计算机图形学

授课教师：王云霄

Email：w_yunxiao@yahoo.com.cn

QQ：63114592

第一章 计算机图形学简介

- 第一节　计算机图形学
- 第二节　计算机图形学的起源
- 第三节　计算机图形学的应用及发展动向
- 第四节　图形系统的硬件
- 第五节　计算机图形标准

第一节　计算机图形学

1. 与图形信息相关的计算机科学的分支

计算机图形学(Computer Graphics)

图像处理(Image Processing)

模式识别(Pattern Recognition)

2. 计算机图形学 指用计算机产生对象图形的输出的技术(简单)。

对象：具体或者抽象

图形分为两大类:(1) 线图(工程图,地图,曲线图等)

(2) 明暗图(照片类似)

计算机图形学是研究通过计算机将数据转换为图形，并在专门显示设备上显示的原理、方法和技术的学科。（**确切**）

计算机生成图形的过程:对象→模型→图形

⑴根据对象建立对象的模型(建模)

模型：能够正确地表达出一个对象性质、结构和行为的所有描述信息。

⑵利用计算机对这个模型进行必要的处理,产生能正确反映对象的某种性质的图形输出.

3. 图象处理是指用计算机来改善图象质量的数字技术.

(1) 如何消除噪声(去噪)

(2) 压缩图像数据以便于传输和存贮

(3) 用对比增强技术突出图像中的某些特征

(4) 用复原技术使模糊的图像清晰

(5) 从CT信息重建二维 三维图像

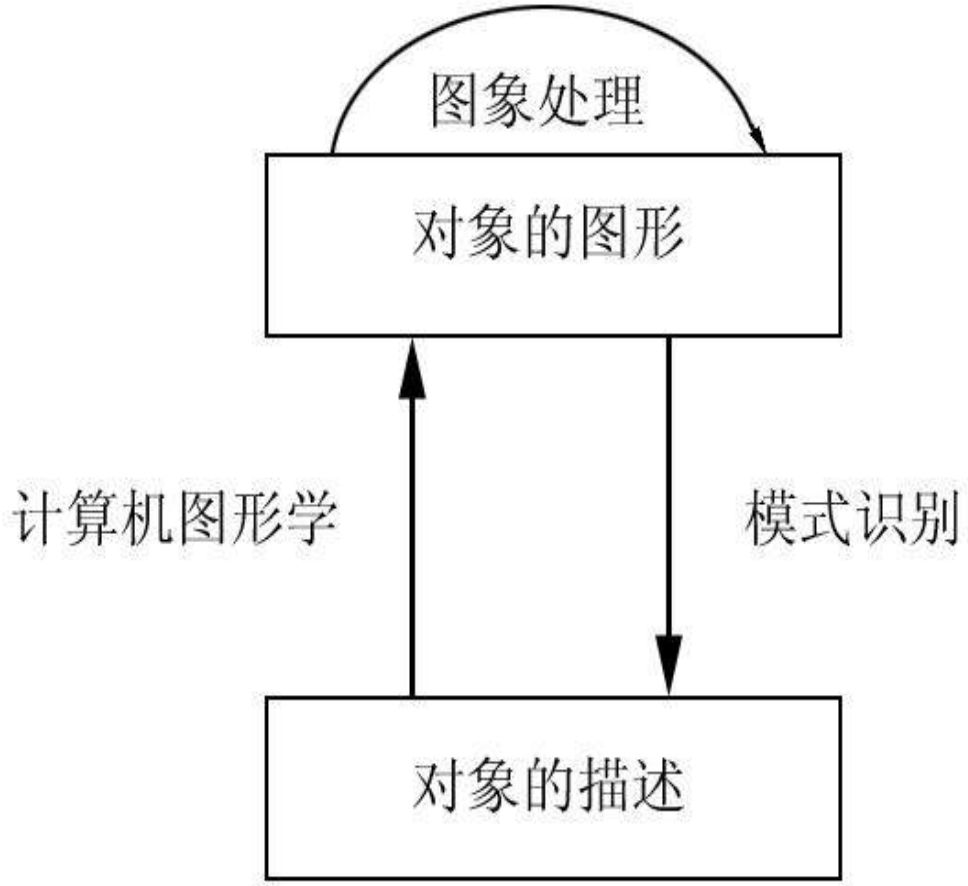
4. 模式识别是指用计算机对输入图形进行识别的技术。

特征提取

统计判定法或者语法分析法

(人像, 指纹, 味觉, 声音, 虹膜, 血管识别)

5.三者之间的关系



6. 与计算机图形学关系密切的学科

(1)计算几何学是研究几何模型和数据处理的学科。

曲线曲面的构造,三维立体造型,拼接

(2)交互式计算机图形学是指用计算机交互式地产生图形的技术。

交互设备是实现交互技术，完成交互任务的基础。

交互设备有定位、键盘、选择、取值和拾取。

交互任务是用户输入到计算机的一个单元信息，基本任务有四种：定位、字串、选择、取数。

第二节　计算机图形学的起源

1. 早期:打印机

1950年 与麻省理工学院的旋风计算机相连的显示器产生了简单的图形

第一台图形显示器的诞生

阴极射线管(CRT)

当时的计算机图形学称为”被动式”图形学

2. 50年代末期　MIT的林肯实验室在”旋风”机上开发的

SAGE空中防御系统

第一次使用了具有指挥和控制空能的CRT显示器(用笔在屏幕上指出目标)

交互式计算机图形学的诞生

//Interactive Computer Graphics

3. 1962年 MIT Ivan E Sutherland发表了博士论文
“sketchpad 人-机图形通信系统
首次使用”Computer Graphics”术语,为计算机图形学奠定了基础
证明交互式计算机图形学有前途的研究领域
4. 60年代中期
国际大公司(通用,贝尔等)研究计算机图形显示
计算机图形学的黄金时代
5. 70年代
计算机图形学广泛应用(CAD,教育等)
6. 80年代
应用领域(工业,管理,艺术),进入家庭
7. 90年代 科学计算可视化,虚拟现实VR的应用

第三节 计算机图形学的应用及发展动向

1. 计算机图形学的主要应用领域

(1) 用户接口(软件人性化设计)
(2) CAD/CAM. (精确易修改)
(3) 计算机仿真与动画(电影特技和飞行员的太空模拟驾驶舱)
(4) 制图学方面(人口分布图, 气象图, 股票走势图等)
(5) 过程控制(铁路, 机场调度)
(6) 办公自动化(多媒体技术, 可视电话, 视频会议)
(7) 艺术模拟(艺术创作, 传统油画, 中国国画, 书法等)
(8) 科学计算可视化(科学计算结果图形化)
(9) 虚拟现实(虚拟校园模拟)
(10) 教育领域


3ds max


3ds max


3ds max™

Low poly modelling
with edit mesh / edit poly,
and meshsmooth with creasing.
(untextured)

Glenn Melenhorst

3ds max™

2. 图形学主要研究内容

(1) 图形的生成和表示技术

(2) 图形的操作与处理方法

(3) 图形输出设备与输出技术的研究

(4) 图形输入设备、交互技术及用户接口技术的研究

(5) 图形信息的数据结构及存储、检索方法

(6) 几何模型构造技术(造型,颅骨”克隆”技术)

(7) 动画技术 3DMAX,FLASH

(8) 图形软硬件的系列化、模块化和标准化的研究

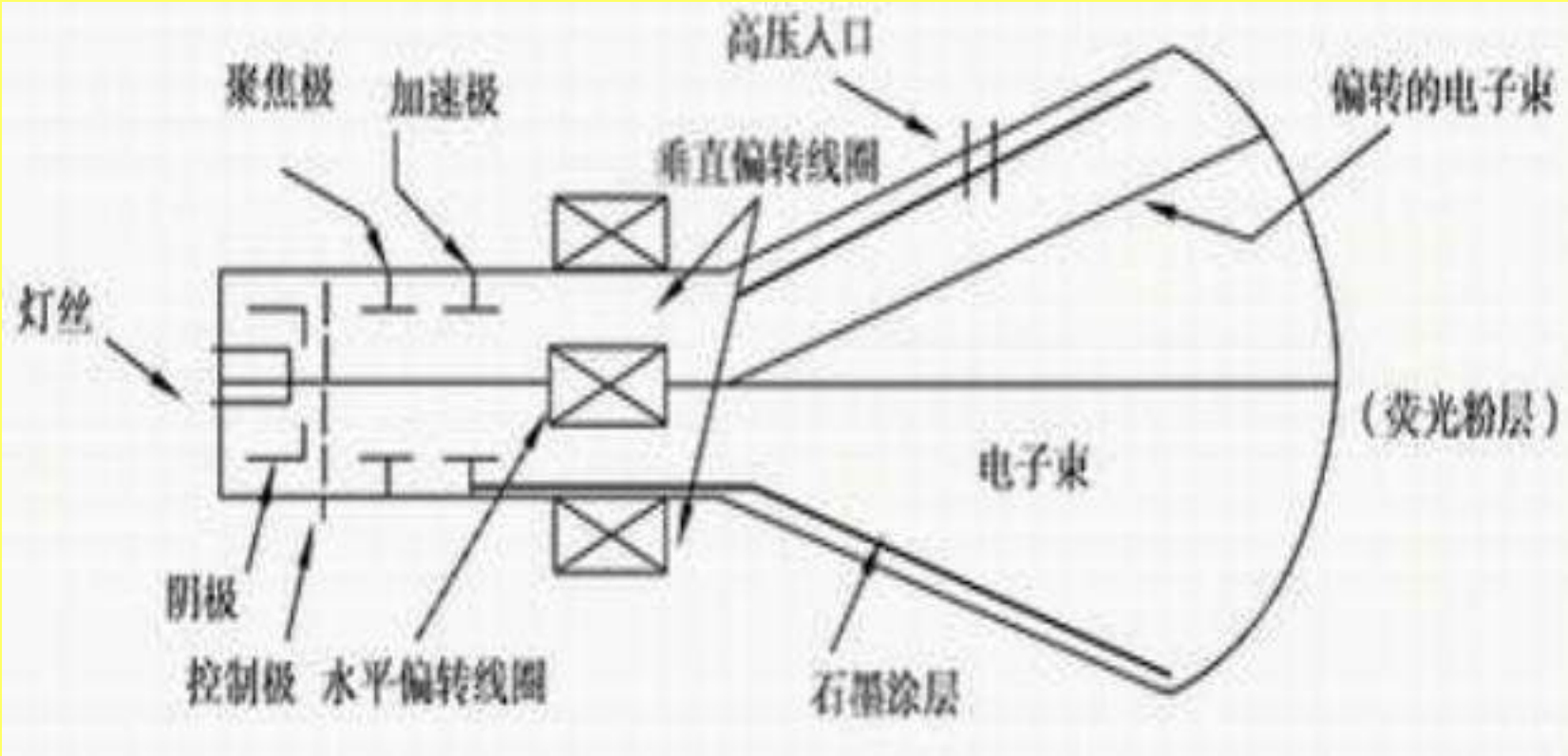
(9) 科学计算的可视化

第四节 图形系统的硬件

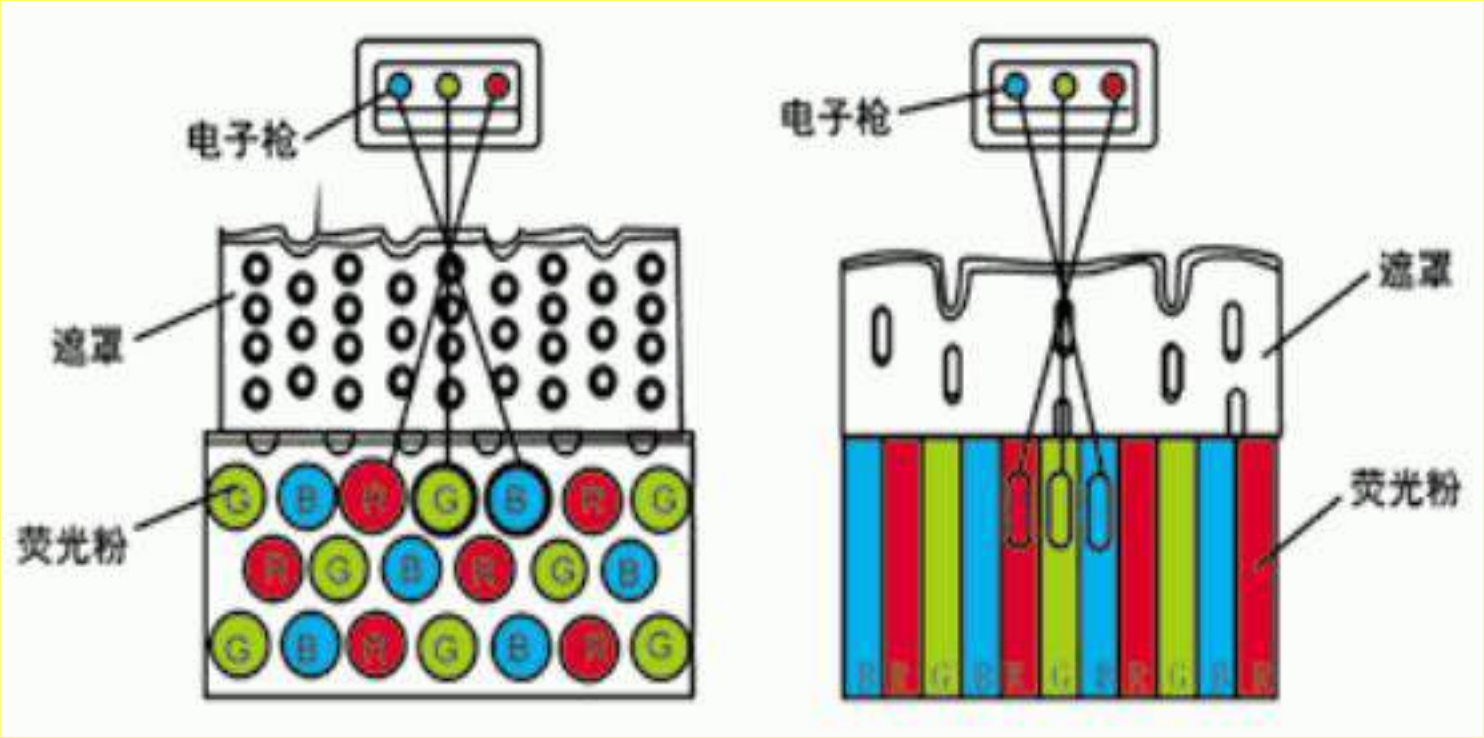
1. 计算机图形系统的硬件组成

   计算机、显示处理器(DPU)、图形显示器、输入设备和硬拷贝设备

2. CRT(Cathode Ray Tube)显示器工作原理图



电子枪(Electron gun)、偏转线圈(Deflection coils)、荫罩(Shadow mask)、荧光粉层(phosphor)和玻璃外壳（荧光屏）五部分组成



CRT显示器工作原理示意图

①阴极:被加热时发射电子

②控制极:控制电子束的偏转方向和运动速度

③聚焦极:保证电子束在轰击屏幕时汇聚成很细的点

④加速极:产生高速电子束

⑤偏转线圈:控制电子束在屏幕上的运动轨迹

⑥荧光粉层:被电子束轰击时发光

荧光屏上涂满了按一定方式紧密排列的红,绿,蓝三种颜色的荧光粉点或粉条,称为荧光粉单元,相邻的红绿蓝荧光粉单元各一个为一组,称之为象素,每个象素都拥有红绿蓝三原色。三个电子束分别受电脑显卡R、 G、 B三个基色视频信号电压的控制，去轰击各自的荧光粉单元

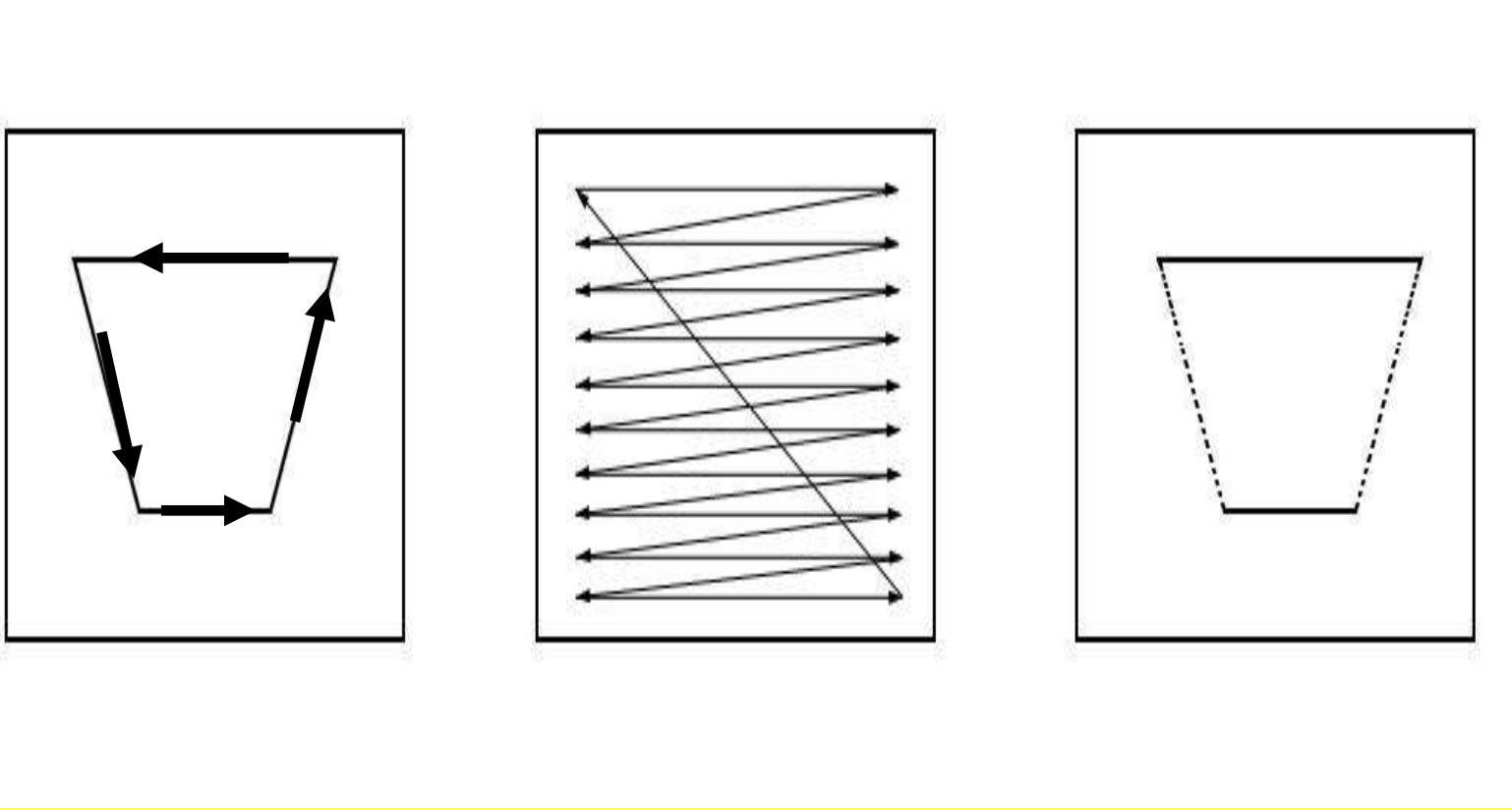
受到高速电子束的激发，这些荧光粉单元分别发出强弱不同的红、绿、蓝三种光。根据空间混色法(将三个基色光同时照射同一表面相邻很近的三个点上进行混色的方法)产生丰富的色彩，这种方法利用人们眼睛在超过一定距离后分辨力不高的特性，产生与直接混色法相同的效果。用这种方法可以产生不同色彩的像素，大量的不同色彩的像素可以组成一张漂亮的画面，而不断变换的画面就成为可动的图像。很显然，像素越多，图像越清晰、细腻，也就更逼真。

视觉残留特性和荧光粉的余辉

3. CRT显示器工作方式

(1) **随机扫描**方式（画线显示器，矢量显示器）

(2) **光栅扫描**方式



随机扫描　　光栅扫描　　光栅扫描

① 基本概念（性能指标）

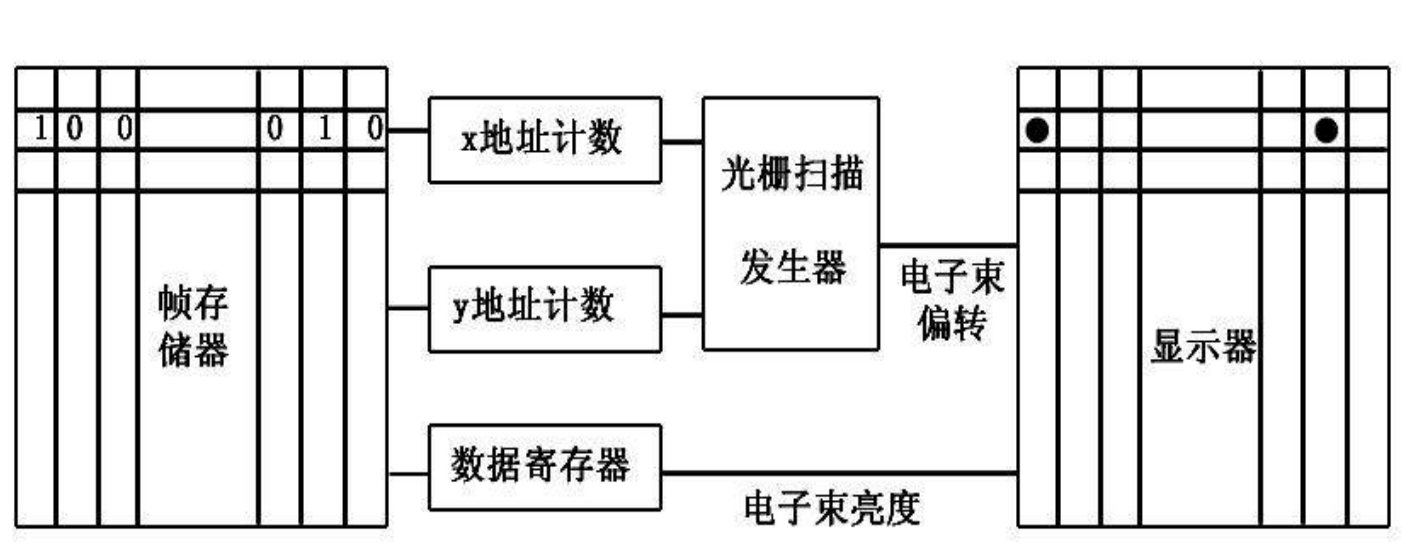
a. 象素(pixel)： 屏幕上可以点亮或熄灭的最小单位

b. 分辨率(resolution)：显示屏上象素的总数，常用每行的象素数与行数的乘积表示

c. 颜色或者亮度等级

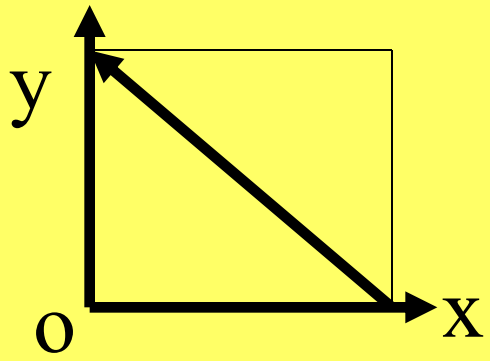
② 帧存储器:二维矩阵，帧存大小＝分辨率×单元字节

存储屏幕上每个象素对应的颜色或亮度值



光栅扫描显示结构图

③工作过程（从左到右，自上而下的扫描过程）



i：开始x←0,y ←ymax

帧存储器（x,y)位置的值被取出，设置CRT强度值，同时(x,y)控制电子束偏转

ii： x←x+1,处理下一个象素，直到处理完该行最后一个象素（x=xmax)

iii: x←0, y←y-1,处理下一行，重复执行直到最下面一条扫描线处理完

iv：回到开始处，刷新过程重复开始

4.等离子显示器PDP (Plasma Display Panel)

等离子显示屏：利用气体放电的显示装置，由三层玻璃板组成，在第一层的里面涂有导电材料的垂直条，中间层是灯泡阵列，第三层表面涂有导电材料的水平条 。

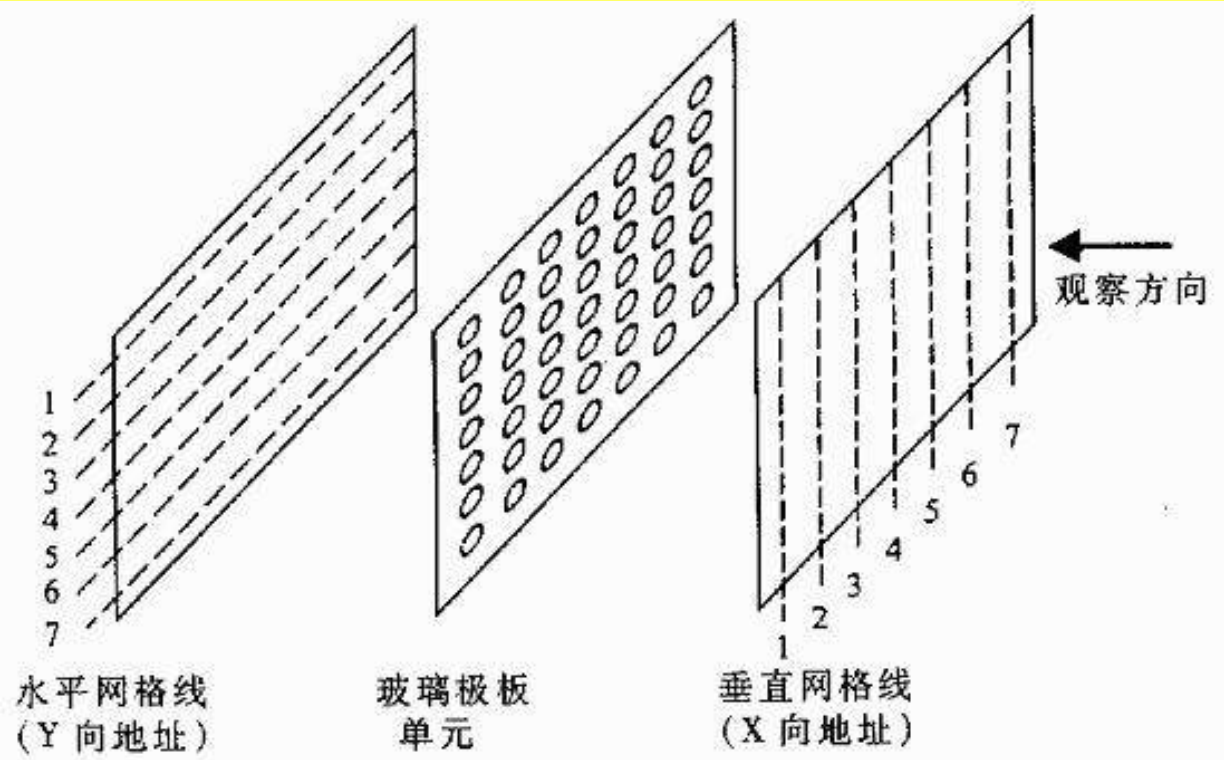


图 1 等离子显示器三层结构示意图

每个象素由一个等离子管构成，大量等离子管排列构成屏幕

↓

内充有氖氙气体

↓加高压

放电空间内的混合气体发生等离子体放电产生紫外光

↓

激活屏幕显示屏上红绿蓝三基色荧光粉而发光

独立的荧光粉象素发光

低电压维持发光（不用刷新存储器）

优点：（1） 高亮度、高对比度

(2) 纯平面图像无扭曲（CRT边缘的扫 描速度不均匀，很难控制到不失真的水平 ）

(3) 超薄设计、超宽视角

(4)具有良好的防电磁干扰功能。（显示原理不需要借助于电磁场 ）

（5）环保无辐射

缺点： 耗电量大

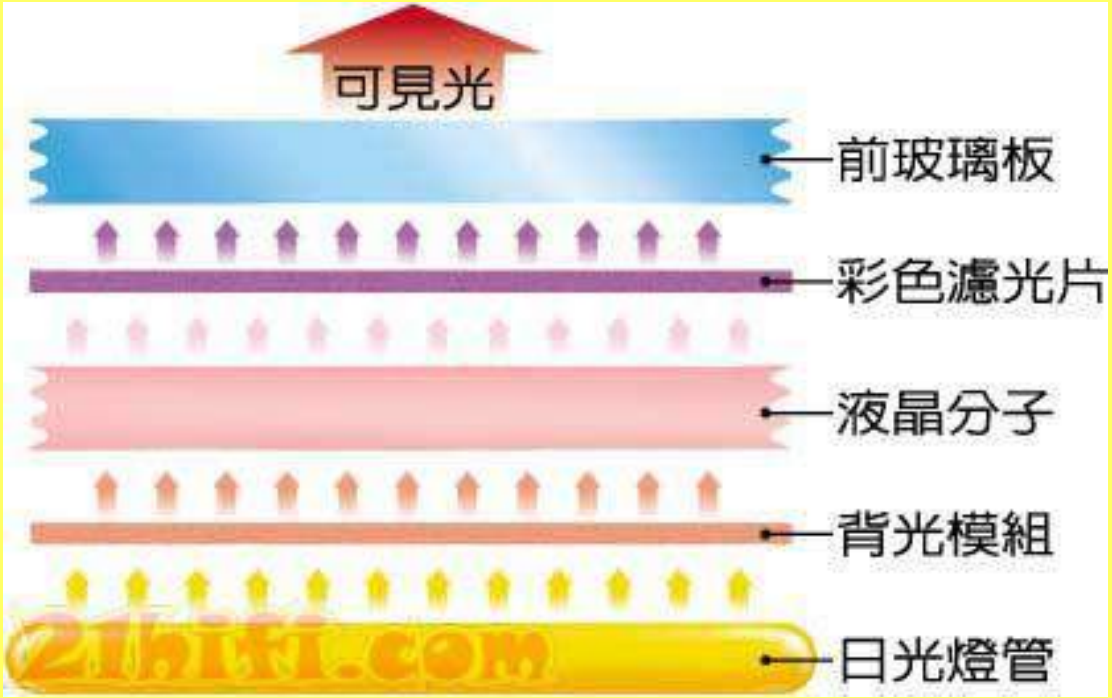
画质随时间递减

散热环境需良好

使用寿命较短

5.液晶显示器LCD (Liquid Crystal Display)

LCD是靠后方一组日光灯管发光，然后经由一组菱镜片与背光模块，将光源均匀地传送到前方，依照所接收的影像讯号，液晶画素玻璃层内的液晶分子会作相对应的排列，决定哪些光线是需偏折或阻隔的



优点：不闪烁省电、

缺点：反应时间慢、有残影、黑阶不纯、色彩饱和度差、视角不广造价高。

第五节　计算机图形标准

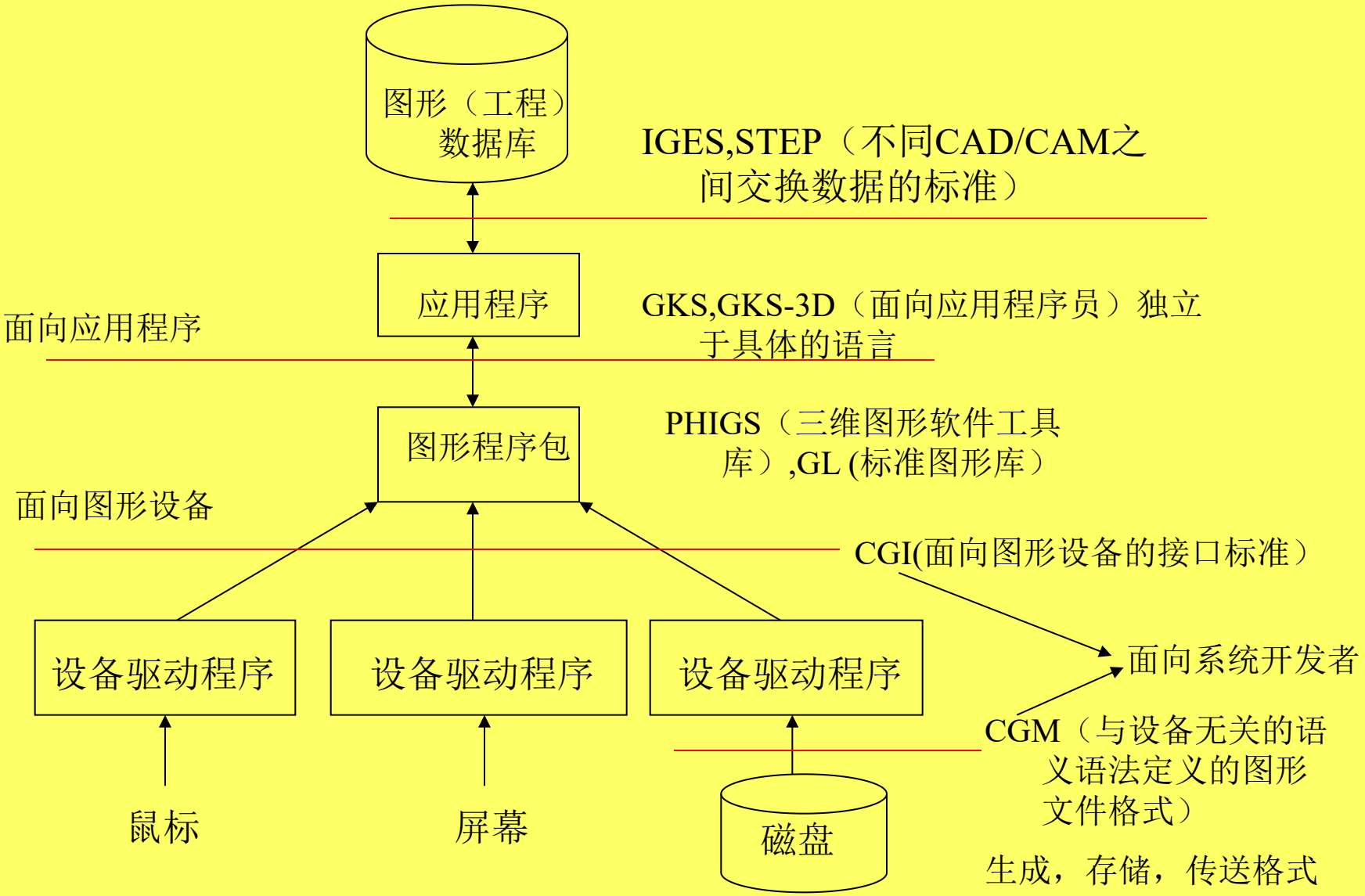
1. 计算机图形的标准是指图形系统及其相关应用系统中各界面之间进行数据传送和通信的接口标准，以及供图形应用程序调用的子程序功能及其格式标准，前者为数据及文件格式标准，后者为子程序界面标准.

1974年 ANSI “与机器无关的图形技术“会议上

1977年 美国计算机协会ACM提出了CGS(核心图形系统）

国际标准化组织ISO：　计算机图形核心系统及其语言联编（GKS）
三维图形核心系统及其语言联编（GKS-3D）
程序员层次交互式图形系统及其语言联编（PHIGS）
计算机图形接口（CGI）
计算机图形源文件（CGM）
基本图形转换规范（IGES)
产品数据转换规范（STEP）

2.图形系统中各界面的标准



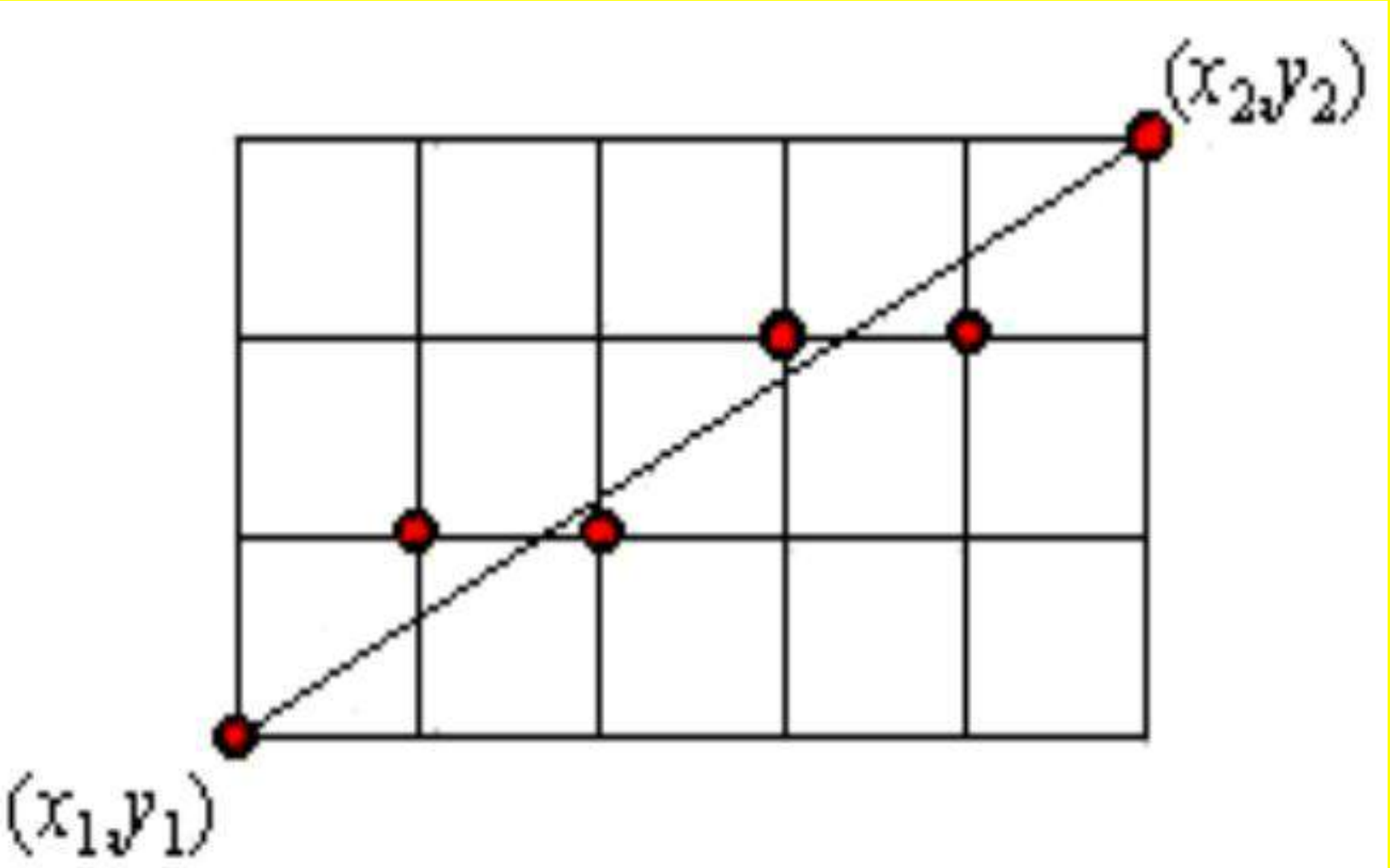
第二章 图形基元的显示

- 扫描转换 将图形描述转换成用象素矩阵表示的过程
- 图形基元（输出图形元素）图形系统能产生的最基本图形
- 线段、圆、多边形

- 第一节 直线扫描转换算法
- 第二节 圆的扫描转换算法
- 第三节 区域填充

第一节 直线扫描转换算法

- DDA直线扫描转换算法
- 中点画线法
- Bresenham画线算法



Mpaint演示

1.DDA线段扫描转换算法

设待画线段两端点的坐标值(x1,y1)和(x2,y2)，假定 x1<x2

y=mx+b

m=(y2-y1)/(x2-x1)

b=(x2y1-x1y2)/(x2-x1)

$|m| \leq 1$，对*x*每增1取允许的各整数值

$$m = \frac{\Delta y}{\Delta x} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

```
void DDALine(int x1,int y1,int x2,int y2)
{
    double dx,dy,e,x,y;
    dx=x2-x1;
    dy=y2-y1;
e=(fabs(dx)>fabs(dy))?fabs(dx):fabs(dy);
    dx/=e;
    dy/=e;
```

```
    x=x1;
    y=y1;
    for(int i=1;i<=e;i++)
    {

    SetPixel((int)(x+0.5), (int)(y+0.5));
    x+=dx;
    y+=dy;
    }
}
```

2.中点画线法

假定直线斜率在0、1之间

x= x_i时已选（x_i，y_i）象素

x= x_i +1与直线最近的象素

P_1（ x_i + 1，y_i ）、 P_2（ x_i+1，y_i+1）

M表示P_1与P_2的中点

M = （ x_i+1，y_i +0.5）。

Q是直线与垂直线$x = x_i + 1$的交点

显然，若M在Q的下方，则P_2离直线近，应取为下一个象素

中点画线法的实现：

起点和终点分别为

（ x_0 ， y_0 ）和（ x_1 ， y_1 ）。

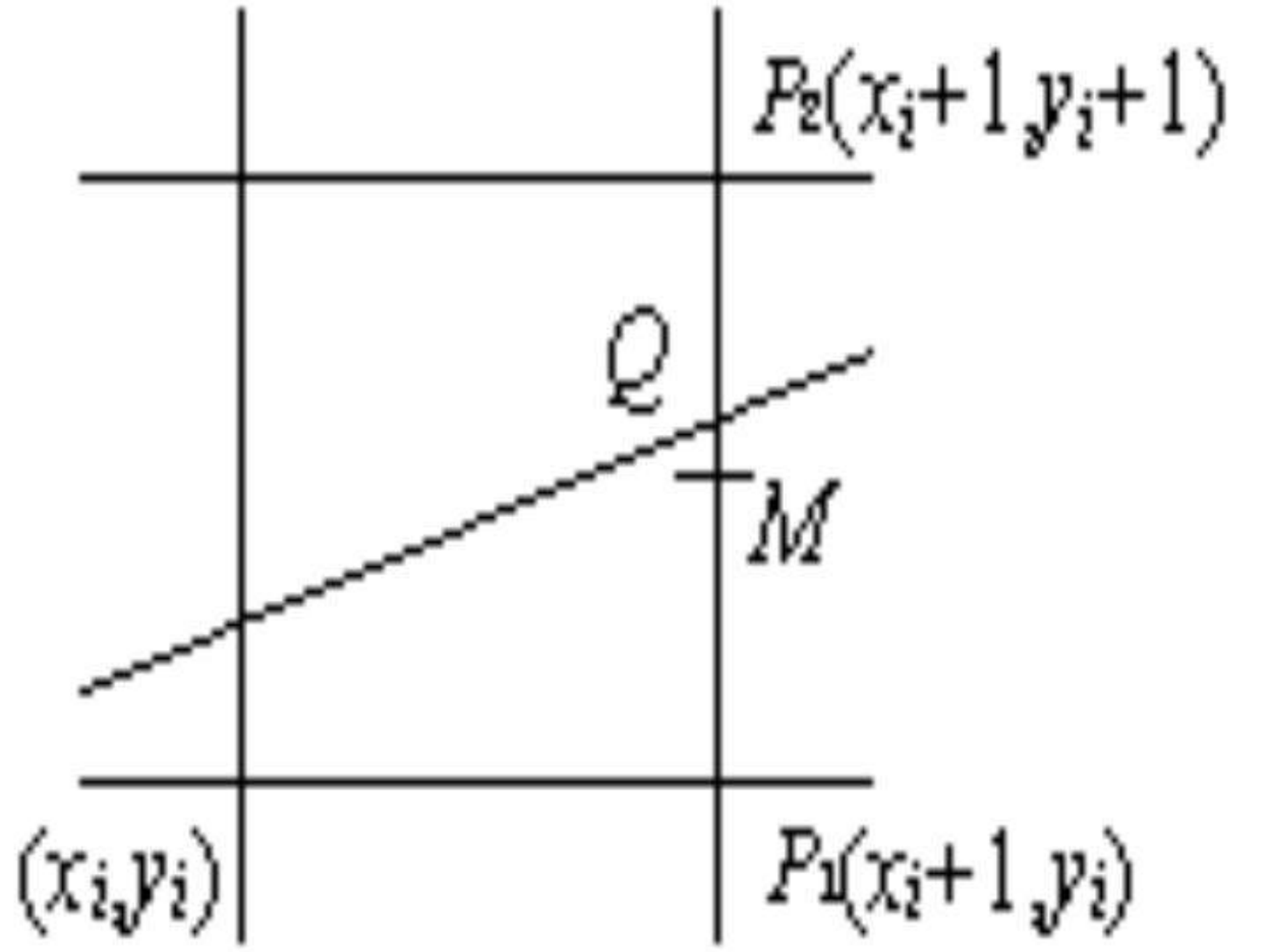
$F(x , y) = ax + by + c = 0$

其中，$a=y_0-y_1$，$b=x_1-x_0$，$c=x_0y_1-x_1y_0$ 。

直线上的点，$F(x, y) = 0$ ；

直线上方的点，$F(x, y)>0$ ；

直线下方的点，$F(x, y)<0$。


$P_2(x_i+1, y_i+1)$
Q
M
(x_i, y_i)
$P_1(x_i+1, y_i)$

Q在M的上方还是下方，只要把M代入$F(x, y)$，并判断它的符号。

$$d = F(M) = F(x_i+1, y_i+0.5)$$
$$=a(x_i+1) + b(y_i+0.5) + c$$

当 $d < 0$ 时，M在直线下方(即在Q的下方），应取右上方的P_2 。

而当$d > 0$，则取正右方的P_1。

当$d = 0$时，二者一样合适，取P_1。

对每一个象素计算判别式d，根据它的符号确定下一象素。

$d \geqslant 0$ 时，取正右方象素P_1，判断再下一个象素应取哪个，应计算

$d_1 = F(x_i+2, y_i+0.5) = a(x_i+2) + b(y_i+0.5) + c = d + a$

故d的增量为a。

而若$d<0$，则取右上方向素P_2。要判断再下一个象素，则要计算

$d_2 = F(x_i+2, y_i+1.5)$

$= a(x_i+2) + b(y_i+1.5) + c$

$= d + a + b$

故在第二种情况，d的增量为$a + b$

再看d的初始值。显然，第一个象素应取左端点(x_0, y_0)，相应的判别式值为

$d_0 = F(x_0+1, y_0+0.5)$

$=(x_0+1)+b(y_0+0.5)+c$

$= a x_0 + b y_0 + c + a + 0.5b$

$=F(x_0, y_0) + a + 0.5b$

但由于(x_0, y_0)在直线上，故$F(x_0, y_0)=0$。

因此，d的初始值为$d_0 = a+0.5b$

考虑用2d来代替d的计算

```
void MidpointLine (int x0,int y0,int x1,int y1)
{
    int a,b,delta1,delta2,d,x,y ;
    a = y0 - y1;
     b = x1 - x0;
     d = 2 * a + b ;
    delta1 = 2 * a ;
   delta2 = 2 *( a + b);
   x = x0 ;
   y = y0 ;
   SetPixel(x,y);
```

```
while( x<x1  )
      {
       if( d<0 )
              {  x ++;
                 y ++;
                 d+= delta2;
              }
      else
             { x ++;
              d+= delta1;
              }
      SetPixel(x,y);
      }
}
```

例:(0，0)、(5，2)　$a = y_0 - y_1 = -2$,

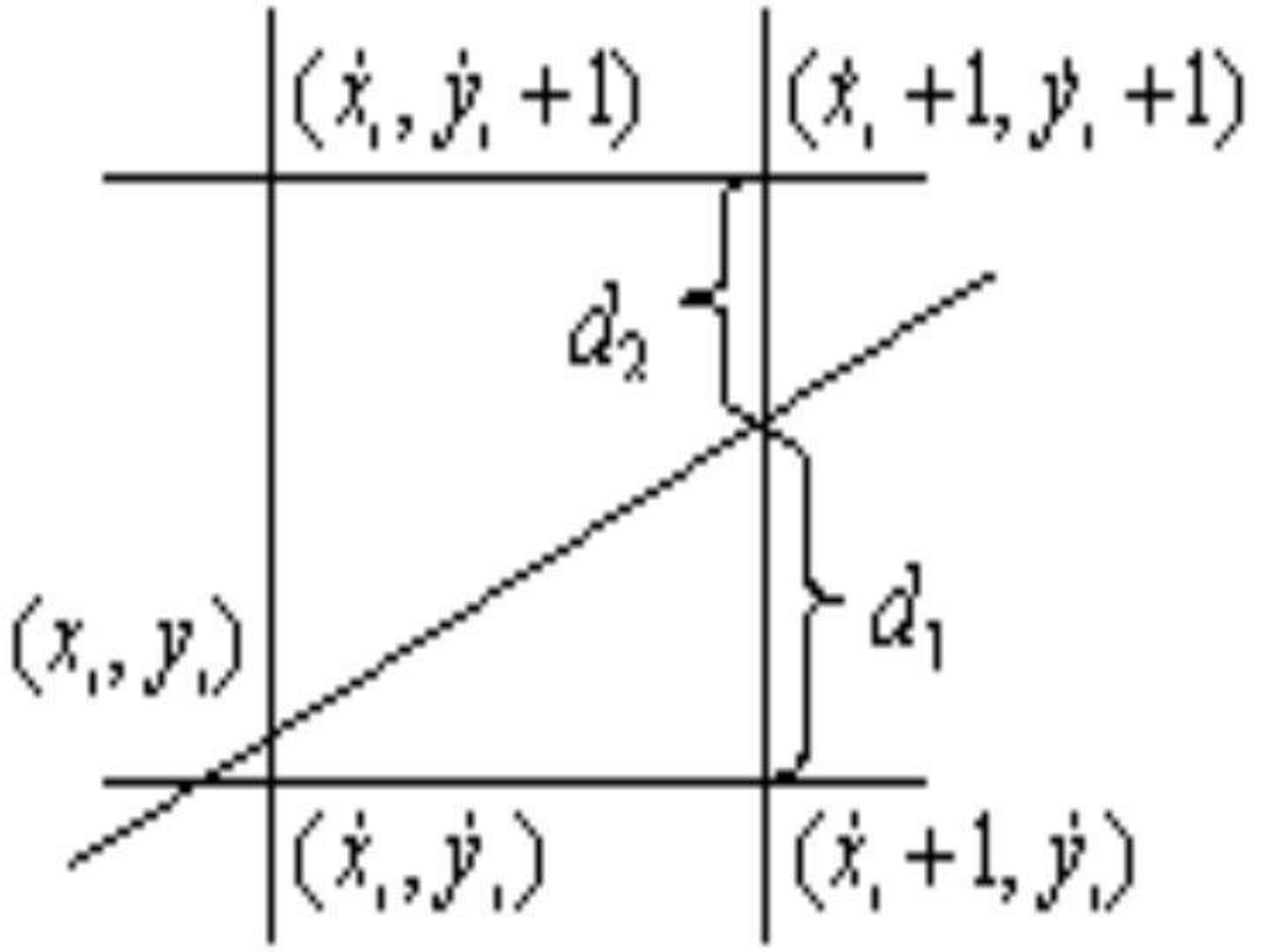
$b = x_1 - x_0 = 5, d_0 = 2 * a + b = 1$,

delta1=2a=-4, delta2=2(a+b)= 6

x	y	d
0	0	1
1	0	-3
2	1	3
3	1	-1
4	2	5
5	2	1

3. Bresenham画线算法

斜率*m*在0到1之间，并且$x_2 > x_1$. 设在第*i*步已经确定第*i*个象素点是(x_i, y_i)，它是直线上点$(\dot{x}_i, \dot{y}_i)$ 的最接近位置，$\dot{x}_i = x_i$ ， 现在看第*i*+1步如何确定第*i*+1个象素点的位置。


(x_i, y_i+1)
(x_i+1, y_i+1)
d_2
d_1
(x_i, y_i)
(x_i, y_i)
(x_i+1, y_i)

$$d_1 = y - \dot{y}_i = m(\dot{x}_i + 1) + b - \dot{y}_i$$

$$d_2 = (\dot{y}_i + 1) - y = (\dot{y} + 1) - m(\dot{x}_i + 1) - b$$

$$d_1 - d_2 = 2m(\dot{x}_i + 1) - 2\dot{y}_i + 2b - 1$$

$d_1 > d_2$，下一个象素点取 $(\dot{x}_i + 1, \dot{y}_i + 1)$

$d_1 < d_2$，下一个象素点取 $(\dot{x}_i + 1, \dot{y}_i)$

$d_1 = d_2$，取两象素点中的任意一个

$$p_i = \Delta x(d_1 - d_2)$$

$$= 2\Delta y \cdot \dot{x}_i - 2\Delta x \cdot \dot{y}_i + c$$

$$p_{i+1} = 2\Delta y \cdot \dot{x}_{i+1} - 2\Delta x \cdot \dot{y}_{i+1} + c$$

$$p_{i+1} - p_i = 2\Delta y - 2\Delta x(\dot{y}_{i+1} - y_i)$$

$p_i \geq 0$ 应取 $y_{i+1}=y_i+1$

$$p_{i+1} = p_i + 2(\Delta y - \Delta x)$$

$p_i < 0$ 应取 $y_{i+1}=y_i$

$$p_{i+1} = p_i + 2\Delta y$$

初始值确定P_1，

$$\dot{x}_1 = x_1, \dot{y}_1 = y_1 = \frac{\Delta y}{\Delta x} x_1 + b$$

令 i=1，可计算求出：

$$p_1 = 2\Delta y - \Delta x$$

```
void BresenhamLine(int x1,int y1,int x2,int y2)
 {
     int x,y,dx,dy,p;
     x=x1;
     y=y1;
     dx=x2-x1;
     dy=y2-y1;
     p=2*dy-dx;
     for(;x<=x2;x++)
     {
```

```
SetPixel(x,y);
        if(p>=0)
        {    y++;
             p+=2*(dy-dx);
        }
        else
        {
             p+=2*dy;
        }
    }
}
```

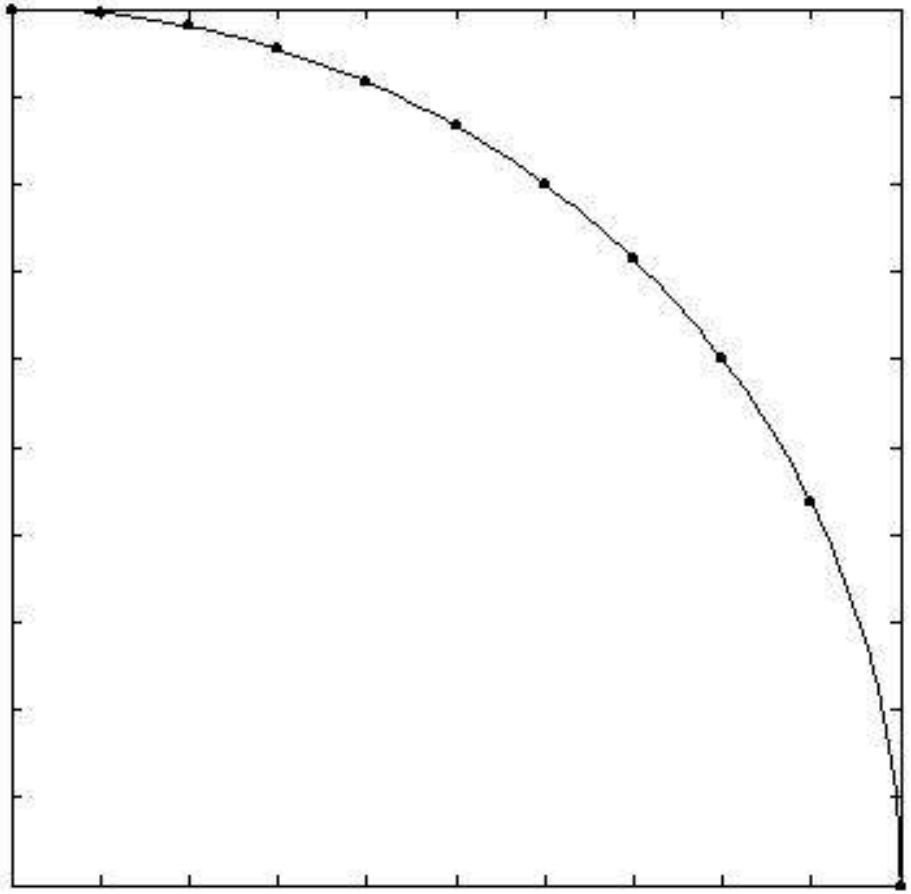
演示

第二节 圆的扫描转换算法

假定圆心是在原点，要画出半径为R的圆。

算法一：　$x^2+y^2=R^2$　$y=\sqrt{R^2-x^2}$

x以单位步长从0增加至R

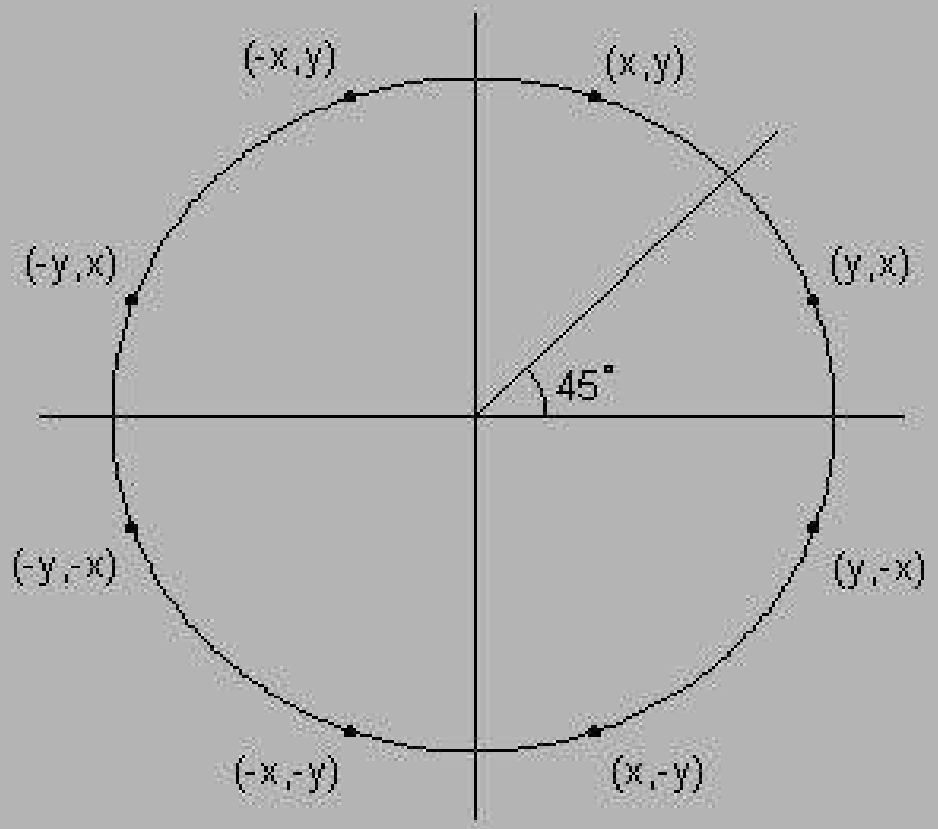


算法二:

$$x = R\cos\theta$$

$$y = R\sin\theta$$

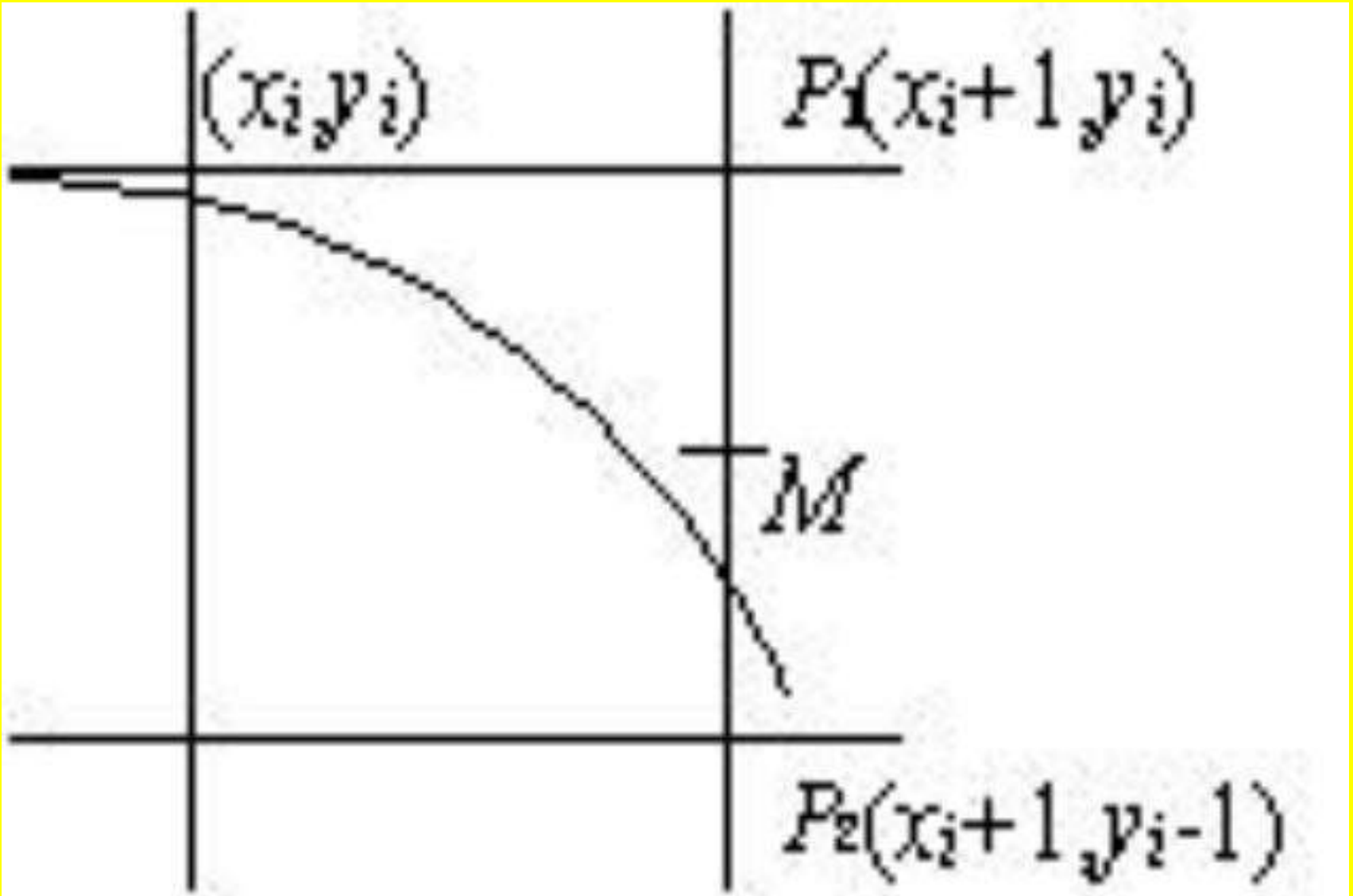
令 θ 以单位步长从0增至90°


(-x,y)
(x,y)
(-y,x)
(y,x)
45°
(-y,-x)
(y,-x)
(-x,-y)
(x,-y)

算法三：中点画圆法

讨论如何从点(0，R)至（$R/\sqrt{2}$，$R/\sqrt{2}$）的1 / 8圆周

若当前象素(x_i, y_i)，下一个象素只能是（x_i+1, y_i）的P_1点或（x_i+1, y_i−1）的P_2点


(xi,yi)
P1(xi+1,yi)
M
P2(xi+1,yi-1)

构造函数：$F(x, y)=x^2+y^2-R^2$

圆上的点，$F(x, y)=0$

圆外的点，$F(x, y)>0$

圆内的点，$F(x, y)<0$

P_1和P_2的中点为 $M=(x_i+1, y_i-0.5)$

$F(M)<0$时，M在圆内，取P_1

$F(M)>0$时，M在圆外，取P_2

$F(M)=0$时，P_1或P_2随取一个即可，约定取P_2

构造判别式

$$d = F(M) = F(x_i + 1, y_i - 0.5)$$

$$= (x_i + 1)^2 + (y_i - 0.5)^2 - R^2$$

若$d<0$，取P_1作为下一个象素，而且再下一个象素的判别式为

$$d' = F(x_i + 2, y_i - 0.5)$$

$$= (x_i + 2)^2 + (y_i - 0.5)^2 - R^2$$

$$= d + 2x_i + 3$$

而若$d \geqslant 0$，应取P_2作为下一个象素，而且再下一个象素的判别式

$$d' = F(x_i + 2, y_i - 1.5)$$

$$= (x_i + 2)^2 + (y_i - 1.5)^2 - R^2$$

$$= d + 2x_i - 2y_i + 5$$

第一个象素是(0, R)，判别式d的初始值为 $d_0 = F(1, R - 0.5) = 1 + (R - 0.5)^2 - R^2 = 1.25 - R$。

```
void MidpointCircle(int R)
{   int x,y;
      double d;
      x=0;y=R;d=1.25-R;
      SetPixel(x,y);
      while(x<y)
              {
                 if(d<0)
                       { d+=2*x+3; x++; }
                 else
                       { d+=2*(x-y)+5;x++;y--;}
                          SetPixel(x,y);
              }
}
```

改进一：在上述算法中，使用了浮点数来表示判别式d。

简化算法，在算法中全部使用整数，使用$e=d-0.25$代替d。

显然，初始化运算$d=1.25-R$对应于$e=1-R$。

判别式$d<0$对应于$e<-0.25$。

算法中可把d直接换成e。又由于e的初值为整数，且在运算过程中的增量也是整数，故e始终是整数，所以$e<-0.25$等价于$e<0$。

```
void MidpointCircle1(int R)
{ int x,y,d;
   x=0;y=R;d=1-R;
   SetPixel(x,y);
   while(x<y)
  {
        if(d<0)
        {d+=2*x+3;x++;}
      else
       {d+=2*(x-y)+5;x++;y--;}
   SetPixel(x,y);
  }
}
```

改进二：上述算法还可以进一步改进。以提高效率。注意到d的增量是x、y的线性函数，每当x递增1，d递增Δx=2。每当y递减1，d递增Δy =2。由于初始象素为(0, R)，所以Δx的初值为3，Δy的初值为$-2R$+2。

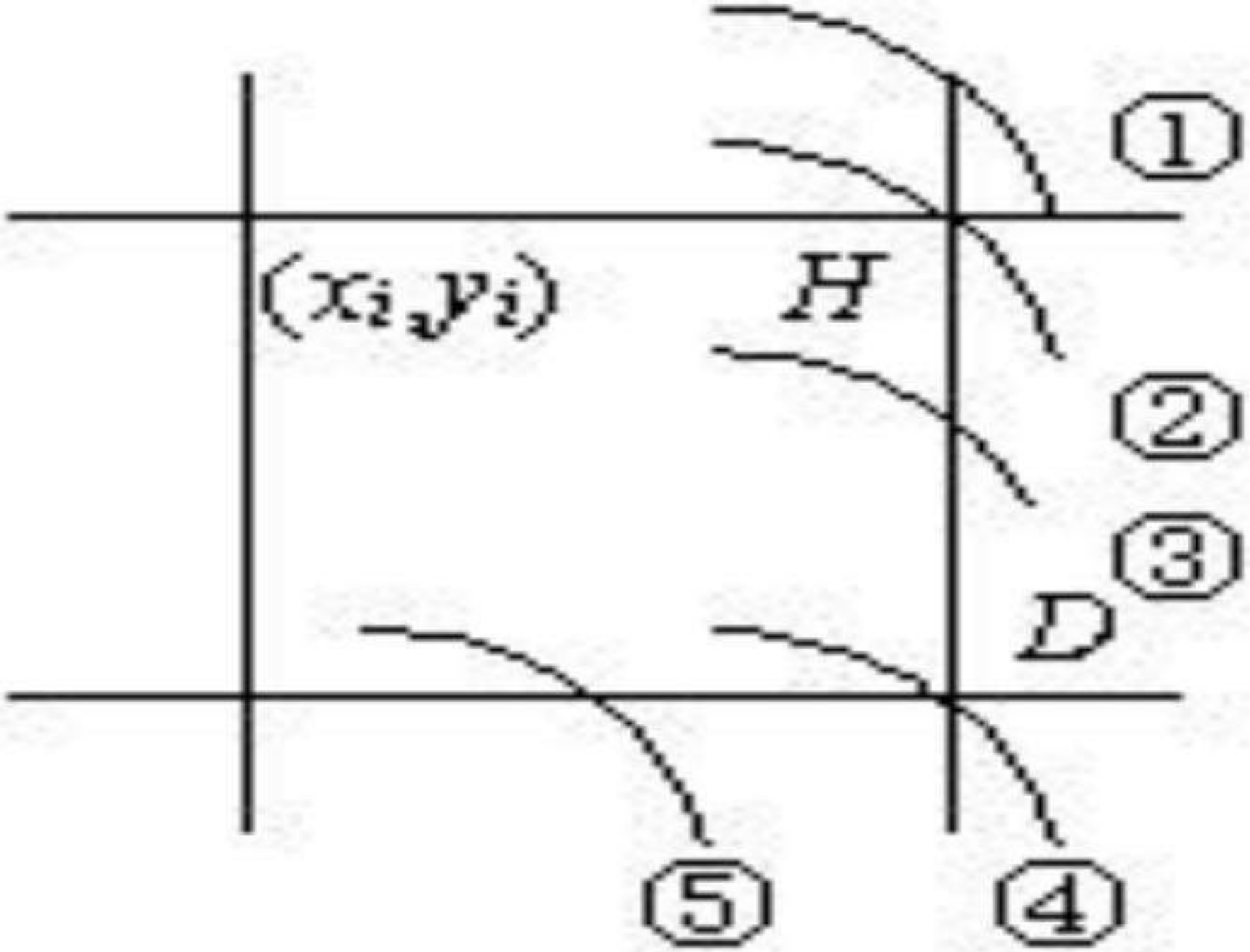
```
void MidpointCircle2(int R)
{    int x,y,deltax,deltay,d;
   x=0;y=R;d=1-R;
   deltax=3;
   deltay=2-R-R;
   SetPixel(x,y);
   while(x<y)
    {              if(d<0)
        {    d+=deltax;
             deltax+=2;
             x++;
```

```
            }
        else
        {     d+=(deltax+deltay);
              deltax+=2;
              deltay+=2;
              x++;
              y--;
        }
        SetPixel(x,y);
    }
}
```

算法四：Bresenham画圆算法

在$0 \leqslant x \leqslant y$的1/8圆周上，象素坐标$x$值单调增加，$y$值单调减少。

设第i步已确定（x_i, y_i）是要画圆上的象素点，看第$i+1$步象素点（x_{i+1}, y_{i+1}）应如何确定。下一个象素点只能是（x_i+1, y_i） 或者（x_i+1, y_i-1）中的一个


(xi,yi)
H
D
①
②
③
④
⑤

$$d_H = (x_i + 1)^2 + y_i^2 - R^2$$

$$d_D = R^2 - (x_i + 1)^2 - (y_i - 1)^2$$

构造判别式：$p_i = d_H - d_D$

若精确圆弧是①或②，显然H是应选择点，而此时$d_H \leqslant 0$，$d_D > 0$，必有$p_i < 0$。

精确圆弧是③，则$d_H > 0$和$d_D > 0$．若$p_i < 0$，即$d_H < d_D$应选H点。若$p_i \geqslant 0$，即$d_H \geqslant d_D$应选D点。

若精确圆弧是④或⑤，显然D是应选择点，而此时$d_H > 0$，$d_D \leqslant 0$，必有$p_i > 0$。

得出结论：p_i做判别量，$p_i < 0$选H点为下一个象素点，当$p_i \geqslant 0$时，选D点为下一个象素点。

从 p_i 计算 p_{i+1}

$$p_i = 2(x_i+1)^2 + 2y_i^2 - 2y_i - 2R^2 + 1$$

$$p_{i+1} - p_i = 2(y_{i+1}^2 - y_i^2 - y_{i+1} + y_i) + 4x_i + 6$$

当$p_i \geqslant 0$时，应选D点，即选$y_{i+1} = y_i - 1$

$$p_{i+1} = p_i + 4(x_i - y_i) + 10$$

当p_i<0时，应选H，即选 $y_{i+1}=y_i$

$$p_{i+1}=p_i+4x_i+6$$

画圆的起始点是(0，R)，
即x_1=0，y_1=R，代入前式，令 i=1，
就得到：

$$p_i=3-2R$$

```
void BresenhamCircle(int R)
{    int x,y,p;
     x=0;  y=R;
     p=3-2*R;
          for(;x<=y;x++)
     {     SetPixel(x,y);
                if(p>=0)
          {
                     p+=4*(x-y)+10;
                     y--;
          }
```

```
else
                    {     p+=4*x+6;
            }
            }
}
```

只需修改语句SetPixel(x,y)，画八个对称的点，就可以画出全部圆周。若加一个平移，就可以画出圆心在任意位置的圆周。

演示

椭圆的扫描转换算法：

中点画圆法推广到一般的二次曲线

$$F(x,y)=b^2x^2+a^2y^2-a^2b^2=0$$

椭圆上一点(x,y)处的法向量为

$$N(x,y)=\frac{\partial F}{\partial x}i+\frac{\partial F}{\partial y}j=2b^2xi+2a^2yj$$

(i,j)为x轴, y轴的单位向量

上半部分： $(x_p,y_p),M(x_p+1,y_p-0.5)$,法向量的$y$的分量更大

下半部分： $(x_p,y_p),M(x_p+0.5,y_p-1)$,法向量的$x$的分量更大

椭圆的扫描转换算法:

$F(x,y)=b^2x^2+a^2y^2-a^2b^2$

法向量为（$2b^2x,2a^2y$)

当前中点:$2b^2(x_p+1)<2a^2(y_p-0.5)$上半部分

下一个中点:$2b^2(x_p+1)>2a^2(y_p-0.5)$下半部分

上半部分:$(x_p, y_p), (x_p + 1, y_p - 0.5)$

$d_1 = F(x_p + 1, y_p - 0.5) = b^2(x_p + 1)^2 + a^2(y_p - 0.5)^2 - a^2b^2$

$d_1 < 0$,中点在椭圆内，取正右方象素

$d_1' = F(x_p + 2, y_p - 0.5) = b^2(x_p + 2)^2 + a^2(y_p - 0.5)^2 - a^2b^2$

$= d_1 + b^2(2x_p + 3)$

$d_1 > 0$,中点在椭圆外，取右下方象素

$d_1' = F(x_p + 2, y_p - 1.5) = b^2(x_p + 2)^2 + a^2(y_p - 1.5)^2 - a^2b^2$

$= d_1 + b^2(2x_p + 3) + a^2(-2y_p + 2)$

d_1初值$d_1^0 = F(1, b - 0.5) = b^2 + a^2(-b + 0.25)$

下半部分:(x_p, y_p),下一个点为(x_p, y_p-1)或者(x_p+1, y_p-1)

$M=(x_p+0.5, y_p-1)$

$d_2=F(x_p+0.5, y_p-1)=b^2(x_p+0.5)^2+a^2(y_p-1)^2-a^2b^2$

$d_2<0$,中点在椭圆内，取右下方象素

$d_2^{'}=F(x_p+1.5, y_p-2)=b^2(x_p+1.5)^2+a^2(y_p-2)^2-a^2b^2$

$=d_2+b^2(2x_p+2)+a^2(-2y_p+3)$

$d_2\geq 0$,中点在椭圆外，取正下方象素

$d_2^{'}=F(x_p+0.5, y_p-2)=b^2(x_p+0.5)^2+a^2(y_p-2)^2-a^2b^2$

$=d_2+a^2(-2y_p+3)$

d_2初值$d_2^{\,0}=F(x,y)=(b(x+0.5))^2+(a(y-1))^2-a^2b^2$

(x,y)为交界点

第三节 区域填充

. 种子填充算法

. 多边形扫描转换算法

一：种子填充算法

1. 区域是指光栅网格上的一组象素。
区域填充是把某确定的象素值送入到区域内部的所有象素中。

2. 区域填充方法分为两大类。

(1)区域由多边形围成，区域由多边形的顶点序列来定义；
优点：占用内存少，缺点:得区分内外侧

(2)是通过象素的值来定义区域的内部
优点:形状可任意复杂，可直接着色
缺点：几何意义不直观

3. 用象素值定义区域

（1）内定义区域（oldvalue)指出区域内部所具有的象素值,（内点表示）

（2）边界定义区域（boundaryvalue)，指出区域边界所具有的象素值。此时区域边界上所有象素具有某个边界boundaryvalue。区域的边界应该是封闭的，指明区域的内部和外部。

以象素为基础的区域填充主要是依据区域的连通性进行。

4. 区域的连通性

（1）**四连通**:从区域中的一个象素出发，经连续地向上下左右四个相邻象素的移动，就可以到达区域内的**任意**另一个象素.

（2）**八连通**:如果除了要经上下左右的移动，还要经左上、右上、左下和右下的移动，才能由一个象素走到区域中另外**任意**一个象素.

四连通区域必定是八连通区域，反之不一定

左上	上	右上
左	p	右
左下	下	右下

5. 种子填充算法

（1）四连通内定义区域填充算法：

```
void   Floodfill(int x,int y,COLORREF oldvalue,COLORREF newvalue)
/*(x，y)为种子 oldvalue是原值 newvalue是新值，应不等于原值。*/
{ if (GetPixel(x,y) == oldvalue) (是否在区域内并且尚未被访问过）
      {  SetPixel(x,y,newvalue);
Floodfill(x,y-1,oldvalue,newvalue);//上
Floodfill(x,y+1,oldvalue,newvalue);//下
Floodfill(x-1,y,oldvalue,newvalue); //左
Floodfill(x+1,y,oldvalue,newvalue); //右
      }
}
```

优点：算法简单　缺点：重复多

（2）四连通边界定义区域填充算法：

```
void   Boundaryfill(int x,int y,COLORREF
boundaryvalue,COLORREF newvalue)

/*(x, y) 为种子位置，boundaryvalue是边界象素值

newvalue是区域内象素新值，未填充前区域内不应有
值为newvalue的象素。*/

{   if( GetPixel(x,y)!=boundaryvalue

        && GetPixel(x,y)!=newvalue)

                        //  未达边界且未访问过
```

```
    {
  SetPixel(x,y,newvalue);//赋以新值
Boundaryfill(x,y-1,boundaryvalue,newvalue);
Boundaryfill(x,y+1,boundaryvalue,newvalue);
Boundaryfill(x-1,y,boundaryvalue,newvalue);
Boundaryfill(x+1,y,boundaryvalue,newvalue);
    }
```

} 算法简单，多层递归，存储空间有限—》堆栈溢出

演示

（3）扫描线种子填充算法(适用于边界定义的四连通区域)

象素段：将区域内由边界点限定的同一行内相连接的不具有新值newvalue的一组象素称为一个象素段，象素段用它最右边的象素来标识。

算法的步骤如下：

1. [初始化]将堆栈置为空，将给定的种子点（x, y)压入堆栈
2. [出栈]如果堆栈为空，算法结束，否则取栈顶元素（x, y）作为种子点
3. [区段填充]从种子点（x, y)开始沿当前扫描线向左右两个方向逐个象素进行填充，直到遇到边界点为止
4. [定范围]用xl和xr分别表示在步骤3中填充的区段的左右两个端点的横坐标
5. [进栈]对当前扫描线上下相邻的两条扫描线从右向左的确定位于[xl, xr]区域内的象素段。如果区域内的象素已经填充或为边界则转到步骤2，否则取象素段的右端点作为种子点，压入堆栈，再转到步骤2。

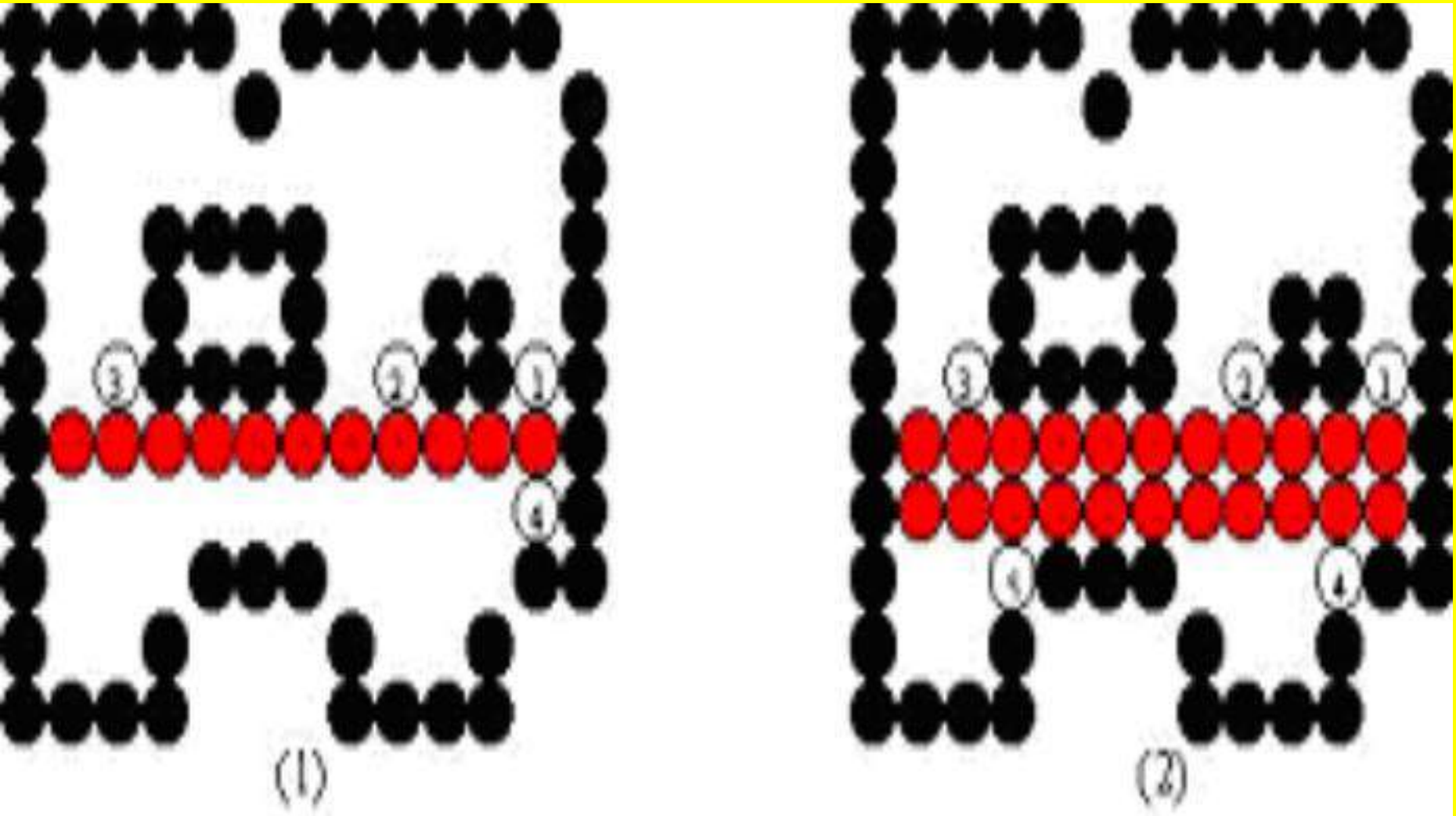
下面我们用伪C语言写出扫描线填充算法。

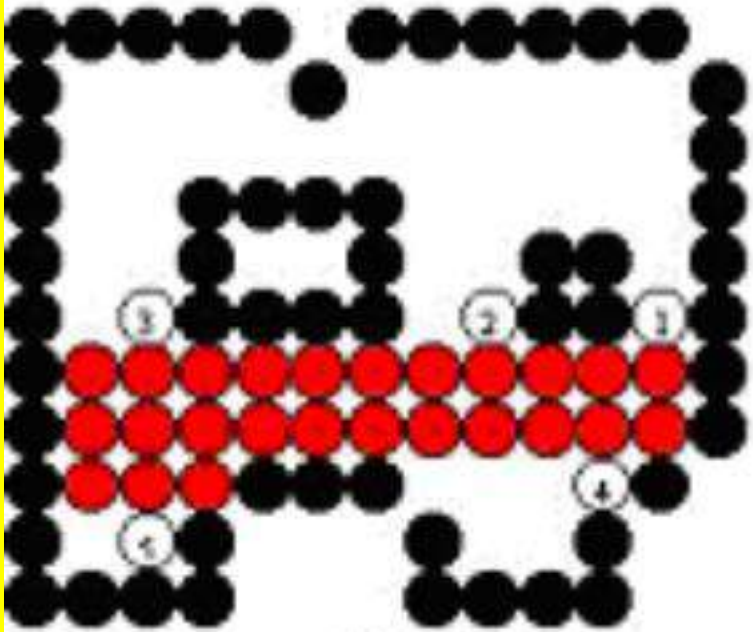
```
void ScanlineSeedfill(intx,inty,COLORREF
                                boundaryvalue,COLORREF newvalue)
{
    int x0,xl,xr,y0,xid;
    int flag;
    Stack s;
    Point p;
    s.push(Point(x,y));//种子象素入栈
    while(!s.isempty())
   {
     p=s.pop(); //栈顶象素出栈
     x=p.x;y=p.y;
     SetPixel(x ,y ,newvalue);
     x0= x+1;
```

```
while(GetPixel(x0,y)!=boundaryvalue)//填充右方象素
{
    SetPixel(x0 ,y ,newvalue);
    x0++;
}
xr=x0-1;//最右边象素
x0= x-1;
while(GetPixel(x0,y)!=boundaryvalue)//填充左方象素
{
    SetPixel(x0 ,y ,newvalue);
    x0--;
}
xl=x0+1;//最左边象素
```

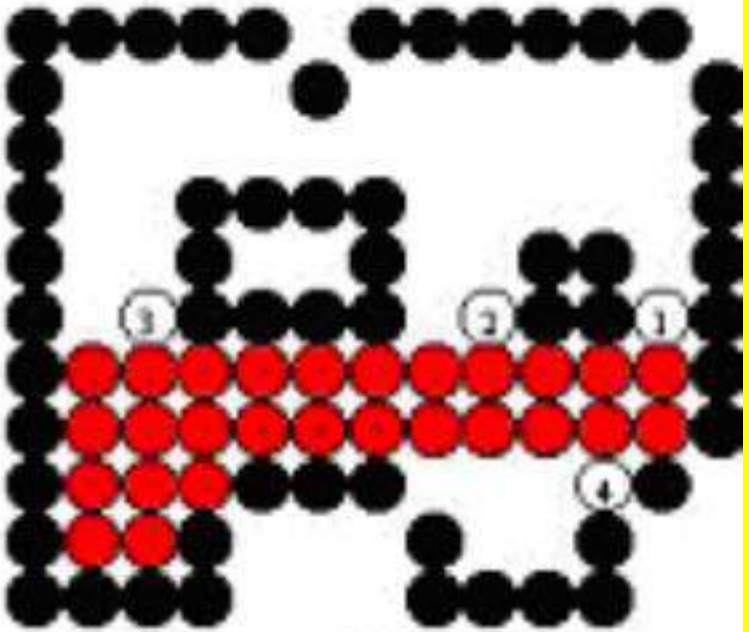
```
//检查上一条扫描线和下一条扫描线，
//若存在非边界且未填充的象素，
//则选取代表各连续区间的种子象素入栈。
 y0=y;
 for(int i=1;i>=-1;i-=2)//从最右象素开始寻找上下两行的象素段
{
   x0=xr;
   y=y0+i;
   while(x0>=xl)
{   flag=0;
     while((GetPixel(x0,y)!=boundaryvalue)&&
                (GetPixel(x0,y)!=newvalue) && (x0>xl))
      {   if(flag==0)
        { flag=1;//标志找到新的象素段
          xid=x0;
        }
          x0--;
      }
```

```
//将最右侧可填充象素压入栈中
        if(flag==1)
         {
             s.push(Point(xid,y));
             flag=0;
         }
//检查当前填充行是否被中断，若被中断，寻找左方第一个可填充象素
 //判断当前点是否为边界点//判断当前点是否为已填充点
    while((GetPixel(x0,y)==boundaryvalue
        ||(GetPixel(x0,y)==newvalue))
    x0--;//若当前点为边界点或已填充点，根据前面的判断，当前点必然未超出左边界，则当前点向左移动
             }//while(x0>=xl)
        }//for(int i=1;i>=-1;i-=2)
  }//while(!s.isempty())
}
```

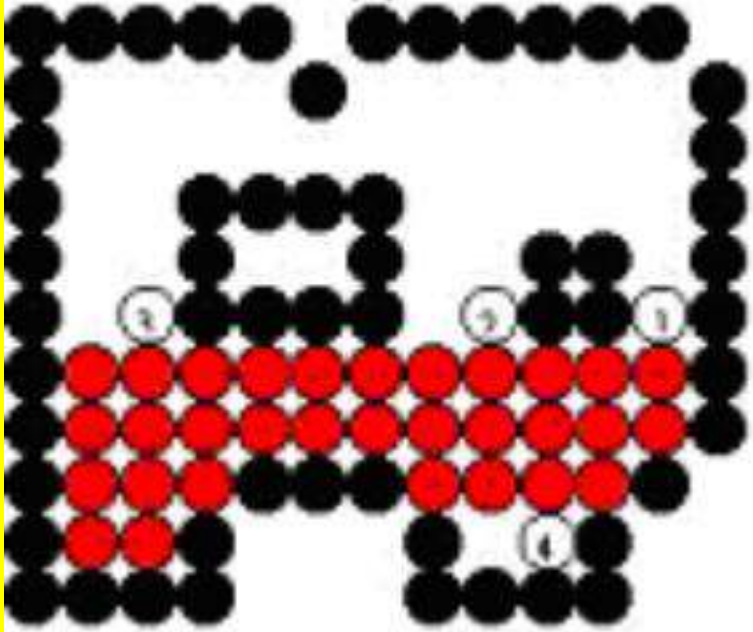

3
2
1
4
(1)
3
2
1
4
4
(2)



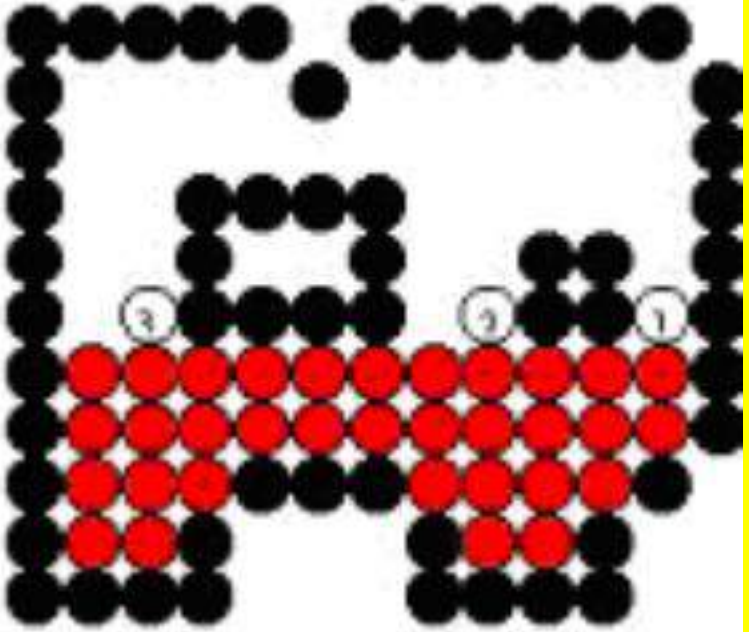
(3)



(4)



(5)



(6)

二. 多边形的扫描转换算法

“奇偶”性质：即一条直线与任意封闭的曲线相交时，总是从第一个交点进入内部，再从第二个交点退出，以下交替的进入退出，即奇数次进入，偶数次退出。当然可能有一些“相切”的点应特殊处理。

扫描转换算法步骤：

step1. 找出扫描线与多边形边界线的所有交点；

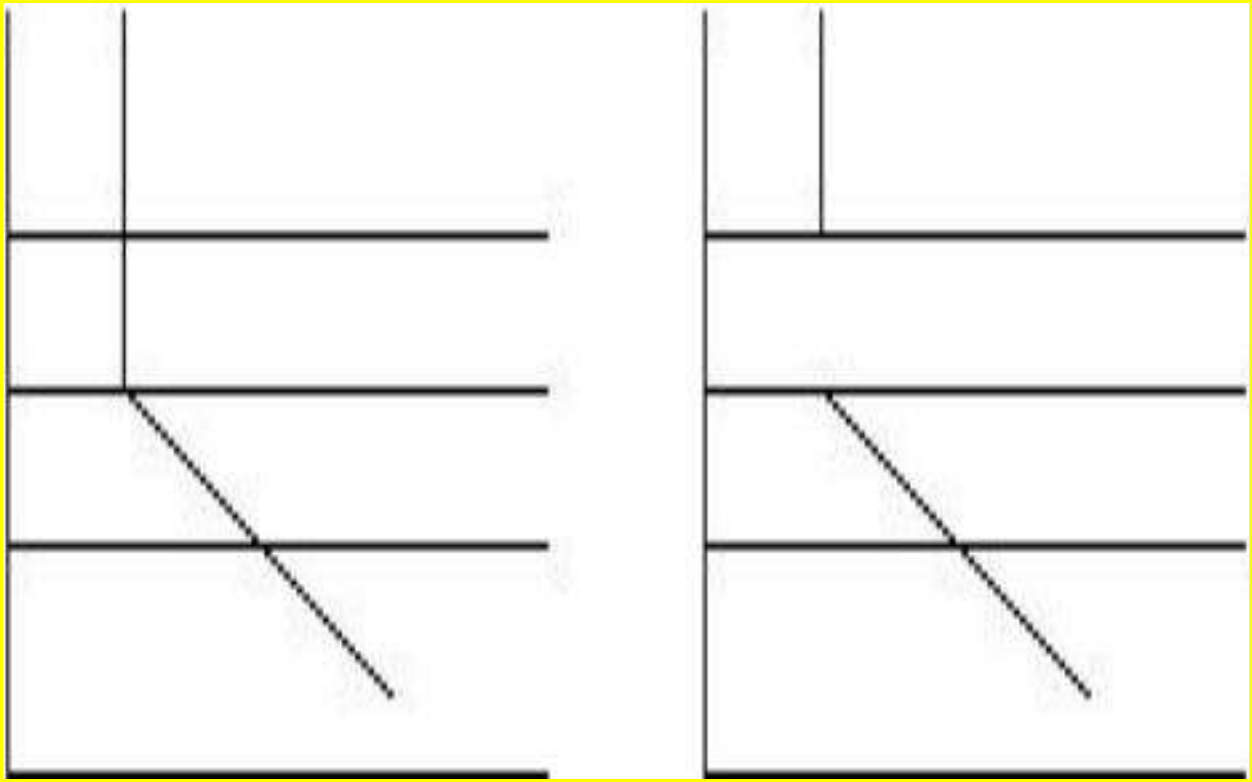
step2. 按x坐标增加顺序对交点排序；

step3. 在交点对之间进行填充。（偶数）

奇异顶点：

局部极大或局部极小点，交点看做是二个

非局部极值点，交点看做一个



对非极值点的简便处理

step1．如何计算扫描线与多边形边界线的所有交点？

若扫描线y_i与多边形边界线交点x的坐标是x_i，则对下一条扫描线y_{i+1}，它与那条边界线的交点的x坐标x_{i+1}，可如下求出：

$$m = \frac{y_{i+1} - y_i}{x_{i+1} - x_i},\ y_{i+1} = y_i + 1 \Rightarrow x_{i+1} = x_i + \frac{1}{m}$$

活跃边：与当前扫描线相交的边

活跃边表AET：存贮当前扫描线相交的各边的表。

y_{max}	x	1/m	next

y_{max}:当前这条边的上端点的y坐标

x:　当前边与扫描线交点的x坐标

1/m:当前边的斜率的倒数

next：指向下一条边的指针

边表ET：记录多边形的所有边

“吊桶”：记录一条边的信息

y_{max}	x_{min}	1/m	next

y_{max}： 边的另一端点的较大的y坐标

x_{min}： 与较小的y坐标对应的边的端点的x坐标

1/m： 斜率的倒数

next： 指向下一条边的指针

Polygonfill(EdgeTableET,COLORREFcolor)

{

1.y=置y为边表ET中各登记项对应的y 坐标中最小的值;

2.活跃边表AET初始化为空表;

3.若AET非空或ET非空，则做下列假设，否则算法结束

{

3.1. 将ET中登记项y对应的各“吊桶”合并到表AET

中,将AET中各吊桶按x坐标递增排序;

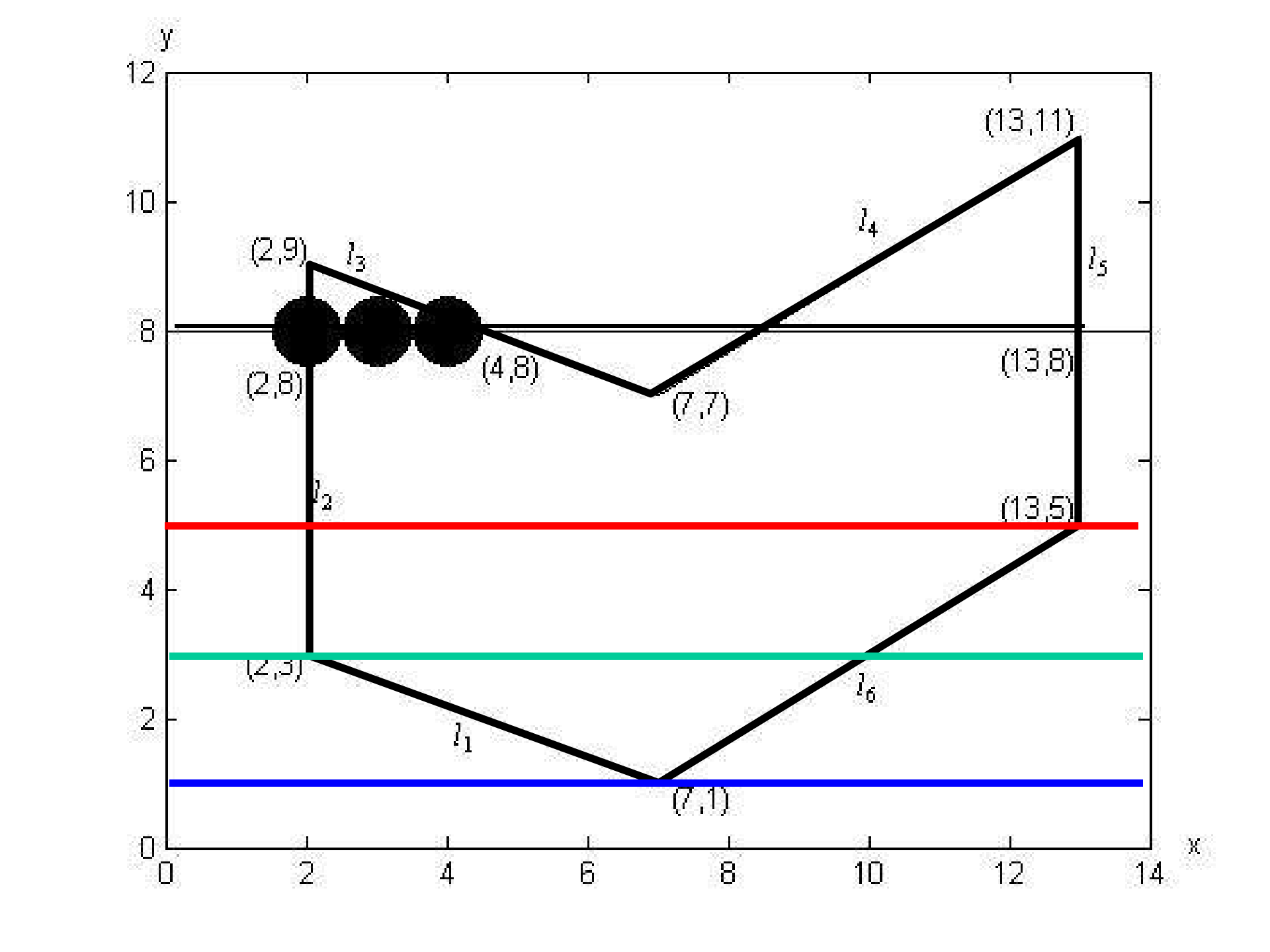
3.2. 在扫描线y上，按照AET表提供的x坐标对，用 color实施填充;

3.3. 将AET表中有$y=y_{max}$的各项清除出表;

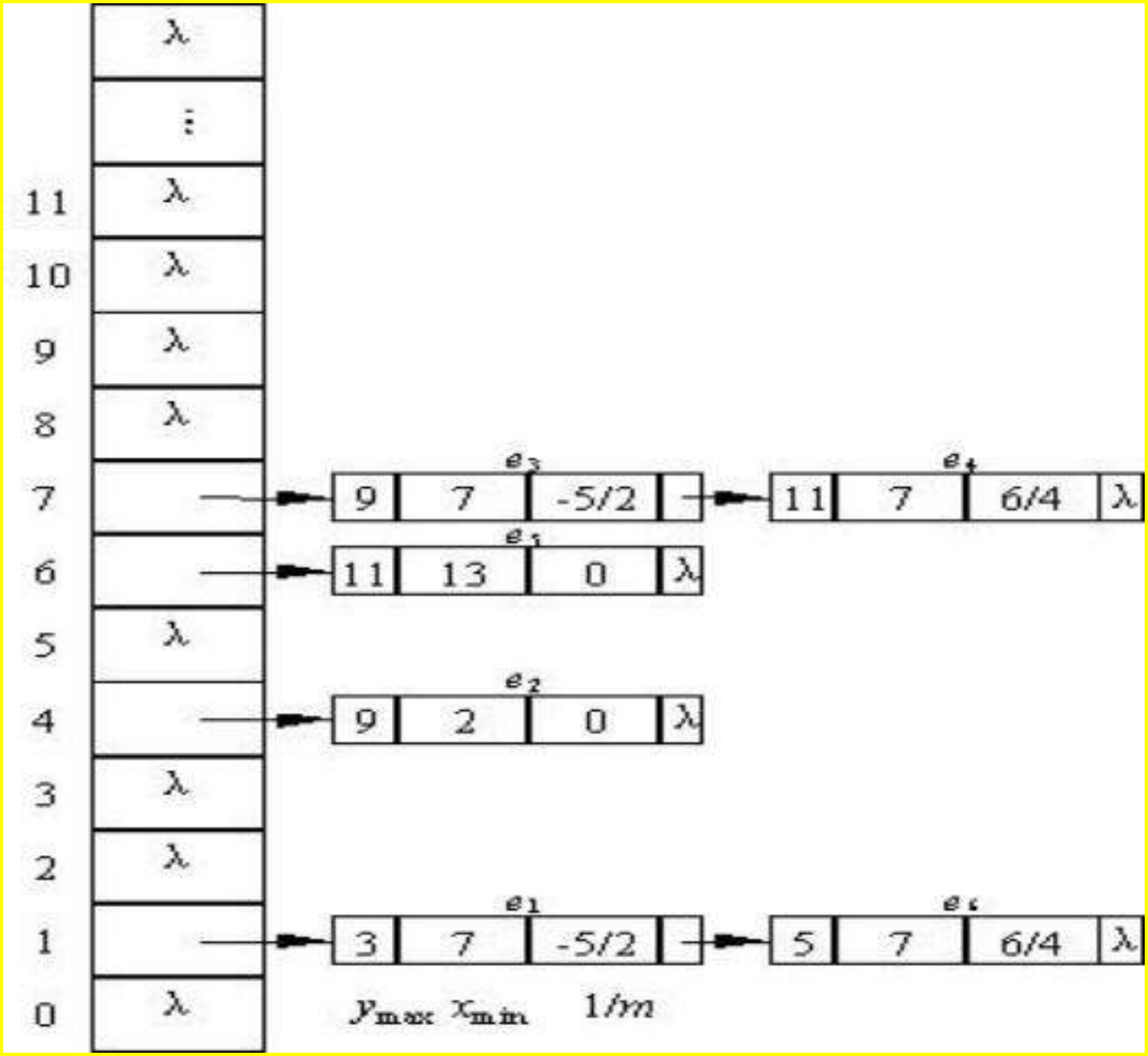
3.4. 对AET中留下的各项，分别将x换为x+1/m，求出AET中各边与下一条扫描线交点的x坐标;

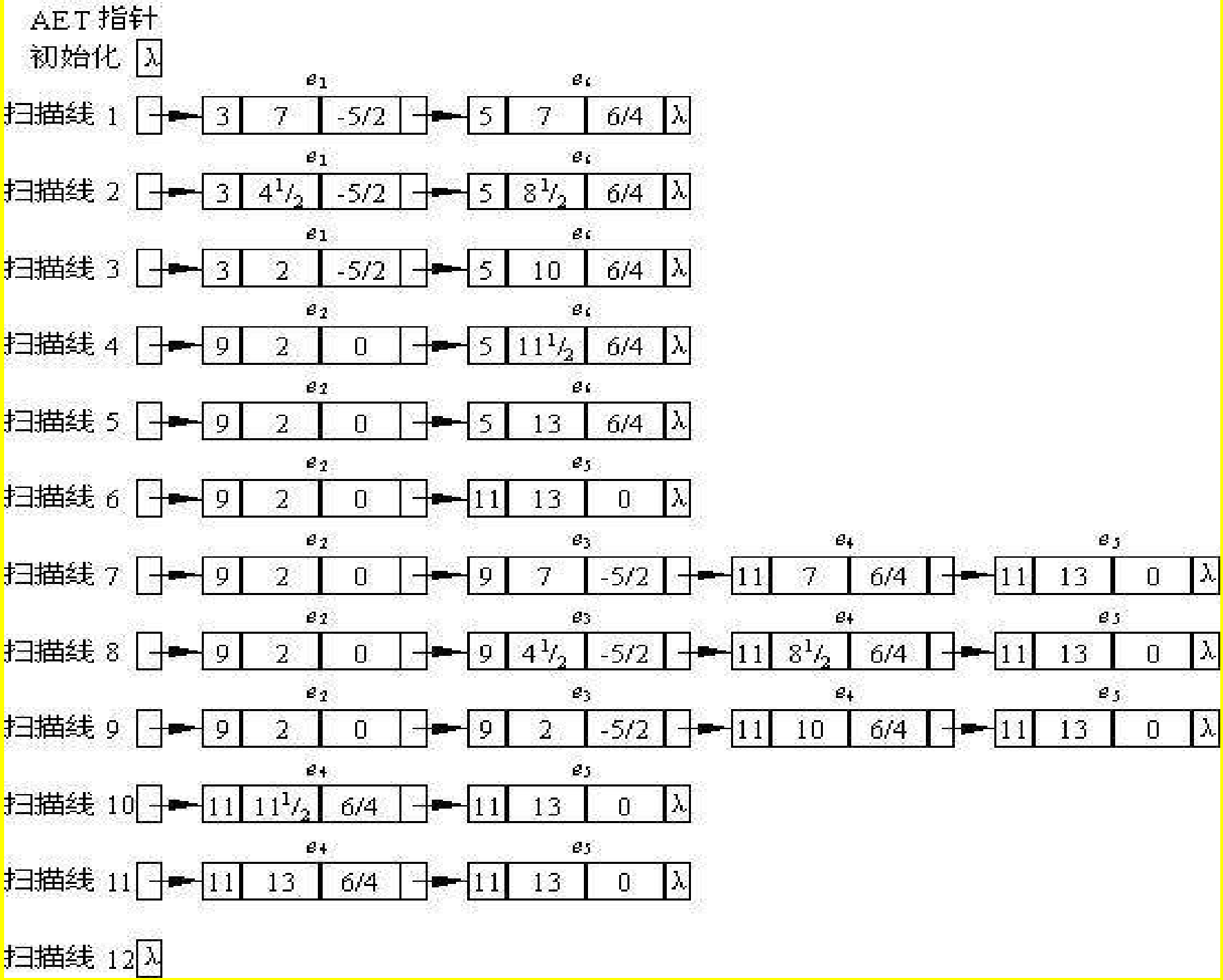
3.5 y=y＋1，返回步骤3处理下一条扫描线。

}


y
x
0
2
4
6
8
10
12
14
(13,11)
(2,9)
(2,8)
(4,8)
(7,7)
(13,8)
(13,5)
(2,3)
(7,1)
l_1
l_2
l_3
l_4
l_5
l_6

ET：


λ
⋮
11
λ
10
λ
9
λ
8
λ
7
6
5
λ
4
3
λ
2
λ
1
0
λ
e_3
9
7
-5/2
e_4
11
7
6/4
λ
e_5
11
13
0
λ
e_2
9
2
0
λ
e_1
3
7
-5/2
e_6
5
7
6/4
λ
y_{max} x_{min} $1/m$


AET指针
初始化 λ
扫描线 1
e1
3 7 -5/2
e4
5 7 6/4 λ
扫描线 2
e1
3 4 1/2 -5/2
e4
5 8 1/2 6/4 λ
扫描线 3
e1
3 2 -5/2
e4
5 10 6/4 λ
扫描线 4
e2
9 2 0
e4
5 11 1/2 6/4 λ
扫描线 5
e2
9 2 0
e4
5 13 6/4 λ
扫描线 6
e2
9 2 0
e5
11 13 0 λ
扫描线 7
e2
9 2 0
e3
9 7 -5/2
e4
11 7 6/4
e5
11 13 0 λ
扫描线 8
e2
9 2 0
e3
9 4 1/2 -5/2
e4
11 8 1/2 6/4
e5
11 13 0 λ
扫描线 9
e2
9 2 0
e3
9 2 -5/2
e4
11 10 6/4
e5
11 13 0 λ
扫描线 10
e4
11 11 1/2 6/4
e5
11 13 0 λ
扫描线 11
e4
11 13 6/4
e5
11 13 0 λ
扫描线 12 λ

第二章　作业

$P_{26\text{-}3}$方法一：分几种情况来讨论

(1) 若$|m|<1$,则若$x_1 > x_2$,则$x_1 \leftrightarrow x_2, y_1 \leftrightarrow y_2$

若$|m|>1$,则若$y_1 > y_2$,则$x_1 \leftrightarrow x_2, y_1 \leftrightarrow y_2$

(2) 处理特殊情况(水平垂直线)

若$x_1 = x_2$,则$y_s = \min(y_1, y_2), y_e = \max(y_1, y_2)$

$for(y = y_s; y <= y_e; y++)$垂直线段

$setpixel(x, y)$

若$y_1 = y_2$,则$x_s = \min(x_1, x_2), x_e = \max(x_1, x_2)$

$for(x = x_s; x <= x_e; x++)$水平线段

$setpixel(x, y)$

(3) $-1 < m < 0$

$(x_i, y_i) \to (x_i + 1, y_i)$或者 $(x_i + 1, y_i - 1)$

$|d_1| = |y - y_i| = y_i - y = y_i - (mx_{i+1} + b)$

$|d_2| = y - (y_i - 1) = mx_{i+1} + b - (y_i - 1)$

$|d_1| - |d_2| = -2mx_{i+1} - 2b + 2y_i - 1$

$p_i = \Delta x(|d_1| - |d_2|) = -2\Delta y x_i + 2\Delta x y_i - 2\Delta y - 2b\Delta x - \Delta x$

若$p_i \geq 0$，下一个点应取 $(x_i + 1, y_i - 1)$

若$p_i < 0$，下一个点应取 $(x_i + 1, y_i)$

$p_{i+1} - p_i = -2\Delta y(x_{i+1} - x_i) + 2\Delta x(y_{i+1} - y_i)$

若$p_i \geq 0$, $p_{i+1} = p_i - 2\Delta y - 2\Delta x$

若$p_i < 0$, $p_{i+1} = p_i - 2\Delta y$

$\because -1 < m < 0, m = \dfrac{\Delta y}{\Delta x}, \Delta y < 0$

$p_i \geq 0$, $p_{i+1} = p_i + 2|\Delta y| - 2|\Delta x|$, 取$y_i - 1$

若$p_i < 0$, $p_{i+1} = p_i + 2|\Delta y|$，取 y_i

[准备]: $dx \leftarrow |x_2 - x_1|, dy \leftarrow |y_2 - y_1|, p \leftarrow 2dy - dx$

若$y_1 < y_2$,则$s \leftarrow 1(0 < m < 1)$;否则$s \leftarrow -1(-1 < m < 0)$

2.3.若$p \geq 0$,则$y \leftarrow y + s, p \leftarrow p + 2(dy - dx)$

否则$p \leftarrow p + 2dy$

(4)

$|m| > 1$,具体看 $m > 1$

$$y = mx + b, x = \frac{1}{m}y - \frac{b}{m} \Rightarrow x = m'y + b'$$

$\therefore 0 < m' < 1$,与$y = mx + b$相比只是 $x \leftrightarrow y, m' = \frac{1}{m}$

$\therefore$,可用书上的算法，只是 x, y位置互换， $\Delta x, \Delta y$的位置互换

具体算法如下：

1.[准备] 最开始的(1) (2)

$x \leftarrow x_1, y \leftarrow y_1, dx \leftarrow |x_2 - x_1|, dy \leftarrow |y_2 - y_1|$

若 $m > 0$, 则 $s = 1$, 否则 $s = -1$

若 $dy > dx, dx \leftrightarrow dy, t = 1(|m| > 1)$, 否则 $t = 0(|m| < 1)$

$p \leftarrow 2dy - dx, i \leftarrow 1$

2.[逐点画线] 若i>dx,则算法结束，否则做以下各步

2.1 setpixel(x,y),i←i+1

2.2 若t=1(|m|>1),则y ←y+1,否则x ←x+1 (|m|<1),

2.3 若p>=0 则若t=1 (|m|>1),则x ←x+s, p ←p+2(dy-dx)

若t=0 (|m|<1),则y ←y+s, p ←p+2(dy-dx)

否则 p ←p+2dy

2.4返回2步开头

方法二. 利用第三章图形变换

把除0<m<1外的三种情况变换到0<m<1的情况，求得要画点后再变换回原来的位置

(1) $m > 1$时

$(x_1, y_1) \rightarrow (y_1, x_1) = (x_1', y_1')$

$(x_2, y_2) \rightarrow (y_2, x_2) = (x_2', y_2')$

$(y', x') \leftarrow (x', y')$

(2) $-1 < m < 0$时，做关于 x 轴的对称变换

$(x_1, y_1) \rightarrow (x_1, -y_1) = (x_1', y_1')$

$(x_2, y_2) \rightarrow (x_2, -y_2) = (x_2', y_2')$

$(x', -y') \leftarrow (x', y')$

(3) $m < -1$做两个变换　　a)关于 y 轴对称　　b)关于 $y = x$ 对称

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

$(x, y, 1) \rightarrow (y, -x, 1)$

$(x_1, y_1) \rightarrow (y_1, -x_1) = (x_1', y_1')$

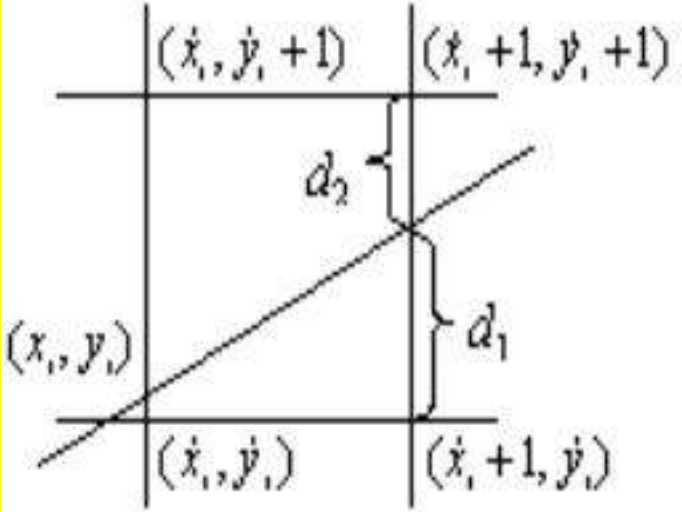
$(x_2, y_2) \rightarrow (y_2, -x_2) = (x_2', y_2')$

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

$(x, y, 1) \rightarrow (-y, x, 1)$

$(-y', x') \leftarrow (x', y')$

P26-4



$$d_1 = y - y_i = m(x_i + 1) + b - y_i$$

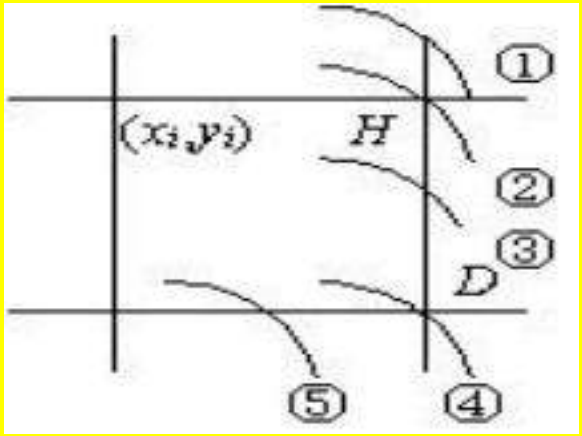
$$d_2 = (y_i + 1) - y = (y + 1) - m(x_i + 1) - b$$

（1）d1<0,d2>0， p=d1-d2<0选择（xi+1,yi)

（2） d1<0,d2>0 , p=d1-d2>0选择（xi+1,yi+1)

$P_{26\text{-}6}$结合Bresenham画圆算法与行扫描填充算法

$P_{26\text{-}7}$



$$d_H = (x_i + 1)^2 + y_i^2 - R^2$$

$$d_D = R^2 - (x_i + 1)^2 - (y_i - 1)^2$$

$$p_i = d_H - d_D$$

当$P \geq 0$时，应选D点

$$p_{i+1} = p_i + 4(x_i - y_i) + 10$$

$$\frac{p_{i+1} - 1}{2} = \frac{p_i - 1 + 4(x_i - y_i) + 10}{2}, \text{令} e_i = \frac{p_{i+1} - 1}{2}, e_{i+1} = e_i + 2(x_i - y_i) + 1 + 4$$

$$\text{令} v_i = 2(x_i - y_i) + 1, v_{i+1} = 2(x_{i+1} - y_{i+1}) + 1 = 2(x_i - y_i) + 1 + 4 = v_i + 4$$

当*P*<0时，应选H点

$$p_{i+1} = p_i + 4x_i + 6$$

$$\frac{p_{i+1}-1}{2} = \frac{p_i - 1 + 4x_i + 6}{2}, \text{令} e_i = \frac{p_{i+1}-1}{2}, e_{i+1} = e_i + 2x_i + 1 + 2$$

$$\text{令} u_i = 2x_i + 1, \therefore e_{i+1} = e_i + u_i + 2$$

$$u_{i+1} = 2x_{i+1} + 1 = 2x_i + 1 + 2 = u_i + 2$$

$$v_{i+1} = v_i + 2;$$

$$p \geq 0\text{时}, \text{令} e = \frac{p-1}{2}, e \geq 0\text{时} v_{i+1} = v_i + 4, u_{i+1} = u_i + 2$$

$$p < 0\text{时},, e < 0\text{时} v_{i+1} = v_i + 2, u_{i+1} = u_i + 2$$

$$p = 3 - 2R, e = \frac{p-1}{2} = 1 - R, p = 2e + 1$$

$$x_0 = 0, y_0 = R, u_0 = 2x_0 + 1, v_0 = 2(x_0 - y_0) + 1 = 1 - 2R$$

$P_{26\text{-}7}$填写ET表

输入：多边形的逆时针顶点序列

输出:ET表

算法，设输入顶点序列为$P_0, P_1, \ldots P_n$的n+1个点

1.[初始化]ET各项置为空，$P_{n+1} \leftarrow P_0$

$S \leftarrow P_0, T \leftarrow P_1, Q \leftarrow P_n, U \leftarrow P_2, i \leftarrow 1$

2.[循环]若i>n+1,则算法结束，否则做以下各步

2.1[处理水平边和非极值点]

若S.y=T.y,记录下i, goto 2.3//$P_{i-1}P_i$是水平边

若$Q.y \le S.y < T.y$

则$S.x \leftarrow S.x + \dfrac{T.x - S.x}{T.y - S.y}, S.y \leftarrow S.y + 1$

若$S.y > T.y \ge U.y$,则$T.x \leftarrow T.x + \dfrac{T.x - S.x}{T.y - S.y}, T.y \leftarrow T.y + 1$

2.2[填吊桶]

若$S.y > T.y$,则$y_{\max} \leftarrow S.y, y_{\min} \leftarrow T.y, x_{\min} \leftarrow T.x$

否则$y_{\max} \leftarrow T.y, y_{\min} \leftarrow S.y, x_{\min} \leftarrow S.x$

$m \leftarrow (T.x - S.x)/(T.y - S.y)$

将吊桶按$x_{\min}$递增的次序插入ET的$y_{\min}$项中

2.3[更新]

$$i \leftarrow i+1, Q \leftarrow S, S \leftarrow T, T \leftarrow U, U \leftarrow P_{i+1} goto 2$$

$P_{27\text{-}8}$：水平边单独处理，先取整再写ET,AET

第三章 图形变换

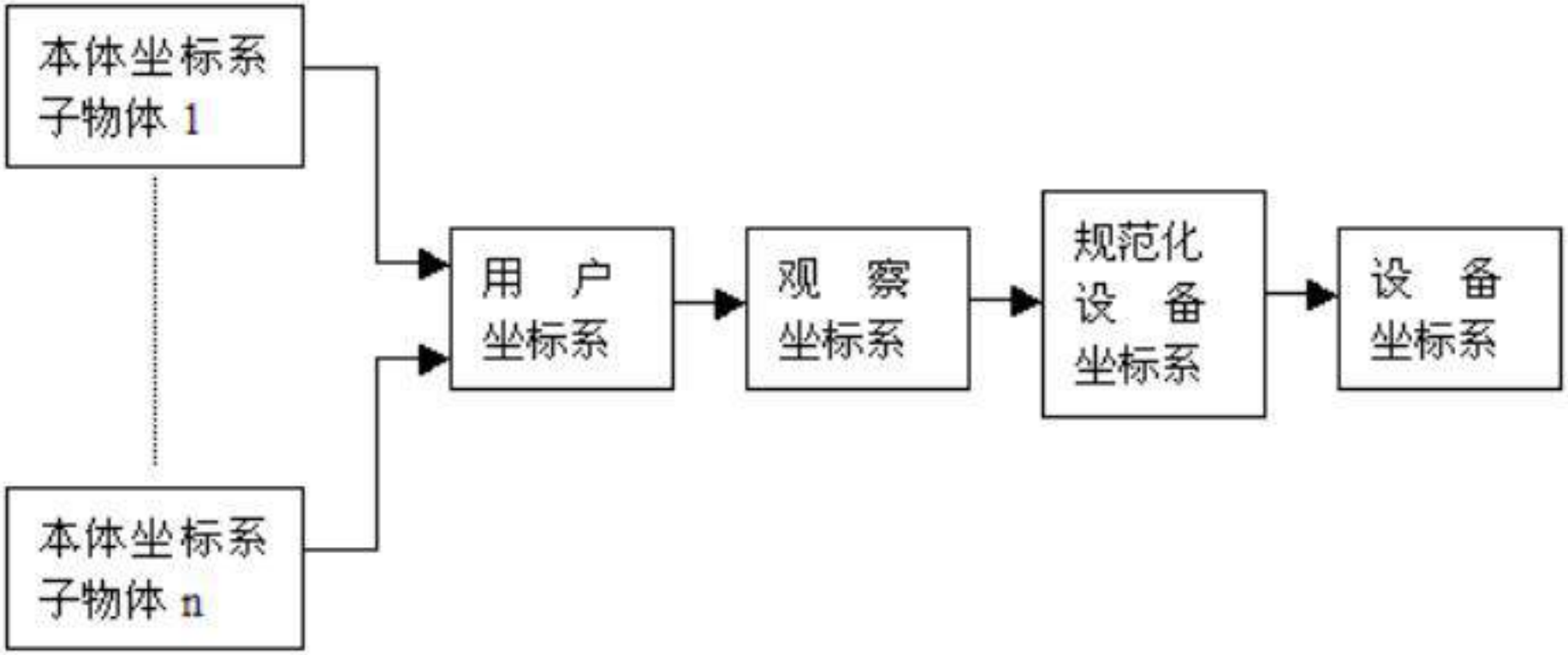
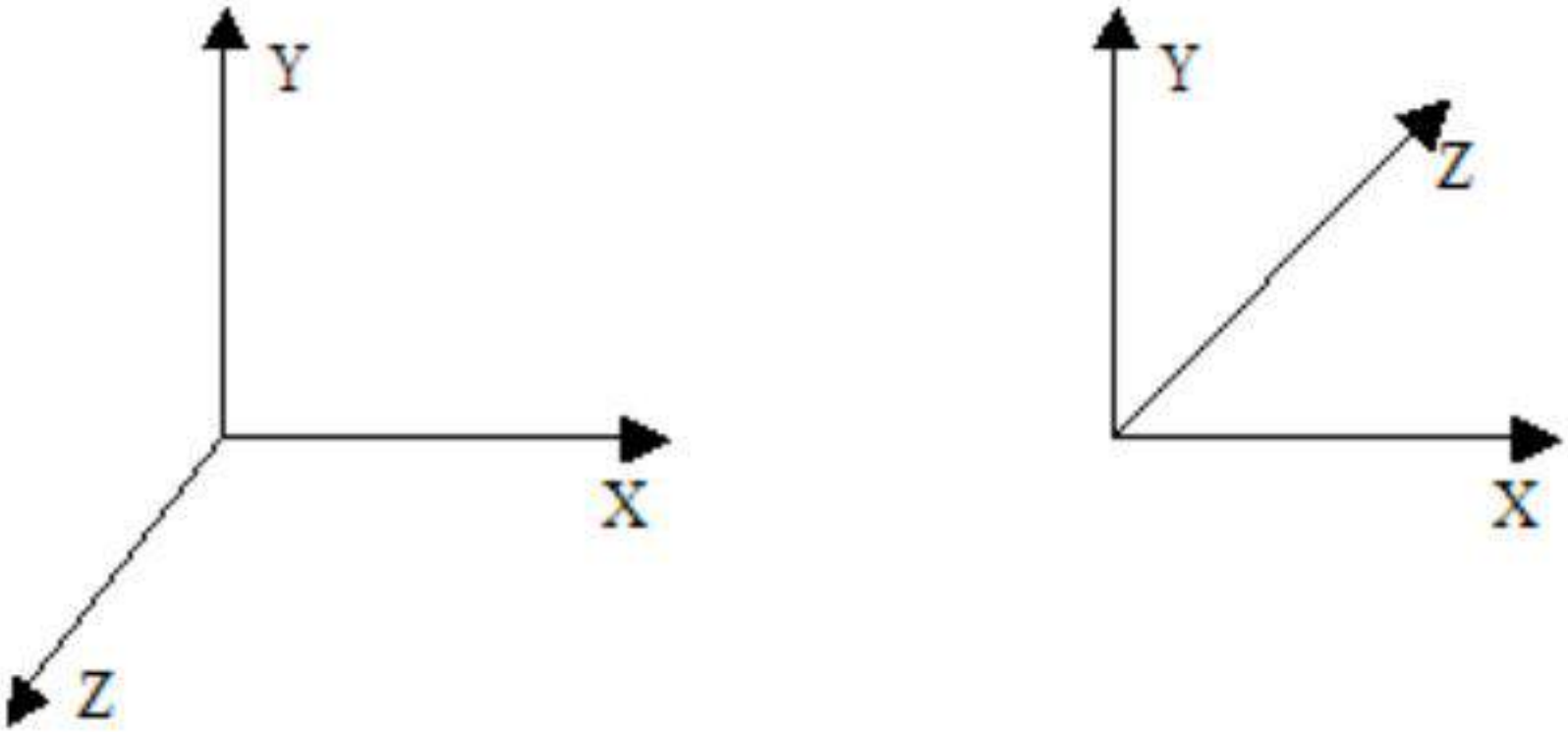


图 3.1 图形显示的坐标变换过程



(1) 右手系　　(2) 左手系

图 3.2　左手系和右手系

第一节　变换的数学基础

一、向量及向量运算

设有向量，$a(x_1,y_1,z_1)$ $b(x_2,y_2,z_2)$

有关的向量运算有：

1）向量的长度

$$|a|=\sqrt{a\bullet a}=\sqrt{x_1x_1+y_1y_1+z_1z_1}$$

2）两个向量的和差运算

$$a\pm b=(x_1\pm x_2,y_1\pm y_2,z_1\pm z_2)$$

3） 两个向量的点乘积

$$a \bullet b = x_1 x_2 + y_1 y_2 + z_1 z_2$$

4） 两个向量的叉乘积

$$a \times b = \begin{vmatrix} i & j & k \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix}$$

$$= (y_1 z_2 - y_2 z_1, z_1 x_2 - z_2 x_1, x_1 y_2 - x_2 y_1)$$

二、矩阵及矩阵运算

由$m\times n$个$a_{ij}(i=1,2,\cdots,m;\ j=1,2,\cdots,n)$数排成的矩形表

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

有关的矩阵运算

1.数乘矩阵

$$tA=\begin{bmatrix} ta_{11} & ta_{12} & \cdots & ta_{1n} \\ ta_{21} & ta_{22} & \cdots & ta_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ ta_{m1} & ta_{m2} & \cdots & ta_{mn} \end{bmatrix}$$

2. 矩阵的加法运算

$$A+B=\begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \cdots & a_{mn}+b_{mn} \end{bmatrix}$$

3. 矩阵的乘法运算

设有矩阵 $A=(a_{ij})_{2\times 3}$、$B=(b_{ij})_{3\times 2}$，则这两个矩阵的乘积矩阵 C 为：

$$C = A \bullet B = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}b_{11}+a_{12}b_{21}+a_{13}b_{31} & a_{11}b_{12}+a_{12}b_{22}+a_{13}b_{32} \\ a_{21}b_{11}+a_{22}b_{21}+a_{23}b_{31} & a_{21}b_{12}+a_{22}b_{22}+a_{23}b_{32} \end{bmatrix}$$

$$C=(C_{ij})_{m\times p}=A_{m\times n}\bullet B_{n\times p}$$

矩阵运算具有如下基本性质：

数乘矩阵适合分配律和结合律，即：

$$t(A+B)=tA+tB$$

$$t(A\bullet B)=(t\bullet A)\bullet B=A\bullet(t\bullet B)$$

$$(t_1+t_2)A=t_1\bullet A+t_2\bullet A$$

$$t_1(t_2A)=(t_1t_2)A$$

矩阵的加法适合交换律和结合律，

其中A ，B ，C 为矩阵。

$$A+B=B+A$$

$$A+(B+C)=(A+B)+C$$

矩阵的乘法适合结合律，即：

$$A\bullet(B\bullet C)=(A\bullet B)\bullet C$$

矩阵的乘法对加法适合分配律，即：

$$(A+B)\bullet C=A\bullet C+B\bullet C$$

$$C\bullet(A+B)=C\bullet A+C\bullet B$$

矩阵的乘法不适合交换律

4. 单位矩阵

$$I=\begin{bmatrix}1 & 0 & \cdots & 0\\ 0 & 1 & \cdots & 0\\ \vdots & \ddots & \ddots & \vdots\\ 0 & 0 & \cdots & 1\end{bmatrix}$$

5. 逆矩阵

对任意矩阵，如果存在 $A\bullet A^{-1}=A^{-1}\bullet A=I$，则称 A^{-1} 为的逆矩阵。

6. 转置矩阵 将矩阵 $A=(a_{ij})_{m\times n}$ 的行、列互换而得到的 $n\times m$ 阶矩阵称作的转置矩阵，记为 A^T。

矩阵的转置具有如下几个基本性质：

$$(A^T)^T = A$$

$$(A+B)^T = A^T + B^T$$

$$(tA)^T = tA^T$$

$$(A \bullet B)^T = B^T \bullet A^T$$

三、齐次坐标

齐次坐标表示法就是用n+1维向量表示一个n维向量。

n维空间中的点的位置向量用非齐次坐标表示时，$(P_1,P_2,\cdots,P_n)$具有n个坐标分量，并且是唯一的。如果用齐次坐标表示时，该向量有n+1个坐标分量$(hP_1,hP_2,\cdots,hP_n,h)$并且是不唯一的。

如二维点(x,y)的齐次坐标表示为$[hx,hy,h]$，则$[h_1x,h_1y,h_1]$，$[h_2x,h_2y,h_2]$…，$[h_mx,h_my,h_m]$都是表示二维空间中的同一个点.

三维空间中的坐标点的齐次坐标可表示为$[hx,hy,hz,h]$.

应用齐次坐标可以有效地用矩阵运算把二维、三维甚至更高维空间中点集从一个坐标系转换到另一个坐标系中。

$$T_{2D}=\begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} \quad T_{3D}=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

第二节 二维图形变换

对一个图形作几何变换就是对该图形上的每一个点作相应的几何变换。

常见的基本二维图形几何变换有**平移变换、比例变换和旋转变换**。

平移变换

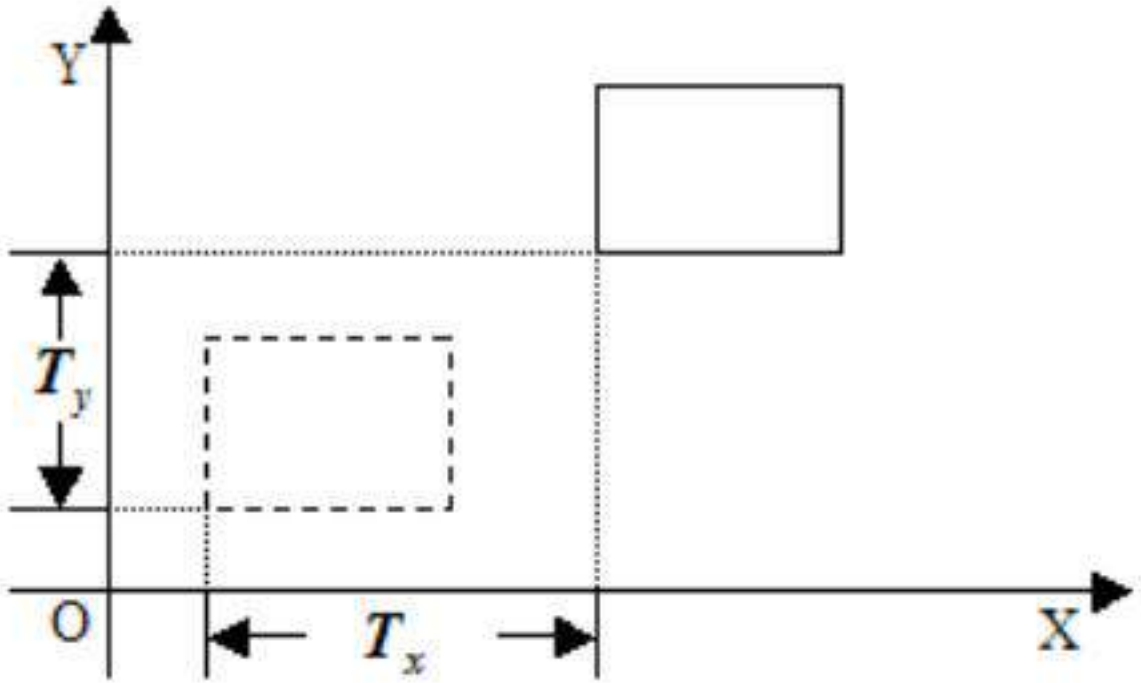


图 3.3 平移变换

(x, y)

(x', y')

$$x' = x + T_x, \quad y' = y + T_y$$

比例变换

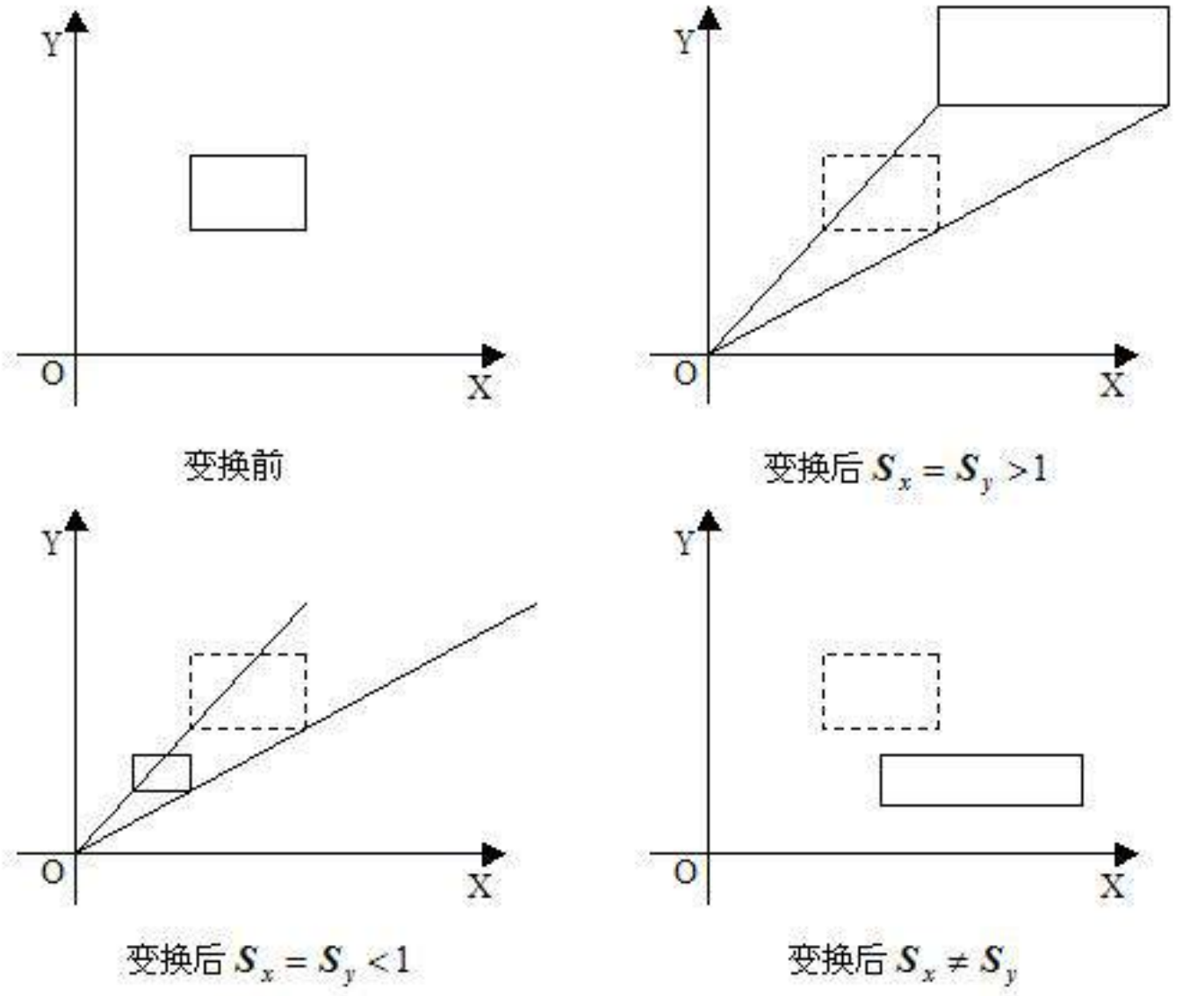


图 3.4　比例变换

$$x' = x \bullet S_x$$

$$y' = y \bullet S_y$$

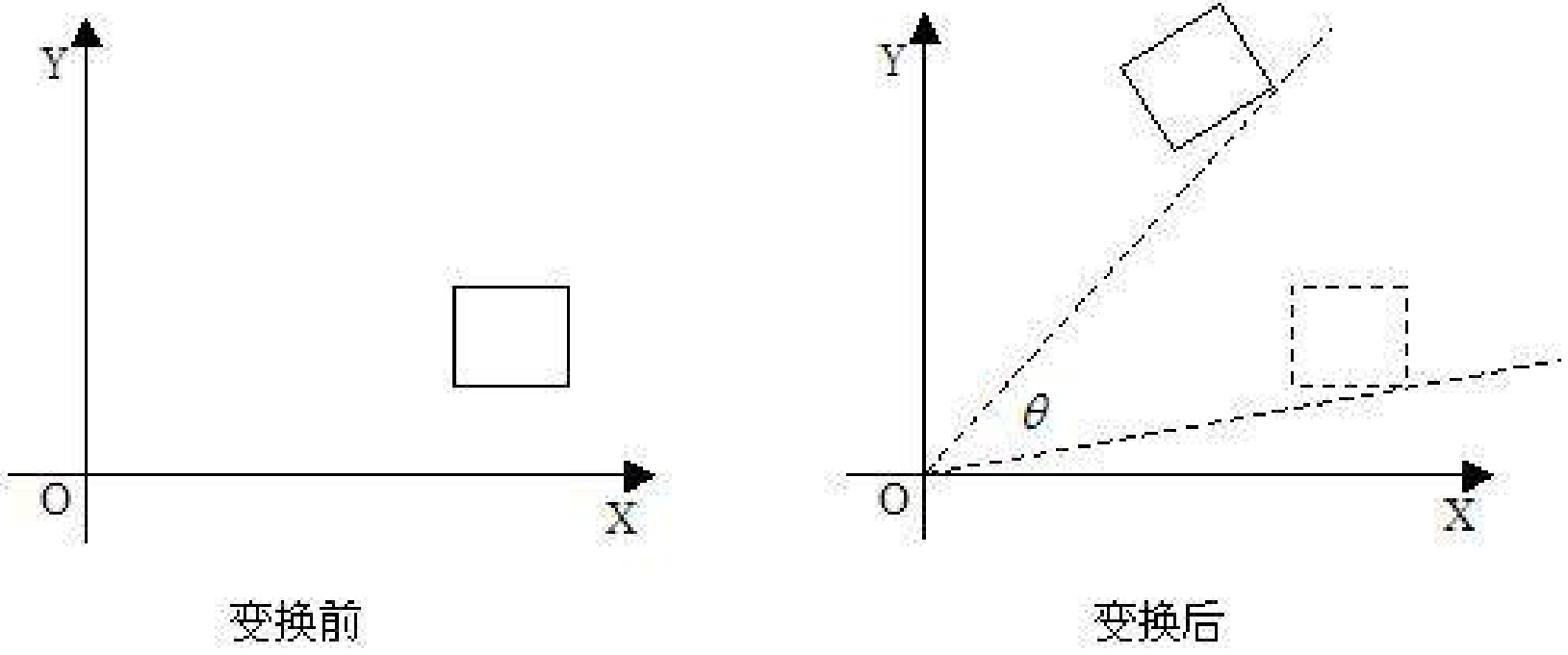


图 3.5 旋转变换

旋转变换

$$x' = x\cos\theta - y\sin\theta$$

$$y' = x\sin\theta + y\cos\theta$$

齐次坐标系：用n+1维向量表示n维向量的方法

设有不全为0的三个数（x1, x2, x3）为齐次坐标

当$x_3 \neq 0$时，普通坐标为$(x_1/x_3, x_2/x_3)$

当$x_3 = 0$时，$(x_1, x_2, 0)$表示平面上通过普通坐标（0,0）和(x_1, x_2)的直线上无穷远点

$(x, y) \leftrightarrow (hx, hy, h)$其中$h \neq 0$,

$h = 1$的齐次坐标称为规范化齐次坐标

$(x, y, 1) \leftrightarrow (x, y)$

设平面上有齐次坐标为P=(x,y,1)的一点，分别经过平移、比例和旋转变换为齐次坐标为P’=(x’,y’,1)的一点，求变换矩阵

1. 平移变换

$$p'=\begin{bmatrix}x' & y' & 1\end{bmatrix}=\begin{bmatrix}x & y & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ T_x & T_y & 1\end{bmatrix}$$

$$=\begin{bmatrix}x+T_x & y+T_y & 1\end{bmatrix}=p\bullet T(T_x,T_y)$$

2. 比例变换

$$p'=\begin{bmatrix}x' & y' & 1\end{bmatrix}=\begin{bmatrix}x & y & 1\end{bmatrix}\begin{bmatrix}S_x & 0 & 0\\ 0 & S_y & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$=\begin{bmatrix}x\bullet S_x & y\bullet S_y & 1\end{bmatrix}$$

$$=p\bullet S(S_x,S_y)$$

3. 旋转变换

$$p'=\begin{bmatrix}x' & y' & 1\end{bmatrix}=\begin{bmatrix}x & y & 1\end{bmatrix}\begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$=\begin{bmatrix}x\bullet\cos\theta-y\bullet\sin\theta & x\bullet\sin\theta+y\bullet\cos\theta & 1\end{bmatrix}$$

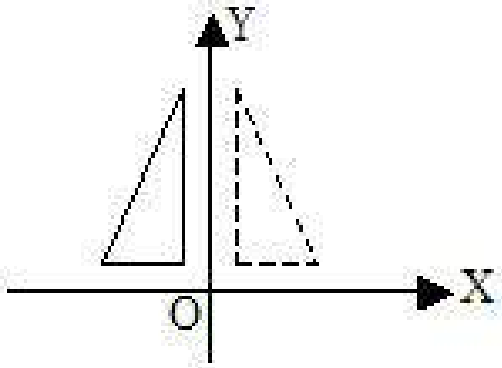
$$=p\bullet R(\theta)$$

常见的变换

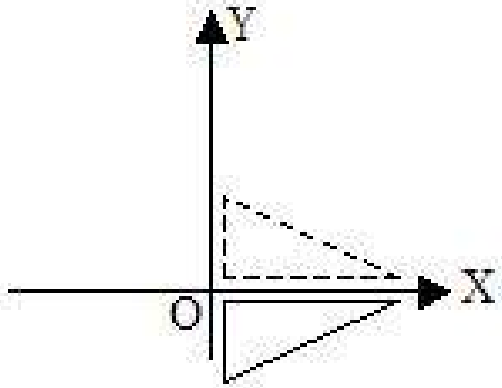
1. 对称变换(反射、镜像变换)

$$p'=\begin{bmatrix}x' & y' & 1\end{bmatrix}=\begin{bmatrix}x & y & 1\end{bmatrix}\begin{bmatrix}a & d & 0\\ b & e & 0\\ 0 & 0 & 1\end{bmatrix}$$

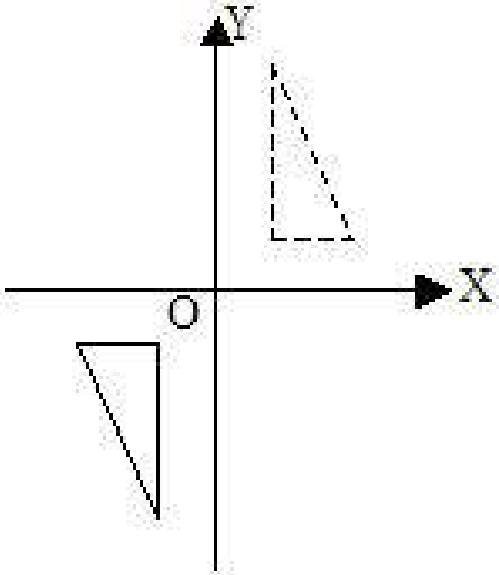
$$=\begin{bmatrix}ax+by & dx+ey & 1\end{bmatrix}$$



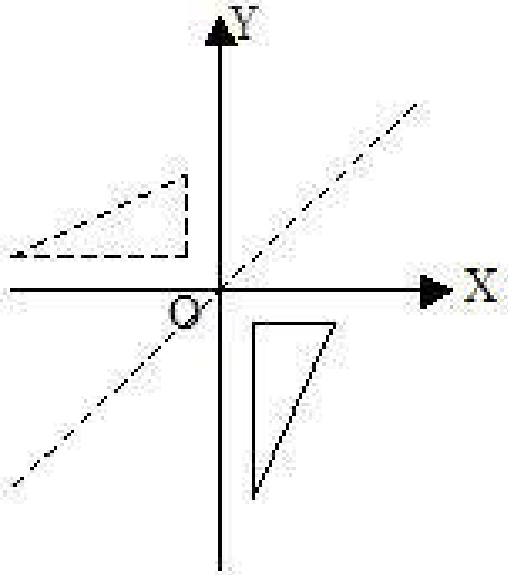
Y 轴对称



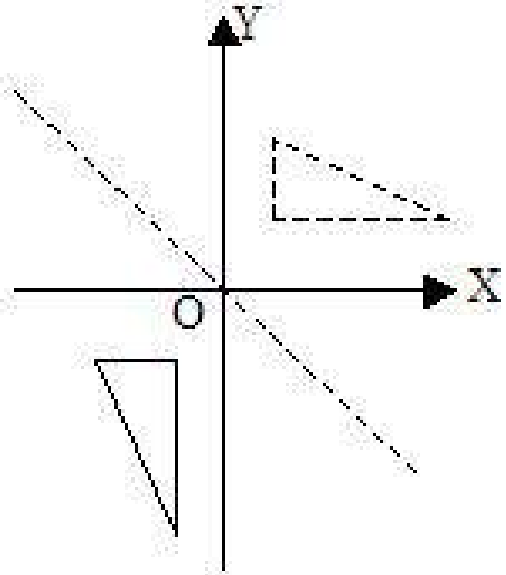
X 轴对称



中心对称



$Y=X$ 对称



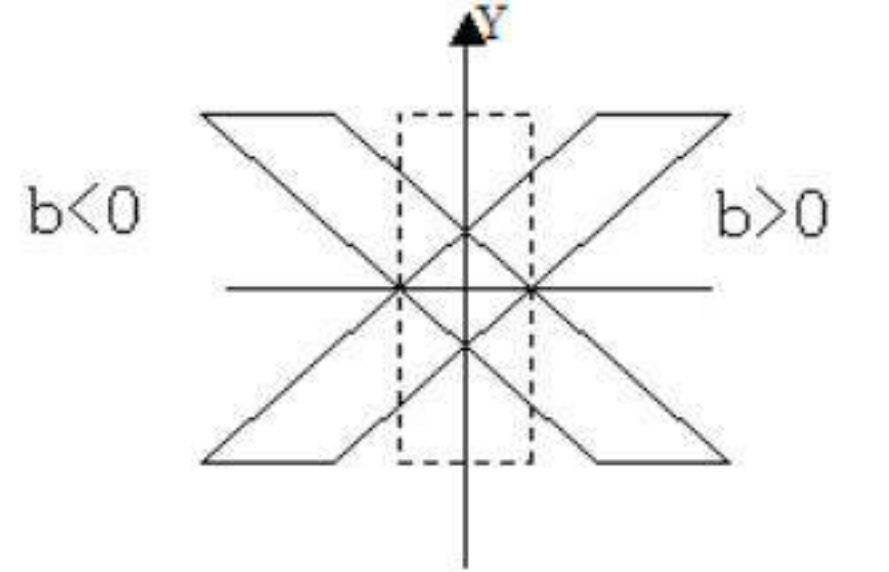
$Y=-X$ 对称

图 3.6　对称变换

2. 错切变换(错移变换)

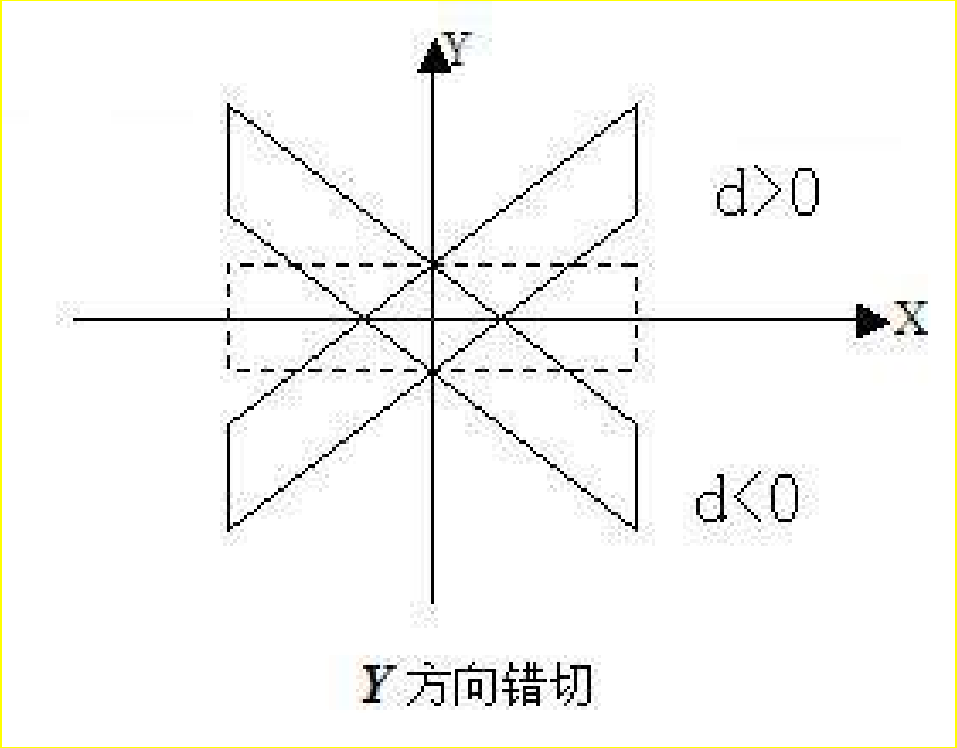
（1）沿X轴方向关于Y的错切（Y不变）

$$p'=\begin{bmatrix}x' & y' & 1\end{bmatrix}=\begin{bmatrix}x & y & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ b & 1 & 0\\ 0 & 0 & 1\end{bmatrix}=\begin{pmatrix}x+by & y & 1\end{pmatrix}$$



（2）沿Y轴方向关于X的错切（X不变）

$$p'=\begin{bmatrix}x' & y' & 1\end{bmatrix}=\begin{bmatrix}x & y & 1\end{bmatrix}\begin{bmatrix}1 & d & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}=\begin{pmatrix}x & y+dx & 1\end{pmatrix}$$

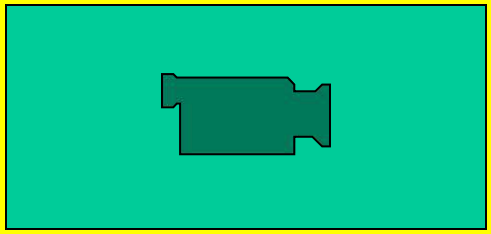


Y 方向错切

（3）沿X轴Y轴两个方向的错切

$$p'=\begin{bmatrix}x' & y' & 1\end{bmatrix}=\begin{bmatrix}x & y & 1\end{bmatrix}\begin{bmatrix}1 & d & 0\\ b & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$=\begin{bmatrix}x+by & dx+y & 1\end{bmatrix}$$



例一：求图形绕平面上任意一点（x0,y0)的旋转变换矩阵，设旋转角度为 θ

(1)平移T(-x0,-y0),使旋转中心与原点重合

(2)绕原点逆时针旋转R(θ)

(3)平移T(x0,y0),使旋转中心移回原处

$$H = T(-x_0, y_0) \bullet R(\theta) \bullet T(x_0, y_0)$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_0 & -y_0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_0 & y_0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ x_0(1-\cos\theta)+y_0\sin\theta & y_0(1-\cos\theta)+x_0\sin\theta & 1 \end{bmatrix}$$

$$x' = x_0 + (x - x_0)\cos\theta - (y - y_0)\sin\theta$$

$$y' = y_0 + (x - x_0)\sin\theta + (y - y_0)\cos\theta$$

例二：相对于平面上任意一点(x_0,y_0)的比例变换如下：

(1)平移T(-x0,-y0),使比例中心与原点重合

(2)相对于原点的比例变换 $S(S_x,S_y)$

(3)平移T(x0,y0),比例中心移回原处

$$H = T(-x_0,-y_0) \bullet S(S_x,S_y) \bullet T(x_0,y_0)$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_0 & -y_0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_0 & y_0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ x_0(1-S_x) & y_0(1-S_y) & 1 \end{bmatrix}$$

$$x' = x_0 + (x - x_0)s_x$$

$$y' = y_0 + (y - y_0)s_x$$

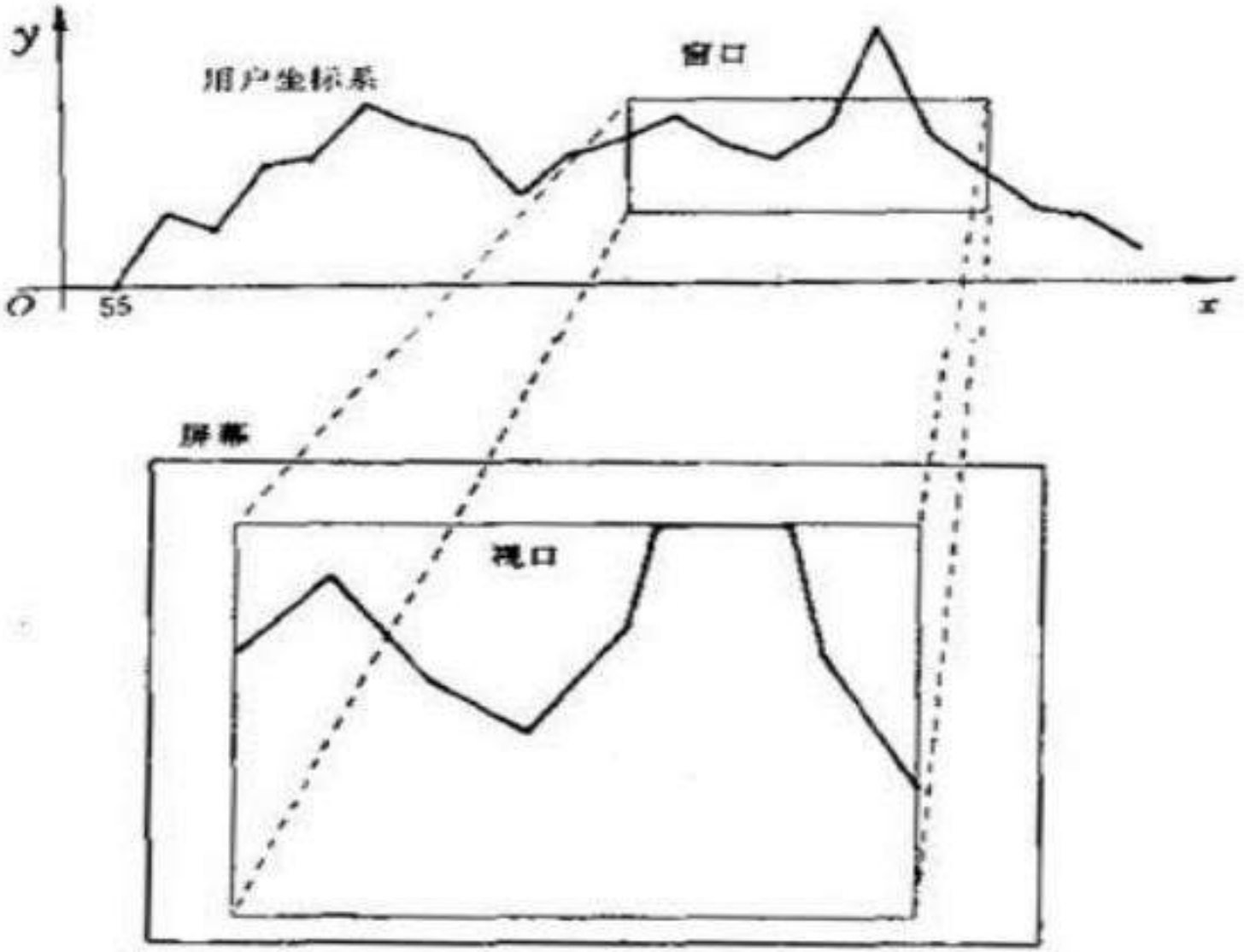
二维图形变换总结：

$$T_{2D}=\left[\begin{array}{cc|c} a & d & g \\ b & e & h \\ \hline c & f & i \end{array}\right]$$

第三节 二维视见变换

窗口就是在用户坐标系中指出的那个要显示出来的区域，这一区域通常为矩形区域。

视见区是屏幕域中的一个子区域，通常为矩形区域，它最大与屏幕域等同。



窗口到视区

窗口与视见区的差别：

窗口是在用户坐标系中确定的，它指出了要显示的图形；

视见区在设备坐标系中确定，它指出了实际显示的图形处于显示屏幕的哪一部分。

视见区用于显示窗口中的图形。

视见变换就是将用户坐标系窗口中指定的图形转换至设备坐标系视区中显示的过程。

视见变换的**实现**：

窗口：左下角点（wxl, wyl）右上角点（wxh, wyh）

视区：左下角点（vxl, vyl）右上角点（vxh, vyh）

1. $$\frac{x_w - wxl}{wxh - wxl} = \frac{x_v - vxl}{vxh - vxl}$$

$$x_v = vxl + (x_w - wxl)\frac{vxh - vxl}{wxh - wxl}$$

同理 $$y_v = vyl + (y_w - wyl)\frac{vyh - vyl}{wyh - wyl}$$

2.另一种处理方式，求一个变换矩阵

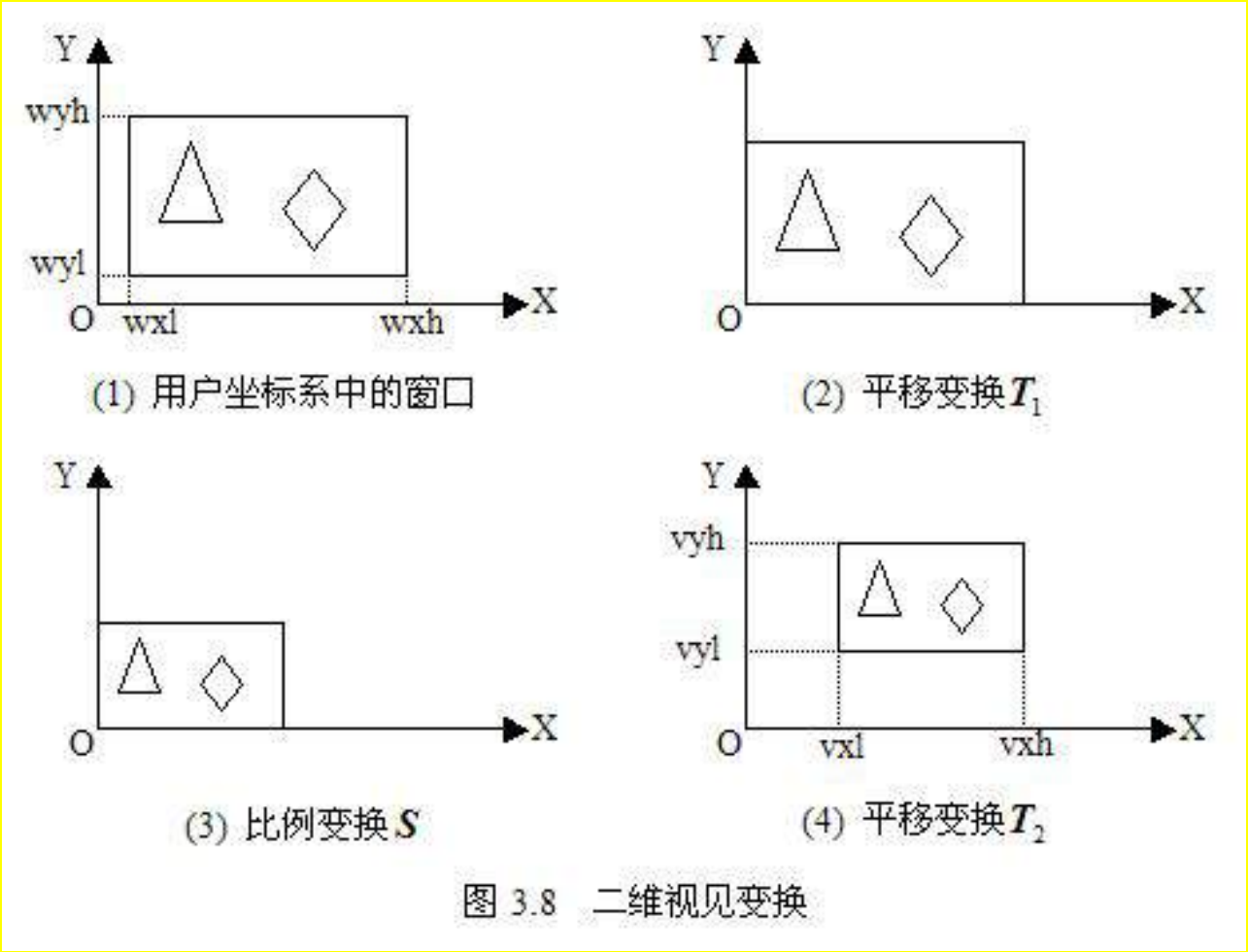


图 3.8　二维视见变换

$$H = T_1(-wxl,-wyl) \bullet S(s_x,s_y) \bullet T_2(vxl,vyl)$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -wxl & -wyl & 1 \end{bmatrix} \begin{bmatrix} \frac{vxh-vxl}{wxh-wxl} & 0 & 0 \\ 0 & \frac{vyh-vyl}{wyh-wyl} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ vxl & vyl & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{vxh-vxl}{wxh-wxl} & 0 & 0 \\ 0 & \frac{vyh-vyl}{wyh-wyl} & 0 \\ vxl-wxl\frac{vxh-vxl}{wxh-wxl} & vyl-wyl\frac{vyh-vyl}{wyh-wyl} & 1 \end{bmatrix}$$

$$s_x = \frac{vxh-vxl}{wxh-wxl}, s_{y=} \frac{vyh-vyl}{wyh-wyl}$$

设窗口中图形上的某一点坐标为(x,y)，该点显示在视见区中的坐标为(x',y')，利用视见变换矩阵可得出以下计算公式：

$$x' = vxl + (x - wxl)\frac{vxh - vxl}{wxh - wxl}$$

$$y' = vyl + (y - wyl)\frac{vyh - vyl}{wyh - wyl}$$

第四节 三维图形变换

$$P=\begin{bmatrix}x & y & z & 1\end{bmatrix} \quad P'=\begin{bmatrix}x' & y' & z' & 1\end{bmatrix}$$

1. 平移变换

$$P'=\begin{bmatrix}x' & y' & z' & 1\end{bmatrix}=\begin{bmatrix}x & y & z & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ D_x & D_y & D_z & 1\end{bmatrix}$$

$$=P\bullet T(D_x,D_y,D_z)$$

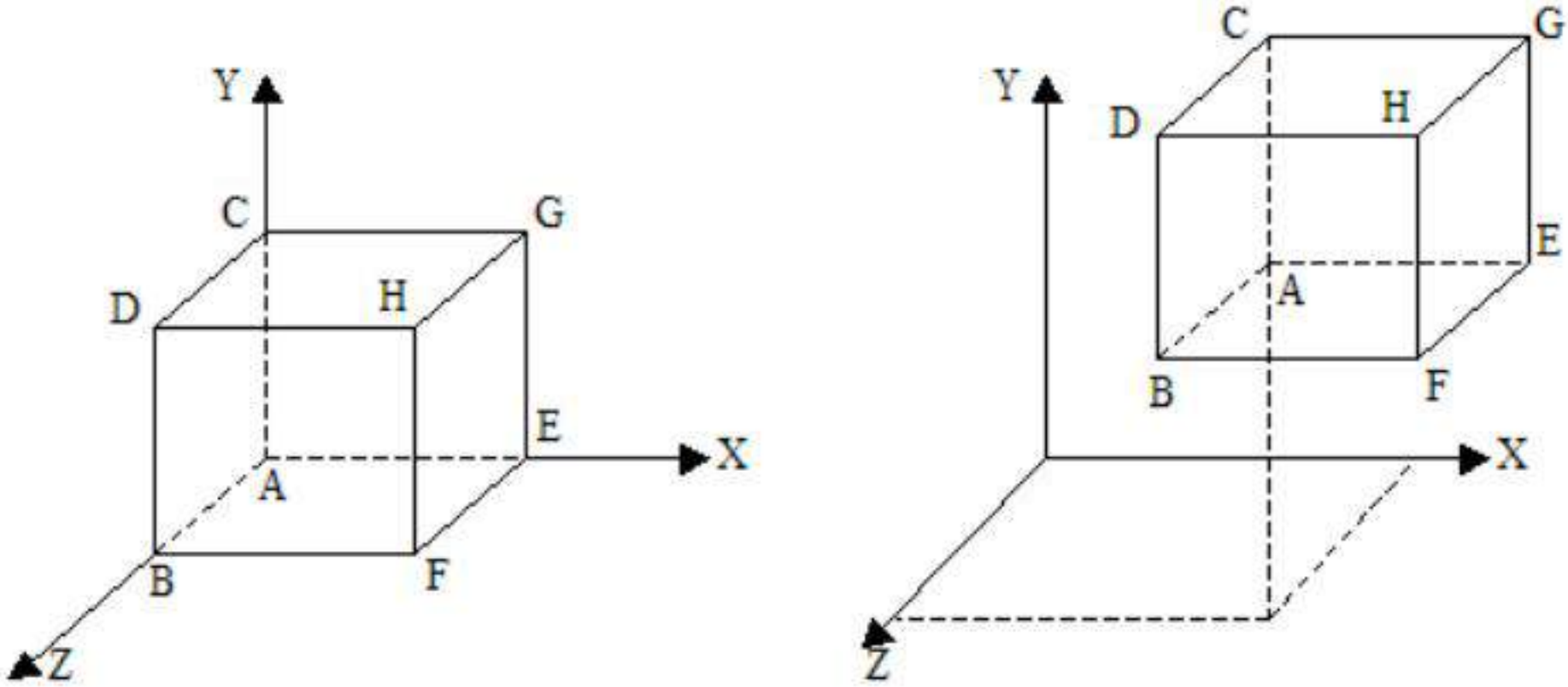


图 3.9　三维平移变换

2. 比例变换

$$P'=\begin{bmatrix}x' & y' & z' & 1\end{bmatrix}=\begin{bmatrix}x & y & z & 1\end{bmatrix}\begin{bmatrix}S_x & 0 & 0 & 0\\ 0 & S_y & 0 & 0\\ 0 & 0 & S_z & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$=P\bullet S(S_x,S_y,S_z)$$

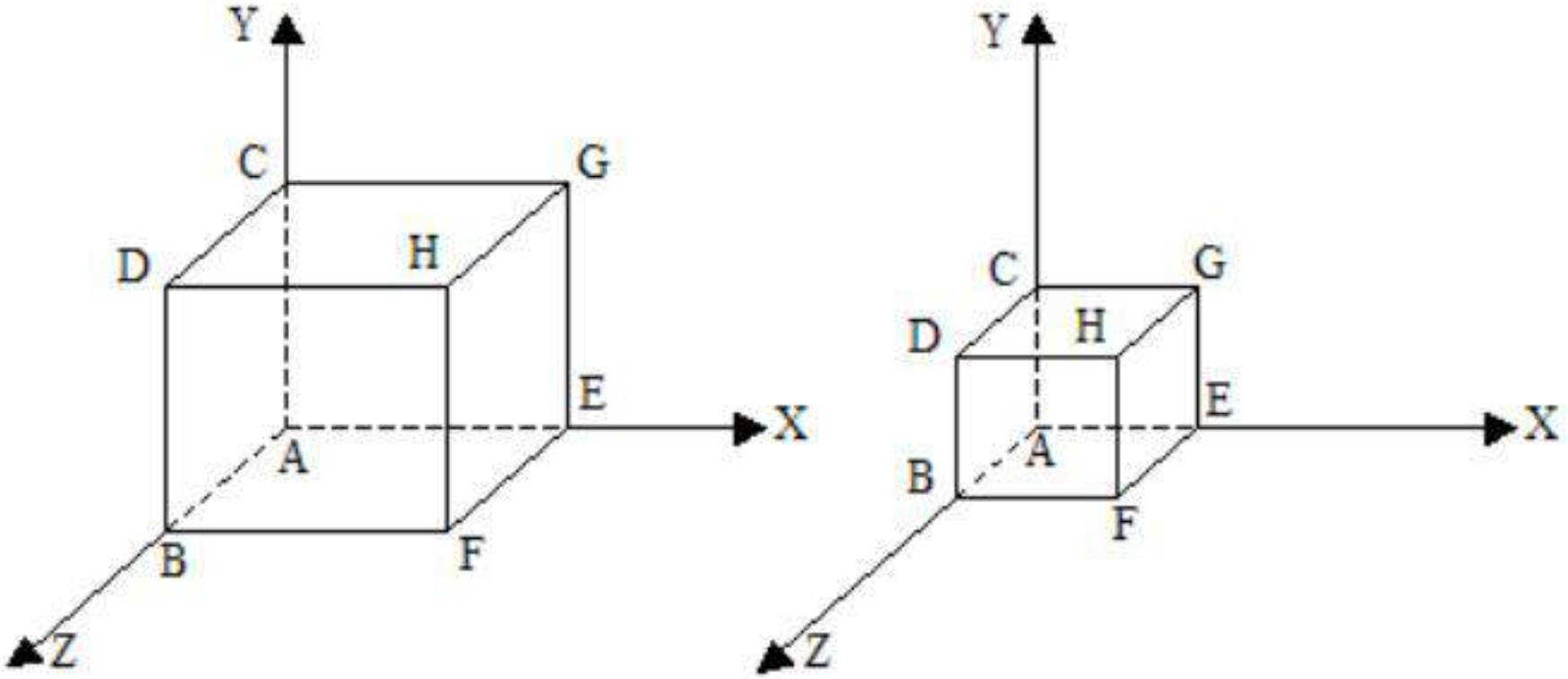
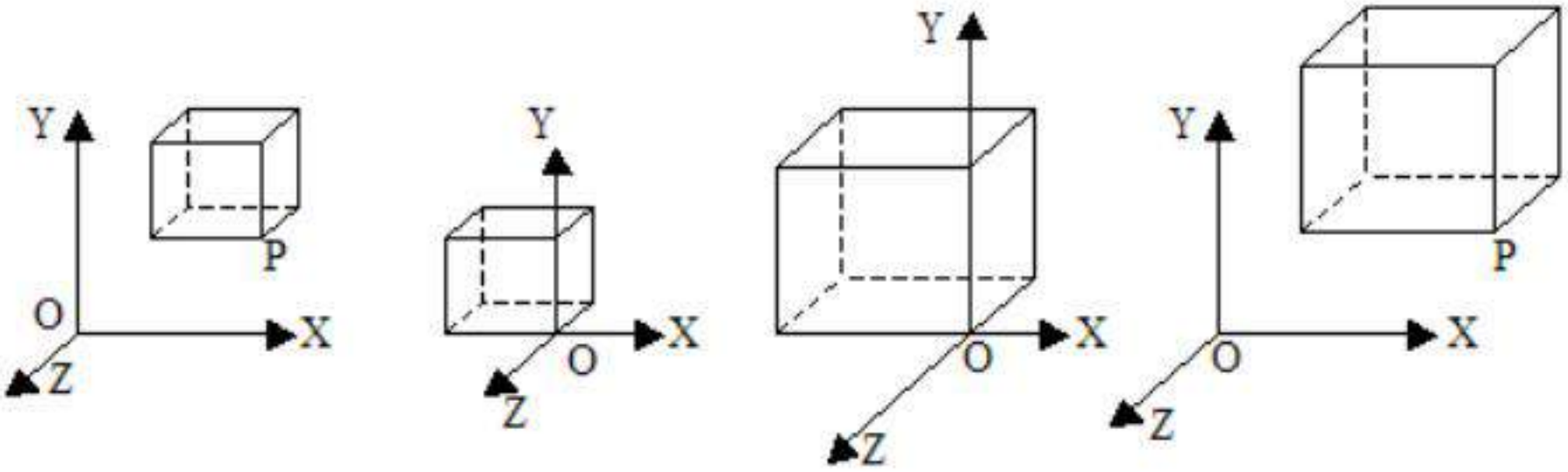


图 3.10　以坐标原点为参考点的三维比例变换



(a) (b) 平移至原点 (c) 比例变换 (d) 平移回 ***P*** 点

图 3.11 以任意点 ***P*** 为参考点的三维比例变换过程

3. 旋转变换

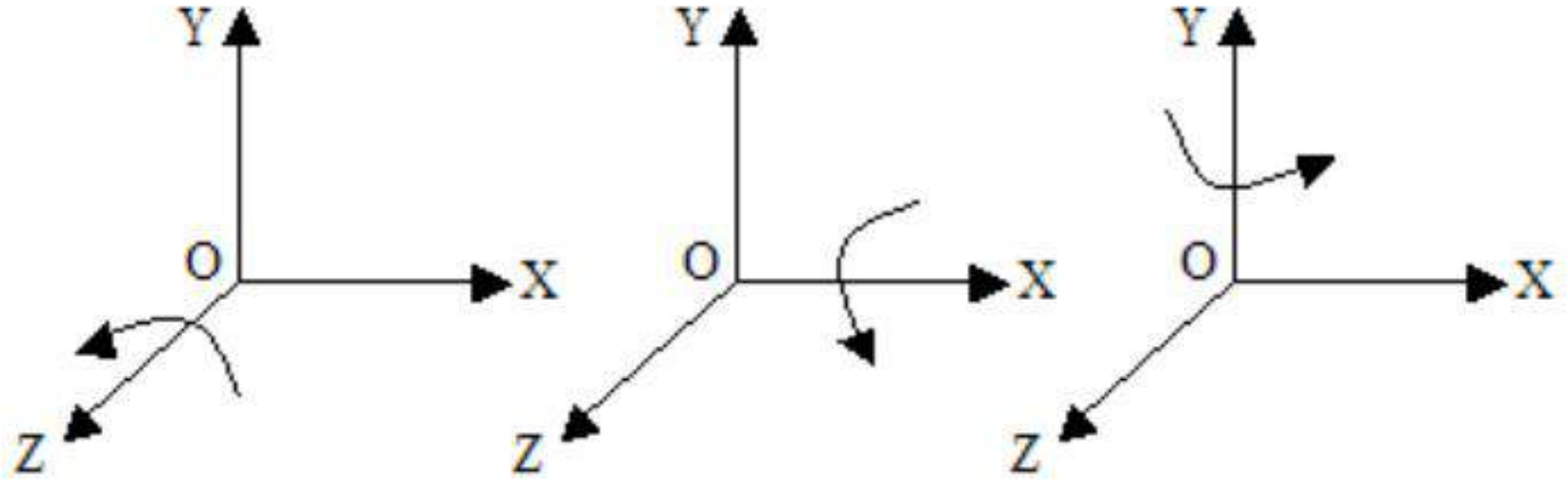


图 3.12　绕三根坐标轴旋转的正方向

① 绕Z轴旋转

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$
$$z' = z$$

$$P' = [x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= P \bullet R_Z(\theta)$$

②绕X轴旋转

$$x' = x$$

$$y' = y\cos\theta - z\sin\theta$$

$$z' = y\sin\theta + z\cos\theta$$

$$P' = \begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= P \bullet R_x(\theta)$$

③绕Y轴旋转

$$x' = x\cos\theta + z\sin\theta$$

$$y' = y$$

$$z' = -x\sin\theta + z\cos\theta$$

$$P' = [x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= P \bullet R_y(\theta)$$

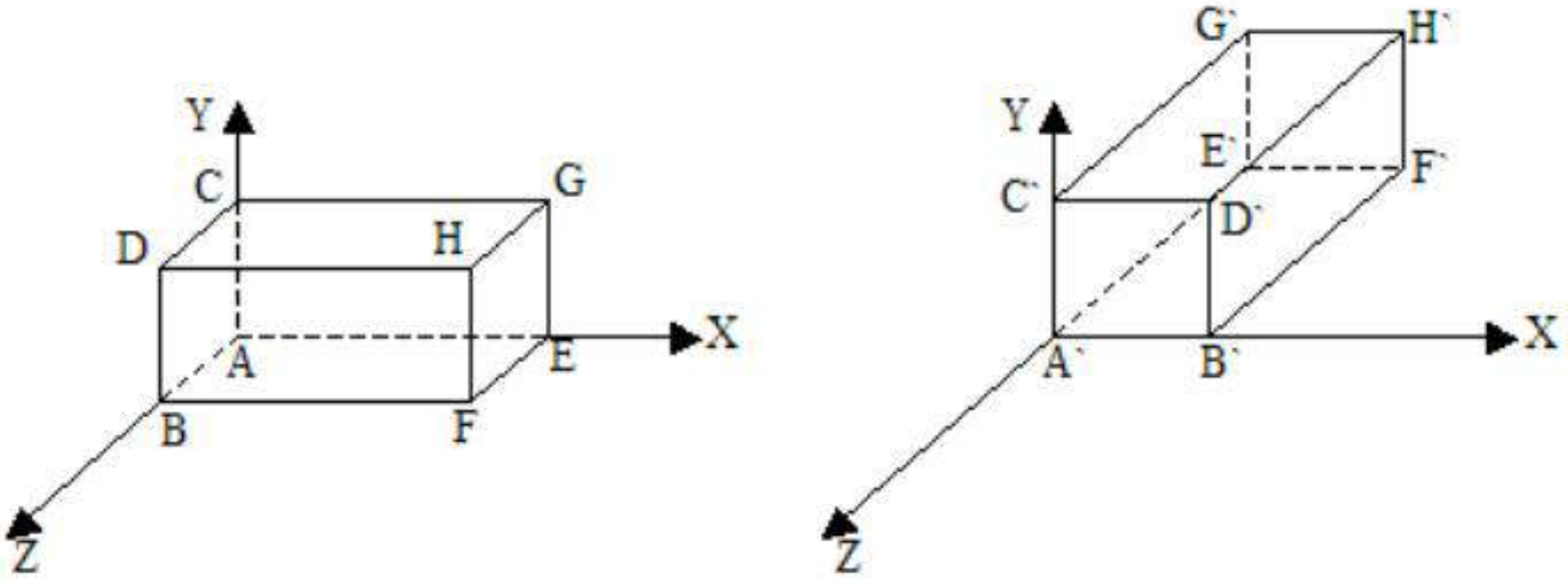


图 3.13　绕 y 轴旋转 90°

4. 对称变换

（1）相对于xy平面的对称变换，z=0平面

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

（2）相对于xz平面的对称变换，y=0平面

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

（3）相对于yz平面的对称变换，x=0平面

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(4) 相对于空间任一平面做对称变换

1）将此平面变换为与某一坐标面重合

2）做相对于该坐标面的对称变换

3）将平面反变换回原位置

5. 错切变换

$$\begin{bmatrix} 1 & b & c & 0 \\ d & 1 & f & 0 \\ g & h & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{cases} x' = x + dy + gz, & d, g \neq 0 \text{关于} x \text{轴方向有错切} \\ y' = y + bx + hz, & b, h \neq 0 \text{关于} y \text{轴方向有错切} \\ z' = z + cx + fy, & c, f \neq 0 \text{关于} z \text{轴方向有错切} \end{cases}$$

三维图形变换

$$\begin{bmatrix} a & b & c & p \\ d & e & f & q \\ g & h & i & r \\ l & m & n & s \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

旋转，比例，对称，错切变换

$$\begin{bmatrix} l & m & n \end{bmatrix}$$

平移变换

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

投影变换

$$\begin{bmatrix} s \end{bmatrix}$$

整体比例变换

例： 设三维空间中有一条任意直线，它由直线上一点Q和沿直线方向的单位方向向量n确定。

Q点坐标为 (x_0, y_0, z_0)，

直线向量 $\mathbf{n}=\begin{bmatrix} n_1 & n_2 & n_3 \end{bmatrix}$ $|\mathbf{n}|=\sqrt{n_1^2+n_2^2+n_3^2}=1$

求绕这条直线旋转 θ 角的**旋转变换矩阵**。

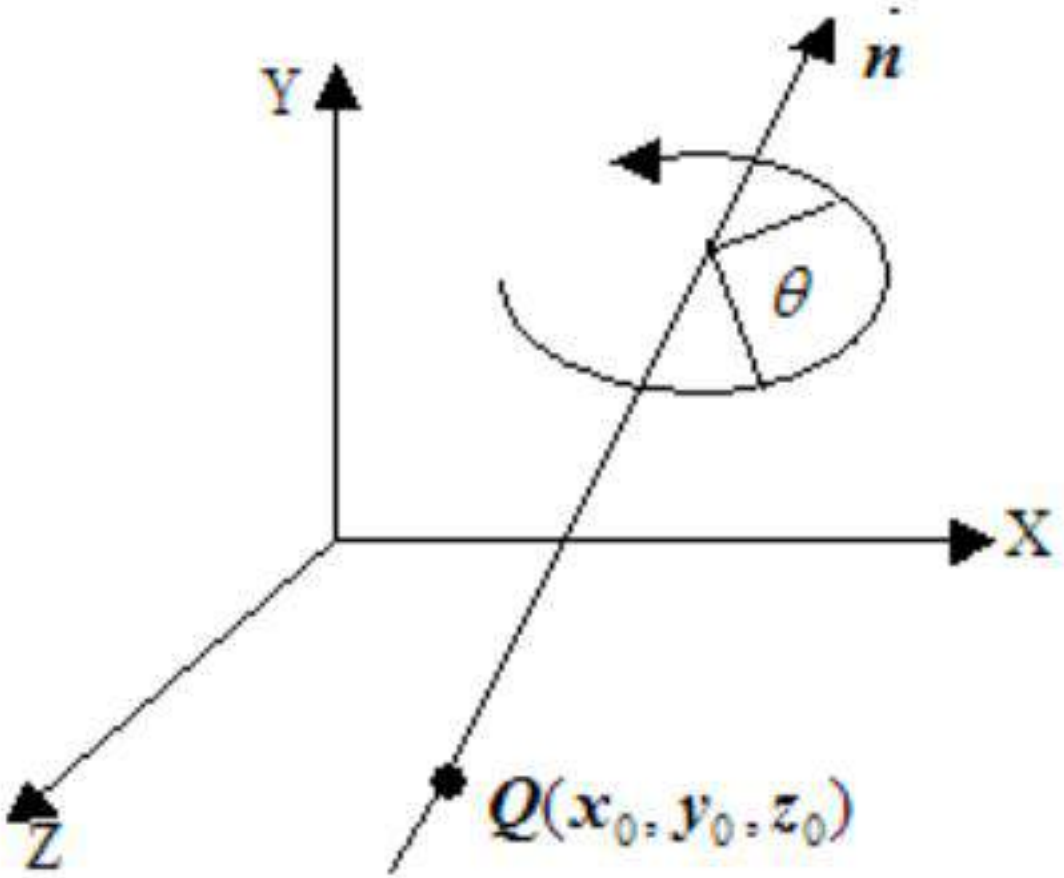


图 3.14　以空间任意直线为轴的三维旋转变换

$$T(-x_0,-y_0,-z_0)\bullet R(\theta)\bullet T(x_0,y_0,z_0)$$

过坐标原点的任意直线为旋转轴的旋转变换可分为五步实现：

（1）做绕X轴旋转 α 角的变换，使旋转轴落在Y=0上。

（2）做绕Y轴旋转 β 角的变换，使旋转轴与Z轴重合。

（3）做绕Z轴旋转 θ 角的旋转变换。

（4）做第2步的逆变换，即做绕Y轴 $-\beta$ 旋转变换

（5）做第1步的逆变换，即做绕X轴 $-\alpha$ 旋转变换

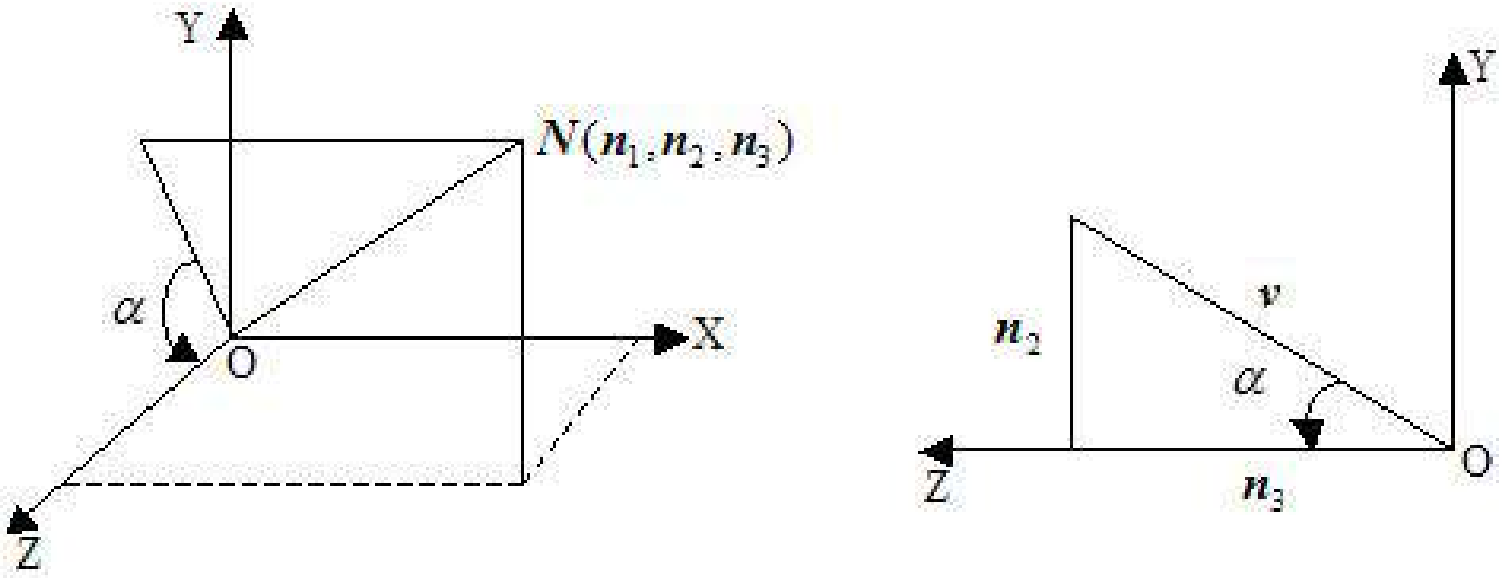


图 3.15　绕 X 轴旋转 α 角（右面是在 X=0 平面上的投影）

$$v = \sqrt{n_2^2 + n_3^2}$$

$$\cos\alpha = \frac{n_3}{v}, \quad \sin\alpha = \frac{n_2}{v}$$

$$R_x(\alpha)=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{n_3}{v} & \frac{n_2}{v} & 0 \\ 0 & -\frac{n_2}{v} & \frac{n_3}{v} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$N'=\begin{bmatrix} n_1 & n_2 & n_3 & 1 \end{bmatrix} R_x(\alpha)$$

$$=\begin{bmatrix} n_1 & 0 & v & 1 \end{bmatrix}$$

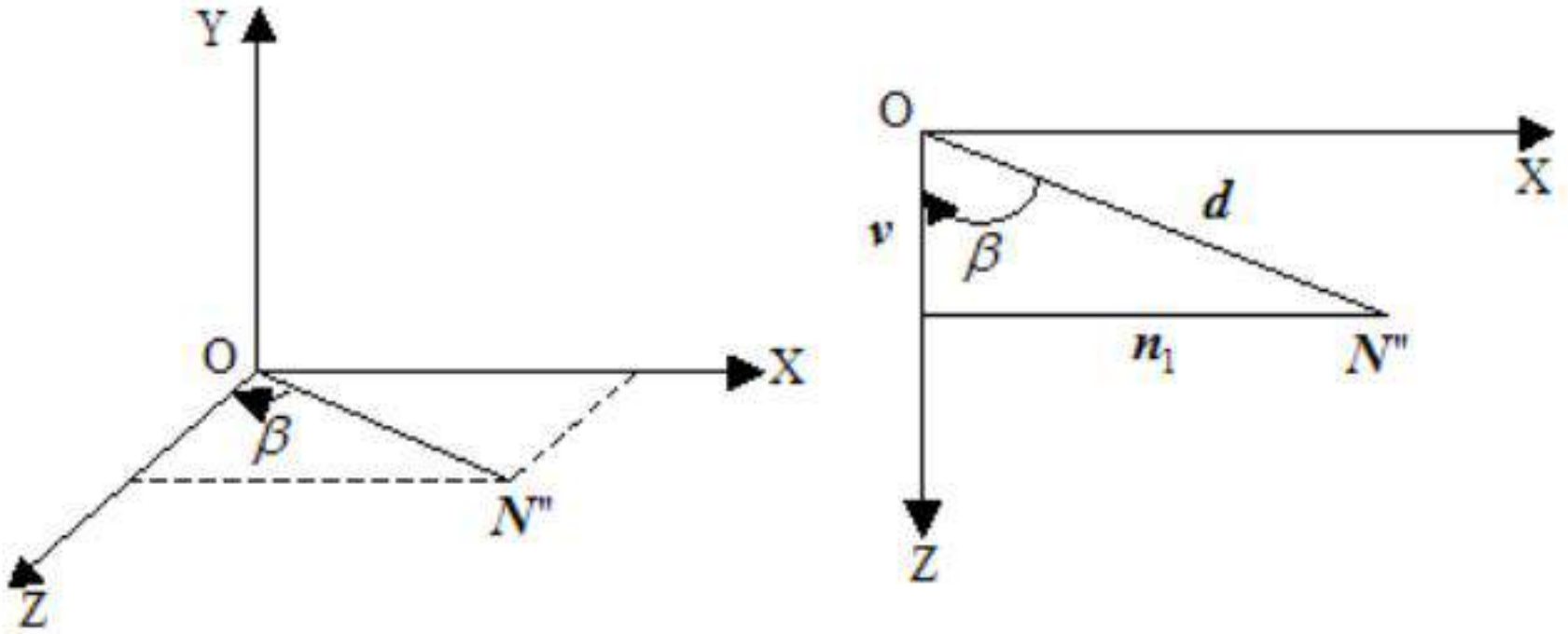


图 3.16　绕 Y 轴旋转 β 角（右面是在 Y=0 平面上的投影）

$$d=\sqrt{n_1^2+n_2^2+n_3^2}=1$$

$$\cos\beta=\frac{v}{d}=v$$

$$\sin\beta=-\frac{n_1}{d}=-n_1$$

$$R_y(\beta)=\begin{bmatrix} v & 0 & n_1 & 0 \\ 0 & 1 & 0 & 0 \\ -n_1 & 0 & v & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$N''=\begin{bmatrix} n_1 & 0 & v & 1 \end{bmatrix} R_y(\beta)$$

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

$$R_z(\theta)=\begin{bmatrix}\cos\theta & \sin\theta & 0 & 0\\ -\sin\theta & \cos\theta & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\quad R_y(-\beta)=\begin{bmatrix}v & 0 & -n_1 & 0\\ 0 & 1 & 0 & 0\\ n_1 & 0 & v & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$R_x(-\alpha)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \frac{n_3}{v} & -\frac{n_2}{v} & 0\\ 0 & \frac{n_2}{v} & \frac{n_3}{v} & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$R(\theta) = R_x(\alpha) \bullet R_y(\beta) \bullet R_z(\theta) \bullet R_y(-\beta) \bullet R_x(-\alpha)$$

$$= \begin{bmatrix} n_1^2 + (1 - n_1^2)\cos\theta & n_1 n_2 (1 - \cos\theta) + n_3 \sin\theta & n_1 n_3 (1 - \cos\theta) - n_2 \sin\theta & 0 \\ n_1 n_2 (1 - \cos\theta) - n_3 \sin\theta & n_2^2 + (1 - n_2^2)\cos\theta & n_2 n_3 (1 - \cos\theta) + n_1 \sin\theta & 0 \\ n_1 n_3 (1 - \cos\theta) + n_2 \sin\theta & n_2 n_3 (1 - \cos\theta) - n_1 \sin\theta & n_3^2 + (1 - n_3^2)\cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

15.

(1)绕x轴旋转到xz平面，然后绕y轴旋转到z轴

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{z_1}{v} & \frac{y_1}{v} & 0 \\ 0 & -\frac{y_1}{v} & \frac{z_1}{v} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{v}{u} & 0 & \frac{x_1}{u} & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{x_1}{u} & 0 & \frac{v}{u} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{v}{u} & 0 & \frac{x_1}{u} & 0 \\ -\frac{x_1 y_1}{uv} & \frac{z_1}{v} & \frac{y_1}{u} & 0 \\ -\frac{x_1 z_1}{uv} & -\frac{y_1}{v} & \frac{z_1}{u} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$v = \sqrt{y_1^2 + z_1^2}, u = \sqrt{x_1^2 + y_1^2 + z_1^2}$$

$$\cos\alpha = z_1 / v, \sin\alpha = y_1 / v, \cos\beta = v / u, \sin\beta = -x_1 / u$$

$$(x_1, y_1, z_1, 1) \to (x_1, 0, v, 1) \to (0, 0, u, 1)$$

(2)绕y轴旋转到yz平面，然后绕x轴旋转到z轴

$$\begin{bmatrix} \frac{z_1}{v} & 0 & \frac{x_1}{v} & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{x_1}{v} & 0 & \frac{z_1}{v} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{v}{u} & \frac{y_1}{u} & 0 \\ 0 & -\frac{y_1}{u} & \frac{v}{u} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{z_1}{v} & -\frac{x_1 y_1}{uv} & \frac{x_1}{u} & 0 \\ 0 & \frac{v}{u} & \frac{y_1}{u} & 0 \\ -\frac{x_1}{v} & -\frac{y_1 z_1}{uv} & \frac{z_1}{u} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$v = \sqrt{x_1^{\,2} + z_1^{\,2}}, u = \sqrt{x_1^{\,2} + y_1^{\,2} + z_1^{\,2}}$$

$$\cos\alpha = -z_1 / v, \sin\alpha = -x_1 / v, \cos\beta = v / u, \sin\beta = y_1 / u$$

$$(x_1, y_1, z_1, 1) \to (0, y_1, v, 1) \to (0, 0, u, 1)$$

(3)绕z轴旋转到xz平面，然后绕y轴旋转到z轴

$$\begin{bmatrix} \frac{x_1}{v} & \frac{-y_1}{v} & 0 & 0 \\ \frac{y_1}{v} & \frac{x_1}{v} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{z_1}{u} & 0 & \frac{v}{u} & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{v}{u} & 0 & \frac{z_1}{u} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{x_1 z_1}{uv} & -\frac{y_1}{v} & \frac{x_1}{u} & 0 \\ \frac{y_1 z_1}{uv} & \frac{x_1}{v} & \frac{y_1}{u} & 0 \\ -\frac{v}{u} & 0 & \frac{z_1}{u} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$v = \sqrt{x_1^{\,2} + y_1^{\,2}}, u = \sqrt{x_1^{\,2} + y_1^{\,2} + z_1^{\,2}}$$

$$\cos\alpha = x_1 / v, \sin\alpha = -y_1 / v, \cos\beta = z_1 / u, \sin\beta = -v / u$$

$$(x_1, y_1, z_1, 1) \to (v, 0, z_1, 1) \to (0, 0, u, 1)$$

第五节 投影

三维观察过程：

世界坐标系→观察坐标系→观察表面

投影：把n维空间中的点投射到小于n维的空间中去

投影的形成：首先在三维空间中确定一个投影中心和一个投影平面，然后从投影中心引出一些投射直线，这些直线通过形体上的每一点，与投影平面相交，在投影平面上就形成了形体的投影。

投影中心与投影平面之间距离为d

d无限 平行投影，投射线平行，立体感差，能保持比例关系

d有限 透视投影，立体感强，更真实，但不能保持原来的比例关系

一. 平行投影

1. 正交投影　投影方向与投影平面的法向相同

常见的正交投影是正视投影、顶视投影和侧视投影

如果投影平面Z=0，投影方向是沿Z轴,设三维空间中有普通坐标为(x,y,z)的一点P，投影后，成为点P’，普通坐标为 (x’,y’,z’) 可知：

$$x' = x, \quad y' = y, \quad z' = 0$$

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

$$M_{顶}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&1\end{bmatrix}$$

$$M_{正}=\begin{bmatrix}0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix},\quad M_{侧}=\begin{bmatrix}1&0&0&0\\0&0&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

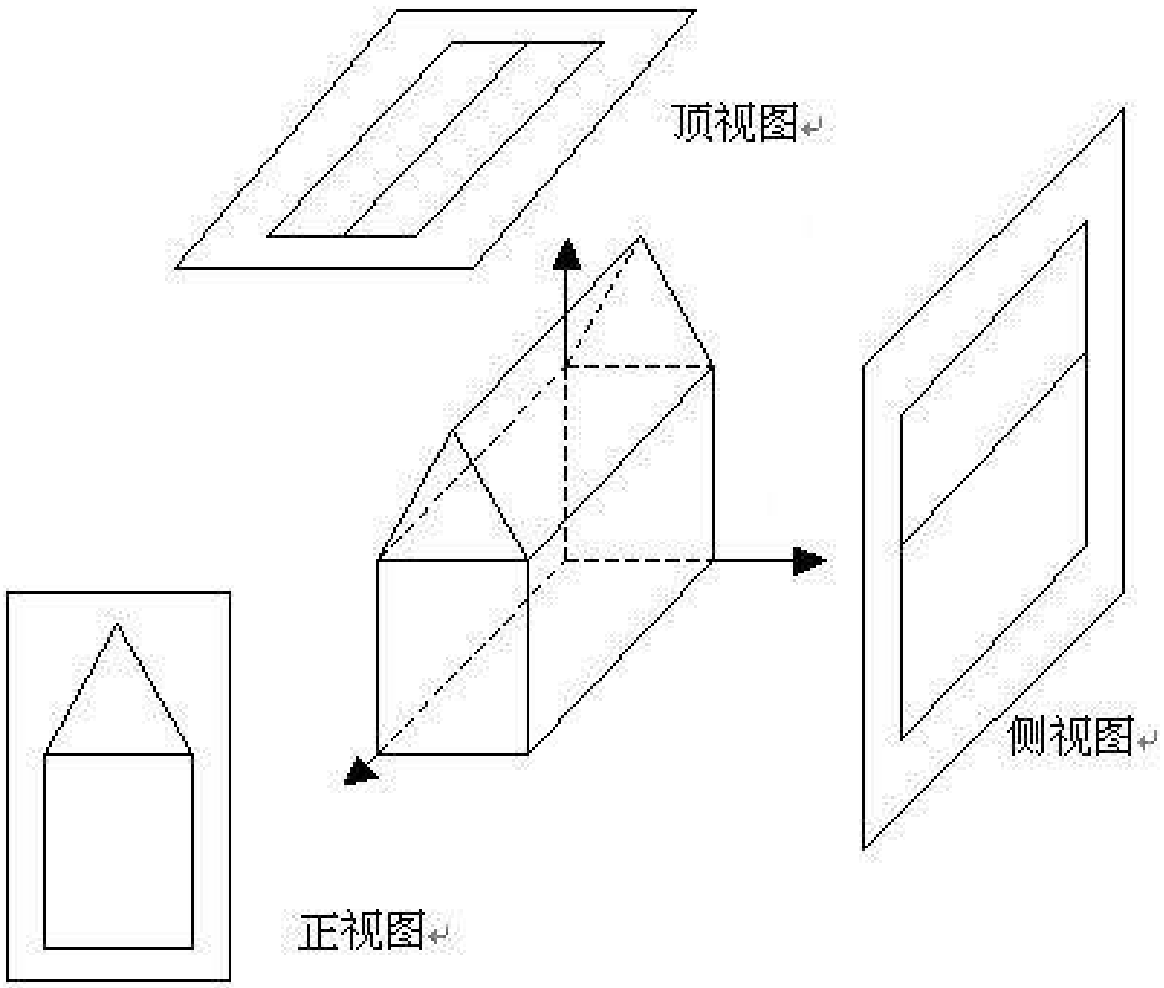


图 3.17　三视图

正投影：投影平面垂直于坐标轴的正交投影。

等轴投影：投影方向与三个坐标轴的夹角都相等。这种投影能使在三个坐标轴方向上有相等的透视缩短。

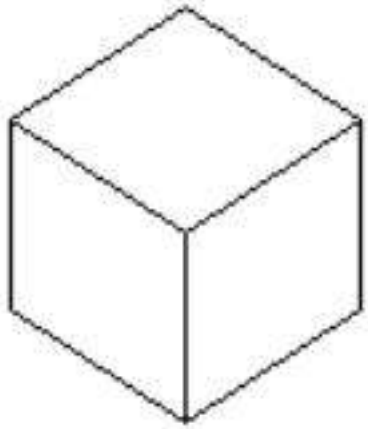


图 3.18　单位正方体的等轴投影图

2. 斜交投影

当平行投影中投影平面的法线方向与投影方向不同时就得到斜交投影。在斜交投影中，投影平面一般取坐标平面。

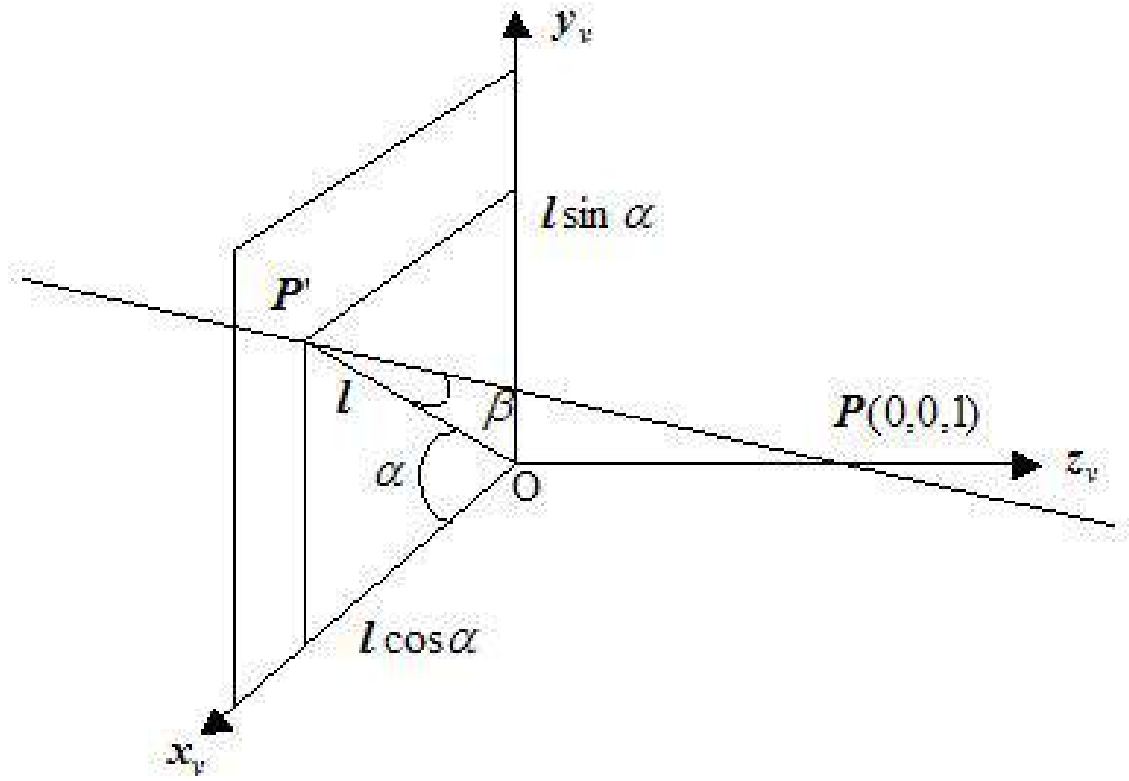


图 3.19　对点 $P(0,0,1)$ 作斜平行投影到点 $P'(l\cos\alpha, l\sin\alpha, 0)$

设三维空间中有普通坐标为的任意一点(x,y,z)，经斜交投影后所得投影点普通坐标为(x',y',z')。显然$z'=0$，有：

$$\frac{x'-x}{z}=\frac{l\cos\alpha}{1},\quad \frac{y'-y}{z}=\frac{l\sin\alpha}{1}$$

$$x'=x+z\bullet(l\bullet\cos\alpha)$$

$$y'=y+z\bullet(l\bullet\sin\alpha)$$

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ l\cos\alpha & l\sin\alpha & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_{ob} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ l\cos\alpha & l\sin\alpha & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

斜二测投影:垂直于投影平面的线段长度缩短为原来的一半;

$$|OP'| = \frac{1}{2}|OP|, tg\beta = 2, \beta = 63.4^o, l = \frac{1}{2}, \alpha 任意取$$

斜等轴投影:使垂直于投影平面的线段仍保持长度.

$$|OP'| = |OP|, \beta = 45^o, l = 1, \alpha 任意取$$

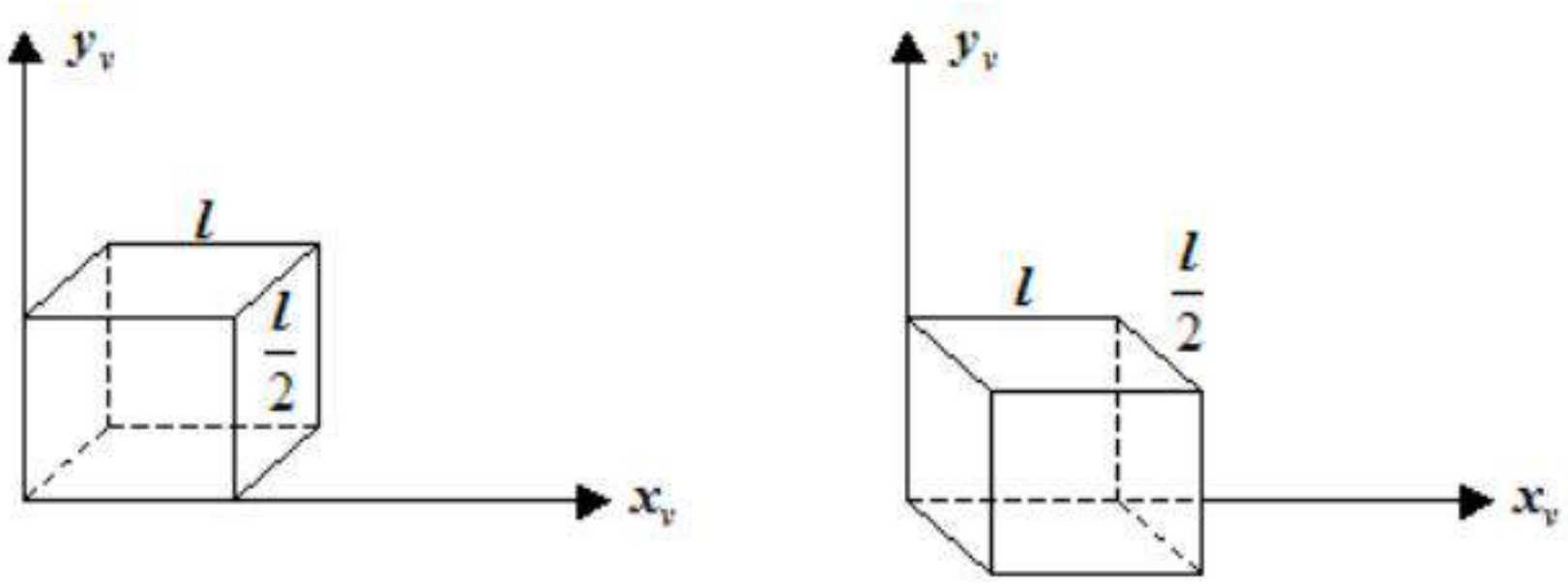


图 3.20　边长为l的正方体的两种斜二测投影的示意图

二. 透视投影

性质： 任意一组平行直线，如果平行于投影平面，则经透视投影后所得到的直线或者重合，或者仍保持平行；如果不平行于投影平面，将不再保持平行，并且必会汇聚于同一点，这个点称为消失点，也称为灭点。

空间中可以取得任意多组不平行于投影平面的平行直线，所以消失点也可以取得任意多个。

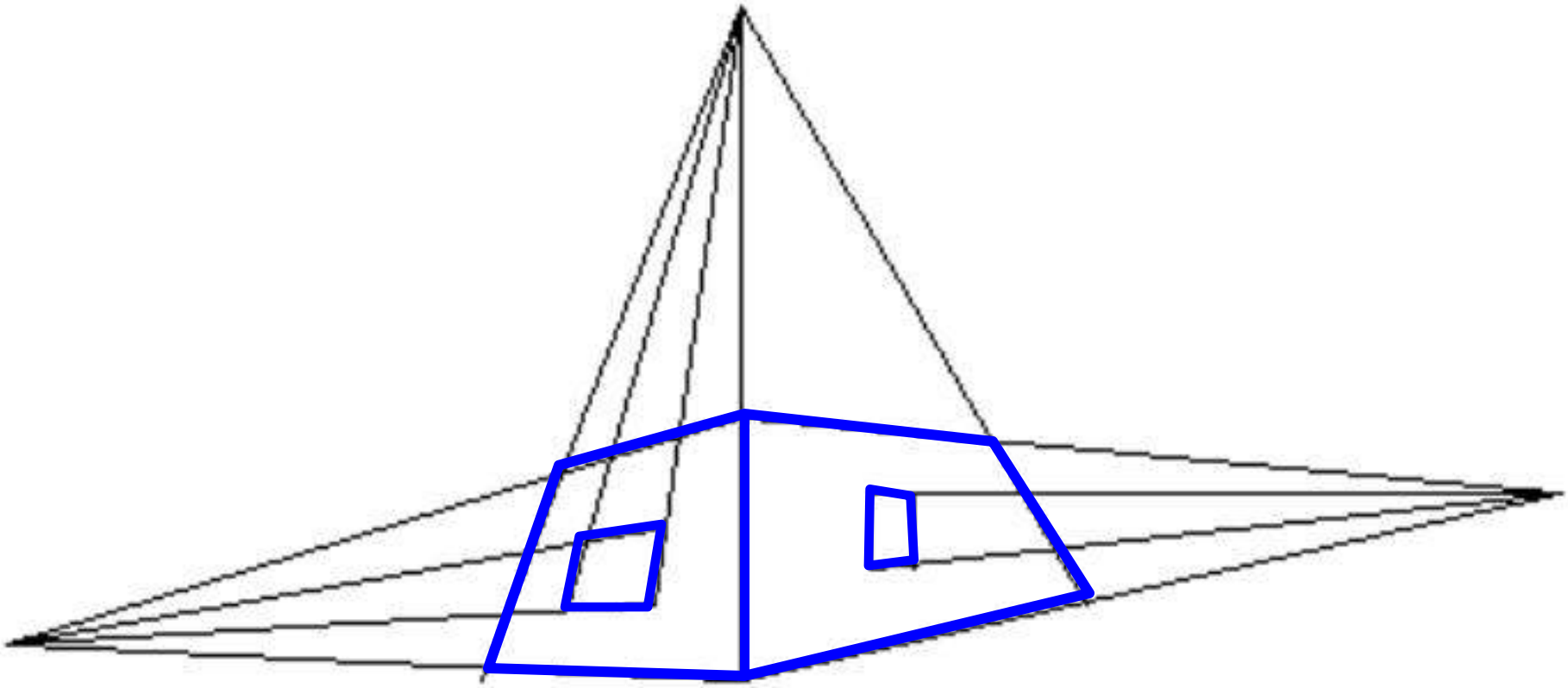
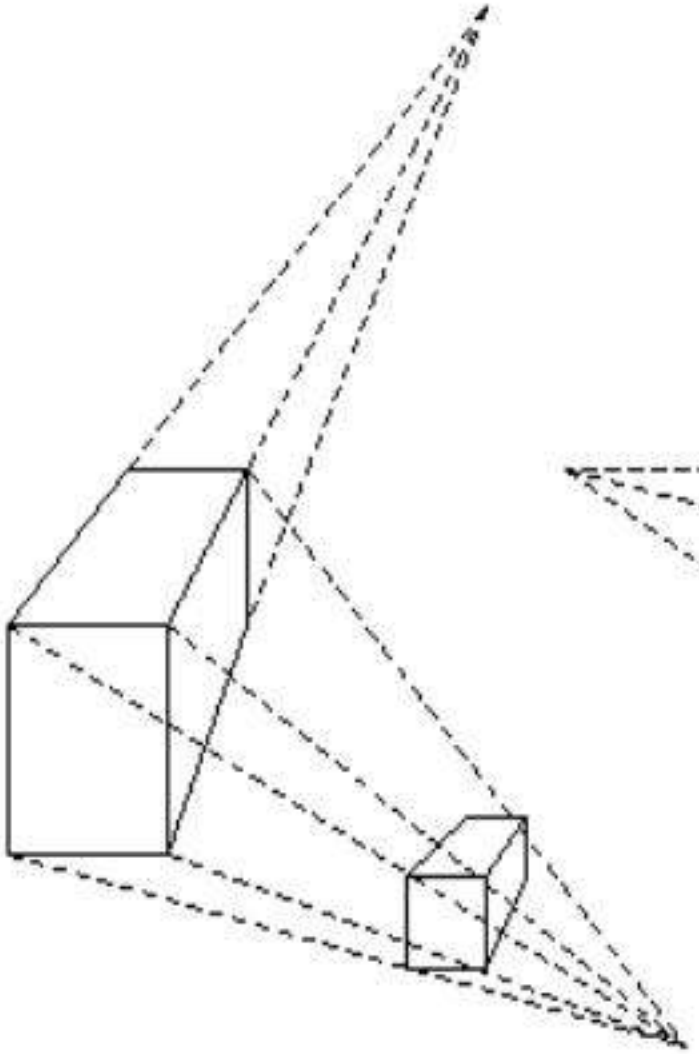


图 3.21　消失点示意图

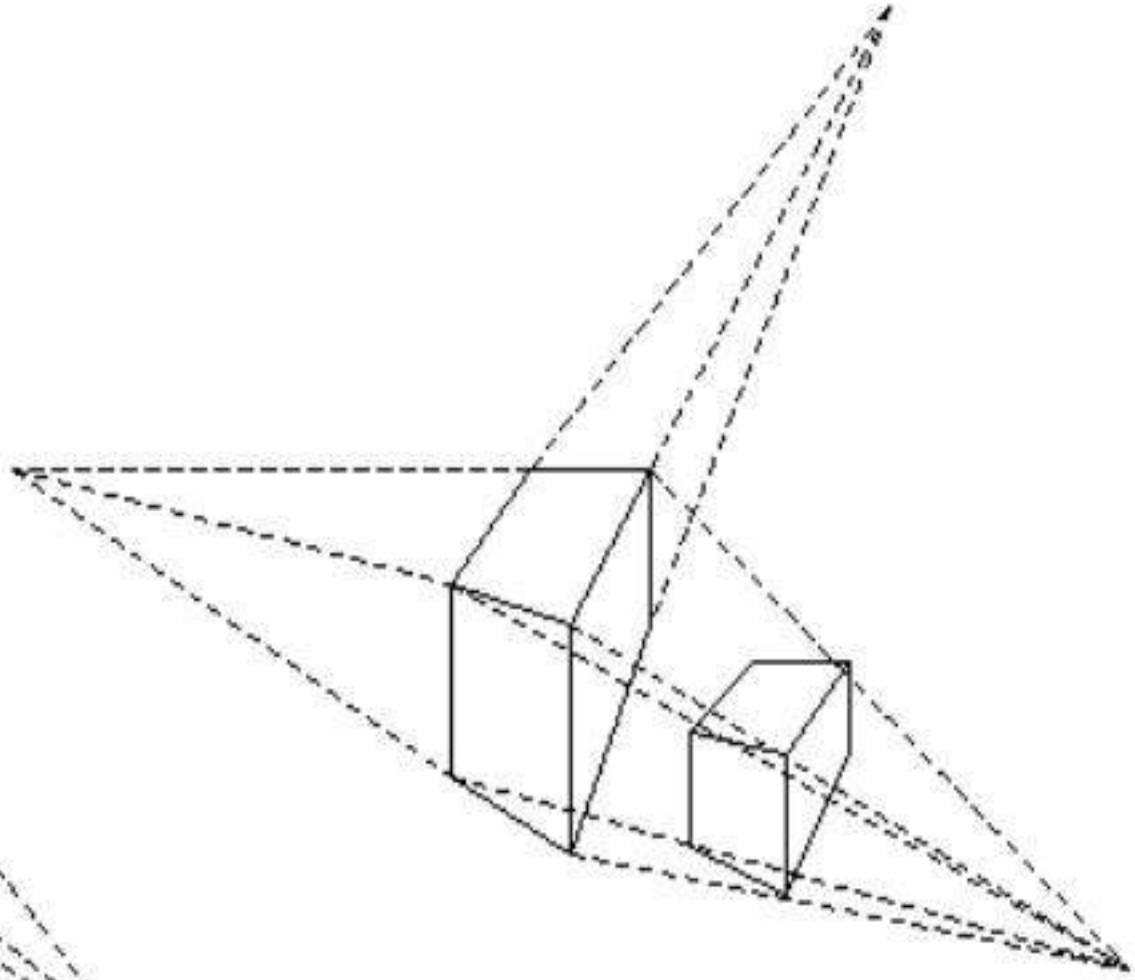
主消失点：如果一组平行直线平行于三个坐标轴中的一个，那么对应的消失点将落在坐标轴上，这样的消失点称为主消失点。

因为只有三个坐标轴，所以最多只有三个主消失点。根据主消失点的数目，透视投影可以分为一点透视、二点透视、三点透视。

如果投影平面截Z轴并与它垂直，这时就只能在Z轴方向上有主消失点。



(1) 一点透视



(2) 两点透视

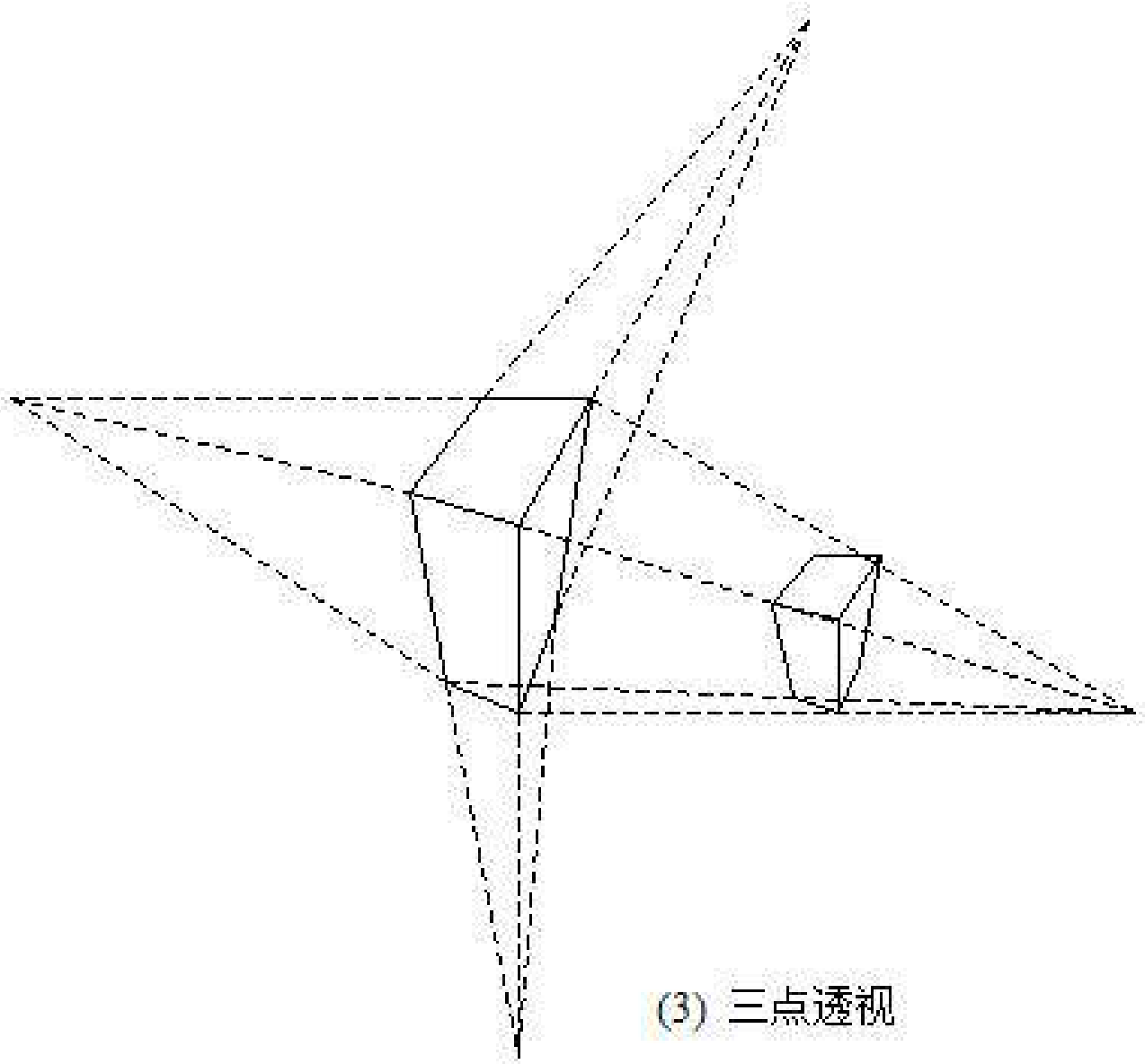
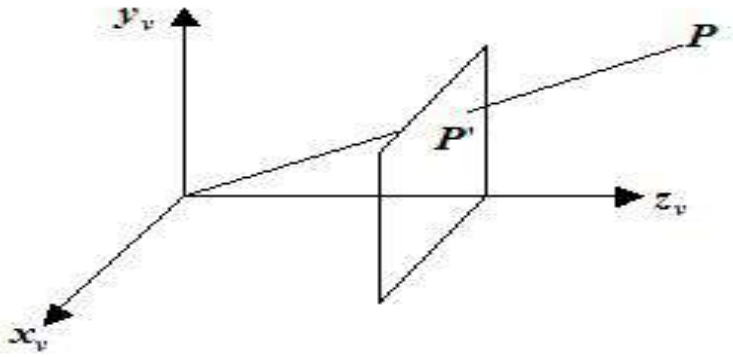


图 3.22　一个长方体的三种透视投影图

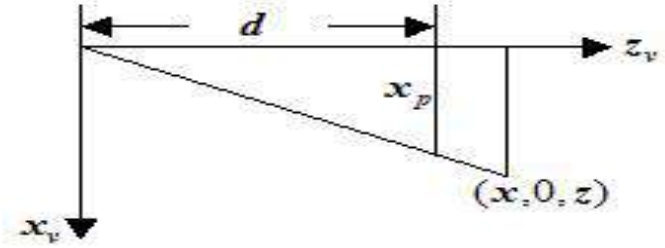
单消失点的透视投影计算

投影中心是坐标原点
投影平面垂直于Z轴

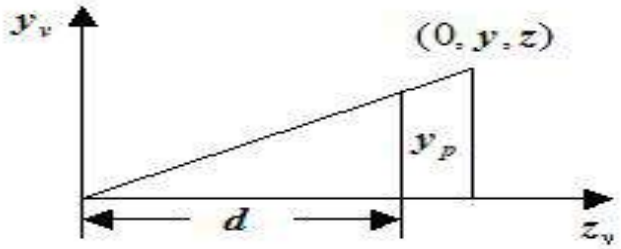
设投影平面位于处Z=d，可设d>0。这样投影平面是平面Z=d，对空间中任意一点P ，其普通坐标为(x, y, z)，它在投影平面上的投影点P’ 的普通坐标为(x’ , y’ , z’)



(1) 求点 P 的透视投影 P'



(2) 沿 y_v 轴顶视



(3) 沿 x_v 轴侧视

图 3.23　单消失点透视投影计算

$$\frac{x_p}{d}=\frac{x}{z}\Rightarrow x_p=\frac{d\bullet x}{z}=\frac{x}{z/d}$$

$$\frac{y_p}{d}=\frac{y}{z}\Rightarrow y_p=\frac{d\bullet y}{z}=\frac{y}{z/d},z_p=d$$

使用齐次坐标可以得出：

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{d} \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} x & y & z & z/d \end{bmatrix}$$

对齐次坐标进行规范化后得：

$$\left[\begin{array}{cccc} \dfrac{x}{z/d} & \dfrac{y}{z/d} & d & 1 \end{array}\right]$$

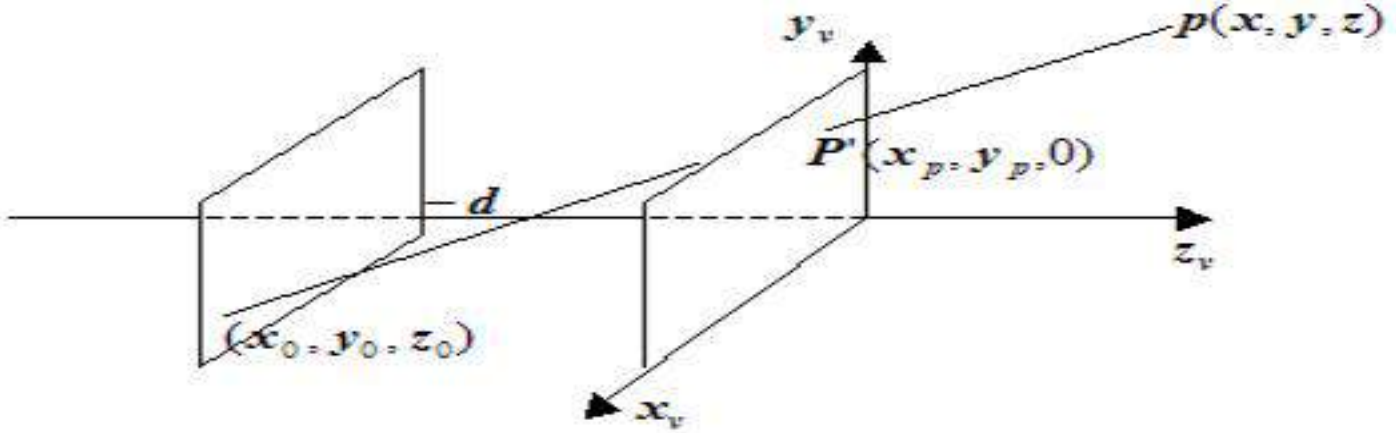
透视投影变换矩阵为：

$$M_{per}=\left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \dfrac{1}{d} \\ 0 & 0 & 0 & 0 \end{array}\right]$$

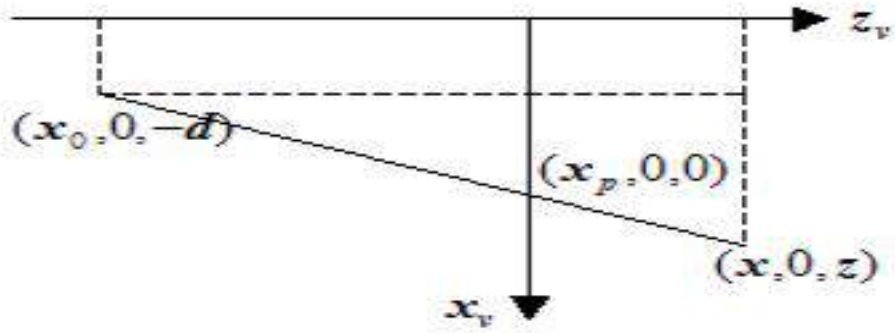
实用中常取为投影平面，这时投影中心可取空间中任意一点 (x_0, y_0, z_0) 如图3.24所示，这里假定了 z_0 是一个负数，即 $z_0 = -d$ $d > 0$ ， 当然这个假定并不是必要的。用前面相同的方法，可以得出：

$$\frac{x_p - x_0}{d} = \frac{x - x_0}{z + d}, \quad \frac{y_p - y_0}{d} = \frac{y - y_0}{z + d}$$

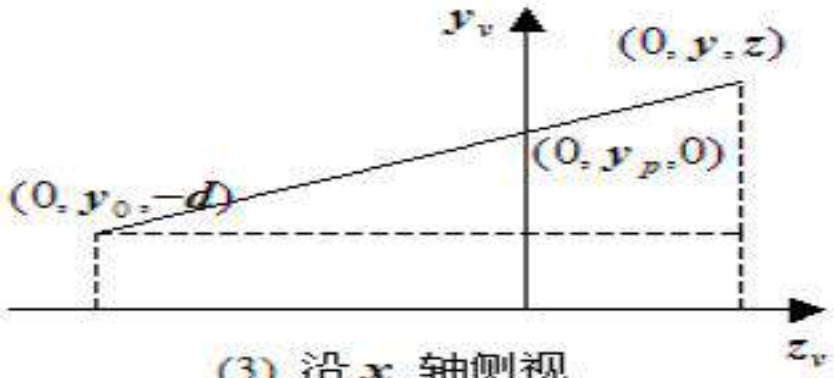
$$x_p = x_0 + \frac{d \bullet (x - x_0)}{z + d} \qquad y_p = y_0 + \frac{d \bullet (y - y_0)}{z + d}$$



(1) 求点 P 的透视投影 P'



(2) 沿 y_v 轴顶视



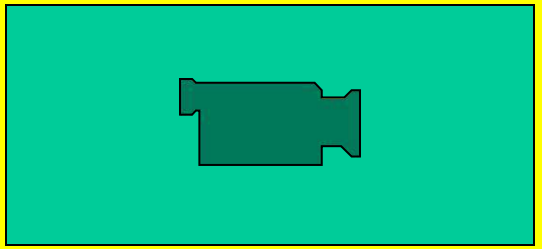
(3) 沿 x_v 轴侧视

图 3.24　投影平面是 $z = 0$，投影中心是 $(x_0, y_0, -d)$ 时的透视投影

$$M'_{per} = T(-x_0,-y_0,d) \bullet M_{per} \bullet T(x_0,y_0,-d)$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_0 & -y_0 & d & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{d} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_0 & y_0 & -d & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{x_0}{d} & \frac{y_0}{d} & 0 & \frac{1}{d} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



可以用齐次坐标验证，有：

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{x_0}{d} & \frac{y_0}{d} & 0 & \frac{1}{d} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x+\frac{x_0 \bullet z}{d} & y+\frac{y_0 \bullet z}{d} & 0 & \frac{z}{d}+1 \end{bmatrix}$$

将结果规范化后得到齐次坐标为：

$$\begin{bmatrix} x_0+\frac{d \bullet (x-x_0)}{z+d} & y_0+\frac{d \bullet (y-y_0)}{z+d} & 0 & 1 \end{bmatrix}$$

例子：求世界坐标系到观察坐标系的变换矩阵

1）首先确定观察坐标系

观察位置定为观察坐标系的原点，

观察方向为zv轴的方向，指向世界坐标系的原点

假定xv轴平行于世界坐标系的z=0平面

这样观察坐标系唯一确定

2）世界坐标系的客体坐标变换到观察坐标系的坐标

3）在观察坐标系做投影变换，得到投影图

设物体上一点在世界坐标系中坐标为（xw, yw, zw, 1), 则在观察坐标系中（xv, yv, zv, 1)= (xw, yw, zw, 1)*V

1）将用户坐标系的原点平移到视点（设视点在用户坐标系下的坐标为（a, b, c), xv轴一定在z=c平面内）。形体上的点H1=(-a, -b, -c)

$$H_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -a & -b & -c & 0 \end{bmatrix}$$

2)令平移后的新坐标系绕x’轴逆时针旋转90度角,则形体上的点顺时针旋转90度 $H_2 = R_x(-90^o)$

$$H_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3)再将新坐标系绕y’顺时针旋转θ 角，此时 θ 大于180度，形体上的点逆时针旋转 θ 角，$H_3 = R_y(\theta)$

记

$$v = \sqrt{a^2 + b^2}, \cos\theta = -\frac{b}{v}, \sin\theta = -\frac{a}{v}$$

$$H_3 = \begin{bmatrix} -\frac{b}{v} & 0 & \frac{a}{v} & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{a}{v} & 0 & -\frac{b}{v} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4）再令新坐标系绕x’轴顺时针旋转φ角，形体上的点逆时针旋转φ 角,$H_4 = R_x(\varphi)$

令 $$u = \sqrt{a^2 + b^2 + c^2}, \cos\varphi = \frac{v}{u}, \sin\varphi = \frac{c}{u}$$

$$H_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{v}{u} & \frac{c}{u} & 0 \\ 0 & -\frac{c}{u} & \frac{v}{u} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5)右手坐标系变成左手坐标系，z轴反向 $H_5 = S(1,1,-1)$

$$H_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$V = H_1 \bullet H_2 \bullet H_3 \bullet H_4 \bullet H_5 = \begin{bmatrix} -\frac{b}{v} & -\frac{ac}{uv} & -\frac{a}{u} & 0 \\ \frac{a}{v} & -\frac{bc}{uv} & -\frac{b}{u} & 0 \\ 0 & \frac{v}{u} & -\frac{c}{u} & 0 \\ 0 & 0 & u & 1 \end{bmatrix}$$

$$u = \sqrt{a^2 + b^2 + c^2}, v = \sqrt{a^2 + b^2}$$

从观察位置（6，8，7.5)向原点方向进行观察

$$a=6,b=8,c=7.5,v=\sqrt{a^2+b^2}=10,u=\sqrt{a^2+b^2+c^2}=12.5$$

$$H=\begin{bmatrix} -0.8 & -0.36 & -0.48 & 0 \\ 0.6 & -0.48 & -0.64 & 0 \\ 0 & 0.8 & -0.6 & 0 \\ 0 & 0 & 12.5 & 1 \end{bmatrix}$$

设有一个中心在原点，边长为2的正方体，经H变换到观察坐标系

$$\begin{bmatrix} -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 \\ -1 & -1 & -1 & 1 \\ -1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -0.8 & -0.36 & -0.48 & 0 \\ 0.6 & -0.48 & -0.64 & 0 \\ 0 & 0.8 & -0.6 & 0 \\ 0 & 0 & 12.5 & 1 \end{bmatrix} = \begin{bmatrix} 1.4 & -0.92 & 12.94 & 1 \\ -0.2 & -1.64 & 11.98 & 1 \\ -1.4 & -0.68 & 13.26 & 1 \\ 0.2 & 0.04 & 14.22 & 1 \\ 1.4 & 0.68 & 11.74 & 1 \\ -0.2 & -0.04 & 10.78 & 1 \\ -1.4 & 0.92 & 12.06 & 1 \\ 0.2 & 1.64 & 13.02 & 1 \end{bmatrix}$$

取投影平面过世界坐标系原点，且与ZV轴垂直的平面，
投影中心（观察位置）在原点
投影矩阵为：

$$M_{per}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & \frac{1}{12.5}\\ 0 & 0 & 0 & 0\end{bmatrix}$$

$$\begin{bmatrix} 1.35 & -0.89 & 12.5 & 1 \\ -0.21 & 1.71 & 12.5 & 1 \\ -1.32 & -0.64 & 12.5 & 1 \\ 0.18 & 0.035 & 12.5 & 1 \\ 1.45 & 0.72 & 12.5 & 1 \\ -0.23 & -0.046 & 12.5 & 1 \\ -1.49 & 0.95 & 12.5 & 1 \\ 0.19 & 1.57 & 12.5 & 1 \end{bmatrix}$$

投影总结：

1平行投影

1）正交

2）斜交：斜二测，斜等轴

2透视投影

1）单点

2）双点

3）三点

第六节 裁剪

裁剪就是去掉窗口外的不可见部分，保留窗口内的可见部分的过程。

裁剪区域：矩形、任意图形

裁剪对象：点、线段、多边形、二维或三维形体

假设窗口的两个对角顶点分别是(x_l,y_b)、(x_r,y_t)，则同时满足下列不等式的点(x,y)是要保留的点，否则就要被舍弃：

$$x_l \le x \le x_r, \quad y_b \le y \le y_t$$

直线段裁剪算法

1. Cohen-Sutherland算法

对象的可见性问题：

①判断直线段完全可见，是，绘制，结束，保留

②判定完全不可见，是，不绘制，结束，舍弃

③ ① ②都不满足，部分可见，求线段与窗口边界线的交点（分割），对子线段继续① ～③

算法步骤如下：

1. 编码：

每个区域用一个4位二进制编码来标识，代码确定如下：

第1位为1：若区域在窗口上方

2　　下方

3　　右侧

4　　左侧

设线段的两个端点p1(x1, y1), p2(x2, y2), 求出p1, p2所处区域的代码c1和c2

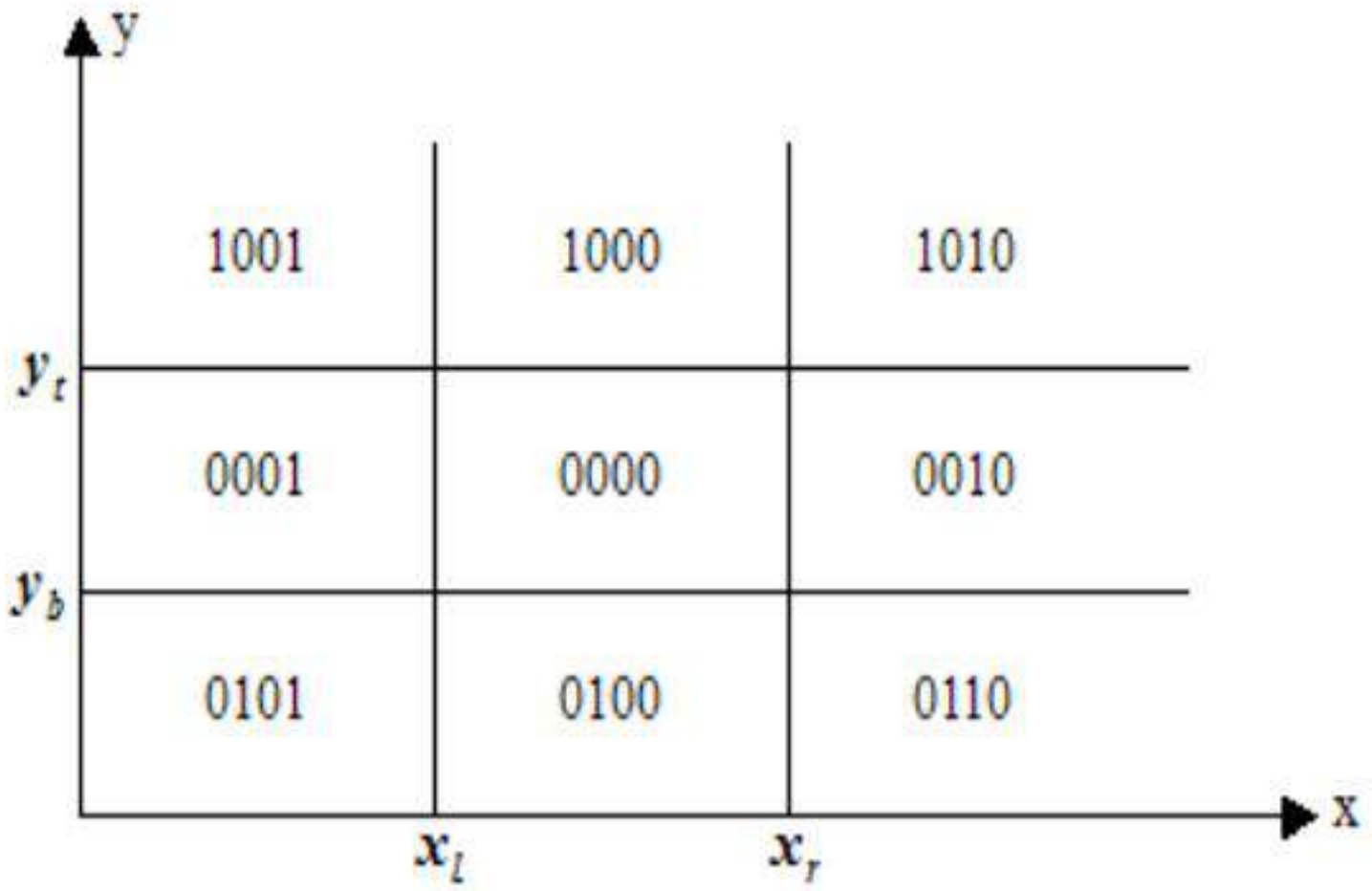


图 3.25　区域编码

2. 判别：

(1)c1=c2=0,完全可见，保留，　　AB

(2)C1&C2!=0 完全不可见，舍弃　　CD

(3)　不满足(1) (2)可分为三种情况

① 一个端点在内另一个端点在外　EF

② 两个端点都在外，且直线段在外JK

③ 两个端点都在外，但直线中部跨越窗口 HI

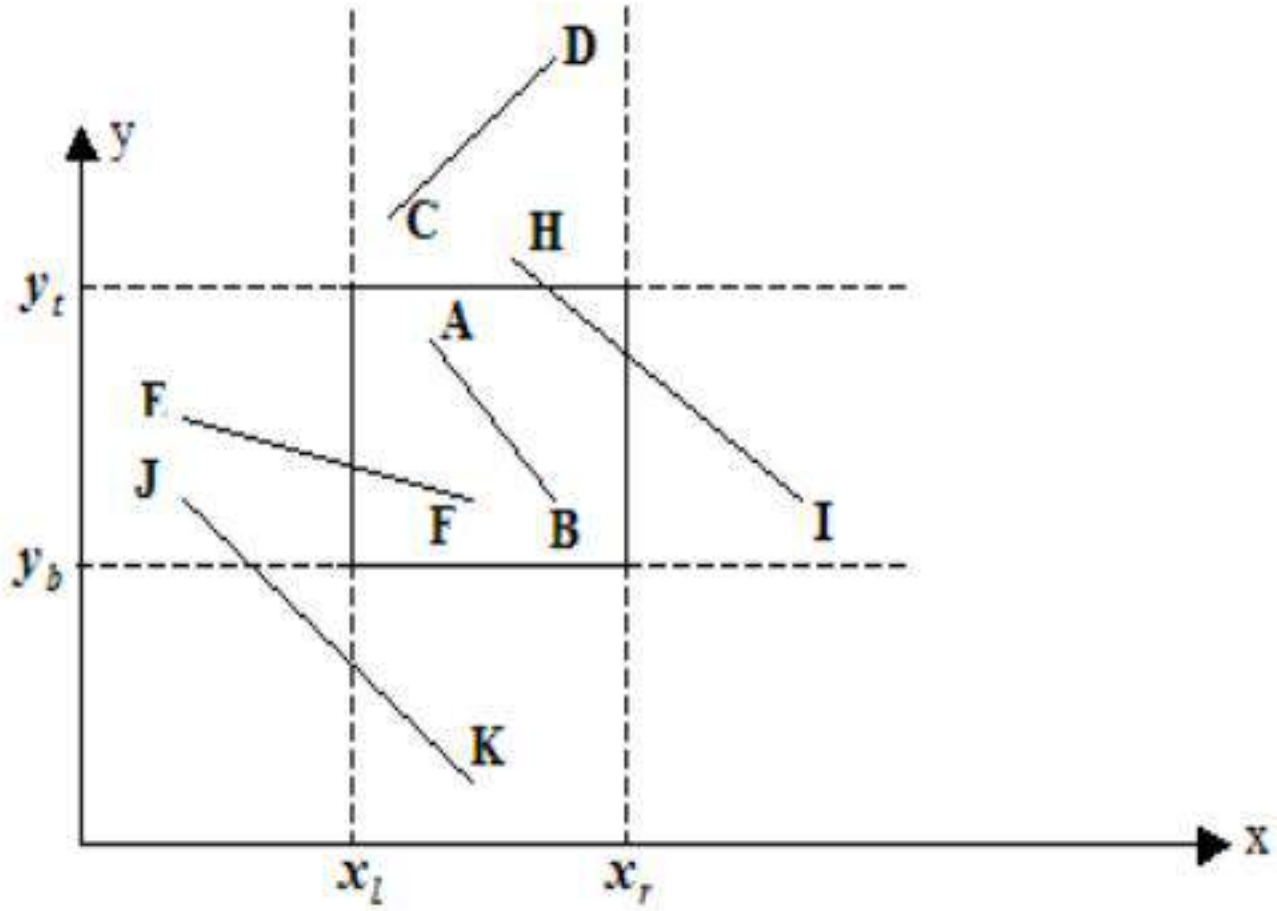


图 3.26 Cohen-sutherland 裁剪算法的例子

3. 求交。

求线段与窗口边界及其延长线的交点

$$\frac{y-y_1}{x-x_1}=\frac{y_2-y_1}{x_2-x_1}$$

$$\Rightarrow y=y_1+m(x-x_1)$$

将 $x=xl, x=xr$ ，代入上式，求出交点坐标

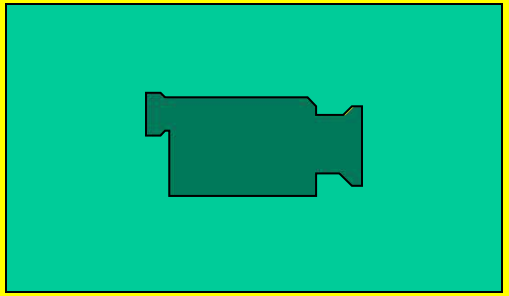
4. 对剩下的子线段重复以上各步，从而完成对直线段的裁剪

算法的程序实现如下

```
void Cohen_Sutherland(double x0, y0, x2, y2)
{
   int c, c1, c2;
   double x, y;
   makecode(x0, y0,c1); makecode(x2, y2, c2);
   while (c1!=0 || c2!=0)
{
  if (c1&c2 !  =0) return;
 c=c1; if (c==0) c=c2;
```

```
if (c&1==1) {y=y0+(y2-y0)*(x1-x0)/(x2-x0); x=x1;}
else if (c&2==2) {y=y0+(y2-y0)*(xr-x0)/(x2-x0);
                                    x=xr;}
else if (c&4==4) {x=x0+(x2-x0)*(yb-y0)/(y2-y0);
                                    y=yb;}
else if (c&8==8) {x=x0+(x2-x0)*(yt-y0)/(y2-y0);
                                    y=yt;}
        if (c==c1)
        {x0=x; y0=y; makecode(x, y, c1);}
        else
            {x2=x; y2=y; makecode(x, y, c2);}
             }
   showline(x0, y0, x2, y2); //显示可见线段}
```

```
void makecode(double x, y; int c)
{
        c=0;
        if (x<xl) c=1;
        else if (x>xr) c=2;
       if (y<yb) c=c+4;
       else if (y>yt) c=c+8;
}
```



2. 中点分割算法

基本思想：可分成两个过程平行进行，

从p0点出发找出离点p0 最近的可见点，

从p1点出发找出离点 p1 最近的可见点。

这两个最近可见点的连线就是原直线段的可见部分。

p0p1->mid(p0,p1)=pm->mid(p0,pm)

->mid(pm,p1)

1)完全可见，绘制线段

2)完全不可见：线段不绘制，舍弃

3)求中点pm,分割成p0pm,pmp1,在分别进行判断

直到 $|p_m p_1| < \varepsilon$

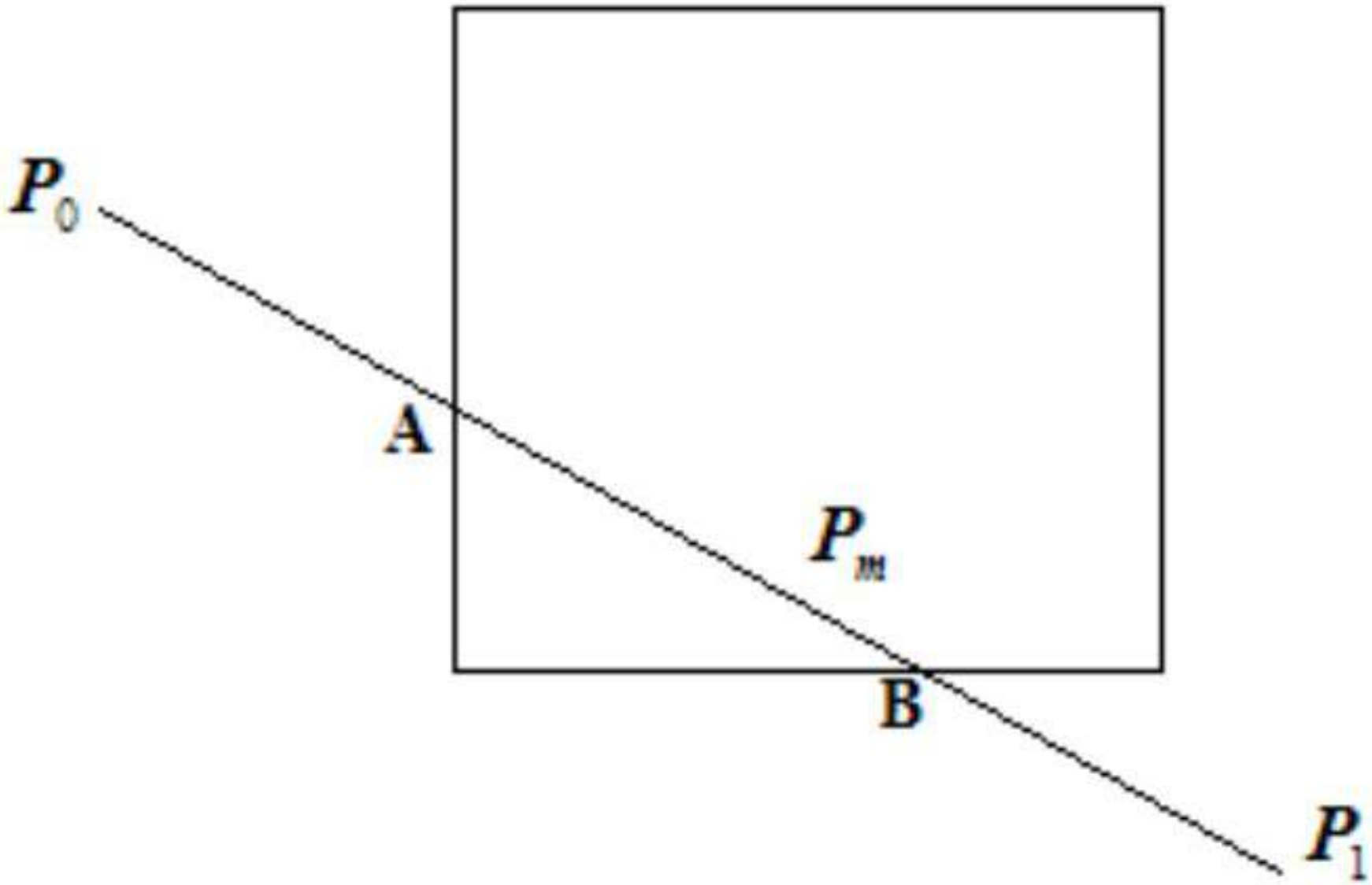


图 3.27　中点分割算法

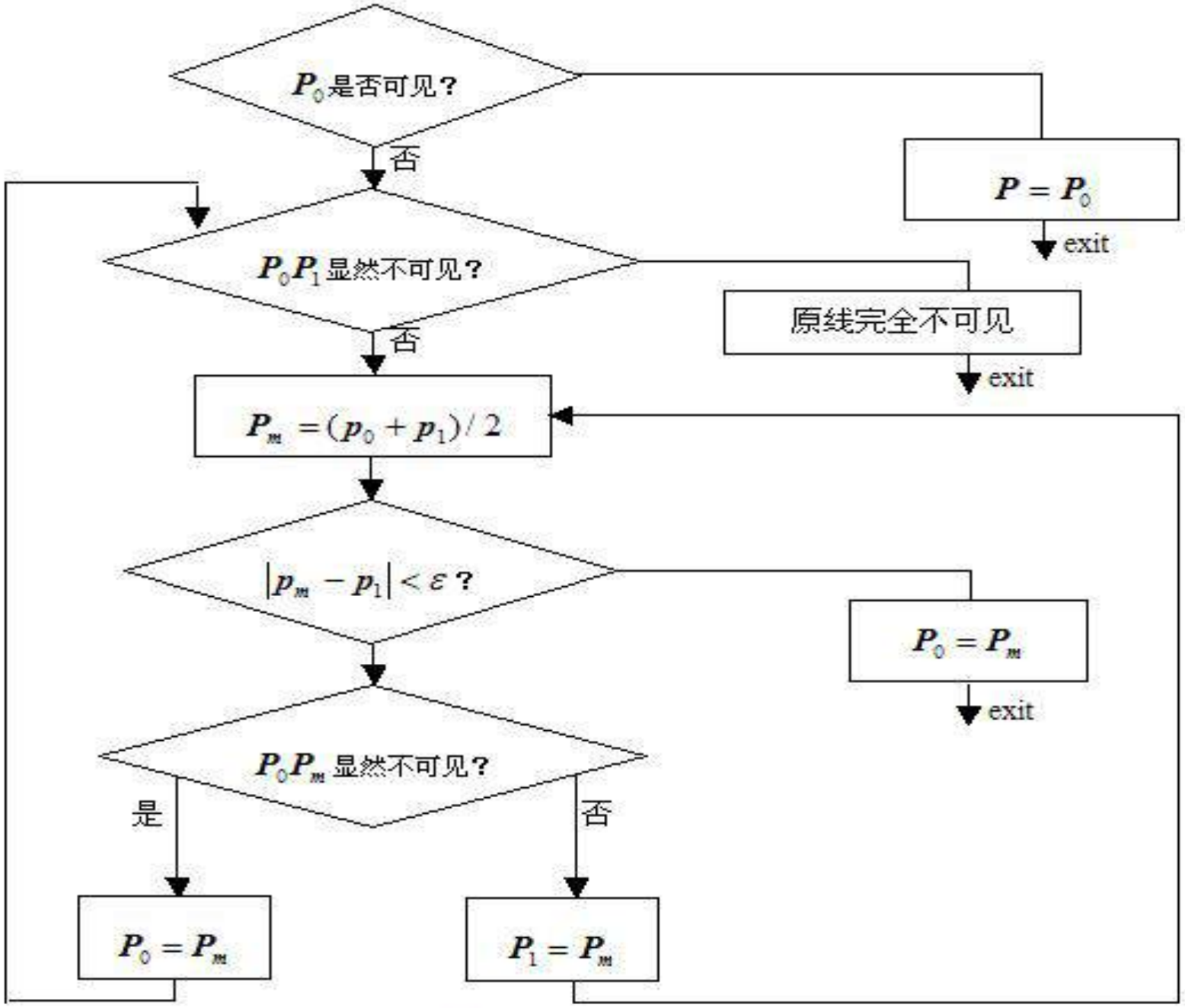


图 3.28　中点分割算法框图

3. 梁友栋-Barsky算法

设要裁剪的直线段为P0P1，Pi的坐标为(xi,yi)，i=0,1。P0P1和窗口边界交于A、B、C和D四个点。该算法的基本思想是从A、B和P0三点中找出最靠近P1的点，在图3.29中该点是P0。从C、D和P1点中找出最靠近P0的点，在图3.29中该点是点C。那么就是线段上的可见部分

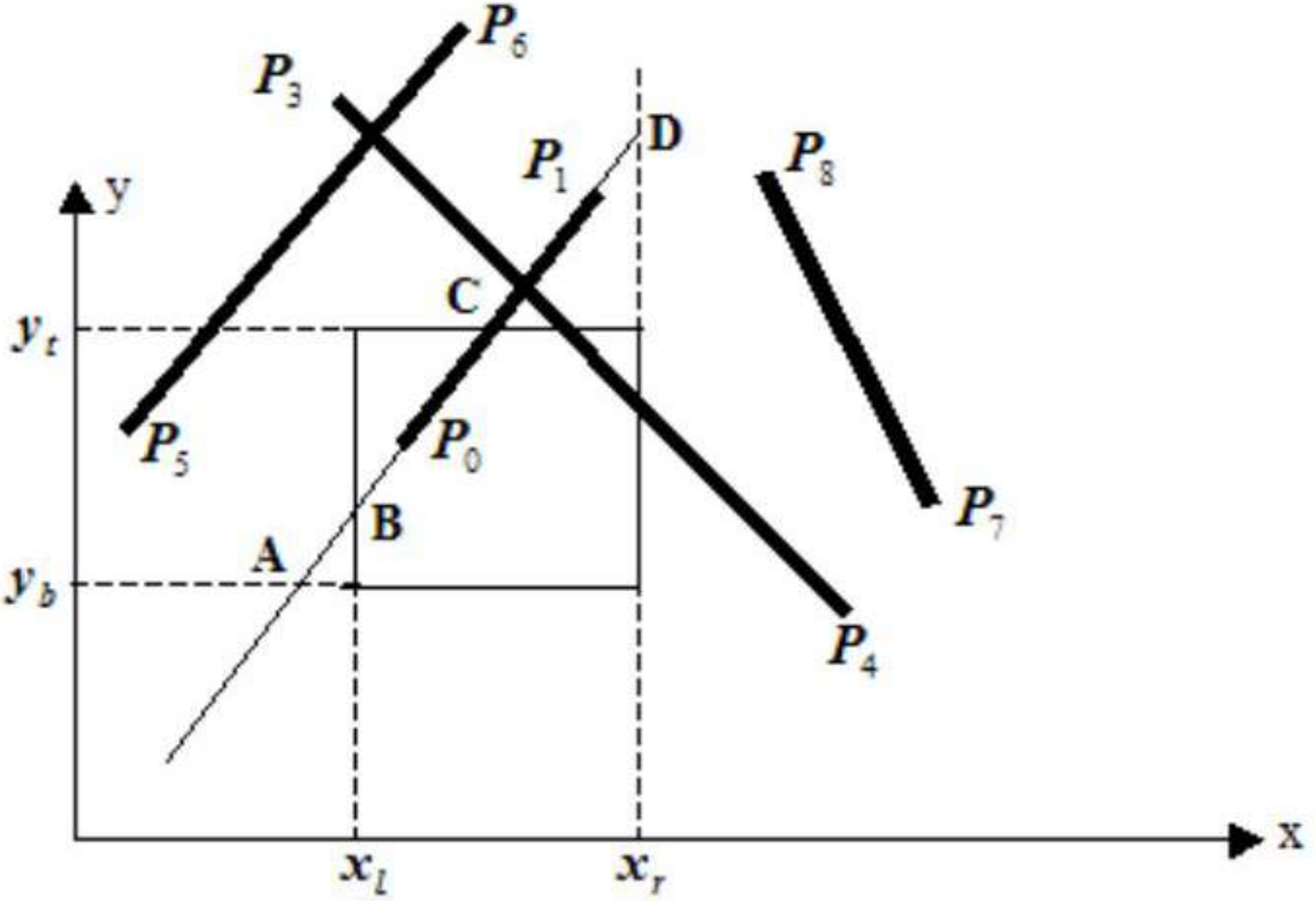


图 3.29 梁友栋-Barsky 裁剪算法

写成参数方程：P_0P_1

$$x = x_0 + \Delta x \bullet t \qquad \Delta x = x_1 - x_0$$

$$y = y_0 + \Delta y \bullet t \qquad \Delta y = y_1 - y_0$$

把窗口边界的四条边分成两类，一类称为始边，另一类称为终边。

当 $\Delta x \geq 0 (\Delta y \geq 0)$ 时，称 $x = x_l (y = y_b)$ 为始边，$x = x_r (y = y_t)$ 为终边。当 $\Delta x < 0 (\Delta y < 0)$ 时，则称 $x = x_r (y = y_t)$ 为始边，$x = x_l (y = y_b)$ 为终边。

求出P_0P_1和两条始边的交点的参数 t'_0 ，t''_0

令： $t_0 = \max(t'_0, t''_0, 0)$

$t \in [t_0, t_1]$

求出P_0P_1和两条终边的交点的参数 t'_1 ，t''_1

令： $t_1 = \min(t'_1, t''_1, 1)$

当$t_1 > t_0$时，参数方程中参数的线段就是的可见部分。当$t_0 > t_1$时，整个直线段为不可见

为了确定始边和终边，并求出P_0P_1与它们的交点，可采用如下方法，令：

$$Q_l = -\Delta x, \qquad D_l = x_0 - x_l$$

$$Q_r = \Delta x, \qquad D_r = x_r - x_0$$

$$Q_b = -\Delta y, \qquad D_b = y_0 - y_b$$

$$Q_t = \Delta y, \qquad D_t = y_t - y_0$$

交点的参数为：

$$t_i = \frac{D_i}{Q_i}, \quad i = l, r, b, t$$

当$Q_i<0$时，求得的t_i必定是P_0P_1和始边的交点的参数。当$Q_i>0$时，t_i则是P_0P_1和终边的交点的参数。 $Q_i=0$时，若$D_i<0$，则p0p1是完全不可见的（如图3.30中的直线段AB,使 $Q_r=0,D_r\geq 0$ ）。

$Q_i=0,D_i\geq 0$时， 如CD,$Q_l=0,D_l>0;Q_r=0,D_r>0$

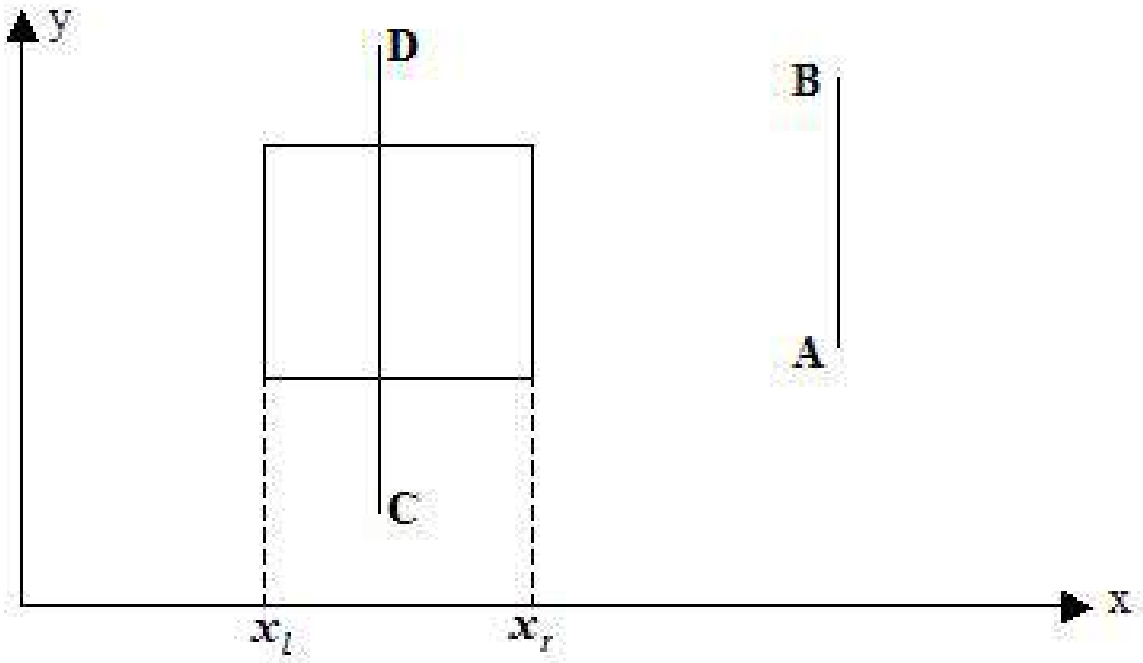


图 3.30 $Q_i = 0$ 的情况

算法的程序实现如下

函数L_Barsky实现算法

函数cansee用于判断直线段是否可见。

double xl, xr, yt, yb;（窗口位置）

```
void L_Barsky(double x0, y0, x2, y2)
{
double t0, t1, deltax, deltay;
t0=0.0; t1=1.0;
deltax=x2-x0;
if (!cansee(-deltax, x0-x1, t0, t1)) return;
```

```
if (!cansee(deltax, xr-x0, t0, t1)) return;
deltay=y2-y0;
if (!cansee(-deltay, y0-yb, t0, t1)) return;
if (!cansee(deltay, yt-y0, t0, t1)) return;
x2=x0+t1*deltax;
y2=y0+t1*deltay;
x0=x0+t0*deltax;
y0=y0+t0*deltay;
showline(x0, y0, x2, y2);   //显示可见线段
}
```

```
bool cansee(double q, d, t0, t1)
{    double r;
      if (q<0)
     {    r=d/q;
          if (r>tl) {cansee=false; return;}
          else if (r>t0) t0=r;
     }
   else if (q>0)
    {    r=d/q;
         if (r<t0) {cansee=false; return;}
         else if (r<t1) t1=r;
   }
  else if (d<0) {cansee=false; return;}
        cansee=true;
}
```

其它图形的裁剪
字符、多边形、圆弧和任意曲线

多边形裁剪后，一部分窗口的边界有可能成为裁剪后多边形的边界，而一个凹多边形裁剪后可能成为几个多边形

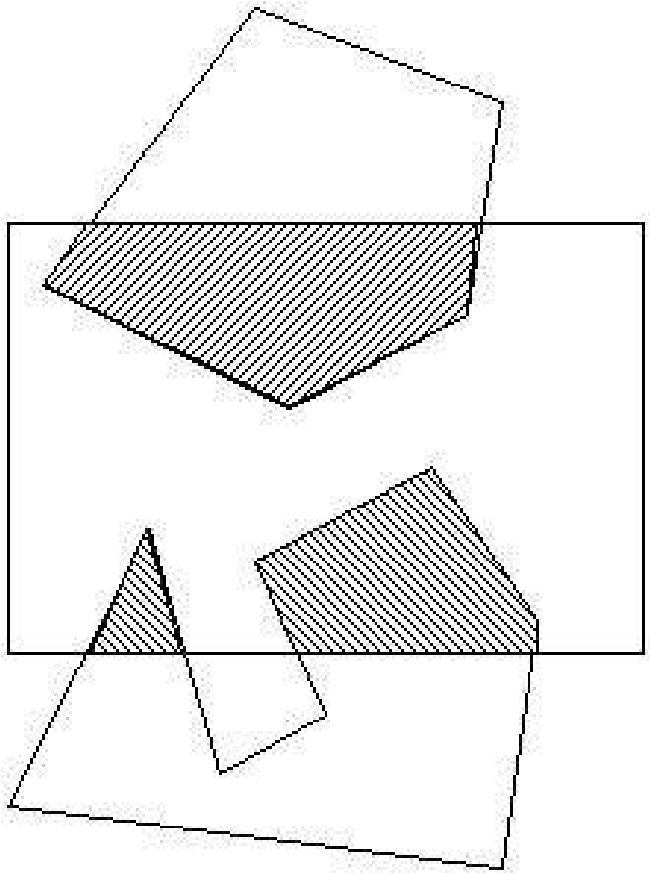


图 3.31 多边形的裁剪（图中阴影部分是裁剪结果）

对多边形的裁剪可以采用Sutherland-Hodgman算法，只要对多边形用窗口的四条边裁剪四次就可得到裁剪后的多边形

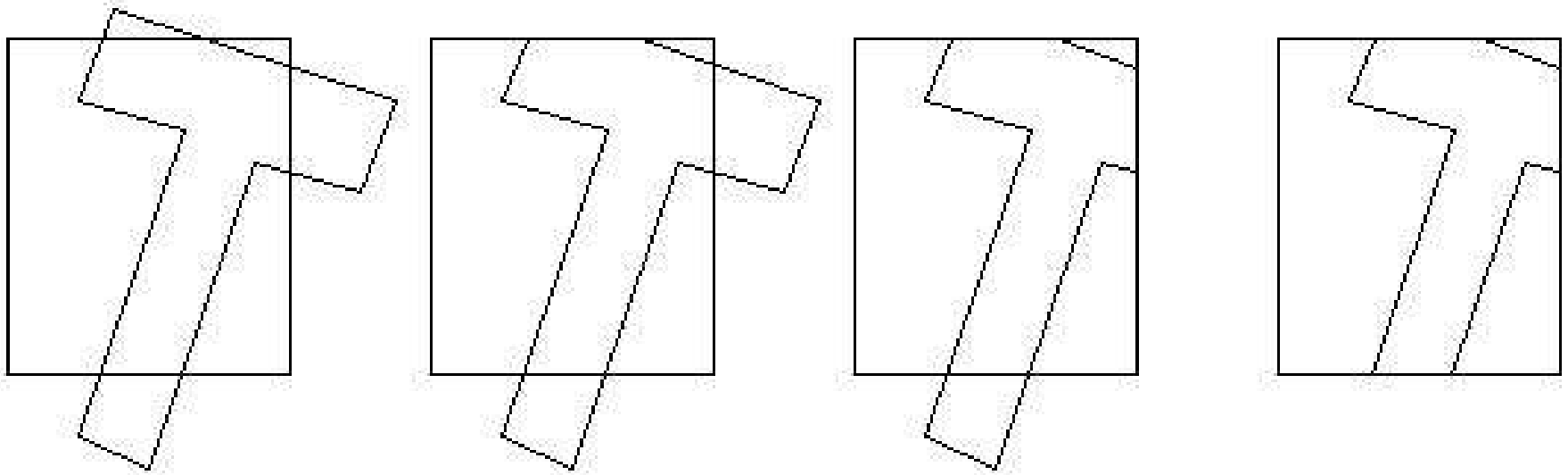


图 3.32 Sutherland-Hodgman 多边形裁剪算法

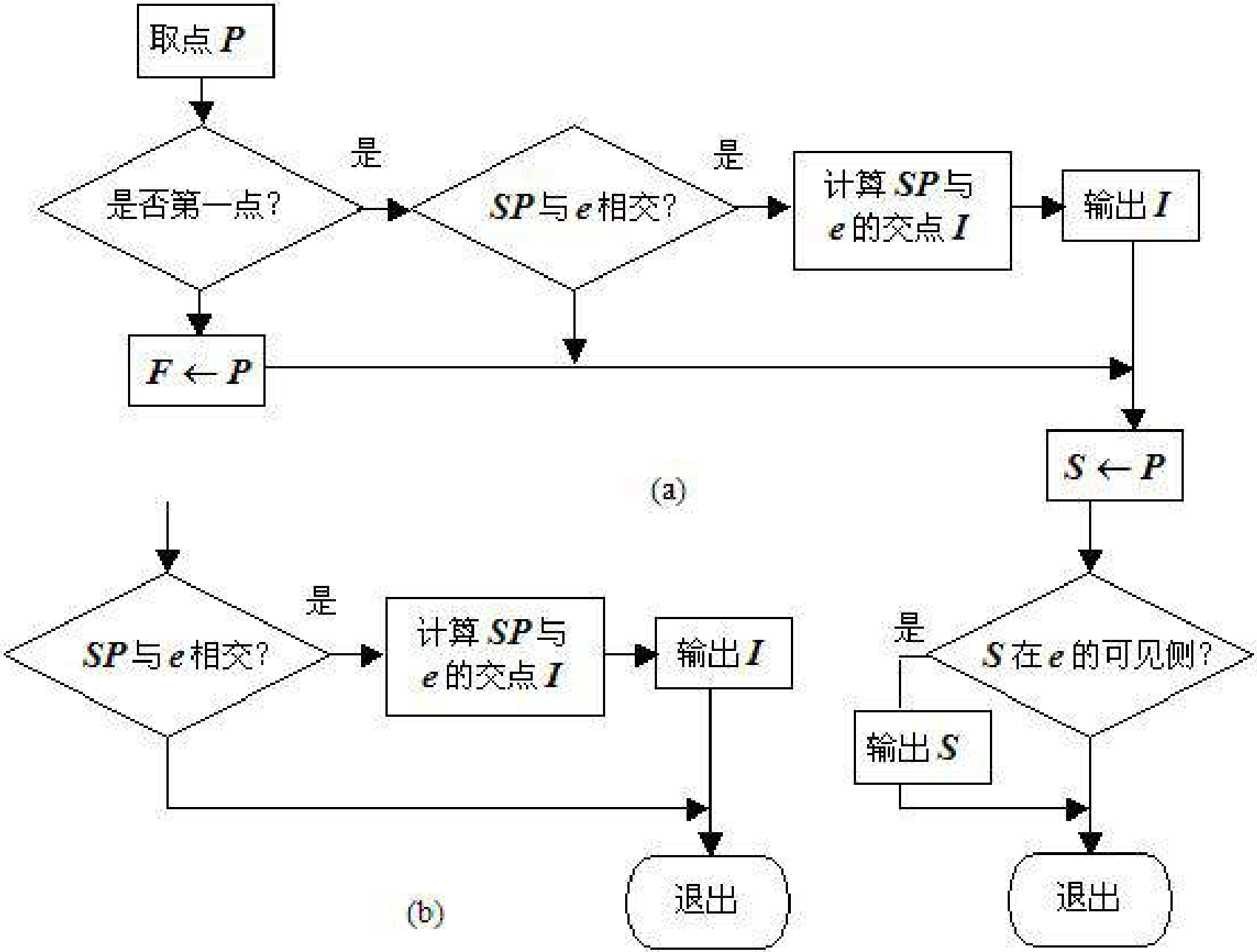


图 3.33 Sutherland-Hodgman 算法的框图

裁剪区域为任意凸多边形区域时的直线段的裁剪算法

1. 计算所要裁剪的直线段所在直线与凸多边形区域的边界直线段的交点。

2. 当交点的个数为0或1时，该直线段处于凸多边形区域外

3. 当交点的个数为2时，通过判断这两个交点与被裁剪的直线段的端点在直线段所处的直线上的关系来进行裁剪。

1)．如果直线段的两个端点和两个交点刚好全部重合，则显而易见该直线段完全可见。

2)． 如果直线段的两个端点和两个交点中有一个

重合，则将重合的点视为一点，则将重合点与另外的一个端点、一个交点按在直线上的顺序排列。根据排列的顺序有如下三种情况：

①重合点和交点不相邻，则直线段完全可见；

②如果重合点与交点相邻，且该交点与另一个端点相邻，则直线段部分可见，且可见部分为重合点与另一个交点确定的直线段，它的可见部分为直线段；

③如果重合点与交点相邻，且该交点与另一个端点不相邻，则直线段完全不可见。

3) 如果直线段的两个端点和两个交点都不重合。将这四个点按在直线上的顺序排列，根据排列的顺序有如下四种情况：

①如果两个端点和两个交点分别相邻，则直线段完全不可见；

②如果两个端点在两个交点之间，则直线段完全可见；

③如果两个交点在两个端点之间，则直线段部分可见，且可见部分为两个交点决定的直线段，它的可见部分为直线段；

④如果交点和端点的排列顺序是交错的，则直线段部分可见，且可见部分为中间的一个交点和一个端点决定的直线段，它的可见部分为直线段 。

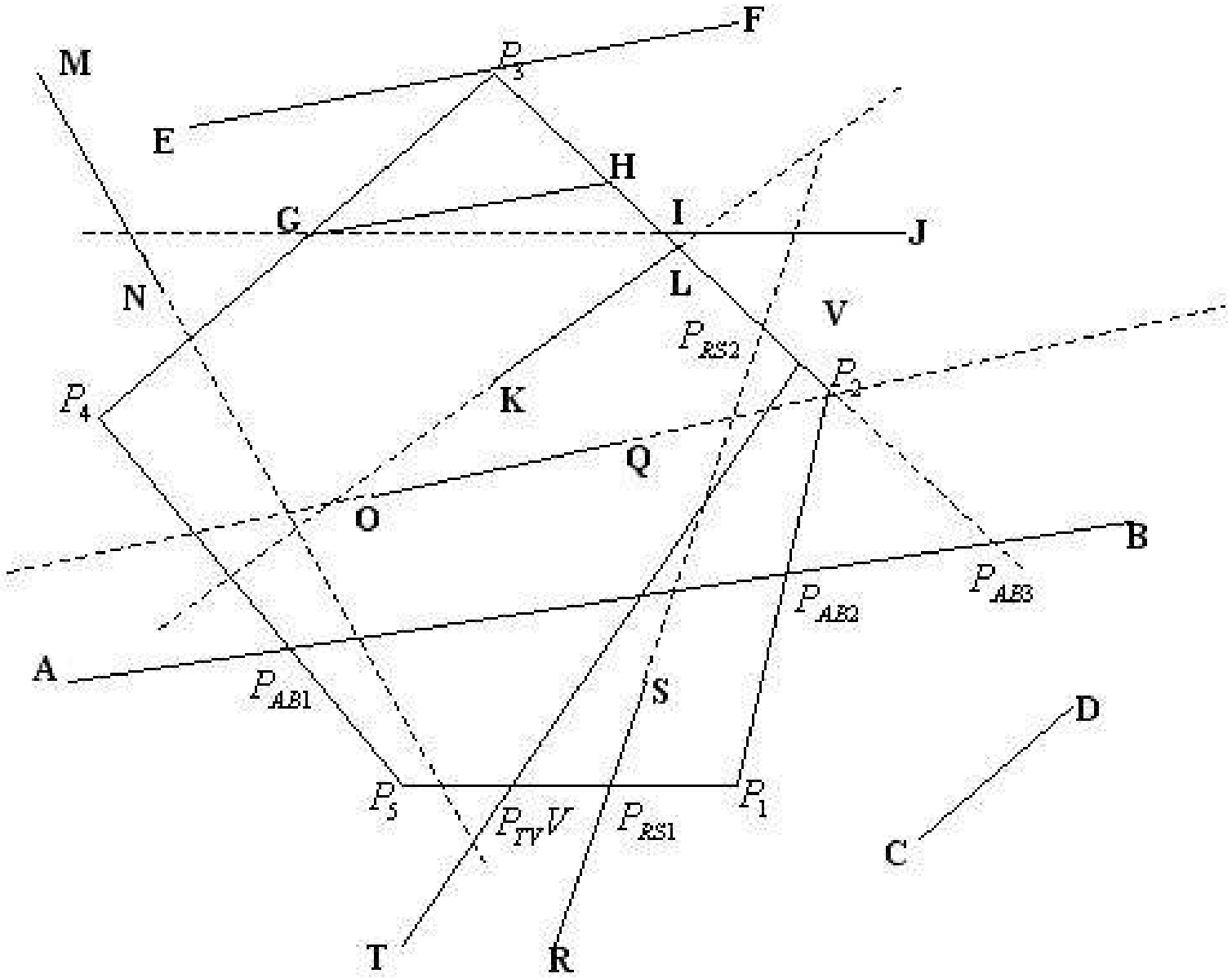


图 3.34 裁剪区域为任意凸多边形区域时对直线段的裁剪

三维图形的裁剪

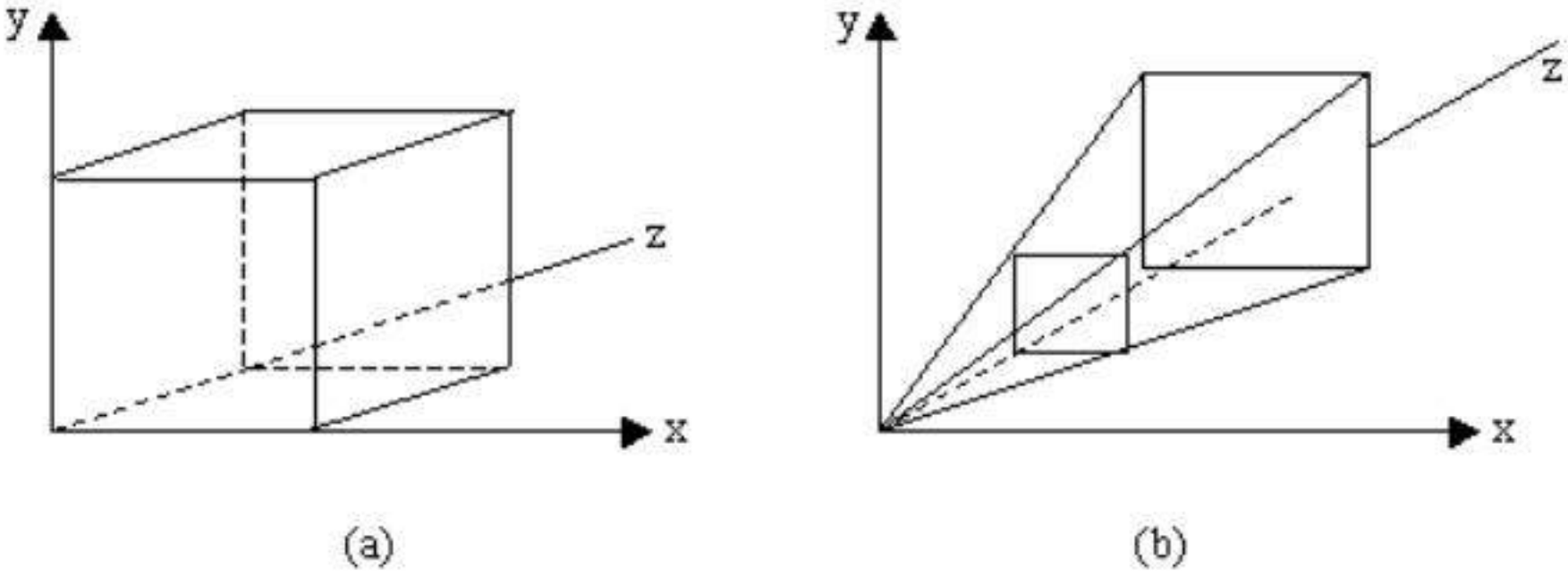


图 3.35 两种三维视域

Cohen-Sutherland算法推广至三维时，用于判断显然不可见的线段的编码应为六位，这六位的安排是：

点在视域上面，第一位为1，$y>1(y>z)$

点在视域下面，第二位为1，$y<0(y<-z)$

点在视域右面，第三位为1，$x>1(x>z)$

点在视域左面，第四位为1，$x<0(x<-z)$

点在视域后面，第五位为1，$z>1(z>1)$

点在视域前面，第六位为1，$z<0$ （$z<z_{min}$）

设直线段的起点和终点分别为$P_0(x_0,y_0,z_0)$和$P_1(x_1,y_1,z_1)$，直线方程可以表示成如下的参数方程形式：

$$x = x_0 + (x_1 - x_0)t$$

$$y = y_0 + (y_1 - y_0)t$$

$$z = z_0 + (z_1 - z_0)t$$

y=1求交时，可由下式求得：

$$1=(y_1-y_0)t'+y_0 \qquad t'=\frac{1-y_0}{y_1-y_0}$$

直线段和平面x=z的交点，可得出交点处的参数为：

$$t'=\frac{z_0-x_0}{(x_1-x_0)-(z_1-z_0)}$$

梁友栋-Barsky算法也可以推广到三维情况下。当视域为立方体时，这种推广是直接的。当视域为棱台时，对于$x = \pm z$，$y = \pm z$ 四个平面来说，对应于二维裁剪时的Q值和D值可如下取值：

$$Q_l = -(\Delta x + \Delta z), \qquad D_l = z_0 + x_0$$

$$Q_r = (\Delta x - \Delta z), \qquad D_r = z_0 - x_0$$

$$Q_b = -(\Delta y + \Delta z), \qquad D_b = y_0 + z_0$$

$$Q_t = (\Delta y - \Delta z), \qquad D_t = z_0 - y_0$$

对z=1和z= z_{min}两个平面，相应的 Q 值和 D 值可如下取值：

$$Q_f = -\Delta z, \quad D_f = z_0 - z_{\min}$$

$$Q_{ba} = \Delta z, \quad D_{ba} = 1 - z_0$$

其中，$\Delta x = x_1 - x_0$ $\Delta y = y_1 - y_0$ 和 $\Delta z = z_1 - z_0$ 可知，相应平面与 P_0P_1 的交点的参数值为，。

$$t_i = \frac{D_i}{Q_i} \qquad (i = l, r, b, t, f, ba)$$

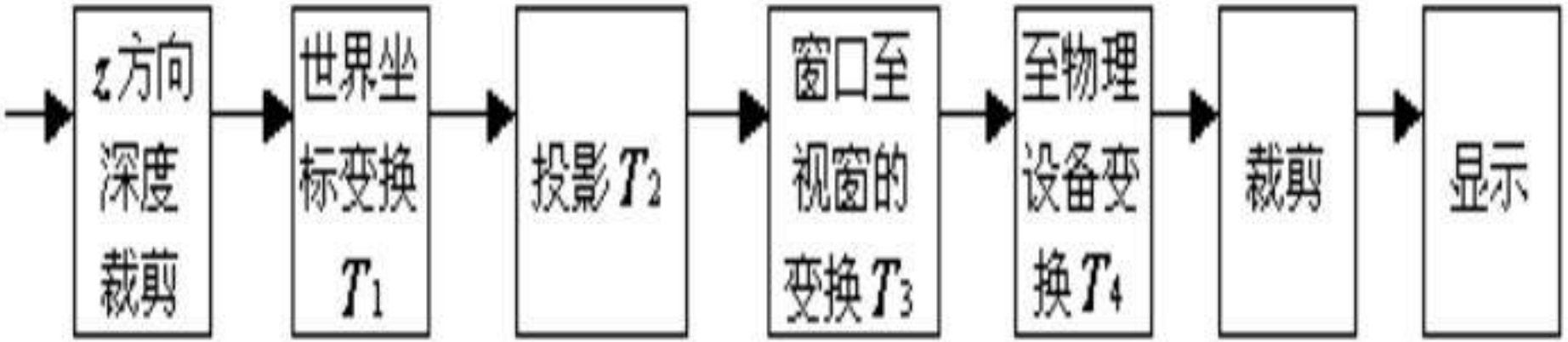


图 3.36 简单的图形处理流程

第三章　　作业

16题：P1(1,1,1),P2(2,2,1+$\sqrt{2}$),P3(3,3,3)

解法一：

①平移T(−1, −1, −1)

(0, 0, 0), (1, 1, $\sqrt{2}$), (2, 2, 2)

②绕Z轴旋转　α　角，

$$\sin\alpha=\frac{\sqrt{2}}{2},\cos\alpha=\frac{\sqrt{2}}{2}$$

$$\begin{bmatrix}1 & 1 & \sqrt{2} & 1\\ 2 & 2 & 2 & 1\end{bmatrix}\begin{bmatrix}\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0\\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}0 & \sqrt{2} & \sqrt{2} & 1\\ 0 & 2\sqrt{2} & 2 & 1\end{bmatrix}$$

③绕X轴旋转－135度

$$\sin 135^o = -\frac{\sqrt{2}}{2}, \cos 135^o = \frac{\sqrt{2}}{2}$$

$$\begin{bmatrix} 0 & \sqrt{2} & \sqrt{2} & 1 \\ 0 & 2\sqrt{2} & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -2 & 1 \\ 0 & -2+\sqrt{2} & -2-\sqrt{2} & 1 \end{bmatrix}$$

④绕Z轴旋转180度

$$\sin 180^o = 0, \cos 180^o = -1$$

$$\begin{bmatrix} 0 & 0 & -2 & 1 \\ 0 & -2+\sqrt{2} & -2-\sqrt{2} & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -2 & 1 \\ 0 & 2-\sqrt{2} & -2-\sqrt{2} & 1 \end{bmatrix}$$

$$H = T(-1,-1,-1)R_z(45^o)R_x(-135^o)R_z(180^o)$$

$$= \begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{1}{2} & -\frac{1}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{1}{2} & -\frac{1}{2} & 0 \\ 0 & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2}-1 & \frac{\sqrt{2}}{2}+1 & 1 \end{bmatrix}$$

解法二：

①$\mathrm{T}(-1,-1,-1)$

② $R_z(-135^o)$

$$\begin{bmatrix} 0 & -\sqrt{2} & \sqrt{2} & 1 \\ 0 & -2\sqrt{2} & 2 & 1 \end{bmatrix}$$

③ $R_x(135^o)$

$$\begin{bmatrix} 0 & 0 & -2 & 1 \\ 0 & 2-\sqrt{2} & -2-\sqrt{2} & 1 \end{bmatrix}$$

解法三：

①$T(-1,-1,-1)$

②$R_x(\alpha), \sin\alpha = \frac{1}{\sqrt{3}}, \cos\alpha = \frac{\sqrt{2}}{\sqrt{3}}$

③ $R_y(\beta), \sin\beta = \frac{1}{2}, \cos\beta = -\frac{\sqrt{3}}{2}$

④$R_z(\theta), \sin\theta = -\frac{1}{\sqrt{3}}, \cos\theta = \frac{\sqrt{2}}{\sqrt{3}}$

解法四：

$$H = T(-1,-1,-1)R_z(-45^\circ)R_y(135^\circ)R_z(-90^\circ)$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & \sqrt{2} \\ 2 & 2 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 \\ \sqrt{2} & 0 & \sqrt{2} \\ 2\sqrt{2} & 0 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -2 \\ -2 & 0 & -2-\sqrt{2} \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -2 \\ 0 & 2-\sqrt{2} & -2-\sqrt{2} \end{bmatrix}$$

19题：平面方程Ax+By+Cz+D=0

法向（A,B,C)

a.....设$D \neq 0$,又$\because A,B,C$不能全为0，设$A \neq 0$,则（$-\frac{D}{A}$,0,0)是平面上的点

$T\left(\frac{D}{A},0,0\right)$

b　做旋转变换使平面的法 向量与z轴重合

$\mathrm{R_x}(\alpha)$,令$v=\sqrt{B^2+C^2}$,$\cos\alpha=\frac{C}{v}$,$\sin\alpha=\frac{B}{v}$

$\mathrm{R_y}(\beta)$,令$d=\sqrt{A^2+B^2+C^2}$,$\cos\beta=\frac{v}{d}$,$\sin\beta=-\frac{A}{d}$

$$\mathrm{H}=\mathrm{T}(\frac{\mathrm{D}}{\mathrm{A}},0,0)\bullet\mathrm{R_x}(\alpha)\bullet\mathrm{R_y}(\beta)=\begin{bmatrix}\frac{v}{d} & 0 & -\frac{A}{d} & 0\\ \frac{AB}{vd} & \frac{C}{v} & \frac{B}{d} & 0\\ \frac{AC}{vd} & -\frac{B}{v} & \frac{C}{d} & 0\\ \frac{Av}{Bd} & 0 & -\frac{A^2}{Bd} & 1\end{bmatrix}$$

22题

设定任一点（x,y,z)投影后的点为（xp,yp,zp),则

（xp-x,yp-y,zp-z)=k(l,m,n)

xp=x+kl

yp=y+km

zp=z+kn

因为投影平面为z=0平面，

z+kn=0,

k=-z/n

xp=x-lz/n

yp=y-mz/n

变换矩阵为

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{l}{n} & -\frac{m}{n} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

24题：

① 绕y轴顺时针旋转α角（$\alpha = 45^o$）

$$\begin{pmatrix}1 & 1 & 1 & 1\end{pmatrix}\begin{bmatrix}\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0\\ 0 & 1 & 0 & 0\\ -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0\\ 0 & 0 & 0 & 1\end{bmatrix}=\begin{pmatrix}0 & 1 & \sqrt{2} & 1\end{pmatrix}$$

② 绕x轴逆时针旋转β角，$\cos\beta = \frac{\sqrt{2}}{\sqrt{3}}, \sin\beta = \frac{1}{\sqrt{3}}$

$$\begin{pmatrix}0 & 1 & \sqrt{2} & 1\end{pmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \frac{\sqrt{2}}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0\\ 0 & -\frac{1}{\sqrt{3}} & \frac{\sqrt{2}}{\sqrt{3}} & 0\\ 0 & 0 & 0 & 1\end{bmatrix}=\begin{pmatrix}0 & 0 & \sqrt{3} & 1\end{pmatrix}$$

③ 做平行投影（投影平面为Z=0平面，方向为Z轴方向）

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

④ 做②的逆变换

⑤ 做①的逆变换

$$H = R_y(-45^o) \bullet R_x(\beta) \bullet M \bullet R_x(-\beta) R_y(45^o)$$

第四章　曲线和曲面

第一节 曲线和曲面表示的基础知识

第二节 Hermite多项式

第三节 Coons曲面

第四节 Bezier曲线和曲面

第五节 B样条曲线和曲面

第一节 曲线和曲面表示的基础知识

1.显式、隐式和参数表示

①显式： $y=f(x)$

不能表示封闭曲线或多值曲线（圆）

②隐式：$f(x,y)=0, f(x,y,z)=0$

球面

显式隐式表示的缺点

（1）与坐标轴相关的，不便于坐标变换；

（2）无法解决斜率为无穷大的情况；

（3）对于空间复杂曲线曲面很难表示

（4）不便于计算和编程

③ 参数表示（解决上述问题）

空间曲线：

$$x = x(t) \quad y = y(t) \quad z = z(t)$$

矢量表示是：

$$\mathbf{P}(t) = [x(t) \quad y(t) \quad z(t)]$$

对参数t求导：

$$\mathbf{P}'(t) = [x'(t) \quad y'(t) \quad z'(t)]$$

曲面：

$$x = x(u,w), y = y(u,w), z = z(u,w)$$

$$\mathbf{P}(u,w) = [x(u,w), y(u,w), z(u,w)]$$

曲线或曲面的某一部分，可以简单地用$a \leqslant t \leqslant b$界定它的范围

空间直线段

$P_1[x_1, y_1], P_2[x_2, y_2]$

$$x = x_1 + (x_2 - x_1)t$$

$$y = y_1 + (y_2 - y_1)t$$

$$z = z_1 + (z_2 - z_1)t$$

$$0 \le t \le 1$$

P(t)代表曲线上的一点

$P(t) = P_1 + (P_2 - P_1)\ t = (1-t)P_1 + tP_2$

$$P'(t) = \begin{bmatrix} x'(t) & y'(t) & z'(t) \end{bmatrix} = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \end{bmatrix}$$

参数方程具有如下优点：

(1)有更大的自由度来控制曲线、曲面的形状。
(2)便于坐标变换
(3)便于处理斜率为无限大的问题，不会因此中断计算
(4) 代数、几何相关和无关的变量是完全分离的，而且对变量个数不限，便于向高维空间扩展。
(5) $t\in[0,1]$，直接定义了边界。便于曲线和曲面的分段、分片描述。
(6) 易于用矢量和矩阵表示，从而简化了计算。

- **基本概念**

1.插值:要求构造一条曲线顺序通过型值点，称为对这些型值点进行插（interpolation）。

给定函数f(x)在区间[a,b]中互异的n个点的值f(xi) i=1,2,…n,基于这些数据寻找某一个函数 $\varphi(x)$ **，要求** $\varphi(x_i)=f(x_i)$ **，** $\varphi(x)$ **为f(x）的插值函数，xi为插值节点**

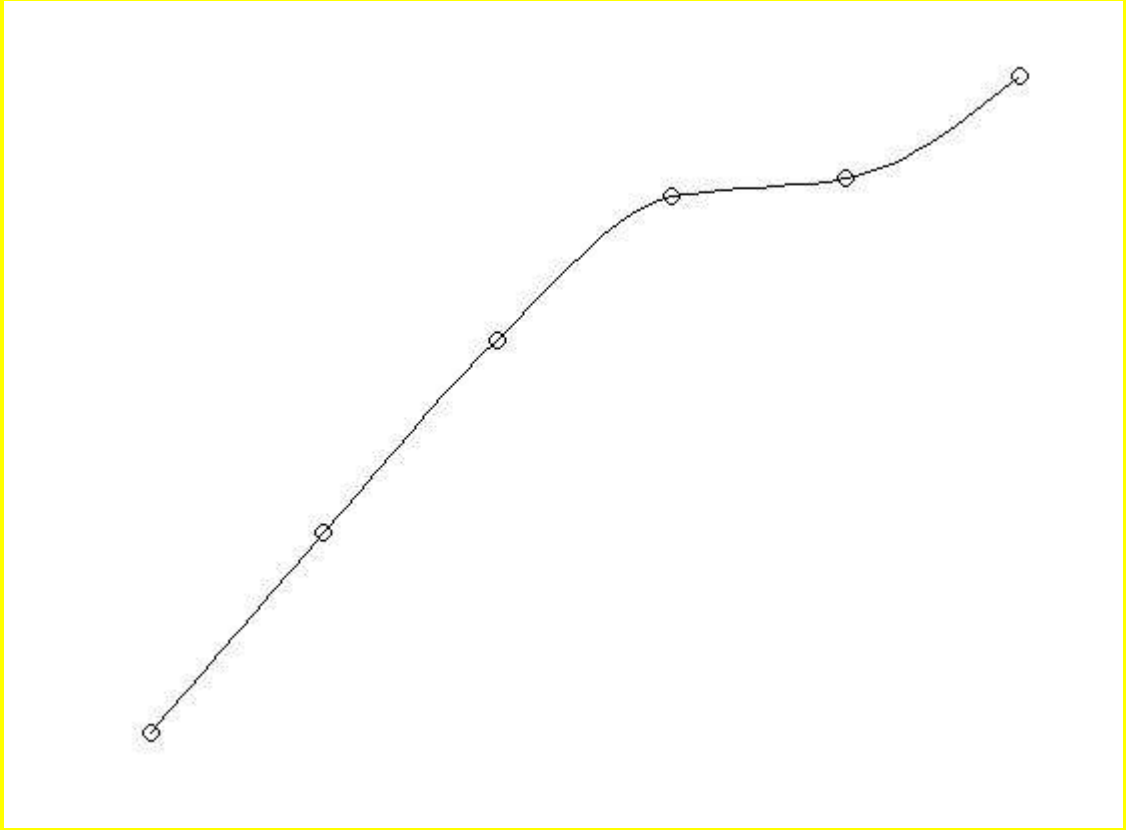
(1)线性插值

函数f(x)在两个不同点x1,x2的值，y1=f(x1),y2=f(x2) 用线性函数 $y=\varphi(x)=ax+b$ **近似代替y=f(x),选择a,b使** $\varphi(x_1)=y_1,\varphi(x_2)=y_2$ **则称** $\varphi(x)$ **为f(x)的线性插值函数**

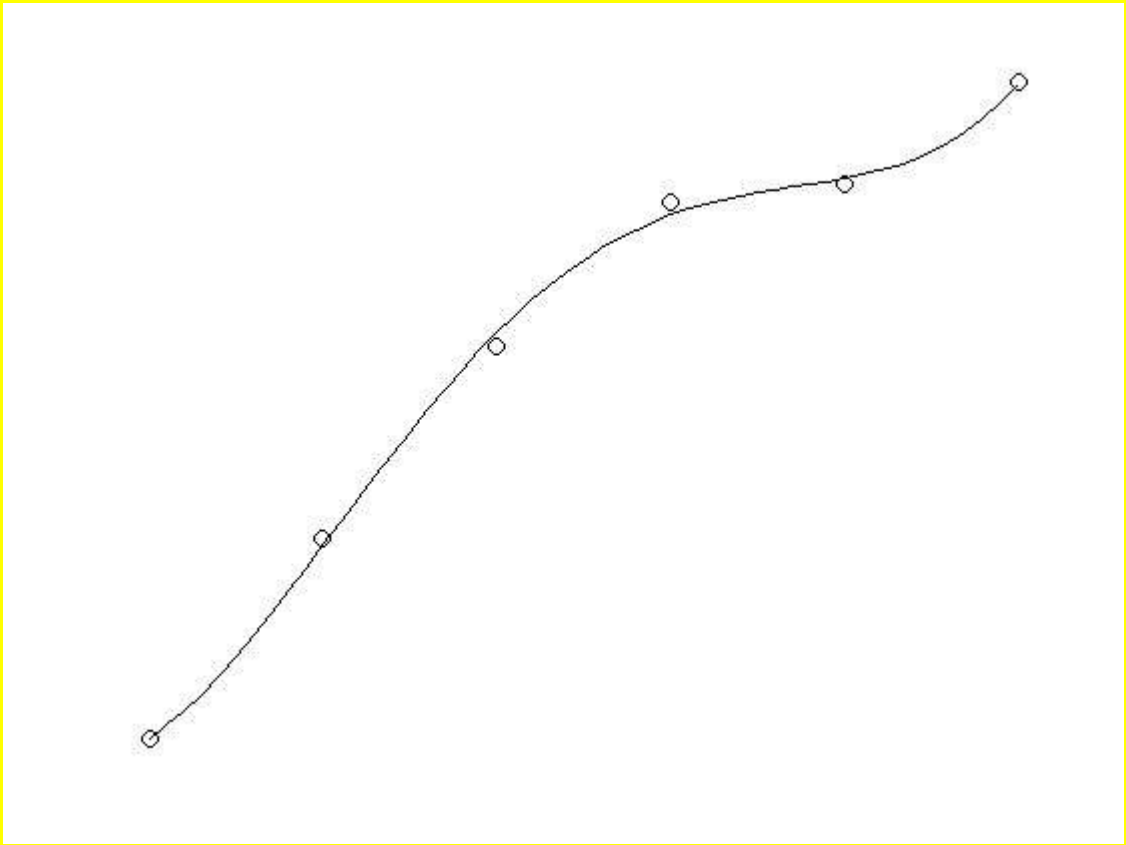
(2)抛物线插值(二次插值）

设已知f(x)在三个互异点x1,x2,x3的函数值为y1,y2,y3,要求构造函数 $\varphi(x) = ax^2 + bx + c, \varphi(x)$

在节点xi处有 $\varphi(x_i) = f(x_i)$



2, 逼近　构造一条曲线，使它在某种意义上最佳逼近这些型值点，称之为对这些型值点进行逼近（approximation）。



常用方法 最小二乘法

假设已知一组型值点（xi,yi), i=1,2,…n,要求构造一个m（m<n-1)次多项式函数y=F(x）逼近这些型值点。

偏差的平方和最小：

$$\varphi = \sum_{k=1}^{n} (F(x_k) - y_k)^2,$$

或加权平方和最小：

$$\varphi = \sum_{k=1}^{n} d_k (F(x_k) - y_k)^2$$

令F(x)为一个m次多项式

$$F(x) = \sum_{i=0}^{m} a_j x^j$$

最小二乘法就是定出ai使偏差平方和最小

3. 参数曲线的代数形式和几何形式

一条三次参数曲线的代数形式：

$$\begin{cases} x(t) = a_{3x}t^3 + a_{2x}t^2 + a_{1x}t + a_{0x} \\ y(t) = a_{3y}t^3 + a_{2y}t^2 + a_{1y}t + a_{0y} \\ z(t) = a_{3z}t^3 + a_{2z}t^2 + a_{1z}t + a_{0z} \end{cases}$$

$$0 \le t \le 1$$

矢量形式：$P(t) = a_3t^3 + a_2t^2 + a_1t + a_0$

给定P(0), P(1), 以及P’(0), P’(1)

$$P(0) = a_0, P(1) = a_3 + a_2 + a_1 + a_0$$

$$P'(0) = a_1, P'(1) = a_1 + 2a_2 + 3a_3$$

解四个方程，求得参数

$$a_0 = P(0), a_1 = P'(0)$$

$$a_2 = -3P(0) + 3P(1) - 2P'(0) - P'(1)$$

$$a_3 = 2P(0) - 2P(1) + P'(0) + P'(1)$$

令 $P_0 = P(0), P_1 = P(1), P_0' = P'(0), P_1' = P'(1)$

$$P(t) = ()t^3 + ()t^2 + ()t + a_0$$

$$= (2t^3 - 3t^2 + 1)P_0 + (-2t^3 + 3t^2)P_1 +$$

$$(t^3 - 2t^2 + t)P_0' + (t^3 - t^2)P_1'$$

$$P(t) = F_1P_0 + F_2P_1 + F_3P_0' + F_4P_1'$$

参数曲线的几何形式 $P(t) = F_1P_0 + F_2P_1 + F_3P_0' + F_4P_1'$

矩阵形式表示参数曲线：

P=TA 其中 $T = \begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix}$

$A = \begin{pmatrix} a_3 & a_2 & a_1 & 1 \end{pmatrix}^T$

P=FB $F = \begin{pmatrix} F_1 & F_2 & F_3 & F_4 \end{pmatrix}$

$B = \begin{pmatrix} P_0 & P_1 & P_0' & P_1' \end{pmatrix}$

$$F = \begin{pmatrix} t^3 & t^2 & t^1 & 1 \end{pmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$= TM$$

$$P = TMB \ , A = MB \ , B = M^{-1} A$$

P=TMB表示一条参数曲线

- **参数连续性**

  一函数在某一点x_0处具有相等的直到k阶的左右导数，称它在x_0处是k次连续可微的，或称它在x_0处是k阶连续的，记作C^k。几何上C^0、C^1、C^2依次表示该函数的图形、切线方向、曲率是连续的。参数曲线的可微性称为参数曲线的连续性。

- **几何连续性**

  两曲线段的相应的弧长参数化在公共连接点处具有C^k连续性，则称它们在该点处具有k阶几何连续性，记作G^k 。零阶几何连续G^0与零阶参数连续C^0是一致的。一阶几何连续G^1指一阶导数在两个相邻曲线段的交点处成比例，即方向相同，大小不同。二阶几何连续G^2指两个曲线段在交点处其一阶和二阶导数均成比例。

4.曲线段间 C^1, C^2 和 G^1, G^2 连续性定义

（1）Q1(1)=Q2(0),则Q1(t)和Q2(t)在P处有C^0 和 G^0 连续性

（2） Q1(1)和Q2(0）在P处重合，且其在P点处的切矢量方向相同，大小不等， 则Q1(t)和Q2(t)在P处有 G^1 连续性

（3） Q1(1)和Q2(0）在P处重合，且其在P点处的切矢量方向相同，大小相等，则Q1(t)和Q2(t)在P处有 C^1 连续性

(4) Q1(1)和Q2(0）在P处已有C^0, C^1连续 且 $Q_1''(1) and Q_2''(0)$ 大小方向均相同，则Q1(t)和Q2(t)在P处有 C^2 连续性

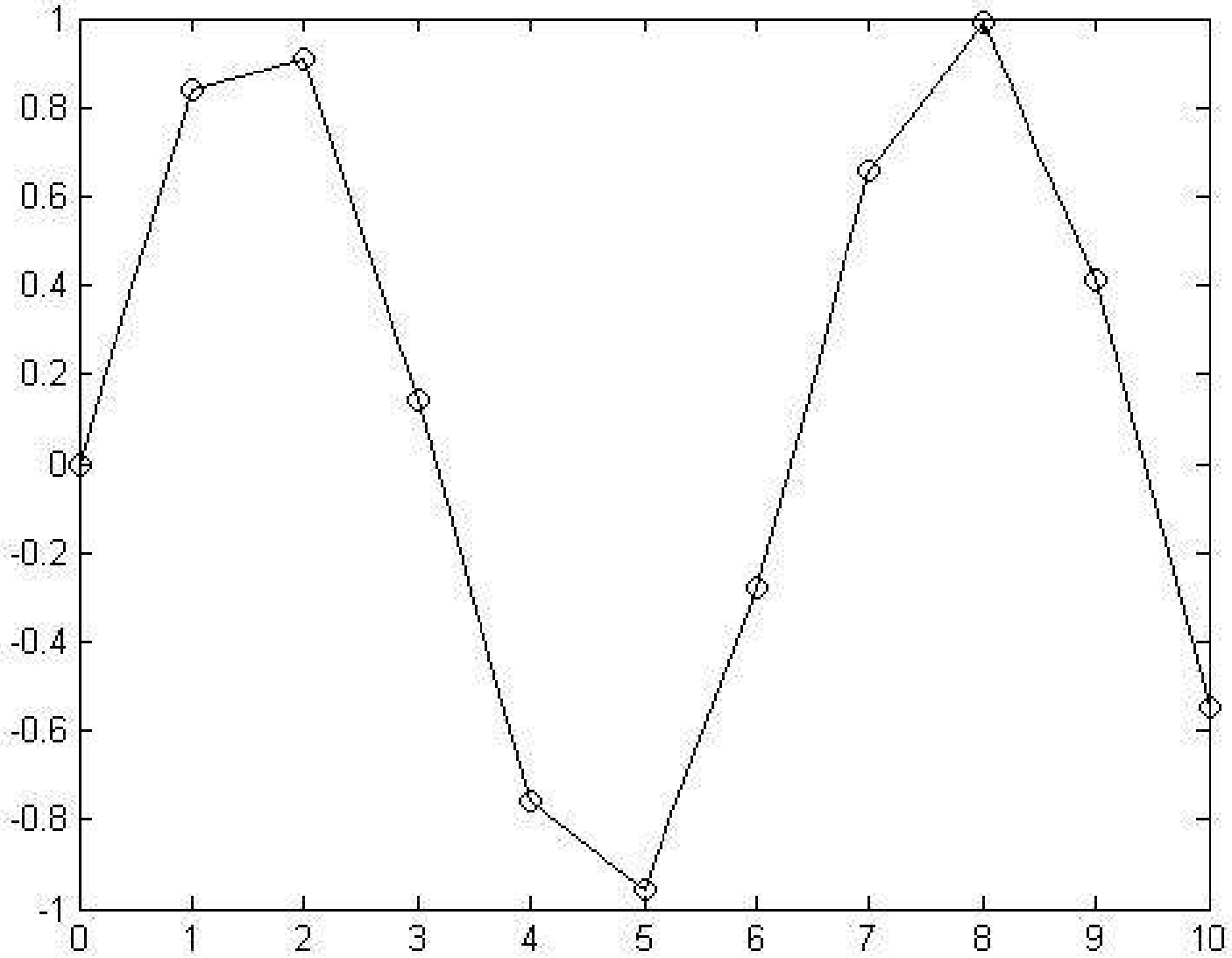
推广之，Q1(1)和Q2(0）在P处$\bar{\in} C^0, C^1 \cdots C^n$ 连续，若

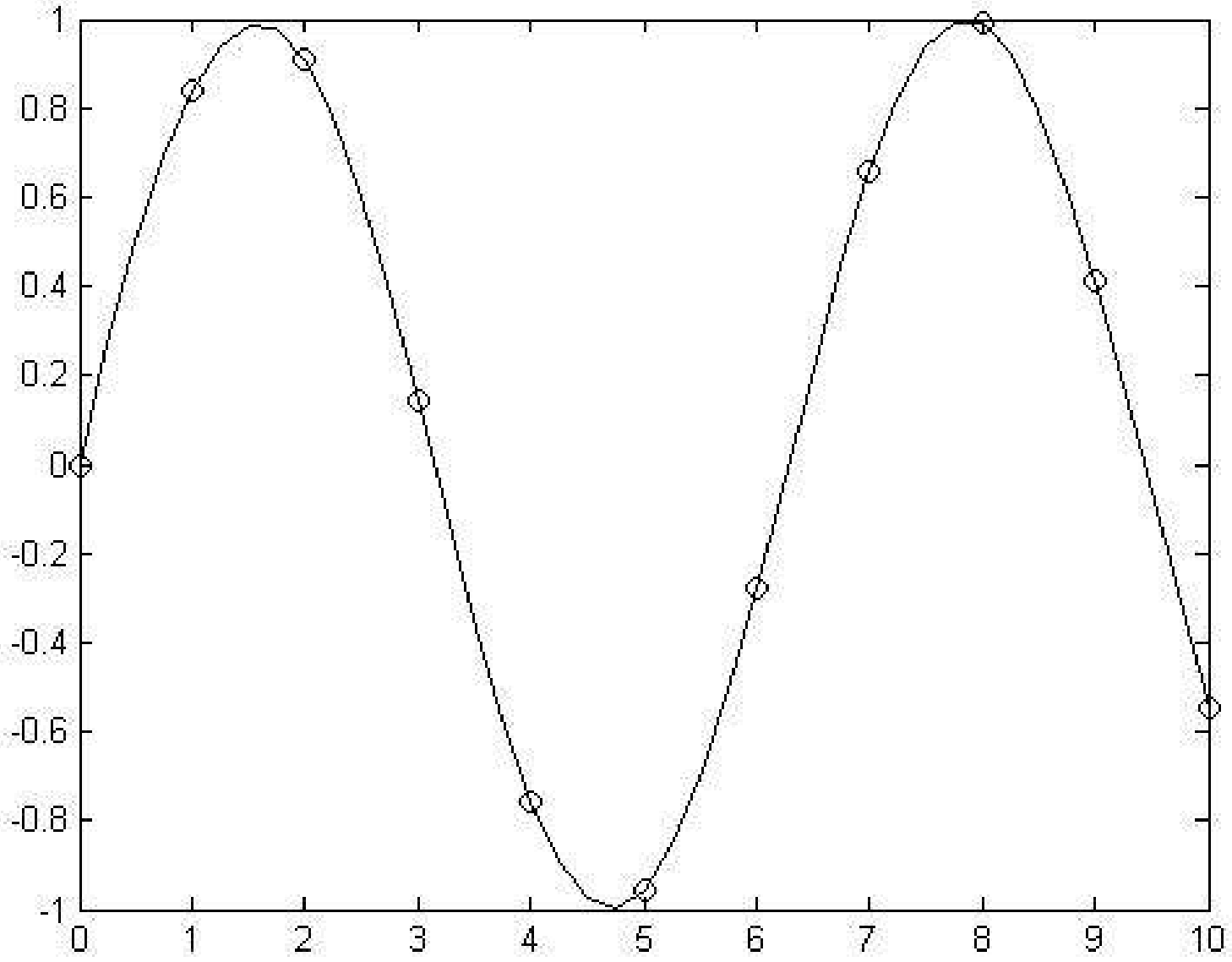
$Q_1^n and Q_2^n$ 在P处大小和方向均相同，则说Q1(t）和Q2(t)在P处具有 C^n 连续性

(5). Q1(1)和Q2(0）在P处已有 G^0, G^1连续 且 $Q_1''(1) and Q_2''(0)$ 方向相同但大小不等，则Q1(t)和Q2(t)在P处有 G_2 连续性

5. 光顺

光顺（smoothness）是指曲线的拐点不能太多，要光滑顺畅。对于平面曲线相对光顺的条件应该是：（1）具有二阶几何连续（G^2）；（2）不存在多余拐点和奇异点；（3）曲率变化较小


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7
8
9
10


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7
8
9
10

第二节　Hermite多项式

已知函数$f(t)$在$k+1$个点$\{t_i\}$处的函数值和导数值$\{f^{(j)}(t_i)\}$，$i=0, 1, \ldots, k$，$j=0, 1, \ldots, m_i-1$，要求确定一个$N = m_0 + m_1 + \ldots + m_k - 1$次的多项式$P(t)$，满足下面的插值条件：

$$P^{(j)}(t_i)=f^{(j)}(t_i)$$

1. Lagrange插值法($m_0 = m_1 = \cdots = m_k = 1$)

已知f(t)在k+1个点的函数值f (t_i) ,求一个k次多项式使p(t_i)=f (t_i)

$$p(t) = \sum_{i=0}^{k} f(t_i) \cdot l_i$$

$$l_i(t) = \frac{(t-t_0)\cdots(t-t_{i-1})\cdot(t-t_{i+1})\cdots(t-t_k)}{(t_i-t_0)\cdots(t_i-t_{i-1})\cdot(t_i-t_{i+1})\cdots(t_i-t_k)}$$

$$\begin{cases} 1 & t = t_i \\ 0 & t = t_j, j \neq i \end{cases}$$ 调和函数

例： k=2, m0=m1=m2=1

已知函数$f(t)$在三个点t_0，t_1，t_2，的函数值$f(t_0)$，$f(t_1)$, $f(t_2)$，求三次多项式P(t)

$$P_0(t)=f(t_0)\cdot l_0(t)+f(t_1)\cdot l_1(t)+f(t_2)\cdot l_2(t)$$

混合函数如下：

$$
\begin{cases}
l_0(t)=\dfrac{(t-t_1)(t-t_2)}{(t_0-t_1)(t_0-t_2)} \\
l_1(t)=\dfrac{(t-t_0)(t-t_2)}{(t_1-t_0)(t_1-t_2)} \\
l_2(t)=\dfrac{(t-t_0)(t-t_1)}{(t_2-t_0)(t_2-t_1)}
\end{cases}
$$

2. $k=1, m_0=m_1=2, t\in[t_0,t_1]$

已知表示一条曲线的某个函数$f(t)$在两点t_0，t_1的函数值$f(t_0)$，$f(t_1)$和一阶导数值$f'(t_0)$，$f'(t_1)$，求三次多项式P(t)：

$$P(t)=a_3t^3+a_2t^2+a_1t+a_0 \quad t\in[t_0,t_1]$$

$$P(t_0)=f(t_0), P(t_1)=f(t_1),$$

$$P'(t_0)=f'(t_0), P'(t_1)=f'(t_1)$$

$$f(t_0)=a_3t_0^3+a_2t_0^2+a_1t_0+a_0$$

$$f(t_1)=a_3t_1^3+a_2t_1^2+a_1t_1+a_0$$

$$f'(t_0)=3a_3t_0^2+2a_2t_0+a_1$$

$$f'(t_1)=3a_3t_1^2+2a_2t_1+a_1$$

$$a_0 = -\frac{(f'(t_0)t_1^3 + 3f(t_0)t_1^2)t_0 - f(t_0)t_1^3}{(t_1 - t_0)^3}$$

$$-\frac{(-f'(t_1)t_1 + f(t_1))t_0^3 + (-f'(t_0)t_1^2 + f'(t_1)t_1^2 - 3f(t_1)t_1)t_0^2}{(t_1 - t_0)^3}$$

$$a_1 = \frac{+(2f'(t_1)t_1^2 + f'(t_0)t_1^2 - 6f(t_1)t_1 + 6f(t_0)t_1)t_0 + f'(t_0)t_1^3}{(t_1 - t_0)^3}$$

$$+\frac{-f'(t_1)t_0^3 + (-f'(t_1)t_1 - 2f'(t_0)t_1)t_0^2}{(t_1 - t_0)^3}$$

$$a_2 = -\frac{f'(t_1)t_1^2 + 2f(t_0)t_1^2 + 3f(t_0)t_1 - 3f(t_1)t_1}{(t_1 - t_0)^3}$$

$$- \frac{(-f'(t_0) - 2f'(t_1))t_0^2 + (-f'(t_0)t_1 + f'(t_1)t_1 + 3f(t_0) - 3f(t_1))t_0}{(t_1 - t_0)^3}$$

$$a_3 = \frac{(-f'(t_1) - f'(t_0))t_0 + f'(t_0)t_1 + f'(t_1)t_1 + 2f(t_0) - 2f(t_1)}{(t_1 - t_0)^3}$$

把a_0，a_1，a_2和a_3代入则有：

$$\mathrm{P}(t)=f(t_0)\cdot\frac{(t-t_1)^2(-2t+3t_0-t_1)}{(t_0-t_1)^3}$$

$$+f(t_1)\cdot\frac{(t-t_0)^2(-2t+3t_1-t_0)}{(t_1-t_0)^3}+f'(t_0)\cdot\frac{(t-t_1)^2(t-t_0)}{(t_1-t_0)^2}$$

$$+f'(t_1)\cdot\frac{(t-t_0)^2(t-t_1)}{(t_1-t_0)^2}\qquad t\in[t_0,t_1]$$

经整理，所求多项式$P_0(t)$可以写出如下：

$$P_0(t) = f(t_0)\cdot g_{00}(t) + f(t_1)\cdot g_{01}(t)$$
$$+ f'(t_0)\cdot h_{00}(t)\cdot(t_1 - t_0) + f'(t_1)\cdot h_{01}(t)\cdot(t_1 - t_0)$$

式中选取两个**端点**及其及其**切向量**作为曲线构造条件

混合函数如下：

$$
\begin{cases}
g_{00}(t)=\left(1-2\cdot\dfrac{t-t_0}{t_0-t_1}\right)\cdot\left(\dfrac{t-t_1}{t_0-t_1}\right)^2 \\
g_{01}(t)=\left(1-2\cdot\dfrac{t-t_1}{t_1-t_0}\right)\cdot\left(\dfrac{t-t_0}{t_1-t_0}\right)^2 \\
h_{00}(t)=\left(\dfrac{t-t_0}{t_1-t_0}\right)\cdot\left(\dfrac{t-t_1}{t_0-t_1}\right)^2 \\
h_{01}(t)=\left(\dfrac{t-t_1}{t_1-t_0}\right)\cdot\left(\dfrac{t-t_0}{t_1-t_0}\right)^2
\end{cases}
$$

经验证可知：

$$\begin{cases} g_{00}(t_0)=1 \\ g_{01}(t_0)=0 \\ h_{00}(t_0)=0 \\ h_{01}(t_0)=0 \end{cases} \qquad \begin{cases} g_{00}(t_1)=0 \\ g_{01}(t_1)=1 \\ h_{00}(t_1)=0 \\ h_{01}(t_1)=0 \end{cases}$$

$$\begin{cases} g'_{00}(t_0)=0 \\ g'_{01}(t_0)=0 \\ h'_{00}(t_0)=\dfrac{1}{t_1-t_0} \\ h'_{01}(t_0)=0 \end{cases} \qquad \begin{cases} g'_{00}(t_1)=0 \\ g'_{01}(t_1)=0 \\ h'_{00}(t_1)=0 \\ h'_{01}(t_1)=\dfrac{1}{t_1-t_0} \end{cases}$$

3. $t \in [t_0, t_1] \rightarrow u \in [0,1]$

为了使$P_0(t)$的定义区间$t_0 \leqslant t \leqslant t_1$变为区间$0 \leqslant u \leqslant 1$，可以做如下变换

$$u = \frac{t - t_0}{t_1 - t_0}$$

解出 $t=t_0+(t_1-t_0)u$，代入混合函数式中，得：

$$g_{00}(t) = g_{00}(t_0 + (t_1 - t_0)u)$$

$$=(1+2u)(u-1)^2$$

$$=2u^3-3u^2+1=q_{00}(u)$$

$$g_{01}(t)=g_{01}(t_0+(t_1-t_0)u)$$

$$=(3-2u)u^2$$

$$= -2u^3 + 3u^2 = q_{01}(u)$$

$$h_{00}(t) = h_{00}(t_0 + (t_1 - t_0)u)$$

$$= u(1-u)^2$$

$$= u^3 - 2u^2 + u = q_{10}(u)$$

$$h_{01}(t) = h_{01}(t_0 + (t_1 - t_0)u)$$

$$= (-1+u)u^2$$

$$= u^3 - u^2 = q_{11}(u)$$

将关于u的混合函数代入，所求的三次多项式成为：

$$\begin{aligned}\tilde{f}(u) &= \tilde{\mathrm{P}}_0(u) = \mathrm{P}_0(t_0 + (t_1 - t_0)u) \\ &= f(t_0)\cdot q_{00}(u) + f(t_1)\cdot q_{01}(u) \\ &\quad + f'(t_0)\cdot(t_1 - t_0)\cdot q_{10}(u) \\ &\quad + f'(t_1)\cdot(t_1 - t_0)\cdot q_{11}(u)\end{aligned}$$

令

$$\widetilde{f}(0) = f(t_0)$$

$$\widetilde{f}(1) = f(t_1)$$

$$\widetilde{f}'(0) = f'(t_0)(t_1 - t_0)$$

$$\widetilde{f}'(1) = f'(t_1)(t_1 - t_0)$$

得

$$\tilde{\mathrm{P}}_0(u) = \tilde{f}(0)\cdot q_{00}(u) + \tilde{f}(1)\cdot q_{01}(u)$$
$$+ \tilde{f}'(0)\cdot q_{10}(u) + \tilde{f}'(1)\cdot q_{11}(u)$$

$$= (q_{00}(u) \quad q_{01}(u) \quad q_{10}(u) \quad q_{11}(u)) \begin{bmatrix} \tilde{f}(0) \\ \tilde{f}(1) \\ \tilde{f}'(0) \\ \tilde{f}'(1) \end{bmatrix}$$

$$=(2u^3-3u^2+1 \quad -2u^3+3u^2 \quad u^3-2u^2+u \quad u^3-u^2)\begin{bmatrix}\tilde{f}(0)\\ \tilde{f}(1)\\ \tilde{f}'(0)\\ \tilde{f}'(1)\end{bmatrix}$$

$$=(u^3 \quad u^2 \quad u \quad 1)\begin{bmatrix}2 & -2 & 1 & 1\\ -3 & 3 & -2 & -1\\ 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\tilde{f}(0)\\ \tilde{f}(1)\\ \tilde{f}'(0)\\ \tilde{f}'(1)\end{bmatrix}$$

4. 曲线拼接

对一般的Hermite插值问题，一般来说得到的插值多项式次数较高，应用起来不方便。通常的处理办法是将前面给出的参数的三次多项式逐段光滑地连接，如此来确定一般情况下的插值多项式。

将前面t_0和t_1视为t_i和t_{i+1}，设给定$f(t_i)$，$f(t_{i+1})$，$f'(t_i)$，$f'(t_{i+1})$，则在区间$[t_i, t_{i+1}]$的Hermite三次插值多项式$P_i(t)$是：

$$\mathrm{P}_i(t) = f(t_i)\cdot g_{i0}(t) + f(t_{i+1})\cdot g_{i1}(t)$$
$$+ f'(t_i)\cdot h_{i0}(t)\cdot(t_{i+1} - t_i)$$
$$+ f'(t_{i+1})\cdot h_{i1}(t)\cdot(t_{i+1} - t_i)$$

$t_0, t_1, t_2, \cdots, t_{n-1}, t_n$

n段曲线拼接

$$
\begin{cases}
g_{i,0}(t)=\left(1-2\cdot\dfrac{t-t_i}{t_i-t_{i+1}}\right)\cdot\left(\dfrac{t-t_{i+1}}{t_i-t_{i+1}}\right)^2 \\
g_{i,1}(t)=\left(1-2\cdot\dfrac{t-t_{i+1}}{t_{i+1}-t_i}\right)\cdot\left(\dfrac{t-t_i}{t_{i+1}-t_i}\right)^2 \\
h_{i,0}(t)=\left(\dfrac{t-t_i}{t_{i+1}-t_i}\right)\cdot\left(\dfrac{t-t_{i+1}}{t_i-t_{i+1}}\right)^2 \\
h_{i,1}(t)=\left(\dfrac{t-t_{i+1}}{t_{i+1}-t_i}\right)\cdot\left(\dfrac{t-t_i}{t_{i+1}-t_i}\right)^2
\end{cases}
\qquad i=0,1,\cdots,n-1
$$

为了完整地写出这个插值多项式，可以在区间$[t_i, t_{i+1}]$中引入如下一些基本函数：

$$a_{0,0}(t)=\begin{cases} g_{0,0}(t), & t_0 \le t \le t_1 \\ 0, & t_1 \le t \le t_n \end{cases}$$

$$a_{i,0}(t)=\begin{cases} g_{i-1,1}(t), & t_{i-1} \le t \le t_i \\ g_{i,0}(t), & t_i \le t \le t_{i+1} \\ 0, & \text{其它} \end{cases} \qquad i=1,2,\cdots n-1$$

$$a_{n,0}(t) = \begin{cases} 0, & t_0 \le t \le t_{n-1} \\ g_{n-1,1}(t), & t_{n-1} \le t \le t_n \end{cases}$$

$$a_{0,1}(t) = \begin{cases} h_{0,0}(t) \cdot (t_1 - t_0), & t_0 \le t \le t_1 \\ 0, & t_1 \le t \le t_n \end{cases}$$

$$a_{i,1}(t)=\begin{cases} h_{i-1,1}(t)\cdot(t_i-t_{i-1}), & t_{i-1}\le t\le t_i \\ h_{i,0}(t)\cdot(t_{i+1}-t_i), & t_i\le t\le t_{i+1} \\ 0, & \text{其它} \end{cases} \qquad i=1,2,\cdots n-1$$

$$a_{n,1}(t)=\begin{cases} 0, & t_0\le t\le t_{n-1} \\ h_{n-1,1}(t)\cdot(t_n-t_{n-1}), & t_{n-1}\le t\le t_n \end{cases}$$

完整的插值多项式可写为：

$$\mathrm{P}(t)=\sum_{i=0}^{n}\left(f(t_i)a_{i,0}(t)+f'(t_i)a_{i,1}(t)\right)$$

上式区间$[t_0，t_n]$中有定义，且为分段定义。在每个区间 $[t_i，t_{i+1}]$上，都恰有四项。满足插值条件

$$\mathrm{P}(t_i)=f(t_i),\mathrm{P}'(t_i)=f'(t_i),i=0,1,\cdots,n$$

$$\mathrm{P}(t)=\sum_{i=0}^{n}(\mathrm{P}_i\cdot a_{i,0}(t)+\mathrm{P}'_i\cdot a_{i,1}(t))$$

每段曲线$\mathrm{P}_i(t)$只在$[t_i,\ t_{i+1}]$中有定义：

$$\begin{aligned}\mathrm{P}_i(t)&=\mathrm{P}_i\cdot a_{i,0}(t)+\mathrm{P}_{i+1}\cdot a_{i+1,0}(t)\\&\quad+\mathrm{P}'_i\cdot a_{i,1}(t)+\mathrm{P}'_{i+1}\cdot a_{i+1,1}(t)\\&=\mathrm{P}_i\cdot g_{i,0}(t)+\mathrm{P}_{i+1}\cdot g_{i,1}(t)\\&\quad+[\mathrm{P}'_i\cdot h_{i,0}(t)+\mathrm{P}'_{i+1}\cdot h_{i,1}(t)](t_{i+1}-t_i)\end{aligned}$$

自变量的线性变换 $u=\dfrac{t-t_i}{t_{i+1}-t_i}$

用逆变换 $t=t_i+u(t_{i+1}-t_i)$ **代入，将所得关于*u*的多项式记为** $\tilde{\mathrm{P}}_i(u)$**，得**

$$\tilde{\mathrm{P}}_i(u)=P_i\cdot(2u^3-3u^2+1)+P_{i+1}\cdot(-2u^3+3u^2)$$

$$+\tilde{P}'_i\cdot(u^3-2u^2+u)+\tilde{P}'_{i+1}\cdot(u^3-u^2)$$

$$= (u^3 \quad u^2 \quad u \quad 1) \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathrm{P}_i \\ \mathrm{P}_{i+1} \\ \widetilde{\mathrm{P}}'_i \\ \widetilde{\mathrm{P}}'_{i+1} \end{bmatrix}$$

其中 $\mathrm{P}_i = \mathrm{P}(t_i) = f(t_i) \quad \mathrm{P}_{i+1} = \mathrm{P}(t_{i+1}) = f(t_{i+1})$

$$\widetilde{\mathrm{P}}'_i = f'(t_i)(t_{i+1} - t_i)$$

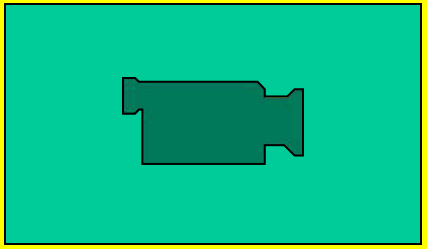
$$\widetilde{\mathrm{P}}'_{i+1} = f'(t_{i+1})(t_{i+1} - t_i)$$

例：设在平面上有两点P_0，P_1，它们的位置向量分别为(1，1)，(4，2)，在P_0的导数值即在该点的切线向量P'_0=(1，1)，在P_1处P'_1=(1，-1)，构造曲线。

$$P(u)=(u^3 \quad u^2 \quad u \quad 1)\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 4 & 2 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$=(u^3 \quad u^2 \quad u \quad 1)\begin{bmatrix} -4 & -2 \\ 6 & 2 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$\begin{cases} x(u) = -4u^3 + 6u^2 + u + 1 \\ y(u) = -2u^3 + 2u^2 + u + 1 \end{cases}$$



第三节 Coons曲面

$$\mathrm{P}(u,w)=(x(u,w),y(u,w),z(u,w))$$

$$0\le u\le 1,0\le w\le 1$$

$uw=\mathrm{P}(u,w)$表示曲面

$$uw_u=\frac{\partial \mathrm{P}(u,w)}{\partial u}$$

$$uw_{uw}=\frac{\partial^2 \mathrm{P}(u,w)}{\partial u\partial w}$$表示双参数的偏导数

$$00_u = \left.\frac{\partial \mathrm{P}(u,w)}{\partial u}\right|_{u=0,w=0}$$

$$00_{uw} = \left.\frac{\partial^2 \mathrm{P}(u,w)}{\partial u \partial w}\right|_{u=0,w=0}$$

*uw*表示了曲面片的方程

0*w*，1*w*，*u*0，*u*1四条边界曲线

$u0_u$边界线的切向量

$u0_w$边界线的跨界切向量

uw_{uu}, uw_{uw}, uw_{ww}曲面片*uw*关于*u*和*w*的二阶偏导数向量

$u0_{uu}$ 表示边界线*u*0上的二阶切向量

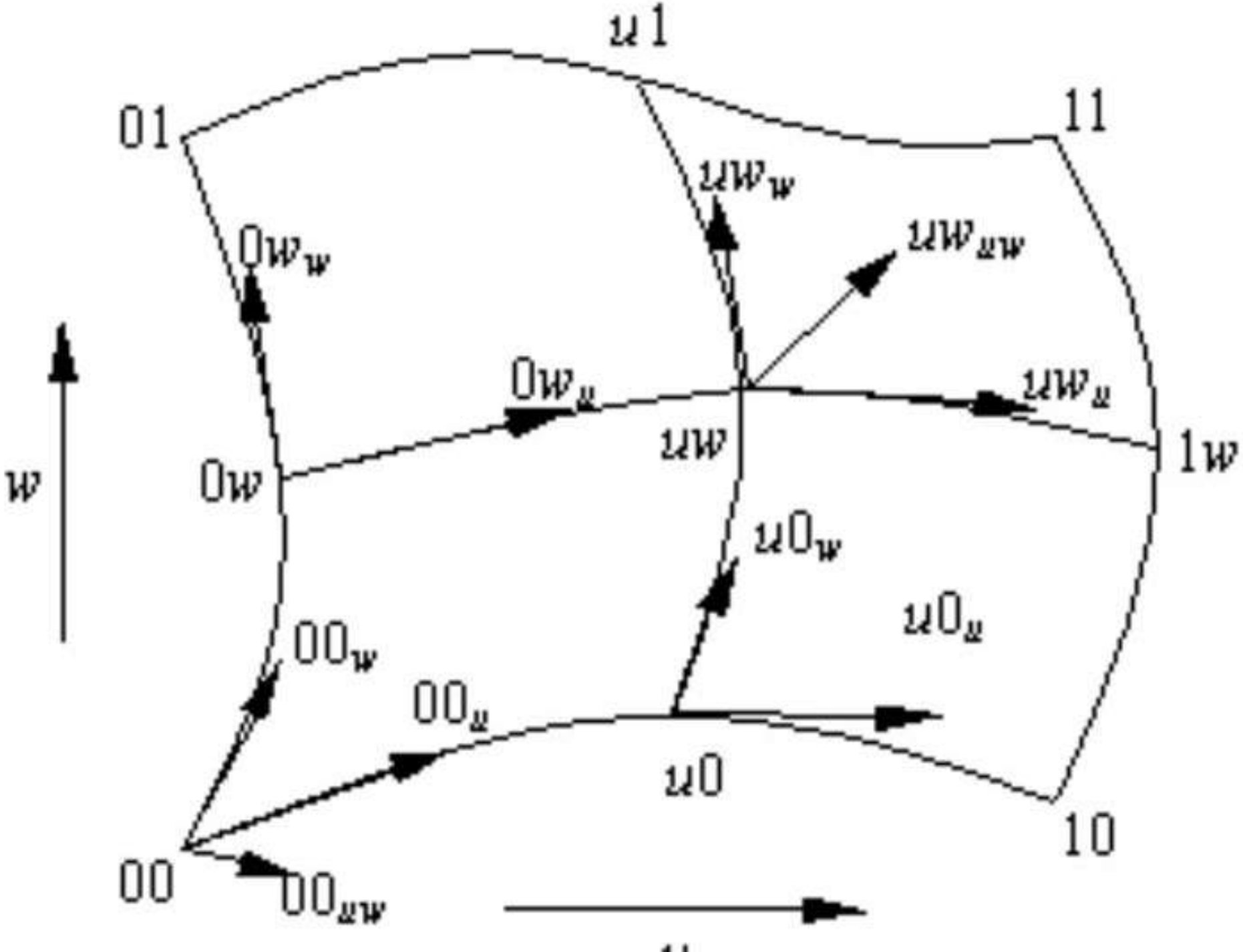
$u0_{ww}$表示边界线*u*0上的二阶跨界切向量

uw_{uw}为曲面片P在点(*u*, *w*)处的扭曲向量。

00，01，10，11表示曲面片四个角点

00*w*，01*w*，10*w*，11*w*，00*u*，01*u*，10*u*，11*u* 四个角点的切向量

00*uw*，01*uw*，10*uw*，11*uw*四个角点的扭曲向量


u1
01
11
uw_w
0w_w
uw_uw
0w_u
uw_u
uw
1w
w
0w
u0_w
u0_u
00_w
00_u
u0
10
00
00_uw

构造具有指定边界曲线的曲面片：

Coons给出的一个解法是：寻找两个混合函数$f_0(t)$和$f_1(t)$，它们是连续的，并且满足$f_0(0)=1$，$f_0(1)=0$，$f_1(0)=0$，$f_1(1)=1$，且$f_0(t)+f_1(t)=1$，$0\leqslant t\leqslant 1$。

$f_0(t)=2t^3-3t^2+1$, $f_1(t)=-2t^3+3t^2$

$f_0(t)=1-t$, $f_1(t)=t$

问题1：求通过四条边界线的曲面

给定四条边界曲线*u*0，*u*1，0*w*，1*w*，且0≤*u*≤1，0≤*w*≤1

①在*u*向进行线性插值，得到直纹面为：

$$P_1(u,w) = f_0(u)\cdot 0w + f_1(u)\cdot 1w$$

$$其中 f_0(u) + f_1(u) = 1, 0 \le u \le 1$$

②在*w*向进行线性插值，得到直纹面为：

$$P_2(u,w) = f_0(w)\cdot u0 + f_1(w)\cdot u1$$

$$其中 f_0(w) + f_1(w) = 1, 0 \le w \le 1$$

③ ① ②叠加可得到一张新曲面$P_3(u,w)$其边界恰好为无用的直线边界 ：

$$\mathrm{P}_3(u,w) = f_0(w)[f_0(u)\cdot 00 + f_1(u)\cdot 10]$$
$$+ f_1(w)[f_0(u)\cdot 01 + f_1(u)\cdot 11]$$
$$= f_0(w)\cdot f_0(u)\cdot 00 + f_0(w)\cdot f_1(u)\cdot 10$$
$$+ f_1(w)\cdot f_0(u)\cdot 01 + f_1(w)\cdot f_1(u)\cdot 11$$
其中$f_0(u) + f_1(u) = 1, 0 \le u \le 1,$
$$f_0(w) + f_1(w) = 1, 0 \le w \le 1$$

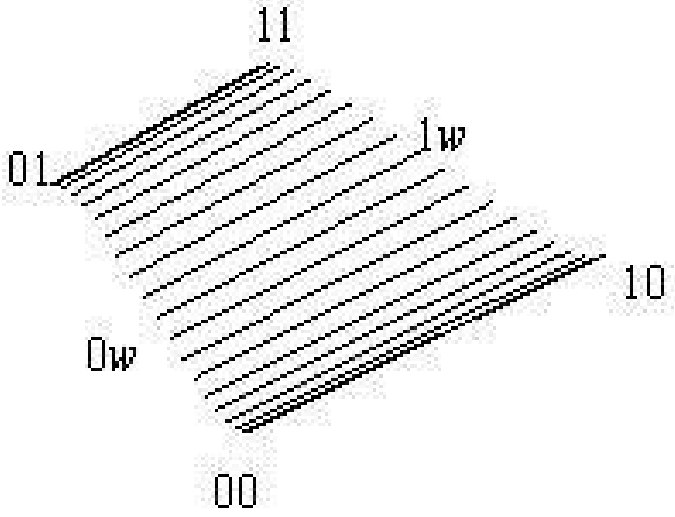
④ 构造Coons曲面P(*u*, *w*)

P(*u*, *w*)=P$_1$(*u*, *w*)+ P$_2$(*u*, *w*)− P$_3$(*u*, *w*)

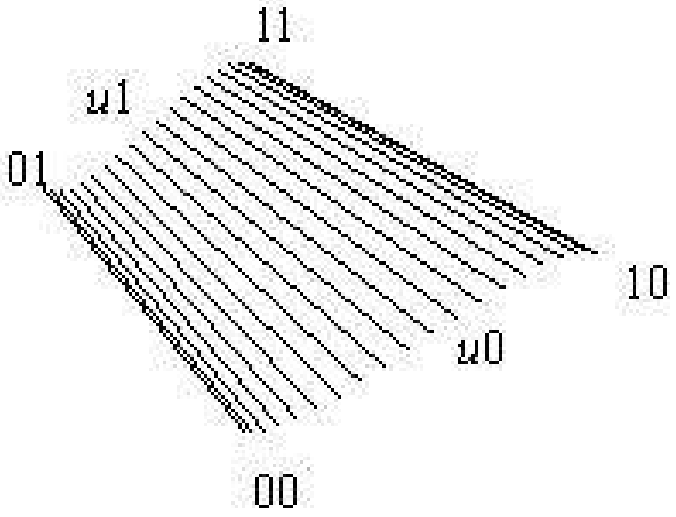
可写成如下形式：

$$
\begin{aligned}
uw &= \mathrm{P}(u,w) \\
&= \mathrm{P}_1(u,w)+\mathrm{P}_2(u,w)-\mathrm{P}_3(u,w) \\
&= [u0,u1]\begin{bmatrix} f_0(w) \\ f_1(w) \end{bmatrix}+[0w,1w]\begin{bmatrix} f_0(u) \\ f_1(u) \end{bmatrix} \\
&\quad -[f_0(u),f_1(u)]M\begin{bmatrix} f_0(w) \\ f_1(w) \end{bmatrix}
\end{aligned}
$$

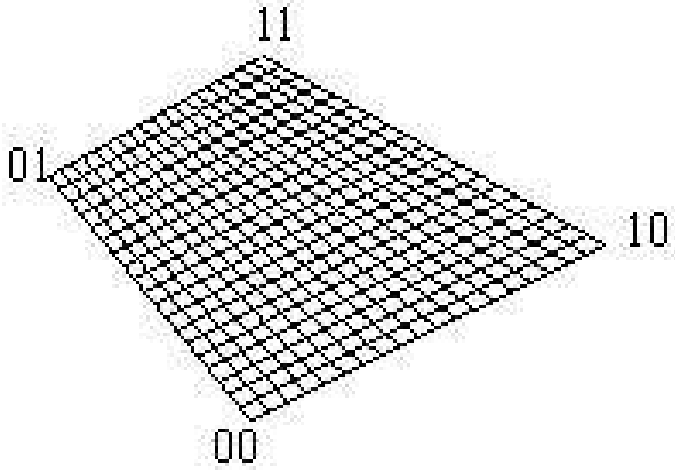
$$
M=\begin{bmatrix} 00 & 01 \\ 10 & 11 \end{bmatrix}
$$



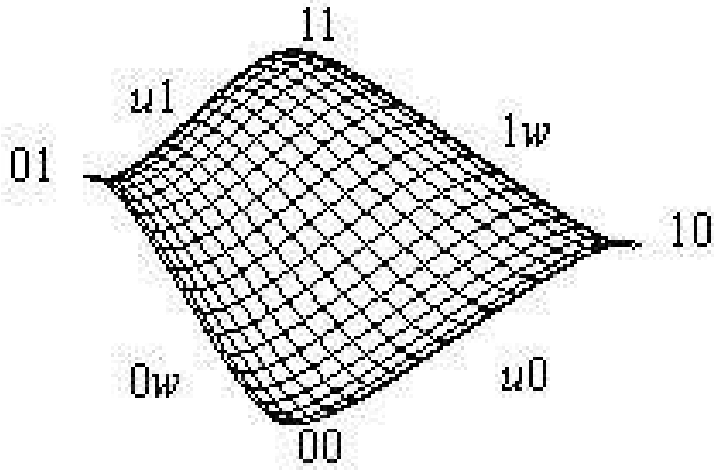
$\mathbf{P}_1(u,w)$



$\mathbf{P}_2(u,w)$



$\mathbf{P}_3(u,w)$



$\mathbf{P}(u,w)$

例如我们来验证0w是它的一条边界线，这只要把u=0代入公式右端，得

$$\text{右端} = 00 \cdot f_0(w) + 01 \cdot f_1(w) + 0w$$

$$-00 \cdot f_0(w) - 01 \cdot f_1(w) = 0w$$

问题2：

①边界线为指定曲线

②且有指定的跨界切向量。

应用上节定义的四个混合函数 $q_{00}(u)$，$q_{01}(u)$，$q_{10}(u)$，$q_{11}(u)$。

$$q_{00}(u) = 2u^3 - 3u^2 + 1$$

$$q_{01}(u) = -2u^3 + 3u^2$$

$$q_{10}(u) = u^3 - 2u^2 + u$$

$$q_{11}(u) = u^3 - u^2$$

这四个函数均是三次多项式。故连续可微，并且还满足下面的条件：

$$\begin{cases} q_{00}(0)=1 \\ q_{01}(0)=0 \\ q_{10}(0)=0 \\ q_{11}(0)=0 \end{cases} \qquad \begin{cases} q_{00}(1)=0 \\ q_{01}(1)=1 \\ q_{10}(1)=0 \\ q_{11}(1)=0 \end{cases}$$

$$\begin{cases} q'_{00}(0)=0 \\ q'_{01}(0)=0 \\ q'_{10}(0)=1 \\ q'_{11}(0)=0 \end{cases} \qquad \begin{cases} q'_{00}(1)=0 \\ q'_{01}(1)=0 \\ q'_{10}(1)=0 \\ q'_{11}(1)=1 \end{cases}$$

设已知四条边界曲线$u0$，$u1$，$0w$，$1w$

及沿这四条边界曲线的跨界切向量

$u0_w$，$u1_w$，$0w_u$，$1w_u$。

求出四个角点的位置向量00，01，10，11，

切向量00_w，01_w，10_w，11_w，00_u，01_u，10_u，11_u，

扭曲向量00_{uw}，01_{uw}，10_{uw}，11_{uw}，

写出符合要求曲面片的数学表达式如下：

$$uw=[u0,u1,u0_{w},u1_{w}]\begin{bmatrix}q_{00}(w)\\q_{01}(w)\\q_{10}(w)\\q_{11}(w)\end{bmatrix}+[0w,1w,0w_{u},1w_{u}]\begin{bmatrix}q_{00}(u)\\q_{01}(u)\\q_{10}(u)\\q_{11}(u)\end{bmatrix}$$

$$-[q_{00}(u),q_{01}(u),q_{10}(u),q_{11}(u)]\cdot M\cdot\begin{bmatrix}q_{00}(w)\\q_{01}(w)\\q_{10}(w)\\q_{11}(w)\end{bmatrix}$$

$$M=\begin{bmatrix} 00 & 01 & 00_{w} & 01_{w} \\ 10 & 11 & 10_{w} & 11_{w} \\ 00_{u} & 01_{u} & 00_{uw} & 01_{uw} \\ 10_{u} & 11_{u} & 10_{uw} & 11_{uw} \end{bmatrix}$$

验证通过边界0w

$$
\begin{aligned}
\text{右端} &= 00\cdot q_{00}(w)+01\cdot q_{01}(w) \\
&\quad +00_w\cdot q_{10}(w)+01_w\cdot q_{11}(w)+0w \\
&\quad -00\cdot q_{00}(w)-01\cdot q_{01}(w) \\
&\quad -00_w\cdot q_{10}(w)-01_w\cdot q_{11}(w)+0w \\
&= 0w
\end{aligned}
$$

验证满足跨界切向量$0w_u$:

$$
\begin{aligned}
\text{右端} &= [00_u \quad 01_u \quad 00_{uw} \quad 01_{uw}]\begin{bmatrix} q_{00}(w) \\ q_{01}(w) \\ q_{10}(w) \\ q_{11}(w) \end{bmatrix} \\
&+[0w \quad 1w \quad 0w_u \quad 1w_u]\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \\
&-[0 \quad 0 \quad 1 \quad 0]\cdot M \cdot \begin{bmatrix} q_{00}(w) \\ q_{01}(w) \\ q_{10}(w) \\ q_{11}(w) \end{bmatrix} \\
&= 0w_u
\end{aligned}
$$

问题3： 指定四个角点以及在这些点上的切向量和扭曲向量后，求解曲面的表达式。

计算求得四个角点的位置向量00，01，10，11，切向量00_w，01_w，10_w，11_w，00_u，01_u，10_u，11_u，以及扭曲向量00_{uw}，01_{uw}，10_{uw}，11_{uw}，

已知角点位置向量00，10以及在这两点关于u的切向量00_u和01_u，可以用Hermite插值公式来指定一条u边界线：

$$\mathrm{u0} = 00 \cdot q_{00}(u) + 10 \cdot q_{01}(u) + 00_u \cdot q_{10}(u) + 10_u \cdot q_{11}(u)$$

$$u1 = 01 \cdot q_{00}(u) + 11 \cdot q_{01}(u) + 01_u \cdot q_{10}(u) + 11_u \cdot q_{11}(u)$$

$$u0_w = 00_w \cdot q_{00}(u) + 10_w \cdot q_{01}(u) + 00_{uw} \cdot q_{10}(u) + 10_{uw} \cdot q_{11}(u)$$

$$u1_w = 01_w \cdot q_{00}(u) + 11_w \cdot q_{01}(u) + 01_{uw} \cdot q_{10}(u) + 11_{uw} \cdot q_{11}(u)$$

$$[u0 \quad u1 \quad u0_w \quad u1_w]$$

$$= [q_{00}(u) \quad q_{01}(u) \quad q_{10}(u) \quad q_{11}(u)] \cdot M$$

$$M = \begin{bmatrix} 00 & 01 & 00_w & 01_w \\ 10 & 11 & 10_w & 11_w \\ 00_u & 01_u & 00_{uw} & 01_{uw} \\ 10_u & 11_u & 10_{uw} & 11_{uw} \end{bmatrix}$$

$$0w = 00 \cdot q_{00}(w) + 01 \cdot q_{01}(w)$$
$$+ 00_w \cdot q_{10}(w) + 01_w \cdot q_{11}(w)$$
$$1w = 10 \cdot q_{00}(w) + 11 \cdot q_{01}(w)$$
$$+ 10_w \cdot q_{10}(w) + 11_w \cdot q_{11}(w)$$
$$0w_u = 00_u \cdot q_{00}(w) + 01_u \cdot q_{01}(w)$$
$$+ 00_{uw} \cdot q_{10}(w) + 01_{uw} \cdot q_{11}(w)$$
$$1w_u = 10_u \cdot q_{00}(w) + 11_u \cdot q_{01}(w)$$
$$+ 10_{uw} \cdot q_{10}(w) + 11_{uw} \cdot q_{11}(w)$$

$$\begin{bmatrix} {}^0w \\ {}^1w \\ {}^0w_u \\ {}^1w_u \end{bmatrix} = M \cdot \begin{bmatrix} q_{00}(w) \\ q_{01}(w) \\ q_{10}(w) \\ q_{11}(w) \end{bmatrix}$$

将上两式代入前式，就可以得到：

$$uw = \begin{bmatrix} q_{00}(u) & q_{01}(u) & q_{10}(u) & q_{11}(u) \end{bmatrix} \cdot M \cdot \begin{bmatrix} q_{00}(w) \\ q_{01}(w) \\ q_{10}(w) \\ q_{11}(w) \end{bmatrix}$$

$$= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\cdot \begin{bmatrix} 00 & 01 & 00_w & 01_w \\ 10 & 11 & 10_w & 11_w \\ 00_u & 01_u & 00_{uw} & 01_{uw} \\ 10_u & 11_u & 10_{uw} & 11_{uw} \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot H \cdot M \cdot H^T \cdot \begin{bmatrix} w^3 & w^2 & w & 1 \end{bmatrix}^T$$

例： 给出四个角点以及在该角点上的切向量和扭曲向量来构造Coons曲面表达式。设在平面上有四点P_0，P_1，P_2，P_3，它们的位置向量分别为(0，0，0)，(0，0.75，0)，(0.75，0，0)，(0.75，0.75，0)。该四点的切向量、跨界切向量和扭曲向量，定义在关于角点的信息矩阵*M*中：

$$M_x=\frac{3}{4}\cdot\begin{bmatrix}0 & 0 & -1 & 1\\ 1 & 1 & -1 & -1\\ 1 & 1 & -1 & 1\\ 1 & 1 & 1 & -1\end{bmatrix}$$

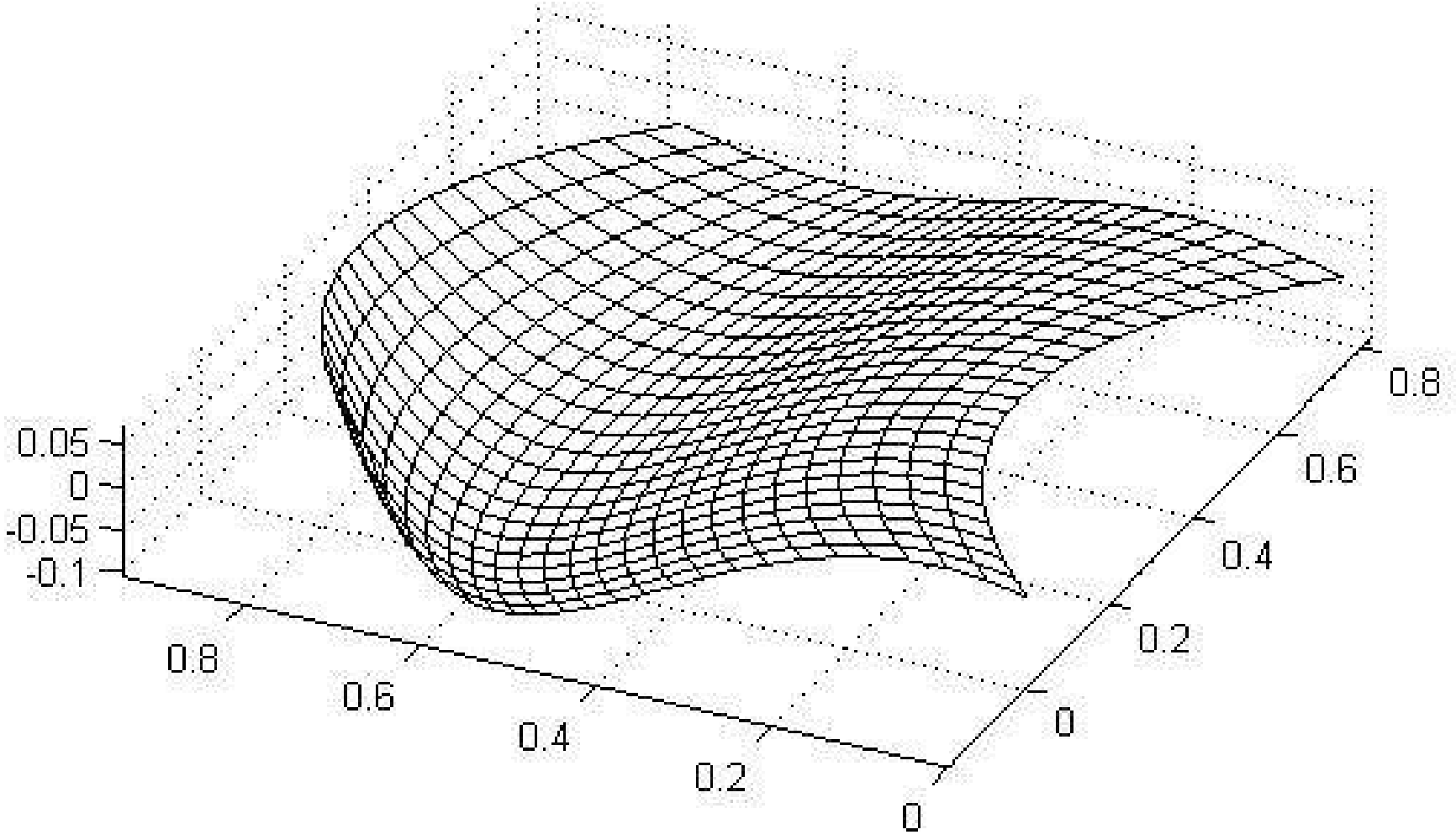
$$M_y = \frac{3}{4} \cdot \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ -1 & -1 & 1 & -1 \end{bmatrix}$$

$$M_z = \frac{3}{4} \cdot \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

$$x(u,w)=\begin{bmatrix}u^3 & u^2 & u & 1\end{bmatrix}\begin{bmatrix}3 & -3 & 0 & 0\\ -4.5 & 3.75 & 0.75 & 0\\ 0 & 0.75 & -0.75 & 0.75\\ 0 & 0.75 & -0.75 & 0\end{bmatrix}\begin{bmatrix}w^3\\ w^2\\ w\\ 1\end{bmatrix}$$

$$y(u,w)=\begin{bmatrix}u^3 & u^2 & u & 1\end{bmatrix}\begin{bmatrix}0 & 0 & 0 & 0\\ 0 & -0.75 & 0.75 & -0.75\\ 0 & 0.75 & -0.75 & 0.75\\ 0 & 0 & 0.75 & 0\end{bmatrix}\begin{bmatrix}w^3\\ w^2\\ w\\ 1\end{bmatrix}$$

$$z(u,w)=\begin{bmatrix}u^3 & u^2 & u & 1\end{bmatrix}\begin{bmatrix}0 & 0 & 0 & 0\\ 0 & -0.75 & 0.75 & 0\\ 0 & 0.75 & -0.75 & 0\\ 1.5 & -2.25 & 0.75 & 0\end{bmatrix}\begin{bmatrix}w^3\\ w^2\\ w\\ 1\end{bmatrix} \quad u\in[0,1], w\in[0,1]$$


0.05
0
-0.05
-0.1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8

第四节 Bezier曲线和曲面

1.Bezier曲线定义

给出型值点P_0，P_1，…，P_n，它们所确定的n次Bezier曲线是：

$$P(t)=\sum_{i=0}^{n} B_{i,n}(t)\cdot P_i \qquad 0\le t\le 1$$

$B_{i,n}(t)$ 是Bernstein多项式，调和函数

$$B_{i,n}(t)=C_n^i t^i(1-t)^{n-i}=\frac{n!}{i!(n-i)!}t^i\cdot(1-t)^{n-i}$$

$$i=0,1,\cdots,n$$

涉及到的0!及0^0，按约定均为1。

当n=1时

$$\mathrm{P}(t)=(1-t)\mathrm{P}_0+t\mathrm{P}_1 \qquad 0\le t\le 1$$

一次 *bezier* 曲线是直线段

在n=2时

$$\mathrm{P}(t)=(1-t)^2\mathrm{P}_0+2t(1-t)\mathrm{P}_1+t^2\mathrm{P}_2$$

$$=(t^2 \quad t \quad 1)\begin{bmatrix}1 & -2 & 1\\ -2 & 2 & 0\\ 1 & 0 & 0\end{bmatrix}\begin{bmatrix}\mathrm{P}_0\\ \mathrm{P}_1\\ \mathrm{P}_2\end{bmatrix},\quad 0\le t\le 1$$

二次*bezier*曲线是抛物线

- **在n=3时**

$$P(t)=(1-t)^3P_0+3t(1-t)^2P_1+3t^2(1-t)P_2+t^3P_3$$

$$=(t^3 \quad t^2 \quad t \quad 1)\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 3 & 0 & 0\\ 1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}P_0\\ P_1\\ P_2\\ P_3\end{bmatrix}$$

$0\le t\le 1$

三次*bezier*曲线是三次参数多项式曲线

混和函数$B_{i,n}(t)=C_n^i t^i(1-t)^{n-i}$的性质：　$(0\le t\le 1)$

①　$B_{i,n}(t)=\begin{cases}0 & t=0,1\text{且}i\neq 0,n\text{时}\\ \neq 0 & t\neq 0,1\text{时}\end{cases}$

②　$0\le B_{i,n}(t)\le 1$

③　$B_{i,n}(t)=B_{n-i,n}(1-t)$

④　$\sum_{i=0}^{n} B_{i,n}(t)=1$

2. Bezier曲线的一些重要性质：

① 端点性质：$P(0)=P_0$，$P(1)=P_n$，曲线通过起点和终点。

② $P'(0)=n(P_1-P_0),\quad P'(1)=n(P_n-P_{n-1})$

$$P'(t)=\sum_{i=0}^{n}\frac{n!}{i!(n-i)!}(i\cdot t^{i-1}\cdot(1-t)^{n-i}-(n-i)\cdot t^{i}\cdot(1-t)^{n-i-1})P_i$$

$$=-\frac{n!}{(n-1)!}(1-t)^{n-1}P_0+\frac{n!}{(n-1)!}(1-t)^{n-1}P_1$$

$$-\frac{n!}{(n-2)!}t(1-t)^{n-2}P_1+\frac{n!}{(n-2)!}t(1-t)^{n-2}P_2-\frac{n!}{2!(n-3)!}t^2(1-t)^{n-3}P_2$$

$$+\cdots+\frac{n!}{(n-2)!}t^{n-2}(1-t)P_{n-1}-\frac{n!}{(n-1)!}t^{n-1}P_{n-1}+\frac{n!}{(n-1)!}t^{n-1}P_n$$

$$=\sum_{i=0}^{n-1}\frac{n!}{i!(n-i-1)!}\cdot t^{i}\cdot(1-t)^{n-i-1}\cdot(P_{i+1}-P_i)$$

③ Bezier曲线的对称性

$$P(t)=\sum_{i=0}^{n} B_{i,n}(t)P_i=\sum_{i=0}^{n} B_{n-i,n}(1-t)P_i$$

$$=B_{0,n}(t)P_0+\cdots+B_{i,n}(t)P_i+\cdots$$

$$+B_{n-i,n}(t)P_{n-i}+\cdots+B_{n,n}(t)P_n$$

$$=B_{n,n}(1-t)P_0+\cdots+B_{n-i,n}(1-t)P_i+\cdots+$$

$$B_{i,n}(1-t)P_{n-i}+\cdots+B_{0,n}(1-t)P_n$$

$$=\sum_{i=0}^{n} B_{i,n}(1-t)P_i^*$$

④曲线的凸包性

对给定的型值点P_0，P_1，…，P_n

$$\text{点集}M=\left\{\sum_{i=0}^{n}\lambda_i P_i \;\middle|\; 0\le\lambda_i\le 1,\ \sum_{i=0}^{n}\lambda_i=1\right\}$$

称作$n+1$个点张成的凸包

$$\sum_{i=0}^{n} B_{i,n}(t)=1, 0\le B_{i,n}(t)\le 1$$

$$0\le i\le 1, 0\le t\le 1$$

3. 三次Bezier曲线光滑拼接的条件

$P_0P_1P_2P_3$和$Q_0Q_1Q_2Q_3$，两个bezier多边形

①曲线在连接点处C^0连续的条件是$P_3=Q_0$

② 曲线在连接点处G^1连续，$Q'_0=aP'_3$

$Q'_0=3(Q_1-Q_0)$，$P'_3=3(P_3-P_2)$，

$Q_1-Q_0=a(P_3-P_2)$

由此得

$$Q_0 = P_3 = \frac{Q_1 + aP_2}{a+1} \Rightarrow Q_0 - Q_1 = \frac{a}{a+1}(P_2 - Q_1)$$

$$\therefore P_2, P_3 = Q_0, Q_1 \text{共线且} Q_1 \text{和} P_2 \text{在} P_3 \text{两侧。}$$

③连接点处C^2连续的条件

对前面的公式求两次导数，可得，

$$P''(t) = (6t \quad 2)\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \end{bmatrix}\begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

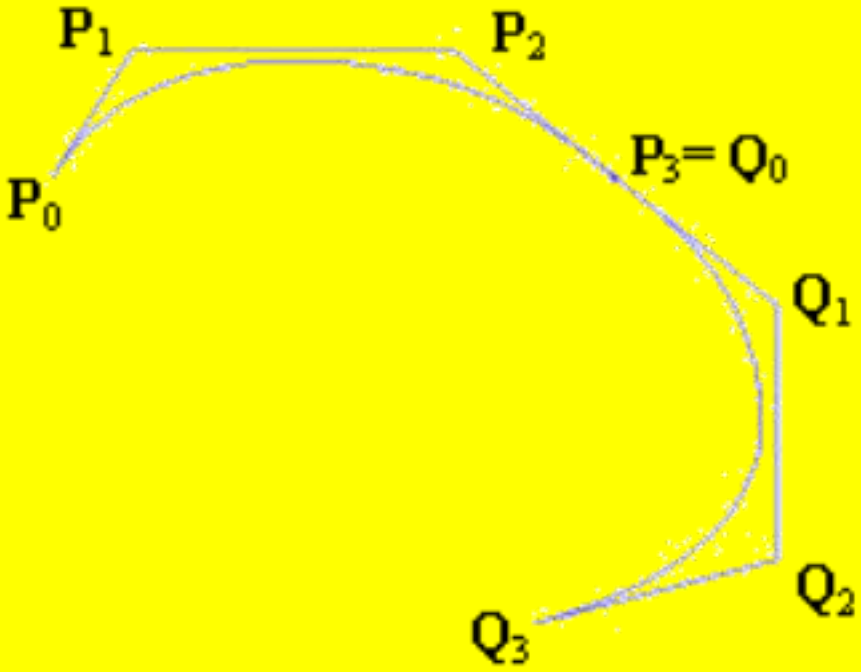
$$= (-6t+6)P_0 + (18t-12)P_1 + (-18t+6)P_2 + 6tP_3$$

于是知，$P''(1)=6P_1-12P_2+6P_3, Q''(0)=6Q_0-12Q_1+6Q_2$

要求，　$P''(1)=Q''(0)$即 $Q_0-2Q_1+Q_2=P_3-2P_2+P_1$

注意到$Q_0=P_3$，由此得：

$$Q_2-P_1=2(Q_1-P_2)\Rightarrow Q_1P_2 // Q_2P_1 \text{且} |Q_1P_2|=\frac{1}{2}|Q_2P_1|$$



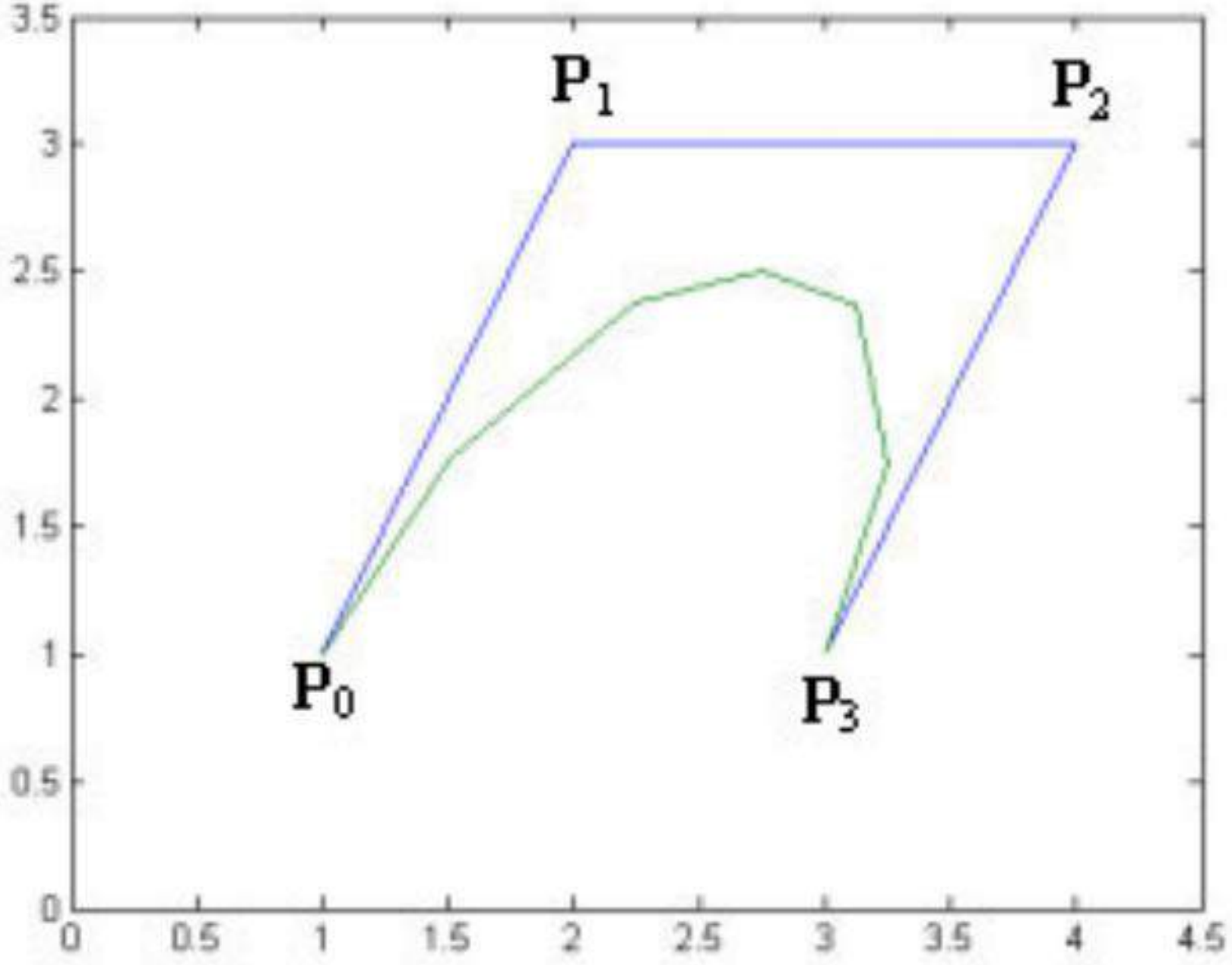
4. Bezier曲线绘制

①利用定义式

Bezier曲线的绘制，可以利用其定义式，对参数t选取足够多的值，计算曲线上的一些点，然后用折线连接来近似画出实际的曲线。随着选取点增多，折线和曲线可以任意接近。

假设给定的四个型值点是P_0=(1，1)，P_1=(2，3)，P_2=(4，3)，P_3=(3，1)，则计算结果见表

t	$(1-t)^3$	$3t(1-t)^2$	$3t^2(1-t)$	t^3	P(t)
0	1	0	0	0	(1,1)
0.15	0.614	0.325	0.0574	0.0034	(1.5058,1.765)
0.35	0.275	0.444	0.239	0.043	(2.248,2.376)
0.5	0.125	0.375	0.375	0.125	(2.75,2.5)
0.65	0.043	0.239	0.444	0.275	(3.122,2.36)
0.85	0.0034	0.0574	0.325	0.614	(3.248,1.75)
1	0	0	0	1	(3,1)


P1
P2
P0
P3
3.5
3
2.5
2
1.5
1
0.5
0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5

②利用曲线性质（几何作图法和分裂法）

a.几何作图法

记点P_k，P_{k+1}，…，P_l可以生成的Bezier曲线为$P_{k,l}(t)$，$0 \leqslant t \leqslant 1$，则成立下面的递推关系

$$P_{0,n}(t)=(1-t)\cdot P_{0,n-1}(t)+t\cdot P_{1,n}(t)$$

$$C_n^i = C_{n-1}^{i} + C_{n-1}^{i-1}$$（组合等式）

左端$=\sum_{i=0}^{n} B_{i,n}(t)\cdot P_i$

$= C_n^0(1-t)^n P_0 + C_n^1 t^1(1-t)^{n-1} P_1 + \cdots + C_n^n t^n P_n$

右端 $=(1-t)\cdot\sum_{i=0}^{n-1} C_{n-1}^i\cdot t^i(1-t)^{n-i-1}\cdot P_i + t\cdot\sum_{i=0}^{n-1} C_{n-1}^i\cdot t^i(1-t)^{n-i-1}\cdot P_{i+1}$

$=(1-t)\left[(1-t)^{n-1}\cdot P_0+\sum_{i=1}^{n-1} C_{n-1}^i t^i(1-t)^{n-i-1}\cdot P_i\right]$

$+t\left[\sum_{i=1}^{n-1} C_{n-1}^{i-1} t^{i-1}(1-t)^{n-i}\cdot P_i + t^{n-1}\cdot P_n\right]$

$=(1-t)^n\cdot P_0+\sum_{i=1}^{n-1} C_{n-1}^i t^i(1-t)^{n-i}\cdot P_i+\sum_{i=1}^{n-1} C_{n-1}^{i-1} t^i(1-t)^{n-i}\cdot P_i + t^n\cdot P_n$

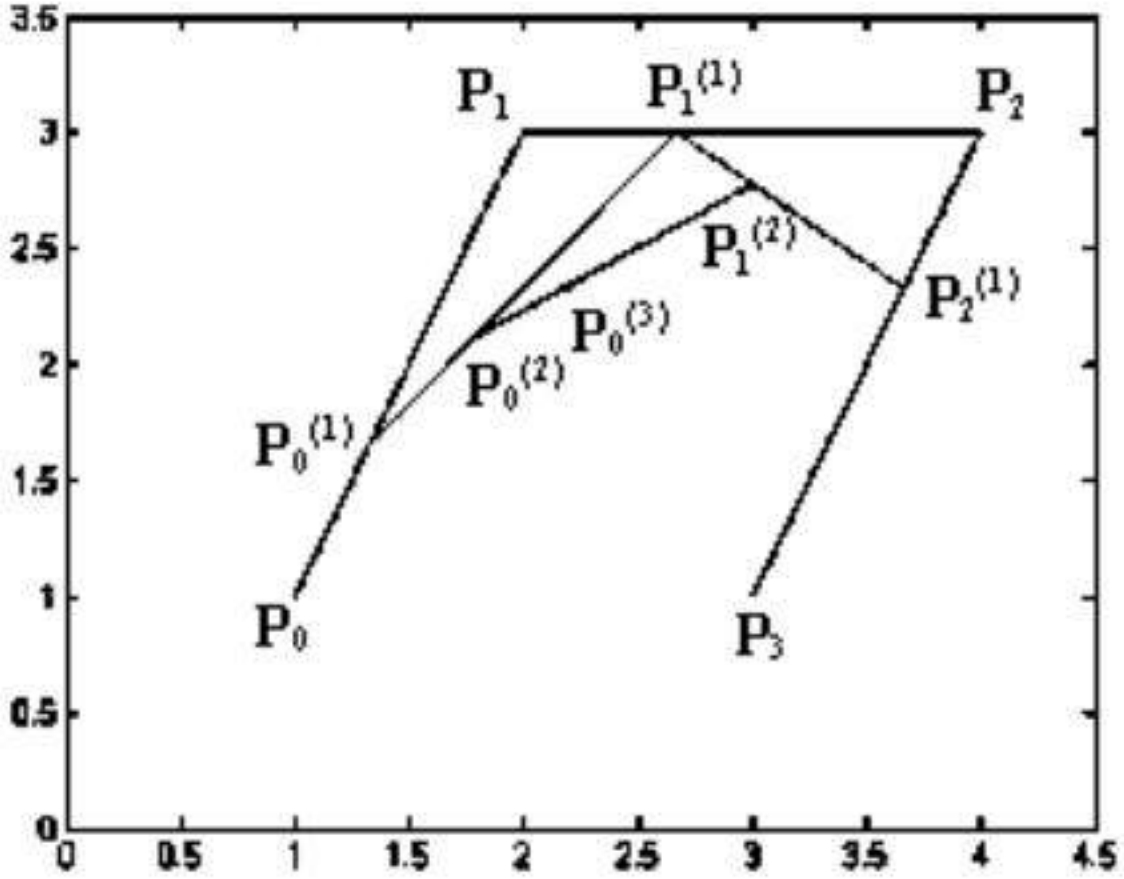
$=(1-t)^n\cdot P_0+\sum_{i=1}^{n-1}(C_{n-1}^i + C_{n-1}^{i-1})t^i(1-t)^{n-i}\cdot P_i + t^n\cdot P_n$

$=\sum_{i=0}^{n} C_n^i t^i(1-t)^{n-i}\cdot P_i$

= 左端

上式改写为：

$$P_{0,n}(t) = P_{0,n-1}(t) + t(P_{1,n}(t) - P_{0,n-1}(t))$$



几何作图法伪代码语言实现：

设控制点 $P_0, P_1, \cdots P_n, t$的增量为$0 \le \Delta t \le 1$

$t = 0, \cdots 1, t = t + \Delta t$

$(1) m \leftarrow n, i = 0 \cdots n, R_i \leftarrow P_i$

(2)若$m > 0$,做（i)至（iii)

$(i) i = 0 \cdots m - 1, Q_i = R_i + t（R_{i+1} - R_i)$

$(ii) m \leftarrow m - 1$

$(iii) i = 0 \cdots m, R_i \leftarrow Q_i$

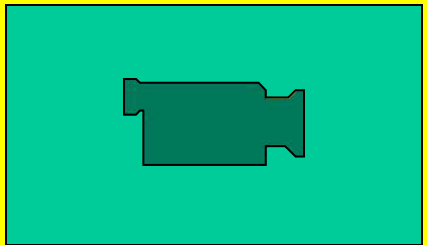
$goto(2)$

$(3) P(t) \leftarrow R_0$

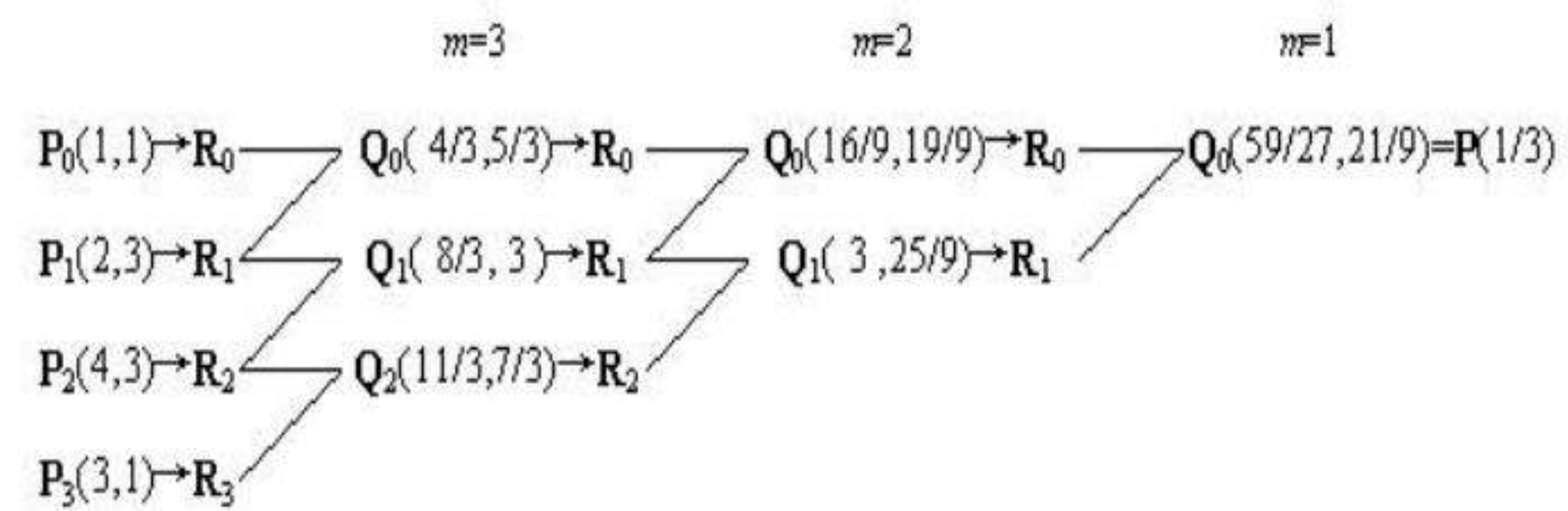
```
void bez_to_points(int n, double P[], int npoints, double points[])
 //    P为控制点坐标
 //控制点P的个数为n +1

//points存储Bezier曲线上的离散点序列
//离散点序列points的个数为npoints+1
{     double t,delt;
       delt=1.0/(double)npoints;//将参数t   npoints等分
       t=0.0;
                 for(int i=0;i<=npoints;i++)
       {         points[i]=decas(n, P, t);
//分别求出npoints+1个离散点points的坐标
                           t+=delt;
       }
}
```

```
double decas(int n,double P[],double t)
{
    int m,i;
    double *R, *Q, P0;
    R = new double[n +1];
     Q = new double[n +1];
    for(i=0;i<=n;i++)
    R[i]= P [i];   //将控制点坐标P保存于R中
```

```
//n次Bezier曲线在点t的值，可由两条n-1次Bezier曲线在点t的值通过线性组合而求得。
for(m=n;m>0;m--)
        {
for(i=0;i<= m -1;i++)
Q[i]= R [i]+t*( R [i+1]- R [i]);
for(i=0;i<= m -1;i++)
R[i]= Q [i];
        }
P0=R[0];
delete R;delete Q;
return (P0);
}
```



设给出四点的坐标是(1，1)，(2，3)，(4，3)，(3，1)，求所确定三次Bezier曲线在t=1/3时的值P(1/3)，算法的计算过程



Bezier几何作图算法计算过程

b. 分裂法

思想：将原控制点集分为两个点数相同的新控制点集，分别对应原曲线的前半段和后半段，新控制点集比原控制点集更接近直线，分裂过程继续进行，控制点集会迅速向曲线靠近，当满足某个允许的界限时，可依次连接各点的折线来表示曲线

设控制点序列P_0，P_1，…，P_n确定的n次Bezier曲线是$P(t)$，用如下递归方式计算另一组点集：

$$P_i^k=\begin{cases}P_i, & k=0,\quad i=0,1,\cdots,n\\ \frac{1}{2}(P_i^{k-1}+P_{i-1}^{k-1}), & k=1,2,\cdots,n,\quad i=k,k+1,\cdots,n\end{cases}$$

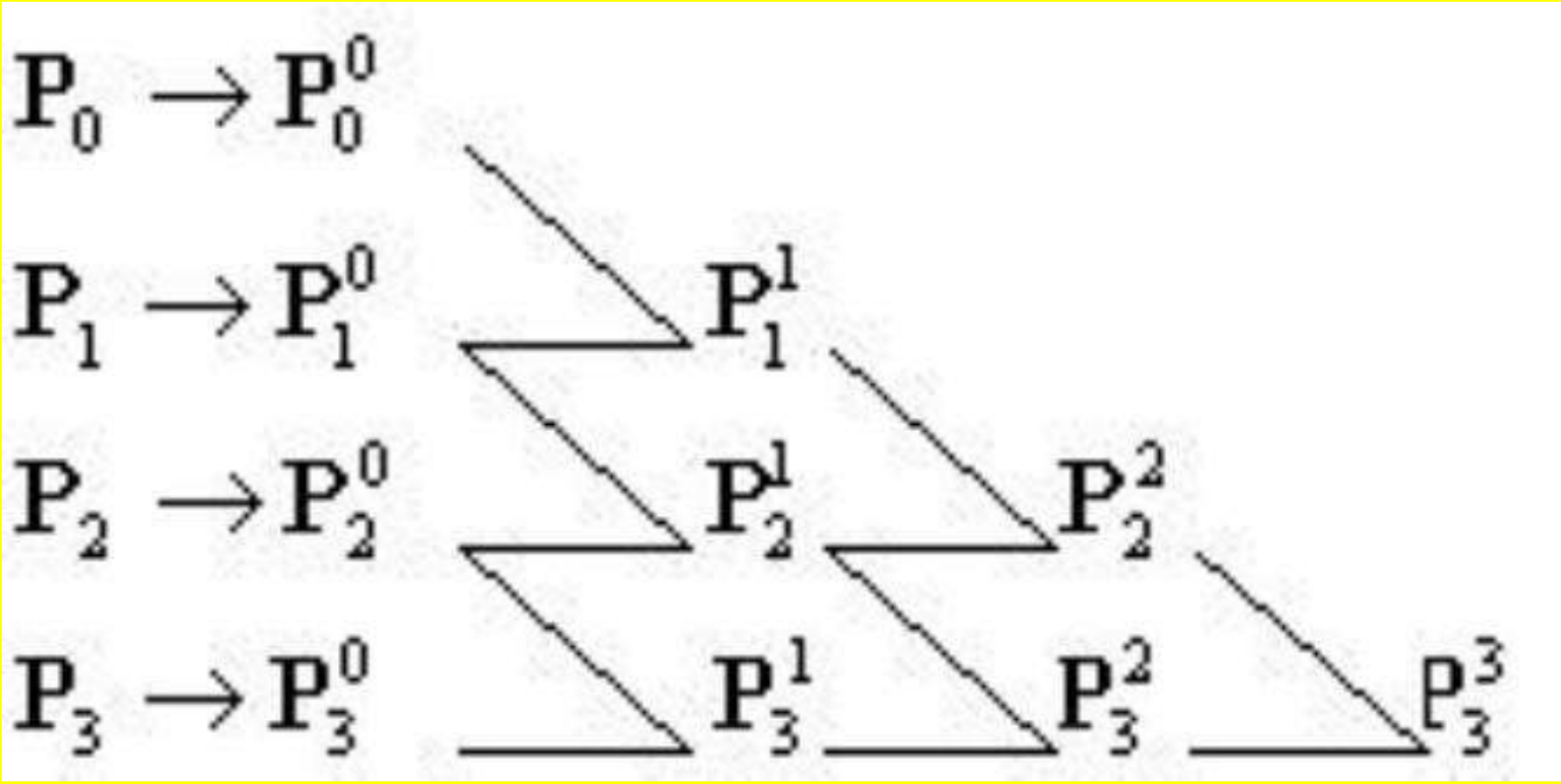
如果令$P_a(s)$和$P_b(s)$分别是以控制点序列和$P_0^0,P_1^1,\cdots,P_n^n$，$P_n^n,P_n^{n-1},\cdots,P_n^0$确定的Bezier曲线，其中$0\leqslant s\leqslant 1$，那么就有：

$$P(t)=\begin{cases} P_a(s)=\sum_{i=0}^{n} C_n^i \cdot s^i (1-s)^{n-i} \cdot P_i^i, \\ \qquad s=2t,\ 0\le t\le \frac{1}{2} \\ P_b(s)=\sum_{i=0}^{n} C_n^i \cdot s^i (1-s)^{n-i} \cdot P_n^{n-i}, \\ \qquad s=2t-1,\ \frac{1}{2}\le t\le 1 \end{cases}$$

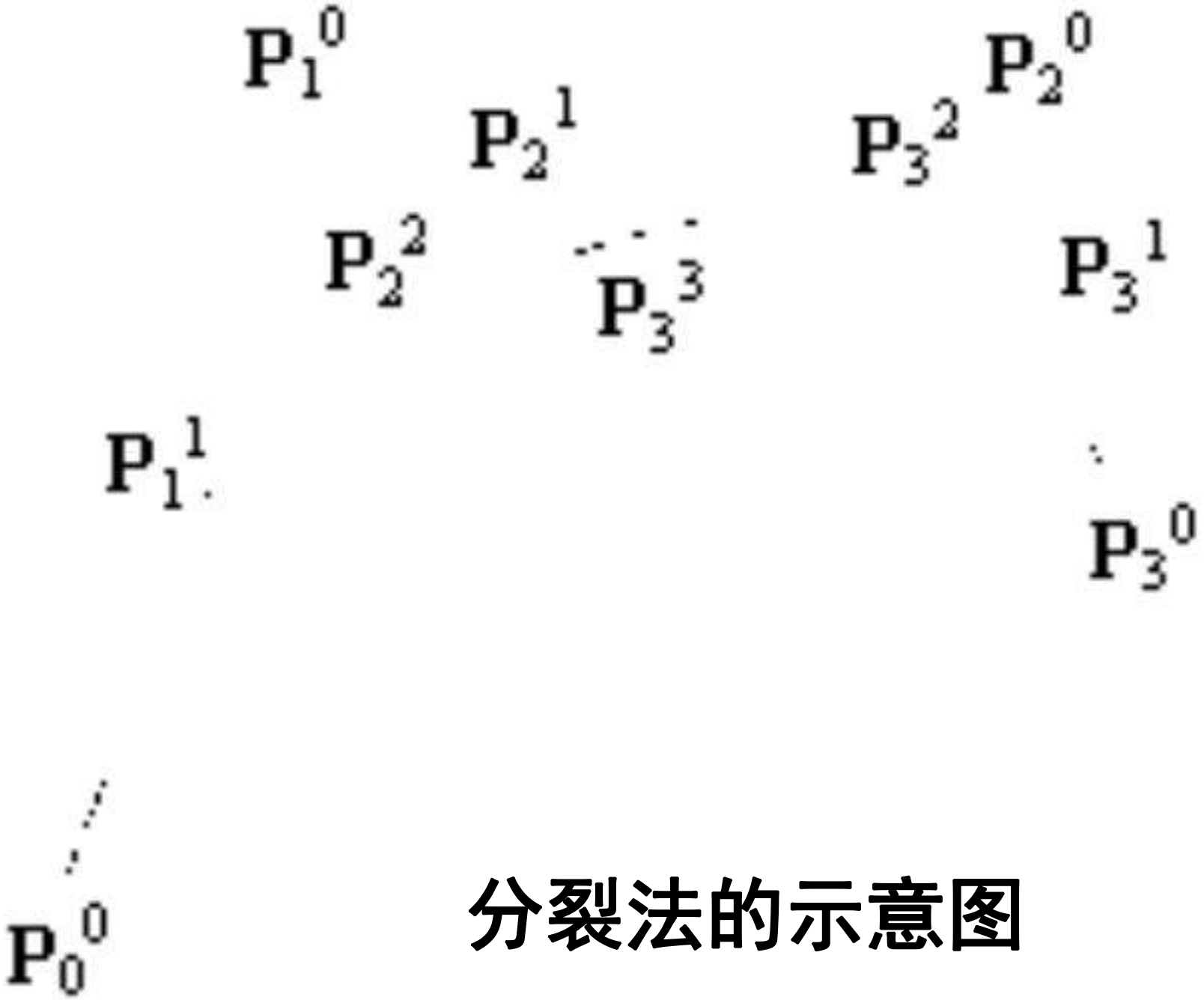
已知四点P_0，P_1，P_2，P_3，确定了一条三次Bezier曲线$P(t)$，可写出下式，

$$P(t)=B_{0,3}(t)P_0+B_{1,3}(t)P_1+B_{2,3}(t)P_2+B_{3,3}(t)P_3$$

$$=\begin{cases} B_{0,3}(2t)P_0^0+B_{1,3}(2t)P_1^1+B_{2,3}(2t)P_2^2+B_{3,3}(2t)P_3^3, \\ \qquad 0\le t\le\frac{1}{2} \\ B_{0,3}(2t-1)P_3^3+B_{1,3}(2t-1)P_3^2+B_{2,3}(2t-1)P_3^1+B_{3,3}(2t-1)P_3^0, \\ \qquad \frac{1}{2}\le t\le 1 \end{cases}$$


$P_0 \rightarrow P_0^0$
$P_1 \rightarrow P_1^0$
P_1^1
$P_2 \rightarrow P_2^0$
P_2^1
P_2^2
$P_3 \rightarrow P_3^0$
P_3^1
P_3^2
P_3^3

分裂法中的递归计算



分裂法的示意图

验证Bezier曲线分成前后两段的正确性，

P_0的系数为例，验证两端它的系数是相等的。

左端＝ $B_{0,3}(t)=(1-t)^3$

$0\le t\le 1/2$

右端＝ $B_{0,3}(2t)+\frac{1}{2}B_{1,3}(2t)+\frac{1}{4}B_{2,3}(2t)+\frac{1}{8}B_{3,3}(2t)$

$=(1-2t)^3+\frac{1}{2}\cdot 3\cdot 2t\cdot(1-2t)^2+\frac{1}{4}\cdot 3\cdot(2t)^2(1-2t)+\frac{1}{8}(2t)^3$

$=(1-2t)^3+3\cdot t\cdot(1-2t)^2+3\cdot t^2\cdot(1-2t)+t^3$

$=(1-2t+t)^3$

$=(1-t)^3$

$1/2\le t\le 1$

右端＝ $\frac{1}{8}B_{0,3}(2t-1)=\frac{1}{8}(1-(2t-1))^2=(1-t)^3$

P_1的系数：

左端＝ $B_{1,3}(t) == 3t(1-t)^2$

$0 \le t \le 1/2$

右端＝ $\frac{1}{2}B_{1,3}(2t) + \frac{1}{2}B_{2,3}(2t) + \frac{3}{8}B_{3,3}(2t)$

$= \frac{1}{2}\cdot 3\cdot 2t\cdot(1-2t)^2 + \frac{1}{2}\cdot 3\cdot(2t)^2(1-2t) + \frac{3}{8}(2t)^3$

$= 3\cdot t\cdot(1-2t)^2 + 6\cdot t^2\cdot(1-2t) + 3t^3$

$= 3t(1-2t+t^2)$

$= 3t(1-t)^2$

$1/2 \le t \le 1$

右端＝ $\frac{3}{8}B_{0,3}(2t-1) + \frac{1}{4}B_{1,3}(2t-1)$

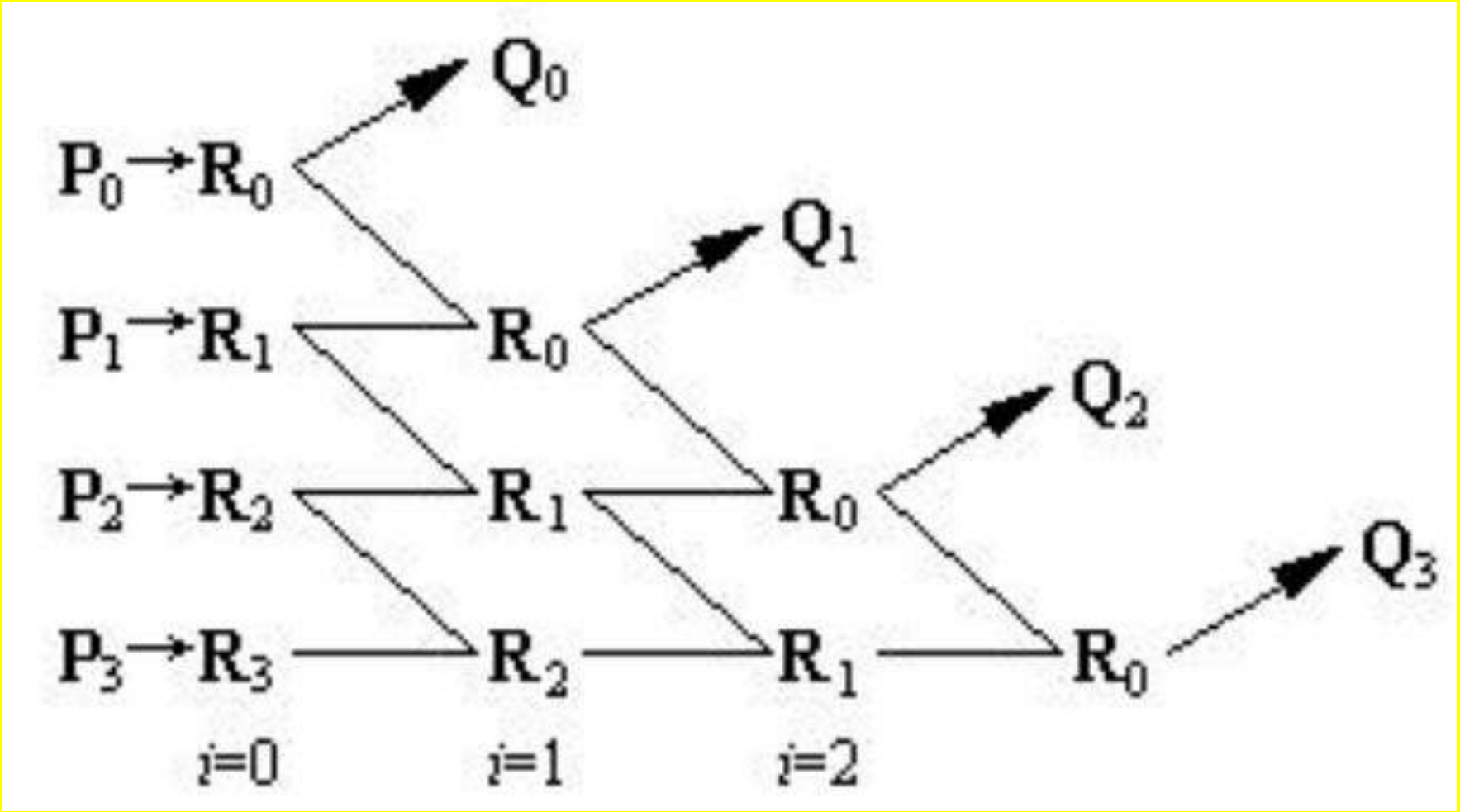
$= \frac{3}{8}(1-(2t-1))^3 + \frac{1}{4}\cdot 3\cdot(2t-1)(1-(2t-1))^2$

$= 3(1-t)^3 + 3(2t-1)(1-t)^2$

$= 3t(1-t)^2$

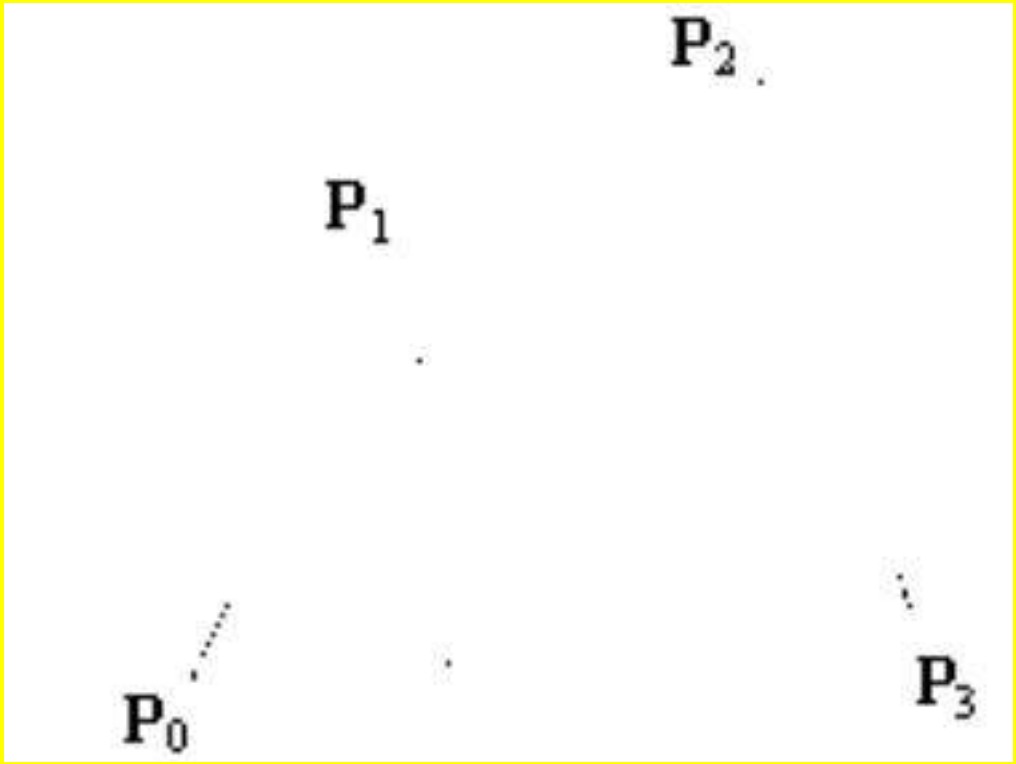
设已知三次Bezier曲线P(*t*)的控制顶点是P_0，P_1，P_2，P_3，在P($\frac{1}{2}$)处将曲线分为两段，求出前半段的控制顶点Q_0，Q_1，Q_2，Q_3和后半段的控制顶点R_0，R_1，R_2，R_3，。有算法如下

```
void split_Bezier(Point P[])
{
        Point R[4],Q[4];
        int i,j;
        for(i=0;i<=3;i++)
        R[i]=P[i];
        for(i=0;i<=2;i++)
            {   Q[i]=R[0];
           for(j=0;j<=2-i;j++)
            {  R[j].x=(R[j].x+R[j+1].x)/2;
//分别对相邻两控制点间的线段进行分裂
                    R[j].y=(R[j].y+R[j+1].y)/2;
                }
            }
        Q[3]=R[0];
 }
```


Q_0
$P_0 \rightarrow R_0$
Q_1
$P_1 \rightarrow R_1$
R_0
Q_2
$P_2 \rightarrow R_2$
R_1
R_0
Q_3
$P_3 \rightarrow R_3$
R_2
R_1
R_0
i=0
i=1
i=2

分裂算法的计算

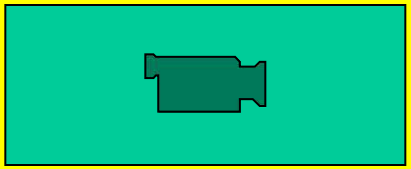
分裂中止的条件：



$$d(P(t),P_0P_3)\leq \max(d(P_1,P_0P_3),d(P_2,P_0P_3))$$

max(*d*(P1，P0P3)，*d*(P2，P0P3))<ε

```
void new_split_Bezier(Point P[])
{    Point R[4],Q[4];
       int i,j;
       const double epsilon=0.01;
       if   (maxdistance(P)<epsilon)
 /*maxdistance(P)为求max(d(P1,P0P3)，d(P2,P0P3))的函数*/
           {     MoveTo(P[0].x,P[0].y);
              LineTo(P[3].x,P[3].y);
        }
         else
       {   for(i=0;i<=3;i++)   R[i]=P[i];
           for(i=0;i<=2;i++)
             {    Q[i]=R[0];
                  for(j=0;j<=2-i;j++)
                  {R[j].x=(R[j].x+R[j+1].x)/2;
                      R[j].y=(R[j].y+R[j+1].y)/2;
                  }
                  }
       Q[3]=R[0];
  new_split_Bezier(Q);new_split_Bezier(R);
       }
}
```



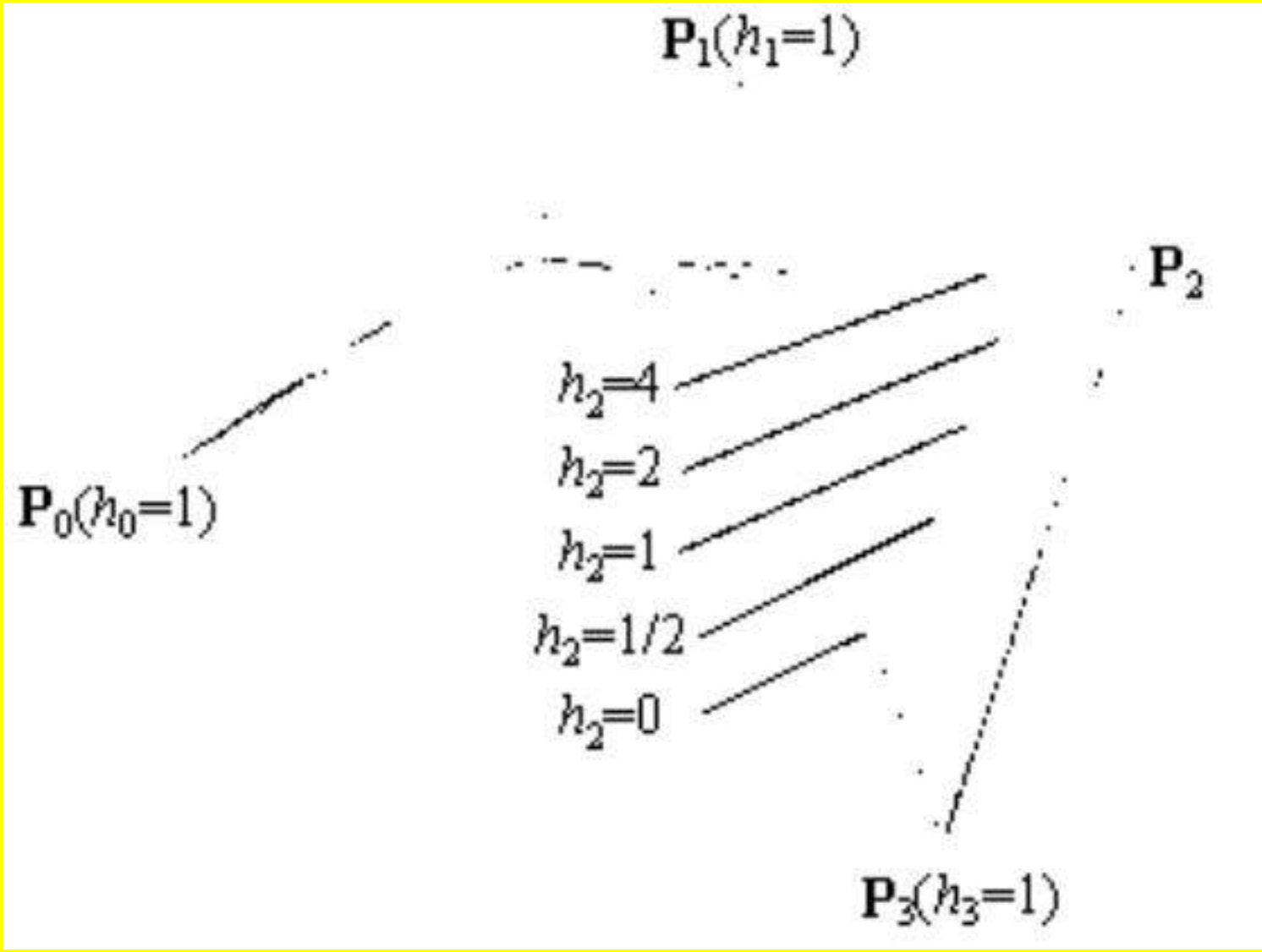
5. 有理Bezier曲线

$$P(t)=\left(\sum_{i=0}^{n} P_i h_i B_{i,n}(t)\right) \Big/ \left(\sum_{i=0}^{n} h_i B_{i,n}(t)\right)$$

$$=\frac{h_0 P_0 B_{0,n}(t)+h_1 P_1 B_{1,n}(t)+\cdots+h_n P_n B_{n,n}(t)}{h_0 B_{0,n}(t)+h_1 B_{1,n}(t)+\cdots+h_n B_{n,n}(t)}$$

$$0 \le t \le 1$$

图中$h_0=h_1=h_3=1$，当$h_2=0、1/2、1、2、4$时曲线逐渐地靠近P_2点



6. Bezier曲面

若在空间给定$(m+1)(n+1)$个控制点，V_{ij}，$i=0,1,\ldots,m$，$j=0,1,\ldots,n$，令

$$P_{m,n}(u,w)=\sum_{i=0}^{m}\sum_{j=0}^{n}B_{i,m}(u)\cdot B_{j,n}(w)\cdot V_{i,j}$$

上式曲面为$m\times n$次的Bezier曲面

$$B_{i,m}(u) = C_m^i u^i (1-u)^{m-i},$$

$$B_{j,n}(w) = C_n^j w^j (1-w)^{n-j}$$

当$m=n=1$，公式成为：

$$P_{1,1}(u,w) = (1-u)(1-w)V_{00}$$
$$+(1-u)wV_{01} + u(1-w)V_{10} + uwV_{11}$$

设v_{00}，v_{01}，v_{10}，v_{11}四点依次是(0，0，0)，(1，0，0)，(0，1，0)，(0，0，1)，则可得$P_{1,1}(u, w)$的坐标形式的参数方程为：

$$\begin{cases} x=w-uw \\ y=u-uw \\ z=uw \end{cases}$$

消去参数，就得**马鞍面**方程：

$$(x+z)(y+z)=z$$

当$m=n=3$，曲面成为：

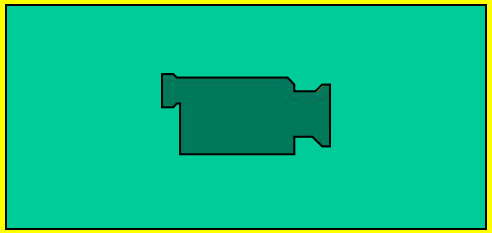
$$P_{3,3}(u,w)=[B_{0,3}(u)\quad B_{1,3}(u)\quad B_{2,3}(u)\quad B_{3,3}(u)]$$

$$\cdot\begin{bmatrix}V_{00}&V_{01}&V_{02}&V_{03}\\V_{10}&V_{11}&V_{12}&V_{13}\\V_{20}&V_{21}&V_{22}&V_{23}\\V_{30}&V_{31}&V_{32}&V_{33}\end{bmatrix}\begin{bmatrix}B_{0,3}(w)\\B_{1,3}(w)\\B_{2,3}(w)\\B_{3,3}(w)\end{bmatrix}$$

$$=[u^3\quad u^2\quad u\quad 1]\begin{bmatrix}-1&3&-3&1\\3&-6&3&0\\-3&3&0&0\\1&0&0&0\end{bmatrix}$$

$$\cdot \begin{bmatrix} V_{00} & V_{01} & V_{02} & V_{03} \\ V_{10} & V_{11} & V_{12} & V_{13} \\ V_{20} & V_{21} & V_{22} & V_{23} \\ V_{30} & V_{31} & V_{32} & V_{33} \end{bmatrix}$$

$$\cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix}$$



$$= [u^3 \quad u^2 \quad u \quad 1] B \cdot V \cdot B^T [w^3 \quad w^2 \quad w \quad 1]^T$$

7. 双三次bezier曲面和双三次Coons曲面之间的关系

设同一个曲面片，用Coons曲面形式确定它，又用Bezier形式确定它，现在来考查、矩阵*M*与矩阵*B*有什么关系。

$$[u^3 \quad u^2 \quad u \quad 1]HMH^T[w^3 \quad w^2 \quad w \quad 1]^T$$

$$=[u^3 \quad u^2 \quad u \quad 1]BVB^T[w^3 \quad w^2 \quad w \quad 1]^T$$

求解上式得：

$$M = H^{-1}BVB^T(H^T)^{-1} = (H^{-1}B)V(H^{-1}B)^T$$

$$V = B^{-1}HMH^T(B^T)^{-1} = (B^{-1}H)M(B^{-1}H)^T$$

$$H=\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad B=\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$H^{-1}B=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix}, \quad B^{-1}H=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & \frac{1}{3} & 0 \\ 0 & 1 & 0 & -\frac{1}{3} \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{M}=\begin{bmatrix} \mathbf{V_{00}} & \mathbf{V_{03}} & \mathbf{3(V_{01}-V_{00})} & \mathbf{3(V_{03}-V_{02})} \\ \mathbf{V_{30}} & \mathbf{V_{33}} & \mathbf{3(V_{31}-V_{30})} & \mathbf{3(V_{33}-V_{32})} \\ \mathbf{3(V_{10}-V_{00})} & \mathbf{3(V_{13}-V_{03})} & \mathbf{9(V_{00}-V_{10}-V_{01}+V_{11})} & \mathbf{9(V_{02}-V_{12}-V_{03}+V_{13})} \\ \mathbf{3(V_{30}-V_{20})} & \mathbf{3(V_{33}-V_{23})} & \mathbf{9(V_{20}-V_{30}-V_{21}+V_{31})} & \mathbf{9(V_{22}-V_{32}-V_{23}+V_{33})} \end{bmatrix}$$

$$M=\begin{bmatrix} 00 & 01 & 00_{w} & 01_{w} \\ 10 & 11 & 10_{w} & 11_{w} \\ 00_{u} & 01_{u} & 00_{uw} & 01_{uw} \\ 10_{u} & 11_{u} & 10_{uw} & 11_{uw} \end{bmatrix}$$

四个角点00，0l，10和11分别对应控制点V_{00}，V_{03}，V_{30}和V_{33}。切向量，例如$00_w=3(V_{01}-V_{00})$，表明00_w，是沿从V_{00}到V_{01}边的方向，大小是两个位置向量差的3倍，其余切向量是类似的。

$$
\begin{aligned}
00_{uw} &= 9(V_{00}-V_{10}-V_{01}+V_{11}) \\
&= 3[3(V_{11}-V_{10})-3(V_{01}-V_{00})] \\
&= 3[3(V_{11}-V_{10})-00_w] \\
&= 3[3(V_{11}-V_{01})-3(V_{10}-V_{00})]
\end{aligned}
$$

V_{10}，V_{00}沿w方向的切向量之差的三倍

V_{01},V_{00} 沿u方向的切向量之差的三倍

$$
\begin{aligned}
01_{uw} &= 9(V_{02} - V_{12} - V_{03} + V_{13}) \\
&= 3[3(V_{13} - V_{03}) - 3(V_{12} - V_{02})] \\
&= 3[3(V_{02} - V_{03}) - 3(V_{12} - V_{13})]
\end{aligned}
$$

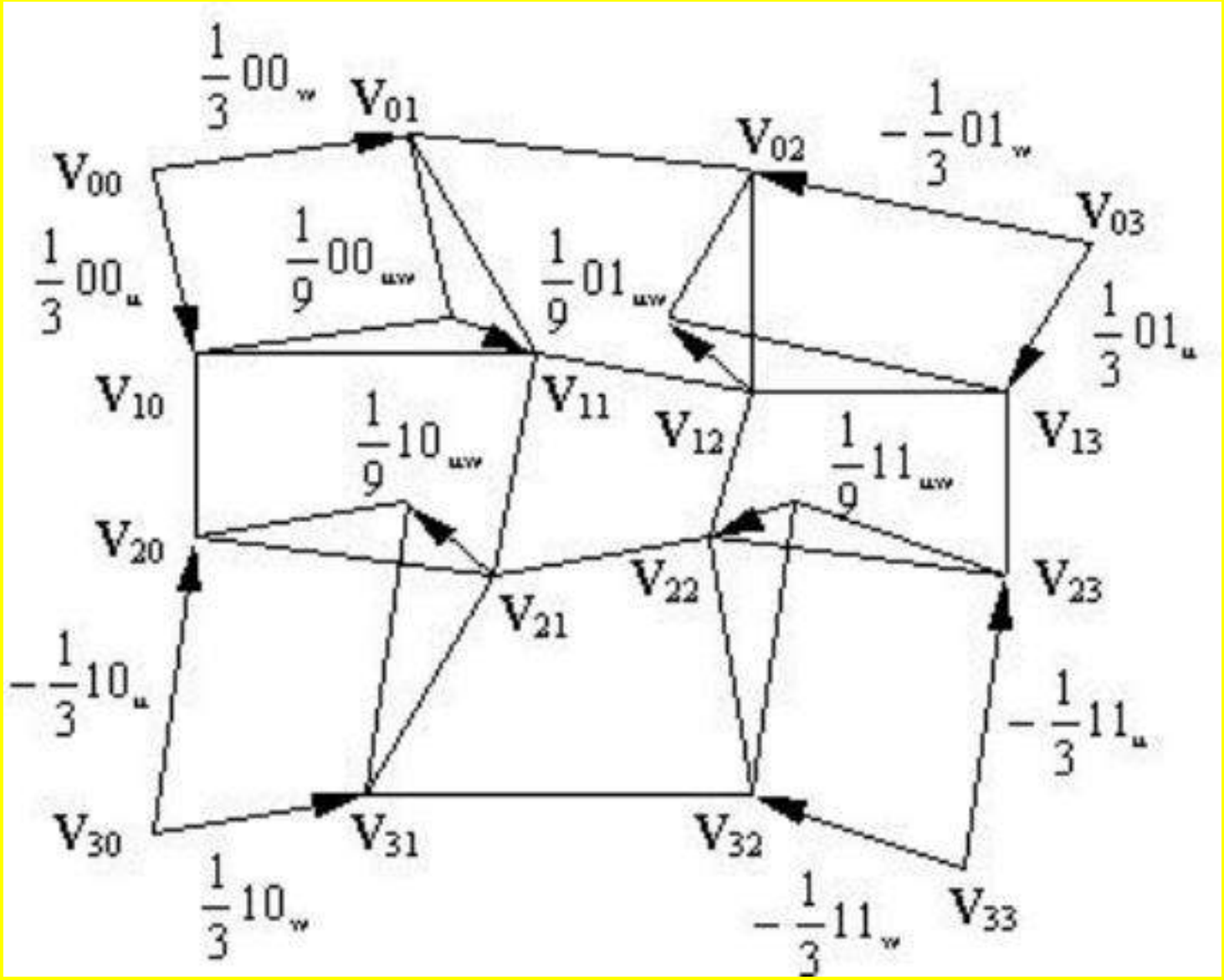
$$
\begin{aligned}
10_{uw} &= 9(V_{20} - V_{30} - V_{21} + V_{31}) \\
&= 3[3(V_{20} - V_{30}) - 3(V_{21} - V_{31})] \\
&= 3[3(V_{31} - V_{30}) - 3(V_{21} - V_{20})]
\end{aligned}
$$

$$
\begin{aligned}
11_{uw} &= 9(V_{22} - V_{32} - V_{23} + V_{33}) \\
&= 3[3(V_{22} - V_{32}) - 3(V_{23} - V_{33})] \\
&= 3[3(V_{22} - V_{23}) - 3(V_{32} - V_{33})]
\end{aligned}
$$

$$V=\begin{bmatrix} 00 & 00+\frac{1}{3}00_w & 01-\frac{1}{3}01_w & 01 \\ 00+\frac{1}{3}00_u & 00+\frac{1}{3}(00_u+00_w)+\frac{1}{9}00_{uw} & 01+\frac{1}{3}(01_u-01_w)-\frac{1}{9}01_{uw} & 01+\frac{1}{3}01_u \\ 10-\frac{1}{3}10_u & 10-\frac{1}{3}(10_u-10_w)-\frac{1}{9}10_{uw} & 11-\frac{1}{3}(11_u+11_w)+\frac{1}{9}11_{uw} & 11-\frac{1}{3}11_u \\ 10 & 10+\frac{1}{3}10_w & 11-\frac{1}{3}11_w & 11 \end{bmatrix}$$

$$V=\begin{bmatrix} V_{00} & V_{01} & V_{02} & V_{03} \\ V_{10} & V_{11} & V_{12} & V_{13} \\ V_{20} & V_{21} & V_{22} & V_{23} \\ V_{30} & V_{31} & V_{32} & V_{33} \end{bmatrix}$$

扭曲向量只与中间四个控制点有关系



双三次Coons曲面与Bezier曲面间的关系

第五节 B样条曲线和曲面

一. B样条曲线（构造具有局部性的调和函数）

给定n+1个控制点P_0，P_1，…，P_n，它们所确定的k阶B样条曲线是：

$$P(u)=\sum_{i=0}^{n} N_{i,k}(u)\cdot P_i$$

其中$N_{i,k}(u)$递归定义如下：

$$\begin{cases} N_{i,1}(u)=\begin{cases}1, & u_i \le u \le u_{i+1}, \ 0 \le i \le n+k-1\\ 0, & \text{其它}\end{cases} \\ N_{i,k}(u)=\dfrac{u-u_i}{u_{i+k-1}-u_i}N_{i,k-1}(u)+\dfrac{u_{i+k}-u}{u_{i+k}-u_{i+1}}N_{i+1,k-1}(u), \\ \qquad\qquad k>1, \ 0 \le i \le n \end{cases}$$

这里u_0，u_1，…，u_{n+k}，是一个**非递减**的序列，称为节点，（u_0，u_1，…，u_{n+k}）称为**节点向量**。定义中可能出现 $\frac{0}{0}$ ，这时约定为0。

例一：选取，$n=2$，$k=1$，控制顶点是P_0，P_1，P_2，这样应选择参数节点$n+k+1=4$个，设节点向量是(u_0，u_1，u_2，u_3)，按式定义，可写出三个基函数：

$$N_{0,1}(u)=\begin{cases}1, & u_0\le u\le u_1\\ 0, & u_1\le u\le u_2\\ 0, & u_2\le u\le u_3\end{cases}$$

$$N_{1,1}(u)=\begin{cases}0, & u_0\le u\le u_1\\ 1, & u_1\le u\le u_2\\ 0, & u_2\le u\le u_3\end{cases}$$

$$N_{2,1}(u)=\begin{cases}0, & u_0\le u\le u_1\\ 0, & u_1\le u\le u_2\\ 1, & u_2\le u\le u_3\end{cases}$$

由公式可知所定义的B样条曲线是

$$P(u)=N_{0,1}(u)P_0+N_{1,1}(u)P_1+N_{2,1}(u)P_2$$

$$=\begin{cases}P_0, & u_0\le u\le u_1\\ P_1, & u_1\le u\le u_2\\ P_2, & u_2\le u\le u_3\end{cases}$$

任意一阶B样条曲线是控制点本身，可以看作是0次多项式

例二. 选取$n=3$，$k=2$，于是有四个控制顶点P_0，P_1，P_2，P_3，应有参数节点$n+k+1=6$个，设节点向量是(0，0，1，2，3，3)，试画出所确定的2阶B样条曲线。

取$u=0.5$，进行计算。因为$0.5\in[0,1]=[u_1, u_2]$，因此$N_{1,1}(0.5)=1$，而其它的$N_{i,1}(0.5)=0$，$i\neq 1$。

$$N_{0,2}(0.5)=\frac{0.5-0}{0-0}\cdot 0+\frac{1-0.5}{1-0}\cdot 1=0.5$$

$$N_{1,2}(0.5)=\frac{0.5-0}{1-0}\cdot 1+\frac{2-0.5}{2-1}\cdot 0=0.5$$

$$
\begin{aligned}
P(0.5) &= N_{0,2}(0.5) \cdot P_0 + N_{1,2}(0.5) \cdot P_1 \\
&\quad + N_{2,2}(0.5) \cdot P_2 + N_{3,2}(0.5) \cdot P_3 \\
&= 0.5 \cdot (P_0 + P_1)
\end{aligned}
$$

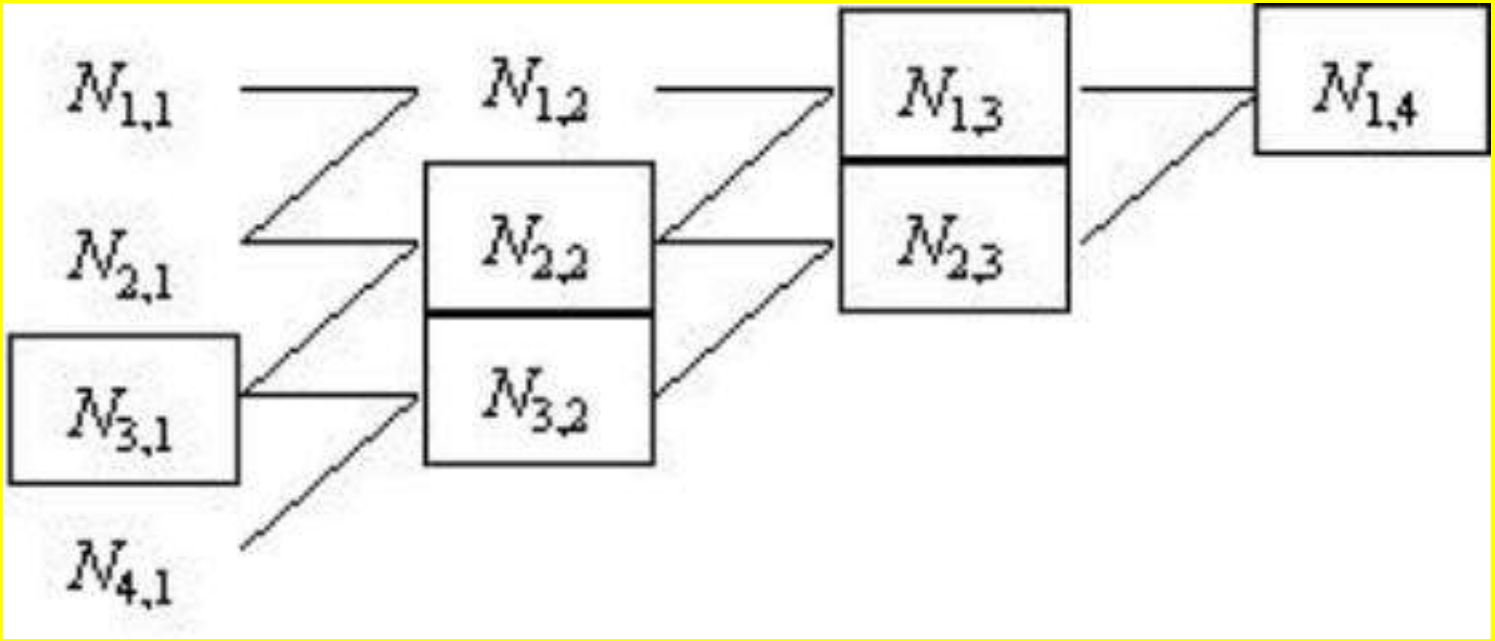
再取u其它一些值进行计算，结果如表所示。

u	0	0.5	1	1.5	2	2.5	3
$N_{0,2}(u)$	1	0.5	0	0	0	0	0
$N_{1,2}(u)$	0	0.5	1	0.5	0	0	0
$N_{2,2}(u)$	0	0	0	0.5	1	0.5	0
$N_{3,2}(u)$	0	0	0	0	0	0.5	1
$P(u)$	P_0	$0.5(P_0+P_1)$	P_1	$0.5(P_1+P_2)$	P_2	$0.5(P_2+P_3)$	P_3

二阶B样条曲线是连接各控制点的线段组成的折线，是一次多项式

例三. 选取$n=3$，$k=4$，平面上四个控制顶点P_0，P_1，P_2，P_3的坐标依次是(1，1)，(2，3)，(4，3)，(3，1)，这时应取参数节点$n+k+1=8$个，设选取节点向量为(0，0，0，0，1，1，1，1)，求所确定的4阶B样条曲线。

计算$N_{1,4}(0.5)$，其中$0.5\in[0,1]=[u_3,u_4]$



$N_{3,1}(0.5)=1$，而对 $i\neq 3$，$N_{i,1}(0.5)=0$。

$$N_{2,2}(0.5)=\frac{0.5-0}{0-0}\times 0+\frac{1-0.5}{1-0}\times 1=0.5$$

$$N_{3,2}(0.5)=\frac{0.5-0}{1-0}\times 1+\frac{1-0.5}{1-0}\times 0=0.5$$

$$N_{1,3}(0.5)=\frac{0.5-0}{1-0}\times 0+\frac{1-0.5}{1-0}\times 0.5=0.25$$

$$N_{2,3}(0.5)=\frac{0.5-0}{1-0}\times 0.5+\frac{1-0.5}{1-0}\times 0.5=0.5$$

$$N_{1,4}(0.5)=\frac{0.5-0}{1-0}\times 0.25+\frac{1-0.5}{1-0}\times 0.5=0.375$$

用类似的过程可计算求出：

$$N_{0,4}(0.5)=0.125$$

$$N_{2,4}(0.5)=0.375$$

$$N_{3,4}(0.5)=0.125$$

曲线上对应参数u=0.5的点是：

$$P(0.5)=N_{0,4}(0.5)\cdot P_0+N_{1,4}(0.5)\cdot P_1$$
$$+N_{2,4}(0.5)\cdot P_2+N_{3,4}(0.5)\cdot P_3$$
$$=(2.75,2.5)$$

证明：四个控制点所确定的四阶B样条曲线（节点向量为（0,0,0,0,1,1,1,1））

$(u_0,u_1,u_2,u_3,u_4,u_5,u_6,u_7)=(0,0,0,0,1,1,1,1)$

就是它们所确定的Bezier曲线

$$u\in[u_3,u_4]\in[0,1]$$

$$N_{2,4}(u)=\frac{u-u_2}{u_5-u_2}N_{2,3}(u)+\frac{u_6-u}{u_6-u_3}N_{3,3}(u)=3u^2(1-u)$$

$$\Uparrow$$

$$\Uparrow$$

$$N_{2,3}(u)=\frac{u-u_2}{u_4-u_2}N_{2,2}(u)+\frac{u_5-u}{u_5-u_3}N_{3,2}(u)=2u(1-u)$$

$$N_{3,3}(u)=\frac{u-u_3}{u_5-u_3}N_{3,2}(u)+\frac{u_6-u}{u_6-u_4}N_{4,2}(u)=u^2$$

$$\Uparrow$$

$$\Uparrow$$

$$N_{2,2}(u)=\frac{u-u_2}{u_3-u_2}N_{2,1}(u)+\frac{u_4-u}{u_4-u_3}N_{3,1}(u)=1-u$$

$$N_{3,2}(u)=\frac{u-u_3}{u_4-u_3}N_{3,1}(u)+\frac{u_5-u}{u_5-u_4}N_{4,1}(u)=u$$

$$N_{4,2}(u)=\frac{u-u_4}{u_5-u_4}N_{4,1}(u)+\frac{u_6-u}{u_6-u_5}N_{5,1}(u)=0$$

$$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

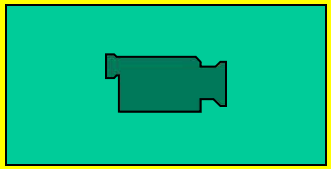
$$P(u)=N_{0,4}(u)P_0+N_{1,4}(u)P_1+N_{2,4}(u)P_2+N_{3,4}(u)P_3$$

Bezier曲线：n+1个控制点$P(t)=\sum_{i=0}^{n} B_{i,n}(t)P_i$——n次

B样条曲线：n+1个控制点$P(t)=\sum_{i=0}^{n} N_{i,k}(t)P_i$——k=n+1,n次

$(t_0,t_1,\cdots t_{n+k})=(t_0,t_1,\cdots t_{2n+1})$,2n+2个节点

t：(0,0,0,0,…1,1,1,1)同bezier曲线



二：等距B样条曲线：

参数节点中参数u的每一区间为等长时所得到的B样条函数称为是等距的，或均匀B样条曲线。可假定$u_i = i$，$i=0$，1，…，$n+k$，设$t_j = u - u_{i+j}$，$u_{i+j} \leqslant u \leqslant u_{i+j+1}$与$0 \leqslant t_j \leqslant 1$是等价的。

递归式，经过计算，可以写出：

$$N_{i,1}(u) = \begin{cases} 1, & u_i \leq u \leq u_{i+1} \\ 0, & \text{其它} \end{cases}$$

$$= \begin{cases} 1, & 0 \leq t_0 \leq 1, t_0 = u - u_i \\ 0, & \text{其它} \end{cases}$$

$$N_{i,2}(u)=\frac{u-u_i}{u_{i+1}-u_i}\cdot N_{i,1}(u)+\frac{u_{i+2}-u}{u_{i+2}-u_{i+1}}\cdot N_{i+1,1}(u)$$

$$=\begin{cases}u-u_i, & u_i\le u\le u_{i+1}\\ u_{i+2}-u, & u_{i+1}\le u\le u_{i+2}\\ 0, & \text{其它}\end{cases}$$

$$=\begin{cases}t_0, & 0\le t_0\le 1, t_0=u-u_i\\ 1-t_1, & 0\le t_1\le 1, t_1=u-u_{i+1}\\ 0, & \text{其它}\end{cases}$$

$$N_{i,3}(u)=\frac{u-u_i}{u_{i+2}-u_i}\cdot N_{i,2}(u)+\frac{u_{i+3}-u}{u_{i+3}-u_{i+1}}\cdot N_{i+1,2}(u)$$

$$=\begin{cases}\frac{1}{2}(u-u_i)^2, & u_i\le u\le u_{i+1}\\ \frac{1}{2}(u-u_i)(u_{i+2}-u) & \\ +\frac{1}{2}(u_{i+3}-u)(u-u_{i+1}), & u_{i+1}\le u\le u_{i+2}\\ \frac{1}{2}(u_{i+3}-u)^2, & u_{i+2}\le u\le u_{i+3}\\ 0, & \text{其它}\end{cases}$$

$$=\begin{cases}\frac{1}{2}t_0^2, & 0\le t_0\le 1, t_0=u-u_i\\ \frac{1}{2}(-2t_1^2+2t_1+1), & 0\le t_1\le 1, t_1=u-u_{i+1}\\ \frac{1}{2}(1-t_2)^2, & 0\le t_2\le 1, t_2=u-u_{i+2}\\ 0, & \text{其它}\end{cases}$$

$$N_{i,4}(u)=\frac{u-u_i}{u_{i+3}-u_i}\cdot N_{i,3}(u)+\frac{u_{i+4}-u}{u_{i+4}-u_{i+1}}\cdot N_{i+1,3}(u)$$

$$=\begin{cases}\frac{1}{6}(u-u_i)^3, & u_i\le u\le u_{i+1}\\ \frac{1}{6}[(u-u_i)^2(u_{i+2}-u)+(u-u_i)(u_{i+3}-u)(u-u_{i+1})+(u_{i+4}-u)(u-u_{i+1})^2], & u_{i+1}\le u\le u_{i+2}\\ \frac{1}{6}[(u-u_i)(u_{i+3}-u)^2+(u-u_{i+1})(u_{i+4}-u)(u_{i+3}-u)+(u_{i+4}-u)^2(u-u_{i+2})], & u_{i+2}\le u\le u_{i+3}\\ \frac{1}{6}(u_{i+4}-u)^3 & u_{i+3}\le u\le u_{i+4}\\ 0, & \text{其它}\end{cases}$$

$$
=\begin{cases}
\frac{1}{6}t_0^3, & 0\le t_0\le 1, t_0=u-u_i \\
\frac{1}{6}(-3t_1^3+3t_1^2+3t_1+1), & 0\le t_1\le 1, t_1=u-u_{i+1} \\
\frac{1}{6}(3t_2^3-6t_2^2+4) & 0\le t_2\le 1, t_2=u-u_{i+2} \\
\frac{1}{6}(1-t_3)^3, & 0\le t_3\le 1, t_3=u-u_{i+3} \\
0, & \text{其它}
\end{cases}
$$

如果固定在$u_{i+3} \leqslant u \leqslant u_{i+4}$区间，可以写出：

$$N_{i,4}(u) = \frac{1}{6}(-t_3 + 1)^3$$

$$N_{i+1,4}(u) = \frac{1}{6}(3t_3^3 - 6t_3^2 + 4)$$

$$N_{i+2,4}(u) = \frac{1}{6}(-3t_3^3 + 3t_3^2 + 3t_3 + 1)$$

$$N_{i+3,4}(u) = \frac{1}{6}t_3^3$$

令 $i=0$，可写出四个B样条基函数：

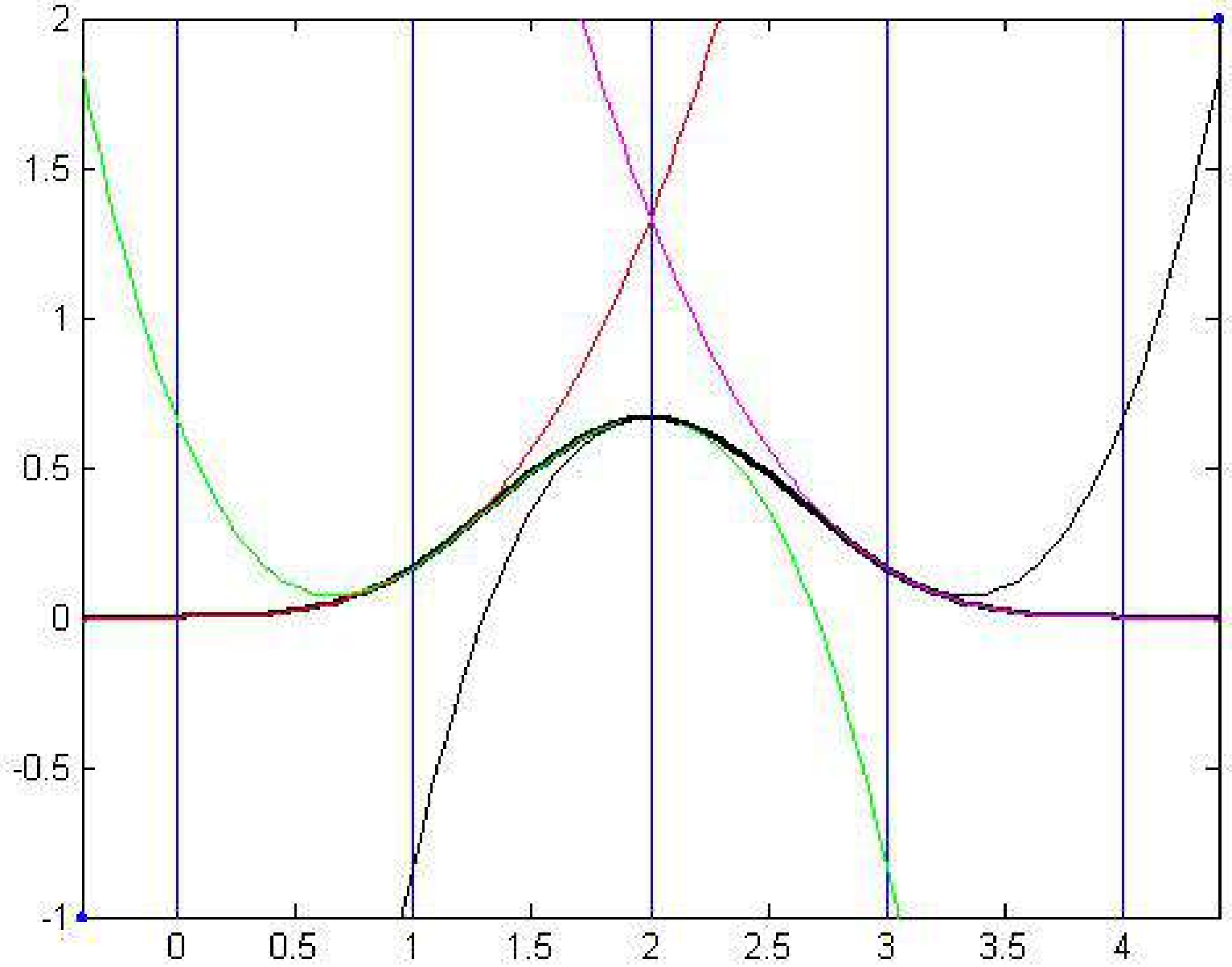
t_3用u代替，只是参数发生变 化，$u \in [0,1]$

$$N_{0,4}(u) = \frac{1}{6}(1-u)^3$$

$$N_{1,4}(u) = \frac{1}{6}(3u^3 - 6u^2 + 4)$$

$$N_{2,4}(u) = \frac{1}{6}(-3u^3 + 3u^2 + 3u + 1)$$

$$N_{3,4}(u) = \frac{1}{6}u^3$$


2
1.5
1
0.5
0
-0.5
-1
0
0.5
1
1.5
2
2.5
3
3.5
4

设给出n+1个控制点P_0，P_1，…，P_n，则所确定的4阶3次等距B样条曲线是：

$$P(u)=Q_i(u)=\sum_{j=0}^{3} N_{j,4}(u)\cdot P_{i+j},$$

$$0\le u\le 1, i=0,1,\cdots,n-3$$

$$Q_i(u) = \begin{bmatrix} N_{0,4}(u) & N_{1,4}(u) & N_{2,4}(u) & N_{3,4}(u) \end{bmatrix} \begin{bmatrix} P_i \\ P_{i+1} \\ P_{i+2} \\ P_{i+3} \end{bmatrix}$$

$$= \frac{1}{6} \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} P_i \\ P_{i+1} \\ P_{i+2} \\ P_{i+3} \end{bmatrix}, \quad 0 \le u \le 1$$

曲线的性质

$$Q_i'(u)=\frac{1}{2}\begin{bmatrix}u^2 & u & 1\end{bmatrix}\begin{bmatrix}-1 & 3 & -3 & 1\\ 2 & -4 & 2 & 0\\ -1 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}P_i\\ P_{i+1}\\ P_{i+2}\\ P_{i+3}\end{bmatrix},\quad 0\le u\le 1$$

$$Q_i''(u)=\begin{bmatrix}u & 1\end{bmatrix}\begin{bmatrix}-1 & 3 & -3 & 1\\ 1 & -2 & 1 & 0\end{bmatrix}\begin{bmatrix}P_i\\ P_{i+1}\\ P_{i+2}\\ P_{i+3}\end{bmatrix},\quad 0\le u\le 1$$

$$Q_i(0) = \frac{1}{6}(P_i + 4P_{i+1} + P_{i+2})$$

$$= P_{i+1} + \frac{1}{3}[\frac{1}{2}(P_i + P_{i+2}) - P_{i+1}]$$

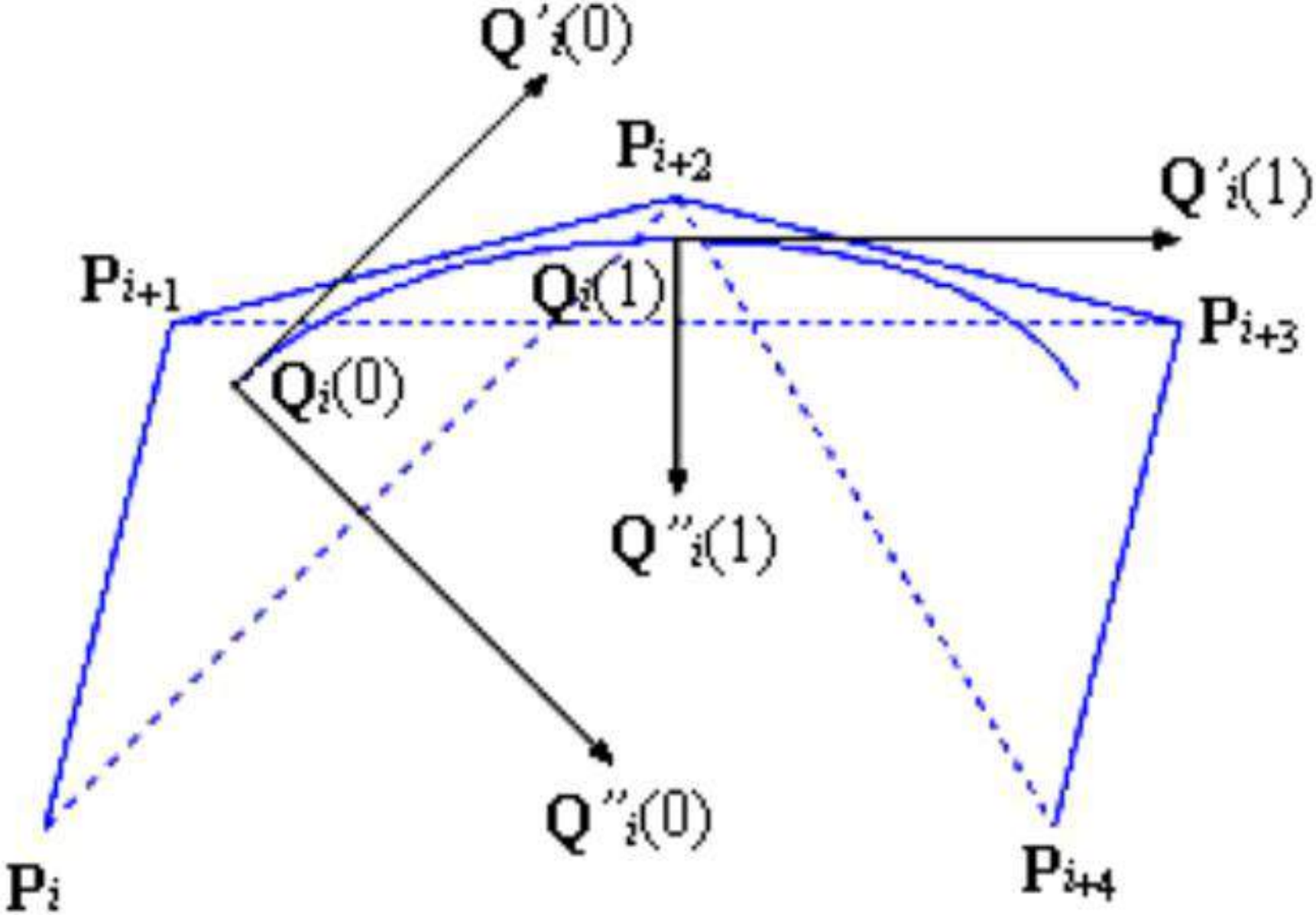
$$Q_i(1) = \frac{1}{6}(P_{i+1} + 4P_{i+2} + P_{i+3})$$

$$= P_{i+2} + \frac{1}{3}[\frac{1}{2}(P_{i+1} + P_{i+3}) - P_{i+2}]$$

$$Q_i'(0) = \frac{1}{2}(P_{i+2} - P_i), Q_i'(1) = \frac{1}{2}(P_{i+3} - P_{i+1})$$

$$Q_i''(0) = P_i - 2P_{i+1} + P_{i+2} = 2[\frac{1}{2}(P_i + P_{i+2}) - P_{i+1}]$$

$$Q_i''(1) = P_{i+1} - 2P_{i+2} + P_{i+3} = 2[\frac{1}{2}(P_{i+1} + P_{i+3}) - P_{i+2}]$$


Q'i(0)
Pi+2
Q'i(1)
Pi+1
Qi(1)
Pi+3
Qi(0)
Q''i(1)
Q''i(0)
Pi
Pi+4

性质一：曲线在拼接处是连续的，一阶和二阶导数也是连续的。

因此4阶3次等距B样条曲线：虽然分段定义，但各段拼接处有直到二阶导数的连续性，整条曲线是光滑的。

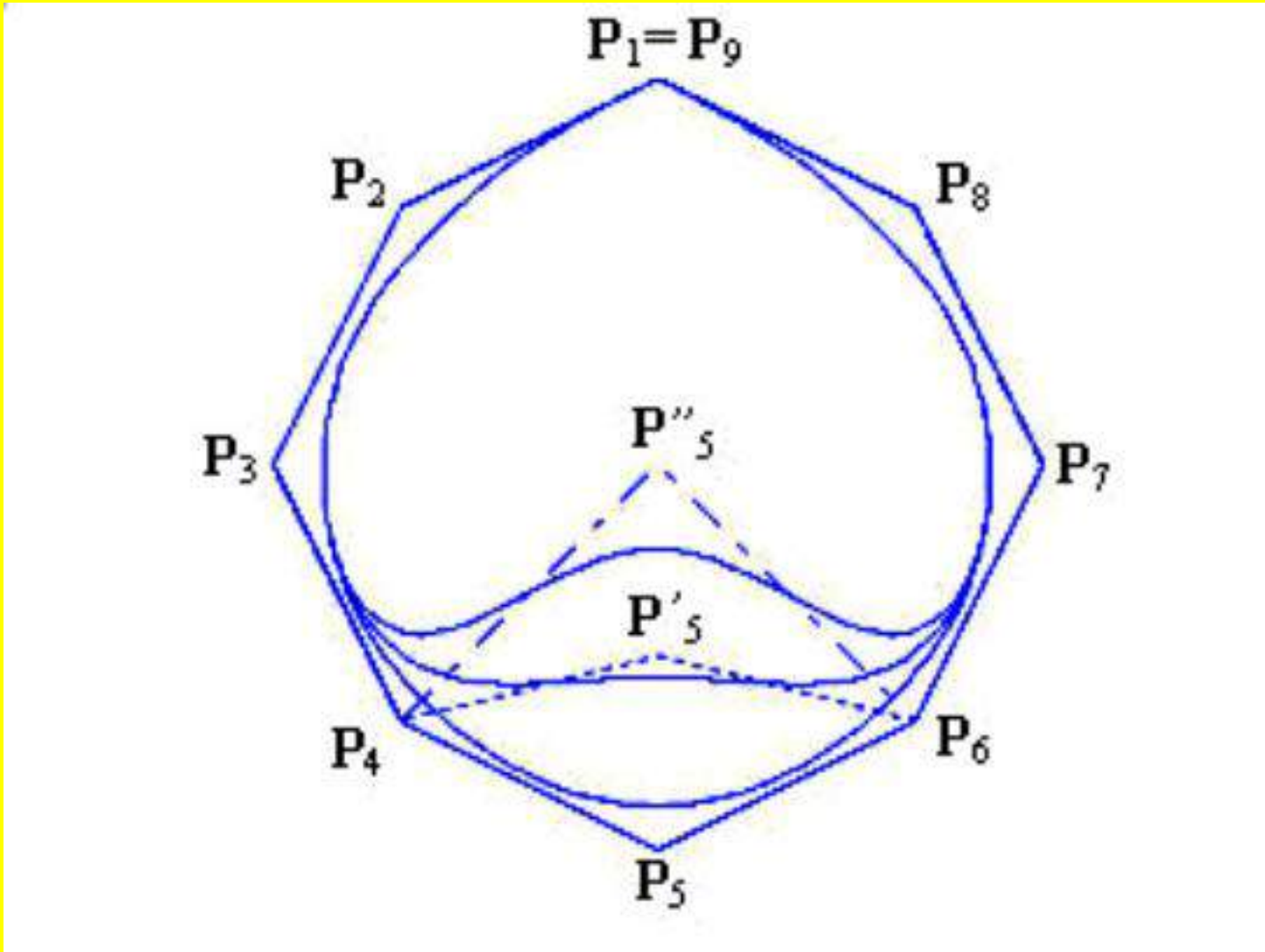
性质2：4阶3次等距B样条曲线具有凸包性，这通过验证0≤$N_{j,4}(u)$≤1，0≤j≤3及

$$\sum_{j=0}^{3} N_{j,4}(u) = 1$$

性质3：B样条曲线具有局部性

$$N_{i,k}(u)\begin{cases} > 0 & u_i \le u \le u_{i+k} \\ = 0 & \text{其它} \end{cases}$$

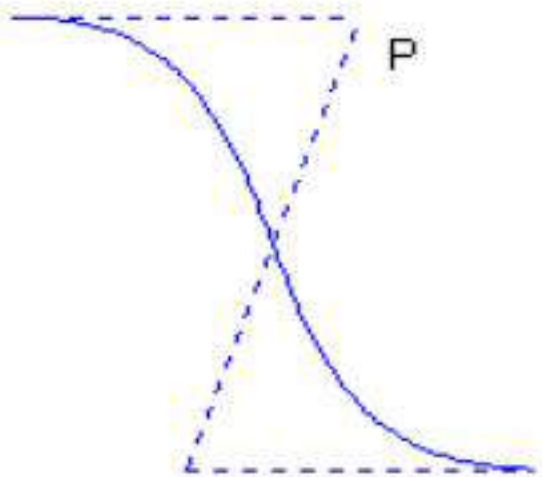
当移动一个控制点时，只对其中的一段曲线有影响，并不对整条曲线产生影响。如图4.22所示是一条B样条曲线。该图表示控制点$\mathbf{P}_5$变化后曲线变化的情况。由图可见$\mathbf{P}_5$变化只对其中一段曲线有影响。


P1=P9
P2
P8
P3
P''5
P7
P'5
P4
P6
P5

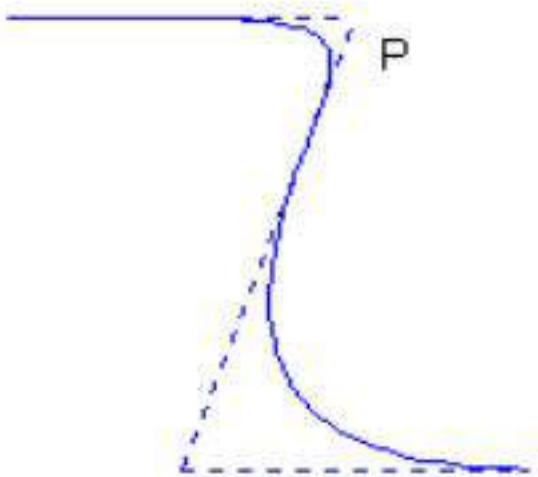
性质4：用B样条曲线可构造直线段、尖点、切线等特殊情况。

灵活选择控制点的位置和节点ui的重复数，可形成许多特殊情况的B样条曲线

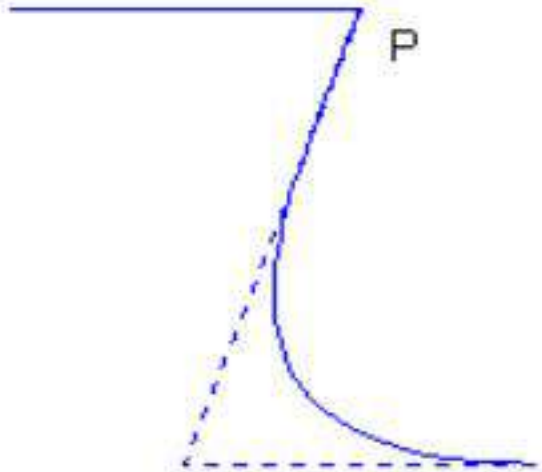
没有重控制点的三次B样条曲线



2重控制点的三次B样条曲线



3重控制点的三次B样条曲线



三. B样条曲面

$$Q_{kl}(u,w)=\sum_{i=0}^{3}\sum_{j=0}^{3} N_{i,4}(u)\cdot N_{j,4}(w)\cdot P_{k+i,l+j}$$

$0\leqslant u\leqslant 1$，$0\leqslant w\leqslant 1$，$0\leqslant k\leqslant K$，$0\leqslant l\leqslant L$

每一个曲面片$Q_{kl}(u, w)$由16个控制点确定。

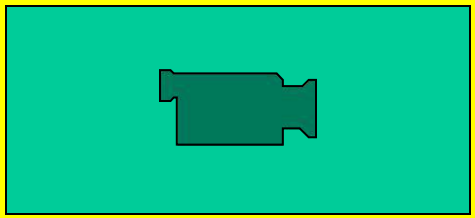
$$Q_{kl}(u,w) = [N_{0,4}(u) \quad N_{1,4}(u) \quad N_{2,4}(u) \quad N_{3,4}(u)]$$

$$\cdot \begin{bmatrix} P_{k,l} & P_{k,l+1} & P_{k,l+2} & P_{k,l+3} \\ P_{k+1,l} & P_{k+1,l+1} & P_{k+1,l+2} & P_{k+1,l+3} \\ P_{k+2,l} & P_{k+2,l+1} & P_{k+2,l+2} & P_{k+2,l+3} \\ P_{k+3,l} & P_{k+3,l+1} & P_{k+3,l+2} & P_{k+3,l+3} \end{bmatrix} \begin{bmatrix} N_{0,4}(w) \\ N_{1,4}(w) \\ N_{2,4}(w) \\ N_{3,4}(w) \end{bmatrix}$$

$$= \frac{1}{36} [u^3 \quad u^2 \quad u \quad 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} P_{k,l} & P_{k,l+1} & P_{k,l+2} & P_{k,l+3} \\ P_{k+1,l} & P_{k+1,l+1} & P_{k+1,l+2} & P_{k+1,l+3} \\ P_{k+2,l} & P_{k+2,l+1} & P_{k+2,l+2} & P_{k+2,l+3} \\ P_{k+3,l} & P_{k+3,l+1} & P_{k+3,l+2} & P_{k+3,l+3} \end{bmatrix}$$

$$\cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 0 & 4 \\ -3 & 3 & 3 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix}$$



$$= \frac{1}{36} [u^3 \quad u^2 \quad u \quad 1] \cdot B \cdot V_{kl} \cdot B^T \cdot [w^3 \quad w^2 \quad w \quad 1]^T$$

第四章　　作业

3. 两段曲线连续的条件：P1=Q0;

二阶导数连续的条件是：P1’=Q0’

P1’’=Q0’’

两段hermite曲线光滑拼接条件：（二阶几何连续）

P1=Q0;

P1’=k1*Q0’

P1’’=k2*Q0’’

9. $$P(\frac{1}{5}) = (\frac{211}{125}, \frac{49}{25})$$

$$P_{0,n}(t) = P_{0,n-1}(t) + t(P_{1,n}(t) - P_{0,n-1}(t))$$

10.

前半段$Q_0(1,1), Q_1(\frac{3}{2},2), Q_2(\frac{9}{4},\frac{5}{2}), Q_3(\frac{11}{4},\frac{5}{2})$

后半段$R_0(\frac{11}{4},\frac{5}{2}), R_1(\frac{13}{4},\frac{5}{2}), R_2(\frac{7}{2},2), R_3(3,1)$

15. 利用凸包性以及起点和终点位置的关系求得

16.

Q1, Q2, P1, P2　　S为B样条曲线的起点

$$S' = \frac{1}{2}(P_1 - Q_1) \Rightarrow Q_1$$

$$S = Q_2 + \frac{1}{3}\left[\frac{1}{2}(P_1 + Q_1) - Q_2)\right] \Rightarrow Q_2$$

Q1, Q2, P1, P2　　S为B样条曲线的终点

$$S' = \frac{1}{2}(P_2 - Q_2) \Rightarrow Q_2$$

$$S = P_1 + \frac{1}{3}\left[\frac{1}{2}(P_2 + Q_2) - P_1)\right] \Rightarrow Q_2$$

只能求出Q_2,无法求出Q_1

21. P0 (−18, −39)　　P3 (27, 15)

P0, V1, V2, V3

24.
$$\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_4 \\ P_1' \\ P_4' \end{bmatrix} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{bmatrix}$$

$$\Rightarrow \begin{cases} B_1 = P_1 \\ B_2 = \frac{1}{3} P_1' + P_1 \\ B_3 = P_4 - \frac{1}{3} P_4' \\ B_4 = P_4 \end{cases}$$

$$\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_4 \\ P_1' \\ P_4' \end{bmatrix} = \frac{1}{6} \times \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix}$$

$$\Rightarrow \begin{cases} V_1 = 2P_4 - P_1 - \frac{7}{3}P_1' - \frac{2}{3}P_4' \\ V_2 = 2P_1 - P_4 + \frac{2}{3}P_1' + \frac{1}{3}P_4' \\ V_3 = 2P_4 - P_1 - \frac{1}{3}P_1' - \frac{2}{3}P_4' \\ V_4 = 2P_1 - P_4 + \frac{2}{3}P_1' + \frac{7}{3}P_4' \end{cases}$$

27. 三次等距B样条曲线，起点和终点分别是单位圆在x轴, y轴上的点，利用三角形的关系求出控制点

第五章　图形运算

第一节　线段的交点计算

一. 两条线段求交

设有两线段AB和CD, 其端点坐标分别为(x_a, y_a), (x_b, y_b)和(x_c, y_c), (x_d, y_d), 它们所在直线的参数方程分别为:

$$\begin{cases} x = x_a + \lambda (x_b - x_a) \\ y = y_a + \lambda (y_b - y_a) \end{cases}$$

$$\begin{cases} x = x_c + \mu (x_d - x_c) \\ y = y_c + \mu (y_d - y_c) \end{cases}$$

若两线段相交，则交点的参数值，应满足：

$$\begin{cases} x = x_a + \lambda (x_b - x_a) = x_c + \mu (x_d - x_c) \\ y = y_a + \lambda (y_b - y_a) = y_c + \mu (y_d - y_c) \end{cases}$$

即

$$\begin{cases} (x_b - x_a)\lambda - (x_d - x_c)\mu = x_c - x_a \\ (y_b - y_a)\lambda - (y_d - y_c)\mu = y_c - y_a \end{cases}$$

因此，若行列式

$$\Delta = \begin{vmatrix} x_b - x_a & -(x_d - x_c) \\ y_b - y_a & -(y_d - y_c) \end{vmatrix} = 0$$

表示两线段AB和CD重合或平行。一般做为它们不相交来处理。如果 $\Delta \neq 0$，则可求出交点对应的两个参数值：

$$\begin{cases} \lambda = \dfrac{1}{\Delta}\begin{vmatrix} x_c - x_a & -(x_d - x_c) \\ y_c - y_a & -(y_d - y_c) \end{vmatrix} \\ \mu = \dfrac{1}{\Delta}\begin{vmatrix} x_b - x_a & (x_c - x_a) \\ y_b - y_a & (y_c - y_a) \end{vmatrix} \end{cases}$$

$0 \le \lambda \le 1 \quad 0 \le \mu \le 1$

两线段AB和CD交点的算法

1.〔计算行列式〕

$$\Delta \leftarrow (x_b-x_a)(y_c-y_d)-(x_c-x_d)(y_b-y_a)$$

若 $\Delta=0$, 则两线段重合或平行, 可算做无交点, 算法结束;

2.〔计算交点参数〕

$$\lambda \leftarrow ((x_c-x_a)(y_c-y_d)-(x_c-x_d)(y_c-y_a))/\Delta$$

若 $\lambda<0$或 $\lambda>1$, 则无交点, 算法结束;

$$\mu \leftarrow ((x_b-x_a)(y_c-y_a)-(x_c-x_a)(y_b-y_a))/\Delta$$

若 $\mu<0$或 $\mu>1$, 则无交点, 算法结束;

3.〔计算交点〕

$x \leftarrow x_a+\lambda(x_b-x_a)$, $y \leftarrow y_a+\lambda(y_b-y_a)$, 输出交点(x, y)后算法结束;

二. 多条线段求交

① 穷举法(很多线段不相交时，效率低，计算浪费）

②扫描线思想求交（高效算法只对有可能相交的两线段计算交点, 对不可能相交的线段不计算交点）

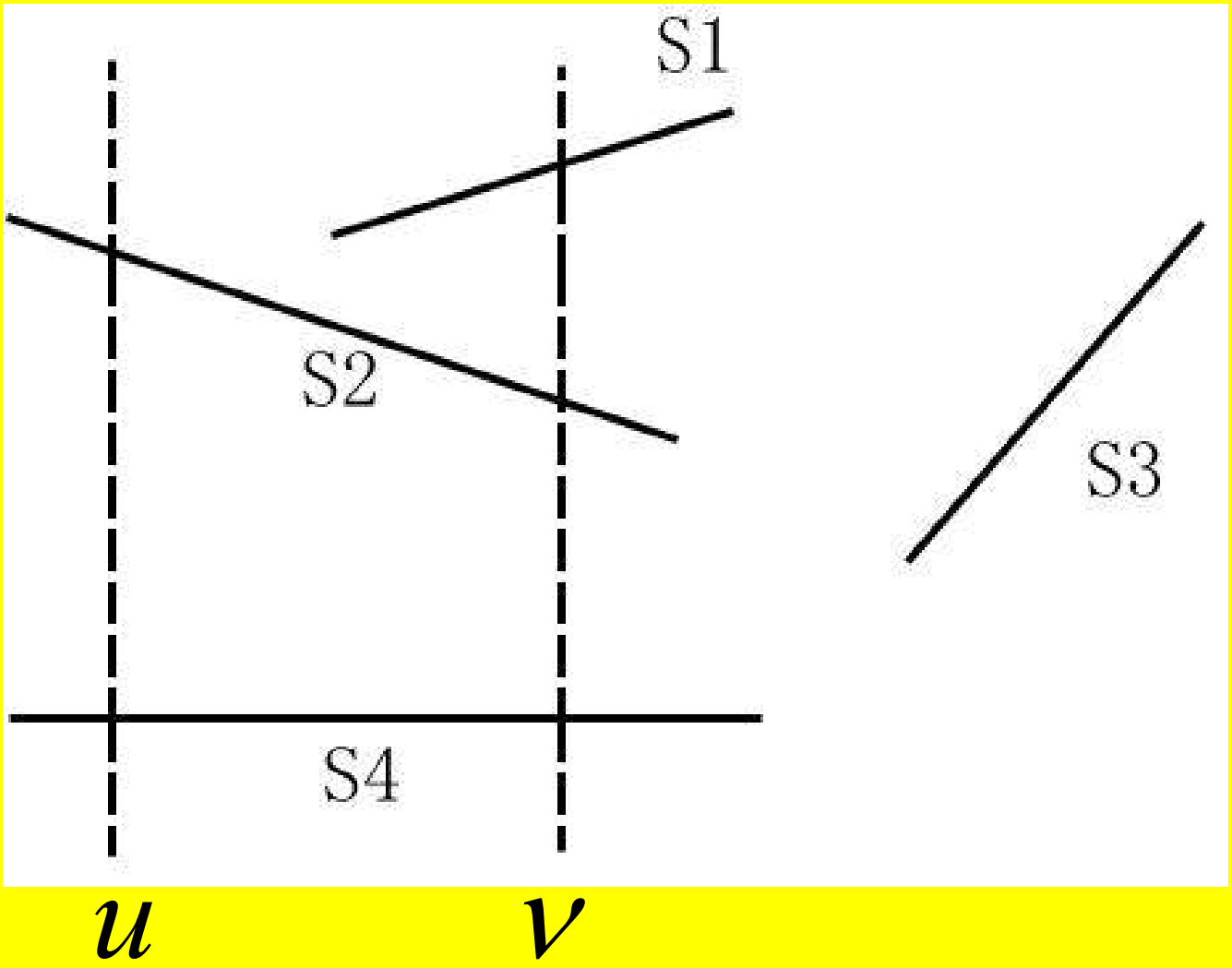
. 定义关系

线段之间是可比较的（两条线段与同一垂线有交点）

x处的“上面”关系为：

在x处, 线段S_1在S_2的上面, 记为$S_1 >_x S_2$, 如果在x处可比较, 且S_1

与垂直线的交点位于S_2与垂直线的交点的上面。



其中，$S_2 >_{\mu} S_4$, $S_1 >_{v} S_2$, $S_2 >_{v} S_4$, $S_1 >_{v} S_4$

假设：为简便起见，要求交点的n条线段中没有垂直的线段，

也没有三条以上的线段交于一点的情况

两线段相交的必要条件：

若两线段相交，则必然存在某个x，使它们在规定的次序关系>x下是相邻的。

从左向右扫描线变化，线段之间的种次序关系只能有三种可能的变化方式：

1. 遇见某条线段S的左端点，此时S应加入次序关系。
2. 遇见某线段S的右端点，此时S应从次序关系中删除。
3. 遇到某两条线段S_1和S_2 的交点，这时在次序关系中S_1和S_2交换位置。

算法的数据结构和实现过程：

扫描线状态表L：存放按次序关系$>_x$排序的线段的序列。初始为空，扫描过程中当关系$>_x$改变时变化。

事件点进度表E：存放事件点的表，初始为n条线段的2n个端点，计算的交点插入该表。

事件点：扫描过程中可能使次序关系$>_x$发生变化的点，

扫描线状态表应能支持以下四个操作:

(1) INSERT(S,L),把线段S插入到扫描线状态表L中,注意应插入到适当位置以保持正确的次序关系。

(2) DELETE(S,L),从L中删除线段S。

(3) ABOVE(S,L),返回次序关系中S上面紧接着的线段的编号。

(4) BELOW(S,L),返回次序关系中S下面紧接着的线段的编号。

事件点进度表E应能支持以下三个操作:

(1) MIN(E),取出表E中的最小元素。

(2) INSERT(x,E),把横坐标为x的一个点插入到表
E中,插入要使E中事件点存放保持递增次序。

(3) MEMBER(x,E),判定横坐标为x的点是否在事件
点进度表E中。

算法:

1 初始化E表：n条线段的2n个端点按x,y字典式排序后存放于表E中;

2. A← ϕ ;{A是一集合,初为空,准备存入找到的交点;}

3. 若表E不为空,则进行3.1~3.3循环。表E为空时算法结束。

3.1 P←MIN(E);

3.2 考查当前事件点P,分三种情况:

(1) 若P是边S的左端点,则做：INSERT(S,L);

S_1=ABOVE(S,L);

S_2=BELOW(S,L);

若S和S_1相交,则求出的交点送入集合A中;

若S和S_2相交,则求出的交点送入集合A中;

goto 3.3

(2) 若P是边S的右端点,则做：

S_1=ABOVE(S,L);

S_2=BELOW(S,L);

若S_1和S_2相交于点P的右边,则求出的交点送入集合A中;

DELETE(S,L);

goto 3.3

(3) 若P是边S_1和S_2的交点,且在P的左边S_1=ABOVE(S_2),则做

S_3=ABOVE(S_1,L);

S_4=BELOW(S_2,L);

若S_3和S_2相交，则求出的交点送入集合A中；

若S_4和S_1相交，则求出的交点送入集合A中；

在L中交换S_1和S_2的位置；

Goto 3.3

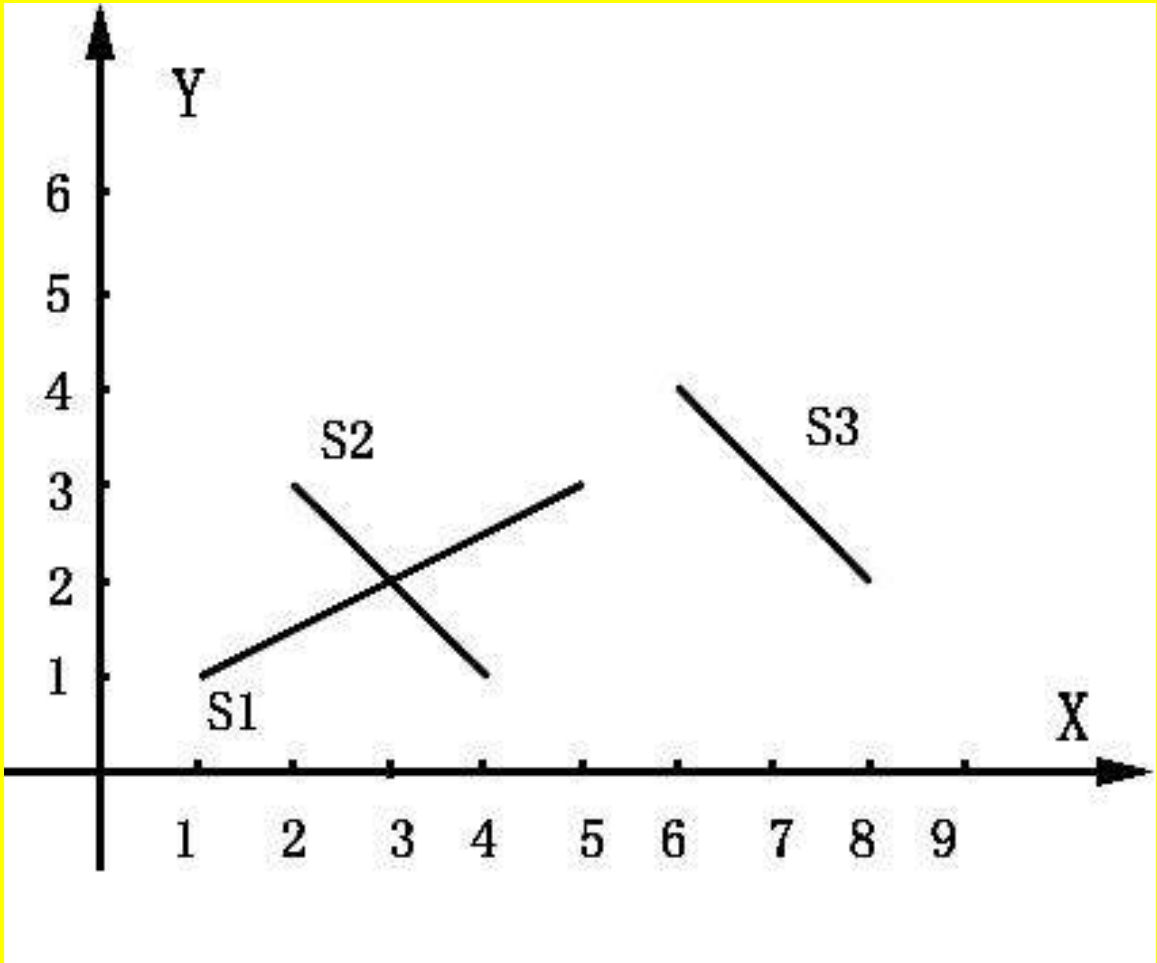
3.3 若集合A不为空，做下面循环，直至A为空：

取出集合A中一个交点，其横坐标是x；

若MEMBER(x,E)为FALSE,则输出此交点， 并做 INSERT(x,E);

(交点是不是第一次求得）

设有三条线段S_1，S_2，S_3，它们的坐标如下(1, 1)，(5, 3,)，(2, 3)，(4, 1)，(6, 4)，(8, 2)。要计算所有交点。



算法初始形成的事件点进度表E：

(((1,1),S1左端点)，((2,3),S2左端点)，((4,1),S2右端点)，((5,3),S1右端点)，((6,4),S3左端点)，((8,8),S3右端点))

算法步骤	从表E前面取出的扫描到达的事件点P	扫描线状态表L	工作解释
3.2(1)	((1,1),S1左端点)	(S1)	
3.2(1)	((2,3),S2左端点)	(S1,S2)	发生S1,S2求交,求出交点(3,2)插入E((4,1),S2右端点)前
3.2(3)	((3,2),S1和S2的交点	(S2,S1)	
3.2(2)	((4,1),S2右端点)	(S1)	
3.2(2)	((5,3),S1右端点)	( )	
3.2(1)	((6,4),S3左端点)	(S3)	
3.2(2)	((8,8),S3右端点)	( )	

第二节　多边形表面的交线计算

多边形：空间多面体的表面，设为凸多边形

求交步骤：(已知两个多边形定点序列）

1.根据顶点坐标求出两个多边形表面分别所在平面的方程

2.根据平面方程计算交线

3.确定出交线同时在两个多边形表面内部的部分

1. 求平面方程

采用多个顶点位置坐标来计算平面方程.

设要求出通过若干顶点的平面方程 $Ax+By+Cz+D=0$, 即要定出系数A, B, C, D, 可采用如下做法

平面方程$Ax+By+Cz+D=0$的系数A, B, C与该平面上多边形分别在$x=0$, $y=0$, $z=0$三个坐标平面上投影的面积成比例

证明：设三角形三点 $(x_1,y_1,z_1),(x_2,y_2,z_2),(x_3,y_3,z_3)$

所在的平面方程为 $Ax+By+Cz+D=0$

法向量为（ A,B,C)

$P_1P_2\times P_1P_3$为平面法向量

P_1P_2=$(x_2-x_1,y_2-y_1,z_2-z_1)$

$P_1P_3=(x_3-x_1,y_3-y_1,z_3-z_1)$

$$P_1P_2\times P_1P_3=\begin{vmatrix} i & j & k \\ x_2-x_1 & y_2-y_1 & z_2-z_1 \\ x_3-x_1 & y_3-y_1 & z_3-z_1 \end{vmatrix}$$

i, j, k的系数：

$$\left(\begin{vmatrix} y_2-y_1 & z_2-z_1 \\ y_3-y_1 & z_3-z_1 \end{vmatrix},-\begin{vmatrix} x_2-x_1 & z_2-z_1 \\ x_3-x_1 & z_3-z_1 \end{vmatrix},\begin{vmatrix} x_2-x_1 & y_2-y_1 \\ x_3-x_1 & y_3-y_1 \end{vmatrix}\right)$$

$$(A,B,C) = k\left(\begin{vmatrix} y_2 - y_1 & z_2 - z_1 \\ y_3 - y_1 & z_3 - z_1 \end{vmatrix}, -\begin{vmatrix} x_2 - x_1 & z_2 - z_1 \\ x_3 - x_1 & z_3 - z_1 \end{vmatrix}, \begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{vmatrix}\right)$$

$$S_z = \frac{1}{2}\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = \frac{1}{2}\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 - x_1 & y_2 - y_1 & 0 \\ x_3 - x_1 & y_3 - y_1 & 0 \end{vmatrix} = \frac{1}{2}\begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{vmatrix}$$

$$C = k \cdot 2 \cdot S_z$$

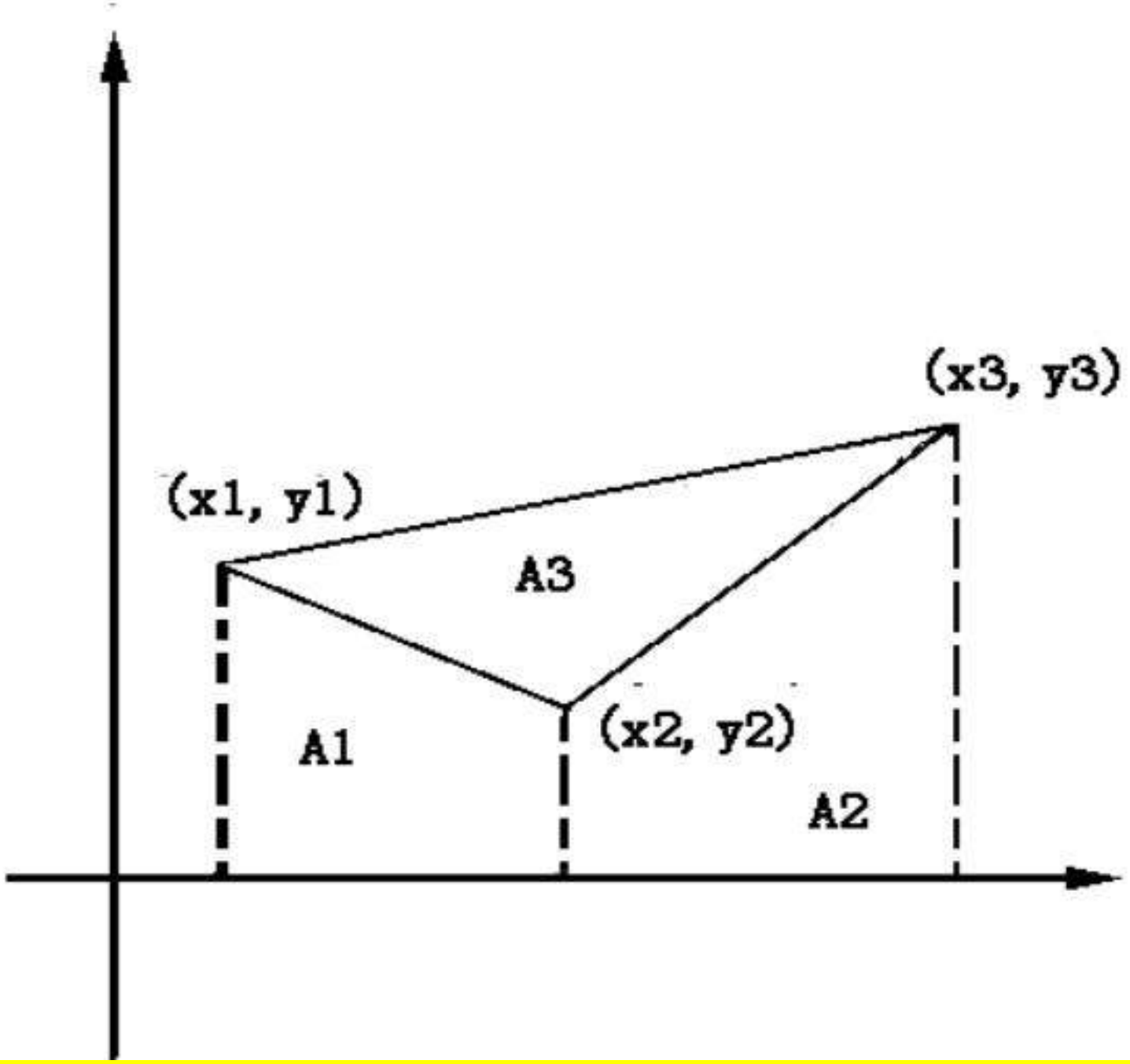
$$A = k \cdot 2 \cdot S_x$$

$$B = k \cdot 2 \cdot S_y$$

多边形在z=0平面上投影的面积S可如下求出：

$$S=\frac{1}{2}\sum_{i=1}^{n}(x_i-x_j)(y_i+y_j)$$

式中若i=n则j=1，否则j=i+1。


(x3, y3)
(x1, y1)
A3
A1
(x2, y2)
A2

$$-A1=\frac{1}{2}(y1+y2)\,(x1-x2)$$

$$-A2=\frac{1}{2}(y2+y3)\,(x2-x3)$$

$$A1+A2+A3=\frac{1}{2}(y1+y3)\,(x3-x1)$$

$$S=\frac{1}{2}\left[(y1+y2)\,(x1-x2)+(y2+y3)\,(x2-x3)+(y3+y1)\,(x3-x1)\right]$$

若给出空间若干点的坐标
(x1, y1, z1), (x2, y2, z2), ... (xn, yn, zn), 求所确定的平面方程。

$$A = \frac{1}{2}\sum_{i=1}^{n}(y_i - y_j)(z_i + z_j)$$

$$B = \frac{1}{2}\sum_{i=1}^{n}(z_i - z_j)(x_i + x_j)$$

$$C = \frac{1}{2}\sum_{i=1}^{n}(x_i - x_j)(y_i + y_j)$$

D=-Ax1-By1-Cz1

式中若i=n, 则j=1, 否则j=i+1

2. 平面方程的求交

$$A_1x+B_1y+C_1z+D_1=0$$
$$A_2x+B_2y+C_2z+D_2=0$$

$$\frac{A_1}{A_2}=\frac{B_1}{B_2}=\frac{C_1}{C_2}$$

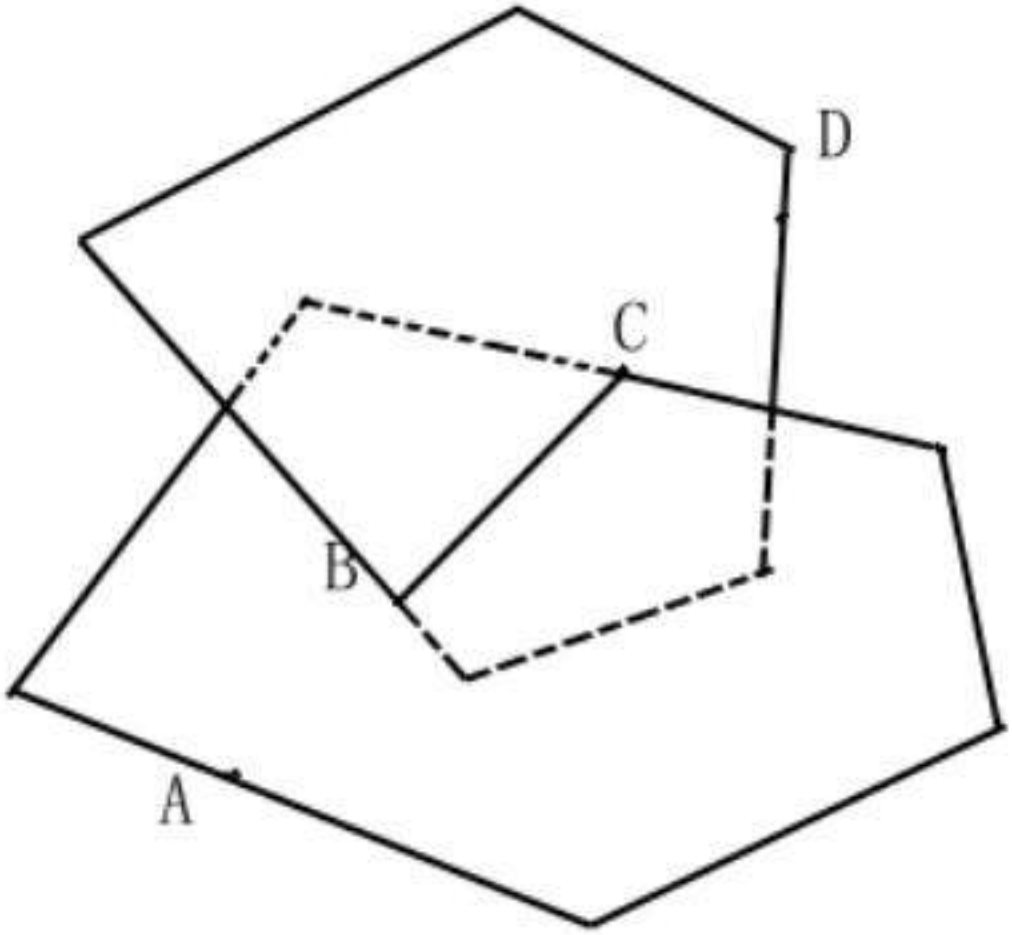
两平面重合或平行，一般算没有交点

3. 确定交线同时在两个多边形内部的部分

分别对每个多边形表面各边相应的线段，计算它与另一个多边形表面所在平面的交点。

注意：求线段（有限）与平面（无限）的交点，即交点在线段延长线上时算不相交。

假定两个多边形表面都是凸的，故共可以交出四个交点。


D
C
B
A

求线段（有限）与平面（无限）的交点

空间线段两个端点的坐标(x1, y1, z1)和x2, y2, z2)给出, 平面方程Ax +By+Cz+D=0。

$$\begin{cases} x = x_1 + (x_2 - x_1)t \\ y = y_1 + (y_2 - y_1)t \\ z = z_1 + (z_2 - z_1)t \end{cases}$$

$A(x_1+(x_2-x_1)t)+B(y_1+(y_2-y_1)t)+C(z_1+(z_2-z_l)t)+D=0$

$[A(x_2-x_1)+B(y_2-y_1)+C(z_2-z_l)]t=-(Ax_1+By_1+Cz_1+D)$

若$A(x_2-x_1)+B(y_2-y_1)+C(z_2-z_1)=0$

即（A,B,C)和（ $x_2-x_1, y_2-y_1, z_2-z_1$)点积为零 则所给线段在平面上或与平面平行,没有唯一确定的交点。

否则,交点对应的参数t可以求出:

$$t=-\frac{Ax_1+By_1+Cz_1+D}{A(x2-x1)+B(y2-y1)+C(z2-z1)}$$

若t<0或t>1,交点在线段延长线上，没有交点

若 $0\le t\le 1$ ，将参数t代回直线的参数方程求出交点的坐标

第三节 平面中的凸壳算法

凸壳 包含一个平面点集的最小凸区域

凸区域指要求区域内任意两点的连线仍在该区域内。

设S是平面上n个点的集合，则S的凸壳是一个凸多边形，它包含所有n点且面积最小。

求点集S的凸壳：

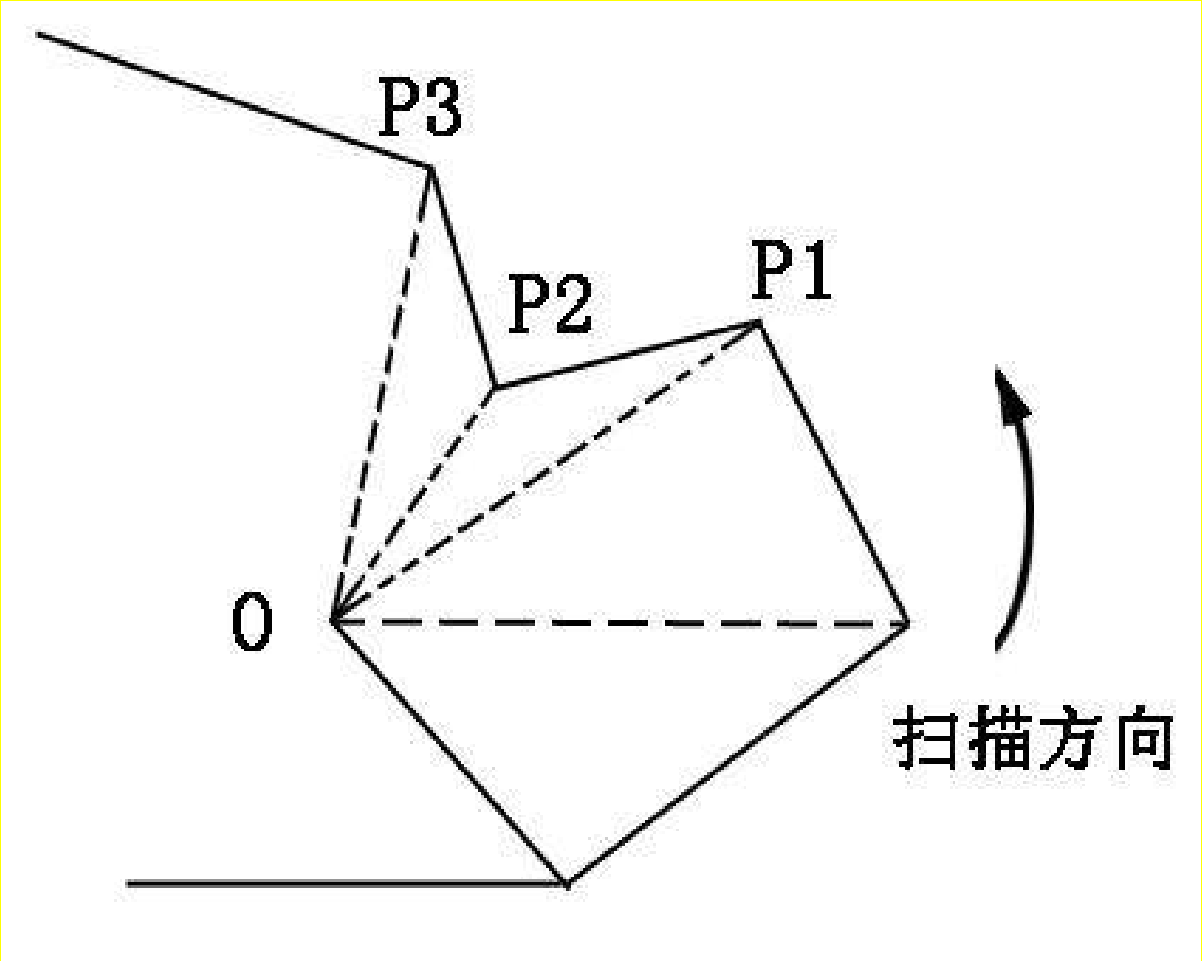
1）在S中选出壳上的点

2）给出围成凸多边形的序列。

- 1.Graham扫描算法

思想：设有一点O，设为原点,对点集S中所有的点计算与OX轴的倾角，并按照倾角递增排序,若某一点不是凸顶点,则它必然在两个壳顶点与点O形成的三角形内部,则删除这个内点。

Graham扫描的实质是围绕已经按"倾角"排序的各顶点进行一次扫描,在扫描过程中消去在凸壳内部的点,留下以希望次序排列的壳顶点。



由于是按倾角递增排序，可知若连续三个顶点P_1, P_2, P_3是一个“右转”，则P_2是一个应去掉的内点。

对给出的三点P_1, P_2, P_3，设它们的坐标是$(x_1, y_1), (x_2, y_2), (x_3, y_3)$，这时要判断三点在$P_2$处形成一个右转还是左转，可以计算下面的行列式

$$\Delta = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

其中△给出的是带有正负号的三角形$P_1P_2P_3$面积的2倍，因此若$\Delta>0$，则P_1, P_2, P_3是左转；$\Delta<0$，则是右转；$\Delta=0$，则三点共线。

Graham扫描算法

1.〔倾角排序〕确定O点，设想有一条从O出发水平向右的射线OX，对S中其余各点P计算相对于OX的倾角POX，之后按倾角递增排序，得到点序列Q_1=O, Q_2, Q_3…, Q_n；

2.〔准备扫描〕v←Q_1；

3.〔扫描〕

if NEXT(v)=Q1,算法结束，序列中剩余的点就是凸壳上的顶点

else if v,NEXT(v),NEXT(NEXT(v))形成一个左转

then v←NEXT(v)

else 删除NEXT(v), v←PRED(v);

goto 3.

next(v)：返回点序列中的v的下一个点（后继节点）

pred(v)：返回点序列中的v的前一个点（前驱节点）

循环链表 next(Qn)=Q1,pred(Q1)=Qn

倾角计算

记P点坐标为(Xp, Yp)，O点坐标(X_0, Y_0)，这个角度数可如下简单地计算：

$$A=\frac{(x_p-x_0)}{\sqrt{(x_p-x_0)^2+(y_p-y_0)^2}}=\cos\theta$$

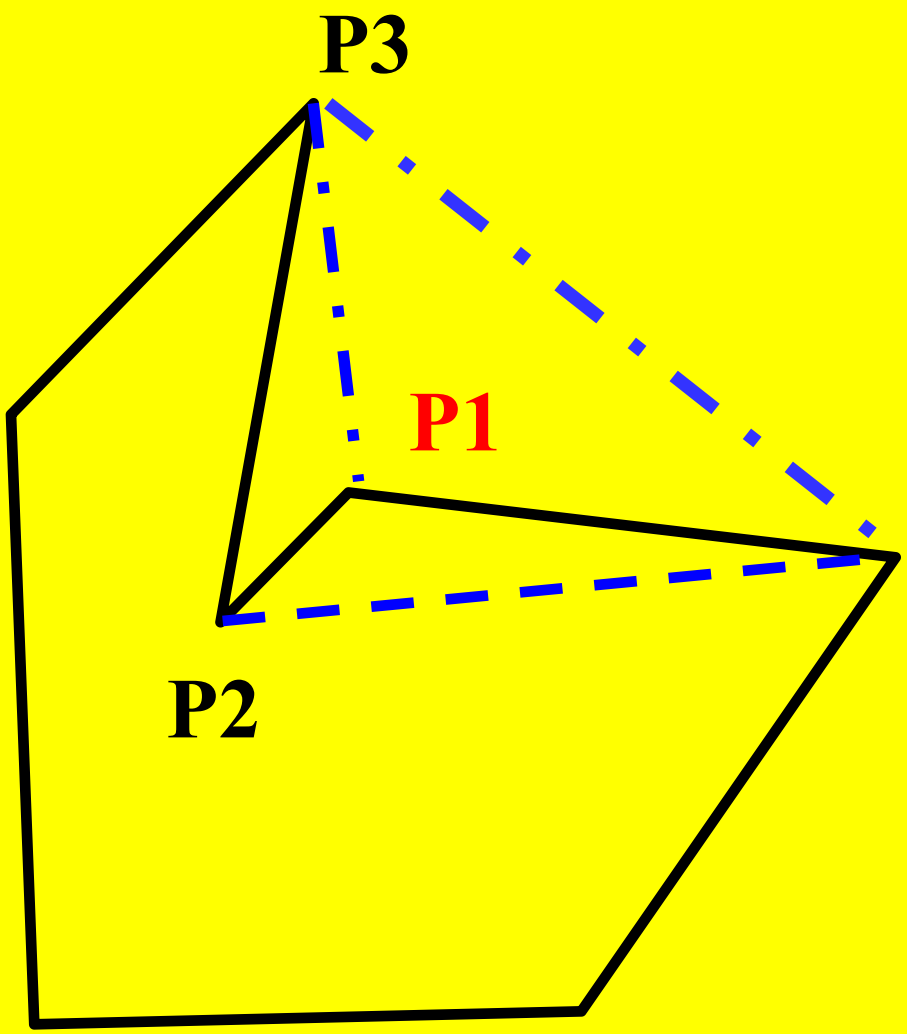
$B=y_p-y_0$

若$B\geqslant 0$ 则角度数=1-A $(0\leq\theta\leq 180^o)$

若$B<0$ 则角度数=3+A。 $(180^o\leq\theta\leq 360^o)$

cos θ 在 Ⅰ Ⅱ 象限递减，在Ⅲ Ⅳ 象限递增

算法中 v←PRED(v)的作用：例子说明

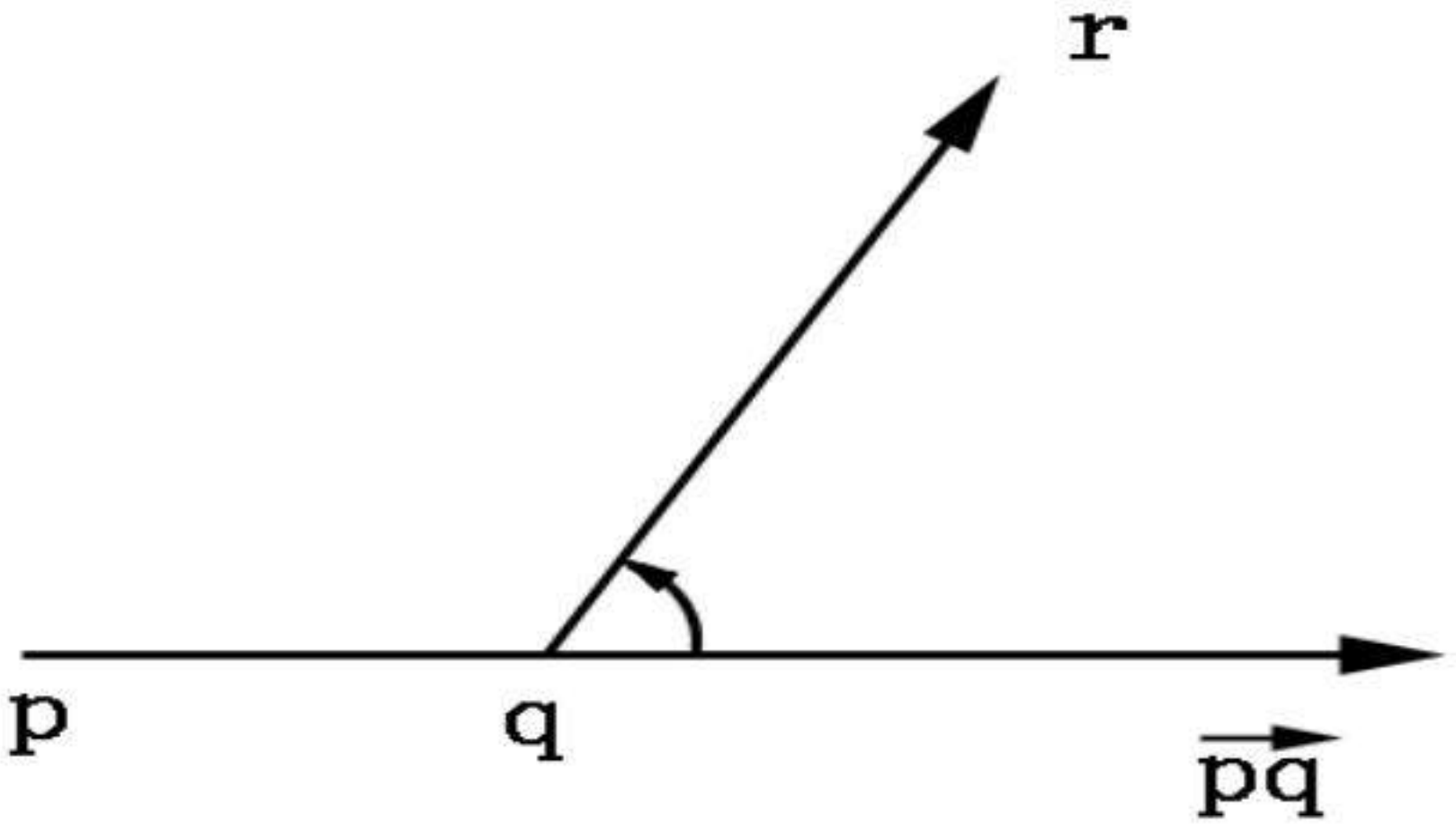


P0P1P2左转，
P1P2P3右转，
P0P1P3右转。

2. Jarvis行进算法

思想：若相继两点构成的边是凸壳的一条边则对于过该边的直线，所有点集中的点在该直线同侧。因此若找到pq是壳上一边，则以q为端点的下一条壳边qr可以如下求出：计算点集中其余各点相对q点发出沿向量pq向的射线的倾角，若倾角最小者对应的点是r，则qr是下一条壳边

第一条壳边的确定：把点集S中所有点按x，y字典式排序，取最低点。从该点引一条竖直向下的射线，在此做一个行进步就找到了第一条壳边。


r
p
q
pq

Jarvis行进算法

1.〔准备〕v_0←点集S中按x,y字典次序最小的点;

d←竖直向下的一个方向向量;

点v_0送入收集凸壳顶点的队列Q中;

S_1←S-{ v_0};

u←v_0

2.〔一步行进〕v_1←Wrapping(u,d,S_1);

S1中各点相对于自u发出方向为d的射线，计算倾角，取倾角最小的点，若倾角相同，取与u距离最远的点，

Wrapping(u,d,S_1);返回下一个壳顶点

3.〔准备下次〕if $v_1 \neq v_0$, v_1送入Q中;

$S_1 = S-\{u,v_1\}$;

d←从u到v_1的一个方向向量;

u←v_1

返步2。

4.〔结束〕Q中存入所有的壳顶点, 算法结束.

练习：

1。给定平面上的n个点，找一个算法，使之成为简单多边形

2。给定一个多边形，如何判断是否为凸多边形。

第四节　包含与重叠

一. 点对简单多边形的包含算法

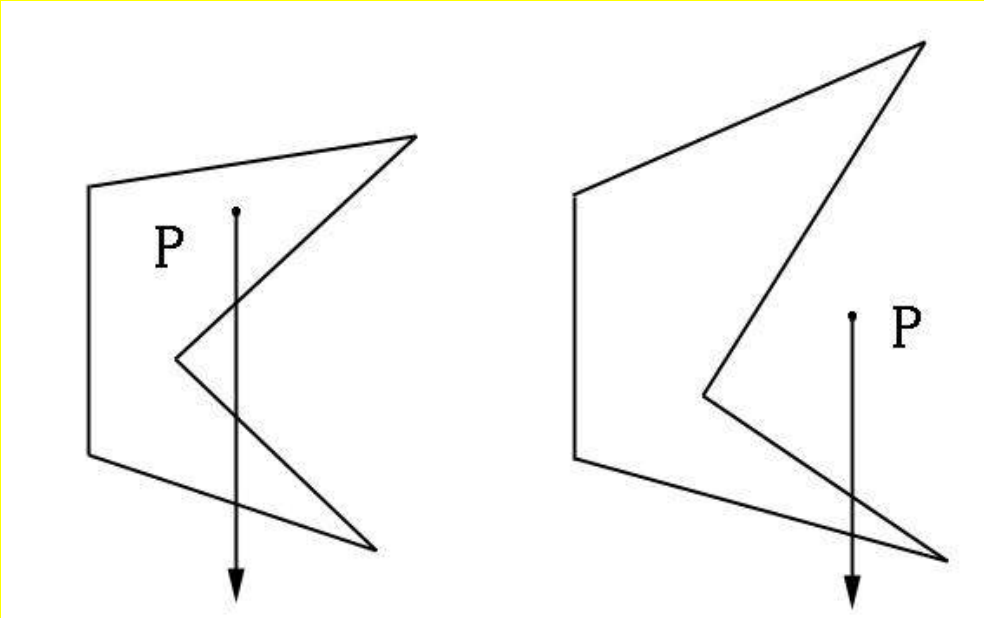
简单多边形是平面上不相邻的边不能相交的多边形

设它用顶点坐标的逆时针序列（x0, y0）, (x1, y1), …, (xn-1, yn-1)确定。

思想：由P竖直向下引射线, 计算此射线与多边形各边交点的个数。

奇数：P在多边形内部

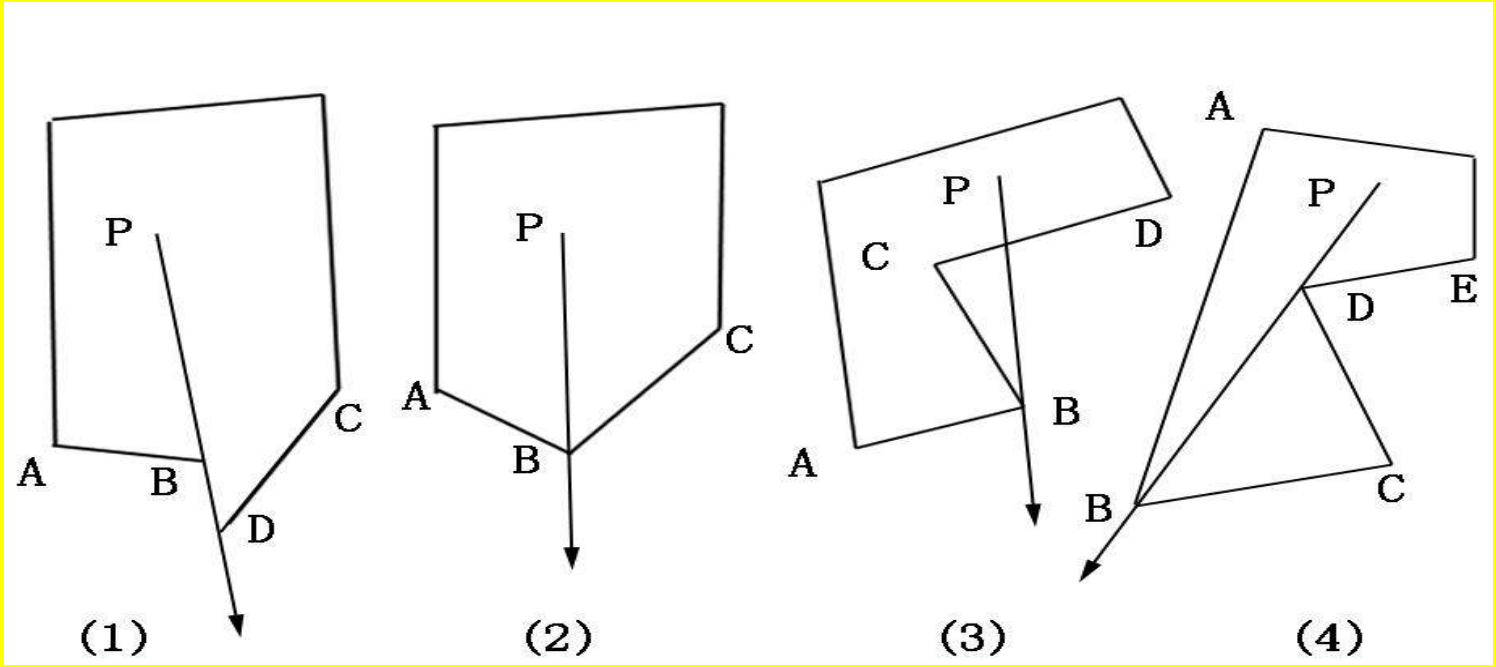
偶数：P在多边形外部



特殊情况处理：左闭右开

即:当多边形一边的两个顶点的x坐标都小于或等于点P的x坐标时,相应交点不计算在内。（左闭）

边全部在射线右侧时，交点计数（右开）



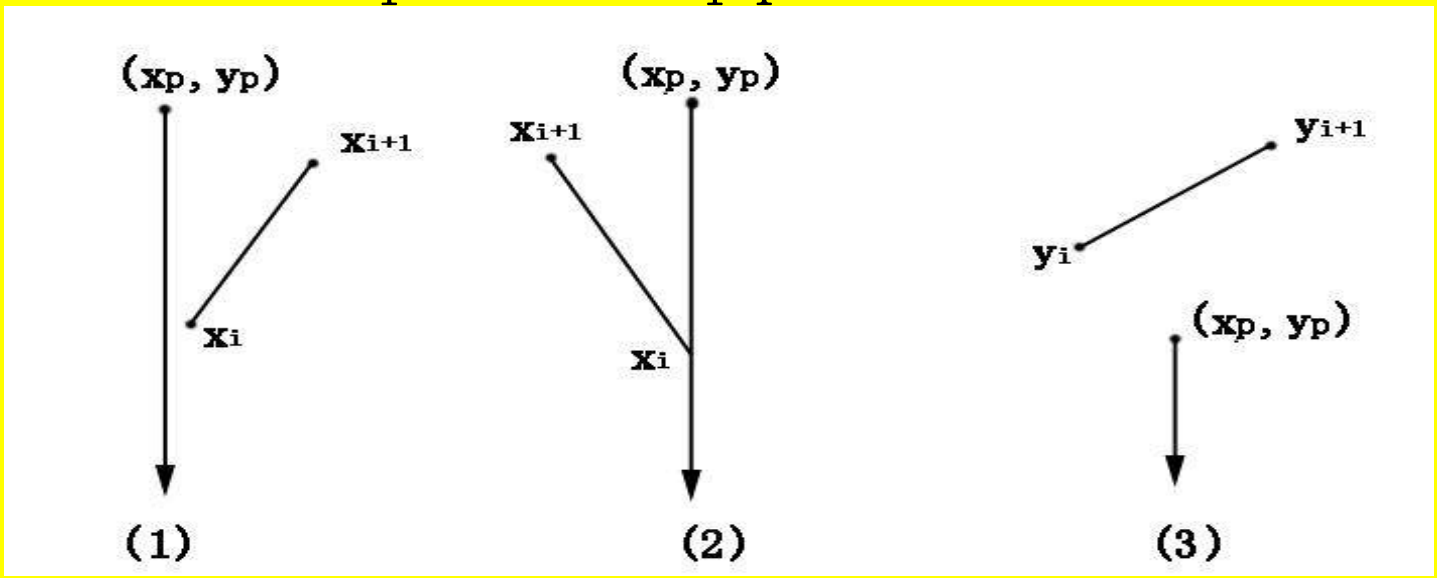
注意 ：射线均为竖直向下的

为了减少计算量，可判断不可能相交的情况

（1）$xp<x_i$并且$xp<x_{i+1}$：

（2）$xp\geqslant x_i$并且$xp\geqslant x_{i+1}$；

（3）$yp<y_i$并且$yp<y_{i+1}$；



算法中用m代表奇偶性：初始 m=-1， $m(-1)^c$

C为偶数　　m　　负　　点在多边形外部

C为奇数　　m　　正　　点在多边形内部

简单多边形包含性检验的算法：

1. 〔准备〕xn←x0, yn←y0, m←-1, i←0;
2. 〔排除必不相交情形〕若下列条件有一个成立, 则到4。
   2.1 $xp<x_i$并且$xp<x_{i+1}$:
   2.2 $xp\geqslant x_i$并且$xp\geqslant x_{i+1}$;
   2.3 $yp<y_i$并且$yp<y_{i+1}$;
3. 〔计算交点〕$y=y_i+(xp-x_i)(y_{i+1}-yi)/(x_{i+1}-x_i)$, 分二种情形:
   (1)若y=yp, 则点P在多边形边界上, 算法结束;
   (2)若y<yp, 则m←(-1)•m;
4. 〔结束判断〕i←i+1, 若i<n″, 则返回到2, 否则算法结束, 此时若m=-1则点P在多边形外部, m=1则在内部。

二.凸多边形的包含算法

如何判断坐标为(xp,yp)的点P在直线的哪一侧？

设直线的端点为(x1,y1)和(x2,y2),直线方向是由(x1,y1)到(x2,y2)的方向。

直线的方程记为Ax+By+C=0,则有:

A=y2-y1,B=x1-x2,C=x2y1-x1y2

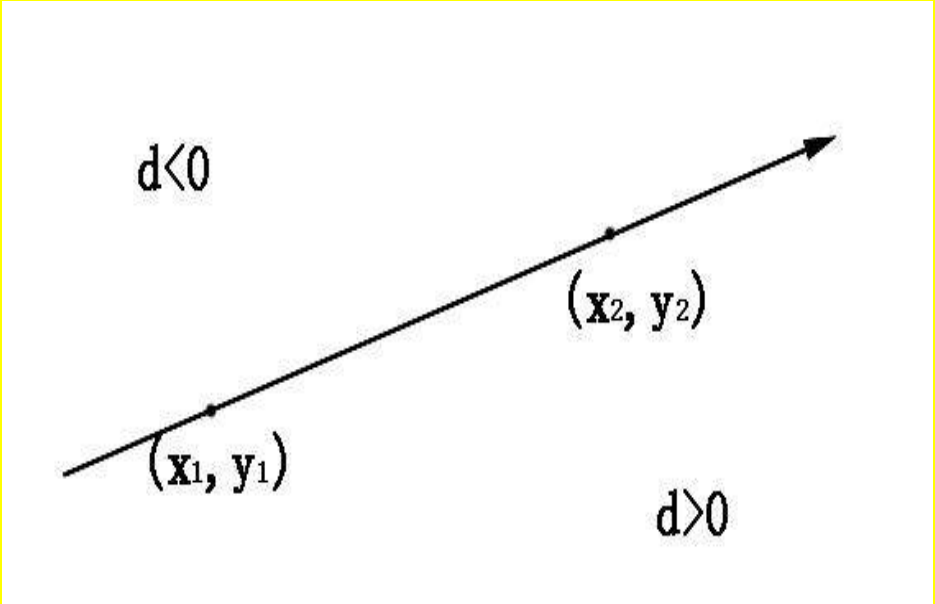
计算D:

D=Axp+Byp+C

若D<0,则点在直线左侧;

若D>0,则点在直线右侧;

若D=0,则点在直线上。



“折半查找”

设算法的输入是一个凸多边形的顶点逆时针序列P0,P1,…,Pn-1

询问点P,可有包含性检验算法:

1.〔准备〕i←1, j←n-1;

2.〔查找是否结束〕若j-i=1则到4, 否则继续;

3.〔折半查找〕k←[(i+j)/2], 检查询问点相对直线P0Pk的位置关系, 分三种情况:

3.1 在直线上, 若点在线段PoPk上或内部, 则点在原凸多边形内部,, 若点在线段PoPk延长线上, 则在原凸多边形外;输出回答后算法结束;

3.2 在左侧, i←k返回步2

3.3 在右侧, j←k返回步2

4.〔最后检查〕检查询问点P对$\triangle P_0P_iP_j$的包含性, 若在内则也在原凸多边形内部, 若在外则也在原凸多边形外部, 输出回答后算法结束.

注：对点在三角形内的判断，三条边分别判断

三. 凸多边形重叠计算

约定凸多边形指它的**边界和内部**, 凸多边形仍用顶点坐标的**逆时针方向**序列确定。

设给出的两个凸多边形P和Q的顶点序列分别是$P_1, P_2, \ldots, P_L$和$Q_1, Q_2, \ldots, Q_m$。

假设P的边界上不包含Q的项点, Q的边界也不包含P的顶点。

1. 如何**有次序**地求出各边的所有**交点**,
2. 如何**排列**求出**交点**和原凸多边形的**顶点**, 形成交得凸多边形的顶点序列。

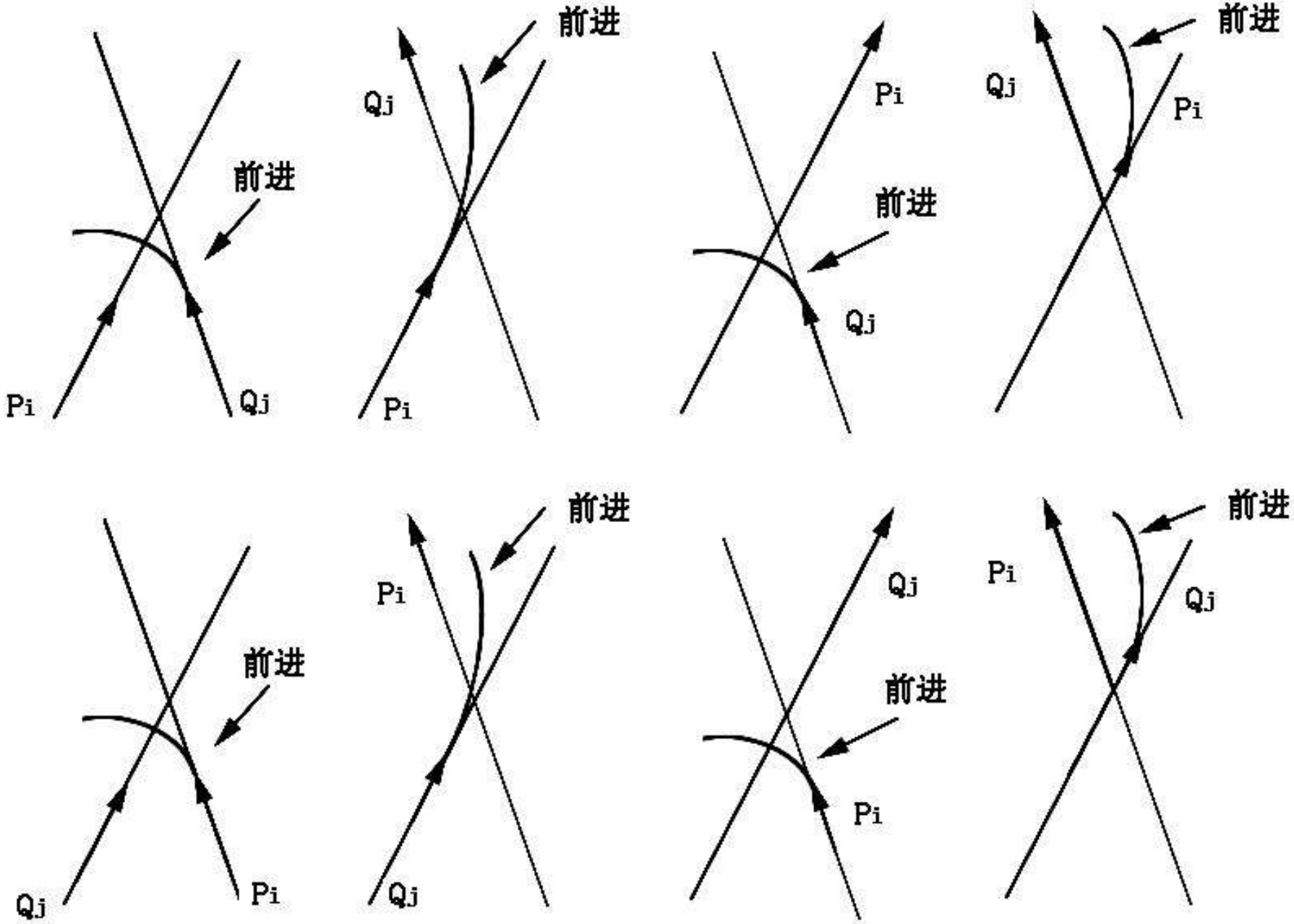
1. 为了有次序地求出交点，可以在两个多边形边上交替地前进，

这里规定了$P_0=P_L$，$Q_0=Q_m$。

设已经算得$P_{i-1}P_i$和$Q_{j-1}Q_j$的交点，依可能性判定接下去计算的是P_iP_{i+1}与$Q_{j-1}Q_j$的交点还是$P_{i-1}P_i$和Q_jQ_{j+1}的交点

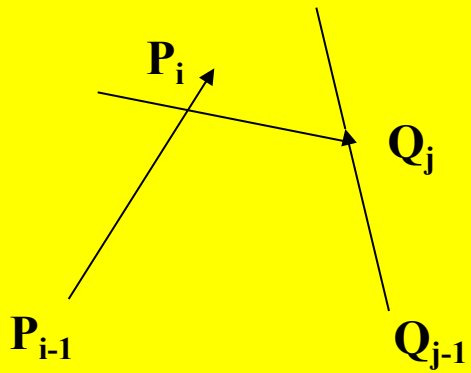
此外还需考虑P_i 和 Q_j之间的夹角是否大于180度

初始从对边P_0P_1与Q_0Q_1的求交开始，注意所有求交是线段的求交。


前进
Pi
Qj
前进
Qj
Pi
Pi
前进
Qj
Qj
Pi
前进
前进
Qj
Pi
前进
Pi
Qj
Qj
前进
Pi
Qj
前进
Pi

情形 a：P_i 在 $Q_{j-1}Q_j$ 左

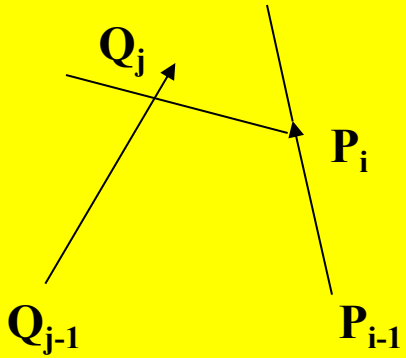
Q_j 在 $P_{i-1}P_i$ 右



Q 前进

情形 e：P_i 在 $Q_{j-1}Q_j$ 右

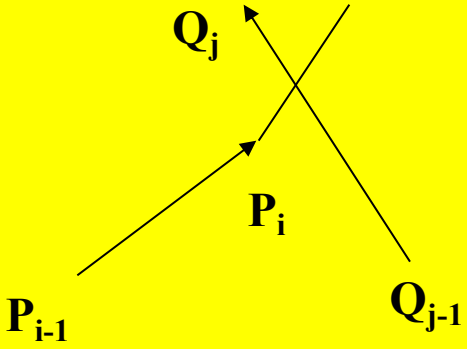
Q_j 在 $P_{i-1}P_i$ 左



P 前进

情形 b：P_i 在 $Q_{j-1}Q_j$ 左

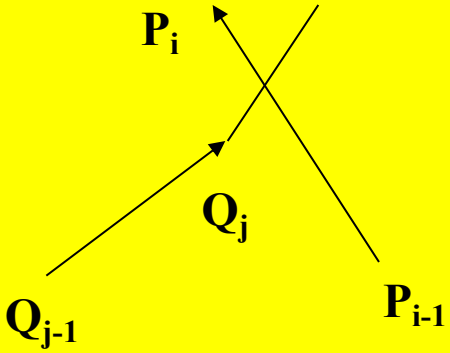
Q_j 在 $P_{i-1}P_i$ 左



约定 P 前进

情形 f：P_i 在 $Q_{j-1}Q_j$ 左

Q_j 在 $P_{i-1}P_i$ 左

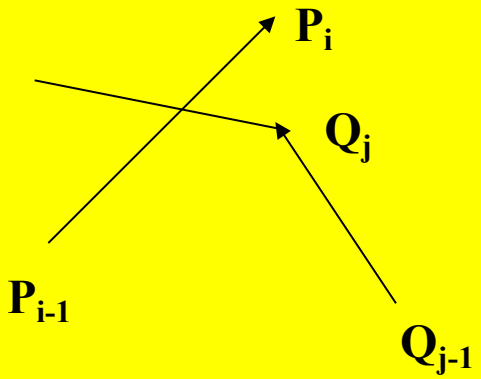


约定 Q 前进

P_i 和 Q_j 前方都有可能有交点，选哪个前进是任意定义的。

情形 c：P_i在$Q_{j-1}Q_j$右

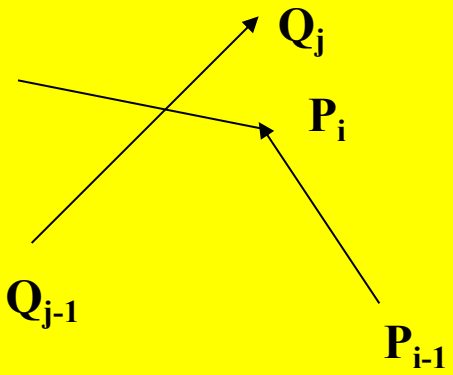
Q_j在$P_{i-1}P_i$右



Q 前进

情形 g：P_i在$Q_{j-1}Q_j$右

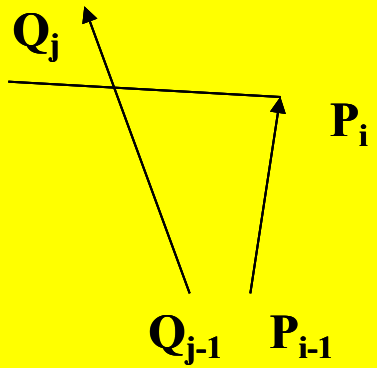
Q_j在$P_{i-1}P_i$右



P 前进

情形 d：P_i在$Q_{j-1}Q_j$右

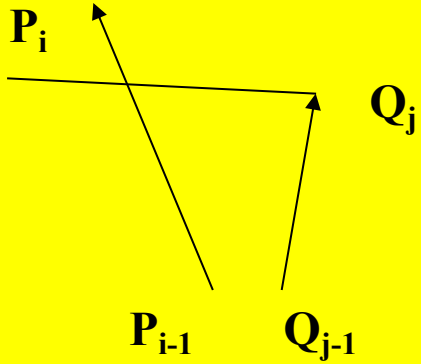
Q_j在$P_{i-1}P_i$左



P 前进

情形 h：P_i在$Q_{j-1}Q_j$左

Q_j在$P_{i-1}P_i$右



Q 前进

情形	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Pi在Qj-1Qj	左	左	右	右	右	左	右	左
Qj在Pi-1Pi	右	左	右	左	左	左	右	右
前进的多边形	Q	P	Q	P	P	Q	P	Q

区分前四种情形还是后四种情形

求矢量积$P_{i-1}P_i \times Q_{j-1}Q_j$的z分量, 若该分量大于等于0, 是前四种情形, 小于0是后四种情形。

- **Advance**

$S \leftarrow P_{i-1}P_i \times Q_{j-1}Q_j$的z分量;分两种情况处理,然后算法就结束;

1. 若$S \geqslant 0$（前四种情况）,则做

若P_i在$Q_{j-1}Q_j$左并且Q_j在$P_{i-1}P_i$左,或者P_i在$Q_{j-1}Q_j$右并且Q_j在$P_{i-1}P_i$左（P前进）,则i前进1,否则j前进1（Q前进）;

2. 若$S<0$（后四种情况）,则做

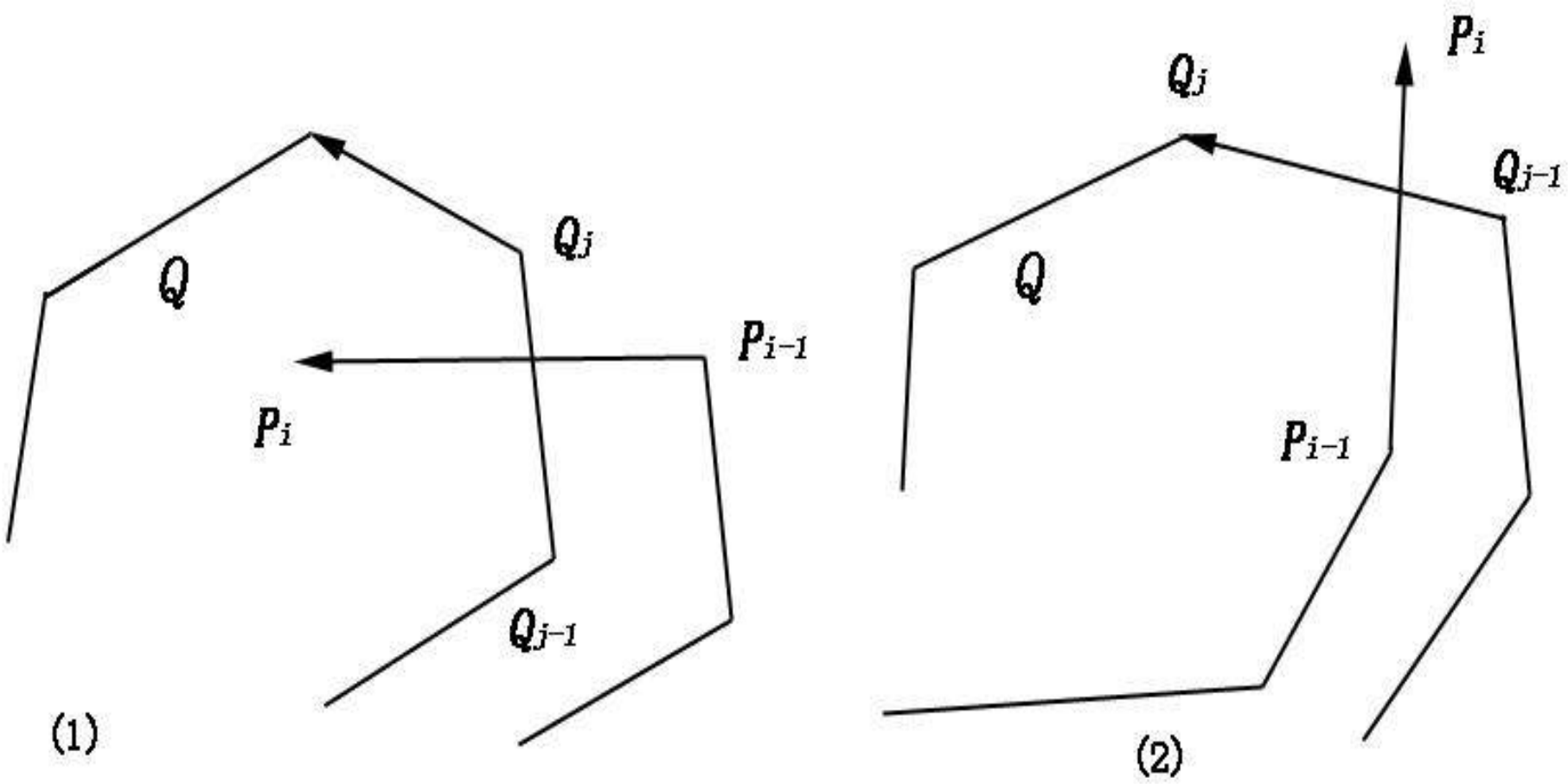
若P_i在$Q_{j-1}Q_j$右并且Q_j在$P_{i-1}P_i$左,或者P_i在$Q_{j-1}Q_j$右并且Q_j在$P_{i-1}P_i$右（P前进）,则i前进1,否则j前进1 （Q前进）;

算法中"i前进1",指若i<l,则前进1是i+1;若i=L,则前进1是1。这因为多边形P是首尾相接的。类似地"j前进1",j<m时是j+1;j=m时是1。i总在多边形P上前进,j在Q上前进。

2. 如何排列求出交点和原凸多边形的顶点，形成交得凸多边形的顶点序列？

可以在每求出一个交点时进行一次输出，求出的第一个交点可只做一下记录，如果在以后交替前进求交点的过程中再次求出与第一次求得相同的交点，就知道整个求交过程已经结束了。

求得一个不是第一个的其它任何一个交点时，为形成交得凸多边形顶点序列，要区分边$P_{i-1}P_i$是进入多边形Q，还是走出Q两种情况。



$P_{i-1}P_i$正进入多边形Q，此时应输出本次求出交点前，上次求得交点后的多边形Q上的各顶点，然后再输出本次交点，

$P_{i-1}P_i$是走出Q，这时应输出本次求出交点前，上次求得交点后的多边形P上的各顶点，再输出本次交点。

这两种情况区分,可通过检查Pi在直线Qj-1Qj的左侧还是右侧来确定。

Output

若本过程是第一次被调用,则做:

R_0←第一次求得的交点,若P_i在$Q_{j-1}Q_j$左则

t←i,否则t←j;

否则做:

若P_i在$Q_{j-1}Q_j$左,则做:

输出多边形Q上t至j-1各顶点,输出当前交点,t←i;

否则做:

输出多边形P上t至i-1各顶点,输出当前交点,t←j;

两个凸多边形求交的完整算法:

CONVEX POLYGON INTERSECTION

1.〔准备〕i←1, j←1, k←1, P_0←P_L, Q_0←Q_m;

2.〔交替前进求交〕若k≤2*(l+m)并且所求出当前交点不是第一次求得交点R_0, 则做2.1~2.3循环:

2.1 若线段$P_{i-1}P_i$与$Q_{j-1}Q_j$相交, 则调用Output;

2.2 调用Advance;

2.3 k←k+1;

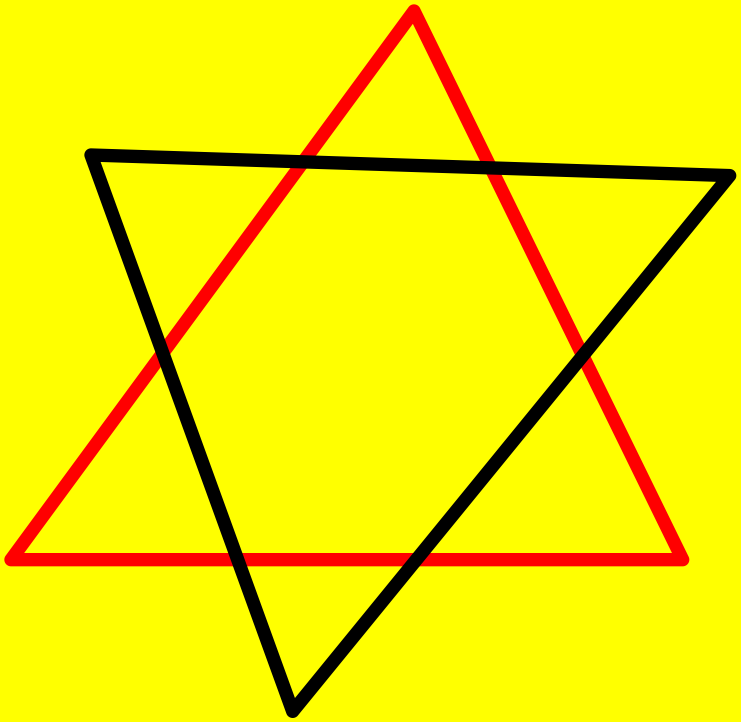
3.〔结束判断〕若在步2找到过交点，则交得凸多边形顶点序列已在调用Output过程中输出，算法结束；

否则，做如下检查：

若P_1包含于多边形Q中，则输出P包含于Q中，算法结束；

若Q_1包含于多边形P中，则输出Q包含于P中，算法结束；

若上述两个检查都不成功，输出交为空，两多边形分离，算法结束；



P和Q，P中的每一条边与P相交最多两个交点，故Q最多与P交2m个交点，同理，P最多与Q交2l个交点，故2（l+m)步是足够的）

第五节　简单多边形的三角剖分

简单多边形做三角剖分，是要求选出完全在内部又互不相交的一组对角线，把整个多边形划分成一些三角形。

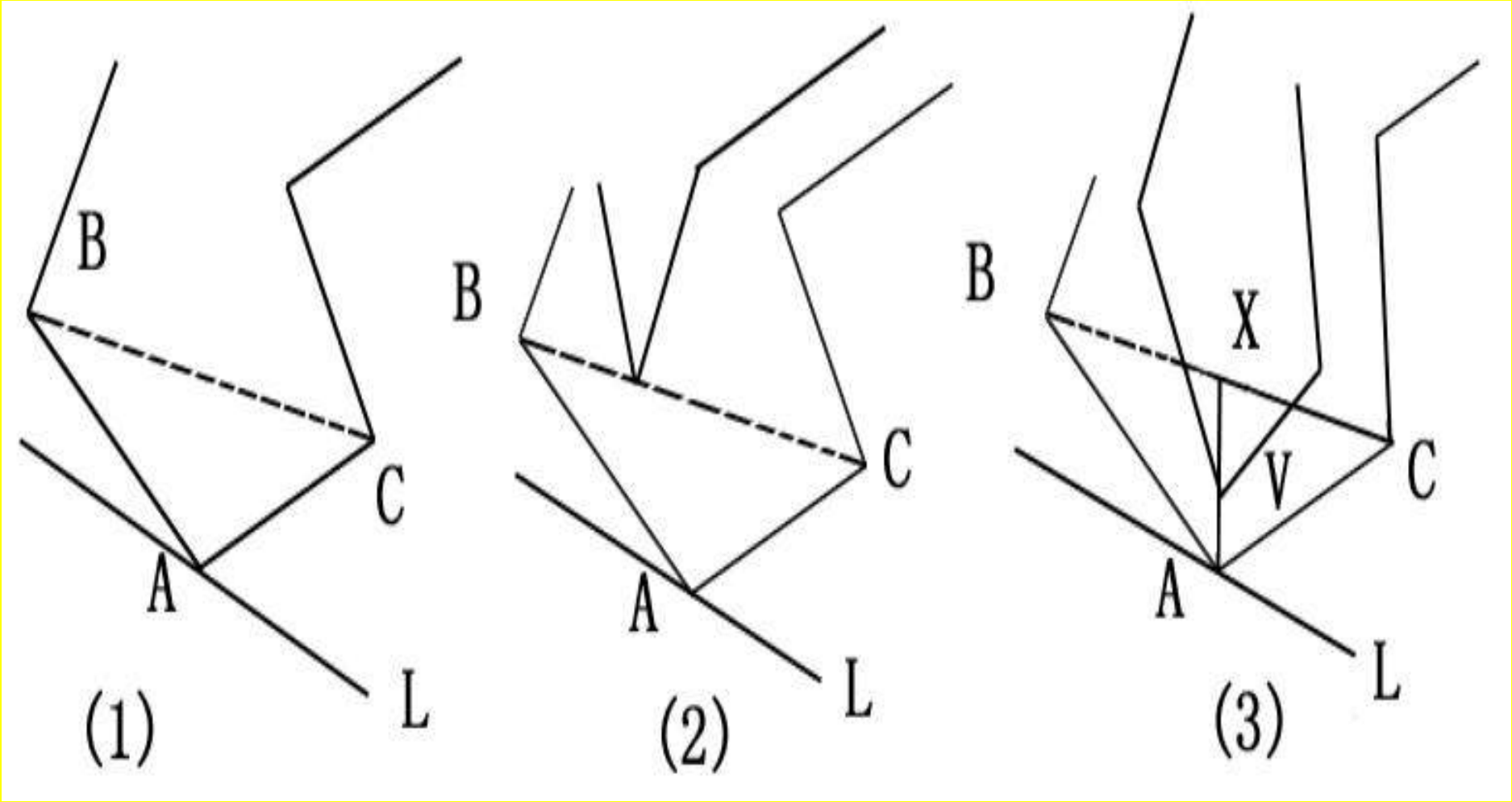
对角线是不相邻顶点间的连线，

选出的对角线的集合称为是简单多边形的三角剖分。

对任意一个简单多边形，其三角剖分不唯一。

事实1　简单多边形必有一条对角线完全在其内部。

事实2　简单多边形上必有连续的三个顶点A，B，C，使对角线AC完全在其内部。



BC BV AV

必有完全在内部的对角线

简单多边形的顶点序列为P_0, P_1, P_{n-1}，那么算法可描述如下：

SIMPLE POLYGON TRIANGULATION

1.〔准备〕$Q_0 \leftarrow P_0$；

2.〔剖分〕若n>3，则做2.1~2.2，否则转到步3：

2.1 Q1←点序列中Q_0的下一个顶点；Q2←点序列中Q_1的下一个顶点；

2.2 若Test(Q_0, Q_1, Q_2)为真，则做：

输出$\triangle Q_0Q_1Q_2$；

从点序列中删除顶点Q_1；

n←n-1；

goto2；

否则做$Q_0 \leftarrow Q_1$，返回步2.1；

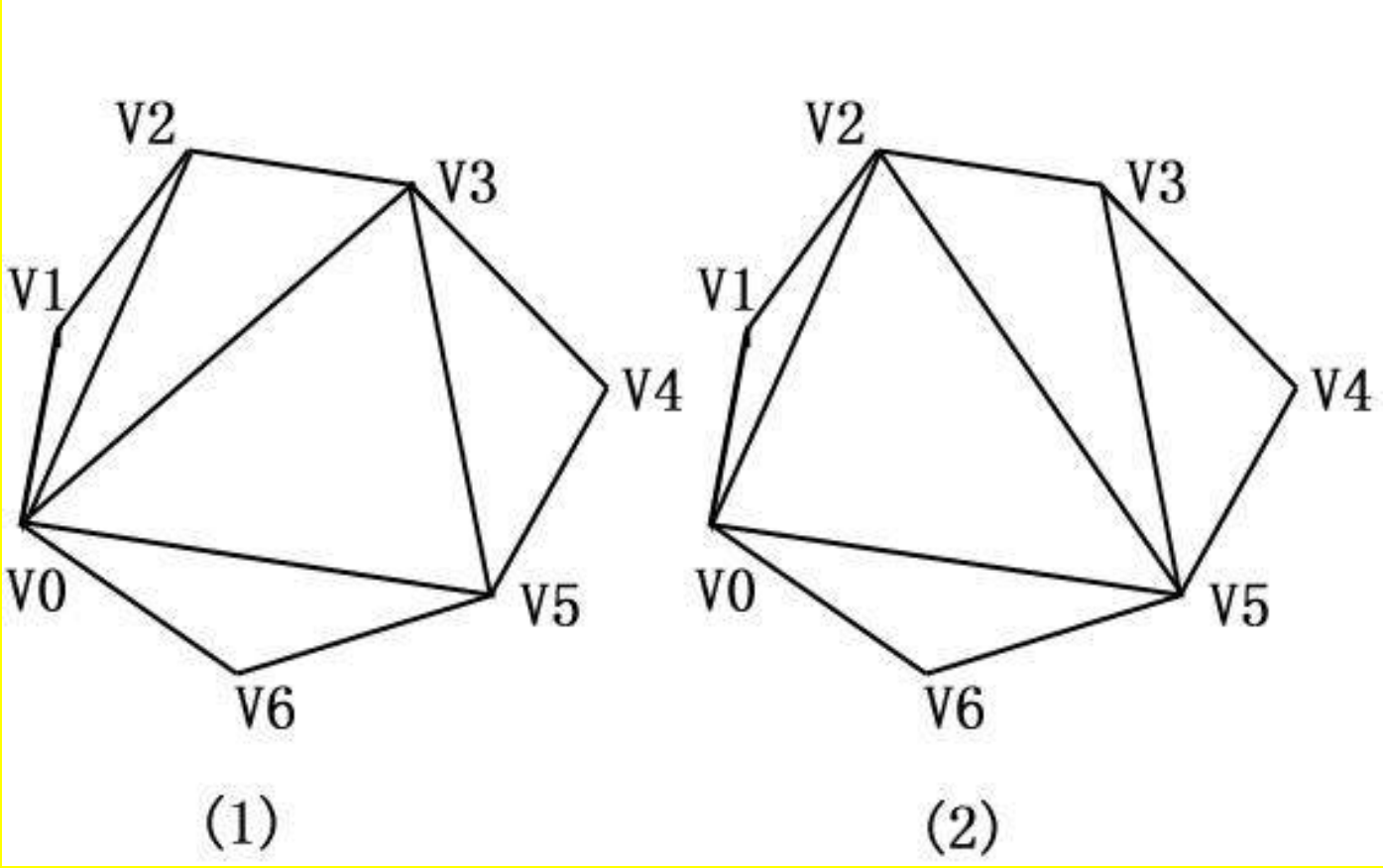
3.〔最后输出〕输出点序列中剩下三点为最后一个三角形，然后算法结束。

函数Test分两步实现，第一步检查Q_0, Q_1, Q_2是否是一个在Q_1的左转，若不然，是右转，则Q_0Q_2在多边形外部而可以回答假而结束。第二步检查可对原多边形中除去Q_0, Q_1, Q_2这三点的其它点，对每一点都考查它对三角形的包含性，若有一点被包含则就可以回答假而结束，只有其它点都在三角形外部时才能回答真而结束。

最小权三角剖分或**最小三角剖分**

如果一个三角剖分中选取的对角线的总长度最小。

任意的**凸**多边形最小三角剖分（简化问题）



事实3　在n边形(n≥3)的任意一种三角剖分中，每一对相邻顶点中至少有一个顶点是某条对角线的端点。

事实4　如果V_iV_j是三角剖分的一条对角线，则一定存在某顶点V_k，使得V_iV_k和V_kV_j是多边形的边或对角线。

因为若不然，一定存在以V_iV_j为边界的某个区域没有被三角剖分。

- 顶点序列$V_0, V_1, \cdots, V_{n-1}$确定的凸n边形的最小三角剖分

  挑选两个相邻顶点比方选V_0和V_1，由事实3知道在最小三角剖分中，必有另一顶点V_k，或者使V_1V_k是对角线，或者使V_0V_K是对角线。

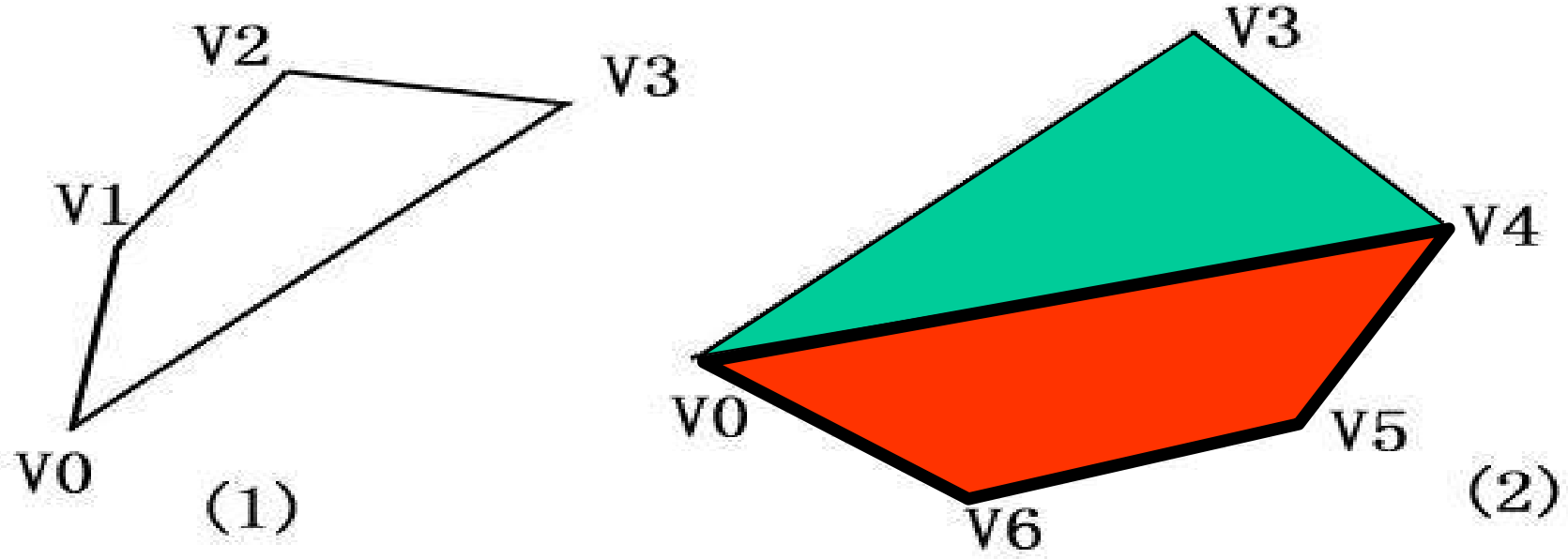
  对有n个顶点的多边形，V_k的选取方法有n−2种。

  对于每个可能的V_k用对角线V_0V_k（或V_1V_k）把原多边形剖分成两个较小的多边形，这样原问题就被分成为两个子问题。

  往下需要寻找分成的两个较小凸多边形的最小三角剖分。（递归）

选择剖分方法，使得每次剖分后所得的子问题只涉及一条原多边形的对角线。

由事实4知在最小剖分中的对角线一定与另外一点构成三角形。

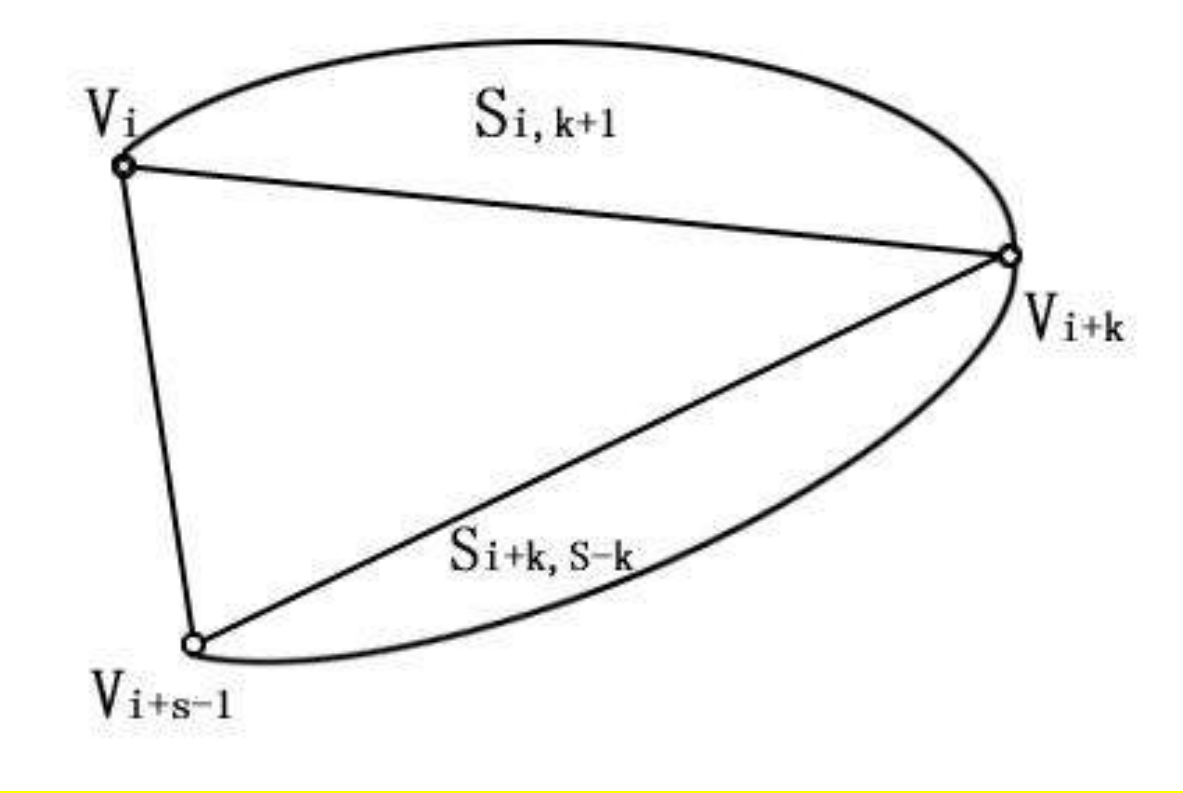


引人记号S_{is}，表示一个子多边形$V_i, V_{i+1}, \ldots, V_{i+s-1}$的最小三角剖分问题。子多边形由$V_i$开始的S个顶点按顺时针向排列围成。（图中$S_{04}$, S_{35} ）

每个子问题S_{is}中都仅涉及原多边形一条对角线。为了解S_{is}，必须考虑如下三种情况：

1. 选择顶点V_{i+S-2}，这时构成一个三角形$V_iV_{i+S-2}V_{i+S-1}$，得到一个子问题$S_{i,S-1}$
2. 选择顶点V_{i+1}，这时构成一个三角形 $V_iV_{i+1}V_{i+S-1}$，得到一个子问题$S_{i+1,S-1}$。
3. 对$2 \leqslant k \leqslant S-3$，选择$V_{i+k}$这时构成一个三角形 $V_iV_{i+k}V_{i+S-1}$，得到两个子问题$S_{i,k+1}$和$S_{i+k,S-k}$。

实际上可以把三种情况概括为一句话，即对$1 \leqslant k \leqslant S-2$，把$S_{is}$分解成两个子问题$S_{i,k+1}$和$S_{i+k,S-k}$。容易验证$k=1$和$k=S-2$时，各自有一个子问题不成其为问题，但这不影响一般的讨论。



C_{iS}记子问题S_{iS}的解

C_{iS}的公式如下：

$$C_{iS}=\min[C_{i,k+1}+C_{i+k,S-k}+D(V_iV_{i+k})+D(V_{i+k}V_{i+S-1})]$$

$$1\le k\le S-2$$

如果VpVq是对角线,则D(VpVq)是它的长度;若VpVq是原多边形的边,则D(VpVq)=0;如果S<4,则C_{iS}=0。这因为C_{iS}是最小三角剖分中引人对角线的总长度,原多边形的边不是对角线,当S<4时也不必引入对角线。

对前面说明的凸七边形,要计算的各C_{iS},有$0 \le i \le 6, 4 \le S \le 6$。可用填表的方式逐个计算。

$C_{07}=75.43$						
$C_{06}=53.54$	$C_{16}=55.22$	$C_{26}=57.58$	$C_{36}=64.49$	$C_{46}=59.78$	$C_{56}=59.78$	$C_{66}=63.62$
$C_{05}=37.54$	$C_{15}=31.81$	$C_{25}=35.49$	$C_{35}=37.74$	$C_{45}=45.50$	$C_{55}=39.98$	$C_{65}=38.09$
$C_{04}=16.16$	$C_{14}=16.16$	$C_{24}=15.65$	$C_{34}=15.65$	$C_{44}=22.69$	$C_{54}=22.69$	$C_{64}=17.89$

$$C_{is} = \min\{C_{i,k+1} + C_{i+k,s-k} + D(V_i V_{i+k}) + D(V_{i+k} V_{i+s-1})\}$$

C_{07}

$i = 0, s = 7, 1 \le k \le 5$

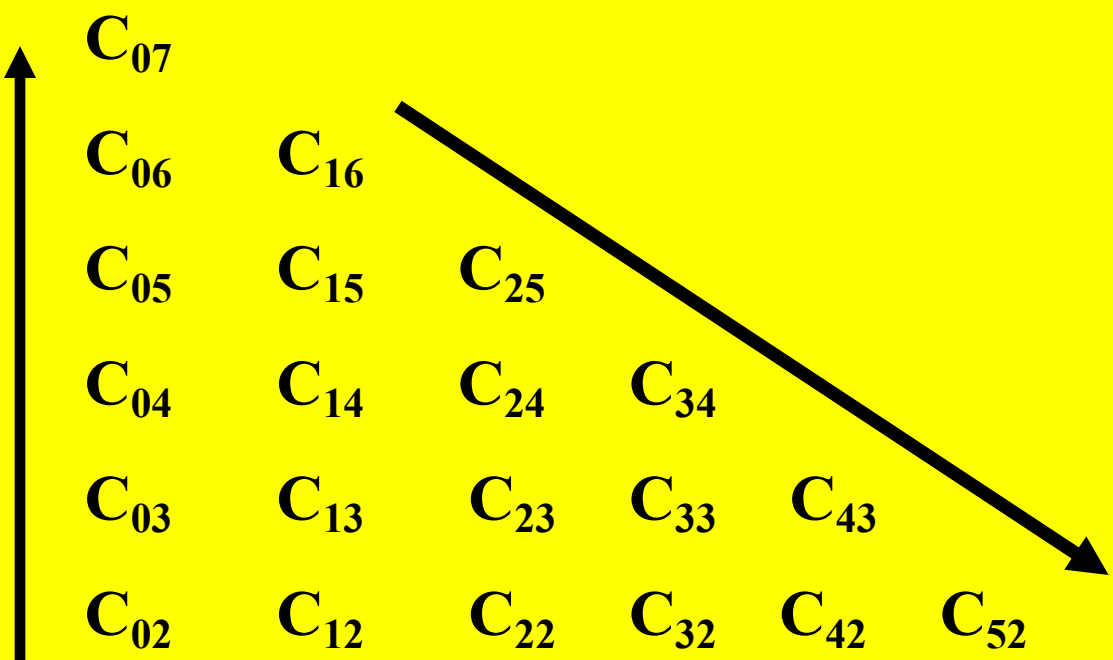
$k = 1, C_{02} + C_{16} + D(V_0 V_1) + D(V_1 V_6)$

$k = 2, C_{03} + C_{25} + D(V_0 V_2) + D(V_2 V_6)$

$k = 3, C_{04} + C_{34} + D(V_0 V_3) + D(V_3 V_6)$

$k = 4, C_{05} + C_{43} + D(V_0 V_4) + D(V_4 V_6)$

$k = 5, C_{06} + C_{52} + D(V_0 V_5) + D(V_5 V_6)$



C_{65}的计算：i=6,s=5　1<=k<=3

k=1　$C_{62}+C_{04}+D(V_6V_0)+D(V_0V_3)$

k=2　$C_{63}+C_{13}+D(V_6V_1)+D(V_1V_3)$

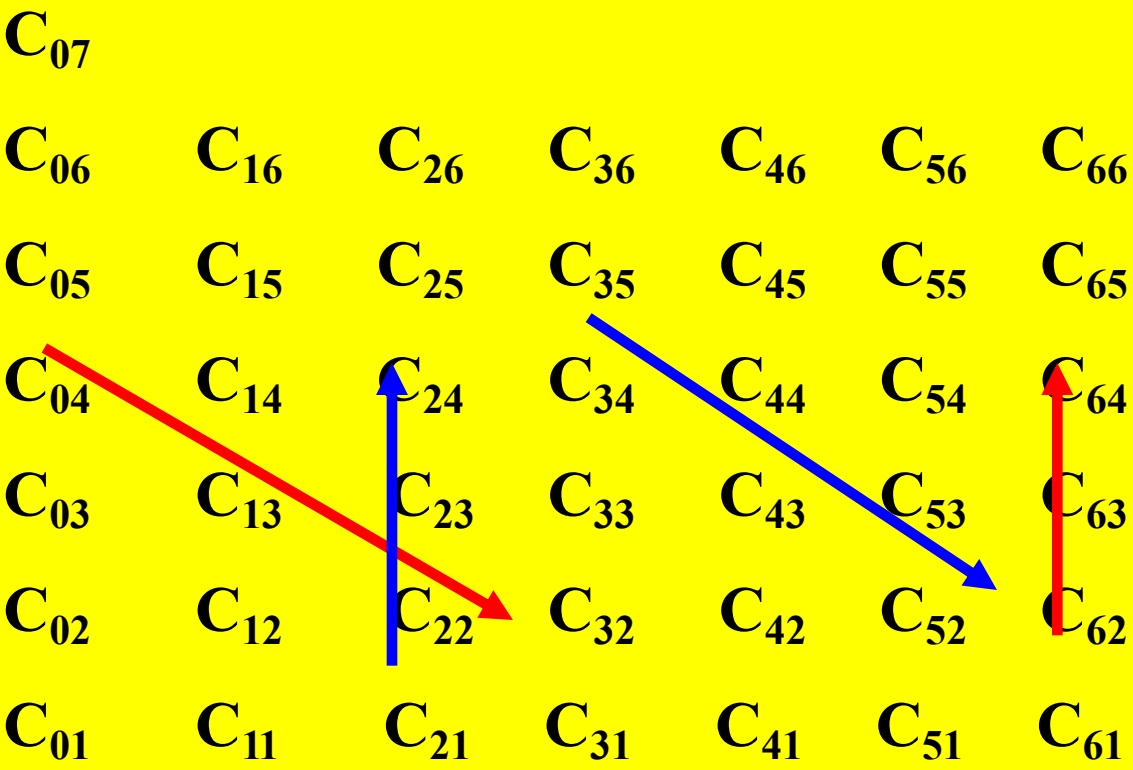
k=3　$C_{64}+C_{22}+D(V_6V_2)+D(V_2V_3)$

C_{25}的计算：i=2,s=5　1<=k<=3

k=1　$C_{22}+C_{34}+D(V_2V_3)+D(V_3V_6)$

k=2　$C_{23}+C_{43}+D(V_2V_4)+D(V_4V_6)$

k=3　$C_{24}+C_{52}+D(V_2V_5)+D(V_5V_6)$



$D(V_6V_o)=D(V_2V_3)=0,D(V_6V_2)=26.08$

$D(V_1V_3)=16.16,D(V_6V_1)=22.36,D(V_0V_3)=21.93$

C_{04},C_{64}在表中已求出,$C_{62},C_{63},C_{13},C_{22}$为0

于是算出上述三式的值分别为

38.09,38.52,43.97。所以知道$C_{65}=38.09$,可以把这个值填入表中,并知道按第一式子问题S_{65}分解出一个子问题S_{04},另一个S_{62}不成为子问题。

最小三角剖分问题的求解适合采用动态规划方法。

顶点坐标序列$V_0, V_1, \ldots, V_{n-1}$给出的凸n边形，求最小三角剖分的动态规划算法的填表过程：

1. 对$j \leqslant 3$，对i=0到n-1，令$C_{i,j}=0$；
2. 对j=4到n-1，循环做：对i=0到n-1循环做：

   2.1　对k=1到j-2

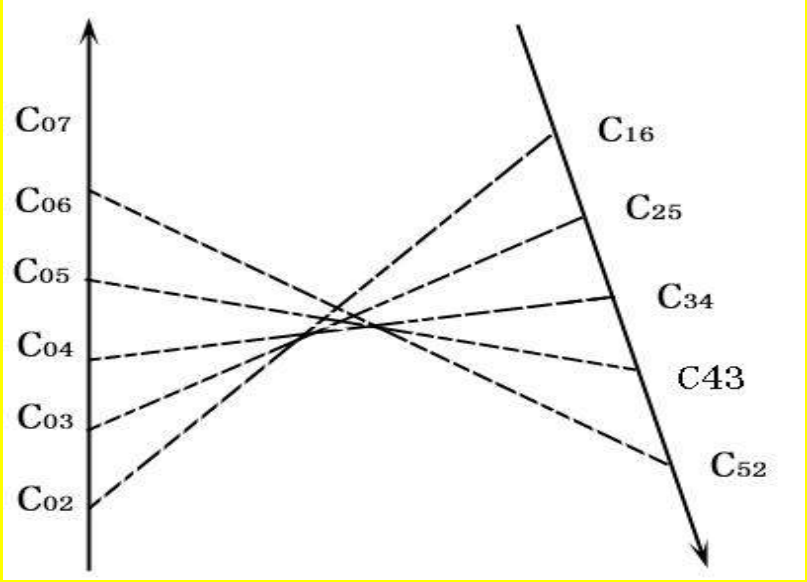
   计算$C_{i,k+1}+C_{i+k,s-k}+D(V_iV_{i+k})+D(V_{i+k}V_{i+j-1})$

   2.2 $C_{i,j}$←2.1步计算各值中的最小值；

以上过程完成填表，其计算是利用公式C_{iS} 。

1. 对k=1到n-2，

   计算$C_{0,k+1}+C_{k,n-k}+D(V_0V_k)+D(V_kV_{n-1})$；
2. C_{0n}←第1步计算各值中最小者；



C_{07}

C_{06} C_{16} C_{26} C_{36} C_{46} C_{56} C_{66}

C_{05} C_{15} C_{25} C_{35} C_{45} C_{55} C_{65}

C_{04} C_{14} C_{24} C_{34} C_{44} C_{54} C_{64}

$C_{03}=0$ $C_{13}=0$ $C_{23}=0$ $C_{33}=0$ $C_{43}=0$ $C_{53}=0$ $C_{63}=0$

$C_{02}=0$ $C_{12}=0$ $C_{22}=0$ $C_{32}=0$ $C_{42}=0$ $C_{52}=0$ $C_{62}=0$

$C_{01}=0$ $C_{11}=0$ $C_{21}=0$ $C_{31}=0$ $C_{41}=0$ $C_{51}=0$ $C_{61}=0$

第七章　消除隐藏线和隐藏面的算法

消隐　　面消隐

　　线消隐

假定：三维形体表示为多边形表面的集合

投影约定为沿着z轴正向的正交投影

消除隐藏面算法：

　　图象空间算法

　　客体空间算法

- 图象空间算法对显示设备上每一个可分辨象素进行判断，看组成物体的多个多边形表面中哪一个在该象素上可见，即要对每一象素检查所有的表面。
- 客体空间算法把注意力集中在分析要显示形体各部分之间的关系上，这种算法对每一个组成形体的表面，都要与其它各表面进行比较，以便消去不可见的面或面的不可见部分。

第一节 线面比较法消除隐藏线

- 多面体的面可见性

  对象：凸多面体

  可见面：朝向观察位置的面

  观察方向：由指向观察位置的一个方向向量k给出，面的外法向量是n，则这两个向量的夹角 α 满足$0 \leq \alpha < \pi/2$时，所考查面是可见的，否则就是不可见的

把n和k记作 $n(n_x,n_y,n_z),k=(k_x,k_y,k_z)$，

则

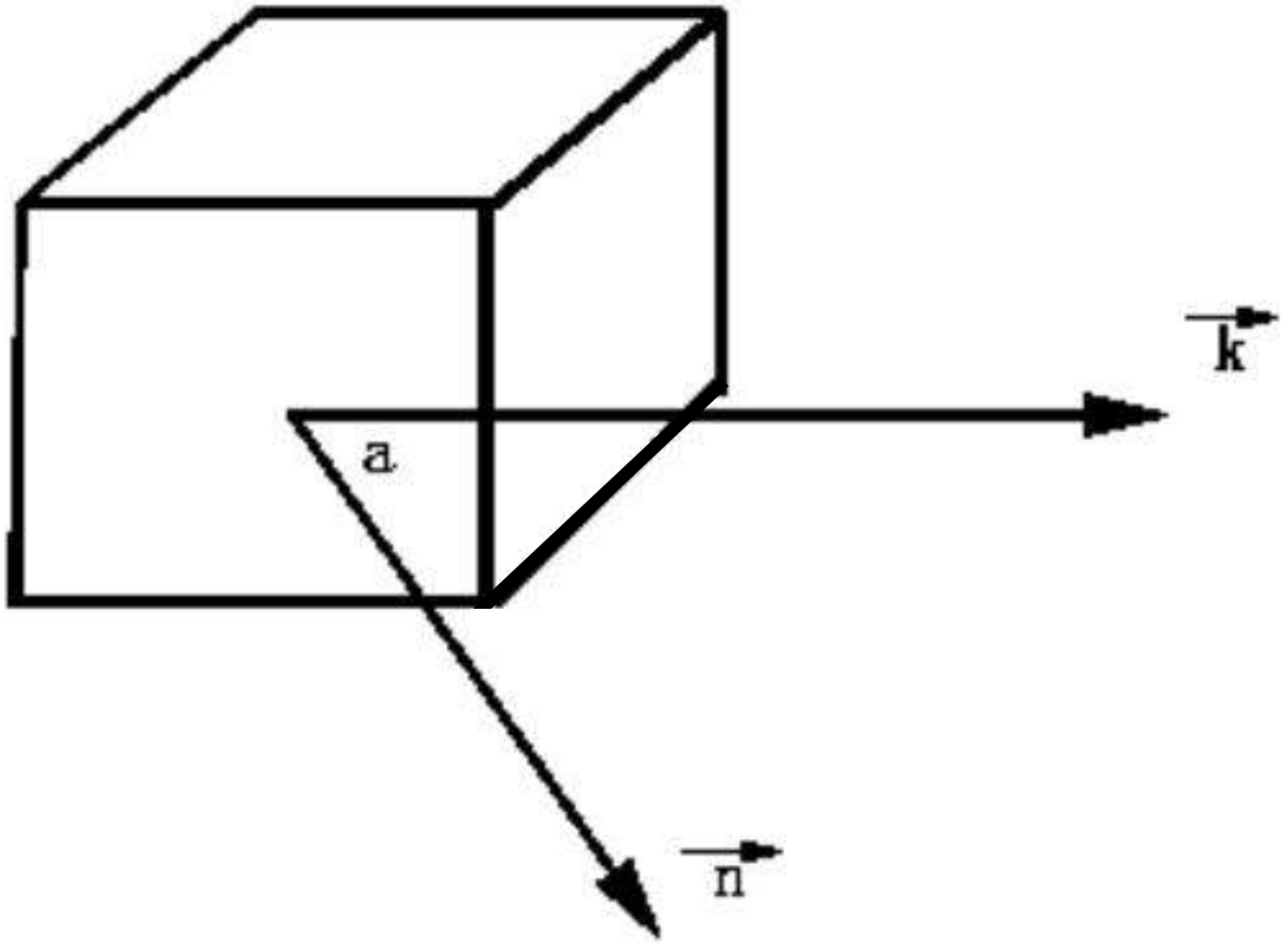
$$\cos\alpha=\frac{n\cdot k}{|n||k|}=\frac{n_xk_x+n_yk_y+n_zk_z}{\sqrt{n_x^2+n_y^2+n_z^2}\cdot\sqrt{k_x^2+k_y^2+k_z^2}}$$

$$\Delta=n_xk_x+n_yk_y+n_zk_z$$

$\Delta>0$ $0\le\alpha<\frac{\pi}{2}$ 面为可见

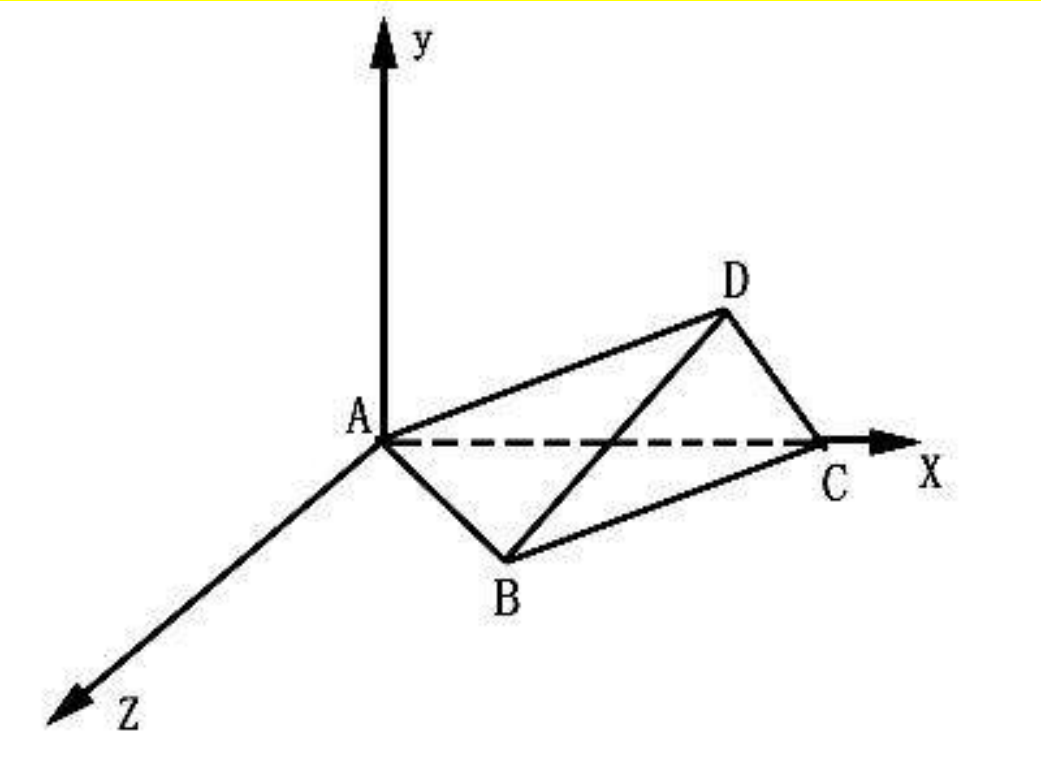
$\Delta<0$ $\frac{\pi}{2}<\alpha\le\pi$ 面不可见

$\Delta=0$ $\alpha=\frac{\pi}{2}$ 面退化为线。


k
a
n

设空间有一个四面体，顶点A，B，C，D的坐标依次是（0，0，0），（2，0，1），（4，0,0），(3,2,1)从z轴正向无穷远处观察,求各面的可见性

观察方向向量是k=(0,0,1),



三角面DAB的法向量是:

$$\begin{aligned} n &= \overrightarrow{DA} \times \overrightarrow{AB} \\ &= (-3,-2,-1) \times (2,0,1) \\ &= (-2,1,4) \end{aligned}$$

因此, $n \cdot k = 4 > 0$ ， 面DAB为可见面. 类似计算可知, 面DBC是可见面, 面ADC是不可见面, 面ACB退化为线。

单个凸多面体——可见面上的线是可见线，

多个凸多面体或非凸多面体——用上面的方法预处理，剩下可能可见面

可能可见的棱线:在可见面上或与可见面有边界

线面比较法：（观察位置位于Z轴负方向无穷远处）

0：预处理，用外法线法判断出所有可能可见面

思想：求出所有可能可见面，可能可见面上的线段是可能可见线。

每一条**可能可见线**和每一个**可能可见面**比较，

⇓ 线段

⇓ 多边形表面

1. 范围检查（xv, yv方向）

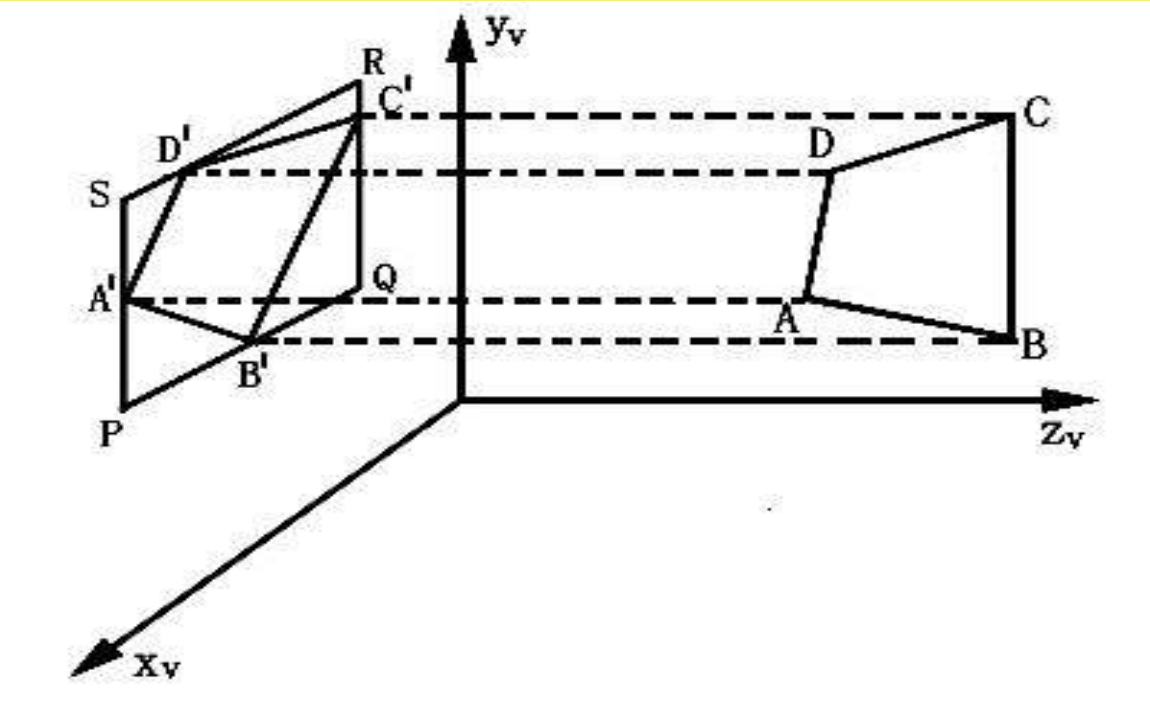
①求出线段的投影

xmin1, xmax1, ymin1, ymax1

② 求出多边形表面的投影范围

（zv平面上包含多边形投影的最小矩形）

xmin2, xmax2, ymin2, ymax2



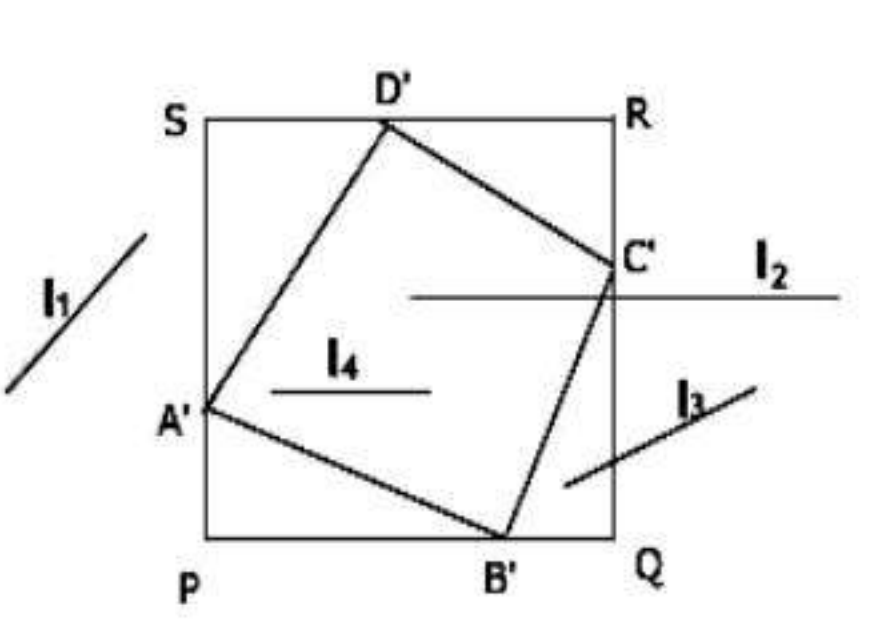
③比较投影范围：

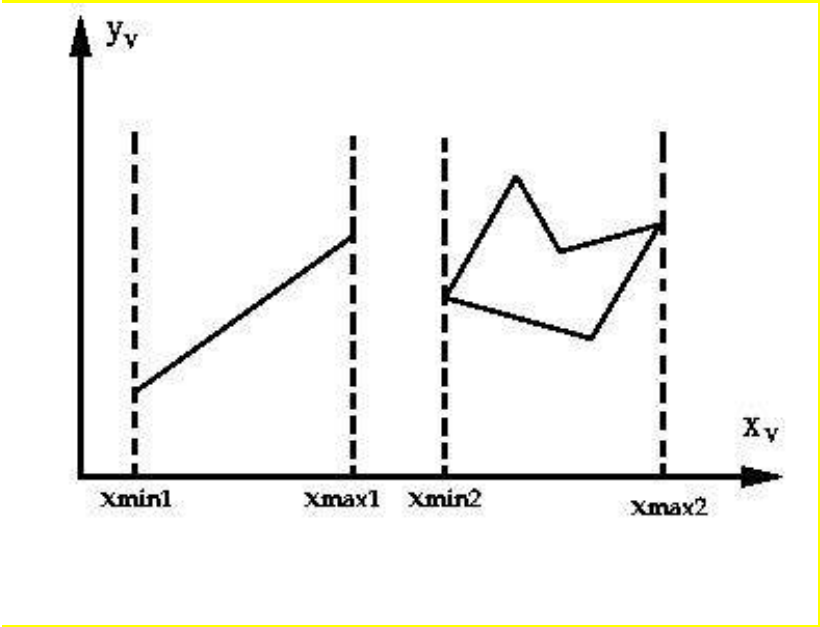
在xv方向：

若xmax1<=xmin2或xmax2<=xmin1,则无遮挡关系

Yv方向：

若ymax1<=ymin2或ymax2<=ymin1,则无遮挡关系

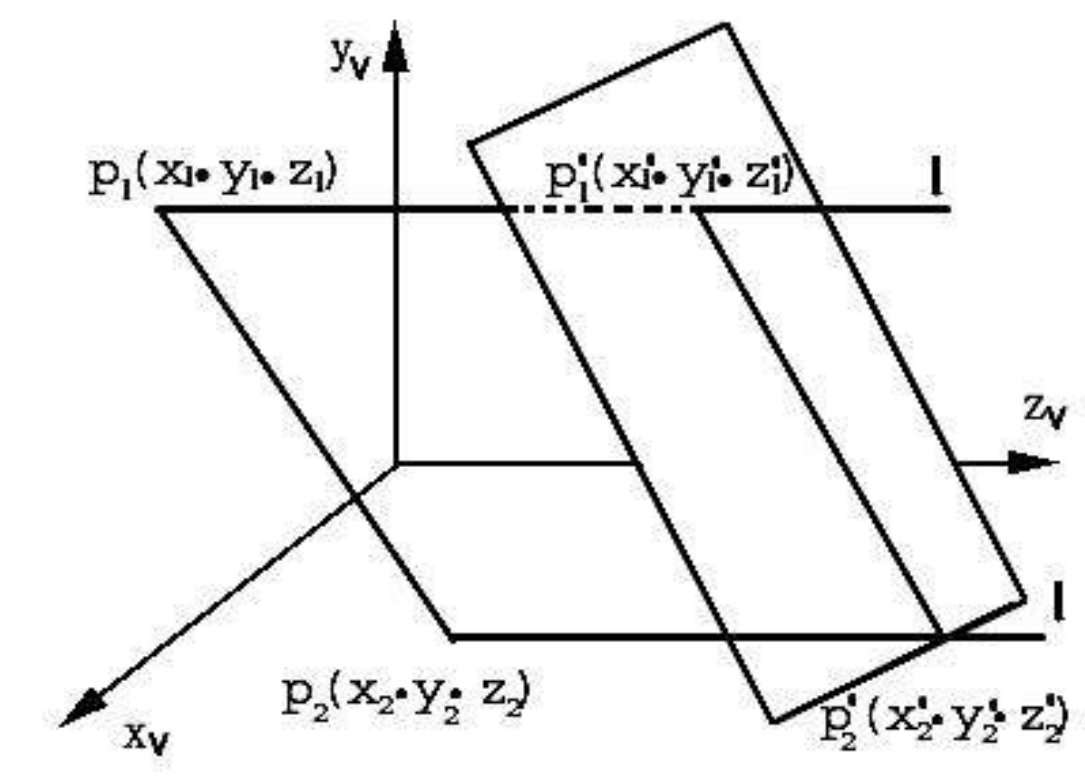




2. z_v方向粗略的深度检查。

若线段投影的最大z坐标z_{max1}小于多边形表面投影范围最小的z坐标z_{min2}，则线段完全在表面前面，根不发生遮挡现象，可以不必再往下做精确的深度检验。

3. 精确的深度检验



若 $z_1 \le z_1'$且$z_2 \le z_2'$，则线段不被遮挡

若 $z_1 \ge z_1'$且$z_2 \ge z_2'$，有可能遮挡需要进一步检查

若非以上两种情况，必然相交，求出交点，交点将原线段分成两段，分别属于上面两种情况

求出z1, z1’

设平面方程为 $Ax+By+Cz+D=0$

直线L_1的参数方程

$X=x_1$，

$Y=y_1$，

$Z=z_1+t$，代入平面方程得：$Ax_1+By_1+C(z_1+t)+D=0$

解得 $t=-\dfrac{Ax_1+By_1+Cz_1+D}{C}$

若t≥0,则z1≤ z1’，靠近视点 可见

若t＜0,则z1>z1’，远离视点

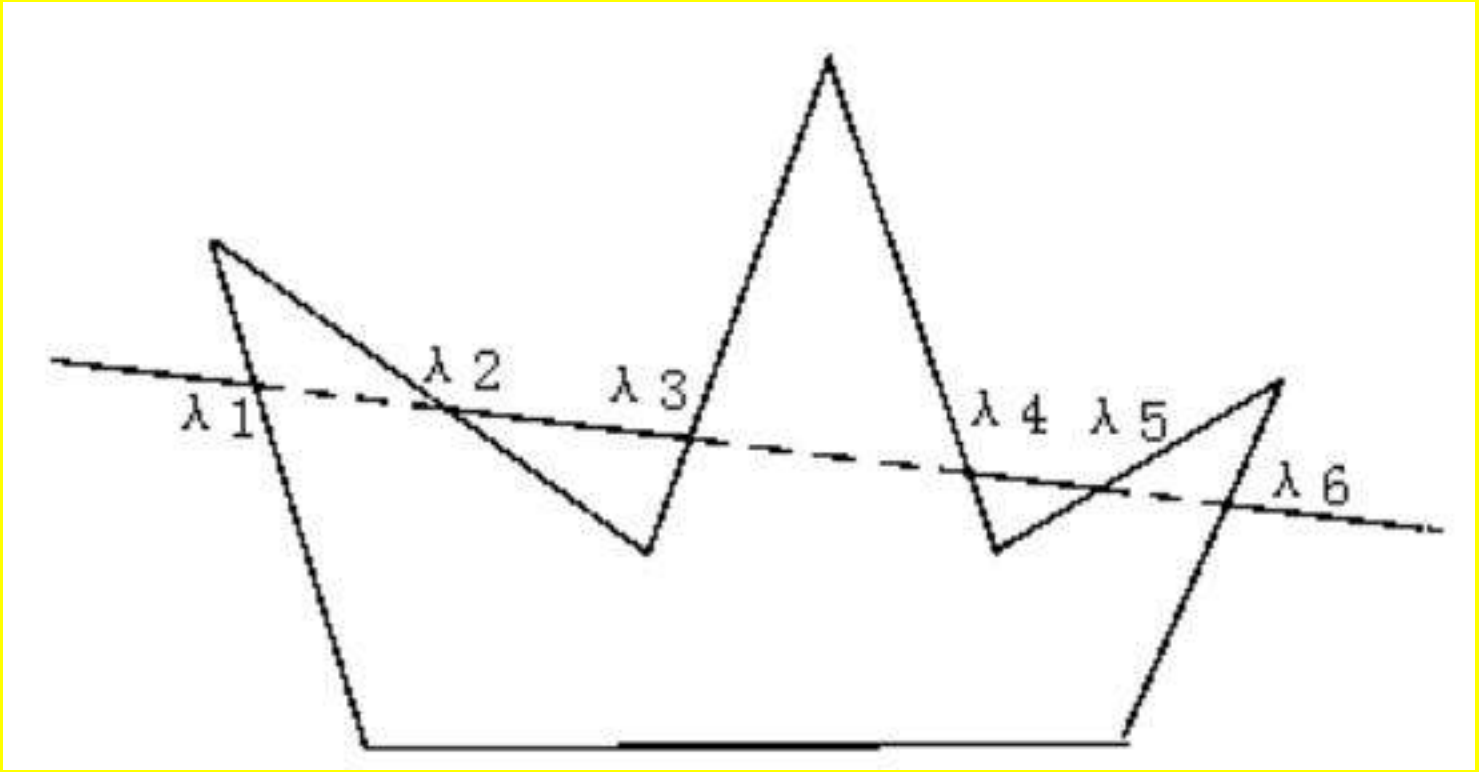
4. 进一步检查

对平面遮挡了线段的哪些部分做精确计算

求线段投影与多边形边框投影的交点（第五章 第一节）

设交点已经求出，设其对应的参数λ ，按从小到大依次排序后是λ1，λ2。。。，则这些交点将投影线段分成的各子线段的可见性应是可见，不可见交替出现。

判断子线段的可见性：（第五章 第四节）



需要检查出某一段子线段是否可见。为此可以取子线段上任意一点，若这点在多边形表面各边线的投影所形成的封闭多边形内，这子线段就不可见，否则就可见。

第三节 深度排序算法

深度排序算法：先画最远的，再画最近的（优先级算法，画家算法）

深度排序算法的主要步骤：

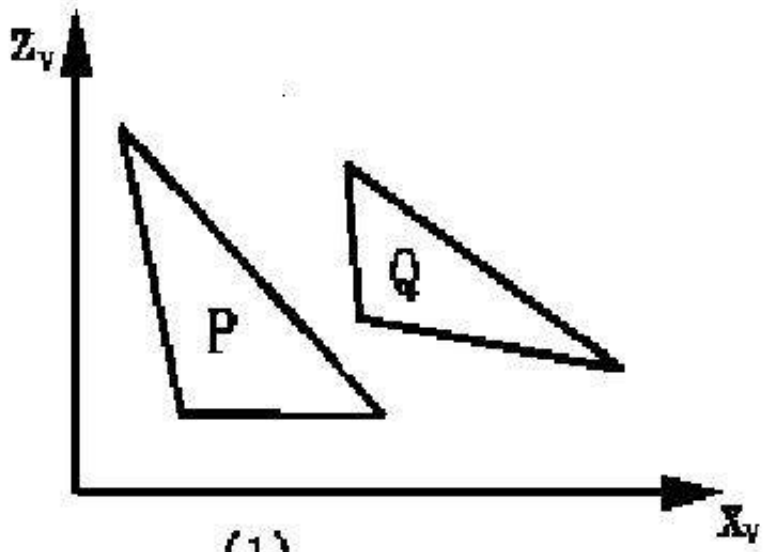
1. 把所有的多边形按顶点最大z坐标值进行排序。（客体空间）
2. 解决当多边形z范围发生交迭时出现的不明确问题。
3. 按最大z坐标值逐渐减小的次序，对每个多边形进行扫描转换。（图像空间）

不明确问题检验方法

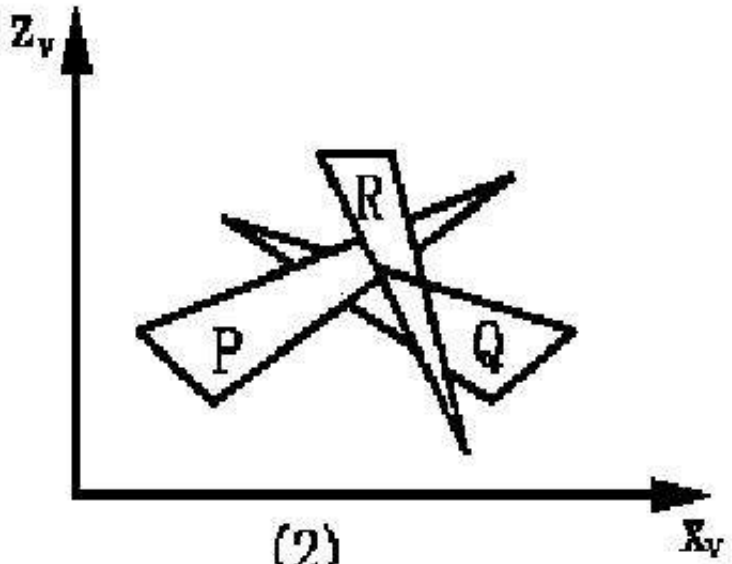
所有多边形按顶点最大z坐标值排序后得到一个排序表，设P是排在表中最后的那个多边形。

设Q是排在P前面并且z坐标范围与其发生交迭的一个多边形，对Q与P的次序关系进行检查。

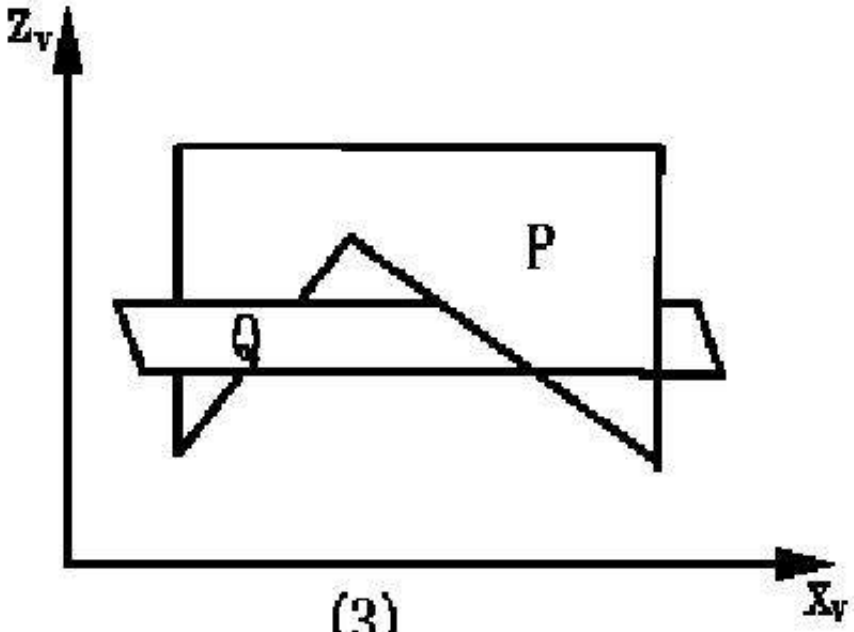
QP



(1)



(2)



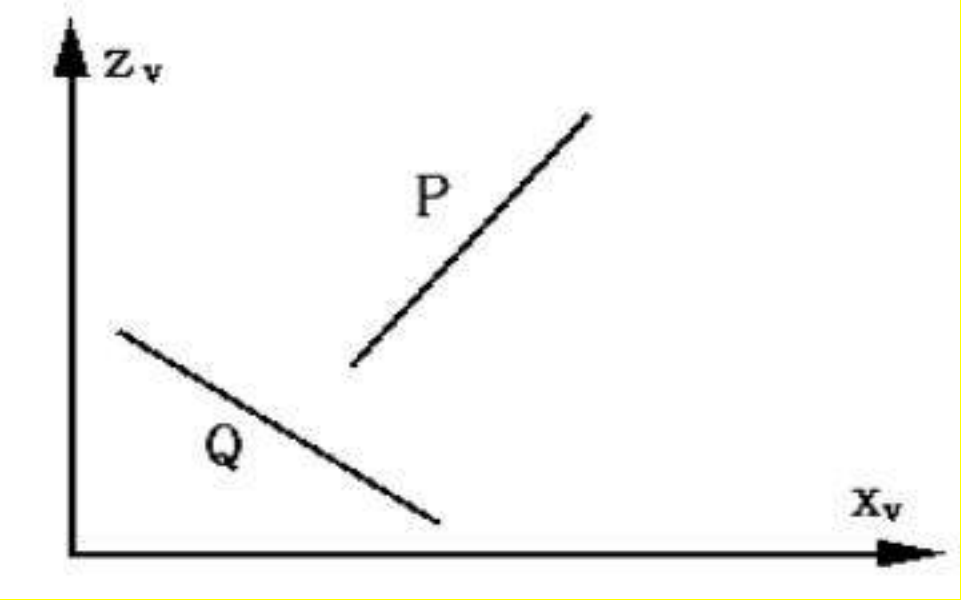
(3)

检查可以按下面列出的五个步骤进行，
每个步骤判断一种情况。

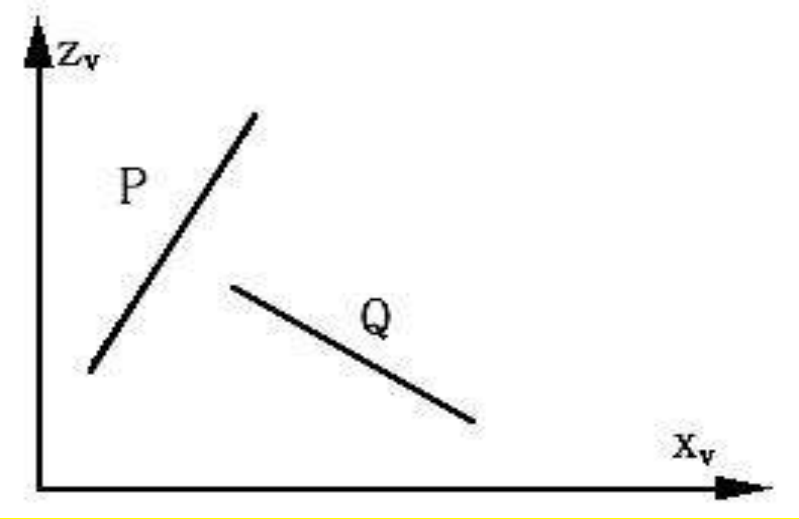
1．多边形的x坐标范围不相交迭，所以多边形不相交迭。QP

2．多边形的y坐标范围不相交迭，所以多边形不相交迭。QP

3．P整个在Q远离观察点的一侧。QP



4. Q整个在P的靠近观察点的一侧。QP



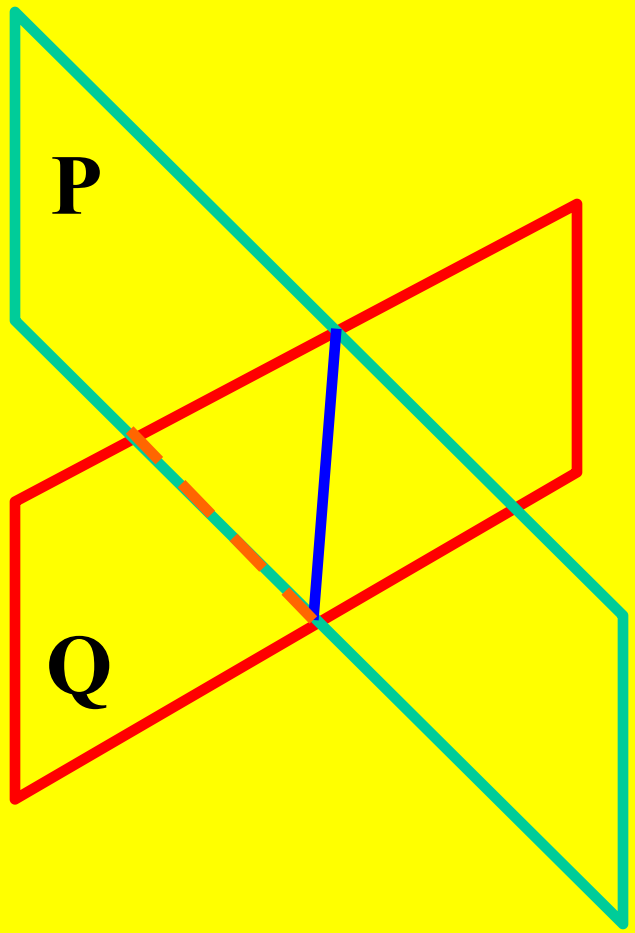
5. 多边形在z=0平面上的投影本身不相交迭。QP

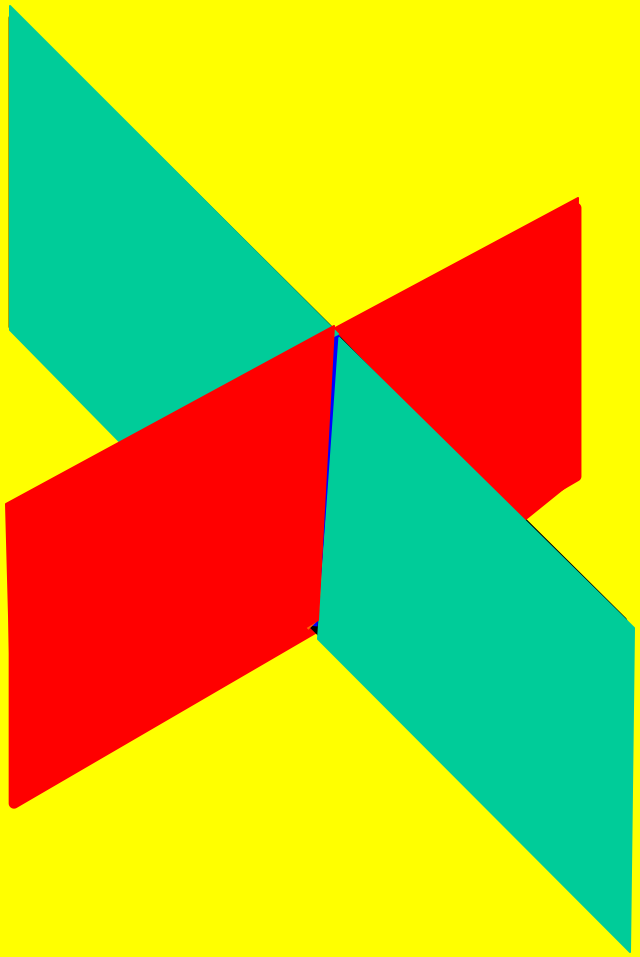
以上五步中任何一步成立，都可以说明当前排序正确

如果五步检查都为假，就假定P是遮挡了Q，交换P和Q在排序表中的位置。（QP→PQ）

如果仍做交换，算法会永远循环下去而没有结果。

为了避免循环，可以做一个限制。当做过首次五步检查后，发生某个多边形被移到排序表的末尾时，就立即加上一个标记，以后就不能再做移动。出现再次应该移动时，用一个多边形所在的平面，把另一个多边形裁剪分为两个。


P
Q



第五节 z-缓冲算法

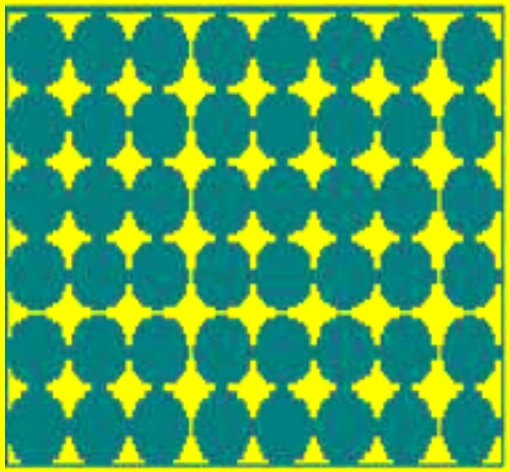
z-缓冲算法:(深度缓冲算法)是一种最简单的图象空间算法。

帧缓冲存储器:存储各点的象素值，初始化为背景值。

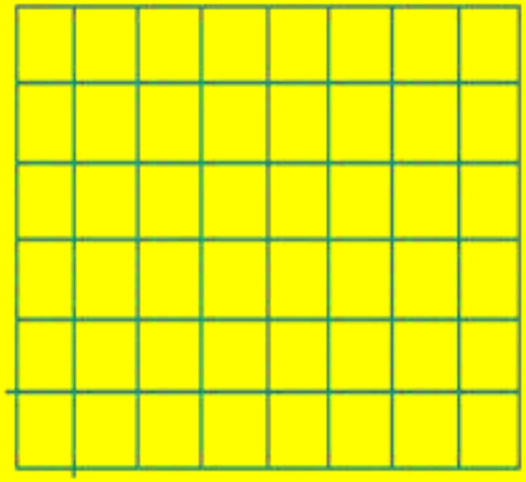
z-缓冲存储器:存储相应的z值。初始化最大z值。

对每一个多边形，不必进行深度排序算法要求的初始排序，立即就可以逐个进行扫描转换。

屏幕

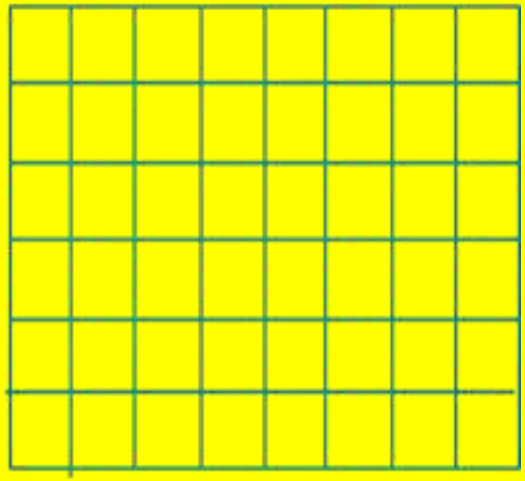


帧缓冲器



每个单元存放对应象素的颜色值

Z缓冲器



每个单元存放对应象素的深度值

扫描转换时，对每个多边形内部的任意点（x, y），实施如下步骤：

1. 计算在点（x，y）处多边形的深度值z（x，y）。

2. 如果计算所得的z（x，y）值，小于在z–缓冲存储器中点(x,y)处记录的深度值，那么就做：

（1）把值z(x，y)送入z–缓冲存储器的点(x, y)处。

（2）把多边形在深度z(x,y)处应有的象素值，送入帧缓冲存储器的点(x，y)处。

算法中深度计算，可通过多边形的顶点坐标求出所在平面的方程，然后再使用平面方程，对每个点(x,y)，解出相应的z。

对面方程 $Ax + By + Cz + D = 0$，

解出 z 是：

$$Z = \frac{-D - Ax - By}{C}$$

设在点（x, y）处的深度值是z_1：

$$\frac{-D-Ax-By}{C}=z_1$$

则在点（x+△x, y）处的深度值就是

$$\frac{-D-A(x+\Delta x)-By}{C}=\frac{-D-Ax-By}{C}-\frac{A}{C}\Delta x$$

$$=z1-\frac{A}{C}\Delta x$$

z-缓冲算法的工作流程:

帧缓冲区置成背景色;

z-缓冲区置成最大z值;

for (各个多边形)

{ 扫描转换该多边形;

for(计算多边形所覆盖的每个象素（x,y）)

{ 计算多边形在该象素的深度值Z(x,y);

if(Z(x,y)小于Z缓冲区中的(x,y)处的值)

{ 把Z(x,y)存入Z缓冲区中的(x,y)处;

把多边形在(x,y)处的亮度值存入帧缓存区的(x,y)处; }

}

}

Z-Buffer算法优点：

在象素级上以近物取代远物。形体在屏幕上的出现顺序是无关紧要的。这种取代方法实现起来远比总体排序灵活简单，有利于硬件实现。

Z-Buffer算法缺点：占用空间大，没有利用图形的相关性与连续性。 需要开一个与图象大小相等的缓存数组ZB。

改进算法:只用一个深度缓存变量zb。

```
Z-Buffer()
{ 帧缓存全置为背景色
   for(屏幕上的每个象素(i,j)) //扫描整个屏幕
{ 深度缓存变量zb置最大值MaxValue
  for(多面体上的每个多边形Pk)
  {  if(象素点(i,j)在pk的投影多边形之内)
    { 计算Pk在(i,j)处的深度值depth;
     if(depth小于zb)
     { zb = depth;
       indexp = k;}}}
  If(zb != MaxValue) 计算多边形Pindexp在交点 (i,j) 处的光照
颜色并显示
  }}
```

第六节 扫描线算法

消除隐藏面的扫描线算法：图象空间算法

第二章 第三节 多边形扫描转换算法 ＝》推广

填充算法：针对一个多边形做扫描转换

消隐算法：同时对多个多边形做扫描转换。

ET表（边表）中吊桶的内容：

1．与较小的y坐标对应的端点的x坐标xmin。
2．边的另一端点的较大的y坐标ymax。
3．x的增量Δx，它实际上是边的斜率的倒数，是从一条扫描线走到下一条扫描线时，按x方向递增的步长。
4．边所属多边形的标记。
5．指针 指向下一条边

PT表（多边形表）：记录多边形信息，一个记录对应一个多边形。

一个记录的内容：

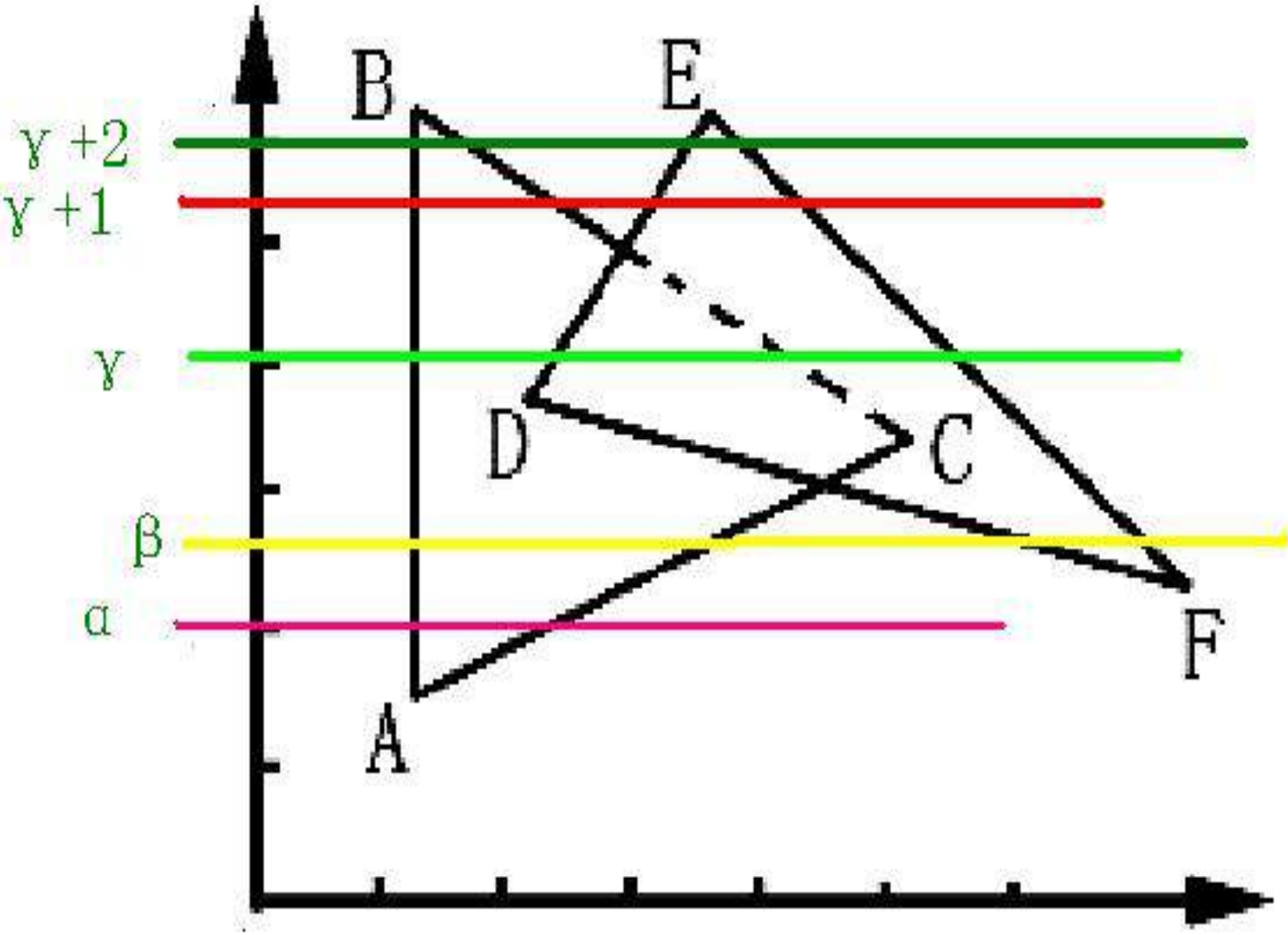
1．多边形所在平面方程的系数A,B,C,D

2．多边形的颜色或亮度值

3．进入/退出标志，标记当前扫描线对该多边形是“进入”还是“退出”

AET表（活跃边表）：存储与当前扫描线相交的各边的信息。

APT表（活跃多边形表）：与当前扫描线相交的所有多边形，按深度递增排序。


B
E
γ +2
γ +1
γ
D
C
β
α
F
A

扫描线y=α：

AET表中有AB, AC,

到AB时，$Flag_{ABC}$=1 ，AB到AC一段用ABC的颜色填充

到AC时，$Flag_{ABC}$=0,

扫描线y=β：

AET表中有四项AB, AC, FD, FEF,

到AB时，$Flag_{ABC}$=1 ，AB到AC一段用ABC的颜色填充

到AC时，$Flag_{ABC}$=0, AC到FD之间填充背景色

到FD时，$Flag_{DEF}$=1, FD到FE之间填充DEF颜色

到FE时，$Flag_{DEF}$=0

扫描线y=γ：

AET表中有四项AB, DE, CB, FE

到AB时，$Flag_{ABC}$=1 ，AB到DE之间填ABC的颜色

到DE时，$Flag_{DEF}$=1，做深度比较，求（x’，γ）的深度

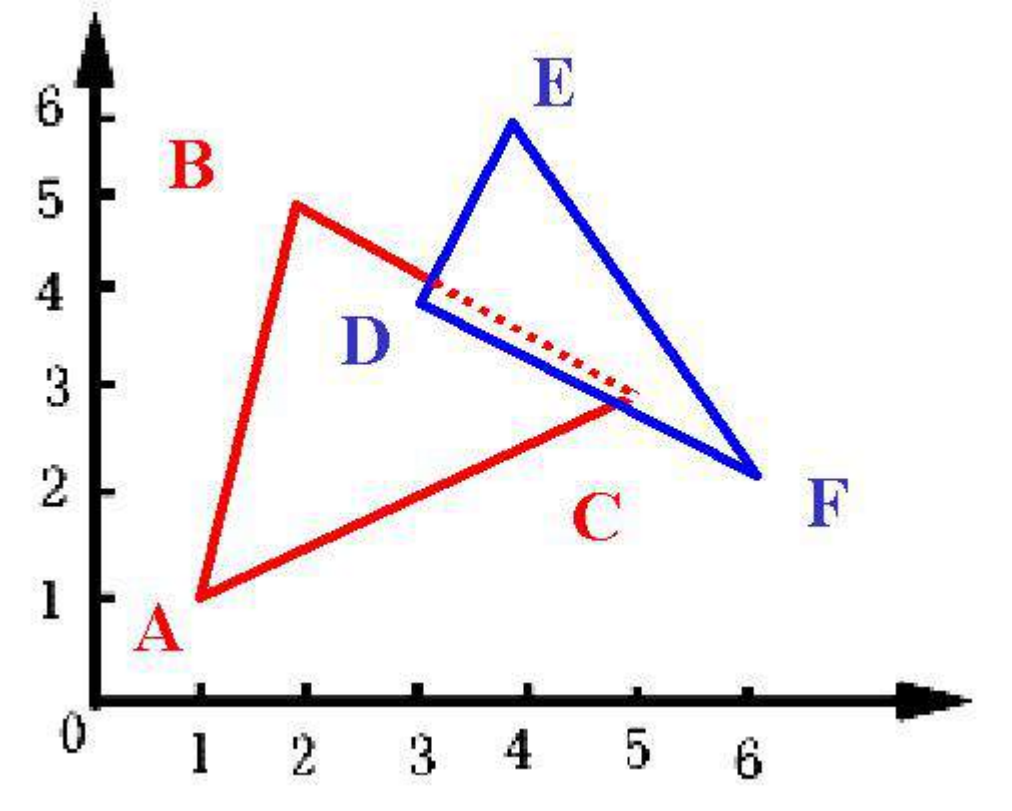
$$z_{ABC}=\frac{-A_1x^{'}-B_1\gamma-D_1}{C_1},z_{DEF}=\frac{-A_2x^{'}-B_2\gamma-D_2}{C_2}$$

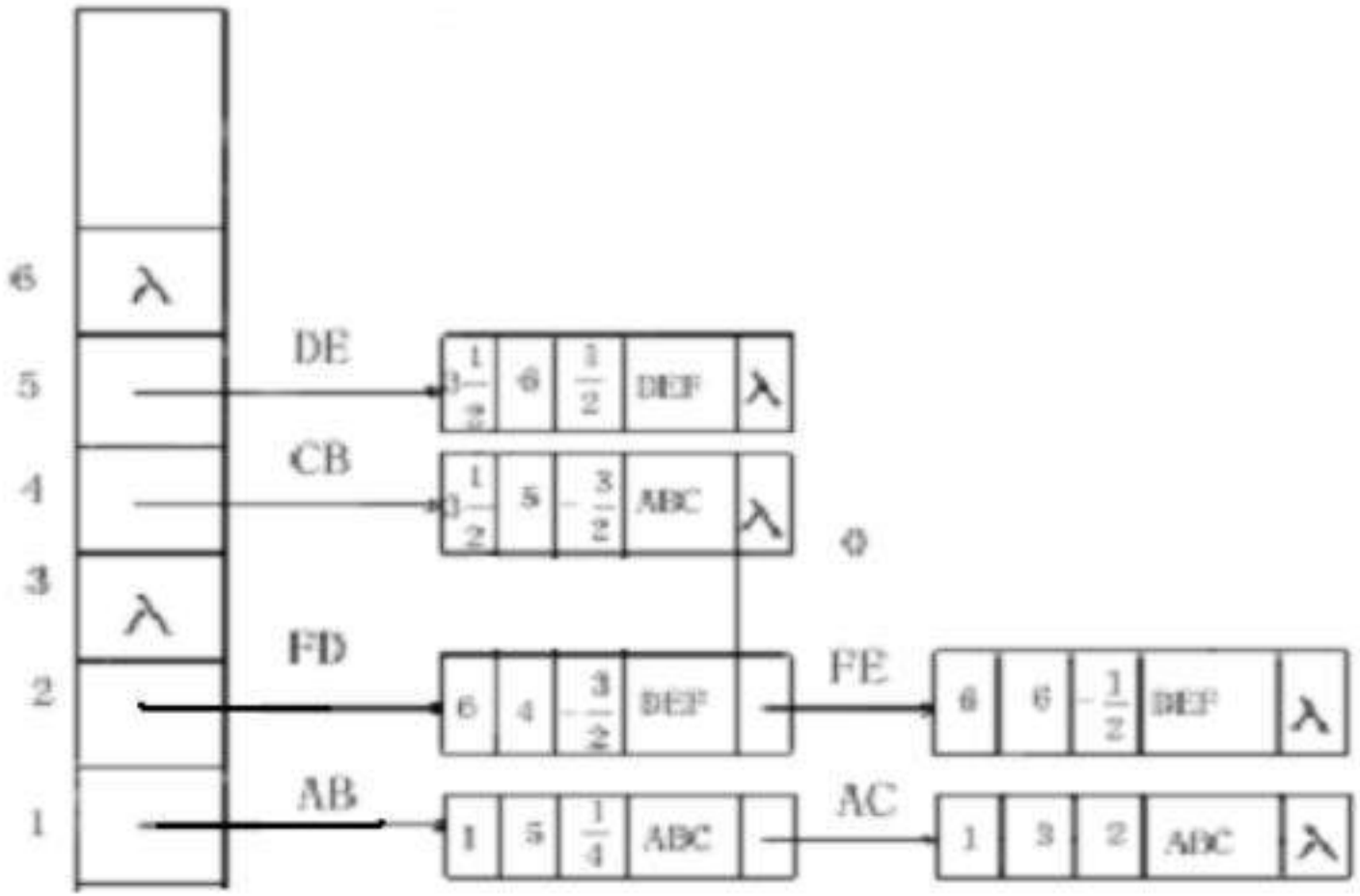
若Z_{DEF} <Z_{ABC} ，则DE到CB之间填DEF的颜色

到CB时，$Flag_{ABC}$=0，BC到EF之间填DEF颜色

到FE时，$Flag_{DEF}$=0

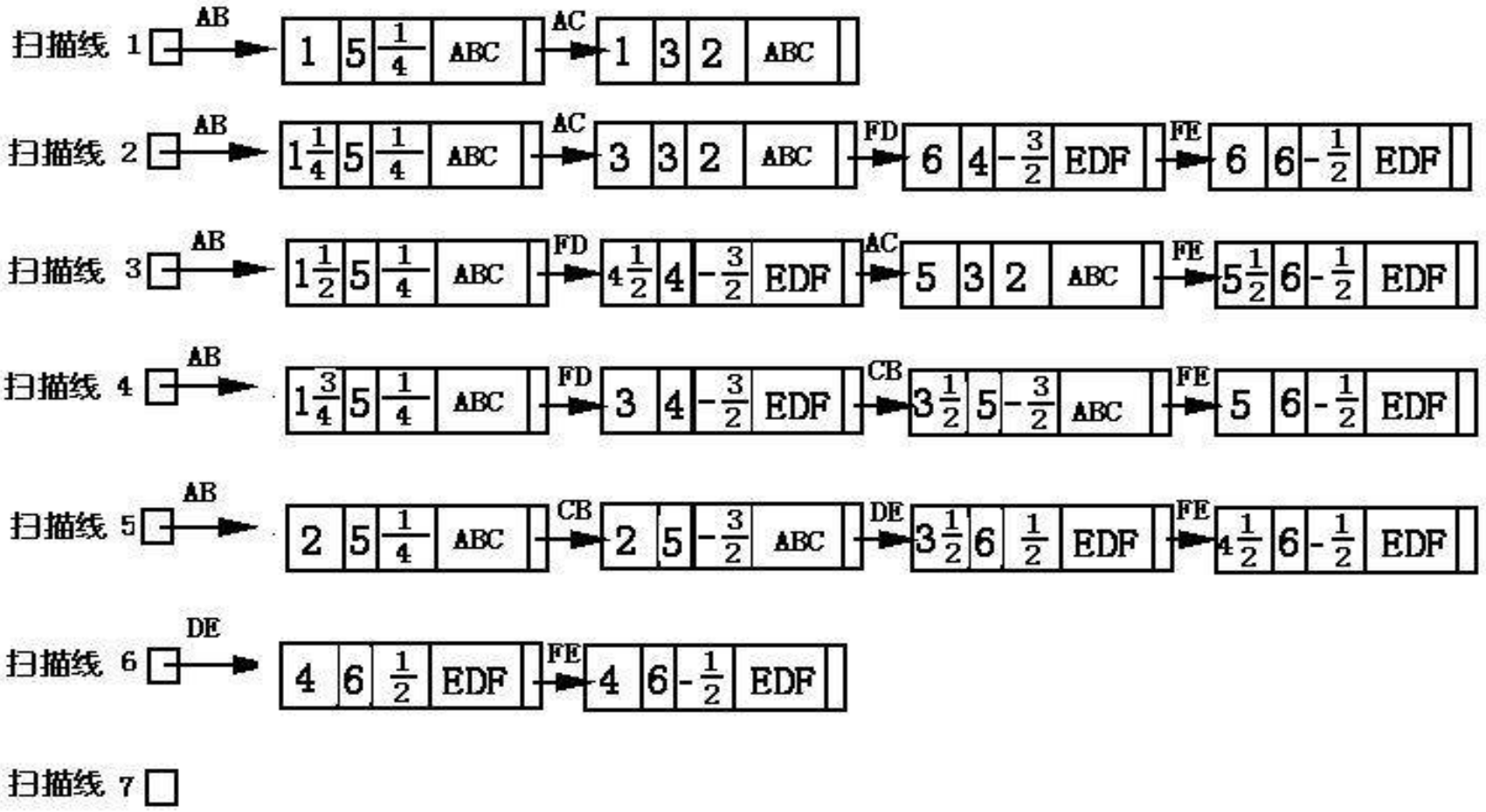
例子：设有两个空间的三角形ABC、DEF，各顶点的坐标依次是（1，1，10），（2，5，10），（5，3，10），（3，4，5），（4，6，5），（6，2，5）




6
λ
5
DE
3 1/2
6
1/2
DEF
λ
4
CB
3 1/2
5
3/2
ABC
λ
Φ
3
λ
2
FD
6
4
3/2
DEF
FE
6
6
1/2
DEF
λ
1
AB
1
5
1/4
ABC
AC
1
3
2
ABC
λ

AET

初始化 □


扫描线 1
AB
1 5 1/4 ABC
AC
1 3 2 ABC
扫描线 2
AB
1 1/4 5 1/4 ABC
AC
3 3 2 ABC
FD
6 4 -3/2 EDF
FE
6 6 -1/2 EDF
扫描线 3
AB
1 1/2 5 1/4 ABC
FD
4 1/2 4 -3/2 EDF
AC
5 3 2 ABC
FE
5 1/2 6 -1/2 EDF
扫描线 4
AB
1 3/4 5 1/4 ABC
FD
3 4 -3/2 EDF
CB
3 1/2 5 -3/2 ABC
FE
5 6 -1/2 EDF
扫描线 5
AB
2 5 1/4 ABC
CB
2 5 -3/2 ABC
DE
3 1/2 6 1/2 EDF
FE
4 1/2 6 -1/2 EDF
扫描线 6
DE
4 6 1/2 EDF
FE
4 6 -1/2 EDF
扫描线 7 □

算法步骤：

第二章第三节 多边形扫描转换算法

3.2:在扫描线y上，按照AET表提供的x坐标对，用color实施填充修改加细如下：

3.2.1

扫描线指向的“吊桶”的指针i←1，

扫描线在多少个多边形内的计数器s←0

活跃多边形表P，初始为空。

3.2.2

设第i个“吊桶”记录的相应多边形是A。

if $Flag_A$=FALSE，

then {$Flag_A$=TRUE，

将A加到表P中，s ← s+1}

else

{$Flag_A$= FALSE

将P中删除A， s ← s-1 }

3.2.3

if s=0，goto 3.2.5（这时扫描线不在任何多边形内，正通过背景，不必做扫描转换）

if s=1，goto 3.2.4（这时扫描线只在一个多边形内，不必做深度比较，去做扫描转换。）

若前面两个判断都为“假”，扫描线至少在两个多边形内，应做深度比较。对表P前面两个多边形做深度比较，比较后排序应保证P表中的多边形按深度递增的次序。

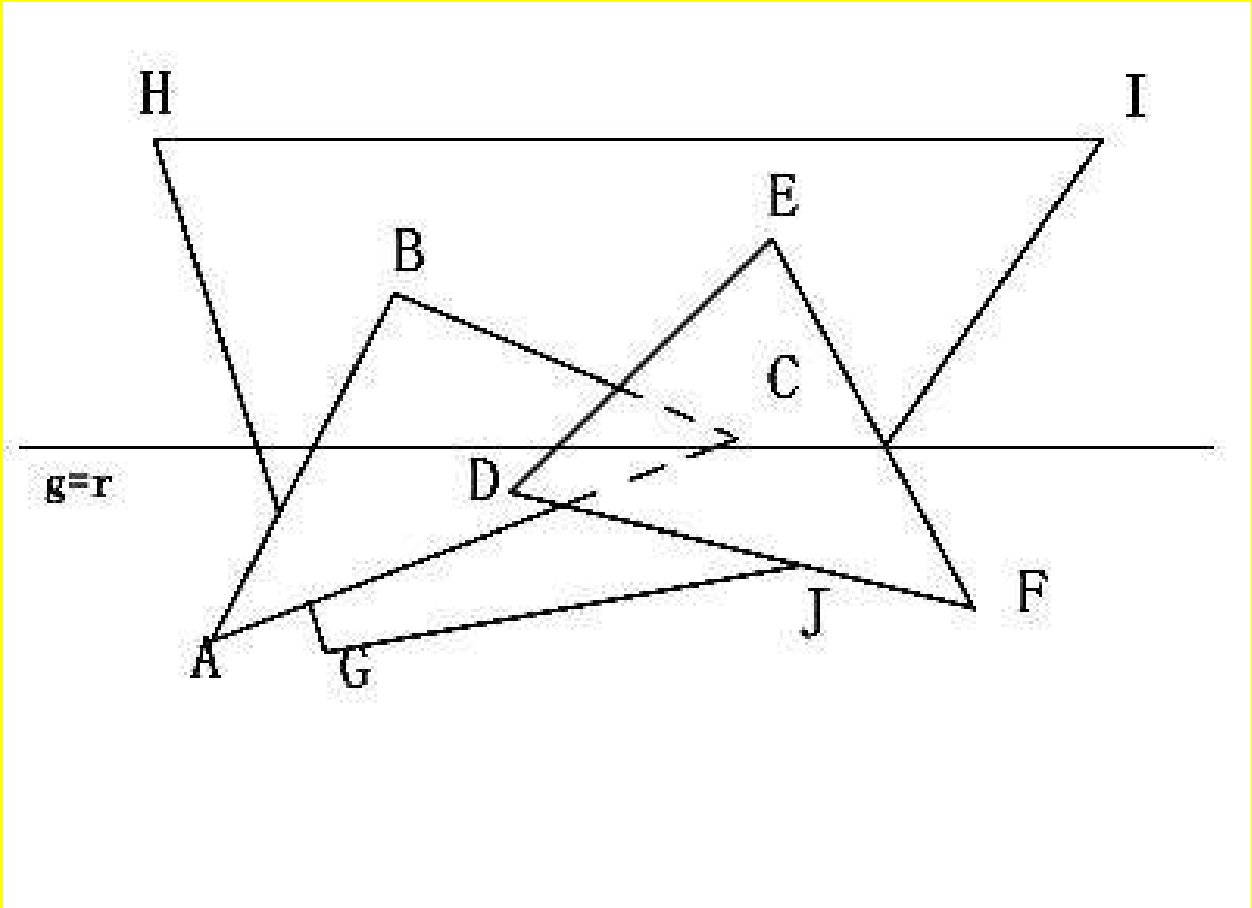
3.2.4　　对第i个和第i+1个“吊桶”存有的x坐标指示的扫描线上的一段，按照P表最前面多边形指示的亮度或颜色，实施扫描转换。

3.2.5　i增加1，若i所指已无“吊桶”，步骤结束，去下一步骤3.3（删除y=ymax的边）。否则，回到步骤3.2.2。

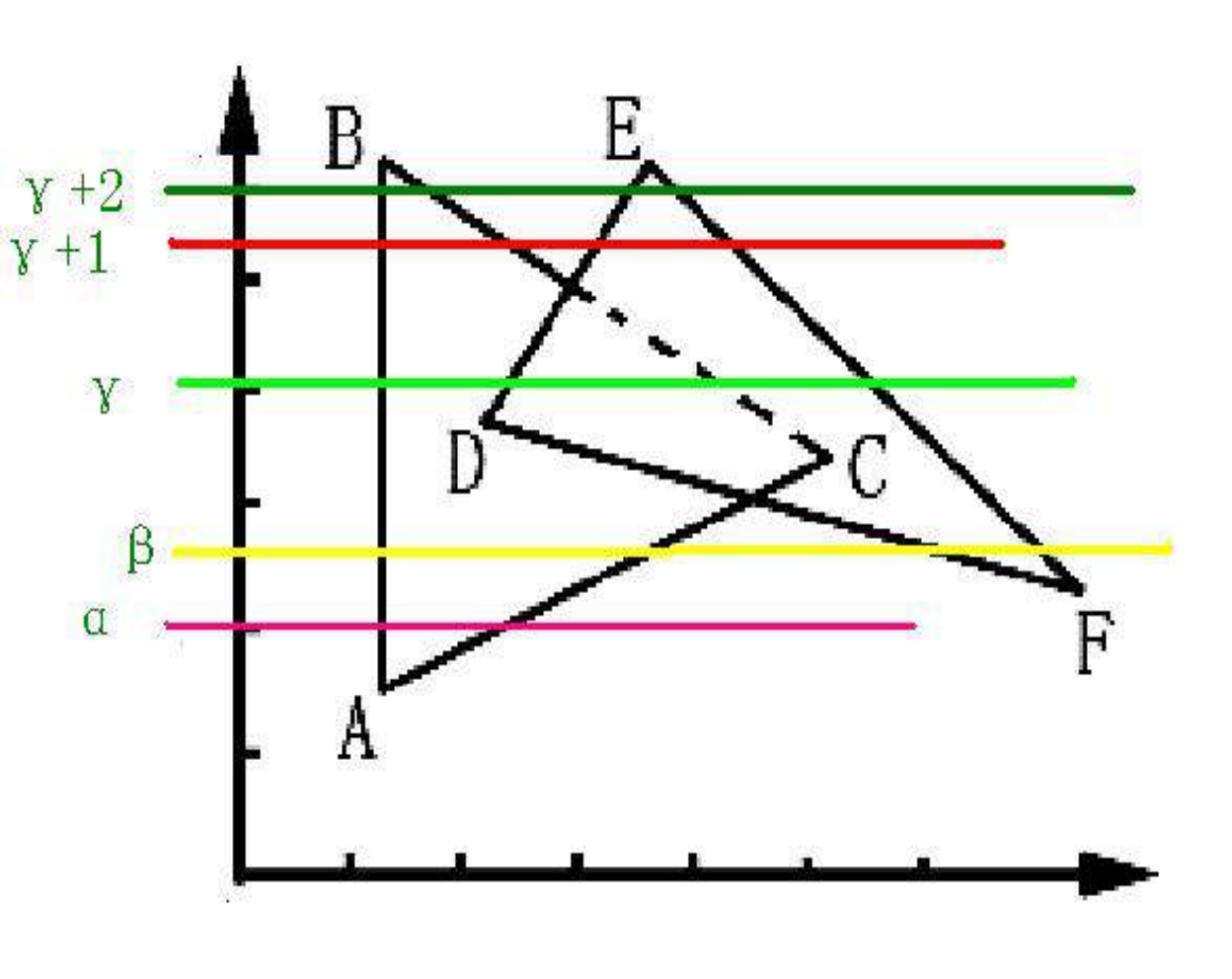
i	s	P	PT表 FABC FDEF		说明
1	1	(ABC)	true		1$\frac{3}{4}$ 到3填ABC的亮度或颜色值
2	2	(DEF, ABC)		true	在步骤3发生深度比较，比较结果DEF更靠近观察者，它仍在P表前面，3到3 $\frac{1}{2}$填DEF的亮度或颜色值
3	1	(DEF)	false		3$\frac{1}{2}$ 到5填DEF的亮度或颜色值
4	0	( )	false		

3.2.3深度比较条件太宽松，实际很多不必要的的比较

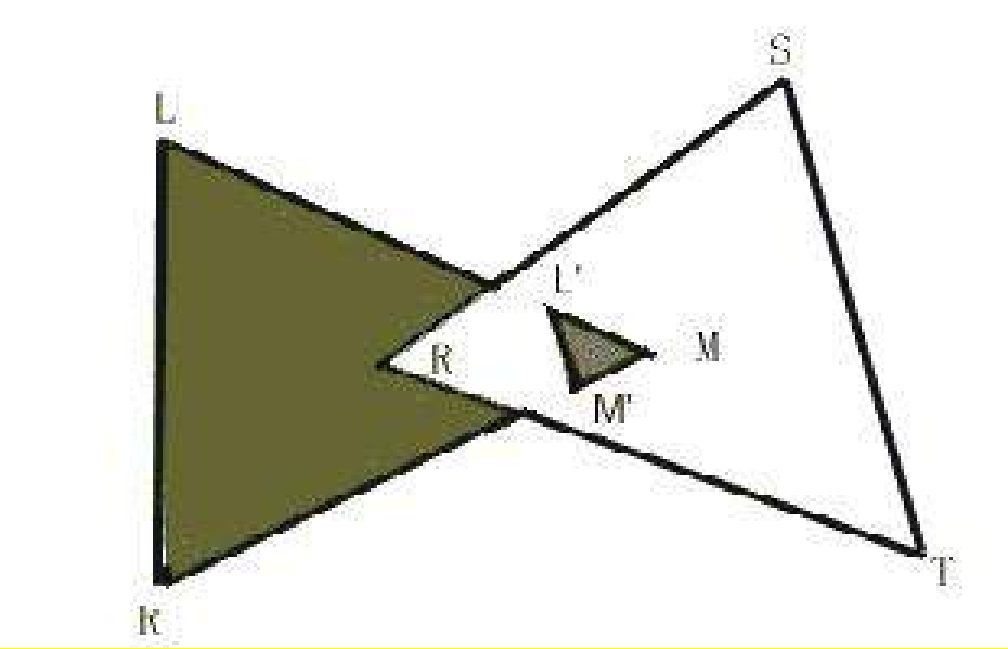
改进1：假定**没有多边形穿透另一个多边形的情况下**，**如果扫描线从一个*被遮挡*的多边形中走出，深度比较将是*不必要*的；扫描线从一个*遮挡*了其它多边形的多边形中走出，深度比较才可能*必要*。**



改进2：利用深度相关性，对于一组相邻的扫描线来说，多边形之间的深度关系常常是不



改进3：对于穿透情况的处理，将一个多边形分成两个子多边形



改进4：背景颜色处理

（1）帧缓存初始为某个特定的值

（2）定义一个包含了客体中所有的多边形，位于比其它多边形都更远离观察者的平行于投影平面的一个平面上，并具有某个合适的亮度或颜色值。

（3）修改算法。当扫描线不在任何多边形内时，就往帧缓冲存储器中送入背景的象素值。

第七节 区域分割算法

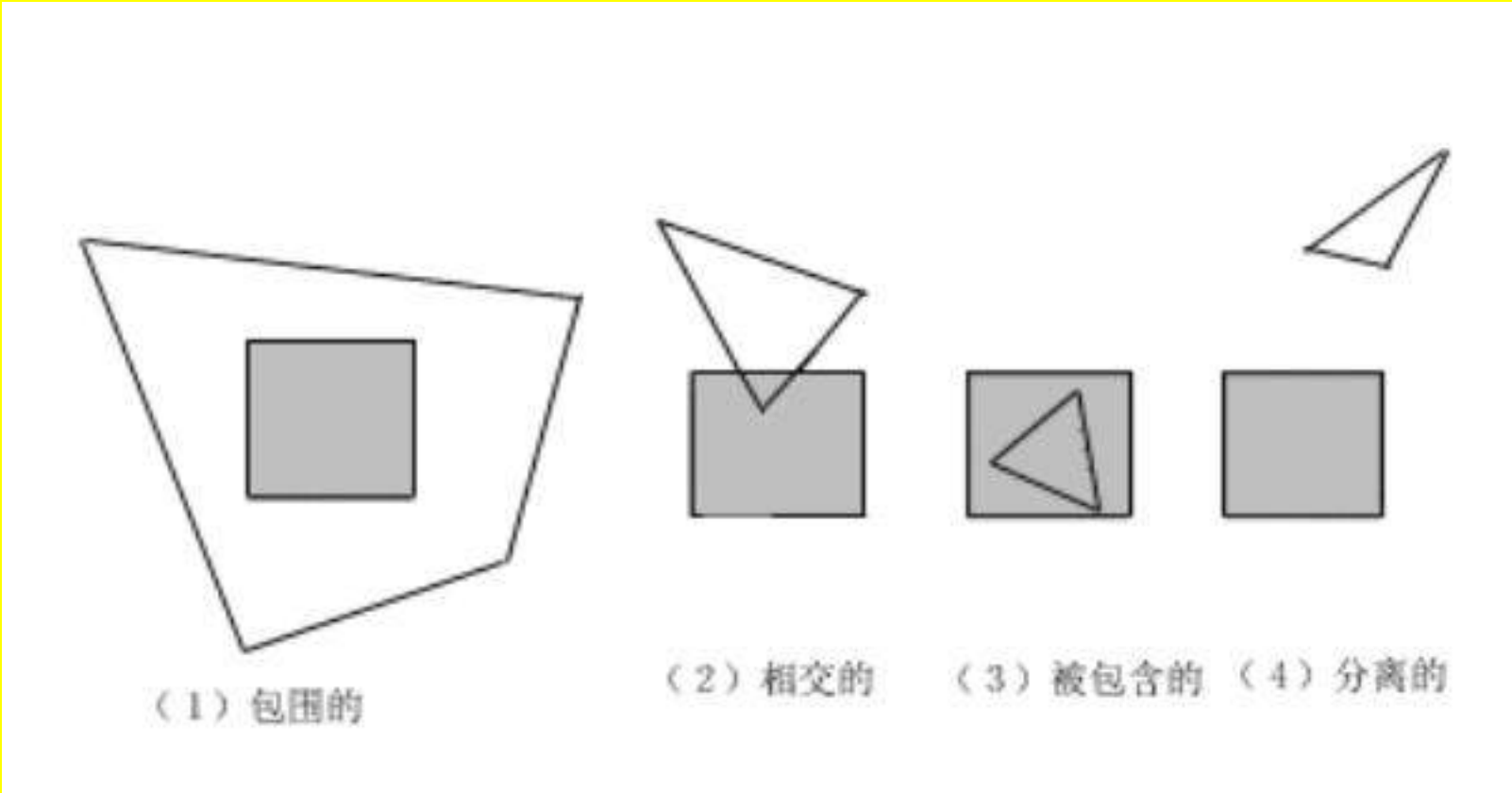
区域分割算法：

将投影平面分割成区域，考察区域内的图象。如果容易决定在这个区域内某些多边形是可见的，那么就可以显示那些可见的多边形，完成对这一区域的显示任务。否则，就将区域再分割成小的区域，对小的区域递归地进行判断。由于区域逐渐变小，在每个区域内的多边形逐渐变小，最终总可以判定哪些多边形是可见的。这个算法利用的区域的相关性，这种相关性是指位于适当大小的区域内的所有象素，表示的其实是同一个表面。

区域分割算法：图像空间算法

形体中多边形的投影多边形与所考察区域之间的关系：

1. 包围的多边形，即多边形包围了所考察的区域。
2. 相交的多边形，即多边形所考察的区域相交。
3. 被包含的多边形，即多边形全部在所考察的区域之内。
4. 分离的多边形，即多边形与所考察的区域完全分离。

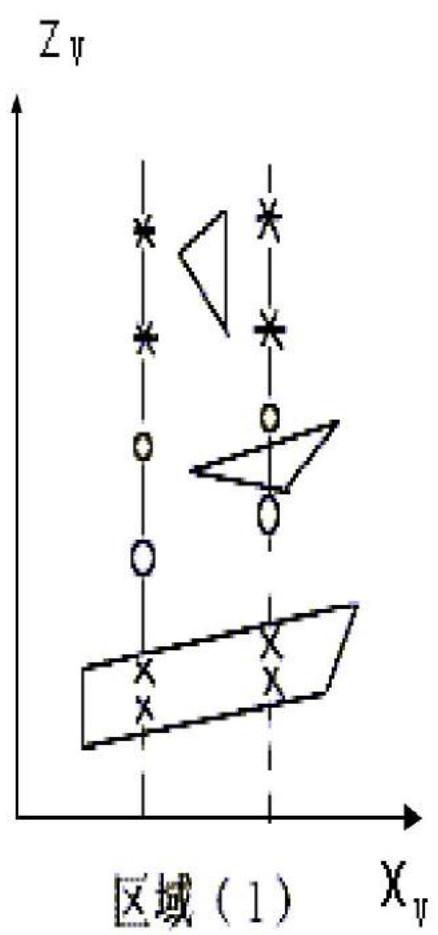


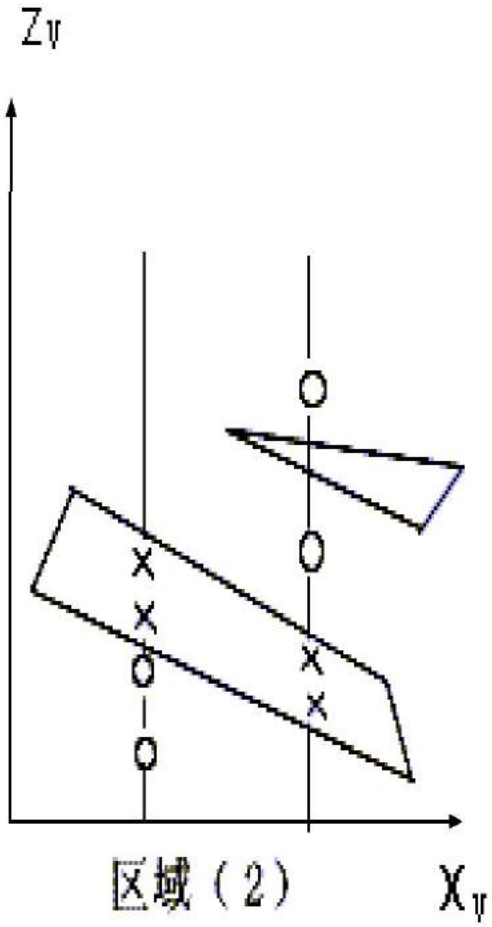
区域不必再进行分割的情况：

1. 所有的多边形与区域分离，所以在区域内只需显示背景值。
2. 只有一个相交的多边形，或者只有一个被包含的多边形。这时可以对区域首先填充背景值，然后对多边形进行扫描转换。（相交的多边形，只对被包含的部分做扫描转换）
3. 只有一个包围的多边形，无其它的多边形。整个区域填充该多边形的象素值。

（1, 2, 3用范围检查）

4. 多个多边形与之相交、包围、或被包含，且有一个包围的多边形位于其它多边形最前面，区域填充为该多边形的颜色。（深度检查）





X--表示包围的多边形平面交点

*--表示被包围的多边形平面交点

O--表示相交的多边形平面交点

情形4的两种情形

对不能做出判断的情形的处理方法：

（1）按照原算法将该区域分割

（2）修改判断是否在最前面的算法

判别相交的多边形是否在整个包围多边形的后面，将相交多边形的顶点坐标代入包围多边形的平面方程，判别符号。

区域分割成子区域：

1. 子区域需考察的情况：

只需考虑父区域包含的或相交的多边形。

对于分离的或包围的多边形，进行区域分割之后仍保持分离包围的关系。

2. 分割中止的条件：一个象素单位或最大数目的分割

对于仍不能判断的，则找出中心点，计算中心点对应的所有多边形的z值，取z值最小的多边形象素值来填充该区域。

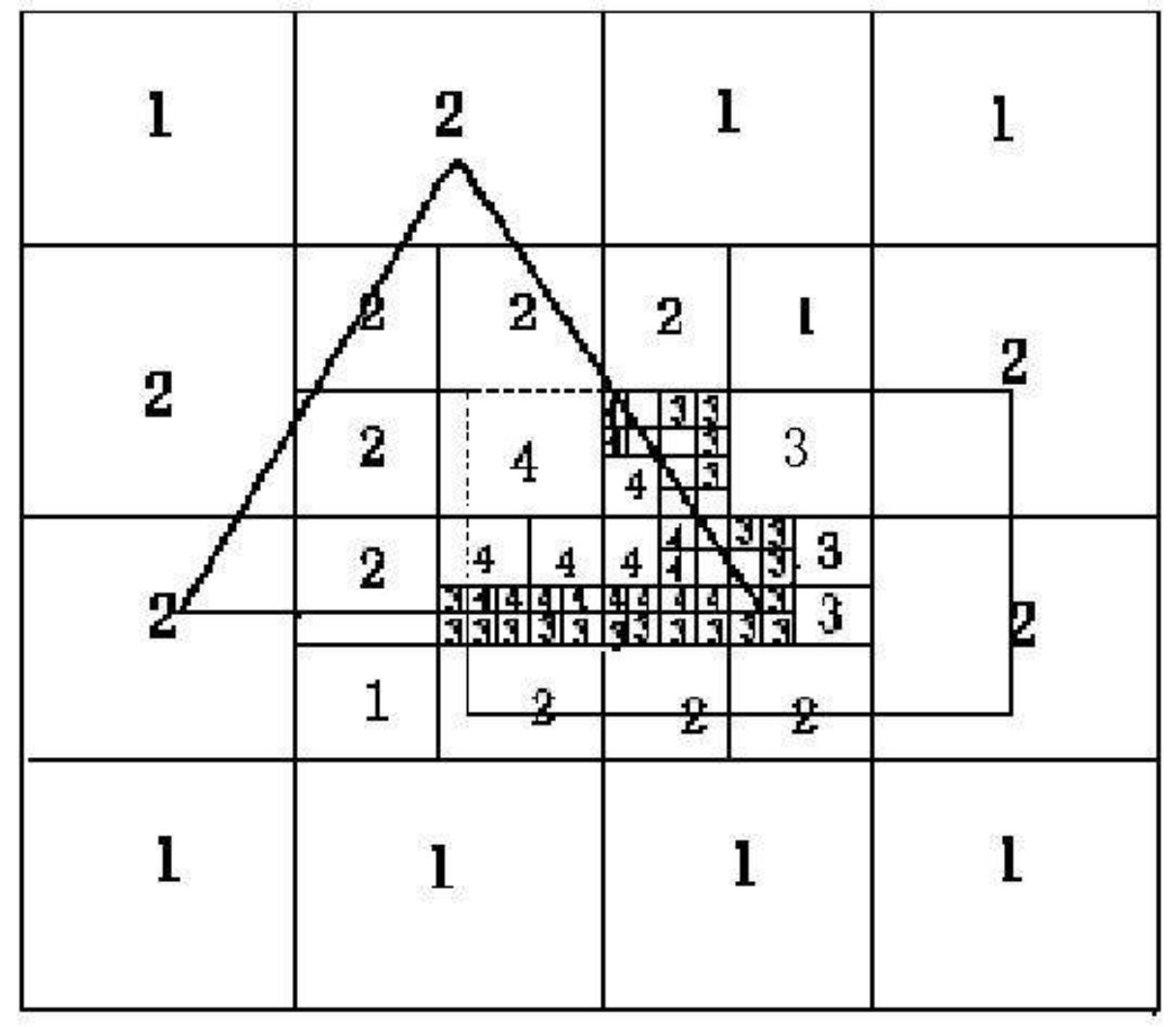
3. 分割方法

（1）等分：分割到能填充为止

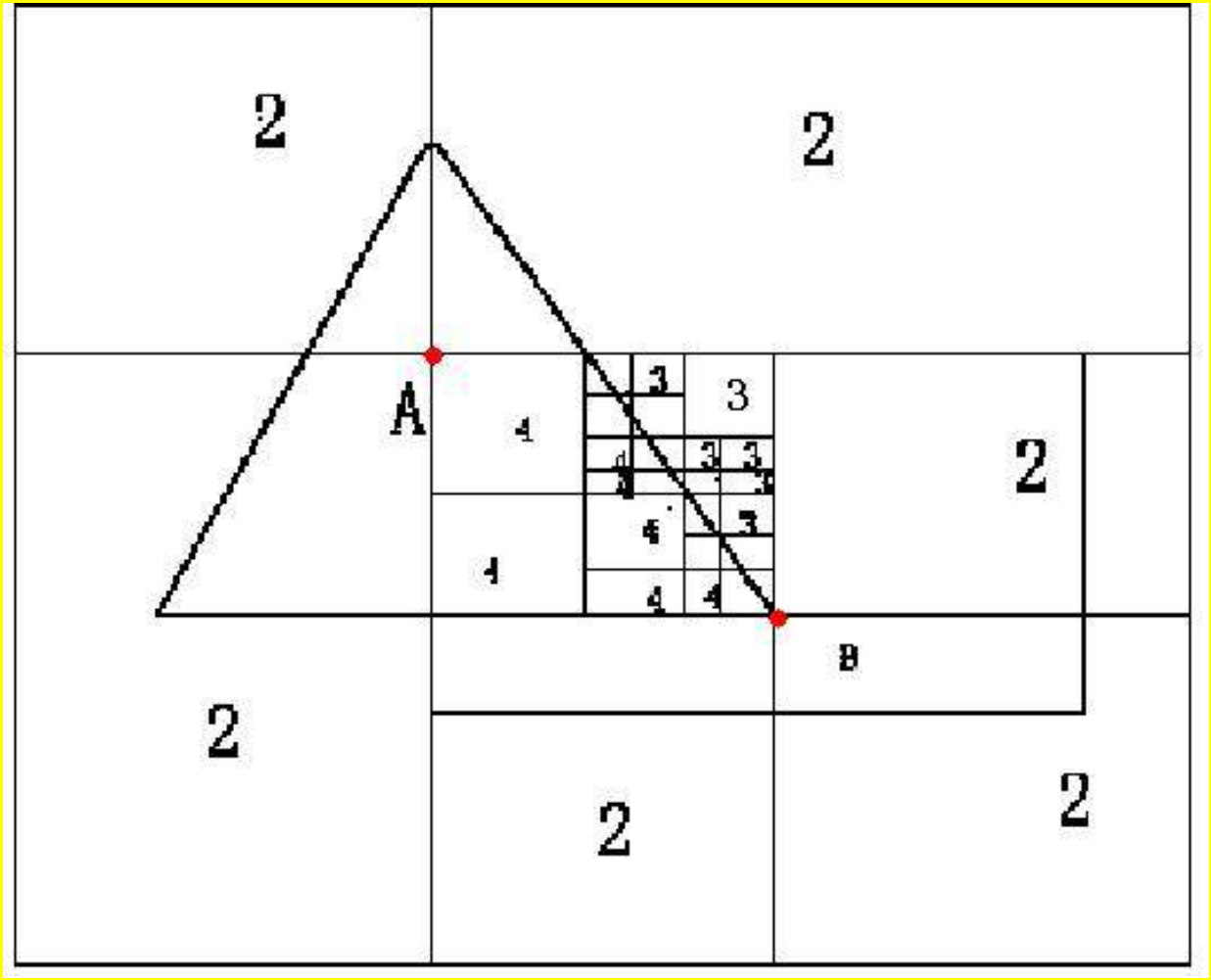
（2）按多边形顶点做分割

（3）按多边形投影进行分割

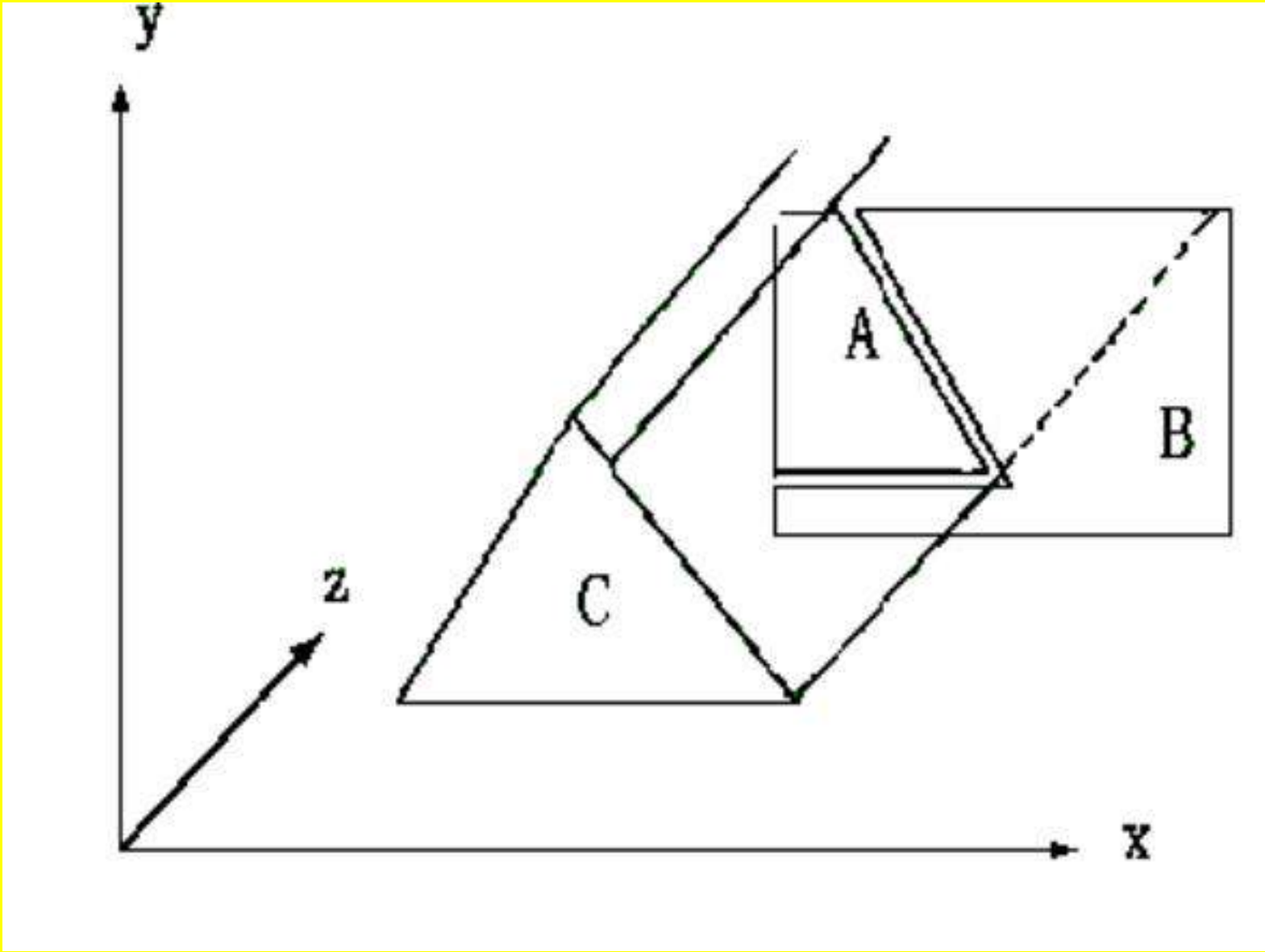
（1）等分：分割区域为正方形



（2）按照顶点位置来做分割(先A后B)



（3）按照多边形投影选择区域做分割



第一节　线面比较法消除隐藏线
　　夹角法
　　线面比较法

第三节　深度排序算法

第五节　Z一缓冲算法
第六节　扫描线算法
第七节　区域分割算法

第四章　　作业

3. 两段曲线连续的条件：P1=Q0;

二阶导数连续的条件是：P1’=Q0’

P1’’=Q0’’

两段hermite曲线光滑拼接条件：（二阶几何连续）

P1=Q0;

P1’=k1*Q0’

P1’’=k2*Q0’’

9. $$P(\frac{1}{5}) = (\frac{211}{125}, \frac{49}{25})$$

$$P_{0,n}(t) = P_{0,n-1}(t) + t(P_{1,n}(t) - P_{0,n-1}(t))$$

10.

前半段$Q_0(1,1), Q_1(\frac{3}{2},2), Q_2(\frac{9}{4},\frac{5}{2}), Q_3(\frac{11}{4},\frac{5}{2})$

后半段$R_0(\frac{11}{4},\frac{5}{2}), R_1(\frac{13}{4},\frac{5}{2}), R_2(\frac{7}{2},2), R_3(3,1)$

15. 利用凸包性以及起点和终点位置的关系求得

16.

Q1, Q2, P1, P2　　S为B样条曲线的起点

$$S' = \frac{1}{2}(P_1 - Q_1) \Rightarrow Q_1$$

$$S = Q_2 + \frac{1}{3}\left[\frac{1}{2}(P_1 + Q_1) - Q_2)\right] \Rightarrow Q_2$$

Q1, Q2, P1, P2　　S为B样条曲线的终点

$$S' = \frac{1}{2}(P_2 - Q_2) \Rightarrow Q_2$$

$$S = P_1 + \frac{1}{3}\left[\frac{1}{2}(P_2 + Q_2) - P_1)\right] \Rightarrow Q_2$$

只能求出Q_2,无法求出Q_1

21. P0 (−18, −39)　　P3 (27, 15)

P0, V1, V2, V3

24.
$$\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_4 \\ P_1' \\ P_4' \end{bmatrix} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{bmatrix}$$

$$\Rightarrow \begin{cases} B_1 = P_1 \\ B_2 = \frac{1}{3} P_1' + P_1 \\ B_3 = P_4 - \frac{1}{3} P_4' \\ B_4 = P_4 \end{cases}$$

$$\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_4 \\ P_1' \\ P_4' \end{bmatrix} = \frac{1}{6} \times \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix}$$

$$\Rightarrow \begin{cases} V_1 = 2P_4 - P_1 - \frac{7}{3}P_1' - \frac{2}{3}P_4' \\ V_2 = 2P_1 - P_4 + \frac{2}{3}P_1' + \frac{1}{3}P_4' \\ V_3 = 2P_4 - P_1 - \frac{1}{3}P_1' - \frac{2}{3}P_4' \\ V_4 = 2P_1 - P_4 + \frac{2}{3}P_1' + \frac{7}{3}P_4' \end{cases}$$

27. 三次等距B样条曲线，起点和终点分别是单位圆在x轴, y轴上的点，利用三角形的关系求出控制点

第七章　消除隐藏线和隐藏面的算法

消隐　　面消隐

　　线消隐

假定：三维形体表示为多边形表面的集合

投影约定为沿着z轴正向的正交投影

消除隐藏面算法：

　　图象空间算法

　　客体空间算法

- 图象空间算法对显示设备上每一个可分辨象素进行判断，看组成物体的多个多边形表面中哪一个在该象素上可见，即要对每一象素检查所有的表面。
- 客体空间算法把注意力集中在分析要显示形体各部分之间的关系上，这种算法对每一个组成形体的表面，都要与其它各表面进行比较，以便消去不可见的面或面的不可见部分。

第一节 线面比较法消除隐藏线

- 多面体的面可见性

  对象：凸多面体

  可见面：朝向观察位置的面

  观察方向：由指向观察位置的一个方向向量k给出，面的外法向量是n，则这两个向量的夹角 α 满足$0 \leq \alpha < \pi/2$时，所考查面是可见的，否则就是不可见的

把n和k记作 $n(n_x,n_y,n_z),k=(k_x,k_y,k_z)$，

则

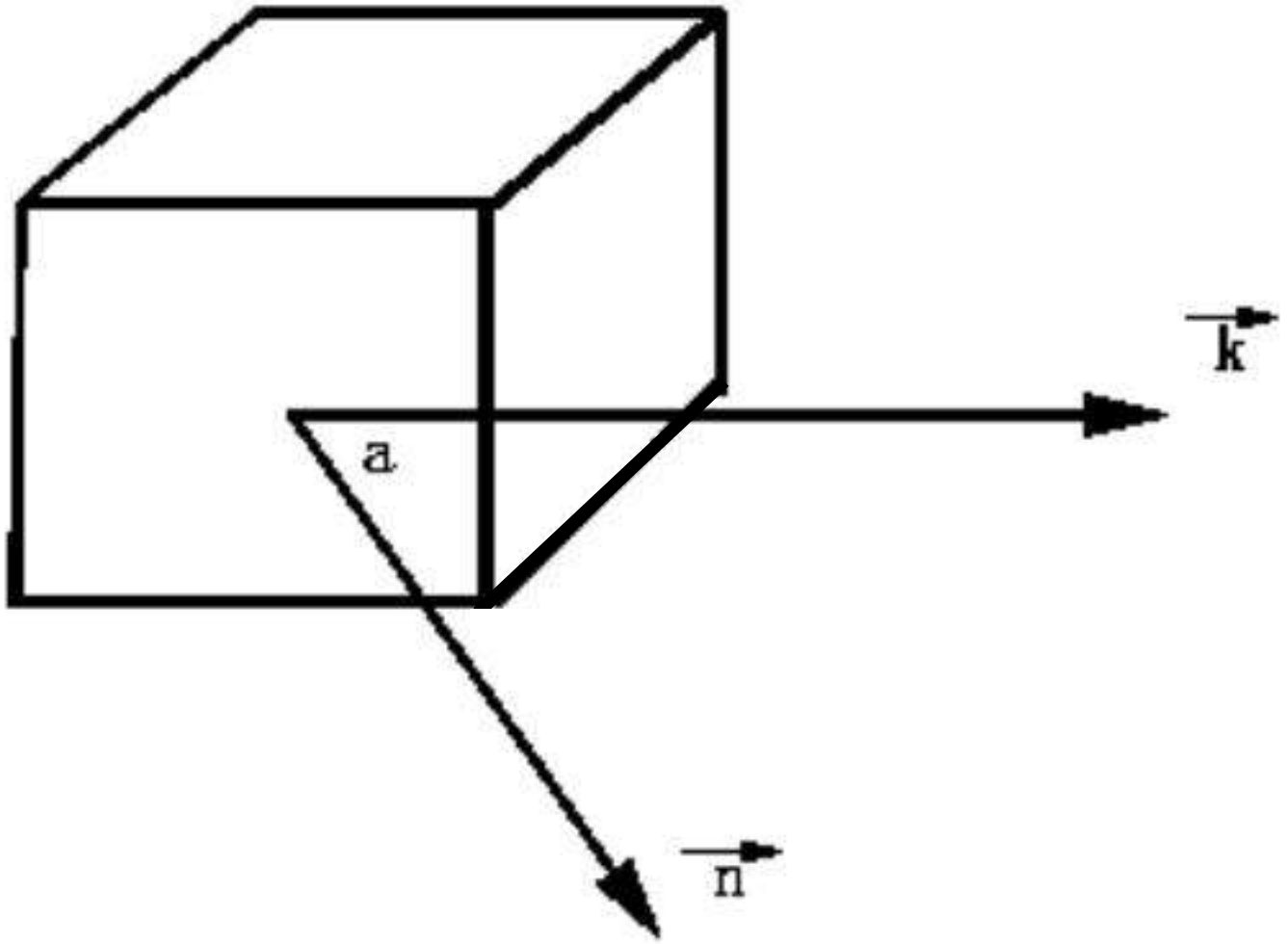
$$\cos\alpha=\frac{n\cdot k}{|n||k|}=\frac{n_xk_x+n_yk_y+n_zk_z}{\sqrt{n_x^2+n_y^2+n_z^2}\cdot\sqrt{k_x^2+k_y^2+k_z^2}}$$

$$\Delta=n_xk_x+n_yk_y+n_zk_z$$

$\Delta>0$ $0\le\alpha<\frac{\pi}{2}$ 面为可见

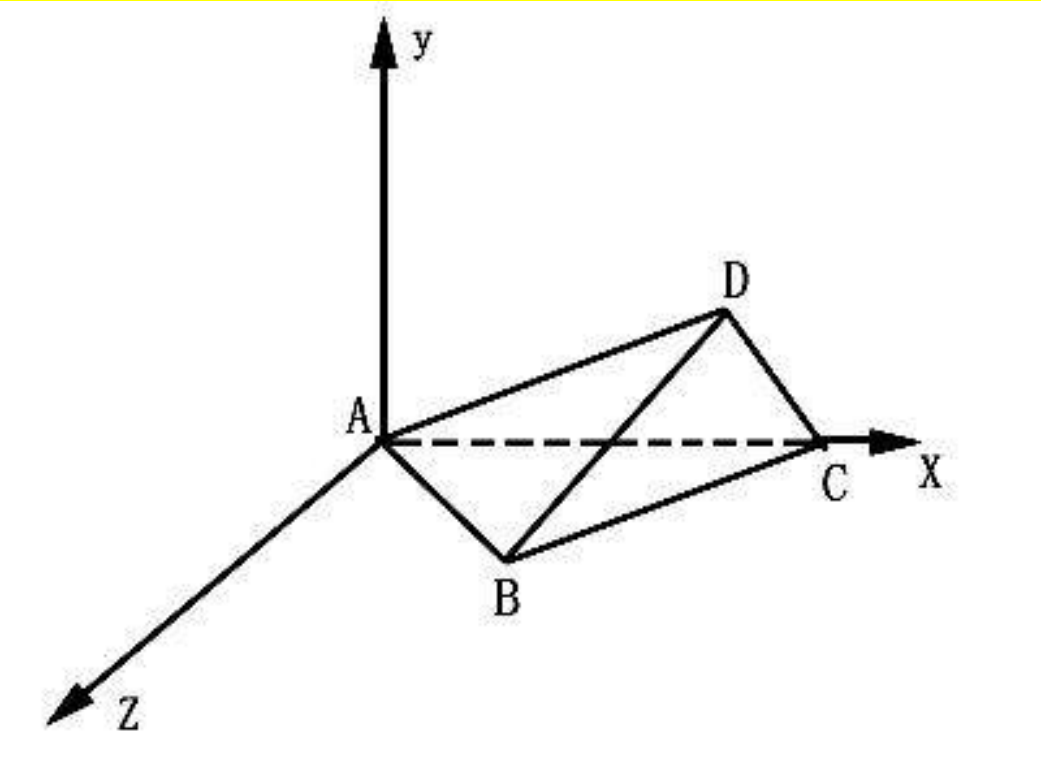
$\Delta<0$ $\frac{\pi}{2}<\alpha\le\pi$ 面不可见

$\Delta=0$ $\alpha=\frac{\pi}{2}$ 面退化为线。


k
a
n

设空间有一个四面体，顶点A，B，C，D的坐标依次是（0，0，0），（2，0，1），（4，0,0），(3,2,1)从z轴正向无穷远处观察,求各面的可见性

观察方向向量是k=(0,0,1),



三角面DAB的法向量是:

$$\begin{aligned} n &= \overrightarrow{DA} \times \overrightarrow{AB} \\ &= (-3,-2,-1) \times (2,0,1) \\ &= (-2,1,4) \end{aligned}$$

因此, $n \cdot k = 4 > 0$ ， 面DAB为可见面. 类似计算可知, 面DBC是可见面, 面ADC是不可见面, 面ACB退化为线。

单个凸多面体——可见面上的线是可见线，

多个凸多面体或非凸多面体——用上面的方法预处理，剩下可能可见面

可能可见的棱线:在可见面上或与可见面有边界

线面比较法：（观察位置位于Z轴负方向无穷远处）

0：预处理，用外法线法判断出所有可能可见面

思想：求出所有可能可见面，可能可见面上的线段是可能可见线。

每一条**可能可见线**和每一个**可能可见面**比较，

⇓ 线段

⇓ 多边形表面

1. 范围检查（xv, yv方向）

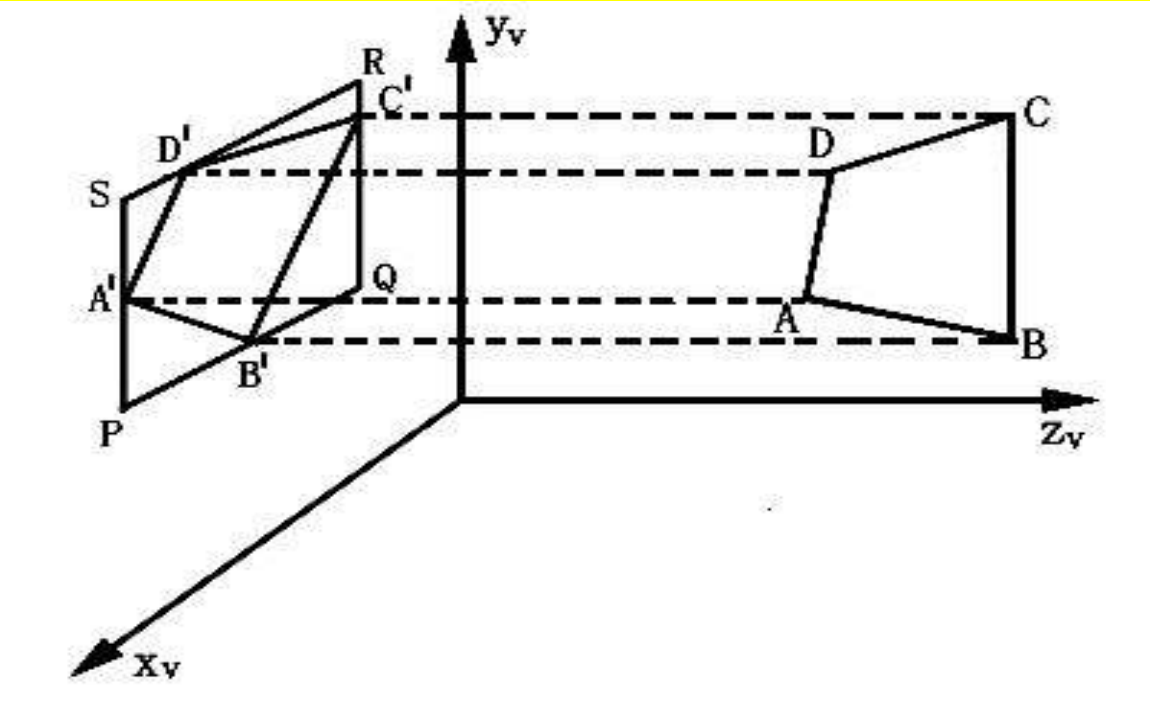
①求出线段的投影

xmin1, xmax1, ymin1, ymax1

② 求出多边形表面的投影范围

（zv平面上包含多边形投影的最小矩形）

xmin2, xmax2, ymin2, ymax2



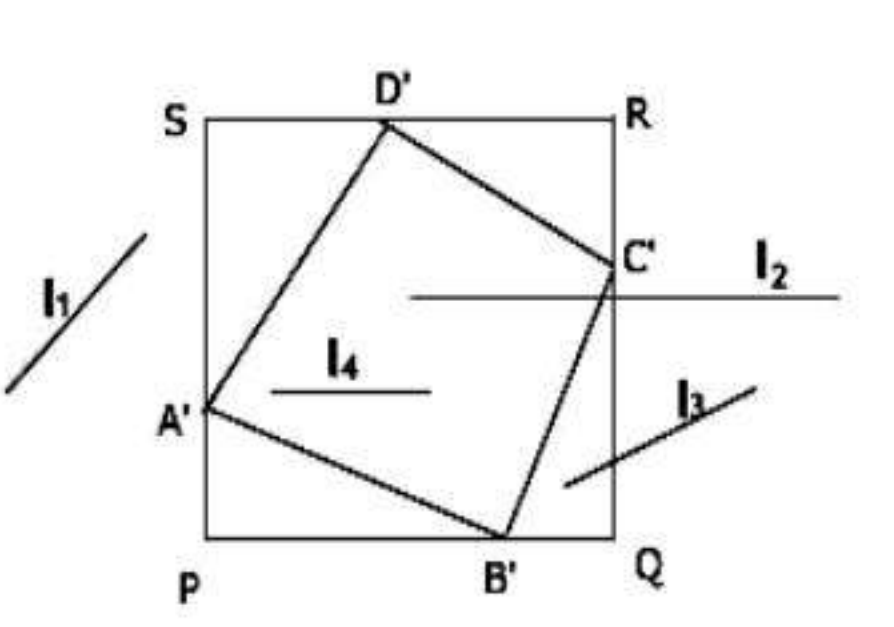
③比较投影范围：

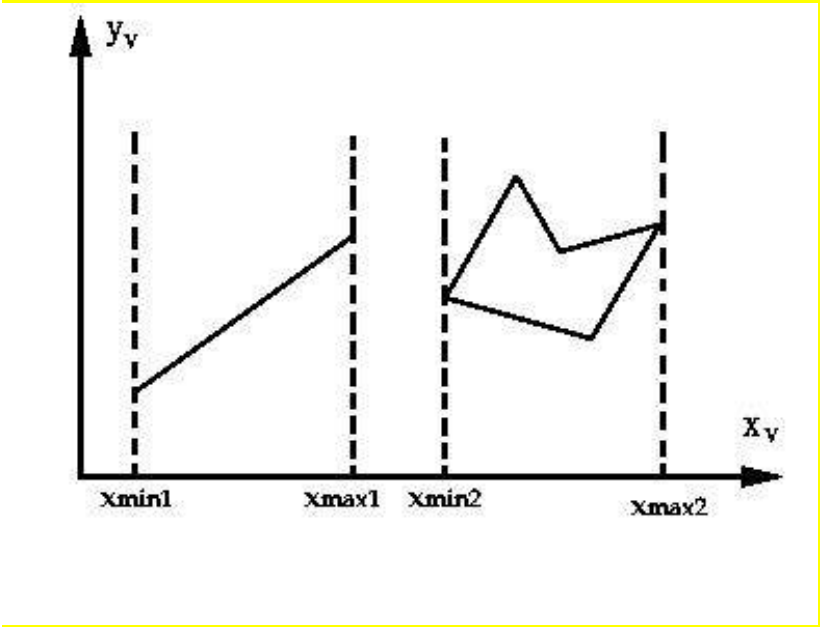
在xv方向：

若xmax1<=xmin2或xmax2<=xmin1,则无遮挡关系

Yv方向：

若ymax1<=ymin2或ymax2<=ymin1,则无遮挡关系

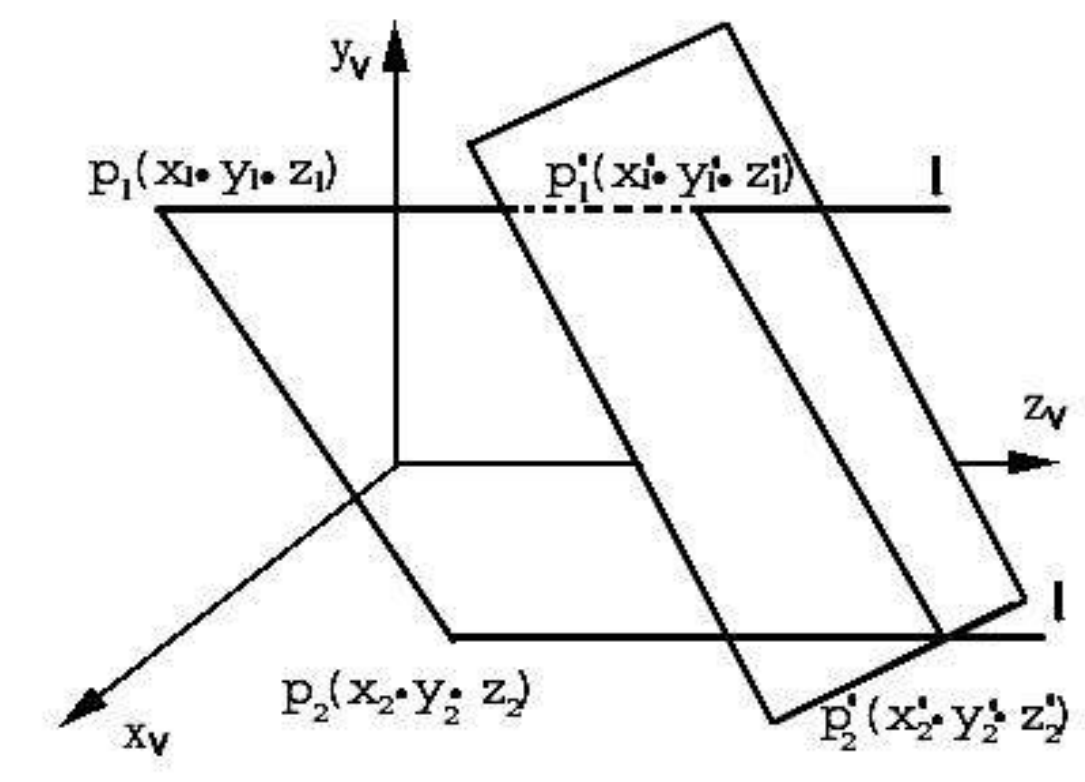




2. z_v方向粗略的深度检查。

若线段投影的最大z坐标z_{max1}小于多边形表面投影范围最小的z坐标z_{min2}，则线段完全在表面前面，根不发生遮挡现象，可以不必再往下做精确的深度检验。

3. 精确的深度检验



若 $z_1 \le z_1'$且$z_2 \le z_2'$，则线段不被遮挡

若 $z_1 \ge z_1'$且$z_2 \ge z_2'$，有可能遮挡需要进一步检查

若非以上两种情况，必然相交，求出交点，交点将原线段分成两段，分别属于上面两种情况

求出z1, z1’

设平面方程为 $Ax+By+Cz+D=0$

直线L_1的参数方程

$X=x_1$，

$Y=y_1$，

$Z=z_1+t$，代入平面方程得：$Ax_1+By_1+C(z_1+t)+D=0$

解得 $t=-\dfrac{Ax_1+By_1+Cz_1+D}{C}$

若t≥0,则z1≤ z1’，靠近视点 可见

若t＜0,则z1>z1’，远离视点

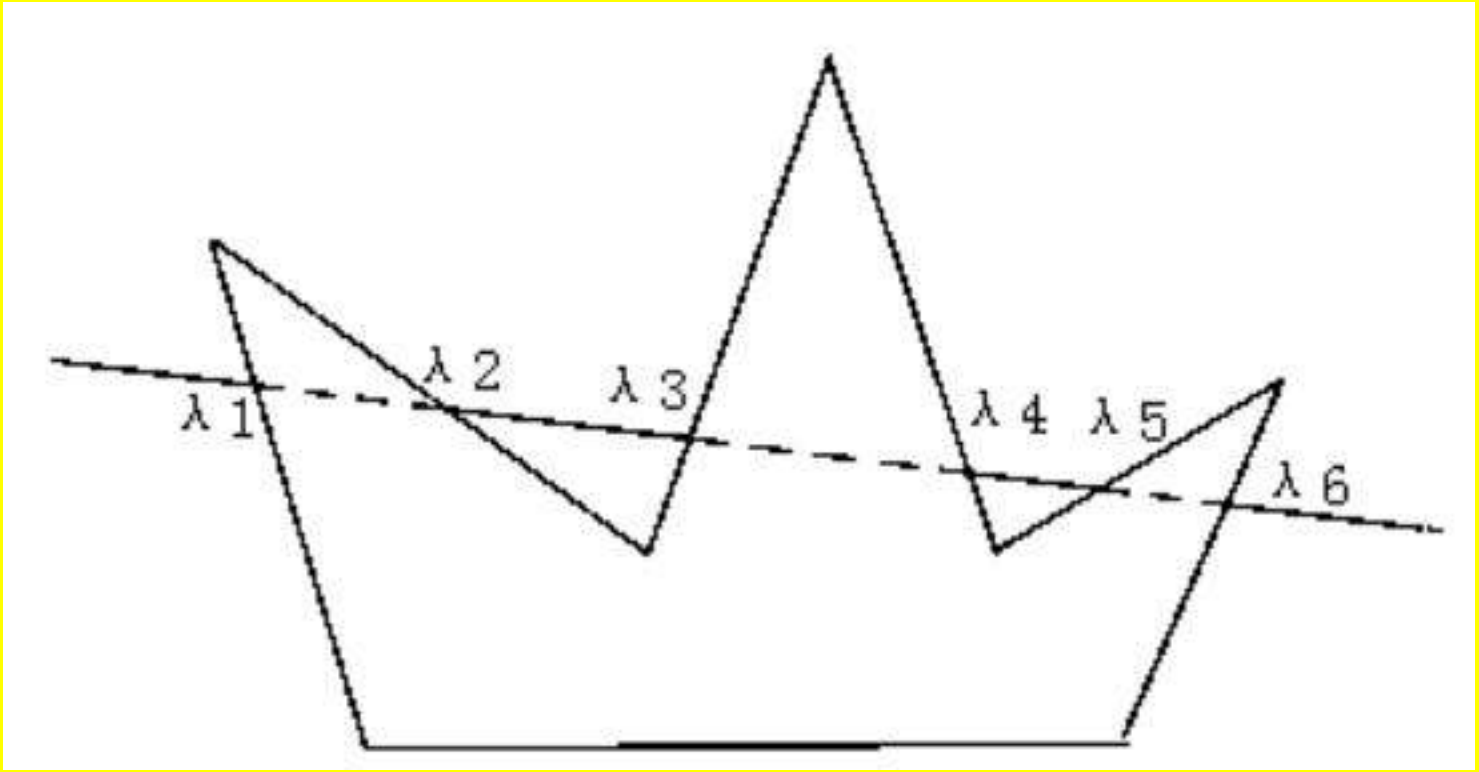
4. 进一步检查

对平面遮挡了线段的哪些部分做精确计算

求线段投影与多边形边框投影的交点（第五章 第一节）

设交点已经求出，设其对应的参数λ ，按从小到大依次排序后是λ1，λ2。。。，则这些交点将投影线段分成的各子线段的可见性应是可见，不可见交替出现。

判断子线段的可见性：（第五章 第四节）



需要检查出某一段子线段是否可见。为此可以取子线段上任意一点，若这点在多边形表面各边线的投影所形成的封闭多边形内，这子线段就不可见，否则就可见。

第三节 深度排序算法

深度排序算法：先画最远的，再画最近的（优先级算法，画家算法）

深度排序算法的主要步骤：

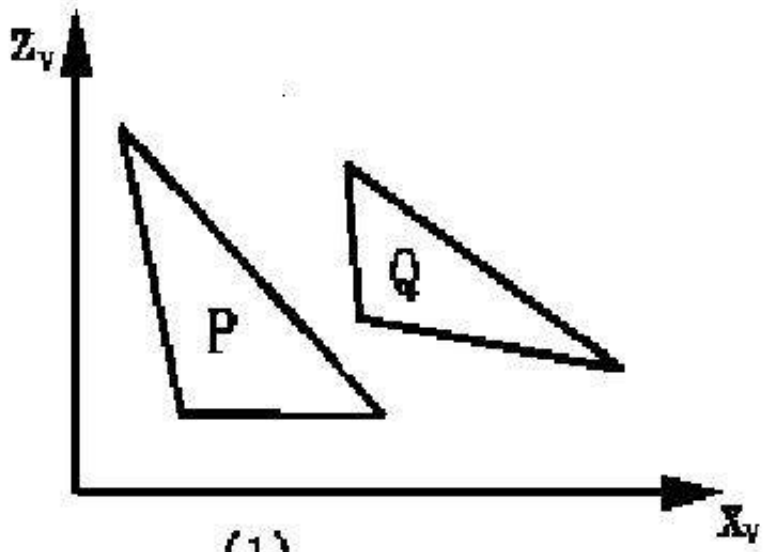
1. 把所有的多边形按顶点最大z坐标值进行排序。（客体空间）
2. 解决当多边形z范围发生交迭时出现的不明确问题。
3. 按最大z坐标值逐渐减小的次序，对每个多边形进行扫描转换。（图像空间）

不明确问题检验方法

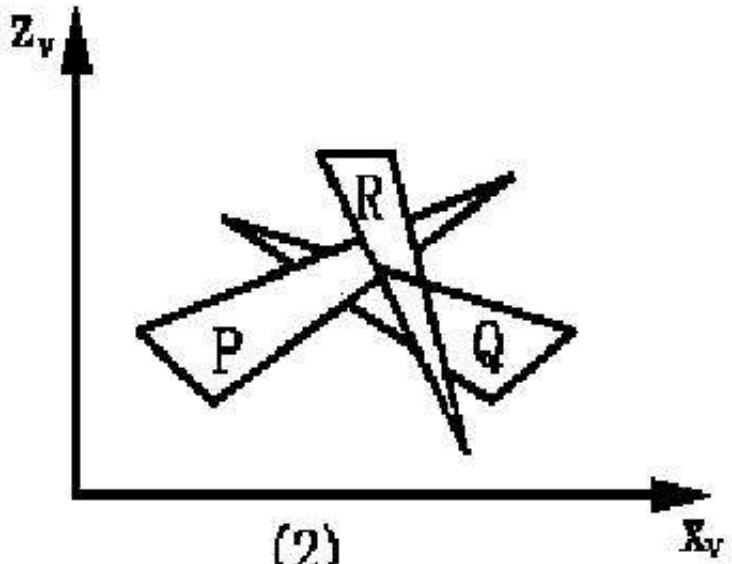
所有多边形按顶点最大z坐标值排序后得到一个排序表，设P是排在表中最后的那个多边形。

设Q是排在P前面并且z坐标范围与其发生交迭的一个多边形，对Q与P的次序关系进行检查。

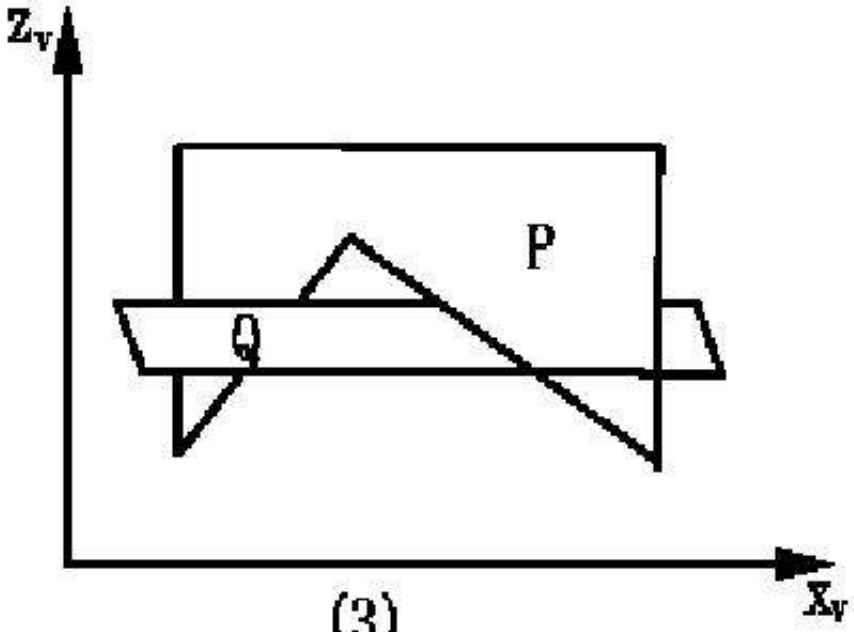
QP



(1)



(2)



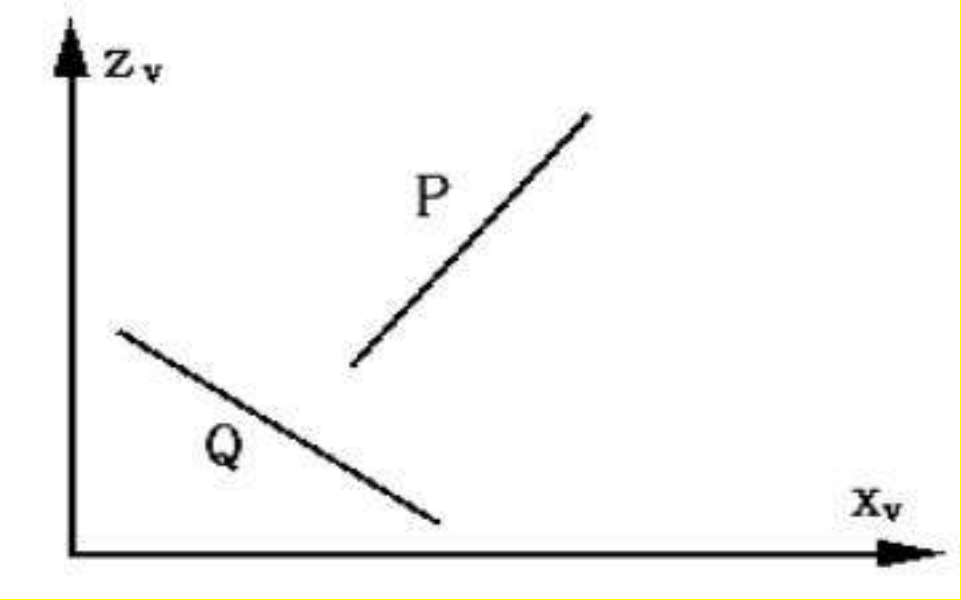
(3)

检查可以按下面列出的五个步骤进行，
每个步骤判断一种情况。

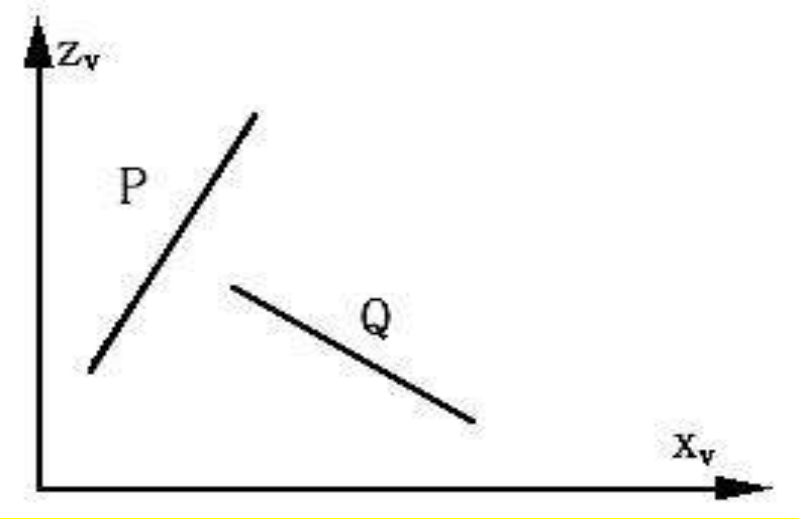
1．多边形的x坐标范围不相交迭，所以多边形不相交迭。QP

2．多边形的y坐标范围不相交迭，所以多边形不相交迭。QP

3．P整个在Q远离观察点的一侧。QP



4. Q整个在P的靠近观察点的一侧。QP



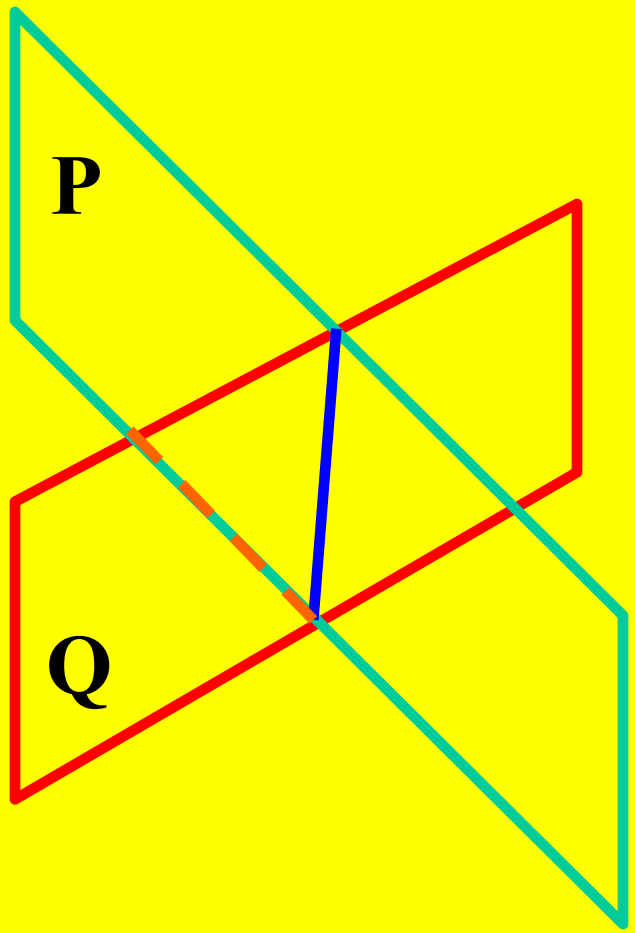
5. 多边形在z=0平面上的投影本身不相交迭。QP

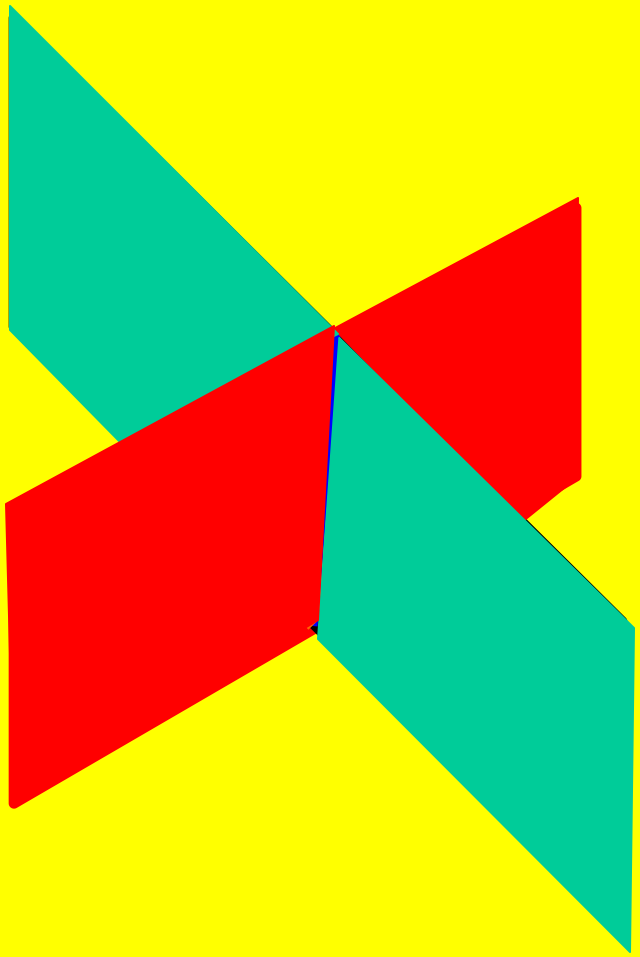
以上五步中任何一步成立，都可以说明当前排序正确

如果五步检查都为假，就假定P是遮挡了Q，交换P和Q在排序表中的位置。（QP→PQ）

如果仍做交换，算法会永远循环下去而没有结果。

为了避免循环，可以做一个限制。当做过首次五步检查后，发生某个多边形被移到排序表的末尾时，就立即加上一个标记，以后就不能再做移动。出现再次应该移动时，用一个多边形所在的平面，把另一个多边形裁剪分为两个。


P
Q



第五节 z-缓冲算法

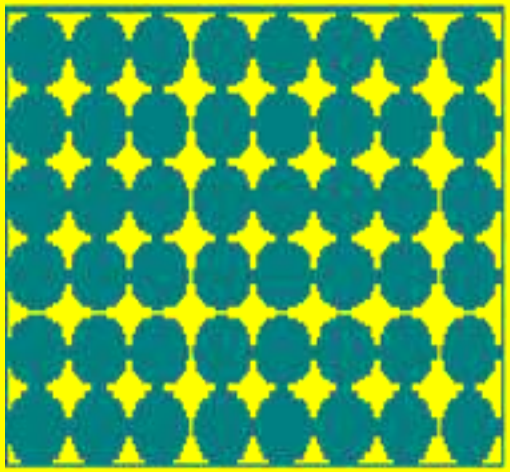
z-缓冲算法:(深度缓冲算法)是一种最简单的图象空间算法。

帧缓冲存储器:存储各点的象素值，初始化为背景值。

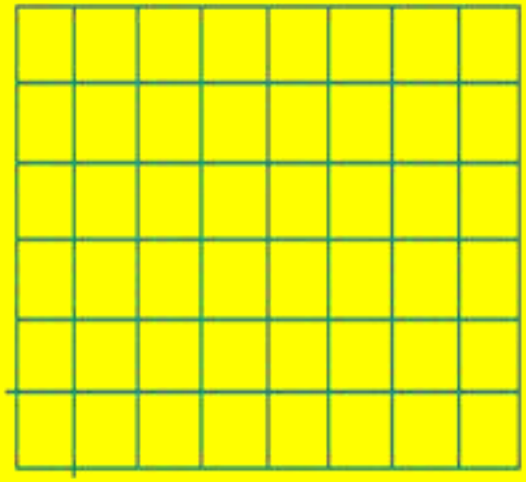
z-缓冲存储器:存储相应的z值。初始化最大z值。

对每一个多边形，不必进行深度排序算法要求的初始排序，立即就可以逐个进行扫描转换。

屏幕

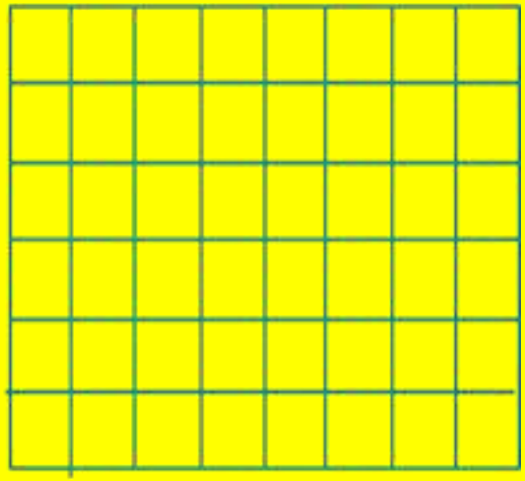


帧缓冲器



每个单元存放对应象素的颜色值

Z缓冲器



每个单元存放对应象素的深度值

扫描转换时，对每个多边形内部的任意点（x, y），实施如下步骤：

1. 计算在点（x，y）处多边形的深度值z（x，y）。

2. 如果计算所得的z（x，y）值，小于在z–缓冲存储器中点(x,y)处记录的深度值，那么就做：

（1）把值z(x，y)送入z–缓冲存储器的点(x, y)处。

（2）把多边形在深度z(x,y)处应有的象素值，送入帧缓冲存储器的点(x，y)处。

算法中深度计算，可通过多边形的顶点坐标求出所在平面的方程，然后再使用平面方程，对每个点(x,y)，解出相应的z。

对面方程 $Ax + By + Cz + D = 0$，

解出 z 是：

$$Z = \frac{-D - Ax - By}{C}$$

设在点（x, y）处的深度值是z_1：

$$\frac{-D-Ax-By}{C}=z_1$$

则在点（x+△x, y）处的深度值就是

$$\frac{-D-A(x+\Delta x)-By}{C}=\frac{-D-Ax-By}{C}-\frac{A}{C}\Delta x$$

$$=z1-\frac{A}{C}\Delta x$$

z-缓冲算法的工作流程:

帧缓冲区置成背景色;

z-缓冲区置成最大z值;

for (各个多边形)

{ 扫描转换该多边形;

for(计算多边形所覆盖的每个象素（x,y）)

{ 计算多边形在该象素的深度值Z(x,y);

if(Z(x,y)小于Z缓冲区中的(x,y)处的值)

{ 把Z(x,y)存入Z缓冲区中的(x,y)处;

把多边形在(x,y)处的亮度值存入帧缓存区的(x,y)处; }

}

}

第六节 扫描线算法

消除隐藏面的扫描线算法：图象空间算法

第二章 第三节 多边形扫描转换算法 ＝》推广

填充算法：针对一个多边形做扫描转换

消隐算法：同时对多个多边形做扫描转换。

ET表（边表）中吊桶的内容：

1．与较小的y坐标对应的端点的x坐标xmin。
2．边的另一端点的较大的y坐标ymax。
3．x的增量Δx，它实际上是边的斜率的倒数，是从一条扫描线走到下一条扫描线时，按x方向递增的步长。
4．边所属多边形的标记。
5．指针 指向下一条边

PT表（多边形表）：记录多边形信息，一个记录对应一个多边形。

一个记录的内容：

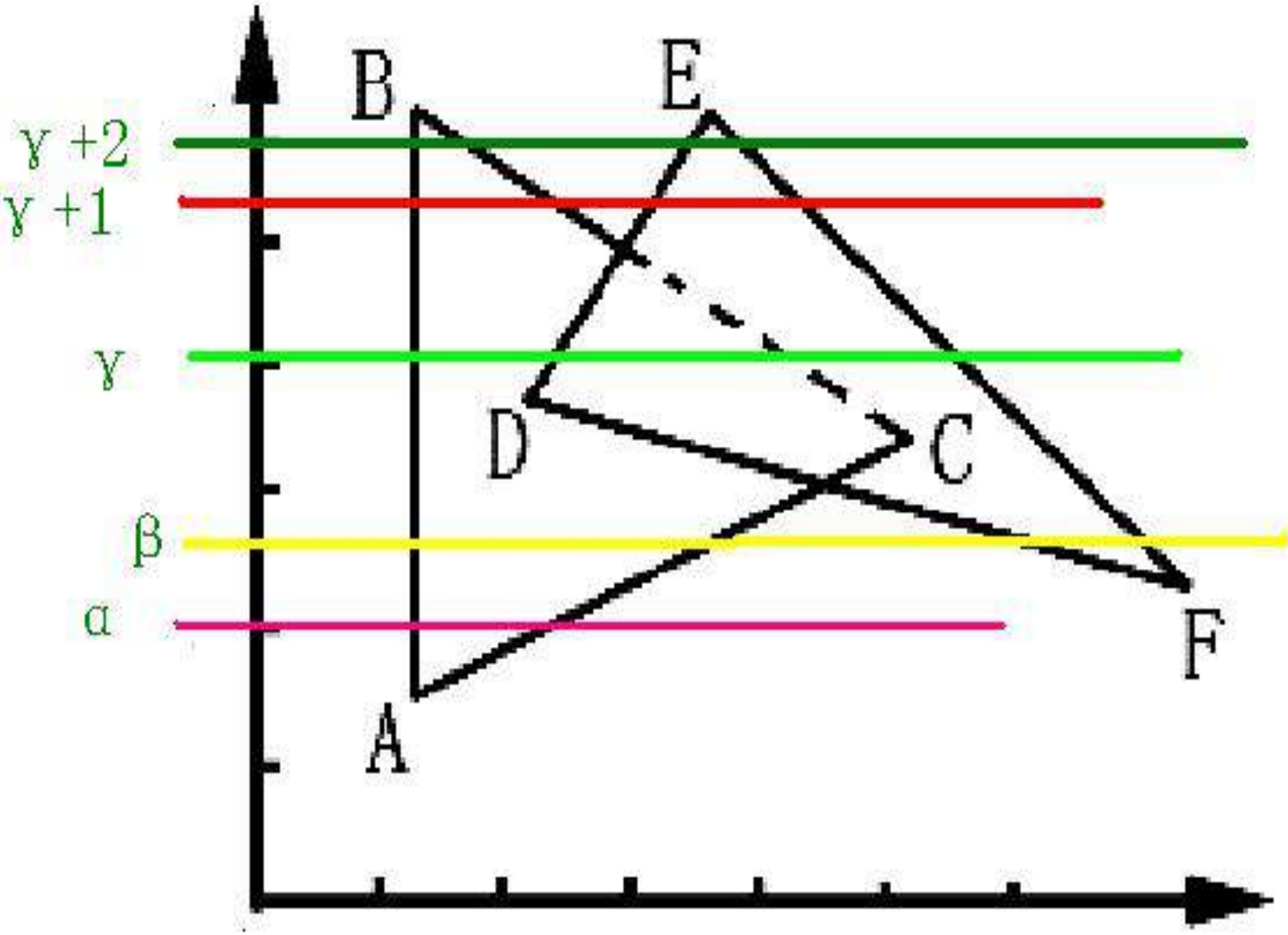
1．多边形所在平面方程的系数A,B,C,D

2．多边形的颜色或亮度值

3．进入/退出标志，标记当前扫描线对该多边形是“进入”还是“退出”

AET表（活跃边表）：存储与当前扫描线相交的各边的信息。

APT表（活跃多边形表）：与当前扫描线相交的所有多边形，按深度递增排序。


B
E
γ +2
γ +1
γ
D
C
β
α
F
A

扫描线y=α：

AET表中有AB,AC,

到AB时，$Flag_{ABC}$=1 ,AB到AC一段用ABC的颜色填充

到AC时，$Flag_{ABC}$=0,

扫描线y=β：

AET表中有四项AB,AC,FD,FEF,

到AB时，$Flag_{ABC}$=1 ,AB到AC一段用ABC的颜色填充

到AC时，$Flag_{ABC}$=0,AC到FD之间填充背景色

到FD时，$Flag_{DEF}$=1,FD到FE之间填充DEF颜色

到FE时，$Flag_{DEF}$=0

扫描线y=γ：

AET表中有四项AB,DE,CB,FE

到AB时，$Flag_{ABC}$=1 ，AB到DE之间填ABC的颜色

到DE时，$Flag_{DEF}$=1，做深度比较，求（x’，γ）的深度

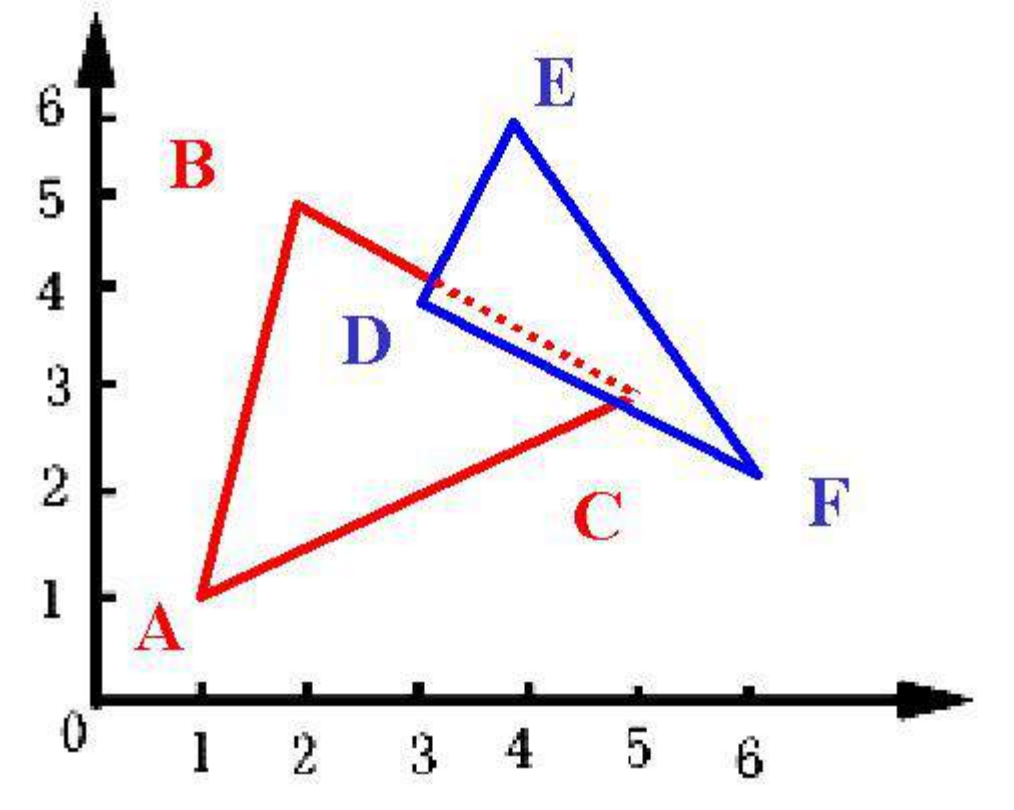
$$z_{ABC}=\frac{-A_1x'-B_1\gamma-D_1}{C_1},z_{DEF}=\frac{-A_2x'-B_2\gamma-D_2}{C_2}$$

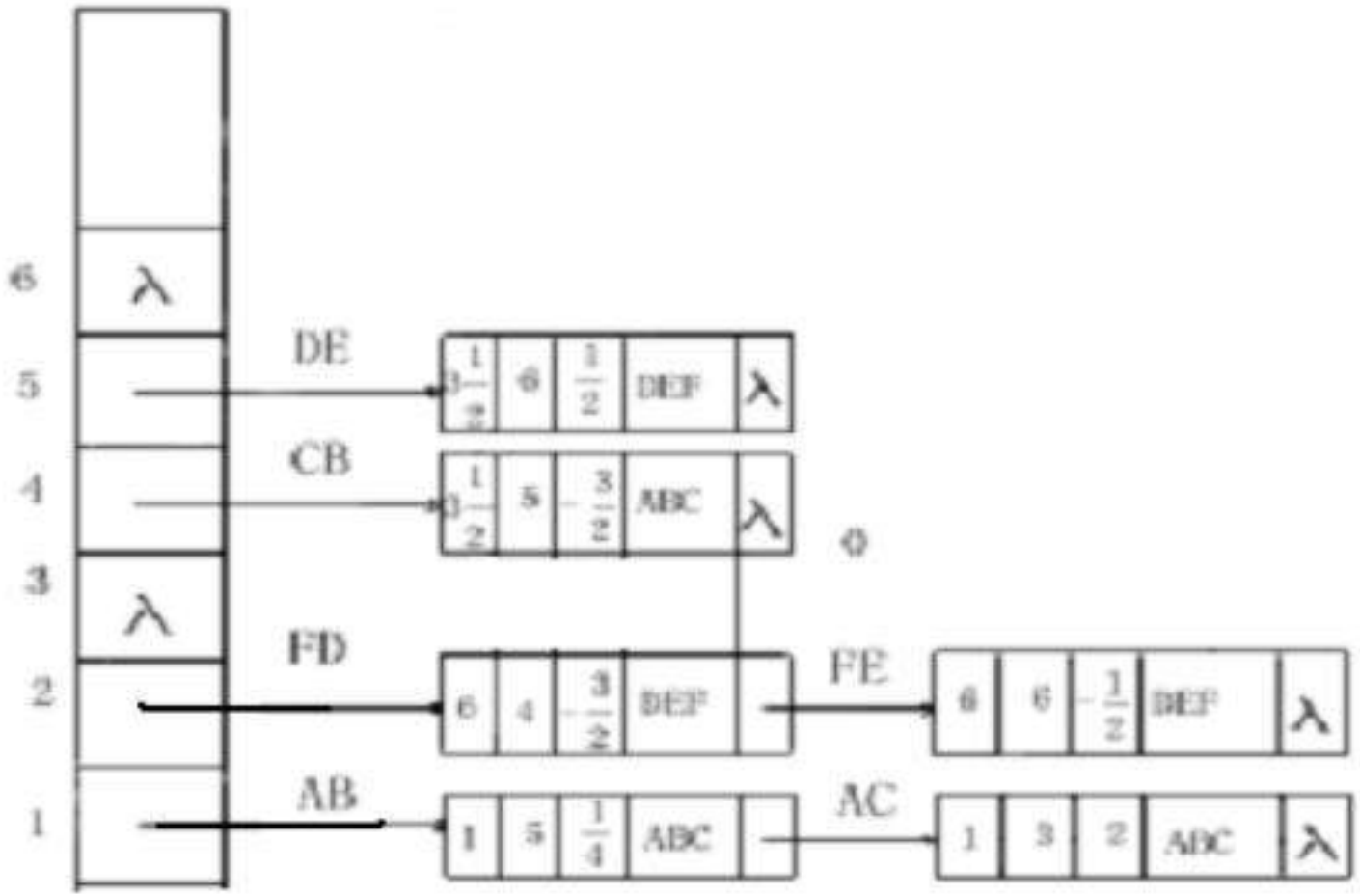
若$Z_{DEF}<Z_{ABC}$ ，则DE到CB之间填DEF的颜色

到CB时，$Flag_{ABC}$=0，BC到EF之间填DEF颜色

到FE时，$Flag_{DEF}$=0

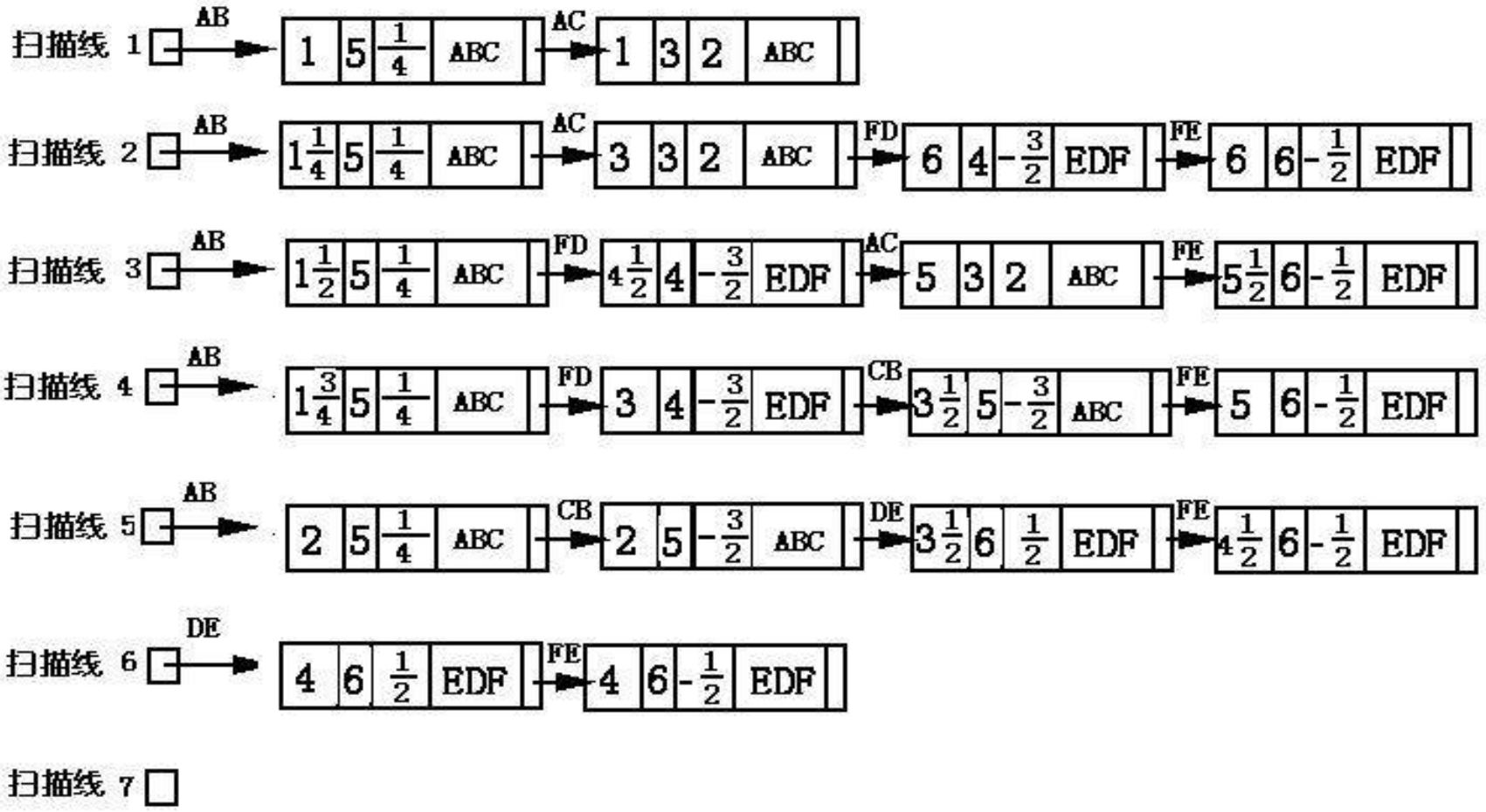
例子：设有两个空间的三角形ABC、DEF，各顶点的坐标依次是（1，1，10），（2，5，10），（5，3，10），（3，4，5），（4，6，5），（6，2，5）




6
λ
5
DE
3 1/2
6
1/2
DEF
λ
4
CB
3 1/2
5
3/2
ABC
λ
Φ
3
λ
2
FD
6
4
3/2
DEF
FE
6
6
1/2
DEF
λ
1
AB
1
5
1/4
ABC
AC
1
3
2
ABC
λ

AET

初始化 □


扫描线 1
AB
1 5 1/4 ABC
AC
1 3 2 ABC
扫描线 2
AB
1 1/4 5 1/4 ABC
AC
3 3 2 ABC
FD
6 4 -3/2 EDF
FE
6 6 -1/2 EDF
扫描线 3
AB
1 1/2 5 1/4 ABC
FD
4 1/2 4 -3/2 EDF
AC
5 3 2 ABC
FE
5 1/2 6 -1/2 EDF
扫描线 4
AB
1 3/4 5 1/4 ABC
FD
3 4 -3/2 EDF
CB
3 1/2 5 -3/2 ABC
FE
5 6 -1/2 EDF
扫描线 5
AB
2 5 1/4 ABC
CB
2 5 -3/2 ABC
DE
3 1/2 6 1/2 EDF
FE
4 1/2 6 -1/2 EDF
扫描线 6
DE
4 6 1/2 EDF
FE
4 6 -1/2 EDF
扫描线 7 □

算法步骤：

第二章第三节 多边形扫描转换算法

3.2:在扫描线y上，按照AET表提供的x坐标对，用color实施填充修改加细如下：

3.2.1:

扫描线指向的“吊桶”的指针i←1，

扫描线在多少个多边形内的计数器s←0

活跃多边形表P，初始为空。

3.2.2:

设第i个“吊桶”记录的相应多边形是A。

if $Flag_A$=FALSE，

then {$Flag_A$=TRUE，

将A加到表P中，s ← s+1}

else

{$Flag_A$= FALSE

将P中删除A， s ← s-1 }

3.2.3:

if s=0，goto 3.2.5（这时扫描线不在任何多边形内，正通过背景，不必做扫描转换）

if s=1，goto 3.2.4（这时扫描线只在一个多边形内，不必做深度比较，去做扫描转换。）

若前面两个判断都为“假”，扫描线至少在两个多边形内，应做深度比较。对表P前面两个多边形做深度比较，比较后排序应保证P表中的多边形按深度递增的次序。

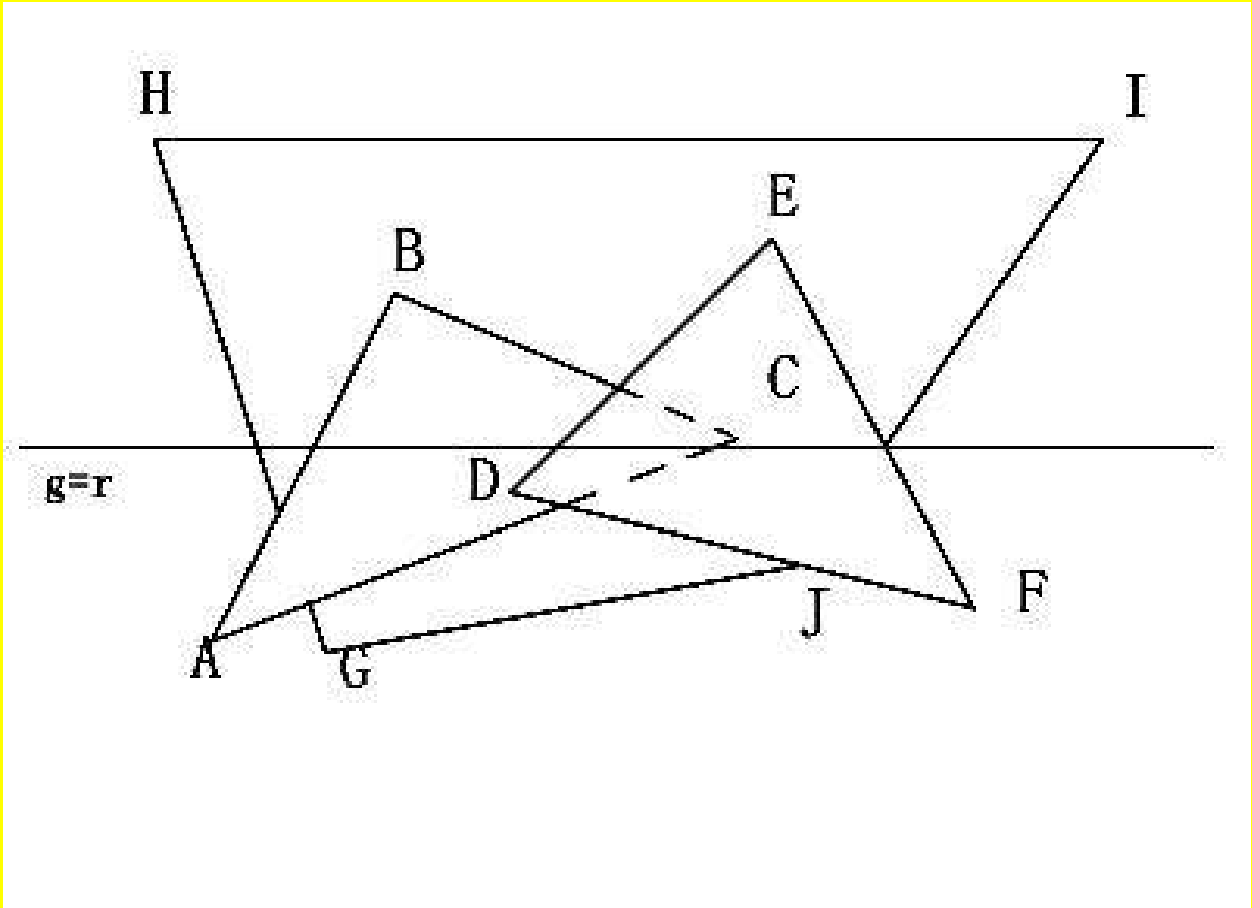
3.2.4：对第i个和第i+1个“吊桶”存有的x坐标指示的扫描线上的一段，按照P表最前面多边形指示的亮度或颜色，实施扫描转换。

3.2.5：i增加1，若i所指已无“吊桶”，步骤结束，去下一步骤3.3（删除y=ymax的边）。否则，回到步骤3.2.2。

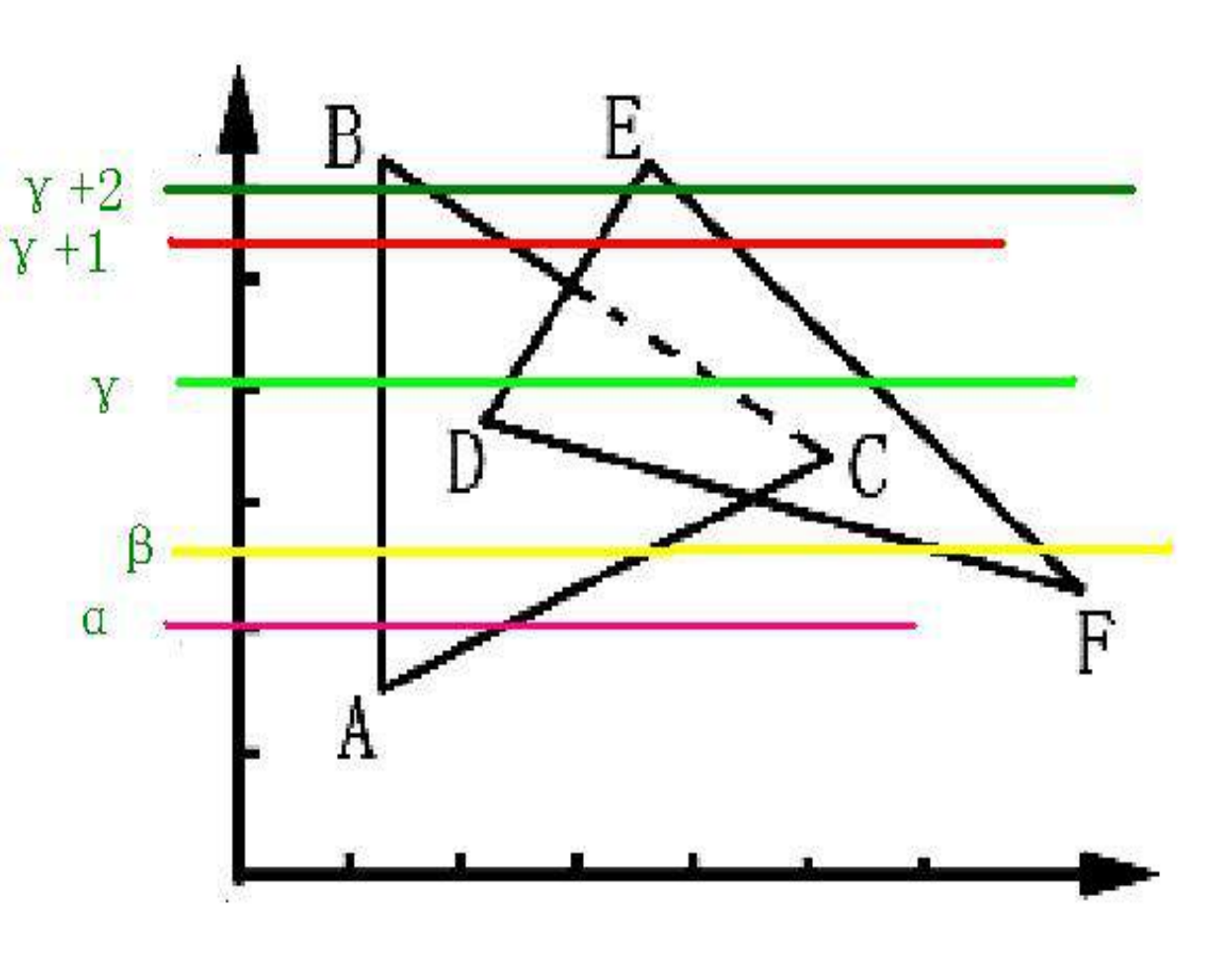
i	s	P	PT表 FABC FDEF		说明
1	1	(ABC)	true		1$\frac{3}{4}$ 到3填ABC的亮度或颜色值
2	2	(DEF, ABC)		true	在步骤3发生深度比较，比较结果DEF更靠近观察者，它仍在P表前面，3到3 $\frac{1}{2}$填DEF的亮度或颜色值
3	1	(DEF)	false		3$\frac{1}{2}$ 到5填DEF的亮度或颜色值
4	0	( )	false		

3.2.3深度比较条件太宽松，实际很多不必要的的比较

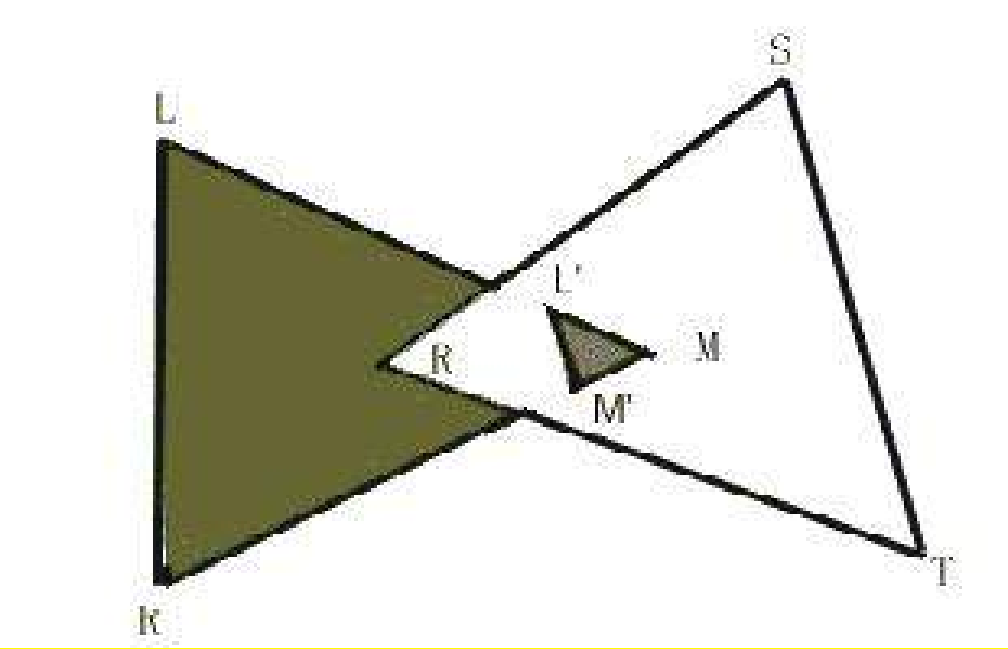
改进1：假定**没有多边形穿透另一个多边形的情况下**，**如果扫描线从一个*被遮挡*的多边形中走出，深度比较将是*不必要*的；扫描线从一个*遮挡*了其它多边形的多边形中走出，深度比较才可能*必要*。**



改进2：利用深度相关性，对于一组相邻的扫描线来说，多边形之间的深度关系常常是不发生变化的。



改进3：对于穿透情况的处理，将一个多边形分成两个子多边形



改进4：背景颜色处理

（1）帧缓存初始为某个特定的值

（2）定义一个包含了客体中所有的多边形，位于比其它多边形都更远离观察者的平行于投影平面的一个平面上，并具有某个合适的亮度或颜色值。

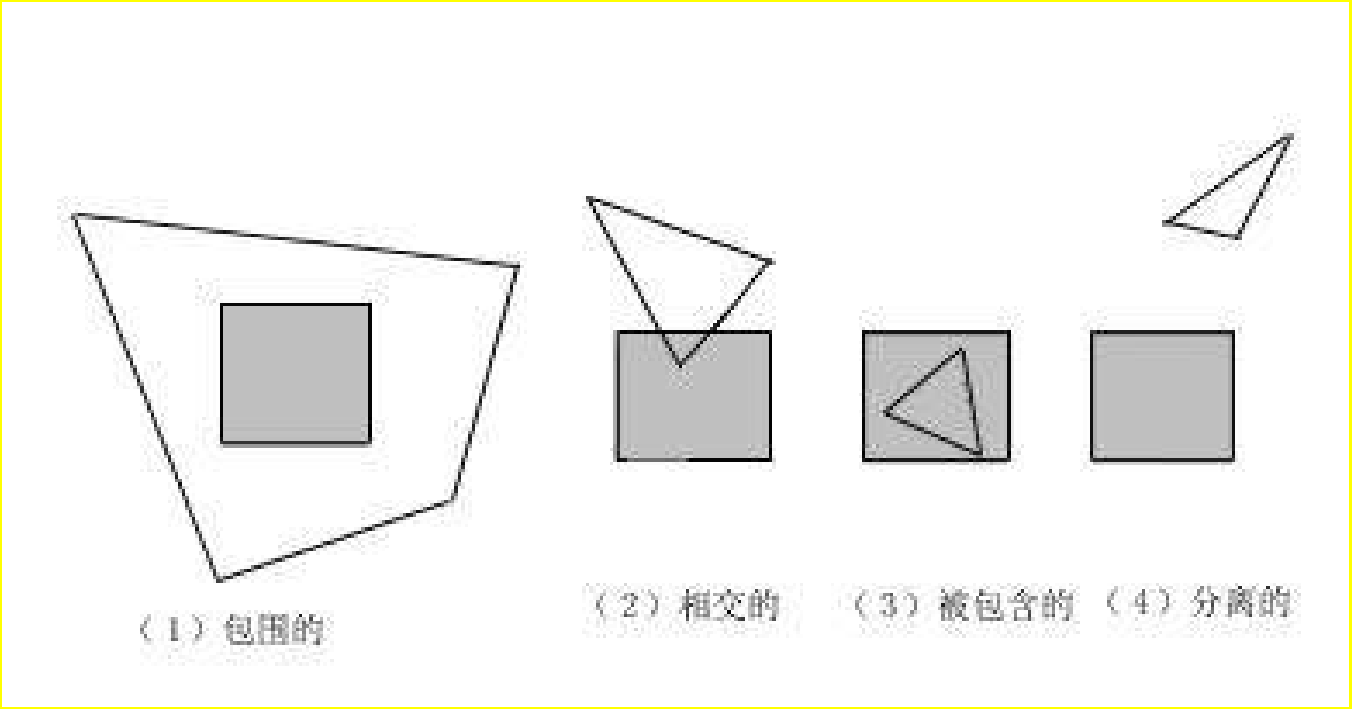
（3）修改算法。当扫描线不在任何多边形内时，就往帧缓冲存储器中送入背景的象素值。

第七节 区域分割算法(图像空间算法)

区域的相关性:是指位于适当大小的区域内的所有象素，表示的其实是同一个表面。

形体中多边形的**投影多边形**与**所考察区域**之间的**关系**：

1. **包围**的**多边形**
2. **相交**的多边形
3. **被包含**的多边形
4. **分离**的多边形

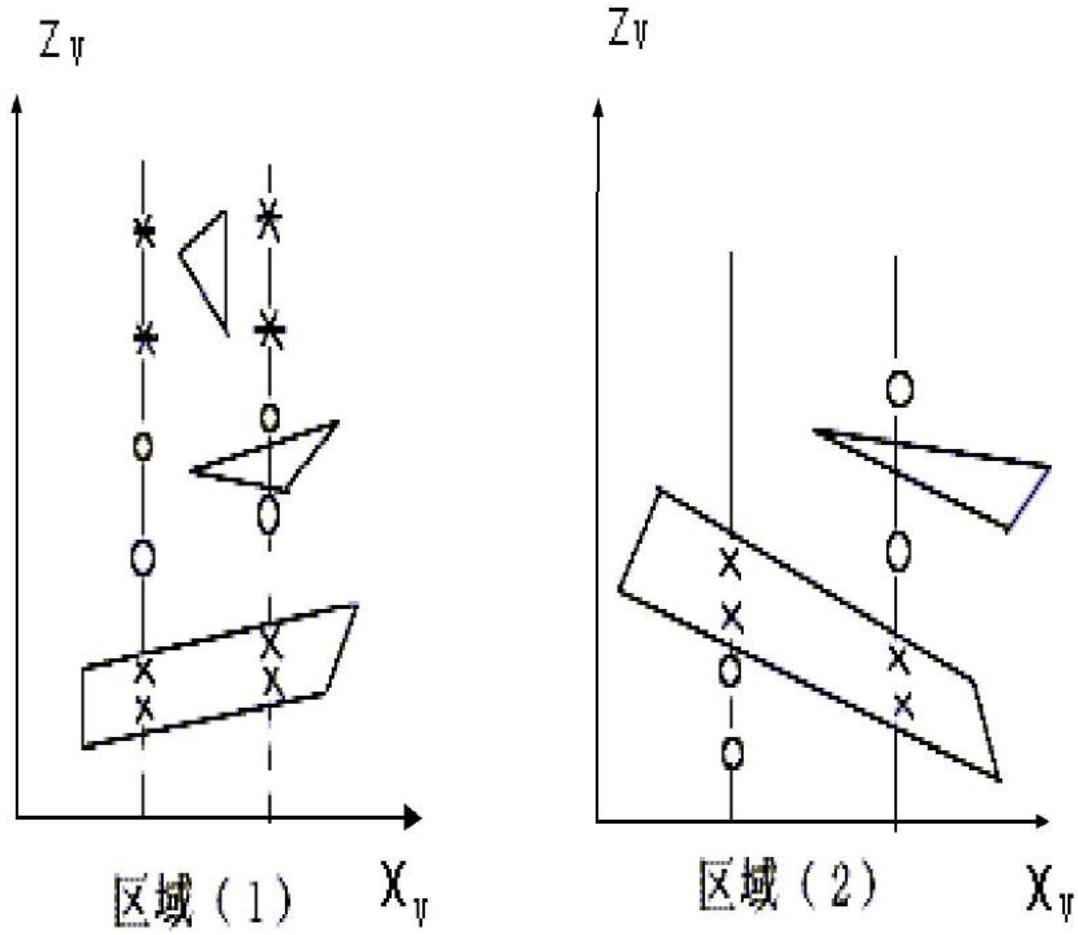


区域不需要分割的几种情况：

1. 所有的多边形与区域分离，所以在区域内只需显示背景值。
2. 只有一个相交的多边形，或者只有一个被包含的多边形。这时可以对区域首先填充背景值，然后对多边形进行扫描转换。（相交的多边形，只对被包含的部分做扫描转换）
3. 只有一个包围的多边形，无其它的多边形。整个区域填充该多边形的象素值。

（1, 2, 3用范围检查）

4. 多个多边形与之相交、包围、或被包含，且有一个包围的多边形位于其它多边形最前面，区域填充为该多边形的颜色。（深度检查）


Zv
区域（1）
Xv
Zv
区域（2）
Xv
X--表示包围的多边形平面交点
*--表示被包围的多边形平面交点
O--表示相交的多边形平面交点

情形4的两种情形

区域分割成子区域：

1. 子区域需考察的情况：

只需考虑父区域包含的或相交的多边形。

对于分离的或包围的多边形，进行区域分割之后仍保持分离包围的关系。

2. 分割中止的条件：一个象素单位或最大数目的分割

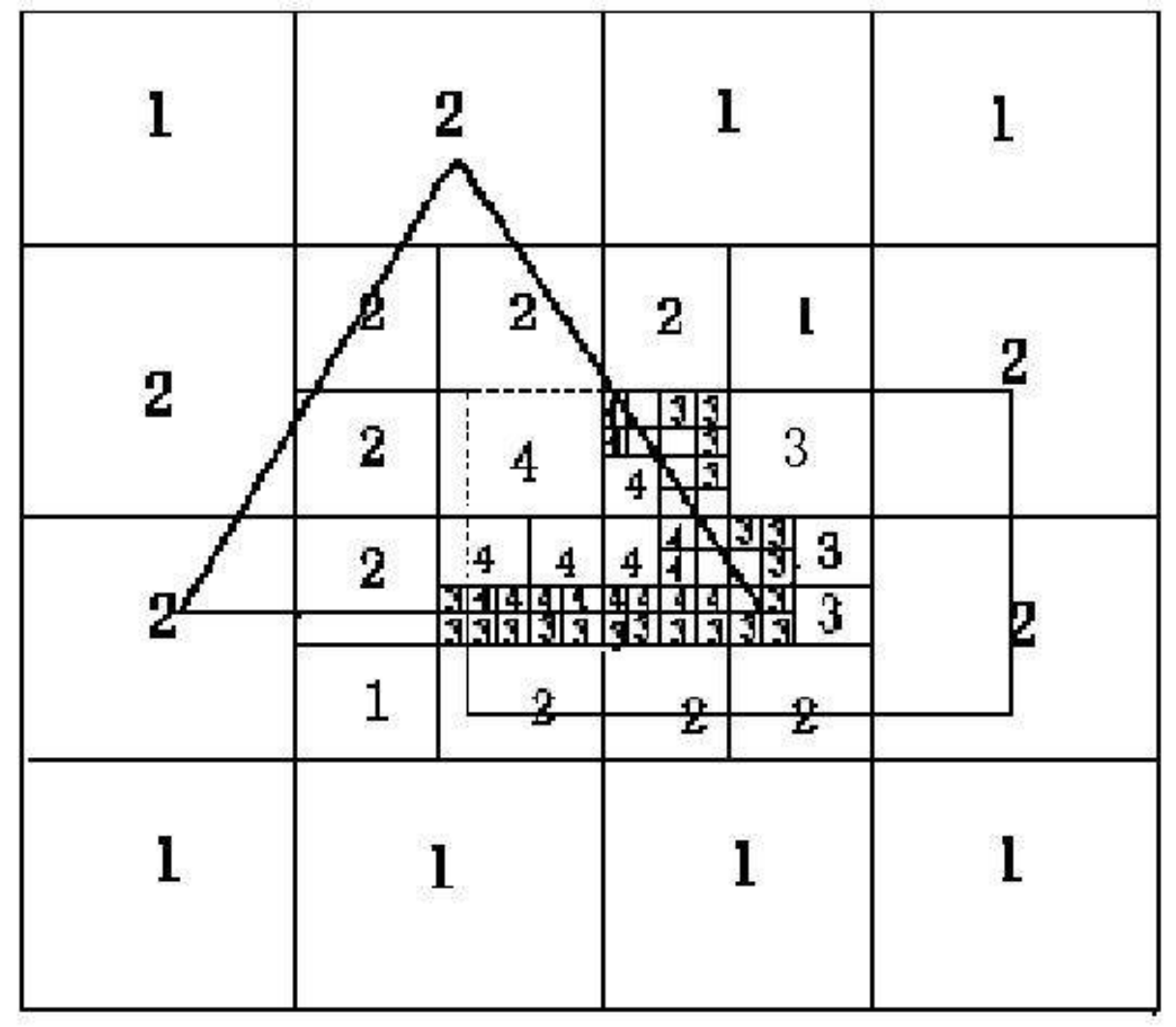
3. 分割方法

（1）等分

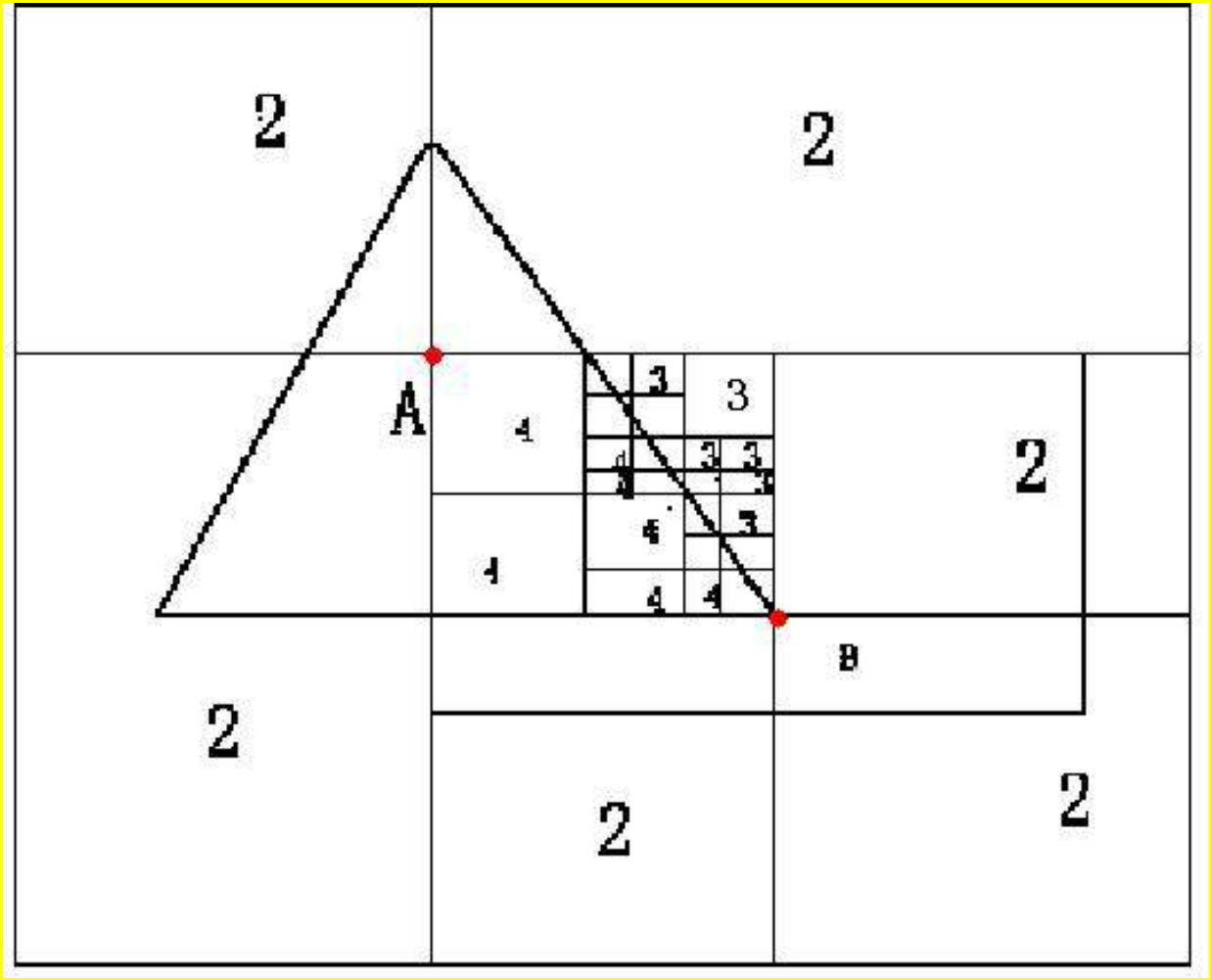
（2）按多边形顶点做分割

（3）按多边形投影进行分割

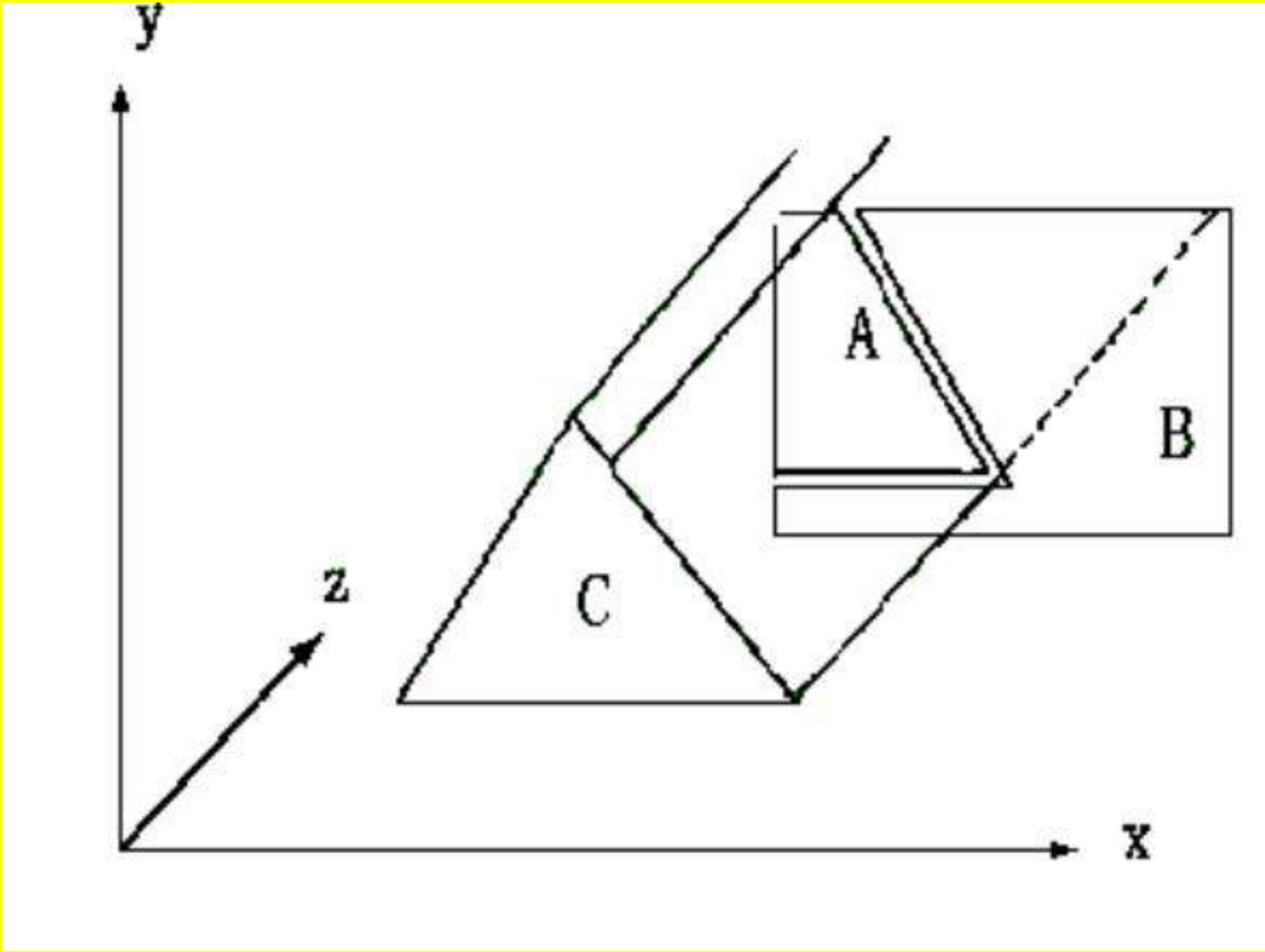
（1）等分：分割区域为正方形



（2）按照顶点位置来做分割(先A后B)



（3）按照多边形投影选择区域做分割



第一节　线面比较法消除隐藏线
夹角法
线面比较法（客体空间）

第三节　深度排序算法（客体空间＋图像空间）

第五节　Z一缓冲算法（图像空间）
第六节　扫描线算法（图像空间）
第七节　区域分割算法(图像空间）