第 3 章 几何建模

计算机图形学的早期发展动力源自于 CAD/CAE/CAM 系统中对几何造型的需求。它们需要通过计算机表示、存储、分析、控制并输出指定形状的几何模型。这里的形状既可以是单个物体的外形，也可以是由一组物体所构成的复杂场景的外形。此外，在各种类型图形的真实感绘制过程中，模型的几何形状也会直接影响其表面光照分布、纹理密度等因素，进而与模型的绘制效果也是密切相关的。因此，长期以来几何建模都是计算机图形学中的重要内容。

本章介绍几何建模涉及的数学基础知识，包括形状表达的数学形式、常用几何性质等。接下来围绕三种重要的几何建模技术：自由曲线/曲面建模、细分曲面建模和三维重建，介绍相应的概念、模型及其使用方法。最后，针对计算机内部数据存储特点，介绍面向几何建模的典型数据结构。

3.1 数学基础

一般来讲，计算机图形学中的几何建模是指构造物体或场景二维/三维形状的数学表达形式及其在计算机内表示的数据结构。通常意义下，这种数学形式具有直接、明确的函数关系，使得所有满足该关系的形状也具有相应的几何性质。因此，首先需要了解表达几何形状的数学形式。

3.1.1 几何形状的数学形式

数学上，连续几何形状的表达方式主要有三种：显式表达式、隐式表达式和参数表达式。其中，显式表达式和隐式表达式又称为代数形式。这三种形式的主要区别在于定义函数时，其因变量和自变量的表达方式和相互之间代数制约关系的不同。下面分别介绍这三种表达式。

1. 显式表达式

显式表达式是一种因变量随自变量变化而变化的函数形式。二维平面上几何形状的显式表达式通常为 $y=f(x)$，其中，x 是自变量，y 是因变量。例如，$y=2x+1$ 代表平面直线，$y=x^2+x+1$ 代表平面抛物线。三维空间中几何形状的显式表达式通常为 $z=f(x,y)$，其中，x 和 y 是自变量，z 是因变量。例如，$z=2x+y+1$ 表示了三维空间中的一张平面，$z=\sqrt{1-x^2-y^2}$ 表示了半径为 1 的半球面。从表达形式上看，显式表达式的因变量和自变量分别位于等号两侧，二者具有明显的对应关系。

2. 隐式表达式

隐式表达式是由多个变量共同定义的一种函数形式。这里的变量是没有自变量和因变量之分的，不能单独进行表示。二维平面上几何形状的隐式表达式通常具有 $f(x,y)=0$ 的形式。例如，$x^2+y^2-1=0$ 表示了平面圆形。三维空间中几何形状的隐式表达式具有 $f(x,y,z)=0$ 的形式。例如，$x^2+y^2+z^2-1=0$ 表示了三维球面。从表达形式上看，隐式表达式的因变量和自变量是混合在一起的，都位于等式的同侧，二者没有明显的对应关系。

3. 参数表达式

参数表达式是采用若干独立变量作为自变量的显式表达式组成的集合。这种表达方式下几何形状是由指定集合的参数作为自变量而得到的因变量所表示的。作为自变量的参数也称为参变量。例如，二维平面和三维空间中的几何形状分别具有如下表达形式：

$$\begin{cases} x=f(t) \\ y=g(t) \end{cases} \qquad \begin{cases} x=f(u,v) \\ y=g(u,v) \\ z=h(u,v) \end{cases} \tag{3.1}$$

其中，t、u 和 v 是参变量。它们在给定取值范围内变化，从而产生相应的几何形状。

从代数形式来讲，参数表达式也可以看作为一种显式表达式。但是，参数表达式具有更加直观的几何意义，反映了参变量变化所产生的各个维度的变化情况。例如，曲线可以看作单个参变量在指定区间内变化所形成的空间点的集合；而曲面则可以看作双参变量变化形成的空间点的集合。此外，参数表达式的代数和微积分运算非常方便，适合对几何形状的性质进行分析。

对于一些几何形状，可能存在多种不同形式的表达式。如图 3.1 所示，三维空间中半径为 1 的球面，除了上述隐式表达式，也可以采用基于球极坐标的参数表达式，也就是

$$\begin{cases} x=\sin\varphi\cos\theta \\ y=\sin\varphi\sin\theta \\ z=\cos\varphi \end{cases} \tag{3.2}$$

其中，参变量 θ 和 φ 分别对应于方位角和俯仰角。但是对于球面而言，一般不存在直接的显式表达式。

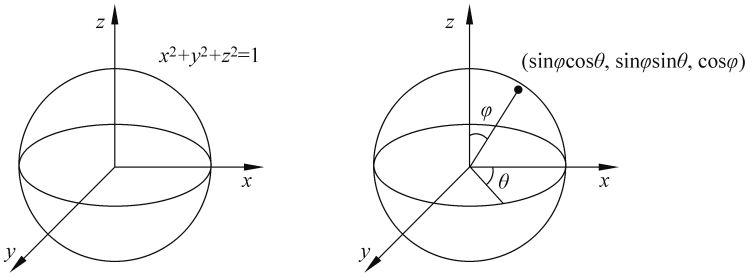


图 3.1　球面的隐式表达式与参数表达式

例题 3-1　平面曲线的数学表达式。

问题：如图 3.2 所示，平面坐标系第一象限内的单位圆弧的显式表达式和参数表达式分别是什么？

解答：该圆弧的显式表达式可以写为：

$$y=\sqrt{1-x^2},\quad 0\leqslant x\leqslant 1$$

对应的参数表达式可以写为：

$$\begin{cases}x(t)=(1-t^2)/(1+t^2)\\ y(t)=(2t)/(1+t^2)\end{cases},\quad 0\leqslant t\leqslant 1$$

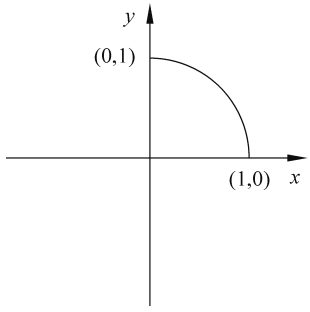


图 3.2　平面上的单位圆弧

□

3.1.2　几何性质

通过几何形状的数学形式，可以研究模型所具备的几何性质，从而更好地设计和控制建模效果。计算机图形学中几何建模所涉及的形状主要包括两种类型：曲线和曲面。曲线又包括二维平面曲线和三维空间曲线，而曲面主要是指三维空间中的曲面。这些也是在日常生产生活中，人们接触最多的物体形状。

1. 二维平面曲线

二维平面曲线理论上可以采用显式、隐式或参数形式进行表达。显式形式是具有一个自变量和一个因变量的表达式，也就是 $y=f(x)$。隐式形式是由两个变量构成的方程式，也就是 $f(x,y)=0$。而参数形式则是单参变量表示的二维向量，也就是 $\gamma(t)=(x(t),y(t))$。例如，平面上椭圆的隐式表达式和参数表达式分别具有如下形式：

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1,\quad \begin{cases}x=a\cos\theta\\ y=b\sin\theta\end{cases}\tag{3.3}$$

其中，a 和 b 是非零常数，θ 是参变量。平面曲线常用的几何性质有曲线弧长、曲率等，反映了曲线本身的形状。

以参数形式表达的曲线为例，它的弧长描述了曲线在某一参变量取值区间内的测度，计算公式为

$$l=\int_{t_0}^{t_1}\sqrt{\dot{x}^2(t)+\dot{y}^2(t)}\,\mathrm{d}t\tag{3.4}$$

其中，$\dot{x}(t)$ 和 $\dot{y}(t)$ 是函数关于参变量 t 的一阶导数，$[t_0,t_1]$ 是 t 的取值区间。曲率定义为切线方向角相对于弧长的变化率，计算公式为

$$\kappa(t)=\frac{|\dot{x}(t)\ddot{y}(t)-\ddot{x}(t)\dot{y}(t)|}{(\dot{x}^2(t)+\dot{y}^2(t))^{\frac{3}{2}}}\tag{3.5}$$

其中，$\ddot{x}(t)$ 和 $\ddot{y}(t)$ 是函数关于参变量 t 的二阶导数。实际上，曲率反映了曲线在某一点的弯曲程度，例如直线的曲率处处为 0。此外，平面曲线上某一点的曲率大小等于该点处密切圆半径 r 的倒数，而密切圆的圆心位于曲线凹向的一侧，如图 3.3(a)所示。

2. 三维空间曲线

三维空间曲线可以采用单个参变量作为参数的三维向量进行表示，也就是 $\boldsymbol{\gamma}(t)=(x(t),y(t),z(t))$。它直观上反映了质点在三维空间中单自由度的运动轨迹。这里需

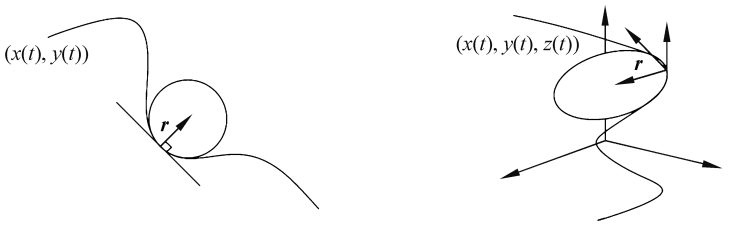


图 3.3　平面二维曲线和三维空间曲线的曲率圆

要注意的是，三维空间曲线一般不存在显式表达式，而其隐式表达式由两张曲面的隐式表达式组成的方程组所构成，可以记为

$$\begin{cases} f(x,y,z)=0 \\ g(x,y,z)=0 \end{cases} \tag{3.6}$$

它反映了三维空间曲面可以看作两张曲面相交而产生。弧长、曲率和挠率是空间曲线的三种基本属性，反映了三维曲线的形状。与平面曲线类似，弧长表示了指定区间内的曲线长度，计算公式为

$$l=\int_{t0}^{t1}\sqrt{\dot{x}^2(t)+\dot{y}^2(t)+\dot{z}^2(t)}\,\mathrm{d}t \tag{3.7}$$

曲率反映了曲线在空间中的弯曲程度，计算公式为

$$\kappa(t)=\frac{|\dot{\boldsymbol{\gamma}}(t)\times\ddot{\boldsymbol{\gamma}}(t)|}{(|\dot{\boldsymbol{\gamma}}(t)|)^{\frac{3}{2}}} \tag{3.8}$$

其中，$\dot{\boldsymbol{\gamma}}(t)$ 和 $\ddot{\boldsymbol{\gamma}}(t)$ 分别是曲线所对应三维向量的一阶和二阶导数。如图 3.3(b)所示，曲率也与该点密切圆半径 r 成反比关系。与平面曲线不同，空间曲线具有挠率属性，计算公式为

$$\tau(t)=\frac{(\dot{\boldsymbol{\gamma}}(t)\times\ddot{\boldsymbol{\gamma}}(t))\cdot\dddot{\boldsymbol{\gamma}}(t)}{|\dot{\boldsymbol{\gamma}}(t)\times\ddot{\boldsymbol{\gamma}}(t)|^2} \tag{3.9}$$

挠率反映了曲线切平面的扭转状况。例如，所有平面曲线的挠率都为 0。

3. 三维空间曲面

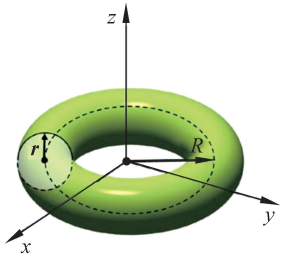


图 3.4　环面

三维空间曲面理论上可以采用显式、隐式或参数形式进行表达。显式形式是具有两个自变量和一个因变量的表达式，通常记为 $z=f(x,y)$。隐式形式是由三个变量构成的方程式，也就是 $f(x,y,z)=0$。参数形式则是由两个参变量表示的三维向量形式，通常记为 $\boldsymbol{\pi}(u,v)=(x(u,v),y(u,v),z(u,v))$。例如，图 3.4 所示环面的隐式表达式和参数表达式分别为：

$$(x^2+y^2+z^2+R^2-r^2)^2-4R^2(x^2+y^2)=0,\quad \begin{cases} x(u,v)=(R+r\cos v)\cos u \\ y(u,v)=(R+r\cos v)\sin u \\ z(u,v)=r\sin v \end{cases} \tag{3.10}$$

其中，R 和 r 分别表示环截面半径和内环半径。

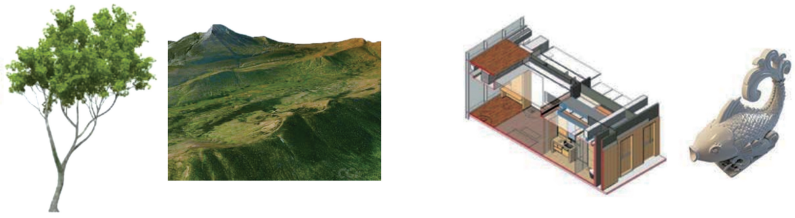
空间曲面常用的几何性质涉及面积、法曲率、主曲率、高斯曲率、平均曲率等属性。以参数形式表示的曲面为例，其面积是指双参变量在取值范围内的表面测度，计算公式为

$$s = \int_{u_0}^{u_1}\int_{v_0}^{v_1} \left| \dot{\boldsymbol{\pi}}_u \times \dot{\boldsymbol{\pi}}_v \right| \mathrm{d}u\,\mathrm{d}v \tag{3.11}$$

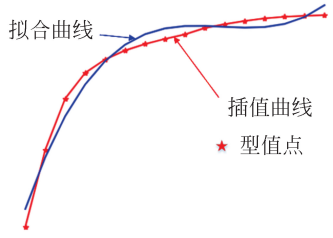
其中，$[u_0, u_1]$和$[v_0, v_1]$分别是参变量 u 和 v 的取值范围。曲面上某点处的曲率和经过该点的曲面上曲线的弯曲程度相关，称为法曲率。事实上，曲面在一点处有无穷多个切方向，因此可以定义无穷多个法曲率。其中，法曲率的最大值和最小值称为主曲率，代表了该点处法曲率的极值分布情况，也就是能达到的最大和最小弯曲程度。曲面上某点处主曲率的平均值，称为平均曲率，而主曲率的乘积则称为高斯曲率。事实上，平均曲率和高斯曲率是反映曲面局部形状的两个重要属性。

3.1.3　建模工具

根据建模对象的不同，几何建模可以简单地分为自然物体建模和人造物体建模。所谓自然物体建模，是指建模对象来自于自然界中正常存在的物体，例如各种动物、植物、地形、地貌等，如图 3.5(a)所示。这类物体的几何形状往往很难用精确的数学公式直接表示。人造物体建模，是指通过手工交互或自动化的方式，利用相应的工具对基本几何形状进行变换和重组，获得具有新形状的物体，如图 3.5(b)所示。这类物体的几何形状，通常可以借助变换或重组时的数学表达式进行准确描述。



(a) 自然物体建模　　(b) 人造物体建模



(c) 型值点的插值与拟合

图 3.5　物体对象几何建模的方式：插值与拟合

无论自然物体建模，还是人造物体建模，主要有插值和拟合两种方式，如图 3.5(c)所示。这两种方式都是先对物体形状有个大概的主观描述，在此基础上按照精度的要求进行形状的构建。插值，是要求形状能够严格满足给定数据点的几何位置约束。这些数据

点也称为型值点，是对建模对象外形的离散采样。拟合则是在一定的几何意义下，建模的形状尽可能逼近给定的型值点。不论对形状进行插值或拟合，常用的数学工具有自由曲线/曲面、细分曲面等，它们为自然物体建模和人造物体建模提供了有效的数学工具。

3.2 自由曲线/曲面建模

经典几何形状，如平面、圆柱面、圆锥面等都具有明确的解析表达式。然而，绝大部分的自然物体，例如人脸等，其形状往往无法使用一个解析函数进行显式表示，需要构造新的函数形式来表示其形状。所谓自由曲线/曲面，是指不能用初等解析函数完全准确地表达全部形状的曲线和曲面。因此，自由曲线/曲面的出现，是为了克服使用解析函数进行形状表示的局限性，采用更广泛意义下的数学函数来表示形状。

3.2.1 平面三次多项式曲线

多项式曲线是一种最常见的平面曲线表达形式。在给定型值点集合后，计算 n 次多项式进行插值或拟合。这样就可以转化为线性方程组来求解该多项式。数学上，多项式次数越高，其对应曲线出现的拐点就越多，能表示的形状也就越复杂。但是次数越高，越可能在拟合时出现过拟合，难以控制那些没有采样点处的形状。此外，五次以上多项式的计算比较复杂，不利于快速建模。如果多项式次数较低，又会出现欠拟合情况，导致形状的表现力不足。在实际应用中，三次多项式曲线是使用较广泛的一类多项式曲线。它对应的参数表达式为：

$$\boldsymbol{p}(t)=\boldsymbol{c}_0+\boldsymbol{c}_1 t+\boldsymbol{c}_2 t^2+\boldsymbol{c}_3 t^3 \tag{3.12}$$

其中，$\boldsymbol{c}_i=(x_i,y_i)^{\mathrm{T}}$ 称为控制顶点，$t\in[0,1]$是参变量取值范围。通过公式(3.12)可以看出，$\{\boldsymbol{c}_i\}$取值不同，就会产生不同的曲线形状。

如图 3.6 所示，给定 4 个型值点$\{\boldsymbol{p}_1,\boldsymbol{p}_2,\boldsymbol{p}_3,\boldsymbol{p}_4\}$，可以计算通过这些型值点的一条三次多项式曲线。这是由于三次多项式总共有 4 个系数，每个系数同时有 x 和 y 两个分量，因此一共有 8 个未知数。而 4 个型值点恰好可以提供 8 个约束条件。如果给定的型值点多于 4 个，那么可以通过最小二乘法计算一条最优的三次多项式曲线来拟合这些型值点，使得它们距离曲线的总体几何距离最小。

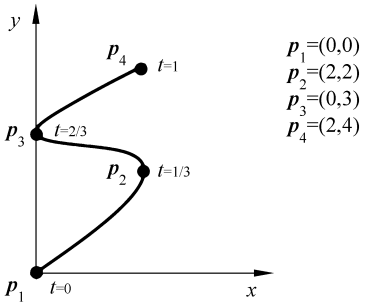


图 3.6 由 4 个型值点定义的三次多项式曲线

例题 3-2　平面三次多项式插值曲线。

问题：计算图 3.6 所示的多项式曲线的参数表达式 $\boldsymbol{p}(t)$。

解答：根据 4 个型值点$\{\boldsymbol{p}_1,\boldsymbol{p}_2,\boldsymbol{p}_3,\boldsymbol{p}_4\}$的 x 和 y 坐标，分别计算 $\boldsymbol{p}(t)=(x(t),y(t))$控制顶点$\boldsymbol{c}_i=(x_i,y_i)^{\mathrm{T}}$。其中，$x$ 和 y 坐标分别满足如下等式：

$$\begin{cases}0=x_0\\2=x_0+\dfrac{1}{3}x_1+\dfrac{1}{9}x_2+\dfrac{1}{27}x_3\\0=x_0+\dfrac{2}{3}x_1+\dfrac{4}{9}x_2+\dfrac{8}{27}x_3\\2=x_0+x_1+x_2+x_3\end{cases}\qquad\begin{cases}0=y_0\\2=y_0+\dfrac{1}{3}y_1+\dfrac{1}{9}y_2+\dfrac{1}{27}y_3\\3=y_0+\dfrac{2}{3}y_1+\dfrac{4}{9}y_2+\dfrac{8}{27}y_3\\4=y_0+y_1+y_2+y_3\end{cases}$$

通过求解上述两个线性方程组，可以得到 4 个控制顶点的坐标分别是 $\boldsymbol{c}_0=(0,0)$、$\boldsymbol{c}_1=(20,8.5)$、$\boldsymbol{c}_2=(-54,-9)$和 $\boldsymbol{c}_3=(36,4.5)$。

□

平面三次多项式能够通过插值或拟合的方式对平面曲线建模，但是像公式(3.12)这样的表达式缺少直观的几何解释(例如，其控制顶点无法直接描述曲线形状)，这样就不利于工程人员按照主观意图进行曲线建模来设计相应的形状。此外，受单个多项式代数性质的影响，能有效表示的曲线形状有限，并不适合更加丰富多样的几何外形设计。因此，需要更灵活的自由曲线/曲面建模技术，典型的建模技术有 Bézier 曲线/曲面和 B 样条曲线/曲面。

3.2.2　Bézier 曲线/曲面

Bézier 曲线是以法国工程师 Pierre Bézier 命名的。而 Bézier 曲线的计算方法最早可以追溯到法国雷诺公司工程师 Paul de Casteljau。他提出了适用于计算机编程的递归算法，只是没有意识到这种算法最后所生成的曲线形状就是 Bézier 曲线的形状。1962 年，Bézier 明确给出了这种曲线的数学公式，并成功地应用于汽车外形设计。

Bézier 曲线是第一种在工业界被广泛采用的自由曲线建模工具，其本质也是多项式曲线。但是，这种曲线具有更直观的几何表达形式，能够方便工程师通过操纵控制顶点来进行建模。具体地讲，n 次 Bézier 曲线是一种单参变量的参数表达式，数学上定义为：

$$\boldsymbol{p}(t)=\sum_{i=0}^{n}B_i^n(t)\boldsymbol{c}_i=\sum_{i=0}^{n}\binom{n}{i}(1-t)^{n-i}t^i\boldsymbol{c}_i,\quad t\in[0,1]\tag{3.13}$$

其中，$\{\boldsymbol{c}_0,\boldsymbol{c}_1,\cdots,\boldsymbol{c}_n\}$是 $n+1$ 个控制顶点。这些控制顶点依次连接形成控制多边形。单参变量函数 $B_i^n(t)$也称为 Bernstein 多项式函数。图 3.7 分别展示了二次和三次 Bézier

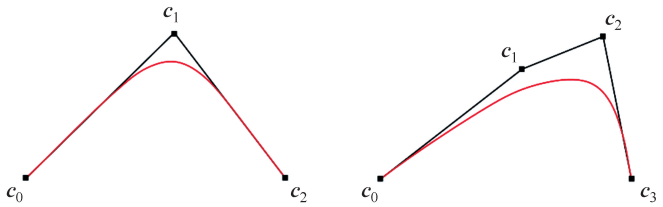


图 3.7　二次和三次 Bézier 曲线

曲线。

数学上，多项式函数空间是可以采用 Bernstein 函数作为一组基函数。因此，Bézier 曲线就是采用这组基函数的线性组合来表示的一类多项式函数。在此基础上，可以将公式(3.12)所表示的任意形式的多项式曲线转化为公式(3.13)表示的 Bézier 曲线。这个过程实质上就是将多项式基函数$\{1,t,t^2,t^3\}$转化为 Bernstein 基函数表示。

Bézier 曲线之所以在工业设计中广受欢迎，主要是因为公式(3.12)定义的几何形状具有以下良好性质，能够方便设计人员对形状进行控制。

(1) 首末端点插值。曲线经过控制多边形的首末端点，也就是曲线端点满足 $\boldsymbol{p}(0)=\boldsymbol{c}_0$ 和 $\boldsymbol{p}(1)=\boldsymbol{c}_n$。此外，曲线在两个端点处的切线方向与控制多边形的边平行，也就是 $\boldsymbol{p}'(0)$与线段$\overline{\boldsymbol{c}_0\boldsymbol{c}_1}$的方向一致，而 $\boldsymbol{p}'(1)$与线段$\overline{\boldsymbol{c}_{n-1}\boldsymbol{c}_n}$的方向一致。这使得 Bézier 曲线插值首末端点的切向量。

(2) 保凸性。曲线位于控制多边形构成的凸包(凸多边形边界)内。这样在给定控制多边形后，就可以通过凸包来限定生成的 Bézier 曲线的范围。例如，图 3.7 所示的二次 Bézier 曲线位于三角形$\triangle\boldsymbol{c}_0\boldsymbol{c}_1\boldsymbol{c}_2$ 的内部，而三次 Bézier 曲线则位于四边形$\square\boldsymbol{c}_0\boldsymbol{c}_1\boldsymbol{c}_2\boldsymbol{c}_3$ 的内部。这个性质方便设计人员通过控制顶点的位置来对曲线形状进行调控。

(3) de Casteljau 递归算法。在计算公式(3.13)表示的 Bézier 曲线时，除了直接采用多项式的代数运算，还可以借助 de Casteljau 递归算法实现更加快速地计算。如图 3.8 所示。该递归算法基于以下递推公式：

$$\begin{cases}\boldsymbol{p}_i^{[r]}(t)=(1-t)\cdot\boldsymbol{p}_{i-1}^{[r-1]}(t)+t\cdot\boldsymbol{p}_i^{[r-1]}(t)\\ \boldsymbol{p}_i^{[0]}(t)\equiv\boldsymbol{p}_i^{[0]}=\boldsymbol{c}_i\end{cases}\tag{3.14}$$

上述 de Casteljau 递归算法证明了当 $r=n$ 时，$\boldsymbol{p}_n^{[n]}(t)$一定是位于 Bézier 曲线上的点。这种递归算法非常适合计算机编程实现，计算效率很高。

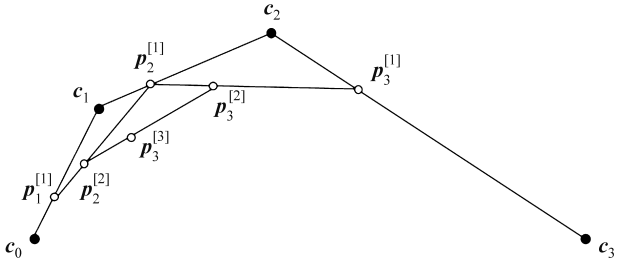


图 3.8　de Casteljau 递归过程

例题 3-3　平面三次 Bézier 曲线。

问题：写出图 3.9 所示的控制顶点定义的三次 Bézier 曲线，其中 4 个控制顶点坐标分别是 $\boldsymbol{c}_0(-2,0)$，$\boldsymbol{c}_1(-1,1)$，$\boldsymbol{c}_2(1,1.5)$，$\boldsymbol{c}_3(2,0)$。然后，给出采用 de Casteljau 递归算法计算曲线上任意一点的递归过程。

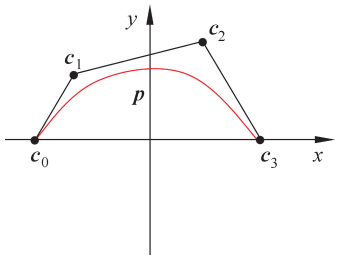


图 3.9　三次 Bézier 曲线

解答：将 4 个控制顶点的坐标代入公式(3.13)，可以得到如下相应的 Bézier 曲线表达式：

$$\begin{aligned}\boldsymbol{p}(t)&=\sum_{i=0}^{3}B_i^3(t)\boldsymbol{c}_i\\&=\begin{pmatrix}-2B_0^3(t)-B_1^3(t)+B_2^3(t)+2B_3^3(t)\\B_1^3(t)+1.5B_2^3(t)\end{pmatrix}\\&=\begin{pmatrix}-2(1-t)^3-3(1-t)^2t+3(1-t)t^2+2t^3\\3(1-t)^2t+4.5(1-t)t^2\end{pmatrix}\end{aligned}$$

其中，$t\in[0,1]$。进一步，根据公式(3.14)定义的 de Casteljau 递归算法，对于曲线上任意一点 $\boldsymbol{p}(t_0)$，$0\leqslant t_0\leqslant 1$，其递归计算过程如下：

$$\boldsymbol{p}(t_0)=\boldsymbol{p}_3^{[3]}(t_0)=(1-t_0)\boldsymbol{p}_2^{[2]}(t_0)+t_0\boldsymbol{p}_3^{[2]}(t_0)$$

$$\begin{cases}\boldsymbol{p}_2^{[2]}(t_0)=(1-t_0)\boldsymbol{p}_1^{[1]}(t_0)+t_0\boldsymbol{p}_2^{[1]}(t_0)\\\boldsymbol{p}_3^{[2]}(t_0)=(1-t_0)\boldsymbol{p}_2^{[1]}(t_0)+t_0\boldsymbol{p}_3^{[1]}(t_0)\end{cases}$$

$$\begin{cases}\boldsymbol{p}_1^{[1]}(t_0)=(1-t_0)\boldsymbol{p}_0^{[0]}(t_0)+t_0\boldsymbol{p}_1^{[0]}(t_0)\\\boldsymbol{p}_2^{[1]}(t_0)=(1-t_0)\boldsymbol{p}_1^{[0]}(t_0)+t_0\boldsymbol{p}_2^{[0]}(t_0)\\\boldsymbol{p}_3^{[1]}(t_0)=(1-t_0)\boldsymbol{p}_2^{[0]}(t_0)+t_0\boldsymbol{p}_3^{[0]}(t_0)\end{cases}$$

其中，$\boldsymbol{p}_0^{[0]}(t_0)=\boldsymbol{c}_0$，$\boldsymbol{p}_1^{[0]}(t_0)=\boldsymbol{c}_1$，$\boldsymbol{p}_2^{[0]}(t_0)=\boldsymbol{c}_2$ 以及 $\boldsymbol{p}_3^{[0]}(t_0)=\boldsymbol{c}_3$。

□

进一步，Bézier 曲面是由两个参变量的 Bernstein 混合函数表示的参数曲面。具体来讲，$m\times n$ 次 Bézier 曲面是由$(m+1)\times(n+1)$个控制顶点组成的多面体(也称为控制网格)来定义，记为：

$$\boldsymbol{p}(s,t)=\sum_{i=0}^{m}\sum_{j=0}^{n}B_i^m(s)B_j^n(t)\boldsymbol{c}_{ij},\quad s\in[0,1],\ t\in[0,1]\tag{3.15}$$

与 Bézier 曲线类似，Bézier 曲面也具有很多良好的性质。例如，Bézier 曲面的边界是由 4 个边界多边形作为控制多边形所定义的 4 条 Bézier 曲线；角点插值性，也就是说 Bézier 曲面在 4 个角点插值控制网格的顶点；凸包性，也就是说 Bézier 曲面位于曲面控制网格形成的凸包内。

与一般的平面多项式曲线/曲面表达式相比，Bézier 曲线/曲面的主要优点是容易编程实现、具有端点和切向插值等性质，而且方便各种微积分运算的数值求解。但 Bézier 曲线/曲面的缺点也很明显，就是缺乏对曲线形状的局部可控性。具体来讲，改变其中一个控制顶点的位置，就会改变整个曲线/曲面的形状。例如，图 3.10 所示的四次 Bézier 曲线，当其他控制顶点位置不变，只要控制顶点 $\boldsymbol{c}_2$ 移动到 $\boldsymbol{c}_2'$的位置，整条曲线上除 $\boldsymbol{c}_0$ 和 $\boldsymbol{c}_4$ 两个端点外，其余点的位置都会发生变化。这种局限性也限制了 Bézier 曲线/曲面的几

何建模能力。

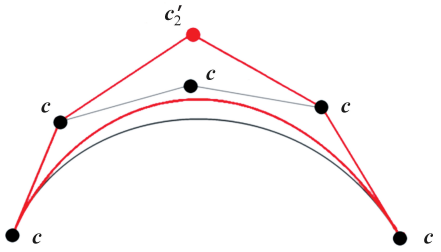


图 3.10　Bézier 曲线形状缺乏局部可控性

3.2.3　B 样条曲线/曲面

为了克服 Bézier 曲线/曲面的不足，美国 Utah 大学的 William Gordon 和 Richard Riesenfeld 在 1974 年将 B 样条参数曲线引入几何建模。样条，源于生产实践，本意是指富有弹性的细长条。样条利用压铁使其通过指定的型值点，并调整样条使它具有满意的形状，然后沿样条画出曲线。B 样条曲线则是通过 B 样条函数表示的曲线，本质上是分段多项式曲线。因此，B 样条曲线是由多个在连接处满足一定连续性的多项式曲线所构成的曲线，是一类多项式组合的曲线。在此之前，美国 Wisconsin-Madison 大学的 Isaac Schoenberg 早在 1946 年就利用 B 样条进行统计数据的光滑处理，开创了样条逼近的现代理论。

由于分段多项式的性质，B 样条曲线的定义除了需要控制顶点 $\{\boldsymbol{c}_0,\boldsymbol{c}_1,\cdots,\boldsymbol{c}_n\}$，还需要设置节点向量 $t_0\leqslant t_1\leqslant\cdots\leqslant t_{n+k+1}$。在此基础上，$k$ 次（$k+1$ 阶）B 样条曲线定义为：

$$\boldsymbol{p}(t)=\sum_{i=0}^{n}N_i^k(t)\boldsymbol{c}_i,\quad n\geqslant k \tag{3.16}$$

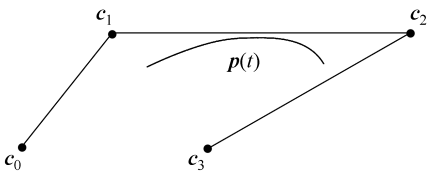


图 3.11　三次 B 样条曲线

这里，相邻两个节点形成的区间 $[t_k,t_{k+1})$ 定义了在其每一段上的多项式函数，而这些多项式函数的组合就定义了 B 样条曲线。图 3.11 展示了一条三次 B 样条曲线。

具体来讲，在公式(3.16)中，$N_i^k(t)$ 是 B 样条基函数，可以通过如下递推公式来定义：

$$\begin{cases}N_i^0(t)=\begin{cases}1, & t\in[t_i,t_{i+1})\\0, & t\notin[t_i,t_{i+1})\end{cases}\\[2ex]N_i^k(t)=\dfrac{t-t_i}{t_{i+k}-t_i}N_i^{k-1}(t)+\dfrac{t_{i+k+1}-t}{t_{i+k+1}-t_{i+1}}N_{i+1}^{k-1}(t)\end{cases} \tag{3.17}$$

公式(3.17)称为 de Boor-Cox 公式。由公式(3.17)可以看出，基函数 $N_i^k(t)$ 只有在位于节点 t_i 和 t_{i+k+1} 之间的区间内取非负值，而在其他处则取值为零。这也意味着在区间 $[t_i,t_{i+1})$ 上，总共只有 $k+1$ 个 B 样条基函数取非负值，它们依次是 $N_{i-k}^k(t),N_{i-k+1}^k(t),\cdots,N_i^k(t)$。基于该递归公式，B 样条曲线就可以采用类似 Bézier 曲线的递归方式进行计

算，也就是通过逐次迭代的方式计算公式(3.16)表示的 B 样条曲线上任意点的位置坐标。这种计算方式称为 de Boor 递归算法。

该递归算法与 Bézier 曲线的 de Casteljau 递归算法类似，都是从控制顶点组成的控制多边形 $\boldsymbol{c}_{j-k}\boldsymbol{c}_{j-k+1}\cdots\boldsymbol{c}_j$ 开始，依次执行 k 次割角操作。其中，如图 3.12 所示第 r 次割角是用线段 $\boldsymbol{p}_i^{[r]}(t)\boldsymbol{p}_{i+1}^{[r]}(t)$ 割去角 $\boldsymbol{p}_i^{[r-1]}$。这里 $\boldsymbol{p}_i^{[r]}(t)=(1-\lambda_{i,k-r+1})\boldsymbol{p}_{i-1}^{[r-1]}(t)+\lambda_{i,k-r+1}\boldsymbol{p}_i^{[r-1]}(t)$，而割角时线段端点在控制多边形边上的比例是 $\lambda_{i,k-r+1}=(t-t_i)/(t_{i+k-r+1}-t_i)$。那么，最后得到的角点 $\boldsymbol{p}_j^{[k]}(t)$ 就是 B 样条曲线上的点 $\boldsymbol{p}(t)$。整个递归割角过程可以用如下公式表示：

$$\boldsymbol{p}(t)=\sum_{i=j-k}^{j}N_i^k(t)\boldsymbol{p}_i^{[0]}=\sum_{i=j-k+1}^{j}N_i^{k-1}(t)\boldsymbol{p}_i^{[1]}\cdots=\sum_{i=j-1}^{j}N_i^1(t)\boldsymbol{p}_i^{[k-1]}\cdots=\boldsymbol{p}_j^{[k]} \tag{3.18}$$

其中，$\boldsymbol{p}_i^{[0]}=\boldsymbol{c}_i$ 是控制顶点。事实上，该 B 样条曲线是定义在区间 $[t_j,t_{j+1})$ 上的一条多项式曲线。该区间也是公式(3.18)中能够使得所有 B 样条基函数取非零值的区间。

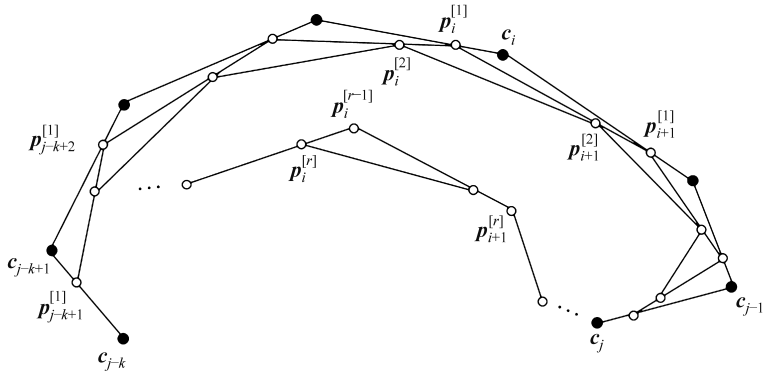


图 3.12　de Boor 递归过程

B 样条曲线保留了 Bézier 曲线良好的几何性质，同时具备了对曲线形状的局部可控性。具体来讲，根据 B 样条基函数的局部支撑性，改动其中一个控制顶点，B 样条曲线上仅仅和该控制顶点相关的曲线形状发生变化。这可以从 B 样条曲线的定义公式(3.16)得出。例如当控制顶点 $\boldsymbol{c}_i$ 的位置变化时，只有 B 样条基函数 $N_i^k(t)$ 不为零值的区间上对应的 B 样条曲线才发生改变，也就是在区间 $[t_i,t_{i+k+1})$ 以外的 B 样条曲线不会发生任何改变。此外，在进行多个 B 样条曲线拼接时，也可以根据节点设置很容易地保持拼接时的几何连续性。因此，B 样条曲线具有更加灵活的几何建模能力。

例题 3-4　平面三次 B 样条曲线。

问题： 如图 3.13 所示的三次均匀 B 样条曲线 $\boldsymbol{p}(t)$ 的控制顶点分别是 $\boldsymbol{c}_0$、$\boldsymbol{c}_1$、$\boldsymbol{c}_2$、$\boldsymbol{c}_3$，节点间距相等，设为 $t_i=i$。写出 $t\in[t_3,t_4)$ 区间上的 B 样条曲线表达式。然后，给出采用 de Boor-Cox 递归算法计算曲线上任意一点的递归过程。

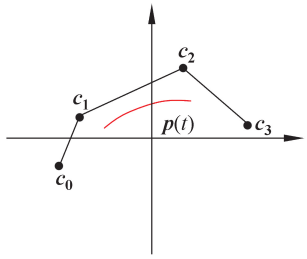


图 3.13　三次均匀 B 样条曲线

解答： 根据公式(3.17)中 B 样条基函数的定义，可以得到区间 $[t_3,t_4)$ 上 4 个非零基函数的表达式：

$$N_0^3(t)=(-t^3+3t^2-3t+1)/6, N_1^3(t)=(3t^3-6t^2+4)/6$$
$$N_2^3(t)=(-3t^3+3t^2+3t+1)/6, N_3^3(t)=t^3/6$$

因此，对应的三次均匀 B 样条曲线可以写为：

$$\boldsymbol{p}(t)=\sum_{i=0}^{3}N_i^3(t)\boldsymbol{c}_i=\frac{1}{6}(t^3,t^2,t^1,1)\boldsymbol{M}_{4\times4}\cdot\begin{bmatrix}\boldsymbol{c}_0\\ \boldsymbol{c}_1\\ \boldsymbol{c}_2\\ \boldsymbol{c}_3\end{bmatrix}$$

$$\boldsymbol{M}_{4\times4}=\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0\end{bmatrix}$$

对于曲线上任意一点 $\boldsymbol{p}(t_0)$，$3\leqslant t_0<4$，其递归计算过程如下：

$$\boldsymbol{p}(t_0)=\boldsymbol{p}_3^{[3]}(t_0)=(4-t_0)\boldsymbol{p}_2^{[2]}(t_0)+(t_0-3)\boldsymbol{p}_3^{[2]}(t_0)$$

$$\begin{cases}\boldsymbol{p}_2^{[2]}(t_0)=\dfrac{4-t_0}{2}\boldsymbol{p}_1^{[1]}(t_0)+\dfrac{t_0-2}{2}\boldsymbol{p}_2^{[1]}(t_0)\\ \boldsymbol{p}_3^{[2]}(t_0)=\dfrac{5-t_0}{2}\boldsymbol{p}_2^{[1]}(t_0)+\dfrac{t_0-3}{2}\boldsymbol{p}_3^{[1]}(t_0)\end{cases}$$

$$\begin{cases}\boldsymbol{p}_1^{[1]}(t_0)=\dfrac{4-t_0}{3}\boldsymbol{p}_0^{[0]}(t_0)+\dfrac{t_0-1}{3}\boldsymbol{p}_1^{[0]}(t_0)\\ \boldsymbol{p}_2^{[1]}(t_0)=\dfrac{5-t_0}{3}\boldsymbol{p}_1^{[0]}(t_0)+\dfrac{t_0-2}{3}\boldsymbol{p}_2^{[0]}(t_0)\\ \boldsymbol{p}_3^{[1]}(t_0)=\dfrac{6-t_0}{3}\boldsymbol{p}_2^{[0]}(t_0)+\dfrac{t_0-3}{3}\boldsymbol{p}_3^{[0]}(t_0)\end{cases}$$

其中，$\boldsymbol{p}_0^{[0]}(t_0)=\boldsymbol{c}_0$，$\boldsymbol{p}_1^{[0]}(t_0)=\boldsymbol{c}_1$，$\boldsymbol{p}_2^{[0]}(t_0)=\boldsymbol{c}_2$ 以及 $\boldsymbol{p}_3^{[0]}(t_0)=\boldsymbol{c}_3$。

□

进一步，B样条曲面是一种双参变量B样条函数组成的混合多项式曲面。B样条曲面可以看作B样条曲线在三维空间沿另外一条B样条曲线滑动形成。例如双三次B样条曲面的表达式为 $\boldsymbol{p}(s,t)=\sum_{i=0}^{3}\sum_{j=0}^{3}N_i^k(s)N_j^k(t)\boldsymbol{c}_{ij}$。与B样条曲线类似，B样条曲面也具有很多优良性质，例如局部性，即曲面形状只和最相关的几个控制顶点有关；凸包性，即B样条曲面的每一片都位于定义该片曲面的控制顶点的凸包之内；磨光性，即同一组控制顶点定义的B样条曲面，随着次数的升高越来越光滑。此外，Bézier 曲面也可以看作B样条曲面的特例。

然而，B样条曲面无法直接表示圆锥等有理曲面。为此，研究人员又提出了非均匀有理B样条曲面(NURBS)。与普通的B样条曲线/曲面相比，NURBS的特点是采用非均匀节点，同时也是一种有理表示形式。与此同时，加入了权重因子，以更好地控制曲线/曲面形状，如图 3.14 所示。具体来讲，NURBS具有如下表达式：

$$\boldsymbol{p}(s,t)=\frac{\sum_{i=0}^{n}\sum_{j=0}^{m}w_{ij}N_i^k(s)N_j^k(t)\boldsymbol{c}_{ij}}{\sum_{i=0}^{n}\sum_{j=0}^{m}w_{ij}N_i^k(s)N_j^k(t)} \tag{3.19}$$

其中，$\boldsymbol{c}_{ij}$ 是控制顶点，w_{ij} 是权重因子。

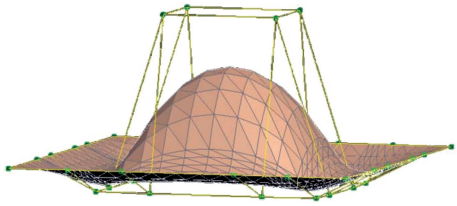


图 3.14　NURBS 定义的曲面形状

NURBS 兼具 B 样条曲线/曲面形状局部可调以及连续阶数可调的优点，又能像有理 Bézier 曲线可精确地表示圆锥曲线的特性。1991 年，国际标准化组织(ISO)在其正式发布的工业产品数据交换 STEP 标准中，把 NURBS 作为自由曲线/曲面的唯一定义，成为设计工业产品几何形状的唯一数学方法。许多国际著名的 CAD 软件也把 NURBS 作为几何造型工具的首选，例如 AutoCAD、CATIA 等软件。

3.3　细分曲面建模

B 样条曲线/曲面、NURBS 等自由曲线/曲面在 CAD 中已得到广泛的应用。然而，这类自由曲线/曲面对它们自身的控制多边形或控制网格的拓扑结构有严格要求。这里的拓扑结构通常是指控制顶点之间的边连接关系。例如，B 样条曲面、NURBS 曲面等只能定义在矩形网格拓扑结构上，如图 3.14 所示。然而，现实中几何建模对象的拓扑结构往往是复杂多样的，例如计算机动画中要表示人的头和人的手。此外，因为模型是活动的，要在曲面的连接处保持光滑也是需要解决的问题。

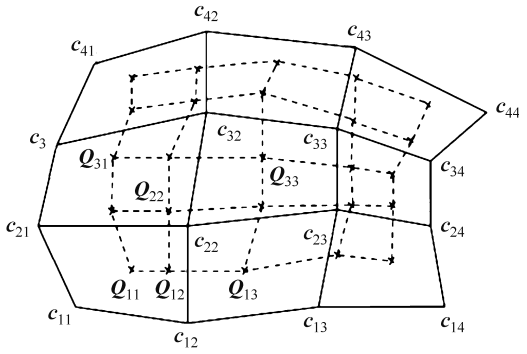
针对上述问题，科研人员进一步发明了细分曲面。所谓细分曲面，是指多面体按照指定的细分规则进行无穷细化的极限。细分曲面可以看作自由曲线曲面在任意拓扑定义域上的推广。例如，B 样条曲线的递归算法实际上就是对其控制多边形或控制网格的切割磨光过程。因此，人们自然地希望将这一算法推广到任意拓扑的多面体上去，也就是通过几何和拓扑的修改，重新添加边、顶点、面来重定义新的控制网格，以及移动顶点的空间位置来平滑该控制网格，并由此定义细分建模过程。这个过程所产生的极限曲面就是细分曲面。

细分曲面提供了更加灵活、更好光滑性的曲面生成方法，早期主要应用于计算机动画领域。最典型的是 Pixar 公司的 Tony DeRose 将细分曲面应用于电影动画 *Geri's Game* 的制作，并获得了 1998 年奥斯卡最佳动画短片奖。对于细分曲面，最核心的是如何设计细分规则，也就是顶点、边、面的几何和拓扑生成方式。接下来主要介绍 4 种典型的细分规则及其生成的极限曲面：Catmull-Clark、Doo-Sabin、Loop 和 Butterfly 细分曲面。

3.3.1 Catmull-Clark 细分曲面

Catmull-Clark 细分曲面将双三次 B 样条曲面的生成方法推广到任意拓扑的控制网格，由 Edwin Catmull 和 Jim Clark 于 1978 年首次提出。普通的双三次 B 样条曲面由 16 个控制顶点$\{\boldsymbol{c}_{ij}\}_{i=1,j=1}^{4,4}$定义。根据 B 样条曲线/曲面的递归算法，在每两个节点的中点处嵌入一个新节点，就会产生 25 个新的控制顶点。这些新的控制顶点就定义了四片新的子曲面。这样，对应于原控制网格中的每一个小四边形（如$\square\boldsymbol{c}_{11}\boldsymbol{c}_{12}\boldsymbol{c}_{22}\boldsymbol{c}_{21}$），都有一个新顶点产生，称为面点；对于每一条边，也会有一个新顶点产生，称为边点；对于每一个顶点 $\boldsymbol{c}_{ij}$，也产生一个新顶点。这些新的面点、边点和顶点组成一个新的控制网格。该过程可重复进行下去，最终控制网格收敛到双三次 B 样条曲面。受此启发，Catmull 和 Clark 提出了面向任意拓扑控制网格的细分规则，具体包括如下两方面的几何细分和拓扑细分规则。

（1）几何规则。如图 3.15 所示，每个控制网格面在中心处加一个新的面点，如 Q_{11}；每条控制网格边加一个新的边点，如 Q_{12}；每个顶点用新的位置取代旧的位置，如 Q_{22}。



$$Q_{11}=(\boldsymbol{c}_{11}+\boldsymbol{c}_{12}+\boldsymbol{c}_{22}+\boldsymbol{c}_{21})/4$$
$$Q_{12}=((Q_{11}+Q_{13})/2+(\boldsymbol{c}_{12}+\boldsymbol{c}_{22})/2)/2$$
$$Q_{13}=(\boldsymbol{c}_{12}+\boldsymbol{c}_{13}+\boldsymbol{c}_{22}+\boldsymbol{c}_{23})/4$$
$$Q_{22}=\boldsymbol{Q}/4+\boldsymbol{R}/2+c_{22}/4$$

图 3.15 Catmull-Clark 细分的几何规则和拓扑规则（图片来自[8]）

新顶点 Q_{22} 的计算公式中有 $\boldsymbol{Q}$ 和 $\boldsymbol{R}$ 两个新的变量，其中，$\boldsymbol{Q}=(Q_{11}+Q_{13}+Q_{33}+Q_{31})/4$，$\boldsymbol{R}=(1/4)((\boldsymbol{c}_{22}+\boldsymbol{c}_{12})/2+(\boldsymbol{c}_{22}+\boldsymbol{c}_{21})/2+(\boldsymbol{c}_{22}+\boldsymbol{c}_{32})/2+(\boldsymbol{c}_{22}+\boldsymbol{c}_{23})/2))$。

（2）拓扑规则。如图 3.15 中的虚线所示，新的边是由连接每个面点到邻接的边点以及连接每个顶点到邻接的边点所形成。这些新的顶点和边就组成了新的控制网格。

与 B 样条曲面采用矩形拓扑的控制网格不同，Catmull-Clark 细分曲面不受控制网格的拓扑限制，可作用到任意拓扑的控制网格上去。上述细分规则在作用一次以后，所有的面均变为四边形，而且从此以后度数不为 4 的顶点（称为奇异点）的个数会保持不变。除了奇异点以外，Catmull-Clark 曲面可以看作由一系列双三次 B 样条曲面覆盖而成，通过上述的细分规则就能够达到几乎处处连续的曲率。而在奇异点处，也能够保持切平面连续，但需要对细分规则做一些相应调整。

例题 3-5 Catmull-Clark 细分曲面。

问题：如图 3.16 所示的正立方体的 8 个顶点坐标分别是 $A(-1,1,1)$、$B(1,1,1)$、$C(1,1,-1)$、$D(-1,1,-1)$、$E(-1,-1,1)$、$F(1,-1,1)$、$G(1,-1,-1)$和 $H(-1,-1,1)$。写出采用 Catmull-Clark 细分生成新的控制网格顶点的伪代码。

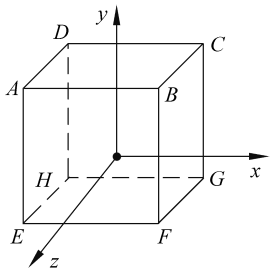


图 3.16　Catmull-Clark 细分曲面

解答： 假设 $k-1$ 次细分后顶点集合是 $V^{k-1}=\{v_1^{k-1},v_2^{k-1},\cdots\}$，其中，$V^0=\{A,B,C,D,E,F,G,H\}$，在 $k-1$ 次细分后所得的面和边的集合分别记为 $F^{k-1}=\{f_1^{k-1},f_2^{k-1},\cdots\}$ 和 $E^{k-1}=\{\boldsymbol{e}_{ij}^{k-1}\}$，那么第 k 次细分生成新的控制网格顶点的伪代码如下。

```
function CCSubdivision (V^{k-1}, F^{k-1}, E^{k-1})
    for each face f^{k-1}_{ijmn} = (v_i^{k-1}, v_j^{k-1}, v_m^{k-1}, v_n^{k-1}) in F^{k-1} do      /* 计算新的面点 */
        ṽ^k_{ijmn} ← (v_i^{k-1} + v_j^{k-1} + v_m^{k-1} + v_n^{k-1}) /4
    end for
    for each edge e^{k-1}_{ij} = (v_i^{k-1}, v_j^{k-1}) in E^{k-1} do                            /* 计算新的边点 */
        ṽ^k_{ij} ← ((v_i^{k-1} + v_j^{k-1}) /2 + (v^k_{ijmn} + v^k_{ijpq}) /2) /2                /* v^k_{ijmn} 和 v^k_{ijpq} 是面点 */
    end for
    for each vertex v_i^{k-1} in V^{k-1} do                                                      /* 计算新的顶点 */
        ṽ^k_i ← Q/4 + R/2 + v_i^{k-1} /4                                                         /* Q 和 R 是新变量 */
    end for
end function
```

□

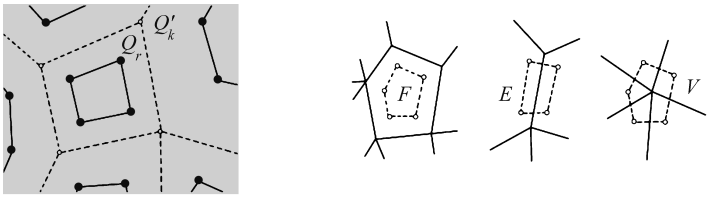
3.3.2　Doo-Sabin 细分曲面

Doo-Sabin 细分曲面将双二次 B 样条曲面的生成方法推广到任意拓扑的控制网格，是由 Daniel Doo 和 Malcolm Sabin 在 1978 年最先提出的。普通的双二次 B 样条曲面在递归计算过程中，每一个面上对应于每一个顶点，产生一个新顶点。连接这些新顶点，就会产生对应于原控制网格中的面点、边点和顶点的新面，并由此形成不断加密的控制网格。最终，这个加密过程得到的极限曲面就是双二次 B 样条曲面。受此启发，Doo-Sabin 细分采用如下两方面的几何和拓扑规则。

(1) 几何规则。每个控制网格面的 K 个顶点 $Q_1,Q_2,\cdots,Q_K$ 生成新的对应顶点 $Q_1',Q_2',\cdots,Q_K'$，其中

$$Q_k'=\sum_{j=1}^{K}\alpha_{ij}Q_j,\quad \alpha_{ij}=\begin{cases}\dfrac{K+5}{4K}, & i=j\\ \dfrac{3+2\cos(2(i-j)\pi/K)}{4K}, & i\neq j\end{cases}\tag{3.20}$$

这些新的顶点就定义了新的控制网格，如图 3.17(a)所示。



(a) 几何规则　　(b) 拓扑规则

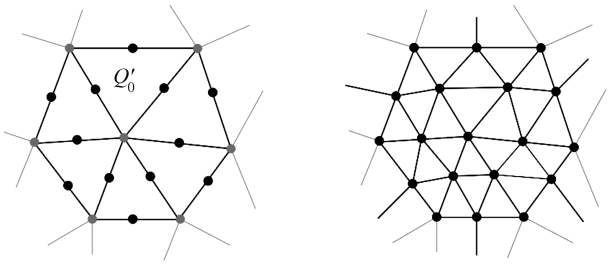
图 3.17　Doo-Sabin 细分规则

(2) 拓扑规则。如图 3.17(b)所示，每个旧控制网格面的新顶点连接形成一个新的面 F；每条旧边两侧的 4 个新顶点连接形成新的面 E；每个旧顶点周围的新顶点连接形成新的面 V。

经过一次 Doo-Sabin 细分后，每个顶点的度数均变为 4。再经过一次细分后，度数不为 4 的面的个数会保持不变。因此除了在有限个奇异点外，Doo-Sabin 细分曲面是由一系列双二次 B 样条曲面覆盖而成。此外，Doo-Sabin 曲面在奇异点处也具有一阶光滑连续性。

3.3.3　Loop 细分曲面

Loop 细分曲面将箱样条推广到三角形组成的控制网格，由 Charles Loop 在 1987 年最先提出。相比于 Catmull-Clark 细分曲面和 Doo-Sabin 细分曲面，Loop 细分曲面的细分规则较为简单。通过生成新的顶点，并依次将其连接形成新的控制网格，如图 3.18 所示。具体来讲，Loop 细分采用如下几何和拓扑规则。



(a) 几何规则　　(b) 拓扑规则

图 3.18　Loop 细分规则

(1) 几何规则。对于控制网格内部顶点 Q_0，假设其 N 个相邻顶点为 $Q_1, Q_2, \cdots, Q_N$，那么对应的新顶点 $Q'_0 = (1-N\beta)Q_0 + \beta\sum_{i=1}^{N} Q_i$，其中，$\beta = \left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{N}\right)^2\right)\Big/N$。对于控制网格边界上顶点 Q_0，假设与它相连的两个顶点是 Q_1 和 Q_2，那么对应的新顶点满足 $Q'_0 = \frac{3}{4}Q_0 + \frac{1}{8}(Q_1 + Q_2)$。假设控制网格内部一条边的两个顶点是 Q_1 和 Q_2，相对的两个

顶点是 Q_3 和 Q_4，那么新增加的顶点是 $Q'_0=\frac{3}{8}(Q_1+Q_2)+\frac{1}{8}(Q_3+Q_4)$。如果 Q_1 和 Q_2 是控制网格的边界边上的两个顶点，则对应于该边界边所新增加的顶点是 $Q'_0=\frac{1}{2}(Q_1+Q_2)$。

(2) 拓扑规则。按照三角形控制网格的顶点和边的连接关系，将新顶点连接形成新的控制网格。

Loop 细分曲面除了一些特殊点外，几乎处处具有二阶光滑的连续性。

例题 3-6　Loop 细分曲面。

问题： 如图 3.19 所示正四面体的 4 个顶点的坐标分别为 $A(0,1,0)$、$B(0,0,1)$、$C(1,0,0)$ 和 $D(0,0,0)$。写出采用 Loop 细分生成新的控制网格顶点的伪代码。

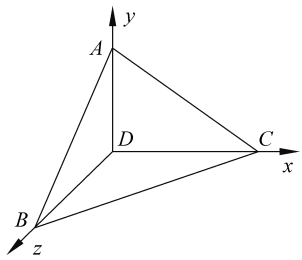


图 3.19　Loop 细分曲面

解答： 假设 $k-1$ 次细分后顶点集合是 $V^{k-1}=\{v_1^{k-1}, v_2^{k-1},\cdots\}$，其中，$V^0=\{A,B,C,D\}$。在 $k-1$ 次细分后所得的边的集合记为 $E^{k-1}=\{\boldsymbol{e}_{ij}^{k-1}\}$，那么第 k 次细分生成新的控制网格顶点的伪代码如下。

```
function LoopSubdivision (V^{k-1}, F^{k-1}, E^{k-1})
    for each vertex v_i^{k-1} in V^{k-1} do
        if v_i^{k-1} in v_i^{k-1} ∈ V_int then                    /* 计算内部顶点 */
            ṽ_i^k ← (1 - Nβ) v_i^{k-1} + β Σ_{j=1}^{N} v_ij^{k-1}    /* {v_ij^{k-1}} 是相邻 N 个顶点 */
        else                                                      /* 计算边界顶点 */
            ṽ_i^k ← 3/4 v_i^{k-1} + 1/8 (v_i1^{k-1} + v_i2^{k-1})    /* v_i1^{k-1} 和 v_i2^{k-1} 是相邻顶点 */
        end if
    end for
    for each edge e_ij^{k-1} = (v_i^{k-1}, v_j^{k-1}) in E^{k-1} do
        if e_ij^{k-1} in E_int then                               /* 计算内部边界顶点 */
            ṽ_ijpq^k ← 3/8 (v_i^{k-1} + v_j^{k-1}) + 1/8 (v_p^{k-1} + v_q^{k-1})    /* v_p^{k-1} 和 v_q^{k-1} 是相对顶点 */
        else                                                      /* 计算边界边顶点 */
            ṽ_ij^k ← 1/2 (v_i^{k-1} + v_j^{k-1})
        end if
    end for
end function
```

□

3.3.4　Butterfly 细分曲面

Butterfly 细分曲面是由 Nira Dyn 等人于 1990 年提出的，主要针对三角形组成的控制网格。具体来讲，对于每个边，使用指定的规则创造一个新顶点，然后和旧的顶点把一

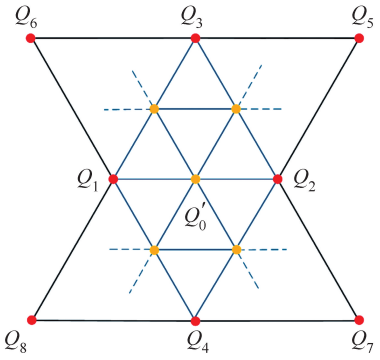


图 3.20　Butterfly 细分的几何规则和拓扑规则

个三角面片转化成四个新的三角面片，如图 3.20 所示。Butterfly 细分采用如下几何和拓扑规则。

（1）几何规则。对于三角形控制网格的每条边，利用可调控的权因子 ω 定义新的控制网格顶点 $Q_0'=\frac{1}{2}(Q_1+Q_2)+\omega(Q_3+Q_4)-\frac{\omega}{2}(Q_5+Q_6+Q_7+Q_8)$。这里的权因子 ω 是可以根据细分曲面的外形要求而设定的。

（2）拓扑规则。依次将新的顶点和旧控制网格顶点连接，形成新的控制网格。

根据上述几何和拓扑规则得到的 Butterfly 细分曲面，除了一些特殊点外，几乎处处具有一阶光滑的连续性。

上述细分曲面在细分过程中使用的几何和拓扑规则不同，由此产生不同性质的细分曲面。图 3.21 展示了同一个控制网格，采用不同细分规则后所生成的细分曲面，可以看出其结果在形状和光滑性上都有着明显的差异。

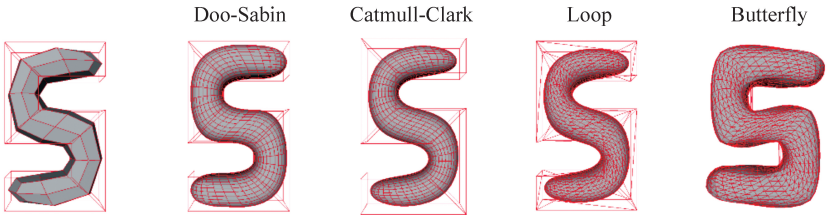


图 3.21　由相同控制网格生成的不同细分曲面

3.4　三维重建

自由曲线/曲面、细分曲面等几何建模方式是从数学模型出发来设计和构造几何形状，一定程度上要依赖设计人员关于几何形状的直觉和理解。与此相反，三维重建是直接或间接地获取真实世界中物体的形状和表观，使之能够在计算机中存储、处理和显示。因此，三维重建是一种从真实世界物体出发的几何建模方式。最具代表性的是美国 Stanford 大学的“数字米开朗基罗”计划。从 1998 年到 2000 年，该项目采用最先进的激光三维扫描仪将文艺复兴时代的著名雕塑作品全部进行几何建模，从而实现了文物的计算机数字化，起到了文物保护和文化传承的作用。

3.4.1　被动式与主动式建模

按照数据来源的不同，采用三维重建进行几何建模主要分为两种方式：被动式和主动式，如图 3.22 所示。这两种都属于光学测量的范畴。其中，被动式包括基于图像的三维重建、基于视频的三维重建等；主动式包括基于激光测距的三维重建、基于 Kinect 的三

维重建等。这两种方式各有优缺点，适用于不同的场合。

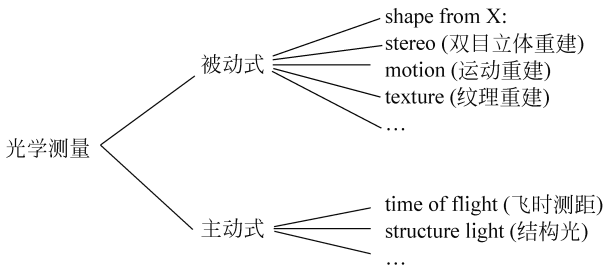


图 3.22　典型的三维重建方法

1. 被动式建模

被动式建模不需要与重建对象接触。在进行重建时，输入的数据是图像、视频等视觉信息，然后通过成像方式测量物体表面来推测三维结构。这类方法通常称为 shape from X，X 可以指代轮廓、着色、纹理、阴影等能够通过设备获取的视觉信息。

被动式方法的优点是破坏性小、安全、成本较低。在数据采集时不需要和物体接触，所以不会对物体造成破坏，因此也避免了安全隐患。在建模时通常只需要根据拍摄的图像、视频数据作为输入，因而成本较低。但缺点是对物体透明度敏感、不能处理镜面反射和内部折射，因此无法对玻璃材质的物体进行有效的几何建模。

2. 主动式建模

主动式建模是在采集数据时通过机械接触或主动观测等方式获取三维信息，例如传感器标记、结构光、激光、超声波等。按照扫描方式的区别可以分为三维扫描仪、飞时测距、三角测距、结构光等。

三维扫描仪，如坐标测量机等，具有测量精确度高的优点，可达到微米级别。但是这种方式价格昂贵，且需要专业的操作者。飞时测距，是通过发出激光脉冲，并计算这束光返回所需要的时间来测算距离。这种方式的优点是扫描速度快、便携、方便，而且测量范围大；但缺点是精度有限，只能达到毫米级别。三角测距，是通过发射一道激光到待测物上，并利用摄影机记录待测物上的激光光点，然后利用激光光点、摄影机、激光源构成的三角形来测算距离。这种方式的优点是精度较高、适合测量大尺寸物体；但缺点是扫描速度慢，需要花费较长时间来完成三维重建。结构光扫描，如 Kinect 等，则使用红外线发射器发射红外光线，然后利用红外线传感器接收反射回来的红外光线来获取深度图像。这种方式的优点是设备价格便宜、易于安装，但缺点是精度较差，而且测量范围有限(0.4～3.5m)。

3.4.2　基于图像的三维重建

基于图像的三维重建(Image-Based Reconstruction，IBR)，是指从拍摄的图像对三维物体的外形进行重建。按照输入图像数量的不同，可分为多视角重建和单幅图像重建。多视角重建需要输入多幅图像，然后恢复物体的外形。单幅图像重建仅需要一幅图像作为输入进行重建。一般情况下，图像像素记录了来自不同方向的入射光信息。因此，增加输入图像数量能够提升几何模型重建的精度。

如图 3.23 所示，多视角重建的算法流程一般包括四个步骤：摄像机标定(Calibration)、三角测量(Triangulation)、从运动恢复结构(稀疏形状估计)(Structure-From-Motion，SFM)以及立体匹配(稠密形状估计)(Stereo Match)。

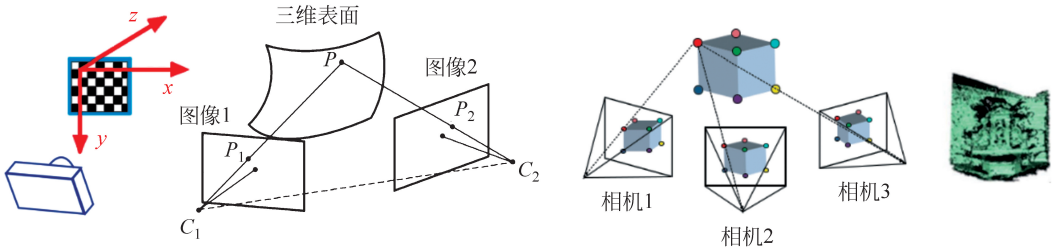


图 3.23　多视角三维重建算法流程

1. 摄像机标定

该步骤从一些已知坐标的三维空间点集，反求摄像机相关一些参数。这些参数可以分为内参(Intrinsic Parameter)和外参(Extrinsic Parameter)两种类型。内参是描述摄像机镜头、传感器特性的参数，例如镜头焦距、传感器畸变、成像中心位置等。外参是描述摄像机在世界坐标系下位置和方向的参数，反映了相机运动时的位姿变化。由标定出来的摄像机内/外参就可以得到和物体三维信息有关的相机运动。因此，摄像机标定是基于图像三维重建的关键步骤，决定了后续重建结果的准确性。

摄像机标定是计算机视觉领域的一个基础问题，往往需要借助特定的相机运动或拍摄图像才能获得可靠、稳定的数值。一般情况下，摄像机的成像模型通常简化为小孔成像模型，那么从三维空间到二维成像平面的成像可以看作投影变换。因此，摄像机的成像模型表示为：

$$\boldsymbol{x}_{3\times1}=\boldsymbol{P}_{3\times4}\cdot\boldsymbol{X}_{4\times1}\Rightarrow\begin{bmatrix}x\\y\\1\end{bmatrix}=\begin{bmatrix}p_{11}&p_{12}&p_{13}&p_{14}\\p_{21}&p_{22}&p_{23}&p_{24}\\p_{31}&p_{32}&p_{33}&p_{34}\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}\tag{3.21}$$

其中，$\boldsymbol{x}$ 是二维成像平面上的像素坐标，$\boldsymbol{X}$ 是三维空间中的点坐标，$\boldsymbol{P}_{3\times4}$是描述小孔成像过程的投影矩阵。为了方便统一的表示，这里采用齐次坐标形式。摄像机标定的主要任务就变成计算投影矩阵 $\boldsymbol{P}_{3\times4}$，进一步可以分解为内参矩阵 $\boldsymbol{K}_{3\times3}$ 和外参矩阵 $\boldsymbol{M}_{3\times4}$，即 $\boldsymbol{P}_{3\times4}=\boldsymbol{K}_{3\times3}\cdot\boldsymbol{M}_{3\times4}$。内参矩阵和外参矩阵的矩阵元素就对应了待标定的内参和外参。

具体来讲，内参矩阵 $\boldsymbol{K}_{3\times3}$描述了在摄像机坐标系下，三维空间中的点到二维图像像素的投影变换。假设摄像机位于世界坐标系的原点，朝向与 z 坐标轴重合，如图 3.16 所示，而二维图像坐标系的原点在图像中心，那么内参矩阵 $\boldsymbol{K}_{3\times3}$可以表示为一个对角矩阵和上三角矩阵的乘积：

$$\boldsymbol{K}_{3\times3}=\begin{bmatrix}m_x&&\\&m_y&\\&&1\end{bmatrix}\begin{bmatrix}f&&p_x\\&f&p_y\\&&1\end{bmatrix}\tag{3.22}$$