第 4 章 图形变换

图形变换是计算机图形学的基础内容之一。一般来说，图形从输入到输出的过程中包含了各种变换。应用于对象几何描述并改变它的位置、方向或大小的操作称为几何变换。通过变换，可由简单图形生成复杂图形，可用二维图形表示三维形体，甚至可使静态图形经过快速变换而获得图形的动态显示效果。

本章主要介绍图形变换的数学基础，二维、三维图形的几何变换、投影变换以及图形的裁剪等知识。

4.1 矩阵表示和齐次坐标

在图形变换的过程中，要用到大量的向量、矩阵表示及其运算。本节将对这部分知识进行介绍。

4.1.1 向量及向量运算

在数学中，既有大小又有方向的量叫作向量。一个 n 维向量可以用 n 个实数组成的有序数组表示：$\boldsymbol{\alpha}=(a_1,a_2,a_i,\cdots,a_n)$，其中 a_i 称为向量 $\boldsymbol{\alpha}$ 的第 i 个分量。

设有向量 $\boldsymbol{a}(x_1,y_1,z_1)$，$\boldsymbol{b}(x_2,y_2,z_2)$，有关的向量运算有：

(1)向量的长度：$|\boldsymbol{a}| = \sqrt{a\cdot a} = \sqrt{x_1^2+y_1^2+z_1^2}$。

(2)两个向量的和差运算：$\boldsymbol{a}\pm\boldsymbol{b}=(x_1\pm x_2,y_1\pm y_2,z_1\pm z_2)$。

(3)两个向量的点乘积：$\boldsymbol{a}\cdot\boldsymbol{b}=x_1x_2+y_1y_2+z_1z_2$。

(4)两个向量的叉乘积：$\boldsymbol{a}\times\boldsymbol{b}=\begin{vmatrix} i & j & k \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix}=(y_1z_2-y_2z_1,z_1x_2-z_2x_1,x_1y_2-x_2y_1)$。

4.1.2 矩阵及矩阵运算

在数学上，矩阵是指纵横排列的二维数据表格。设有一个 m 行 n 列的矩阵 $\boldsymbol{A}$：

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

可以看出，这个矩阵是由 $m\times n$ 个数 $a_{ij}(i=1,2,\cdots,m;j=1,2,\cdots,n)$ 按一定位置排列的一个整体，简称 $m\times n$ 矩阵。矩阵通常用大写英文字母表示（如矩阵 $\boldsymbol{A}$）。其中，$a_{11},a_{12},\cdots,a_{1n}$ 称为矩阵 $\boldsymbol{A}$ 的行，$a_{11},a_{21},\cdots,a_{m1}$ 称为矩阵 $\boldsymbol{A}$ 的列，a_{ij} 称为 $\boldsymbol{A}$ 的第 i 行第 j 列元素。

当 $m=1$ 时，矩阵只有一行，即 $(a_{11},a_{12},\cdots,a_{1n})$，这样的矩阵称为行矩阵或行向量。

当 $n=1$ 时，矩阵只有一列，即 $\begin{bmatrix} a_{11} \\ a_{21} \\ \cdots \\ a_{m1} \end{bmatrix}$，这样的矩阵称为列矩阵或列向量。

如果两个矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的行数、列数都相等，并且矩阵中对应位置的元素的值全部相等，那么我们说 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 是相等的。

1）矩阵的运算

（1）数乘矩阵。设有 m 行 n 列的矩阵 $\boldsymbol{A}$，用数 t 乘矩阵 $\boldsymbol{A}$ 的每一个元素而得的矩阵叫做与数 t 的乘积，记为 $t\boldsymbol{A}$ 或 $\boldsymbol{A}t$：

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

（2）矩阵的加法运算。设有矩阵 $\boldsymbol{A}$、$\boldsymbol{B}$ 都是 m 行 n 列的矩阵，将它们对应位置的元素相加得到的矩阵叫做 $\boldsymbol{A}$ 与 $\boldsymbol{B}$ 的和，记为 $\boldsymbol{A}+\boldsymbol{B}$：

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

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

$$\boldsymbol{C}=\boldsymbol{A}\cdot\boldsymbol{B}=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

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

需要注意的是，只有在两个矩阵中前一个矩阵的列数等于后一个矩阵的行数时，这两个矩阵才能相乘，得到的结果矩阵的行数等于前一个矩阵的行数，列数等于后一个矩阵的列数，即

$$\boldsymbol{C}=(\boldsymbol{C}_{ij})_{m\times p}=\boldsymbol{A}_{m\times n}\cdot\boldsymbol{B}_{n\times p}$$

(4)有关零矩阵的运算。矩阵的所有元素均为零的矩阵称为零矩阵。一个 m 行 n 列的零矩阵记为 $0_{m\times n}$。对于任意矩阵 $\boldsymbol{A}_{m\times n}$ 都有下式成立：

$$\boldsymbol{A}_{m\times n}+0_{m\times n}=\boldsymbol{A}_{m\times n}$$

(5)单位矩阵。在 $n\times n$ 矩阵中，如其主对角线上各元素 $a_{ii}=1$，其余各元素均为零，则该矩阵称为单位矩阵，如下：

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

n 阶的单位矩阵记为 $\boldsymbol{I}_n$。

对于任意的矩阵 $\boldsymbol{A}_{m\times n}$，都有

$$\boldsymbol{A}_{m\times n}\cdot\boldsymbol{I}_n=\boldsymbol{A}_{m\times n}$$
$$\boldsymbol{I}_n\cdot\boldsymbol{A}_{m\times n}=\boldsymbol{A}_{m\times n}$$

成立。

(6)逆矩阵。对任意矩阵 $\boldsymbol{A}$，如果存在 $\boldsymbol{A}\cdot\boldsymbol{A}^{-1}=\boldsymbol{A}^{-1}\cdot\boldsymbol{A}=\boldsymbol{I}$，则称 $\boldsymbol{A}^{-1}$ 为 $\boldsymbol{A}$ 的逆矩阵。

设 $\boldsymbol{A}$ 是一个 n 阶矩阵，如果有 n 阶矩阵 $\boldsymbol{B}$ 存在，使得

$$\boldsymbol{A}\cdot\boldsymbol{B}=\boldsymbol{B}\cdot\boldsymbol{A}=\boldsymbol{I}$$

则称 $\boldsymbol{A}$ 是一个非奇异矩阵，并且 $\boldsymbol{B}$ 是 $\boldsymbol{A}$ 的逆矩阵。否则，称 $\boldsymbol{A}$ 是一个奇异矩阵。由于 $\boldsymbol{A}$、$\boldsymbol{B}$ 处于对称地位，所以如果矩阵 $\boldsymbol{A}$ 存在逆矩阵 $\boldsymbol{B}$，则 $\boldsymbol{A}$ 也是 $\boldsymbol{B}$ 的逆矩阵，矩阵 $\boldsymbol{B}$ 也是一个非奇异矩阵。任何非奇异矩阵的逆矩阵是唯一的。

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

矩阵的转置具有如下几个基本性质：

$$(\boldsymbol{A}^{\mathrm{T}})^{\mathrm{T}}=\boldsymbol{A}$$
$$(\boldsymbol{A}+\boldsymbol{B})^{\mathrm{T}}=\boldsymbol{A}^{\mathrm{T}}+\boldsymbol{B}^{\mathrm{T}}$$
$$(t\boldsymbol{A})^{\mathrm{T}}=t\boldsymbol{A}^{\mathrm{T}}$$
$$(\boldsymbol{A}\cdot\boldsymbol{B})^{\mathrm{T}}=\boldsymbol{B}^{\mathrm{T}}\cdot\boldsymbol{A}^{\mathrm{T}}$$

其中，$\boldsymbol{A}$、$\boldsymbol{B}$ 为矩阵，t 为常数。

当 $\boldsymbol{A}$ 是一个 n 阶矩阵并且有 $\boldsymbol{A}=\boldsymbol{A}^{\mathrm{T}}$ 时，则称 $\boldsymbol{A}$ 是一个对称矩阵。

2)矩阵运算的基本性质

(1)数乘矩阵适合分配律和结合律，即

$$t(\boldsymbol{A}+\boldsymbol{B})=t\boldsymbol{A}+t\boldsymbol{B}$$
$$t(\boldsymbol{A}\cdot\boldsymbol{B})=(t\cdot\boldsymbol{A})\cdot\boldsymbol{B}=\boldsymbol{A}\cdot(t\cdot\boldsymbol{B})$$
$$(t_1+t_2)\cdot\boldsymbol{A}=t_1\cdot\boldsymbol{A}+t_2\cdot\boldsymbol{A}$$
$$t_1(t_2\boldsymbol{A})=(t_1t_2)\boldsymbol{A}$$

其中 $\boldsymbol{A}$、$\boldsymbol{B}$ 为矩阵，t、t_1、t_2 为常数。

(2)矩阵的加法适合交换律和结合律，即

$$\boldsymbol{A}+\boldsymbol{B}=\boldsymbol{B}+\boldsymbol{A}$$

$$\boldsymbol{A}+(\boldsymbol{B}+\boldsymbol{C})=(\boldsymbol{A}+\boldsymbol{B})+\boldsymbol{C}$$

其中，$\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$ 为矩阵。

(3)矩阵的乘法适合结合律，即

$$\boldsymbol{A}\cdot(\boldsymbol{B}\cdot\boldsymbol{C})=(\boldsymbol{A}\cdot\boldsymbol{B})\cdot\boldsymbol{C}$$

(4)矩阵的乘法对加法适合分配律，即

$$(\boldsymbol{A}+\boldsymbol{B})\cdot\boldsymbol{C}=\boldsymbol{A}\cdot\boldsymbol{C}+\boldsymbol{B}\cdot\boldsymbol{C}$$

$$\boldsymbol{C}\cdot(\boldsymbol{A}+\boldsymbol{B})=\boldsymbol{C}\cdot\boldsymbol{A}+\boldsymbol{C}\cdot\boldsymbol{B}$$

(5)矩阵的乘法不适合交换律，因为当两个矩阵 $\boldsymbol{A}$、$\boldsymbol{B}$ 能够相乘时，如果它们不是方阵，则 $\boldsymbol{B}$、$\boldsymbol{A}$ 不能相乘。即使 $\boldsymbol{A}$、$\boldsymbol{B}$ 均为方阵，在一般情况下 $\boldsymbol{A}\cdot\boldsymbol{B}$ 和 $\boldsymbol{B}\cdot\boldsymbol{A}$ 也不相等。

4.1.3 齐次坐标

我们知道，在二维平面中点的坐标通常用二维向量(x,y)表示。但是为了实际运算的需要，往往需要将二维点 $P(x,y)$用三维坐标向量来表示。通用的实现方法是将二维坐标表示(x,y)扩充到三维表示$[x_h \quad y_h \quad h]$，称为齐次坐标，其中 h 称为齐次参数，是一个非零值。(x,y)和$[x_h \quad y_h \quad h]$满足：

$$x=\frac{x_h}{h},y=\frac{y_h}{h}$$

也就是说，普通二维坐标(x,y)可以表示成$[hx \quad hy \quad h]$的齐次坐标形式。为了运算方便，通常简单地将 h 取值为 1，这样(x,y)对应的齐次坐标就是$[x \quad y \quad 1]$。

齐次坐标这一术语在数学中用来指出笛卡尔方程的表达效果。二维点(x,y)转换成三维齐次坐标$[hx \quad hy \quad h]$，由于 h 可以取非零的任何值，代表的是一条直线。

同样可得出，三维空间中的坐标点(x,y,z)的齐次坐标可表示为$[hx \quad hy \quad hz \quad h]$。为了便于计算，使 $h=1$。

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。使用齐次坐标和利用矩阵可以统一处理各种变换。例如，二维齐次坐标变换矩阵的形式为：

$$T_{2D}=\begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$$

三维齐次坐标变换矩阵的形式为：

$$T_{3D}=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

在投影变换时，齐次坐标也有很重要的应用。

4.2　基本二维几何变换

图形变换通常是指对构成图形的每个点分别做相应的变换，从而构成新的图形。但事实上，很多情况下并不需要对图形中的所有点进行几何变换，而只需要对图形中的一些特殊点做变换，通过变换后的点即可求得变换后的完整图形。本节将介绍有关二维图形变换的知识。

二维图形变换主要是针对平面图形，常见的二维图形几何变换有平移变换、比例变换和旋转变换等。

4.2.1　平移变换

设平面上一点 $P(x,y)$，将 P 点沿 x 轴方向移动 T_x 距离，沿 y 轴方向移动 T_y 距离，得到一个新点 $P'(x',y')$，如图 4-1 所示。也就是说，在原始坐标 (x,y) 上加一个平移距离 t_x 和 t_y，以获得一个新的坐标位置 (x',y')，可以实现一个二维位置的平移：

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

如果分别用向量表示原始点、目标点和平移距离：

$$P=\begin{bmatrix}x\\y\end{bmatrix},P'=\begin{bmatrix}x'\\y'\end{bmatrix},T=\begin{bmatrix}t_x\\t_y\end{bmatrix}$$

则有二维平移方程：

$$P'=P+T \tag{4-1}$$

平移是一种移动对象而不改变其形状的刚体变换，即对象上的每一点移动了同样的距离。一条线段的平移可以通过使用公式(4-1)分别对其两个端点进行移动，得到新的端点，在新的端点间连线来实现。多边形的平移也可以通过分别移动多边形的每个顶点来实现。如图 4-1 所示为四边形的平移。

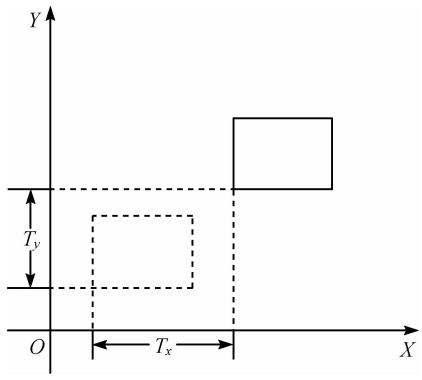

图 4-1　四边形的平移

4.2.2 比例变换

改变一个图形的大小，可以通过缩放来实现。简单的比例变换是相对于原点的，将平面上的一点(x,y)分别沿 X 轴、Y 轴方向进行缩放，也就是分别乘以缩放系数 S_x、S_y 后，得到点(x',y')。即

$$x'=x\cdot S_x,y'=y\cdot S_y$$

基本的二维缩放方程也可以用矩阵形式表示如下：

$$\begin{bmatrix}x'\\y'\end{bmatrix}=\begin{bmatrix}s_x & 0\\0 & s_y\end{bmatrix}\cdot\begin{bmatrix}x\\y\end{bmatrix}$$

或

$$P'=S\cdot P \tag{4-2}$$

其中，S 是等式中的 2×2 缩放矩阵。

图 4-2(a)、(b)、(c)、(d)分别是原图形、比例系数相等且大于 1、比例系数相等且小于 1 以及比例系数不等的比例变换结果。可见，如果 $S_x=S_y$，则变换后的图形与原图形保持相对比例一致；当 $S_x=S_y>1$ 时，图形被放大了，到原点的距离也变大了；当 $S_x=S_y<1$ 时，图形被缩小了，到原点的距离也变小了；如果 $S_x\neq S_y$，则图形在 X 轴、Y 轴两个方向上有不同的拉长或缩短。特别地，当 $S_x=S_y=1$ 时，图形保持不变。

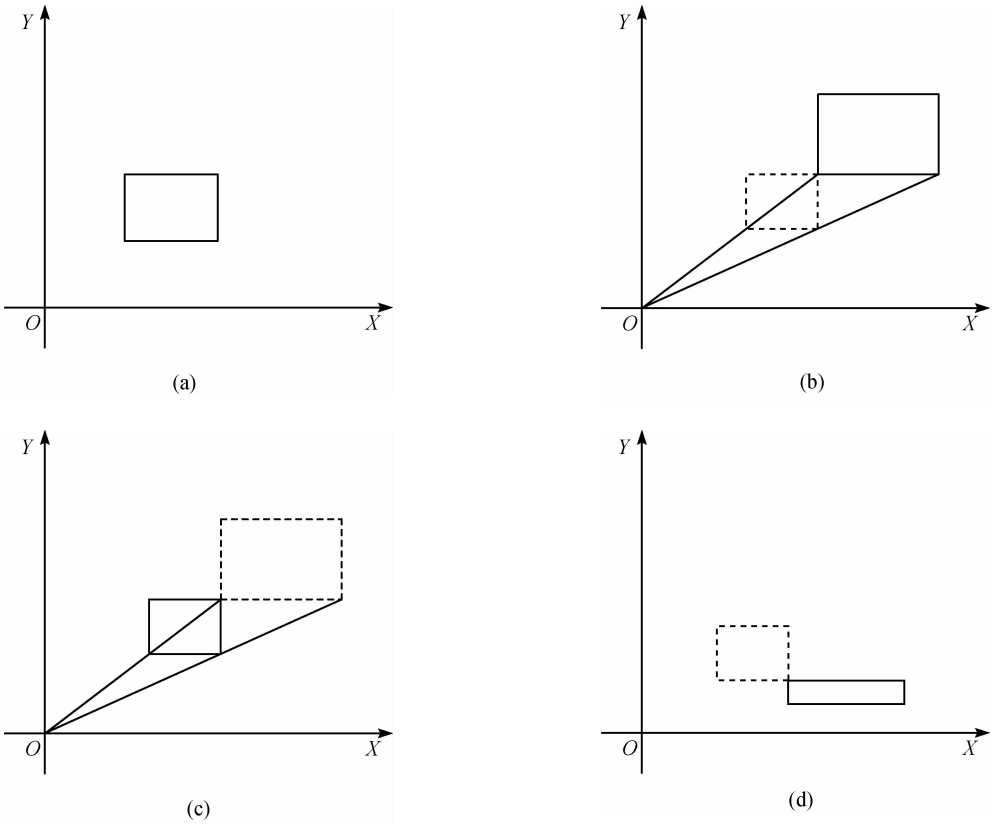

图 4-2 四边形的比例变换

(a)变换前；(b)变换后 $S_x=S_y>1$；(c)变换后 $S_x=S_y<1$；(d)变换后 $S_x\neq S_y$

4.2.3　旋转变换

给出旋转中心和旋转角度，将图形围绕旋转中心逆时针旋转给定的角度，就可以实现图形的旋转变换。最简单的旋转变换是以坐标原点(0,0)为中心的旋转。将平面上的一点(x,y)绕(0,0)点旋转θ角度后，得到旋转后的点(x',y')，则有：

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

上式的矩阵形式可表示为：

$$P'=R\cdot P \tag{4-3}$$

其中，旋转矩阵为：

$$R=\begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}$$

图 4-3 是围绕坐标原点作旋转变换的示意图。

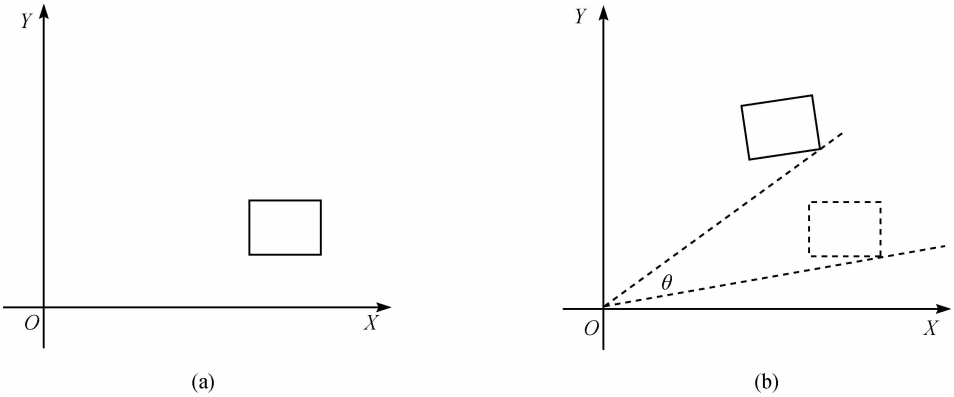

图 4-3　四边形绕坐标原点的旋转变换

(a)原图；(b)绕原点的旋转变换

而对于旋转中心(x_r,y_r)不为坐标原点的情况，可以先将(x_r,y_r)点平移$(-x_r,-y_r)$至坐标原点(0,0)，相应的原始点$P(x,y)$平移至点$P_1(x-x_r,y-y_r)$。现在，问题又变为绕坐标原点进行旋转。应用公式(4-3)，得到旋转之后的点$P_2((x-x_r)\cos\theta-(y-y_r)\sin\theta,(x-x_r)\sin\theta+(y-y_r)\cos\theta)$。需要注意的是，$P_2$点是旋转中心平移后的旋转结果，要得到$P$点绕旋转中心旋转的结果，还应该在$P_2$点的$x$和$y$坐标上分别加上平移量$(x_r,y_r)$。由此可以看出，最终结果$P'(x',y')$可以通过下面的公式计算得出：

$$x'=x_r+(x-x_r)\cos\theta-(y-y_r)\sin\theta$$
$$y'=y_r+(x-x_r)\sin\theta+(y-y_r)\cos\theta$$

4.2.4　齐次坐标表示二维图形变换

我们已经知道，每个二维图形的基本变换(平移、比例、旋转)都可以表示为普通矩阵形式：

$$P'=M_1\cdot P+M_2$$

其中，点 P' 和 P 表示为列向量，M_1 和 M_2 均为 2×2 矩阵，M_1 为比例和旋转变换矩阵，M_2 为平移矩阵。对于任何给定的基本变换，我们都可以依据上面的公式，代入相应的比例、旋转、平移矩阵，逐步计算求得最终结果。更有效的方法是将变换矩阵组合，直接从初始点经计算得到结果点，消除中间点的计算，也就是说，要消除 M_2 中与平移项相关的矩阵加法。

如果将 2×2 矩阵表达式扩充为 3×3 矩阵，就可以将二维几何变换的乘法和平移项组合成单一矩阵表示。也就是我们在本章开始介绍的齐次坐标。

1)平移变换

使用齐次坐标方法，点的二维平移可以表示为如下矩阵乘法：

$$P'=[x' \quad y' \quad 1]=[x \quad y \quad 1]\cdot\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ T_x & T_y & 1\end{bmatrix}$$

或

$$P'=P\cdot T(T_x,T_y)$$

上式中的 T_x、T_y 分别是图形在 X 轴和 Y 轴上的平移量。

2)比例变换

点的二维比例变换可以表示为如下矩阵乘法：

$$P'=[x' \quad y' \quad 1]=[x \quad y \quad 1]\cdot\begin{bmatrix}S_x & 0 & 0\\ 0 & S_y & 0\\ 0 & 0 & 1\end{bmatrix}$$

或

$$P'=P\cdot S(S_x,S_y)$$

上式中的 S_x、S_y 分别是图形在 X 轴和 Y 轴上的缩放比例。该比例变换是以坐标原点为参考点的。

3)旋转变换

点的二维旋转变换可以表示为如下矩阵乘法：

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

或

$$P'=P\cdot R(\theta)$$

上式中的 θ 是图形以坐标原点为旋转中心的旋转角度。

4)二维复合变换

上面分别介绍了平移、比例和以原点为中心的旋转变换。如果分别用 $T(T_x,T_y)$、$S(S_x,S_y)$、$R(\theta)$ 表示这三种基本变换，那么多数常见的二维几何变换都可以通过这三种基本变换的组合来实现。

假设进行二维图形几何变换前图形上一点的坐标为 $P=[x \quad y \quad 1]$，变换后的坐标为

$P'=[x'\quad y'\quad 1]$，可以先将变换中心点平移到原点，相对于原点作变换后，再将变换中心点平移回原来的位置。相对于任意一点(x_0,y_0)的比例变换和旋转变换矩阵如下：

比例变换矩阵：

$$T(-x_0,-y_0)\cdot S(S_x,S_y)\cdot T(x_0,y_0)$$

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

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

旋转变换矩阵：

$$T(-x_0,y_0)\cdot R(\theta)\cdot T(x_0,y_0)$$

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

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

4.3　其他二维变换

除了上面介绍的平移、比例和旋转变换，在二维图形几何变换中，对称变换和错切变换也比较常见。

4.3.1　对称变换

$$P'=[x'\quad y'\quad 1]=[x\quad y\quad 1]\cdot\begin{bmatrix}a&d&0\\b&e&0\\0&0&1\end{bmatrix}=[ax+by\quad dx+ey\quad 1]$$

当a、b、d、e取不同的值时将产生不同的对称变换。

(1)当$b=d=0$，$a=-1$，$e=1$时，有$x'=-x$，$y'=y$，产生相对于Y轴对称的反射图形。

(2)当$b=d=0$，$a=1$，$e=-1$时，有$x'=x$，$y'=-y$，产生相对于X轴对称的反射图形。

(3)当$b=d=0$，$a=e=-1$时，有$x'=-x$，$y'=-y$，产生相对于原点对称的反射图形。

(4)当$b=d=1$，$a=e=0$时，有$x'=y$，$y'=x$，产生相对于直线$y=x$对称的反射图形。

(5)当$b=d=-1$，$a=e=0$时，有$x'=-y$，$y'=-x$，产生相对于直线$y=-x$对称的反射图形。

图 4-4 是对称变换示意图。对称轴分别为坐标轴、坐标原点以及其他直线。

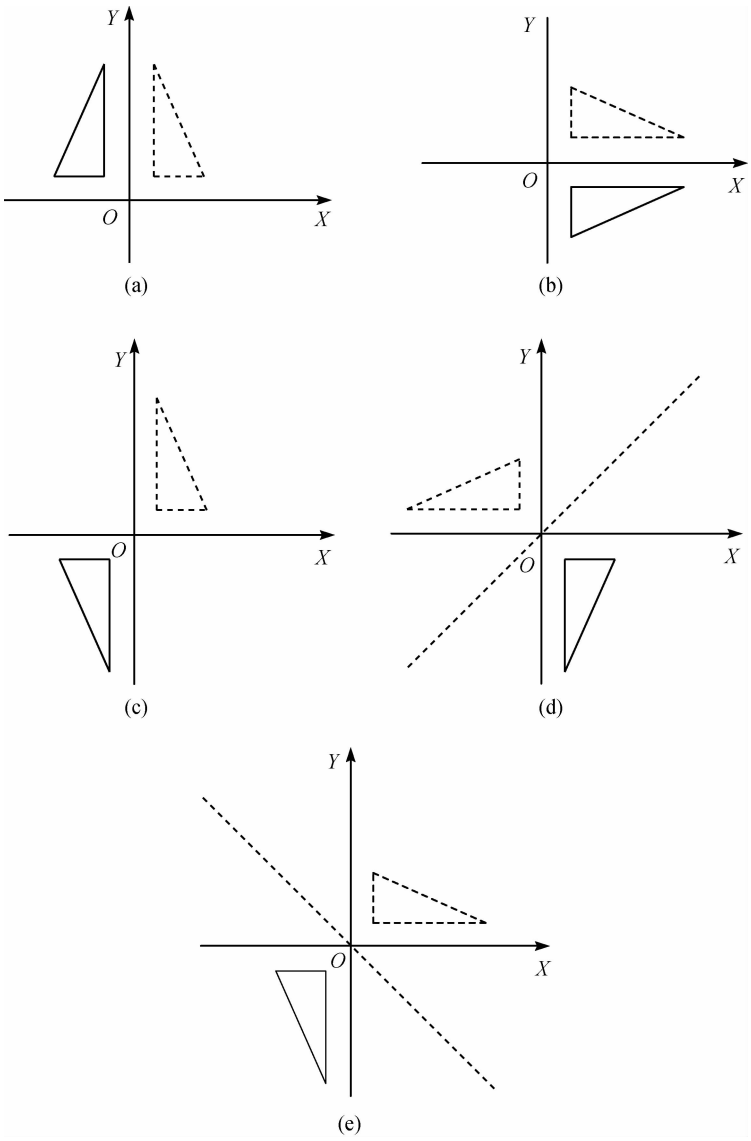

图 4-4　对称变换

(a)Y 轴对称；(b)X 轴对称；(c)中心对称；(d)$y=x$ 对称；(e)$y=-x$ 对称

4.3.2　错切变换

错切变换通常用如下公式实现：

$$P'=[x' \quad y' \quad 1]=[x \quad y \quad 1]\cdot\begin{bmatrix}1 & d & 0\\ b & 1 & 0\\ 0 & 0 & 1\end{bmatrix}=[x+by \quad dx+y \quad 1]$$

(1)当 $d=0$ 时，$x'=x+by$，$y'=y$，此时图形的 y 坐标值不变，x 坐标值随初值(x,y)及变换系数 b 作线性变化：如 $b>0$，图形沿 x 轴正方向作错切移位；若 $b<0$，图形沿 x 轴负方

向作错切移位。

(2)当 $b=0$ 时，$x'=x$，$y'=dx+y$，此时图形的 x 坐标值不变，y 坐标值随初值(x,y)及变换系数 d 作线性变化：如 $d>0$，图形沿 y 轴正方向作错切移位；$d<0$，图形沿 y 轴负方向作错切移位。

(3)当 $b\neq0$，且 $d\neq0$ 时，$x'=x+by$，$y'=dx+y$，图形沿 X 轴、Y 轴两个方向作错切移位。

图 4-5 是两种错切变换的示意图。

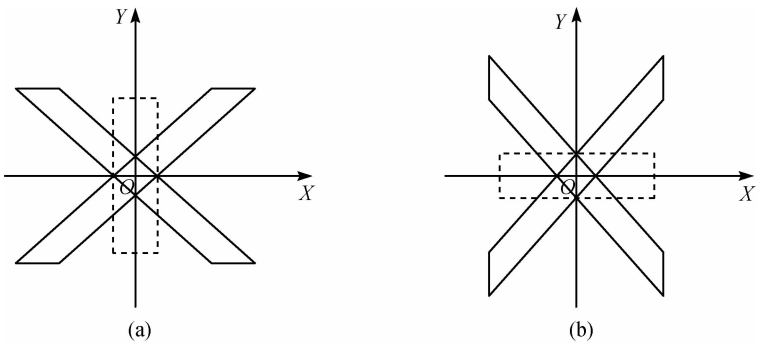

图 4-5　错切变换

(a)X 方向错切；(b)Y 方向错切

以上是常见的二维图形几何变换，有几点需要注意：平移变换只改变图形的位置，不改变图形的大小和几何形状；旋转变换也保持了图形的大小和几何形状；比例变换改变了图形的大小和几何形状；错切变换将引起图形各部分之间角度关系的变化，会导致图形发生畸变；拓扑不变的几何变换不改变图形的连接关系和平行关系。

4.4　二维坐标系变换

实际应用中，常常需要在场景处理的不同阶段将图形对象描述从一个坐标系变换到另一个坐标系。如在图形显示时将图形对象从世界坐标系转换到设备坐标系。而在建模和设计时，往往先在局部坐标系中设计每个对象，再变换到整个场景坐标系中。

如图 4-6 所示，给出了在直角坐标系 XOY 中定义的由坐标原点(x_0,y_0)及方向角 θ 指定的用户坐标系 $X'OY'$。为了将对象描述从 XOY 坐标系变换到 $X'OY'$ 坐标系，必须建立 $X'OY'$ 坐标系与 XOY 坐标系重合的变换。

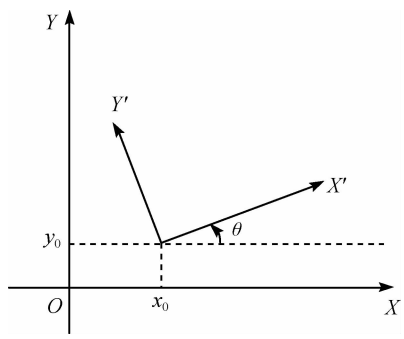

图 4-6　初始两个坐标系位置关系

这样的变换可以分两步进行：

(1)把 $X'OY'$ 坐标系的坐标原点(x_0,y_0)平移到 XOY 坐标系的原点$(0,0)$。

(2)将 X' 轴旋转到 X 轴上。将(x_0,y_0)平移至$(0,0)$点的齐次变换矩阵如下：

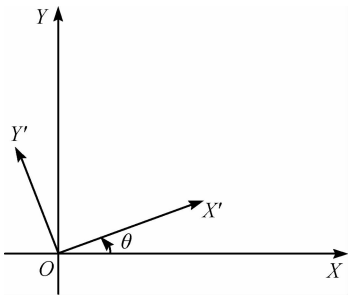

图 4-7　平移后的两个坐标系

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

平移变换后，坐标系变为如图 4-7 所示。

将 X' 轴旋转到 X 轴上的齐次旋转矩阵如下：

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

完整的齐次变换矩阵为：

$$T \cdot R$$

4.5　二维图形裁剪

前面已经介绍了二维图形的几何变换方法，这里来讨论如何将几何对象在输出设备上显示出来。用户定义的图形从窗口到视见区的输出过程是这样的：首先定义要显示图形的用户坐标系，然后定义用户坐标系中的窗口，对窗口进行裁剪后，作窗口到规格化设备坐标系中指定的视见区的变换，然后作视见区从规格化坐标系到设备坐标（即显示器像素坐标）系的变换，最后在图形设备上输出图形。二维观察变化的主要步骤如图 4-8 所示。

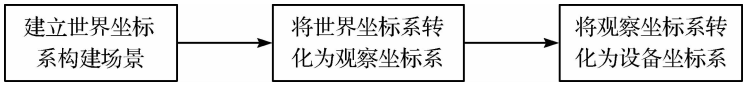

图 4-8　二维观察变换的主要步骤

4.5.1　窗口变换

通常来说，用户坐标系是无限的，它在各个方向上是无限延伸的，而所有的显示界面，如绘图纸或显示屏幕都是有限的，是有边界的。在二维场景中，显示的部分通常称为裁剪窗口。图形位于窗口内的部分被显示出来，而位于窗口外的部分则会被裁去。除裁剪窗口外，还有另外一种窗口，称为“视口”，用来定位裁剪窗口的显示位置。通过改变视口，可以在输出设备的不同区域观察物

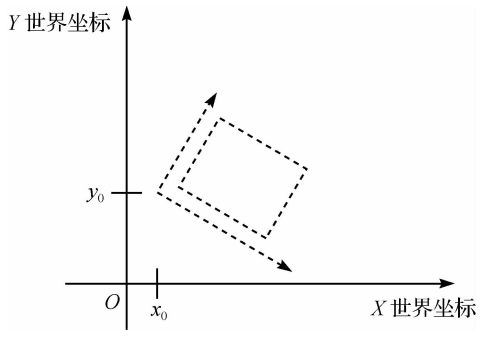

图 4-9　在世界坐标系中定义旋转的裁剪窗口

体，通过改变视口的尺寸，可以改变显示对象的尺寸和位置。将不同尺寸的裁剪窗口连续映射到固定尺寸的视口中，则可得到类似“拉镜头”的效果。一般的方法是在世界坐标系中制定一个观察坐标系统，以该系统为参考指定矩形的裁剪窗口，如图 4-9 所示。

为了把在用户坐标系中定义的图形对象在实际的显示器上显示出来，就必须把用户坐标转换成具体的设备坐标，也就是通过变换，使观

察坐标系与世界坐标系重合。坐标系的变换方法已经在 4.4 节中介绍过了，这里不再赘述。

4.5.2　视见变换

裁剪窗口规定了显示图形的范围，视见区规定了显示图形在显示屏幕上的位置和大小，视见变换就是将用户坐标系窗口内的图形变换到显示屏幕设备坐标系的视见区中显示。图 4-10 演示了一个简单的二维视见变换。

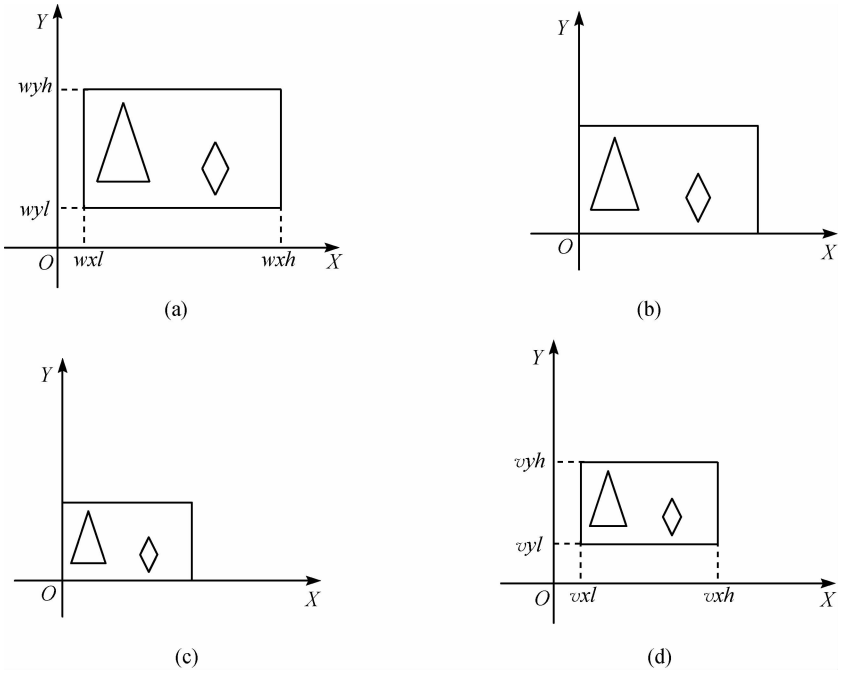

图 4-10　简单的二维视见变换

(a)用户坐标系中的窗口；(b)平移变换 T_1；(c)比例变换 S；(d)平移变换 T_2

图 4-10(a)定义了窗口和位于窗口内部的图形对象，用其左下角点坐标(wxl，wyl)和右上角点坐标(wxh，wyh)来确定，图 4-10(d)定义了视口，用左下角点坐标(vxl，vyl)和右上角点坐标(vxh，vyh)来确定。现要将窗口中的图形对象显示在视口当中。这需要对两个不同的坐标系的矩形区域作变换，所以可以将两坐标系视为重合后再应用平移变换和比例变换求得总的变换关系。

如图 4-10(b)和图 4-10(c)所示，先平移窗口使其左下角与坐标系原点重合，再进行比例变换使其大小与视口相等，最后再通过平移使其与视口重合。窗口中的全部图形经过与此相同的变换后就成为视口中的图形了。因此可得视见变换矩阵为：

$$H = T_1 \cdot S \cdot T_2$$

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

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

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：

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

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

在多数绘图系统中都可以由用户设置窗口和视见区的大小和位置，系统根据用户设定构成视见变换矩阵，并作用于之后的所有图形显示，使在用户坐标系中的图形描述变换成设备坐标系中的图形描述。在实际应用中可以利用这一点来达到改变图形形状的目的，如保持视见区的大小不变，而减小窗口就可以使显示图形放大。

4.5.3 直线段裁剪算法

图形元素不同，采用的裁剪算法也不同。点是组成图形的基本单位，相对于裁剪窗口来说，对点作裁剪是很简单的。

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)是位于裁剪窗口内的点，予以保留，否则就要舍弃：

$$x_l\leqslant x\leqslant x_r,y_b\leqslant y\leqslant y_t$$

对直线段的裁剪算法是很多图形裁剪方法的基础，是我们讨论的重点。直线段的裁剪算法有很多，下面介绍常用的几种。

1)Cohen-Sutherland 算法

这是一个最早开发的快速线段裁剪算法，已经得到了广泛的应用。该算法首先判断直线段是否全部在窗口内，若是，则保留；若不是，则再判断是否完全在窗口之外，若是，则舍弃。如果这两种情况都不属于，则将此直线段分割，对分割后的子线段再进行如前判断，直至所有直线段和由直线段分割出来的子线段都已经确定了是保留还是舍弃为止。

为了方便判断，先将窗口的四边延长，将整个平面划分为 9 个区域，如图 4-11 所示。然后对这些区域用 4 位二进制代码进行编码，每一区域中的点采用同一代码，称为区域码。编码规则如下：如果该区域在窗口的上方，则代码的第四位为 1；如果该区域在窗口的下方，则代码的第三位为 1；如果该区域在窗口的右侧，则代码的第二位为 1；如果该区域在窗口的左侧，则代码的第一位为 1，如图 4-11 所示。根据此规则，就可以得到图 4-12 所示的代码。

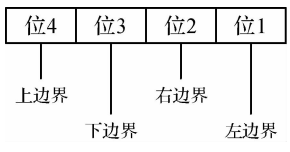

图 4-11 边界编码

下面介绍如何利用这样的编码来实现直线的裁剪。

给所有线段的端点设置了编码后，就可以快速地判断出哪条线段完全位于窗口内，哪条线段完全位于窗口外了。完

全位于窗口内的线段，其两个端点的区域码均为 0000，予以保留，如图 4-13 中所示的线段 AB；两个端点的区域码中，有一对相同位置均为 1 的线段则完全落在裁剪窗口之外，我们可以将该线段从场景中删除，如图 4-13 中所示的线段 CD。

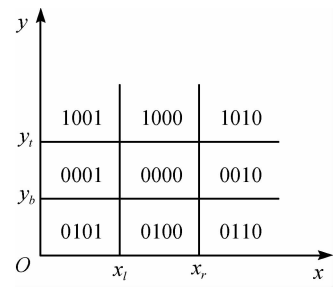

图 4-12　区域编码

对于剩余的线段，则要测试其余窗口边界的交点，如图 4-13 中所示的线段 EF、HI 及 JK。这些线段可能穿过也可能不穿过裁剪窗口，因此可能需要进行多次求交运算才能完成一条线段的裁剪。求交的次数依赖于选择裁剪边界的次序。每次处理一条裁剪边界后，裁掉其中的一部分，余下部分对照窗口的其余边界进行检查，直到余下部分完全位于裁剪窗口内或者完全被裁剪掉。

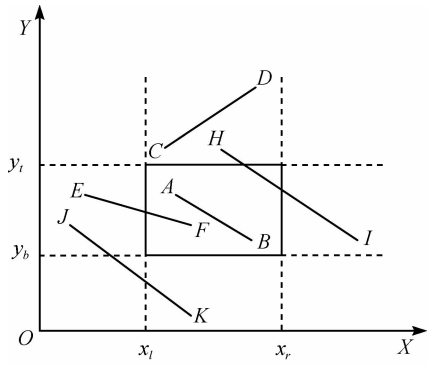

图 4-13　Cohen-sutherland 裁剪算法的例子

具体判断和裁剪方法描述如下：

设线段两端点编码分别为 c_1 和 c_2。

(1)如果 $c_1=c_2=0$，说明线段的两个端点全部位于裁剪窗口内，如图 4-13 中所示的线段 AB，显然此时整个直线段位于窗口内，应该保留。

(2)$c_1\times c_2\neq 0$，这里的×是逻辑乘，即 c_1 和 c_2 至少有某一位同时为 1，表明两端点必定同处于某一边界的同一外侧，则整个直线段全在窗口外，应该舍弃，如图 4-13 中所示的线段 CD。

(3)如不属于上面两种情况，则又可分为 3 种情况：

①一个端点在内，另一个端点在外，如图 4-13 中所示的线段 EF。

②两个端点均在外，但直线段中部跨越窗口，如图 4-13 中所示的线段 HI。

③两个端点均在外，且直线段也在外，如图 4-13 中所示的线段 JK。

(4)求交。对不能确定取舍的直线段，求其与窗口边界及其延长线的交点，从而将直线段分割。求交点时，可以有针对性地与某一确定边界求交。如图 4-13 中所示的直线段 EF，点 E 所在区域代码为 0001，F 所在区域代码为 0000，这表明 E 在窗口的左侧，而 F 不在左侧，则 EF 与 $x=x_l$ 必定相交。可求得交点 E'，从而可舍弃 EE'，而保留 $E'F$。

(5)对剩下的线段 $E'F$ 重复以上各步。可以验证，至多重复到第三遍的判断为止(如对图 4-13 中所示的直线段 HI)，这时剩下的直线段或者全在窗口内，或者全在窗口外，从而完成了对直线段的裁剪。

算法的程序实现如下(函数 Cohen_Sutherland 用来实现算法,函数 makecode 用来编码,程序中利用了数值的位运算):

```
double xl, xr, yt, yb; //这里事先给出窗口的位置,4 个数值是已知的
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==1)
        {
            y=y0+(y2-y0)*(xr-x0)/(x2-x0);
            x=xr;
        }
        else if(c&4==1)
        {
            x=x0+(x2-x0)*(yb-y0)/(y2-y0);
            y=yb;
        }
        else if(c&8==1)
        {
            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

设要裁剪的直线段为 P_0P_1，如图 4-14 所示。中点分割算法主要思想如下：从 P_0 点出发找出离 P_0 点最近的可见点(图 4-14 中的 A 点)以及从 P_1 点出发找出离 P_1 点最近的可见点(图 4-14 中的 B 点)，则这两个最近可见点的连线 AB 就是原直线段的可见部分。求两个最近可见点的计算可分成两个过程平行进行。

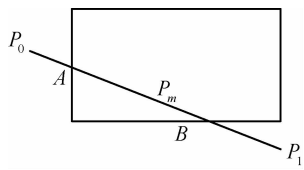

图 4-14　中点分割算法

从 P_0 出发找最近可见点的方法是先求 P_0P_1 的中点 P_m，若 P_0P_m 不能定为显然不可见，则取 P_0P_m 代替 P_0P_1，否则取 P_mP_1 代替 P_0P_1，再对新的 P_0P_1 求中点。重复上述过程，直到 P_1P_m 长度小于给定的阈值 ε 为止。

图 4-15 是求 P_0 的最近可见点的算法流程图。求 P_1 的最近可见点的步骤是一样的，只要把 P_0 和 P_1 互换即可。在显示时，ε 可以取成一个像素的宽度，对于一个分辨率为 $2^N \times 2^N$ 的显示器来说，这个二分过程最多需要进行 N 次。由于在此计算过程中只要作加法和除 2 运算，所以特别适合用硬件来实现。如果能够使两个查找最近点的过程平行进行，就可以使裁剪速度加快。

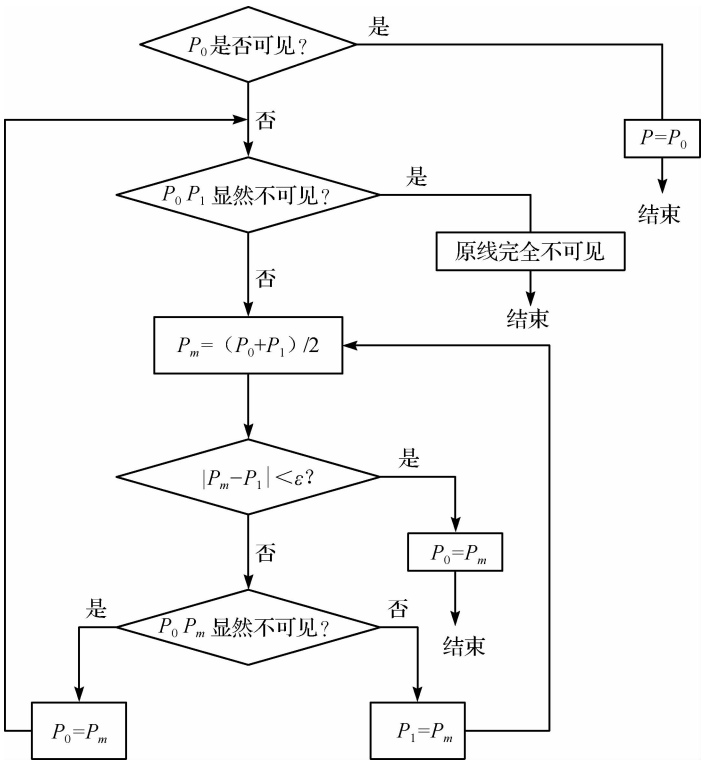

图 4-15　中点分割算法流程图

3)梁友栋-Barsky 算法

梁友栋和 Barsky 分别提出了参数线裁剪的快速算法。设要裁剪的直线段为 P_0P_1，P_i 的坐标为(x_i,y_i)，$i=0,1$。P_0P_1 所在直线和窗口的 4 条边界所在直线分别交于 A、B、C 和 D 四个点，如图 4-16 所示。该算法的基本思想是从 A、B 和 P_0 三点中找出最靠近 P_1 的点，在图 4-16 中，该点是 P_0；从 C、D 和 P_1 三点中找出最靠近 P_0 的点，在图 4-16 中，该点是 C 点。那么 P_0C 就是 P_0P_1 线段上的可见部分。

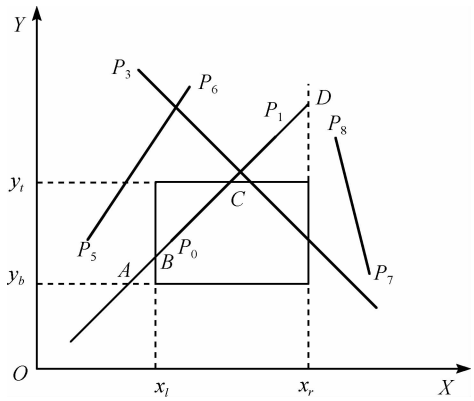

图 4-16　梁友栋-Barsky 裁剪算法

在具体计算时，可以把 P_0P_1 写成关于 t 的参数方程：

$$x=x_0+\Delta x\cdot t$$

$$y=y_0+\Delta y\cdot t$$

其中，$\Delta x=x_1-x_0$，$\Delta y=y_1-y_0$，$t\in[0,1]$。

我们把裁剪窗口的 4 条边分成两类，一类称为始边，另一类称为终边。当 $\Delta x\geqslant 0(\Delta y\geqslant 0)$ 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。对于图 4-16 中所示的 P_0P_1 来说，$x=x_l$ 和 $y=y_b$ 为始边，$x=x_r$ 和 $y=y_t$ 为终边。对于图 4-16 中所示的 P_3P_4 来说，$x=x_l$ 和 $y=y_t$ 为始边，$x=x_r$ 和 $y=y_b$ 为终边。求出 P_0P_1 和两条始边的交点的参数 t'_0 和 t''_0，令

$$t_0=\max(t'_0,t''_0,0)$$

则参数为 t_0 的点就是图 4-16 中 A、B 和 P_0 三点中最靠近 P_1 的点。求出 P_0P_1 和两条终边的交点的参数 t'_1 和 t''_1，令

$$t_1=\min(t'_1,t''_1,1)$$

则参数为 t_1 的点就是图 4-16 中 C、D 和 P_1 三点中最靠近 P_0 的点。

当 $t_1>t_0$ 时，P_0P_1 参数方程中参数 $t\in[t_0,t_1]$ 的线段就是 P_0P_1 的可见部分。当 $t_0>t_1$ 时，整个直线段为不可见，图 4-16 中的直线段 P_5P_6 和 P_7P_8 就属于这种情况。

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

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

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

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

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

可知，交点的参数为：

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

在这里，t_l 是 P_0P_1 和 $x=x_l$ 的交点的参数，t_r 是 P_0P_1 和 $x=x_r$ 的交点的参数，t_t 是 P_0P_1 和 $y=y_t$ 的交点的参数，t_b 是 P_0P_1 和 $y=y_b$ 的交点的参数。

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 是 P_0P_1 和终边的交点的参数。当 $Q_i=0$ 时，若 $D_i<0$，则 P_0P_1 是完全不可见的（如图 4-17 中所示的直线段 AB，它使 $Q_r=0$，$D_r<0$）。当 $Q_i=0$ 而相应的 $D_i\geqslant 0$ 时，如图 4-17 中所示的直线段 CD，它使 $Q_l=0$，$D_l>0$ 和 $Q_r=0$，$D_r>0$。这时由于 CD 和 $x=x_l$ 及 $x=x_r$ 平行，所以不必去求出 CD 和 $x=x_l$ 及 $x=x_r$ 的交点，而让 CD 和 $y=y_t$ 及 $y=y_b$ 的交点决定直线段上的可见部分。

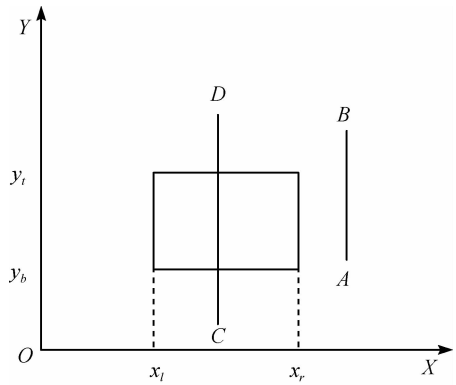

图 4-17　$Q_i=0$ 的情况

算法的程序实现如下（函数 L_Barsky 用来实现算法，函数 cansee 用于判断直线段是否可见）：

```
double xl, xr, yt, yb;     //这里事先给出窗口的位置,4 个数值是已知的
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;
    }
```

4.5.4　其他图形的裁剪

除了对点和直线段的裁剪之外，在实际应用中常会遇到裁剪其他图形的情况，如对字符、多边形、圆弧和任意曲线的裁剪。

1)字符裁剪

对字符的裁剪可以采用几种方法。如果把字符的每一笔看成是由一条直线段或几条直线段组成的，那么就可以用直线段裁剪的方法处理每一笔，这样就得到了字符的裁剪方法，如图 4-18(a)所示。或者把包含一个字符的最小矩阵的中心或左下角在窗口外的字符认为不可见，如图 4-18(b)所示。也可以以一个字符串为单位来裁剪，如果包含字符串的最小矩形的中心或左下角在窗口外，则认为整个字符串为不可见，如图 4-18(c)所示。

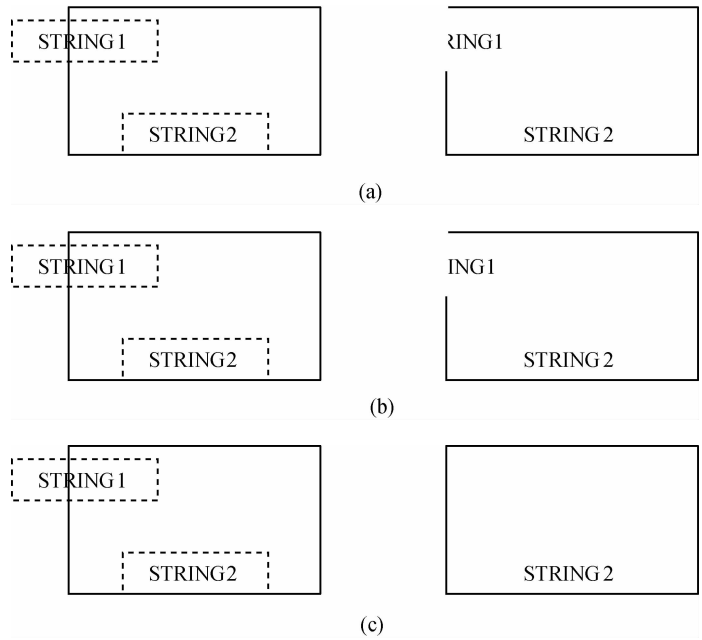

图 4-18　三种字符裁剪方法的不同结果

(a)按笔画裁剪；(b)按字符裁剪；(c)按字符串裁剪

2)多边形裁剪

虽然多边形是由若干直线段首尾相连而成的,但裁剪多边形却不能直接使用直线裁剪算法对每条边进行裁剪,因为该方法一般情况下不能生成封闭的折线,裁剪后往往产生若干不连接的线段。而在图形学中,多边形常被认为是一个封闭多边形,它把平面分成多边形的内部和外部两部分。对多边形的裁剪结果要求仍是多边形。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,如图 4-19 所示。

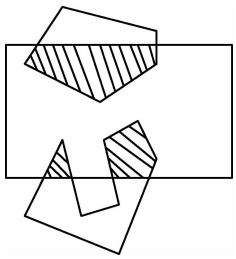

图 4-19　多边形的裁剪(阴影部分为裁剪后保留下来的)

对多边形的裁剪可以采用 Sutherland-Hodgman 算法,该算法十分简便,只要对多边形用窗口的 4 条边裁剪 4 次就可得到裁剪后的多边形,如图 4-20 所示。图 4-21 是这一算法的流程图。

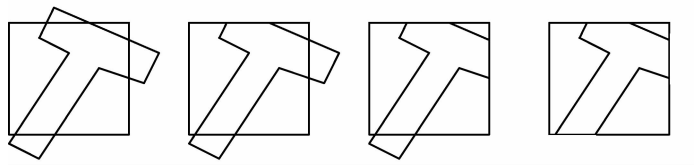

图 4-20　Sutherland-Hodgman 多边形裁剪算法

设封闭多边形顶点为 $P_1,P_2,\cdots,P_n$,图 4-21 中的 e 是表示窗口的 4 条边中正在进行裁剪的一条边,每次裁剪时将多边形的第一个顶点存放在 F 中,以便对最后一条边裁剪时用。用图 4-21(a)中所示的算法对边 $P_1P_2,P_2P_3,\cdots,P_{n-1}P_n$ 中的每一条作裁剪,再用图 4-21(b)中所示的算法对最后一条边 P_nP_1 作裁剪。裁剪好一条边就输出一条边。该算法要对窗口的 4 条边用 4 次。算法执行完毕后,再将产生出来的不属于多边形的边去掉,就可以得到最后的裁剪结果。

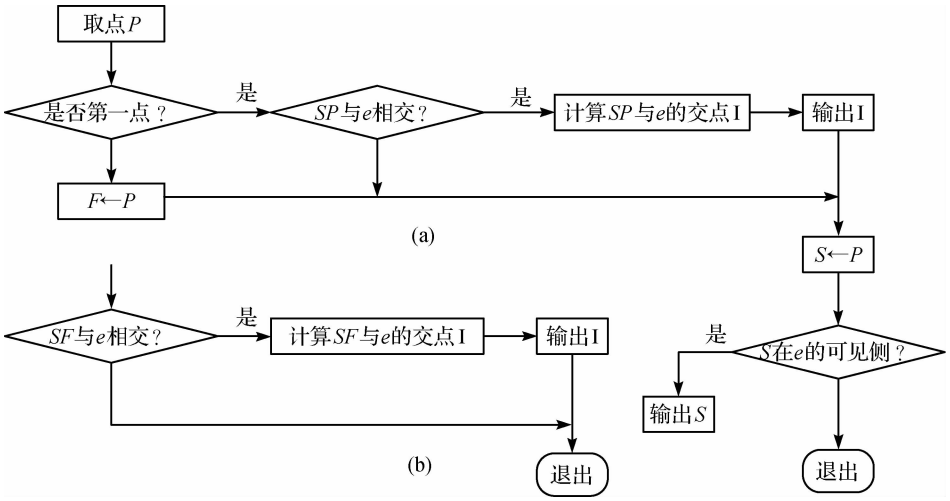

图 4-21　Sutherland-Hodgman 算法流程图

(a)对多边形前 $n-1$ 条边中的任意一条作裁剪;(b)对多边形的最后一条边作裁剪

3)其他裁剪

裁剪圆弧时,可把圆弧和窗口的 4 条边的交点求出来,再按交点对圆心幅角的大小排序,排序后,相邻的两个交点决定了圆弧上一段可见或不可见的弧。

由于任意曲线可以用直线和圆弧来逼近,所以任意曲线的裁剪问题可以转化为对直线或圆弧的裁剪。

以上介绍的是裁剪区域为矩形区域时的裁剪算法。而当裁剪区域为非矩形区域时,对于不同形状的裁剪区域有不同的裁剪算法。下面介绍一个简单的裁剪区域为任意凸多边形区域时的直线段的裁剪算法。假设裁剪区域为一个任意凸多边形区域,它是由 n 个直线段 P_1P_2,P_2P_3,…,$P_{n-1}P_n$,P_nP_1 连接而成的折线围成的,如图 4-22 所示。

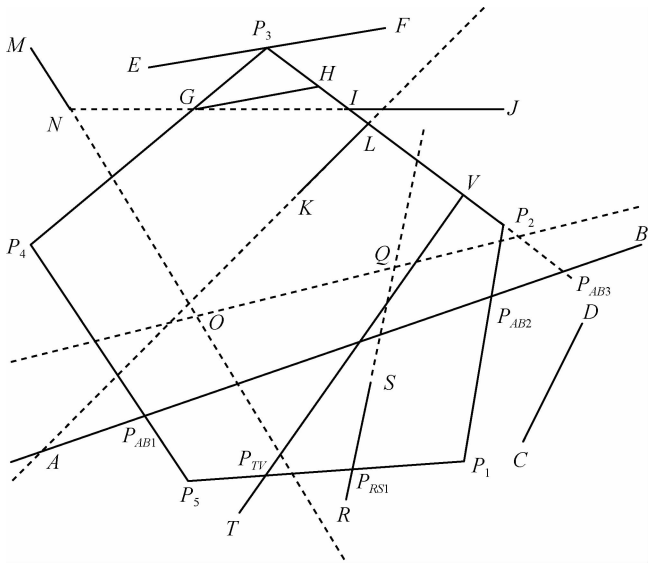

图 4-22　裁剪区域为凸多边形区域时对直线段的裁剪

算法如下:

第一步,计算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。这里的交点必须是在凸多边形区域的边界直线段上的点。在图 4-22 中,直线段 AB 与 P_2P_3 的交点 P_{AB3} 因为不在直线段 P_2P_3 上,所以不记为交点。而 P_{AB1} 和 P_{AB2} 因为分别在直线段 P_4P_5 和 P_1P_2 上,所以记为交点。根据凸多边形的特性可知,这样的交点至多有两个,所以如果已经有两个交点了,则可终止计算,直接进行下一步。下面根据交点个数的不同来进行裁剪工作。

第二步,当交点的个数为 0 或 1 时,该直线段处于凸多边形区域外,如图 4-22 中所示的直线段 CD 和 EF,所以该直线段完全不可见。

第三步,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。有如下几种情况:

(1)如果直线段的两个端点和两个交点刚好全部重合,如图 4-22 中所示的直线段 GH,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,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 3 种情况：

- 如果重合点和交点不相邻，则直线段完全可见，如图 4-22 中所示的直线段 KL。
- 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，如图 4-22 中所示的直线段 TV，它的可见部分为直线段 $P_{TV}V$。
- 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，如图 4-22 中所示的直线段 IJ。

(3)如果直线段的两个端点和两个交点都不重合，则可将这 4 个点按在直线上的顺序排列，根据排列的顺序有如下 4 种情况：如果两个端点和两个交点分别相邻，则直线段完全不可见，如图 4-22 中所示的直线段 MN；如果两个端点在两个交点之间，则直线段完全可见，如图 4-22 中所示的直线段 OQ；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，如图 4-22 中所示的直线段 AB，它的可见部分为直线段 $P_{AB1}P_{AB2}$；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，如图 4-22 中所示的直线段 RS，它的可见部分为直线段 $P_{RS1}S$。

经过以上三步即可完成对直线段的裁剪。上面算法的第三步比较复杂，可以加以简化。其实在凸多边形区域中直线段的可见部分只能是处于两个交点之间的部分。再根据直线段的端点位置就可以确定直线段是完全可见，还是部分可见，而且可以确定直线段的可见部分。读者可以自己试着来设计具体算法。

4.6 三维空间的几何变换

三维图形的几何变换是在二维图形几何变换的基础上考虑 z 坐标而得到的。通常约定三维世界坐标系为右旋坐标系，即张开右手，拇指向外伸，当拇指和食指分别与 x 轴和 y 轴方向一致时，垂直掌心向外的方向就是 z 轴方向。

三维点坐标在齐次坐标中表示为四维向量，三维几何变换的齐次变换矩阵为 4×4 矩阵。下面我们就一一介绍三维图形的几何变换。

4.6.1 平移变换

设三维空间点 P 的齐次坐标为 $P=[x \quad y \quad z \quad 1]$，平移变换后得到的目标点 P' 的齐次坐标为 $P'=[x' \quad y' \quad z' \quad 1]$，则得到下面的平移变换公式：

$$P'=[x' \quad y' \quad z' \quad 1]=[x \quad y \quad z \quad 1]\cdot\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ D_x & D_y & D_z & 1\end{bmatrix}=P\cdot T(D_x,D_y,D_z)$$

上式中的 D_x、D_y、D_z 分别是沿 X 轴、Y 轴、Z 轴方向上的平移量。图 4-23 是三维平移变换示意图。

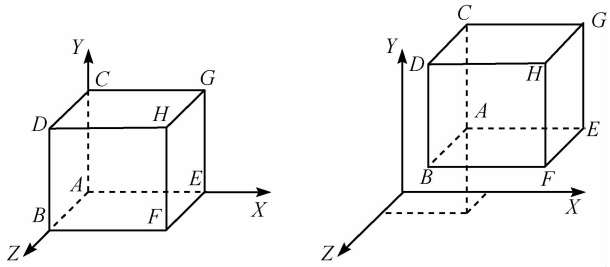

图 4-23　三维图形的平移变换

4.6.2　比例变换

设三维空间点 P 的齐次坐标为 $P=[x \quad y \quad z \quad 1]$，比例变换后得到的目标点 P' 的齐次坐标为 $P'=[x' \quad y' \quad z' \quad 1]$，则得到下面的比例变换公式：

$$P'=[x' \quad y' \quad z' \quad 1]=[x \quad y \quad z \quad 1]\cdot\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}=P\cdot S(S_x,S_y,S_z)$$

该比例变换以坐标原点为参考点。上式中的 S_x、S_y、S_z 分别是沿 X 轴、Y 轴、Z 轴方向上的缩放比例。图 4-24 就是以坐标原点为参考点的三维比例变换示意图。

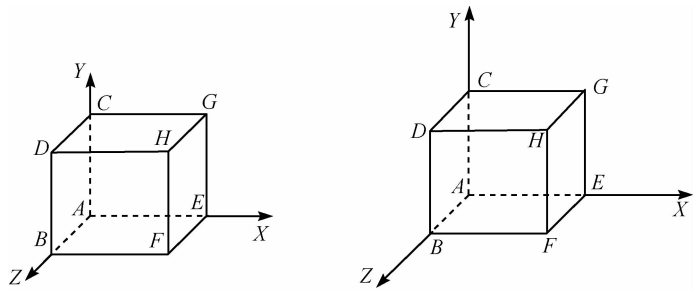

图 4-24　以坐标原点为参考点的比例变换

如果要以三维空间中的任意一点 (x_0,y_0,z_0) 为参考点作比例变换，只需采用与二维比例变换相同的方法，先将参考点平移至原点，作以原点为参考点的比例变换，再将参考点平移回点 (x_0,y_0,z_0) 的位置。比例变换矩阵为：

$$S=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_0 & -y_0 & -z_0 & 1 \end{bmatrix}\cdot\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\cdot\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_0 & y_0 & z_0 & 1 \end{bmatrix}$$

$$=\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ (1-S_x)\cdot x_0 & (1-S_y)\cdot y_0 & (1-S_z)\cdot z_0 & 1 \end{bmatrix}$$

图 4-25 就是以任意点(x_0, y_0, z_0)为参考点的比例变换过程。

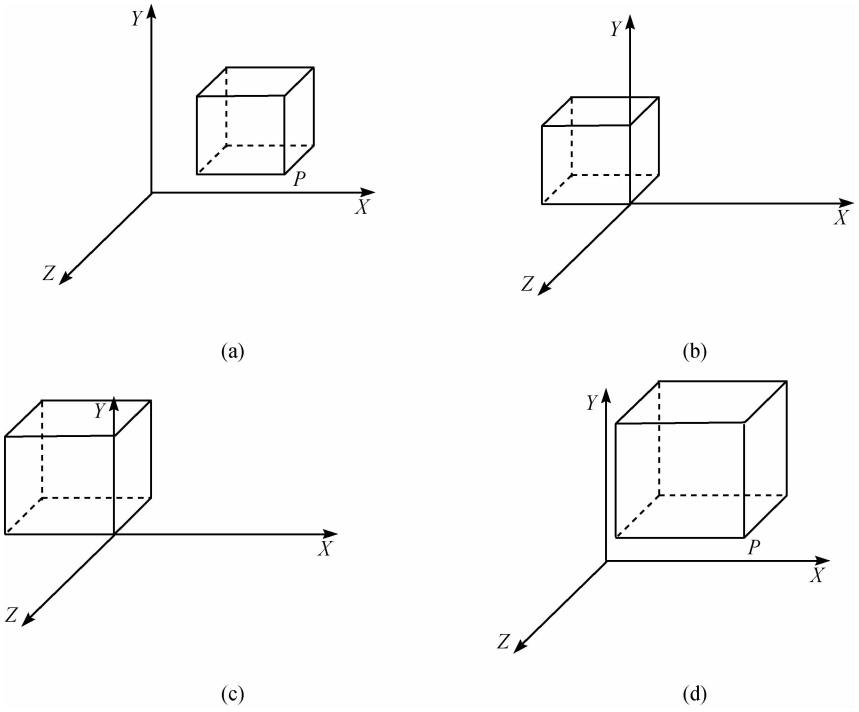

图 4-25　以任意点为参考点的比例变换

(a)原图；(b)平移回原点；(c)比例变换；(d)平移回 P 点

4.6.3　旋转变换

之前我们讨论 XOY 平面上的二维旋转时，旋转中心为一个点，或者说垂直于 XOY 平面的坐标轴。而在三维旋转变换时，可以选择空间中的任意方向作为旋转轴。在所有的三维旋转中，旋转轴为某一坐标轴的旋转是最容易处理的。而任意方向轴的旋转，又可以通过若干次绕坐标轴的旋转(适当结合三维平移操作)来实现。因此，这里我们先给出绕 3 个坐标轴旋转的变换矩阵。旋转的正方向通常约定按右手法则来确定，即面向旋转变换所绕坐标轴的正方向看，逆时针方向为旋转的正方向，如图 4-26 所示。

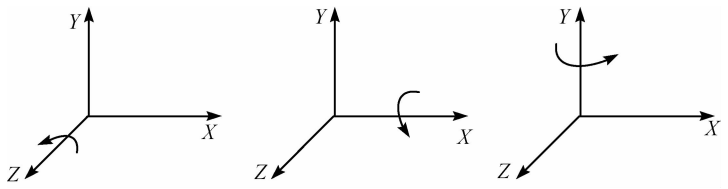

图 4-26　绕坐标轴旋转示意图

设三维空间点 P 的齐次坐标为 $P=[x \quad y \quad z \quad 1]$，旋转变换后得到的目标点 P' 的齐次坐标为 $P'=[x' \quad y' \quad z' \quad 1]$。

1)绕 *Z* 轴旋转

绕 Z 轴的三维旋转很容易由二维旋转得到：

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \\ z' &= z \end{aligned}$$

绕 Z 轴的三维旋转公式可用齐次坐标表示如下：

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

上式中的 θ 为图形绕 Z 轴旋转的角度。用类似的方法可以得到另外两个旋转变换矩阵。

2)绕 *X* 轴旋转

绕 X 轴的三维旋转公式可用齐次坐标表示如下：

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

上式中的 θ 为图形绕 X 轴旋转的角度。

3)绕 *Y* 轴旋转

绕 Y 轴的三维旋转公式可用齐次坐标表示如下：

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

上式中的 θ 为图形绕 Y 轴旋转的角度。

图 4-27 表示了绕坐标轴 Y 轴旋转 90°的情形。

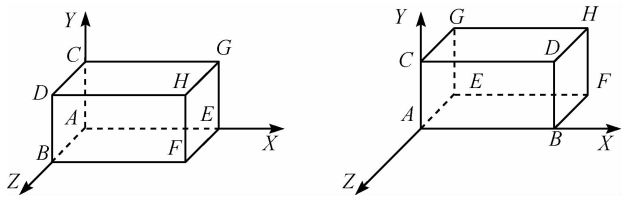

图 4-27　绕 *Y* 轴旋转 90°

4)一般三维旋转

如果绕不平行于坐标轴的旋转轴将空间图形旋转角度 θ，那么我们需要首先将变换转换

为绕某一个坐标轴的旋转变换，即将旋转轴与某一坐标轴平行，再对预处理后的对象绕坐标轴旋转角度 θ，最后将旋转轴通过变换回复原位。

当旋转轴平行于某个坐标轴时，可以通过下列变换来得到所需的旋转矩阵：

(1)平移对象使其旋转轴与所平行的坐标轴重合。

(2)对对象完成指定的旋转变换。

(3)平移对象将其旋转轴移回到原来的位置。

例如，绕平行于 X 轴的旋转轴的旋转变换可用齐次坐标表示为：

$$P'=P\cdot T^{-1}\cdot R_x(\theta)\cdot T$$

当旋转轴不与任何一个坐标轴平行时，旋转需要通过 5 个步骤来实现：

(1)平移对象，使得旋转轴通过坐标原点。

(2)旋转对象，使得旋转轴与某一个坐标轴重合。

(3)绕坐标轴完成指定的旋转。

(4)利用逆旋转使旋转轴回到其原始方向。

(5)利用逆平移使旋转轴回到其原始位置。

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)，而 $n=[n_1 \quad n_2 \quad n_3]$，$|n|=\sqrt{n_1^2+n_2^2+n_3^2}=1$。以这条直线为旋转轴做旋转 θ 角的旋转变换，使三维空间中任意一点 P 变成 P'，如图 4-28 所示。

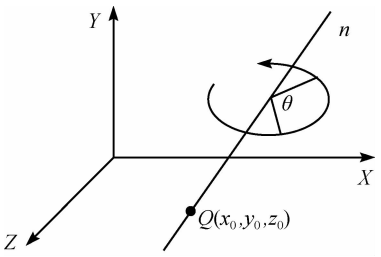

图 4-28　以空间任意直线为旋转轴的旋转变换

具体的计算方法如下：

(1)平移变换，其中令平移矩阵为 $T(-x_0,-y_0,-z_0)$，使旋转轴成为通过坐标原点的一条直线。

(2)进行绕 X 轴旋转 α 角的变换 $R_x(\alpha)$，使旋转轴落在 $y=0$ 平面上；再进行绕 Y 轴旋转 β 角的变换 $R_y(\beta)$，使旋转轴与 Z 轴重合。

(3)进行绕 Z 轴旋转 θ 角的旋转变换。

(4)进行步骤(2)的逆变换，即先进行旋转变换 $R_y(-\beta)$，再进行旋转变换 $R_x(-\alpha)$。

(5)进行步骤(1)的逆变换，即平移变换 $T(x_0,y_0,z_0)$，使旋转轴平移回到原来的位置。

这样完成所要求变换的变换矩阵为：

$$T(-x_0,-y_0,-z_0)\cdot R_x(\alpha)\cdot R_y(\beta)\cdot R(\theta)\cdot R_x(-\alpha)\cdot R_y(-\beta)\cdot T(x_0,y_0,z_0)$$

其中，$R_x(\alpha)$和 $R_y(\beta)$可以通过下面两个步骤求得：

第一步，如图 4-29 所示，图中 O 是坐标原点，点 N 的坐标是 (n_1,n_2,n_3)，因此 ON 是被作为旋转轴的任意直线。

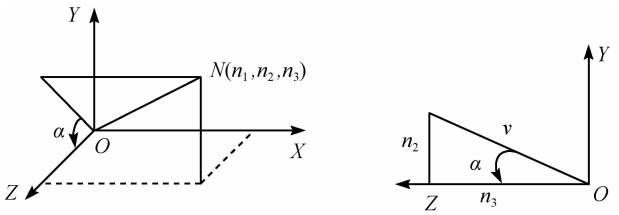

图 4-29　绕 X 轴旋转 α 角

记 $v=\sqrt{n_2^2+n_3^2}$，则有：

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

因此可得：

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

此变换将点 N 变换成 N'，用齐次坐标记法有：

$$N'=[n_1 \quad n_2 \quad n_3 \quad 1]\cdot R_x(\alpha)=[n_1 \quad 0 \quad v \quad 1]$$

可见旋转轴已经落在 $y=0$ 平面上了。

第二步，如图 4-30 所示，记 $d=\sqrt{n_1^2+n_2^2+n_3^2}$，这里需要注意的是面向 Y 轴，图中的 β 角是沿顺时针方向给出的，按照对旋转正方向的约定，这是旋转的负方向。又因为 $d=1$，所以有：

$$\cos\beta=\frac{v}{d}=v,\sin\beta=-\frac{n_1}{d}=-n_1$$

因此所进行变换的矩阵是：

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

此变换将点 N' 变换成 N''，用齐次坐标记法有：

$$N''=[n_1 \quad 0 \quad v \quad 1]\cdot R_y(\beta)=[0 \quad 0 \quad 1 \quad 1]$$

可见旋转轴已经落到 Z 轴上。

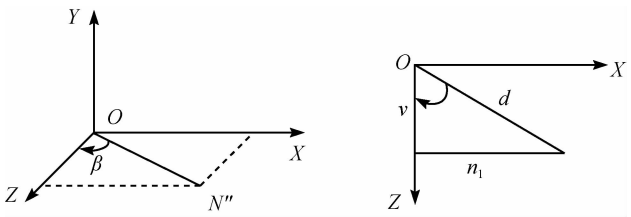

图 4-30　绕 *Y* 轴旋转 *β* 角

4.7 投　　影

现实中的物体都是三维的，而在现阶段，绝大多数的图形显示设备的显示表面都是平面的，也就是二维的。为了解决把我们观察到的三维物体在显示设备的二维显示表面上显示出来的问题，就需要用到投影的方法。把三维物体变为二维图形表示的过程称为投影变换。

投影的过程是这样的：首先在三维空间中确定一个点作为一个投影中心以及一个平面作为投影平面；然后从投影中心引出一些投射直线，这些直线通过三维形体上的每一点与投影平面相交，所有交点在投影平面上就形成了三维形体的二维投影。根据投影中心与投影平面之间距离的不同，投影可以分为平行投影和透视投影。

4.7.1 世界坐标系到观察坐标系的变换

在进行投影变换之前，首先需要将三维对象从世界坐标系转换到观察坐标系中。这个过程实际上是通过一系列的几何变换，使观察坐标系与世界坐标系重合。具体需通过以下步骤实现：

(1)平移：设观察坐标系坐标原点在世界坐标系中的坐标为 $P(x_0, y_0, z_0)$，将观察坐标系的坐标原点平移到世界坐标系的坐标原点的齐次变换矩阵如下：

$$T=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_0 & -y_0 & -z_0 & 1 \end{bmatrix}$$

(2)旋转：设观察坐标系的 3 个坐标轴分别对应世界坐标系中的单位向量：u、v 和 n，分别将 x_{view}、y_{view} 和 z_{view} 轴旋转至与世界坐标系的 X、Y 和 Z 轴重合的齐次变换矩阵如下：

$$R=\begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

综合起来，世界坐标系到观察坐标系的齐次变换系数为：

$$T \cdot R$$

4.7.2 平行投影

当投影中心与投影平面的距离为无穷远时，投射直线变为一组平行线，这种投影即为平行投影。平行投影也可以分为两种类型：正交投影和斜交投影。

1)正交投影

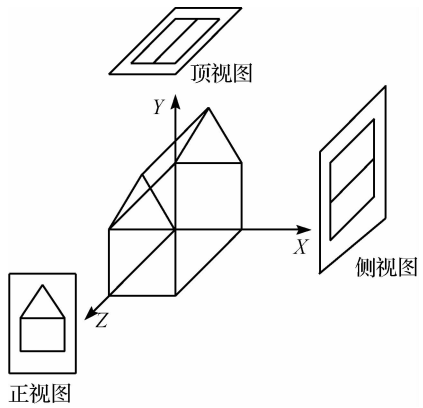

图 4-31 三视图

如果投影方向与投影平面垂直，此时的平行投影即为正交投影。最常见的正交投影是正视投影、顶视投影和侧视投影，也就是我们常说的三视图。三视图的投影平面分别选取垂直于 3 个坐标轴，而投影方向则与坐标轴方向一致，如图 4-31 所示。

如果投影平面 $z=0$，投影方向为 Z 轴正方向，设三维空间中的一点 $P(x, y, z)$，投影后得到点 $P'(x', y', z')$，那么：

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

用齐次坐标表示，则有：

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

这就是顶视变换，对应的变换矩阵为：

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

用类似方法可以容易地得到正视图和侧视图的正交投影变换矩阵：

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

还有一种常见的正交投影，即等轴投影。选择与 3 个坐标轴的交点到原点等距的平面作为投影平面，投影方向依然选择与投影平面法向量一致，这样的投影称为等轴投影。等轴投影中，能够显示投影对象的多个侧面，并且在 3 个坐标轴方向上使对象产生相等的比例。投影对象在空间坐标系的 8 个象限中各有一个等轴投影。

2)斜交投影

投影方向不与投影平面垂直的平行投影，称为斜交投影。如图 4-32 所示，取投影平面为 $z_v=0$，空间点 $P(0,0,1)$经斜交投影后得到投影点 P'，设 P'到坐标原点 O 的距离为 l，直线 $P'O$ 与 X 轴正向夹角为 α，则容易得到 P' 坐标为($l\cos\alpha$，$l\sin\alpha$，0)。这时所做斜交投影的方向就是向量$\overrightarrow{PP'}$的方向，这个方向是 $P'=P(l\cos\alpha,l\sin\alpha,-1)$。直线 $P'P$ 与平面 $z_v=0$ 形成的夹角记为 β。

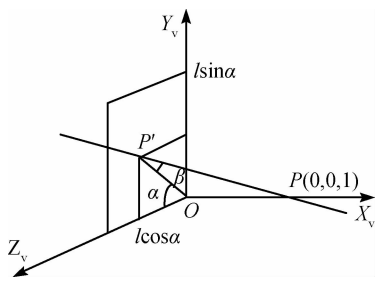

图 4-32　对点 $P(0,0,1)$作斜交投影到点 $P'(l\cos\alpha,l\sin\alpha,0)$

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

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

因此有：

$$x'=x+z\cdot(l\cdot\cos\alpha),y'=y+z\cdot(l\cdot\sin\alpha)$$

使用齐次坐标有：

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

所以作斜交投影的变换矩阵是：

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

当夹角 β 为 45°时，获得的投影为斜等侧投影，当 $\tan\beta=2$ 时，β 约为 63.4°，获得的投影为斜二侧投影，如图 4-33 所示。

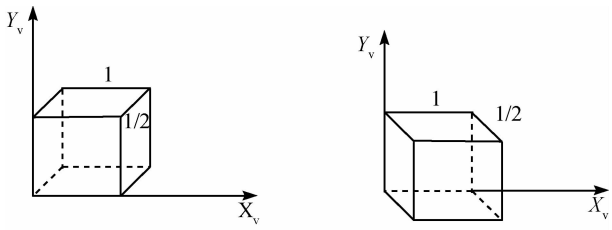

图 4-33　边长为 l 的正方体的两种斜二测投影示意图

4.7.3　透视投影

尽管场景的平行投影视图容易生成并且能够保持对象的比例不变，但是却无法提供真实感图形表达。将对象沿交于投影中心的一组直线投影到投影平面的方法即为透视投影。经过透视投影，距离投影中心远的对象投影比较小，距离投影中心近的对象投影比较大。

1）单点透视

假设投影中心（视点）为 $C(x_c,y_c,z_c)$，投影平面为 $z=0$ 平面（XOY 平面），任一三维点 $P(x_p,y_p,z_p)$，过 C 点和 P 点的直线交 $z=0$ 平面于点 $Q(x,y,z)$，则点 Q 即为点 P 在 $z=0$ 平面的投影。下面我们来讨论如何计算投影点的坐标。

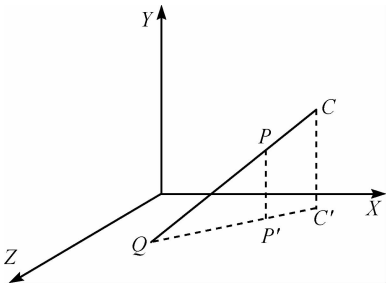

图 4-34　点 *P* 的透视投影示意图

分别从 C 点和 P 点向 $z=0$ 平面引垂线，与 $z=0$ 平面相交于 C' 和 P' 点，如图 4-34 所示。

则有 3 个点：$C'(x_c,y_c,0)$、$P'(x_p,y_p,0)$、$Q(x,y,0)$。在三角形 $CC'Q$ 中，有：

$$\frac{x-x_c}{0-z_c}=\frac{x_p-x_c}{z_p-z_c}$$

变换可得：

$$x=x_c-(x_p-x_c)\frac{z_c}{z_p-z_c};$$

使用同样的方法，可得：

$$y=y_c-(y_p-y_c)\frac{z_c}{z_p-z_c}$$

将上述公式表示成齐次坐标的形式，则有：

$$[x'\quad y'\quad z'\quad 1]=[x\quad y\quad z\quad 1]\cdot\begin{bmatrix} x_c-(x_p-x_c)\dfrac{z_c}{z_p-z_c} & 0 & 0 & 0 \\ 0 & y_c-(y_p-y_c)\dfrac{z_c}{z_p-z_c} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2)灭点

在透视投影中，平行于投影平面的直线投影后仍然平行，而与投影平面不平行的平行线组投影后会相交于一点。一组平行线投影后的交点称为灭点。每一组平行线都有自己的灭点，如图 4-35 所示。

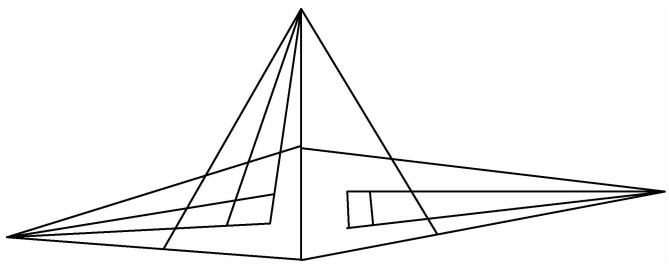

图 4-35　灭点示意图

平行于坐标轴的平行线组，其灭点称为主灭点。投影平面方向不同，主灭点数量也不同。投影平面与坐标轴的交点数(一个、两个或三个)与主灭点的数量相同，透视投影也相应地分为一点、两点或三点投影，如图 4-36 所示。

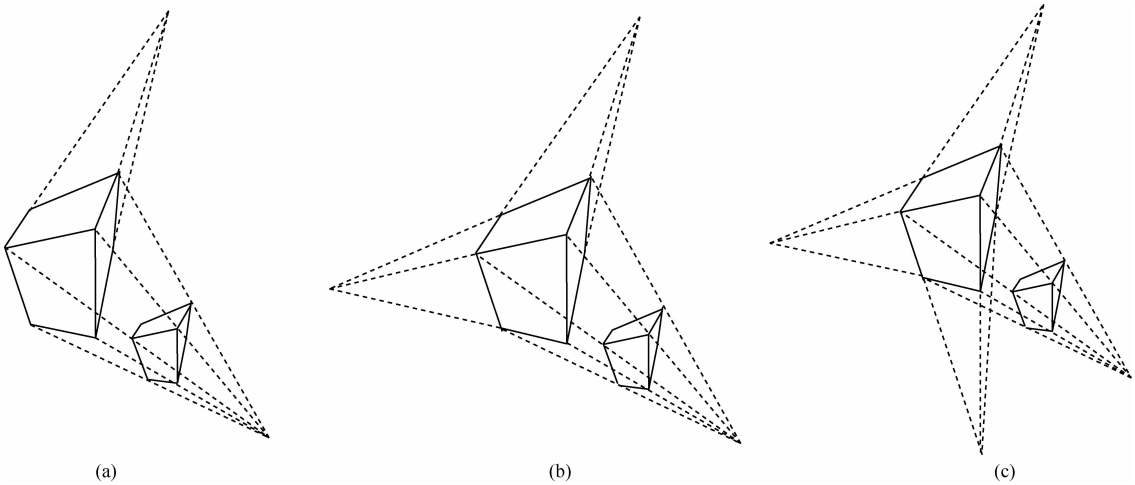

图 4-36　三种透视投影示意图

(a)一点透视投影；(b)两点透视投影；(c)三点透视投影

在实际应用中，为了获得更好的投影效果，需要改变投影平面的位置，而不能只用 $z=0$

平面作为投影平面。在投影平面是任意平面的情况下进行透视投影可以采用如下方法：设投影平面所在坐标系为 $OXYZ$，可以再建立一个过渡坐标系 $\overline{OXYZ}$，该坐标系可以成为观察坐标系，使投影平面在观察坐标系中恰好是 $z=0$ 平面。我们可以通过平移变换和旋转变换，使坐标系 $OXYZ$ 与坐标系 $\overline{OXYZ}$ 重合，求出变换矩阵，利用该变换矩阵就可以把要进行投影变换的物体变换到观察坐标系下，在观察坐标系中就可以应用我们已经得到的投影变换矩阵了。

一般来说，透视投影产生的透视投影图立体感较好，而平行投影产生的平行投影图能够较好地保留物体各部分的相对尺寸关系，在应用中可以根据需要选择不同的投影方法，这两种投影方法都有广泛的应用。

4.7.4 三维图形的裁剪

在实际应用中还会用到对三维视域的裁剪。平行投影时的视域如图 4-37(a)所示，它是由方程 $x=0$，$x=1$，$y=0$，$y=1$，$z=0$ 和 $z=1$ 所代表的 6 个平面围成的立方体。透视投影时的视域如图 4-37(b)所示，它是由方程 $x=z$，$x=-z$，$y=z$，$y=-z$，$z=z_{min}$ 和 $z=1$ 所代表的 6 个平面围成的棱台。对三维视域的裁剪就是把视域内的图形保留下来，把视域外的部分裁剪掉。

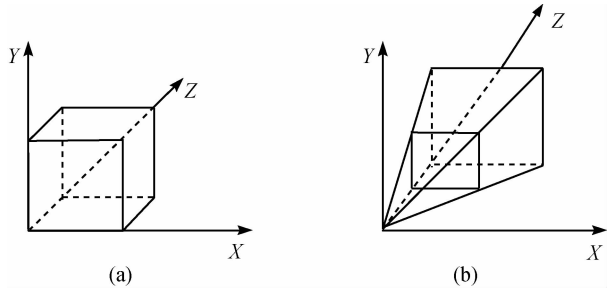

图 4-37 两种三维裁剪的视域

(a)平行投影视域；(b)透视投影视域

三维裁剪方法可以由二维裁剪方法扩展而来。如 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 6 位(见图 4-38)，这 6 位的安排是：

点在视域左面，第一位为 1，$y>1$，($y>z$)。

点在视域右面，第二位为 1，$y<0$，($y<-z$)。

点在视域下面，第三位为 1，$x>1$，($x>z$)。

点在视域上面，第四位为 1，$x<0$，($x<-z$)。

点在视域前面，第五位为 1，$z>1$，($z>1$)。

点在视域后面，第六位为 1，$z<0$，($z<z_{min}$)。

括号中的条件适用于透视投影的情况，平行投影时用括号外的条件。

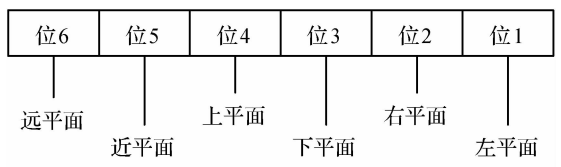

图 4-38 三维图形裁剪编码方法

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：

$$x=x_0+(x_1-x_0)t$$
$$y=y_0+(y_1-y_0)t$$
$$z=z_0+(z_1-z_0)t$$

其中参数 $t\in[0,1]$。

当视域为立方体时，直线段与视域的边界面（例如，$y=1$ 边界面）求交时，可由下式求得：

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

求得交点参数 t'，从而求得交点坐标。

当视域为棱台时，直线段和视域的边界面（例如，$x=z$ 边界面）求交时，可得出交点处的参数为：

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

从而求得交点坐标。

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 值可按如下取值：

$$Q_l=-(\Delta x+\Delta z),\qquad D_l=z_0+x_0$$
$$Q_r=(\Delta x-\Delta z),\qquad D_r=z_0-x_0$$
$$Q_b=-(\Delta y+\Delta z),\qquad D_b=y_0+z_0$$
$$Q_t=(\Delta y-\Delta z),\qquad D_t=z_0-y_0$$

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

$$Q_f=-\Delta z,\qquad D_f=z_0-z_{\min}$$
$$Q_{bk}=\Delta z,\qquad D_{bk}=1-z_0$$

其中，$y=R\sin\theta$，$y=B\sin\theta$ 和 $\Delta z=z_1-z_0$。可知，相应平面与 C' 的交点的参数值为 $x=A\cos\theta x=R\cos\theta$。

通过本章的讨论，可以得出简单的图形处理流程，如图 4-39 所示。

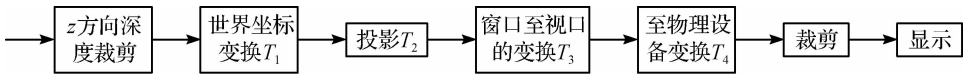

图 4-39　简单的图形处理流程

4.8 习　　题

（1）写出完成如下平面图形变换的变换矩阵：

①保持点(5,10)固定,x 方向放大 3 倍,y 方向放大 2 倍。

②绕坐标原点顺时针旋转 90°。

③对直线 $y=-x$ 成轴对称。

④沿与水平方向成 θ 角的方向扩大 S_1 倍,沿与水平方向成 $90°+\theta$ 角的方向扩大 S_2 倍。

(2)设程序中允许用 Set-Window(x_{min},x_{max},y_{min},y_{max})和 Set-Viewport(x_{min},x_{max},y_{min},y_{max})来设定二维视见变换,问:

①若用了 Set-Window(10,100,10,150)和 Set-Viewport(0,0.25,0,0.25),则在齐次坐标系中的视见变换矩阵是什么?

②显示与(1)中同样的图形,但显示图形在 x 方向上为(1)的 1.5 倍,y 方向上为(1)的 2 倍,且视见区右下角位置不变,请写出 Set-Windows 和 Set-Viewport 中的参数值。

③从(1)中显示图形选出右一半图形使它放大显示在(1)中规定的视见区上,写出 Set-Windows 和 Set-Viewport 中的参数值。

(3)求完成如下空间图形变换的变换矩阵:

①图形中点(0.5,0.2,-0.2)保持不动,x 和 y 方向放大 3 倍,z 方向不变。

②产生与原点对称的图形。

③产生对 $z=3$ 平面对称的图形。

④绕过原点和(1,1,1)的直线旋转 45°。

(4)设三维空间有一个平面,其方程为 $Ax+By+Cz+D=0$,要通过平移和旋转组合的变换,使其重合于 $z=0$ 坐标平面,求变换矩阵。

(5)设斜交平行投影方向是(l,m,n),求做这个投影的变换矩阵。

(6)修改 Cohen-Sutherland 直线裁剪算法,使其成为一个直线“开窗”算法,即指定一个窗口后,窗口内舍弃,窗口外保留。