段裁剪

裁剪的定义

- 一般情况下，任何用来消除指定区域内或区域外的图形部分的过程称为裁剪算法（**Clipping Algorithm**），简称裁剪。

- 裁剪区域一般是一个正则的矩形，其边界位于**XWmin, XWmax, YWmin, YWmax**。

二维点裁剪

- 如果点P(x,y)满足下列不等式，则保存该点用于显示：

$$XWmin <= X <= XWmax;$$

$$YWmin <= Y <= YWmax$$

如果上述四个不等式有任何一个不满足，则裁剪掉该点（将不会存储或显示）。

- 应用：用于包含云、海面泡沫、烟或者爆炸等用小圆或小球这样的粒子进行建模的场景。

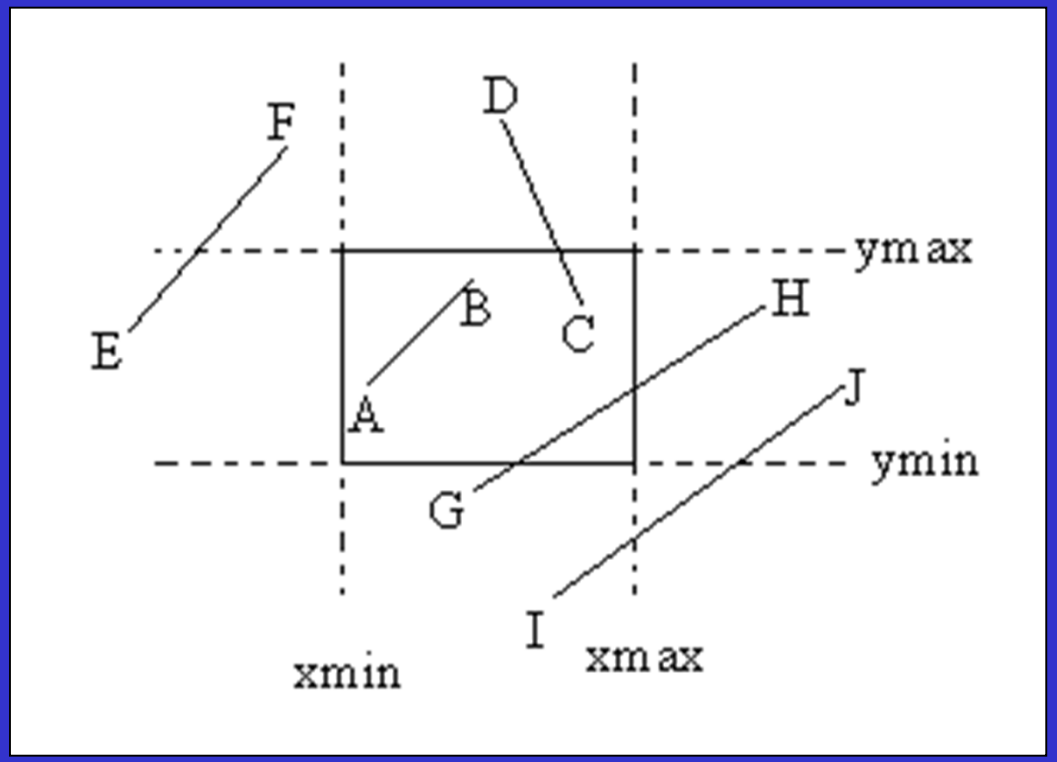
二维线裁剪

- 裁剪的目的
  - 判断线段是否落在裁剪窗口之内并找出其位于内部的部分
- 裁剪的处理的基础
  - 图元与窗口的求交（耗时）

- 假定条件
  - 矩形裁剪窗口：**[xmin,xmax]X[ymin,ymax]**
  - 待裁剪线段： $P_0(x_0, y_0)P_1(x_1, y_1)$

二维线裁剪

- 待裁剪线段和窗口的关系
  - 线段完全可见
  - 显然不可见
  - 线段至少有一端点在窗口之外，但非显然不可见



二维线裁剪

- 直接求交算法

- 简单，但是效率不高。是否可以重新组织初始测试和求交计算来减少一组线段的处理时间？

Cohen-Sutherland 算法

- 最早开发的快速线段裁剪算法，得到了广泛应用。通过初始测试来减少求交计算，从而提高效率。

- 特点：对显然不可见线段的快速判别，可以直接推广到三维

Cohen-Sutherland 算法

– 编码方法

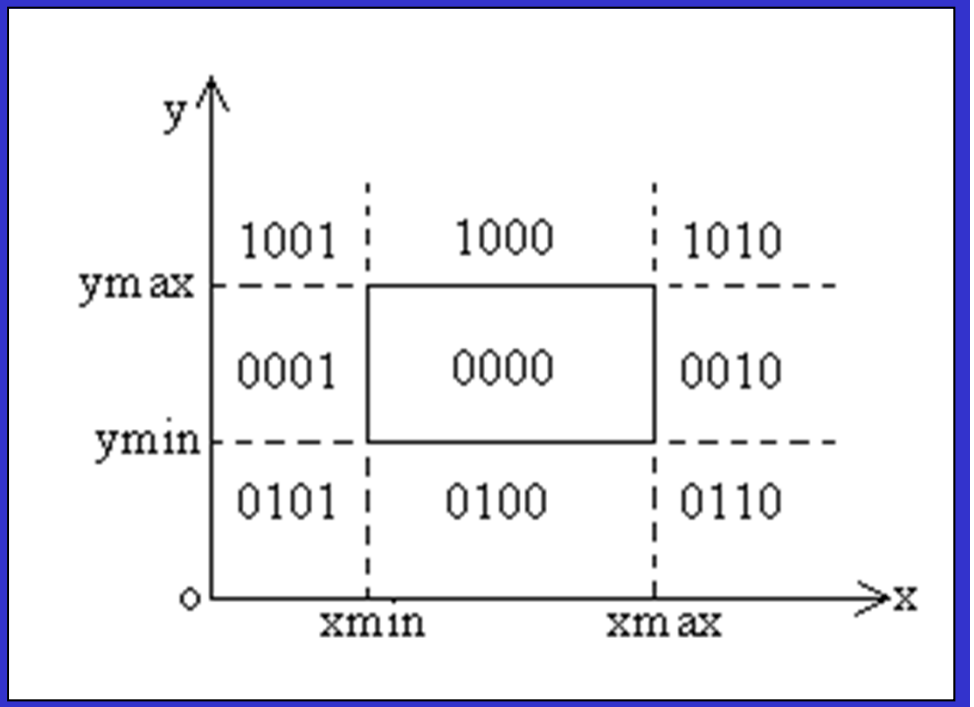
- 区域编码：用窗口四边所在的直线将整个平面分成9个区域，每个区域赋予一个四位的编码 $C_tC_bC_rC_l$

$$C_t = \begin{cases} 1 & \text{当} y > y\max \\ 0 & else \end{cases}$$

$$C_b = \begin{cases} 1 & \text{当} y < y\min \\ 0 & else \end{cases}$$

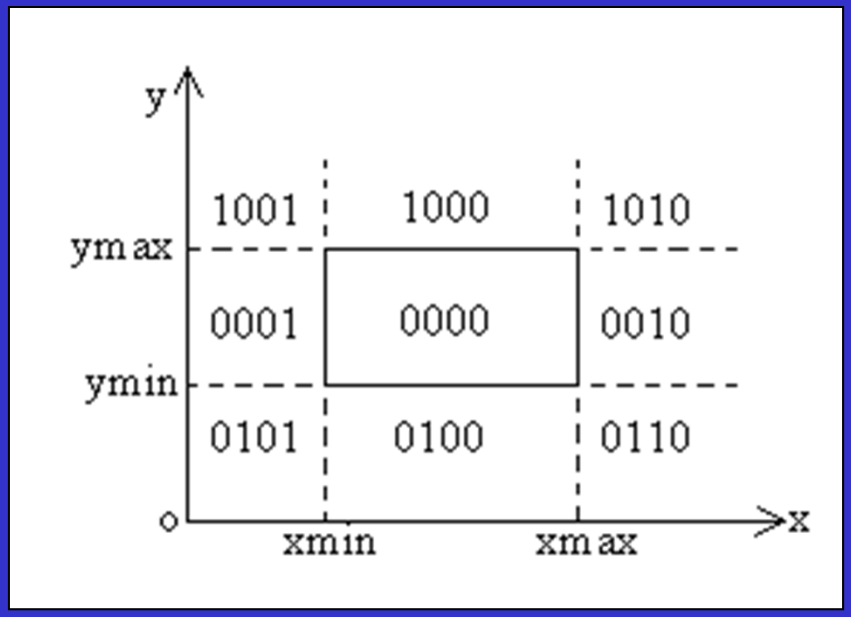
$$C_r = \begin{cases} 1 & \text{当} x > x\max \\ 0 & else \end{cases}$$

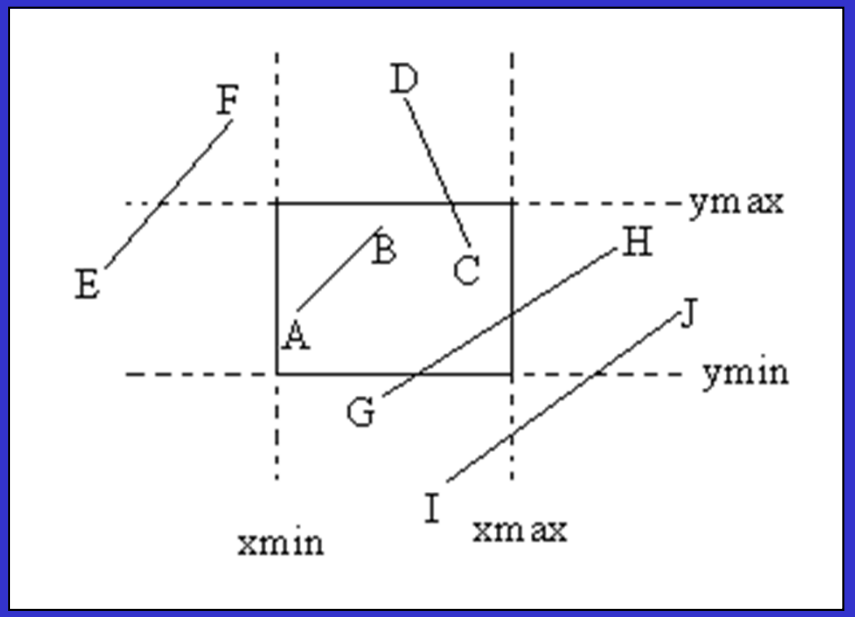
$$C_l = \begin{cases} 1 & \text{当} x < x\min \\ 0 & else \end{cases}$$



Cohen-Sutherland 算法

- 端点编码：定义为它所在区域的编码
- 结论：当线段的两个端点的编码的逻辑“与”非零时，线段为显然不可见的





Cohen-Sutherland 算法

- 对于不能判断完全在窗口外或窗口内的线段，则要测试其与窗口的交点。
- 方法：线段与裁剪边界逐个求交

- 线段与裁剪边界的交点计算可使用斜率截矩式方程：

$$y = y_0 + m(x - x_0)$$

$$x = x_0 + \frac{y - y_0}{m} \qquad x = xw_{\min}; xw_{\max}$$ 垂直边界

Cohen-Sutherland 算法

- 步骤：

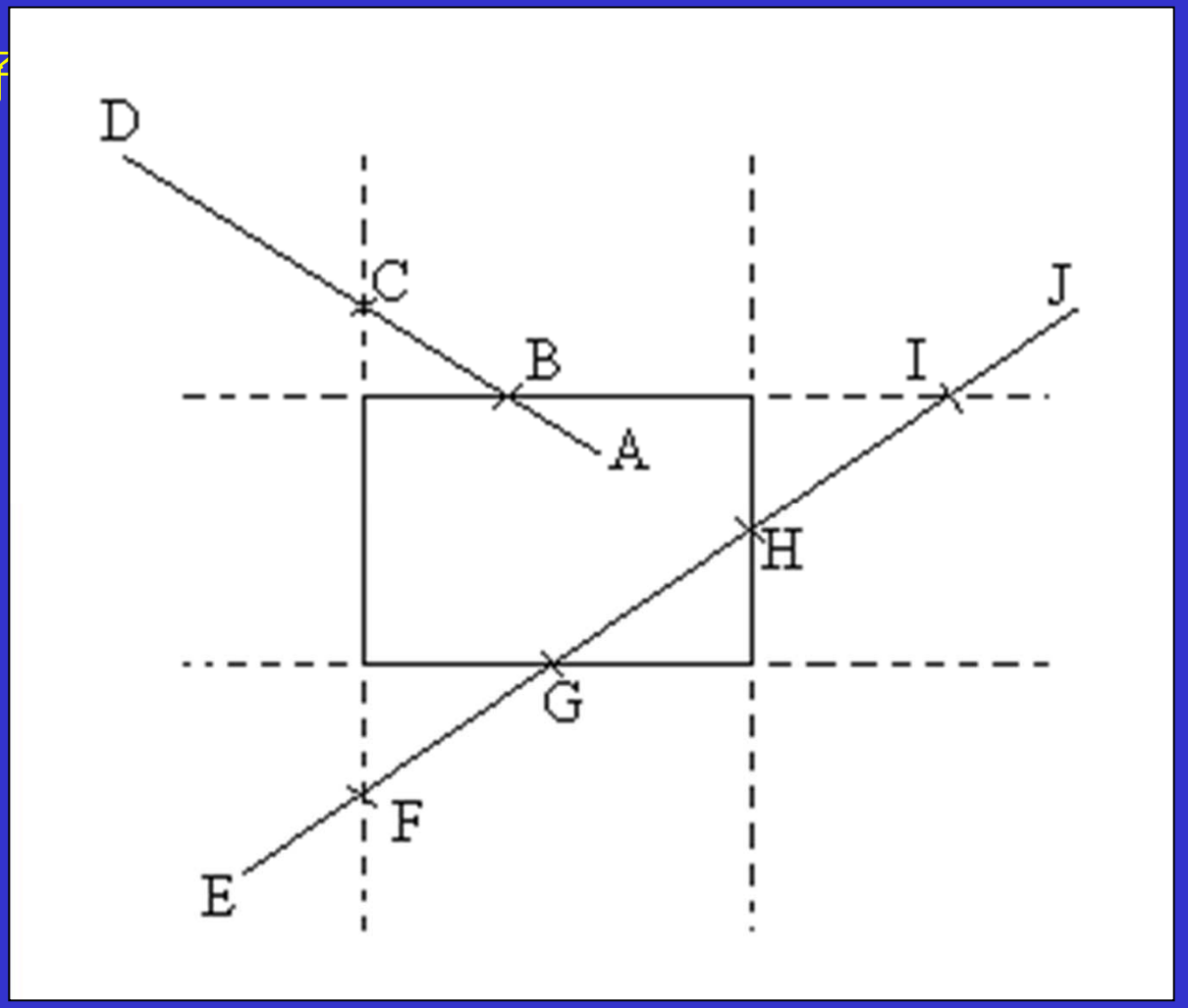
第一步　判别线段两端点是否都落在窗口内，如果是，则线段完全可见；否则进入第二步；

第二步　判别线段是否为显然不可见，如果是，则裁剪结束；否则进行第三步 ；

第三步　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一步、第二步判断。

Cohen-Sutherland 算法

例子



Cohen-Sutherland 算法

- 算法缺点：

  裁剪一条直线段仍然需要多次求交！

Nicholl-Lee-Nicholl算法

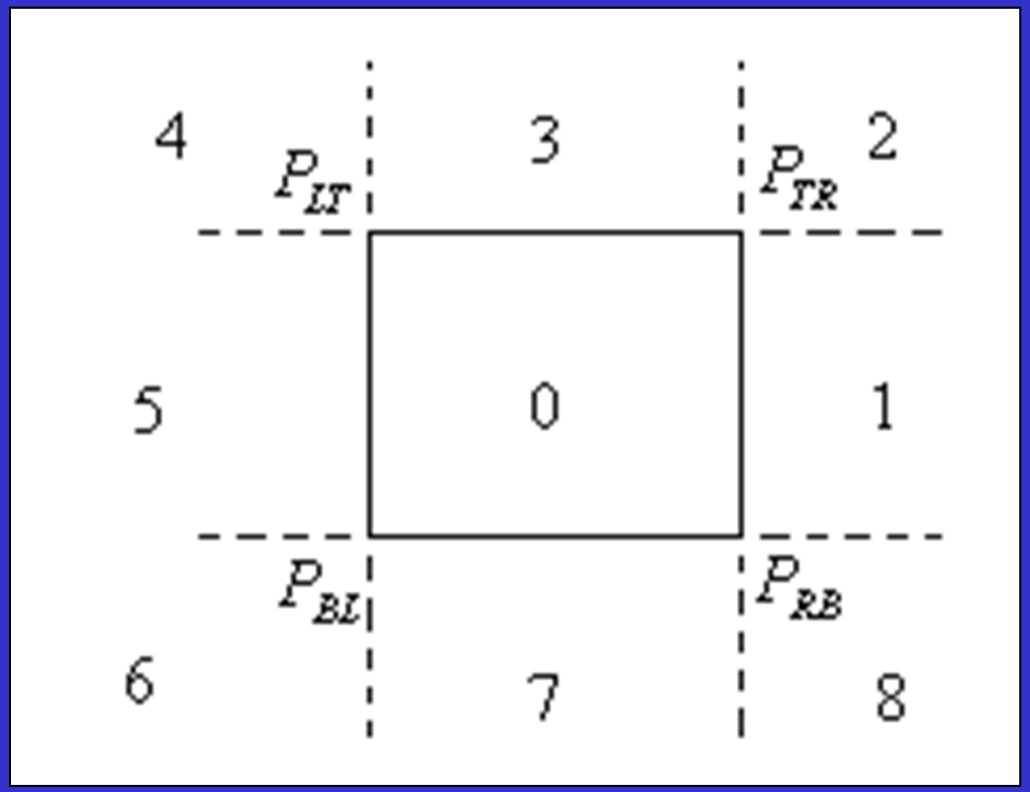
- ✓ NLN算法通过在裁剪窗口边界创立多个区域，从而避免对一条直线段进行多次裁剪，即在求交以前进行更多的区域测试，从而减少求交运算。

- ✓算法特点：仅仅用于二维裁剪。

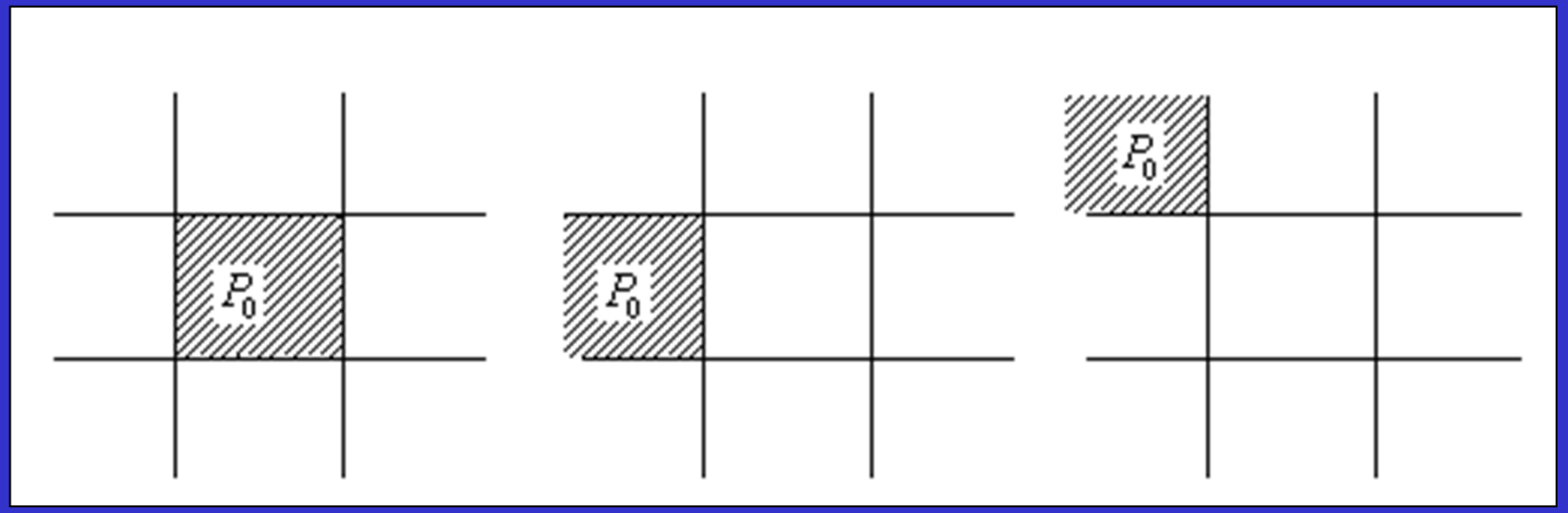
Nicholl-Lee-Nicholl算法

– 步骤：

第一步：窗口四边所在的直线将二维平面划分成9个区域，假定 P_0 落在区域0、4、5

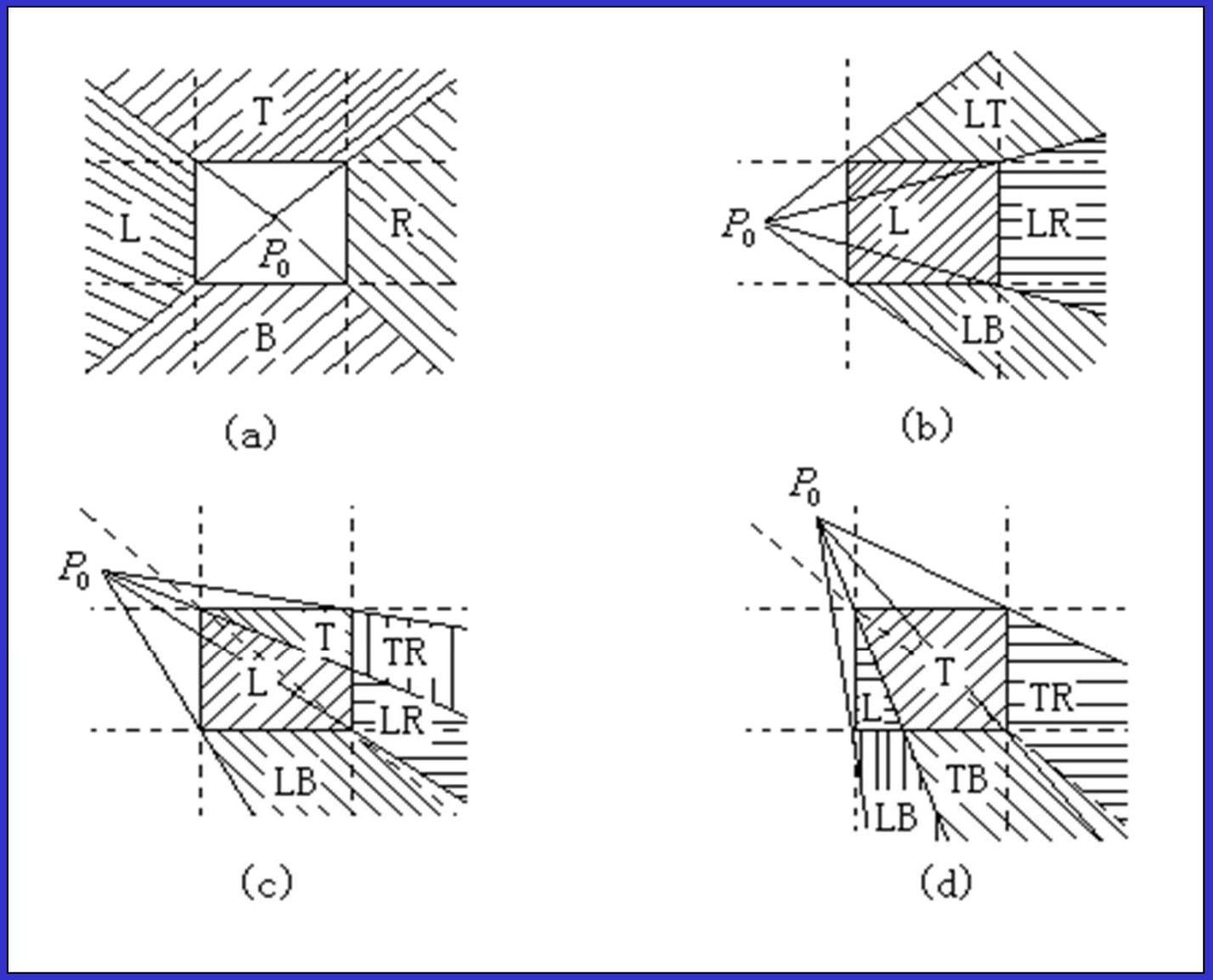


Nicholl-Lee-Nicholl算法



第二步：从P_0点向窗口的四个角点发出射线，这四条射线和窗口的四条边所在的直线一起将二维平面划分为更多的小区域 (四类情况)。

Nicholl-Lee-Nicholl算法



Nicholl-Lee-Nicholl算法

第三步：确定P_1所在的区域 。

方法：比较该线段的斜率和裁剪区域边界的斜率：

图（b）中：

$$\text{斜率}\overleftrightarrow{P_0P_{TR}} < \text{斜率}\overleftrightarrow{P_0P_{end}} < \text{斜率}\overleftrightarrow{P_0P_{TL}}$$

第四步：求交点，确定P_0P_1的可见部分 。

梁友栋-Barsky算法

- 已有的快速线段裁剪算法都在计算交点以前进行了更多的测试。梁友栋和Barsky分别提出了参数线裁剪的算法。

梁友栋-Barsky算法

- 直线的参数形式：

$$x = x_0 + u\Delta x$$
$$y = y_0 + u\Delta y \quad 0 \le u \le 1$$

其中，$\Delta x = x_{end} - x_0 \quad \Delta y = y_{end} - y_0$

- 直线在裁剪窗口内的条件：

$$xw_{\min} \le x_0 + u\Delta x \le xw_{\max}$$
$$yw_{\min} \le y_0 + u\Delta y \le yw_{\max}$$

梁友栋-Barsky算法

改写上式：

$$u \cdot p_k \le q_k, \quad k = 1,2,3,4$$

$$p_1 = -\Delta x,\ q_1 = x_0 - xw_{wmin}$$
$$p_2 = \Delta x,\ q_2 = xw_{wmax} - x_0$$
$$p_3 = -\Delta y,\ q_3 = y_0 - yw_{wmin}$$
$$p_4 = \Delta y,\ q_4 = yw_{wmax} - y_0$$

梁友栋-Barsky算法

- 当P_k=0时，(k=1,2,3,4对应于左、右、下、上边界)，如果还满足 $q_k < 0$, 则线段完全在边界之外，舍弃线段。

- 当$q_k \geq 0$时，线段位于平行边界内。

梁友栋-Barsky算法

- 当$P_k<0$时，线段从裁剪边界K的延长线的外部延伸到内部；当$P_k>0$时，反之。可以计算线段与边界K的延长线的交点u值：

$$u = \frac{q_k}{p_k}$$

梁友栋-Barsky算法

- 对每条直线，可计算出参数u_1和u_2，它们定义了裁剪矩形内的部分。
- u_1的值由线段从外到内遇到的矩形边界所决定$(p>0)$对于这些边界，计算$r_k=q_k/p_k$。u_1取0和各个r值中最大值。
- u_2的值由线段从内到外遇到的矩形边界所决定$(p>0)$。根据这些边界计算出r_k, u_2取1和各个r值中最小值。
- 如果$u_1>u_2$，则线段完全落在裁剪窗口之外，舍弃。否则由参数u的两个值计算出裁剪后的线段端点。

梁友栋-Barsky算法

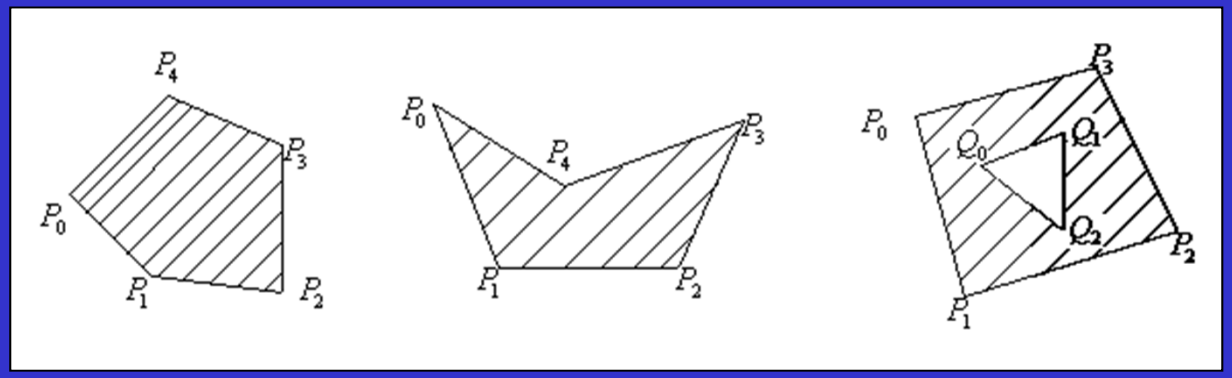
- 更新参数u_1和u_2仅仅需要一次除法；计算出u_1和u_2的最后值之后，线段与窗口的交点只计算一次。

- 比Cohen-Sutherland 算法减少了求交次数，算法效率更高；可扩展到三维裁剪算法。

非矩形裁剪窗口的线段裁剪算法

- 在某些应用中，需要使用任意形状的多边形对线段进行裁剪。

- 基于参数化的算法如梁-Barsky算法可扩充到凸多边形窗口。凹多边形窗口可以转换为多个凸多边形窗口来处理。



非线性裁剪窗口边界的线裁剪

- 也可以使用圆或其他曲线边界进行裁剪，由于求交涉及非线性曲线方程，速度慢。