中国科学技术大学

读书报告

多边形网格处理

作者姓名：周 航
学科专业：网络空间安全
导师姓名：俞能海 教授
张卫明 教授
完成时间：二〇二〇年四月

目 录

第一章　背景与相关理论

本章节中，我们主要介绍计算机图形学的研究方向，以及与本文所要介绍的多边形网格处理的关系。

1.1　什么是计算机图形学？

计算机图形学（Computer Graphics，简称 CG）的内容较为丰富，与很多学科都有交叉难以严格定义。

在“Wikipedia”和“百度百科”上，对“计算机图形学”的解释为：计算机图形学是一种使用数学算法将二维或三维图形转化为计算机显示器的栅格形式的科学。简单地说，计算机图形学的主要研究内容就是研究如何在计算机中表示图形、以及利用计算机进行图形的计算、处理和显示的相关原理与算法。虽然通常认为 CG 是指三维图形的处理，事实上也包括了二维图形及图像的处理。

狭义地理解，计算机图形学是数字图象处理或计算机视觉的逆过程：计算机图形学是用计算机来画图像的学科，数字图象处理是把外界获得的图象用计算机进行处理的学科，计算机视觉是根据获取的图像来理解和识别其中的物体的三维信息及其他信息。

实际上，计算机图形学、数字图像处理和计算机视觉在很多地方的区别不是很明晰，很多概念是相通的，随着研究的深入，这些学科方向不断的交叉融入，形成一个更大的学科方向，可称之为“可视计算”（Visual Computing）。

1.2　主要内容

早期计算机图形学要解决的是如何在计算机中表示三维几何图形，以及如何利用计算机进行图形的生成、处理和显示的相关原理与算法，产生令人赏心悦目的真实感图像。这是狭义的计算机图形学的范畴。随着近 40 年的发展，计算机图形学的内容已经远远不止这些了。广义的计算机图形学的研究内容非常广泛，如图形硬件、图形标准、图形交互技术、光栅图形生成算法、曲线曲面造型、实体造型、真实感图形计算与显示算法，以及科学计算可视化、计算机动画、自然景物仿真、虚拟现实等。

计算机图形学主要包含四大部分的内容：建模（Modeling）、渲染（Rendering）、动画（Animation）和人机交互（Human–Computer Interaction, HCI）。

1.2.1　建模

要在计算机中表示一个三维物体，首先要有它的几何模型表达。因此，三维模型的建模是计算机图形学的基础，是其他内容的前提。表达一个几何物体可以是用数学上的样条函数或隐式函数来表达；也可以是用光滑曲面上的采样点及其连接关系所表达的三角网格来表达（即连续曲面的分片线性逼近）。

1.2.2　渲染

有了三维模型或场景，怎么把这些三维几何模型画出来，产生令人赏心悦目的真实感图像？这就是传统的计算机图形学的核心任务，在计算机辅助设计，影视动漫以及各类可视化应用中都对图形渲染结果的高真实感提出了很高的要求。

1.2.3　动画

动画是采用连续播放静止图像的方法产生物体运动的效果。计算机动画借助于编程或动画制作软件生成一系列的景物画面，是计算机图形学的研究热点之一。另外，高度物理真实感的动态模拟也是动画领域的主要问题。这些技术是各类动态仿真应用的核心技术，可以极大地提高虚拟现实系统的沉浸感。计算机动画的应用领域广泛，比如动画片制作，广告、电影特技，训练模拟，物理仿真，游戏等。

1.2.4　人机交互

人机交互是指人与计算机之间以一定的交互方式或交互界面，来完成确定任务的人与计算机之间的信息交换过程。简单来讲，就是人如何通过一定的交互方式告诉计算机来完成他所希望完成的任务。比如 2010 年微软推出的 Kinect 就是一种无需任何操纵杆的基于体感的人机界面，用户本身就是控制器。Kinect 在微软的 Xbox 游戏上取得了极大的成功，之后在其他方面也得到了很多的应用。

1.3　多边形网格处理

多边形网格处理属于计算机图形学的“建模”分支，主要研究的是多边形网格的微分几何、平滑处理、网格参数化、网格重划分、网格简化、网格修复和网格变形等。

第二章　曲面的表示

本章主要讨论几何物体的不同数学表示方式。尽管表现方式多样，但处理的基本上是 3D 物体的 2D 曲面。对于不同任务，需要选择合适 3D 表征方式来实现高效的几何操作。

从抽象的角度来说，曲面的表示主要分为两类：**参数方程表示**（Parametric Representations）和**隐式方程表示**（Implicit Representations）。参数方程表示是一个从 2 维参数 $\Omega \subset \mathbb{R}^2$ 到表示 3D 物体平面上点的空间坐标 $\mathcal{S} = \mathbf{f}(\Omega) \subset \mathbb{R}^3$ 的 3 维参数的一个映射 $\mathbf{f} : \Omega \to \mathcal{S}$；隐式方程表示是一个等号左边为表示 3D 物体平面上点的空间坐标的 3 维参数构成的标量表达式 $F : \mathbb{R}^3 \to \mathbb{R}$，等号右边为 0 的方程，例如 $\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 | F(\mathbf{x}) = 0\}$。

参数方程和隐式方程各有优劣，因此在实际中，一般根据需求采用不同的表示方法。这些需求一般分为如下 3 点：

- 评价：在对曲面进行采样时，需要对其附加上除了空间信息之外的信息。例如渲染时，除了几何体的空间坐标外，还需要其法向量信息。
- 查询：一个典型的空间查询是判断空间中的某个点是否在几何体的内部，另外还比如空间中某个点到某个曲面的距离。
- 修改：一个曲面可以在几何上被修改（将一个平面卷起来），还可以在拓扑关系上被修改（把几张纸合并成一张更大的纸或者挖掉纸的一部分）。

2.1　曲面的定义

曲面（Surface）的定义是“an orientable continuous 2D manifold embedded in $\mathbb{R}^3$”，可以理解为处于 3 维空间中的，方向连续的，不存在通过无限细小特征连接的的部分。

2.2　参数方程表示

一个三角网格 $\mathcal{M}$ 包含几何和拓扑元素。拓扑元素包含多个顶点：

$$\mathcal{V} = \{v_1, .., v_V\}, \tag{2.1}$$

以及连接这些顶点的三角面片的集合：

$$\mathcal{F} = \{f_1, .., f_F\}, \quad f_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}. \tag{2.2}$$

有时通过图的边来表示三角网格的连接关系更为高效：

$$\mathcal{E} = \{e_1, .., e_E\}, \quad e_i \in \mathcal{V} \times \mathcal{V}. \tag{2.3}$$

三角网格的几何嵌入是将 3D 坐标 $\mathbf{p}_i$ 与顶点 $v_i \in \mathcal{V}$ 关联起来：

$$\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_V\}, \quad \mathbf{p}_i := \mathbf{p}(v_i) = \begin{pmatrix} x(v_i) \\ y(v_i) \\ z(v_i) \end{pmatrix} \in \mathbb{R}^3. \tag{2.4}$$

欧拉公式（Euler Formula）证明了在一个闭合连接网格内点，边和面的数量存在下面的关系：

$$V - E + F = 2(1 - g), \tag{2.5}$$

其中 g 表示的是曲面的亏格，可理解为几何体上洞眼的数目。

对于一个三角形网格，基于上述公式还能得到以下关系：

- 三角形的数量大约是顶点数量的 2 倍：$F \approx 2V$。
- 边的数量大约是顶点数量的 3 倍：$E \approx 3V$。
- 顶点邻边的数量平均是 6。

第三章　网格数据结构

选择网格数据结构时一般考虑以下两个因素：拓扑需求和算法需求。

- 拓扑需求：需要表示什么样的网格？是二维流形，还是其它更复杂的网格？是单纯的三角形网格，或是其它任意多边形网格？是需要给当前的网格附加上其它的网格？

- 算法需求：使用什么样的算法？是简单的渲染还是需要频繁的访问相邻的点边面？是静态网格还是动态网格？对于网格是否需要附加一些其它的属性？对内存的消耗是否特别在意？

评估一个数据结构的好坏有四个标准：建立数据结构所需要预处理的时间、查询操作的响应时间、执行某项具体操作的时间、内存消耗和冗余。

3.1　面数据结构

面数据结构的优点是简单，由网格所有面的集合构成，而对于每一个面则使用组成面多边形的点来表示。如图 3.1 左图所示，以三角形网格为例，假设使用 32 位的单精度浮点数来表示坐标，那么使用顶点的数据来表示一个三角形则需要 36 个字节（3 顶点 ×3 维 ×4 字节 =36 字节）。由于顶点邻边的平均数量为 6，即在这种数据结构中一个顶点平均会被存储 6 次，所以可以估算使用这种数据结构平均每一个顶点要使用 72 个字节（3 维 ×4 字节 ×6 次 =72 字节）。

由于一个顶点的数据被多次存储，因此一种可以改进这种空间冗余的方法是使用数组来存储所有的顶点数据，如图 3.1 右图所示。对于每一个三角形面，只需存储组成其的三个顶点的索引号即可。这样存储一个三角形或一个顶点只需 12 个字节（3 个 4 字节的浮点数或 3 个整数）。这样平均一个顶点只需 36 个字节（3 维 ×4 字节 +6 次 ×4 字节 =36 字节，一份顶点数据，6 个索引号）。但是这种数据结构缺显式连通性信息，因为在大多数算法中都会用到以下操作：

- 迭代访问所有的点、边和面；

- 有方向地遍历一个面周围的边，这需要通过一条边找到它的下一条或者上一条边，可能还需使用到对应顶点的数据；

- 通过一条边找到与它相邻的面或者它的两个端点；

- 通过一个顶点迭代访问它周围的边和面。

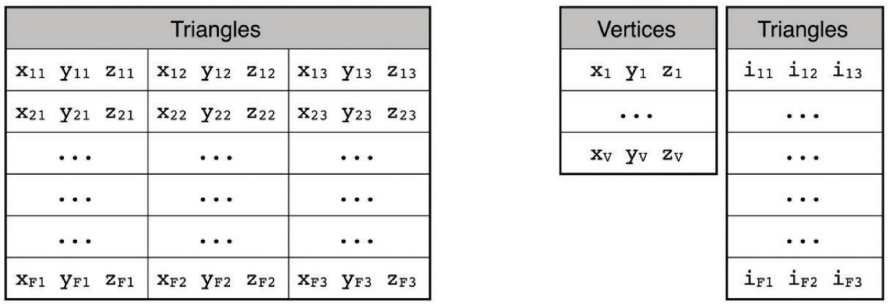


图 3.1　面数据结构（左图），带索引的面数据结构（右图）。

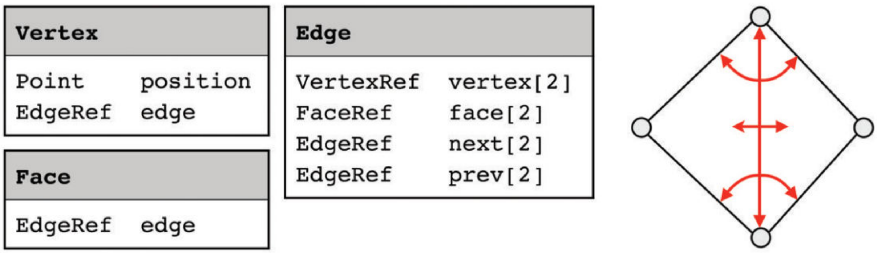


图 3.2　边数据结构存储的拓扑连接信息。

要通过以上的数据结构完成这些操作，可以在当前的数据结构中添加一部分信息。添加了附加信息后，存储一个面需要 24 个字节，存储一个顶点需要 16 个字节，平均一个顶点需要 64 个字节。

3.2　边数据结构

边数据结构改进了面数据结构中没有显式存储边的问题，并附加存储了许多其它信息到边上（两个端点、相邻的面、边），如图 3.2所示。这种数据结构存储一个面需要 4 个字节，存储一个顶点需要 16 个字节，存储一条边需要 32 个字节，平均一个顶点需要 120 个字节。

3.3　半边数据结构

在半边数据结构中，一个重要的概念是半边（Halfedge）。半边代表一条边的一半，即把一条边保持长度不变，形式上分为两条半边。这两条半边组合在一起称为一条边，即一条边等于一对半边。半边是有方向的，并且一条边的一对半边有相反的方向。对于每一条半边，我们存储以下信息：

- 半边指向的顶点；
- 半边相邻的面（当半边为网格边界的时候没有相邻的面）；

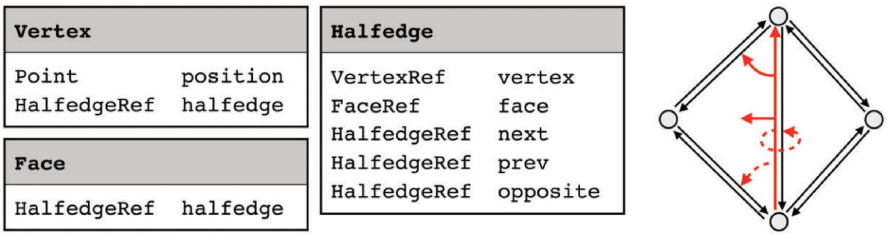


图 3.3　半边数据结构存储的拓扑连接信息。

- 一个面中的上或下一条半边；
- 与它配对的另一条半边。

如图 3.3所示，存储一个面需要 4 个字节，存储一个顶点需要 16 个字节，存储一条半边需要 20 个字节，平均一个顶点需要 144 个字节。可以优化使用数组的索引号而不是指针来存储具体的信息，这样在需要附加额外信息的时候会更加方便，同时由于数组在内存中是连续排布的，在进行内存管理的时候也会更加简单高效。

3.4　有向边数据结构

有向边数据结构是半边数据结构的一种特殊形式，只能用于三角形网格。这种数据结构将连通性信息隐藏在了数组索引号的关系中。假设 f 是面的索引号，那么这个面 3 条半边的索引号分别为：

$$\text{halfedge}\,(f,i) = 3f + i, \quad i = 0,1,2 \tag{3.1}$$

假设 h 为半边的索引，那么其相邻面的索引以及该半边在该面中的索引号为：

$$\text{face}\,(h) = h/3, \quad \text{face\_index}(h) = h \bmod 3 \tag{3.2}$$

假设要获得该半边指向的下一个半边的索引号，那么只需要计算 $(h+1) \bmod 3$ 即可。每一个顶点存储其位置信息和从该点射出的半边的索引号；每一个半边存储与其配对的另一条反向的半边以及该半边对应的顶点（射出该半边的）的索引号。这样存储一个顶点需要 16 个字节，存储一条半边需要 8 个字节，平均一个顶点需要 64 个字节。该数据结构在网格的边界部分需要特殊处理索引号。另外，该数据结构只能单纯地表示某种类型的网格，例如三角形网格或者四边形网格，而不能表示混合了三角形和四边形的网格。

3.5　总结

为了权衡变通性、存储使用和计算复杂度，作者建议使用开源工具箱：CGAL[*]，OpenMesh[†]和 MeshLab[‡]。

[*]CGAL: http://www.cgal.org

[†]OpenMesh: http://www.openmesh.org

[‡]MeshLab: http://www.meshlab.org

第四章　微分几何

4.1　曲面

4.1.1　度量性质

假设一个三维曲面 $\mathcal{S} \subset \mathbb{R}^3$ 的参数方程如下:

$$\mathbf{x}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}, \quad (u,v) \in \Omega \subset \mathbb{R}^2 \tag{4.1}$$

其中 x，y，z 是关于参数 u，v 的可微函数，Ω 是参数 u，v 的定义域。

曲面的度量是由它的一阶导数决定。$\mathbf{x}$ 关于参数 u，v 的偏导数如下：

$$\begin{aligned} \mathbf{x}_u(u_0,v_0) &:= \frac{\partial \mathbf{x}}{\partial u}(u_0,v_0), \\ \mathbf{x}_v(u_0,v_0) &:= \frac{\partial \mathbf{x}}{\partial v}(u_0,v_0). \end{aligned} \tag{4.2}$$

定义一个在曲线方程参数空间 u，v 下的方向向量：

$$\overline{\mathbf{w}} = (u_w, v_w)^T. \tag{4.3}$$

如图 4.1所示，向量 $\mathbf{w}$ 定义在三维空间下，而方向向量在二维参数空间上，现在要将其从参数空间变换为曲面上的切向量，只需要应用到雅可比矩阵即可完成这个变换：

$$\mathbf{w} = \mathbf{J}\overline{\mathbf{w}}, \tag{4.4}$$

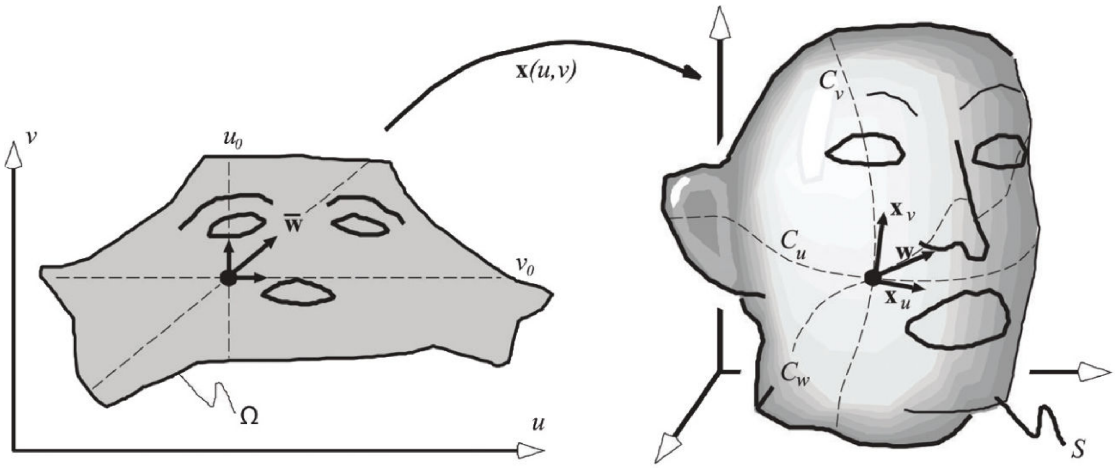


图 4.1　二维参数空间上的方向向量 $\overline{\mathbf{w}}$ 变换为曲面上的切向量 $\mathbf{w}$。

其中雅可比矩阵的值为：

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \\ \frac{\partial z}{\partial u} & \frac{\partial z}{\partial v} \end{bmatrix} = \left[\mathbf{x}_u, \mathbf{x}_v\right]. \tag{4.5}$$

通过雅可比矩阵可以知道一些量，诸如角度、距离和面积等在这两个空间之间的映射关系。

$$\mathbf{w}_1^T \mathbf{w}_2 = \left(\mathbf{J}\overline{\mathbf{w}}_1\right)^T \left(\mathbf{J}\overline{\mathbf{w}}_2\right) = \overline{\mathbf{w}}_1^T \left(\mathbf{J}^T\mathbf{J}\right) \overline{\mathbf{w}}_2, \tag{4.6}$$

在该等式中，**J** 乘以 **J** 的转置称为**曲面的第一基本型**：

$$\mathbf{I} = \mathbf{J}^T\mathbf{J} = \begin{bmatrix} E & F \\ F & G \end{bmatrix} := \begin{bmatrix} \mathbf{x}_u^T\mathbf{x}_u & \mathbf{x}_u^T\mathbf{x}_v \\ \mathbf{x}_u^T\mathbf{x}_v & \mathbf{x}_v^T\mathbf{x}_v \end{bmatrix}. \tag{4.7}$$

借由 **I**，正切向量 **w** 的模长为 $\|\mathbf{w}\|^2 = \overline{\mathbf{w}}^T\mathbf{I}\overline{\mathbf{w}}$。同样地，曲面的面积为：

$$A = \iint_U \sqrt{\det(\mathbf{I})}\mathrm{d}u\mathrm{d}v = \iint_U \sqrt{EG - F^2}\mathrm{d}u\mathrm{d}v. \tag{4.8}$$

4.1.2　曲面的曲率

如图 4.3所示，对于曲面上任意一点来说，存在着无数个切向量。对于曲面上的一点 $\mathbf{p} \in \mathcal{S}$，以及一条切向量 $\mathbf{t} = u_t\mathbf{x}_u + v_t\mathbf{x}_v$，这时定义曲率为切向量 **t** 和曲面在这一点的法向量所成平面与曲面相交形成的直线在点 **p** 处的曲率：

$$\kappa_n(\bar{\mathbf{t}}) = \frac{\bar{\mathbf{t}}^T\mathbf{II}\bar{\mathbf{t}}}{\bar{\mathbf{t}}^T\mathbf{I}\bar{\mathbf{t}}} = \frac{eu_t^2 + 2fu_tv_t + gv_t^2}{Eu_t^2 + 2Fu_tv_t + Gv_t^2}, \tag{4.9}$$

其中 **II** 为**第二基本型**：

$$\mathbf{II} = \begin{bmatrix} e & f \\ f & g \end{bmatrix} := \begin{bmatrix} \mathbf{x}_{uu}^T\mathbf{n} & \mathbf{x}_{uv}^T\mathbf{n} \\ \mathbf{x}_{uv}^T\mathbf{n} & \mathbf{x}_{vv}^T\mathbf{n} \end{bmatrix}. \tag{4.10}$$

上面关于曲面曲率的函数在切线方向变化的时候会有两个极值（极大值和极小值），一般称它们为主曲率。如果两极值不相等，就把取这两个极值时的两个切向量称为主方向；如果两极值相等，则曲面上这一点称为脐点，此时曲面上这一点的所有切向量都可以称为主方向，并且曲面这一点各方向的曲率相等。特殊情况下，当且仅当曲面为球面或平面时，其上所有的点都是脐点。

欧拉定理

对于曲面的两个主曲率和其在同一点任意方向的曲率，有如下的关系：

$$\kappa_n(\bar{\mathbf{t}}) = \kappa_1\cos^2\psi + \kappa_2\sin^2\psi, \tag{4.11}$$

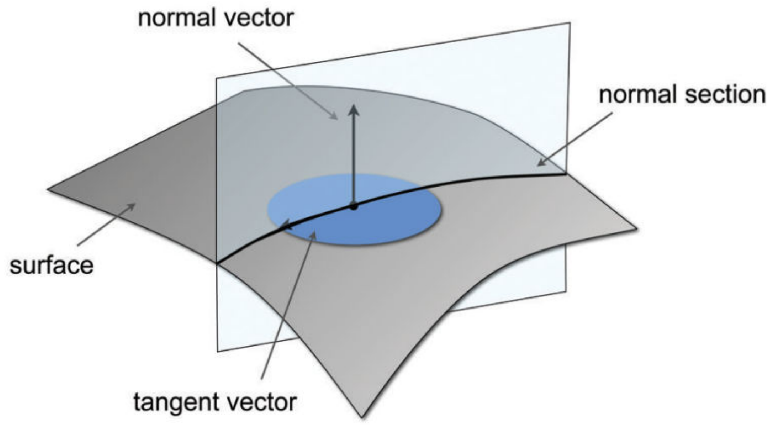


图 4.2　切向量和曲面在某点的法向量所成平面与曲面相交的示意图。

其中 ψ 为主方向 $\mathbf{t}_1$ 和指定方向 $\mathbf{t}$ 的夹角。可以看出，曲面的曲率仅仅由其两个主曲率决定，这一点任意方向的法曲率都是这两个主曲率的凸组合。同时，主方向始终相互正交。

曲面的某个区域内的性质同样可以用曲率张量 $\mathbf{C}$ 来表示：

$$\mathbf{C} = \mathbf{PDP}^{-1}, \tag{4.12}$$

其中 $\mathbf{D}$ 是对角线元素为 κ_1，κ_2，0 的三阶方阵：$\mathbf{D} = \mathrm{diag}\,(\kappa_1, \kappa_2, 0)$，$\mathbf{P}$ 也为三阶方阵：$\mathbf{P} = [\mathbf{t}_1, \mathbf{t}_2, \mathbf{n}]$。

另外，还有两种广泛使用的描述曲率的方式：平均曲率和高斯曲率。

平均曲率：

$$H = \frac{\kappa_1 + \kappa_2}{2}. \tag{4.13}$$

高斯曲率：

$$K = \kappa_1 \kappa_2. \tag{4.14}$$

高斯曲率可以将曲面上的点分为 3 类：

- 椭圆点 $K > 0$：椭圆点在其附近的区域上通常是凸出的。
- 双曲线点 $K < 0$：双曲点在其附近的区域上通常是马鞍形。
- 抛物线点 $K = 0$：抛物线点通常在椭圆曲线和双曲线区域的分界线处。

高斯曲率和平均曲率通常用在曲面的可视化分析上。

内蕴几何学（Intrinsic geometry）

在微分几何中，那些只依赖于第一基本型的属性被称为是内蕴的。直观上来说它们可以仅仅通过曲面二维特征来导出。例如曲面上曲线的长度，角度等都是内蕴的。对于高斯曲率和平均曲率，前者在等距变换下是不变的，所以它

是内蕴的，即高斯曲率是可以由第一基本型直接决定的；而后者则不是，它依赖于曲面。内蕴通常用来表示参数的独立性。

拉普拉斯算子

一般称某函数梯度的散度为拉普拉斯算子（Laplace Operator）。对于二元函数 $f(u,v)$，其在欧式空间上的二阶差分算子（拉普拉斯算子）可以写为：

$$\Delta f = \operatorname{div} \nabla f = \operatorname{div} \begin{pmatrix} f_u \\ f_v \end{pmatrix} = f_{uu} + f_{vv}. \tag{4.15}$$

拉普拉斯算子还可以推广到二阶流形曲面 $\mathcal{S}$ 上，其推广形式称为**拉普拉斯——贝尔特拉米算子**（Laplace-Beltrami Operator），定义为：

$$\Delta_{\mathcal{S}} f = \operatorname{div}_{\mathcal{S}} \nabla_{\mathcal{S}} f. \tag{4.16}$$

对于曲面上某一个具体的点 x，其拉普拉斯 -贝尔特拉米算子和其平均曲率存在下面的关系：

$$\Delta_{\mathcal{S}} \mathbf{x} = -2H\mathbf{n}. \tag{4.17}$$

虽然这个式子说明平均曲率（非内蕴的）和拉普拉斯——贝尔特拉米算子之间存在某种关系，但是拉普拉斯 -贝尔特拉米算子本身仅取决于第一基本型，是内蕴的。

4.2　离散微分算子

由于 3D 网格并不连续，而以上讨论建立在曲面是光滑的基础之上。要将上述算子运用到 3D 网格上，需要将网格看作一个很粗糙的曲面，然后通过网格数据去计算这个近似曲面的微分属性。

4.2.1　局部平均区域

直观的想法就是计算网格某个点以及与其相邻点的微分属性的平均值。当网格某个点以及与其相邻点组成这个区域的面积较大时，通过计算平均值得到的微分属性较为稳定；而面积较小时，精细的变化则会被更好地保留。

定义面积的方法主要有以下三种方式，其主要区别是在顶点周围的三角形中取点的方式不同。如图 4.3所示，当三角形为钝角三角形时则取中心点对边的中点，否则取三角形的外心。

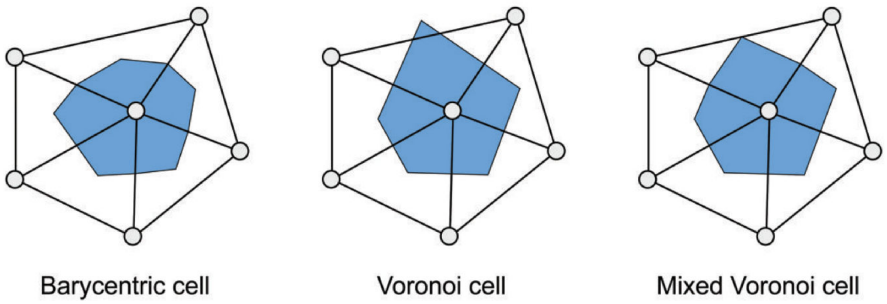


图 4.3　重心（左图），外心（中图），混合点（右图）。

4.2.2　法向量

在 3D 网格中，要计算某个三角面的法向量是比较容易的，只需要取两条边向量坐叉乘即可：

$$\mathbf{n}(T) = \frac{(\mathbf{x}_j - \mathbf{x}_i) \times (\mathbf{x}_k - \mathbf{x}_i)}{\|(\mathbf{x}_j - \mathbf{x}_i) \times (\mathbf{x}_k - \mathbf{x}_i)\|}. \tag{4.18}$$

如果要计算某个顶点的法向量，同样考虑对顶点周围相邻的三角形的法向量做加权平均：

$$\mathbf{n}(v) = \frac{\sum_{T \in \mathcal{N}_1(v)} \alpha_T \mathbf{n}(T)}{\left\| \sum_{T \in \mathcal{N}_1(v)} \alpha_T \mathbf{n}(T) \right\|}, \tag{4.19}$$

其中，权值 α_T 的取法，一般常用的有下面几种：

- α_T 取常数 1。这样计算时忽略了邻边的长度、三角形的面积和角度。对于不规则的网格，计算的结果一般都是违反直觉的；
- α_T 取三角形的面积。好处是便于计算（只需要进行叉乘运算，且无需对向量进行单位化），不过此方法得到的结果有时也会出现违反直觉的情况；
- α_T 取邻边的夹角。由于计算过程涉及到了三角函数，效率较低，但是效果比前两者好。

4.2.3　梯度

同样是基于加权平均的方法，求解网格中某个三角形上某一点的坐标可以由三个顶点的梯度根据重心坐标的三个权值做加权平均。

对于分段线性函数 f 来说，其在三角形顶点上有对应的值。三角形上分段线性函数的梯度为

$$\nabla f(\mathbf{u}) = (f_j - f_i) \frac{(\mathbf{x}_i - \mathbf{x}_k)^\perp}{2A_T} + (f_k - f_i) \frac{(\mathbf{x}_j - \mathbf{x}_i)^\perp}{2A_T}. \tag{4.20}$$

4.2.4　离散形式的拉普拉斯算子

均匀形式的拉普拉斯算子

$$\Delta f\left(v_i\right)=\frac{1}{\left|\mathcal{N}_1\left(v_i\right)\right|}\sum_{v_j\in\mathcal{N}_1(v_i)}\left(f_j-f_i\right). \tag{4.21}$$

表示以中心点 i 为起点，相邻顶点平均值为终点的向量。由于平面的平均曲率 H 为 0，此时算子的结果是 0。不过上式结果并不一定是非 0 的，所以这种方法不太适合用在非等距网格上。这种方法只考虑了网格的连接性，所以使用范围有限。

余切形式的拉普拉斯算子

该形式更为精准，直接计算顶点 v_i 周围的平均区域，然后对其梯度的散度进行曲面积分，然后使用散度定理（高斯公式）进行展开计算，最后可以得到：

$$\Delta f\left(v_i\right):=\frac{1}{2A_i}\sum_{v_j\in\mathcal{N}_1(v_i)}\left(\cot\alpha_{i,j}+\cot\beta_{i,j}\right)\left(f_j-f_i\right). \tag{4.22}$$

离散散度

由于拉普拉斯算子定义为是梯度的散度，因此对于每一个三角形 T 给定一个向量 $\mathbf{w}$（即给定分段线性函数 f 下的梯度向量），其散度为

$$\operatorname{div}\mathbf{w}\left(v_i\right)=\frac{1}{A_i}\sum_{T\in\mathcal{N}_1(v_i)}\nabla B_i\Big|_T\cdot\mathbf{w}_T A_T. \tag{4.23}$$

4.2.5　离散曲率

根据上式可得离散形式下的平均曲率：

$$H\left(v_i\right)=\frac{1}{2}\left\|\Delta\mathbf{x}_i\right\|. \tag{4.24}$$

离散形式下高斯曲率的表示方式：

$$K\left(v_i\right)=\frac{1}{A_i}\left(2\pi-\sum_{v_j\in\mathcal{N}_1(v_i)}\theta_j\right). \tag{4.25}$$

根据高斯曲率、平均曲率和两个主曲率的关系，可以得到主曲率的计算方法：

$$\kappa_{1,2}\left(v_i\right)=H\left(v_i\right)\pm\sqrt{H\left(v_i\right)^2-K\left(v_i\right)}. \tag{4.26}$$

4.2.6　离散形式的曲率张量

$$\mathbf{C}(v) = \frac{1}{A(v)} \sum_{\mathbf{e} \in A(v)} \beta(\mathbf{e}) \|\mathbf{e} \cap A(v)\| \overline{\mathbf{e}}\overline{\mathbf{e}}^T, \tag{4.27}$$

其中 $\beta(\mathbf{e})$ 表示和边 **e** 相邻三角形所在平面的有方向的二面角，$\mathbf{e} \cap A(v)$ 表示边 **e** 在区域 A 中的长度，$\overline{\mathbf{e}}$ 指边 **e** 的单位向量。

第五章　曲面的平滑和滤波

一般有两种曲面平滑的方式：

- 去噪（Denoising）：一般是去掉凸出曲面的部分（高频部分），而保留和曲面相当的部分（低频部分），即需要一个在离散三角形网格曲面上的低通滤波器。
- 抹平（Fairing）：在抹平的过程中，所做的不仅仅只是去除高频的部分。抹平的过程相当于是对曲面做了一个变换，使其从各个角度（曲率、高阶导数）上看尽可能的光滑。

5.1　傅里叶变换和流形谐波

5.1.1　一维傅里叶变换

傅里叶变换（Fourier Transform）表示一种映射 f，它将函数从空间域 $f(x)$ 变换到频域 $F(\omega)$:

$$\begin{aligned} F(\omega) &= \int_{-\infty}^{\infty} f(x)\mathrm{e}^{-2\pi\mathrm{i}\omega x}\mathrm{d}x, \\ f(x) &= \int_{-\infty}^{\infty} F(\omega)\mathrm{e}^{2\pi\mathrm{i}\omega x}\mathrm{d}\omega. \end{aligned} \tag{5.1}$$

上式中的指数部分通过欧拉公式可以展开成如下的复数形式：

$$\mathrm{e}^{2\pi\mathrm{i}\omega x} = \cos(2\pi\omega x) - \mathrm{i}\sin(2\pi\omega.x) \tag{5.2}$$

这是一个包含有正弦和余弦函数的，以 ω 为自变量的（可以看作是频率）函数，可以将这个函数看做向量空间上的一组正交基，即频域。

可以将 $f(x)$（可积的复数函数）看作向量空间中的一个元素，然后对它做下面的内积运算：

$$\langle f, g\rangle = \int_{-\infty}^{\infty} f(x)\overline{g(x)}\mathrm{d}x \tag{5.3}$$

.

那么，傅里叶变换在这里表示了一种基的变换。通过将向量 f 投影到不同频率的基向量上，然后对其进行累加操作，这样就完成了从空间域到频域的转换：

$$f(x) = \sum_{\omega=-\infty}^{\infty} \langle f, e_\omega\rangle\, e_\omega \tag{5.4}$$

如果我们要除去频率比较高的部分，保留

$$|\omega| < \omega_{\max}, \tag{5.5}$$

只需要将上面的累加式改写为如下的形式：

$$\tilde{f}(x)=\int_{-\omega_{\max}}^{\omega_{\max}}\langle f,e_{\omega}\rangle\, e_{\omega}\mathrm{d}\omega. \tag{5.6}$$

5.1.2　流形谐波

对于离散的三角形网格，需要将连续的函数 $f(x)$ 用如下逐顶点的矩阵形式来表示：

$$\left(f\left(v_1\right),\ldots,f\left(v_n\right)\right). \tag{5.7}$$

同样地，如果要将拉普拉斯——贝尔特拉米算子应用到函数上，同样也需要将 $f\left(v_i\right)$ 变成逐顶点的矩阵形式，这样算子就变成了相应的拉普拉斯一贝尔特拉米矩阵 $\mathbf{L}$ 了：

$$\begin{pmatrix}\Delta f\left(v_1\right)\\ \vdots\\ \Delta f\left(v_n\right)\end{pmatrix}=\mathbf{L}\begin{pmatrix}f\left(v_1\right)\\ \vdots\\ f\left(v_n\right)\end{pmatrix}. \tag{5.8}$$

根据上述分析可知，上式中的每一行是按以下方式运算得到：

$$\Delta f\left(v_i\right)=\sum_{v_j\in\mathcal{N}_1(v_i)}w_{ij}\left(f\left(v_j\right)-f\left(v_i\right)\right), \tag{5.9}$$

其中，权重 w_{ij} 取值时要保证矩阵 $\mathbf{L}$ 是对称的。前文已提及过两种取值方法：

- 均匀形式：$w_{ij}=1$；
- 余切形式：$w_{ij}=(\cot\alpha_{i,j}+\cot\beta_{i,j})$。

可以发现，函数 e_{ω} 是拉普拉斯算子的特征函数，因为：

$$\begin{gathered}\Delta\left(e^{2\pi\mathrm{i}\omega x}\right)=\frac{\mathrm{d}^2}{\mathrm{d}x^2}e^{2\pi\mathrm{i}\omega x}=-(2\pi\omega)^2e^{2\pi\mathrm{i}\omega x},\\ \Delta e_i=\lambda_i e_i.\end{gathered} \tag{5.10}$$

这样一维傅里叶变换中的基就是拉普拉斯——贝尔特拉米算子的特征矩阵，很自然的可以想到，对于二维流型曲面同样成立。

在处理离散形式时，e_{ω} 变为矩阵 $\mathbf{L}$ 的特征向量 $\mathrm{e}_1,...,\mathrm{e}_n$，对于 e_i，用如下逐顶点的矩阵来表示

$$f=\sum_{i=1}^{n}\langle\mathrm{e}_i,\mathrm{f}\rangle\,\mathrm{e}_i, \tag{5.11}$$

其中 e_i 的特征值代表了点 v_i 所处频域的频率，$\mathrm{e}_i(v_k)$ 表示顶点的振幅。

如果要滤掉高频部分，那么只需要对前 m 个特征向量进行累加即可：

$$\tilde{\mathrm{f}}=\sum_{i=1}^{m}\langle\mathbf{e}_i,\mathrm{f}\rangle\,.\mathrm{e}_i \tag{5.12}$$

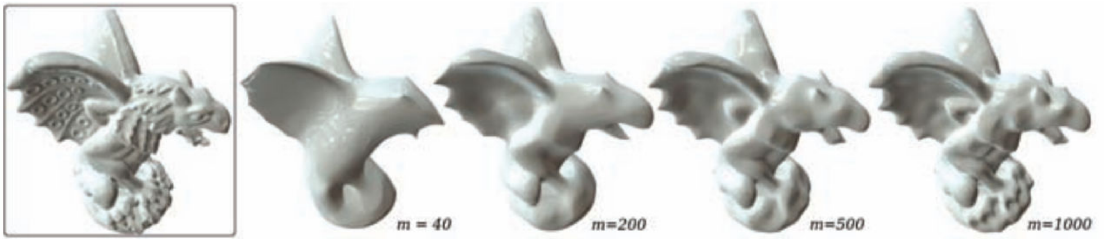


图 5.1　流型谐波基函数的数量 m 增加时，模型重构质量随之变化的示意图。

如图 5.1所示，随着 m 的逐渐减小，即滤掉的频率越来低，模型的凸出部分的细节信息逐渐地消失。

为了得到特征向量需要对拉普拉斯矩阵进行特征分解，而当模型的顶点比较多的时候，代价是十分昂贵的。

而下面的扩散流（Diffusion Flow）方法则相对来说更容易实现，效率也更高。

5.2　扩散流

诸如热扩散和布朗运动之类的物理过程可以使用下面的扩散方程来表示

$$\frac{\partial f(\mathbf{x},t)}{\partial t}=\lambda\Delta f(\mathbf{x},t). \tag{5.13}$$

从形式上看，这是一个二阶线性偏微分方程，通常 $f(\mathbf{x},t)$ 表示某点 $\mathbf{x}$ 在 t 时刻的温度，这个方程描述了无体内热运动的规律。

要将其运用到曲面网格上，首先是将连续形式的拉普拉斯——贝尔特拉米算子替换为离散形式，然后将函数 f 改写为逐顶点形式：

$$\frac{\partial}{\partial t}f\left(v_i,t\right)=\lambda\Delta f\left(v_i,t\right),\quad i=1,\ldots,n \tag{5.14}$$

上式采用矩阵形式以简介表示：

$$\partial\mathbf{f}(t)/\partial t=\lambda\mathbf{L}\mathbf{f}(t), \tag{5.15}$$

其中，等式左边的偏导数可以改写成微商的形式：

$$\frac{\partial\mathbf{f}(t)}{\partial t}\approx\frac{\mathbf{f}(t+h)-\mathbf{f}(t)}{h}, \tag{5.16}$$

化简得到

$$\mathbf{f}(t+h)=\mathbf{f}(t)+h\frac{\partial\mathbf{f}(t)}{\partial t}=\mathbf{f}(t)+h\lambda\mathbf{L}\mathbf{f}(t). \tag{5.17}$$

为了保证 h 比较大时的准确性，通常会将上式改写成如下形式：

$$\mathbf{f}(t+h)=\mathbf{f}(t)+h\lambda\mathbf{L}\mathbf{f}(t+h), \tag{5.18}$$

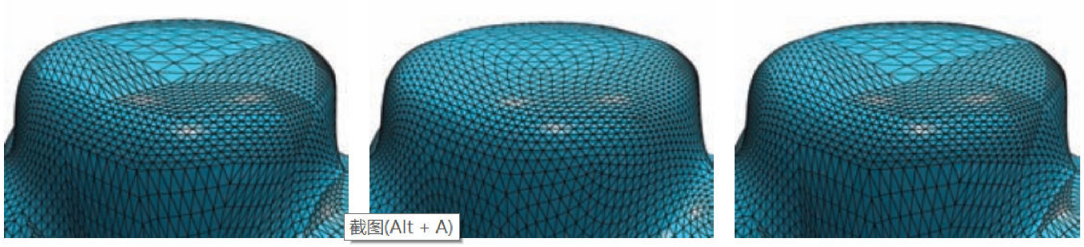


图 5.2　模型平滑示意图。原模型（左），均匀形式（中），余切形式（右）。

然后将自变量相同的移到一边，写成如下矩阵方程：

$$(\mathbf{Id} - h\lambda\mathbf{L})\mathbf{f}(t+h) = \mathbf{f}(t). \tag{5.19}$$

最后要做的就是用上式去更新网格的每一个定点：

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + h\lambda\Delta\mathbf{x}_i. \tag{5.20}$$

由于定点的拉普拉斯——贝尔特拉米算子等于其平均曲率法向量：

$$\Delta_{\mathcal{S}}\mathbf{x} = -2H\mathbf{n}, \tag{5.21}$$

所以上式实际上等价于让每一个定点沿着其法向量的方向移动，移动距离由这一点的平均曲率 H 决定。

值得注意的是，只有在算子的权重系数 w_{ij} 取法是余切形式时，定点才能按照上述方式更新。当系数的取法是均匀形式时，每个顶点会朝向重心方向移动。

扩散流比之前使用傅里叶变换来说计算量更低，但其主要思想仍然是移除高频噪音，即模型上不平滑之处而保留低频部分。

5.3　抹平

抹平（Fairing）的思想与前两种不同，其目标是通过计算使得网格尽可能地光滑。判定光滑程度的方法各不相同，但是总的原则是要尽可能的光滑，避免不必要的细节或毛刺。

抹平的总思路是：

1. 定义一个量来衡量曲面的不平滑程度；
2. 更新曲面以最小化曲面的不平滑程度。

一个较常用的描述不平滑程度的函数是使用曲面面积：

$$E_{\mathrm{M}}(\mathbf{x}) = \iint_{\Omega} \sqrt{\det(\mathbf{I})}\mathrm{d}u\mathrm{d}v. \tag{5.22}$$

在限定了曲面边界的情况下，不平滑的曲面相对的面积应该更大。当其取最小值的时候，外形应接近没有鼓起来的被夹紧的肥皂泡一样。

由于该函数高度非线性，包含第一基本型行列式的平方根，因此其计算效率较低。其改进形式如下：

$$\tilde{E}_{\mathrm{M}}(\mathbf{x}) = \iint_{\Omega} \|\mathbf{x}_u\|^2 + \|\mathbf{x}_v\|^2 \,\mathrm{d}u\mathrm{d}v. \tag{5.23}$$

为了求最小值，简化上面的模型至一维的情况：

$$E(f) = \int_a^b (f_x)^2 \,\mathrm{d}x. \tag{5.24}$$

在一维的情况下，限制条件就从边界变成了区间。假设当函数 E 的自变量函数取 f 时 E 取到最小值，并且限制条件的区间为 $[a,b]$，那么对于任意函数 u，并且 $u(a)=u(b)=0$，都有 $E(f)<E(f+\lambda u)$，那么当 $\lambda=0$ 时，E 取最小值，其对于 λ 的偏导数的值也为 0：

$$\left.\frac{\partial E(f+\lambda u)}{\partial \lambda}\right|_{\lambda=0} = \int_a^b 2 f_x u_x = 0. \tag{5.25}$$

用分部积分公式展开上面的积分：

$$\int_a^b 2 f_x u_x = 2 f_x u_x|_a^b - \int_a^b 2 f_{xx} u_x. \tag{5.26}$$

因为 $u(a)=u(b)=0$，所以最终：

$$\int_a^b f_{xx} u_x = 0. \tag{5.27}$$

由于上式对于任意满足 $u(a)=u(b)=0$ 的函数 u 都成立，所以

$$f_{xx} = \Delta f = 0. \tag{5.28}$$

上式推广到二维情况同样试用，所以对于改进后的函数同样可以用该方法：

$$\tilde{E}_{\mathrm{M}}(\mathbf{x}) \to \min \quad \Leftrightarrow \quad \Delta \mathbf{x}(u,v) = 0 \text{ for } (u,v) \in \Omega. \tag{5.29}$$

为了将上述方法应用至三角形网格的曲面上，只需要将算子和函数改为离散形式即可：

$$\mathrm{L}\mathbf{x} = 0. \tag{5.30}$$

除了用面积来衡量不光滑程度之外，还可以使用曲面的主曲率：

$$E_{\mathrm{TP}}(\mathbf{x}) = \iint_{\Omega} \kappa_1^2 + \kappa_2^2 \mathrm{d}u\mathrm{d}v, \tag{5.31}$$

或者主曲率相对切向量的变化率：

$$\iint_{\Omega} \left(\frac{\partial \kappa_1}{\partial \mathbf{t}_1}\right)^2 + \left(\frac{\partial \kappa_2}{\partial \mathbf{t}_2}\right)^2 \mathrm{d}u\mathrm{d}v. \tag{5.32}$$

图 5.3　不同抹平参数 k 对模型平滑影响的示意图。（左）$k=1$，（中）$k=2$，（右）$k=3$。

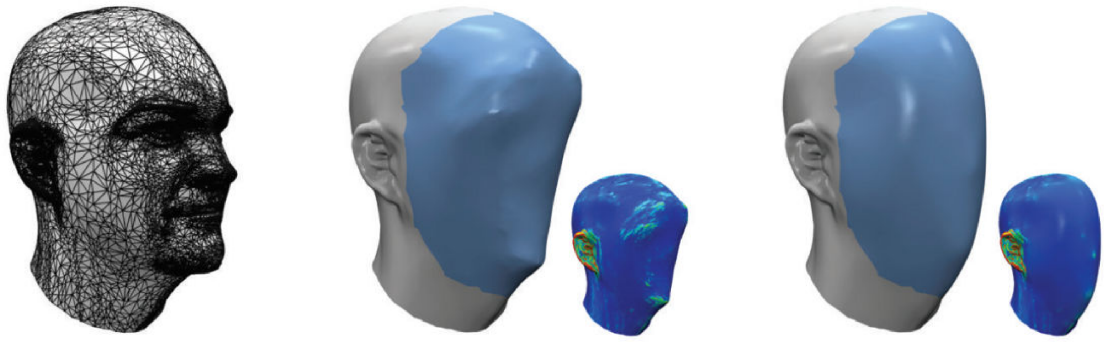


图 5.4　不同拉普拉斯——贝尔特拉米算子求解 $\Delta^2 x=0$ 后的示意图。（左）原模型，（中）均匀形式，（右）余切形式。

当 $k=1$ 时，以面积为度量的不平滑度取最小值；$k=2$ 时以主曲率为度量的不平滑度取最小值；$k=3$ 时以主曲率相对切向量的变化率为度量的不平滑度取最小值：

$$\Delta^k \mathbf{x}=0. \tag{5.33}$$

如图 5.3所示，紫色的区域通过上述方法平滑得到。

算子中权重取法的不同也会产生不同的结果。如图 5.4所示，使用均匀方式会导致部分区域的顶点密度过高，使用余切形式则能够达到期望的平滑效果。

第六章　参数化

6.1　表面参数化

参数化（Parameterization）的主要目标是将复杂的 3 维模型转换到 2 维空间上，如图 6.1所示。对于一个三角形网格模型来说，就是把它拍平到一个平面上。

从数学角度上分析，对三角形网格参数化的过程就是寻找一种映射，它能够把网格的每一个顶点 i 映射到 (u_i, v_i) 上。值得注意的是，经过映射后，任意两个三角形公共的部分只可能是：一条边，一个顶点或者不存在。

6.2　重心映射

重心映射（Barycentric Mapping）是一种在三角化网格中较为常用的参数方法。使用此方法的网格必须满足以下条件：

- 三角形网格与圆盘是同胚的，即网格必须有边界且不能有洞，并且网格的边界在一个凸多边形上，同时内部的顶点是其周围顶点的凸组合。

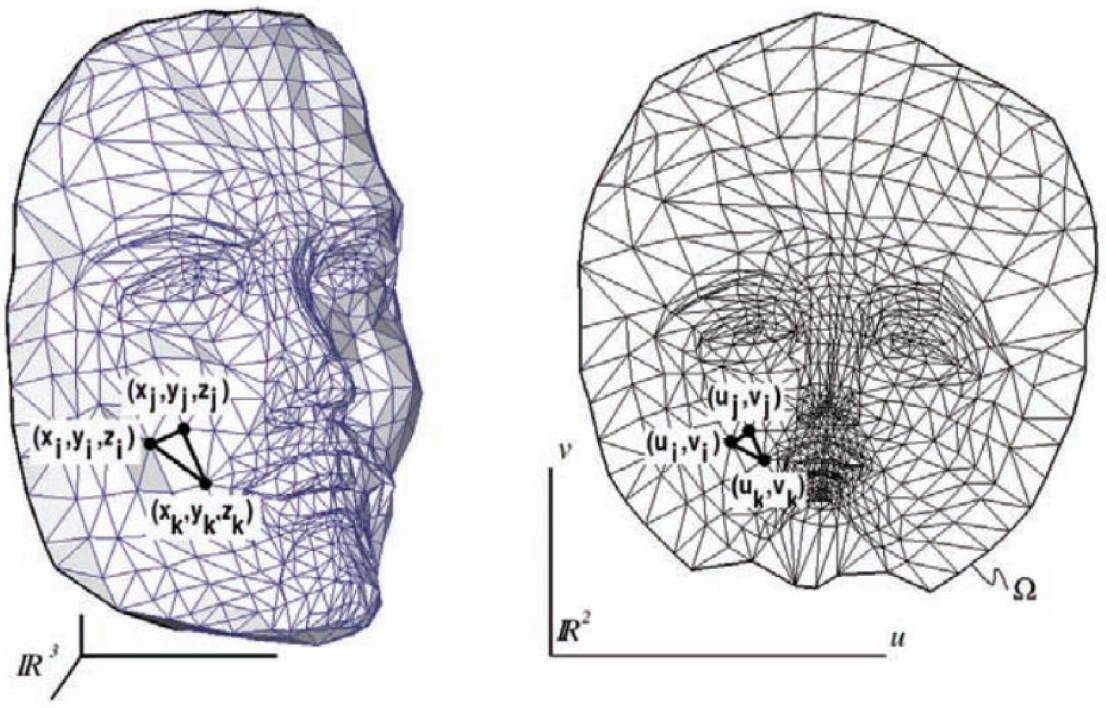


图 6.1　参数表面化示意图。

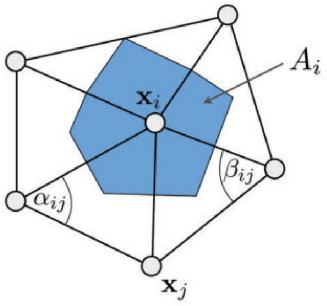


图 6.2　离散余切拉普拉斯的夹角和面积示意图。

假设顶点的索引记作内部的顶点（1 到 n）排列和边界上的顶点（$n+1$ 到 N）排列，那么，对于任取实数 λ，满足：

$$\lambda_{i,j}=0,\quad (i,j)\notin E,\quad \lambda_{i,j}>0,\quad (i,j)\in E,\quad \sum_{j=1}^{N}\lambda_{i,j}=1, \tag{6.1}$$

其中 $(i,j)\in E$ 表示点 i 和点 j 相邻。

然后将处于边界的点（索引为 $n+1$ 到 N 的点）固定，用下面的线性方程去更新处于内部的顶点：

$$\mathbf{u}_i-\sum_{j=1}^{n}\lambda_{i,j}\mathbf{u}_j=\sum_{j=n+1}^{N}\lambda_{i,j}\mathbf{u}_j,\quad i=1,\ldots,n \tag{6.2}$$

写成矩阵的形式就是要分别求解下面两个方程：

$$\begin{aligned} A\mathbf{u}&=\mathbf{b}_1,\\ A\mathbf{v}&=\mathbf{b}_2. \end{aligned} \tag{6.3}$$

其中，矩阵 A 的元素 a_{ij} 的取法有三种，其中最为简单的是当下标 i 和 j 不相等时取 1，下标 i 和 j 相等时取顶点 i 周围顶点个数的相反数。这种取法的优势是简单并且计算较快，缺点是没有考虑到网格的集合属性，例如边的长度和三角形的角度，所以会不可避免地产生形变。

6.2.1　离散拉普拉斯变换

另一种方法就是使用下面的余切形式：

$$\begin{aligned} a_{i,j}&=\frac{1}{2A_i}\left(\cot\alpha_{i,j}+\cot\beta_{i,j}\right),\\ a_{i,i}&=-\sum_{j\neq i}a_{i,j}, \end{aligned} \tag{6.4}$$

其中 $\alpha_{i,j}$，$\beta_{i,j}$ 以及 A_i 的含义，如图 6.2所示。

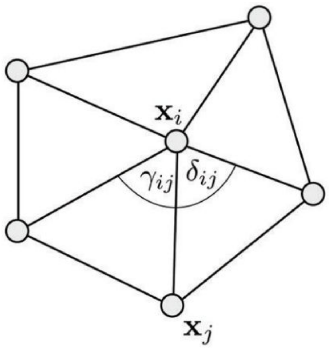


图 6.3　均值坐标的夹角和面积示意图。

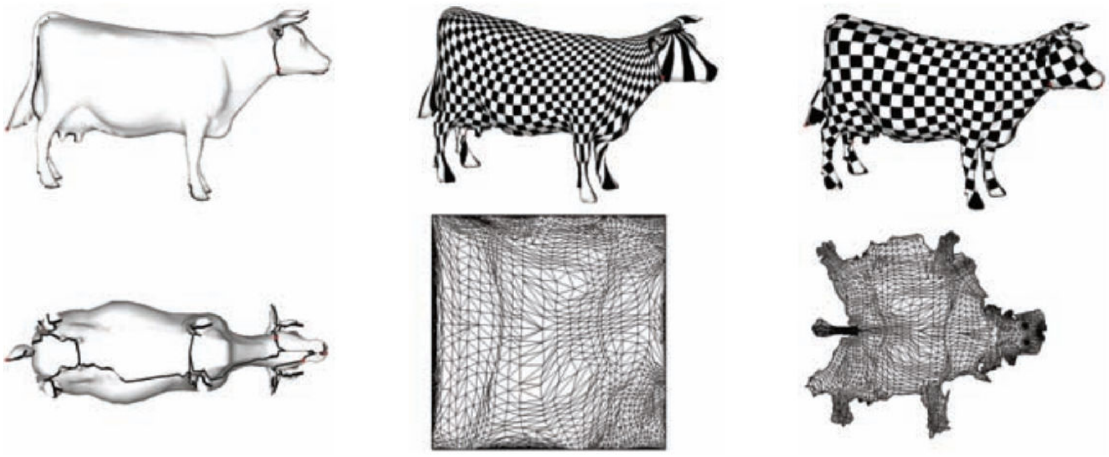


图 6.4　不同边界选取与固定的选取对网格参数化的影响示意图。

如果 $\alpha_{i,j}+\beta_{i,j}>\pi$，那么余切形式就可能出现负数。通过网格面分割求解，也可通过下式取法求解：

$$a_{i,j}=\frac{1}{\|\mathbf{x}_i-\mathbf{x}_j\|}\left(\tan\left(\frac{\delta_{i,j}}{2}\right)+\tan\left(\frac{\gamma_{i,j}}{2}\right)\right),\quad a_{i,i}=-\sum_{j\neq i}a_{i,j},\tag{6.5}$$

其中 $\gamma_{i,j}$ 和 $\delta_{i,j}$ 的含义由如图 6.3所示。

无论使用上面哪种方法，都需要手动指定并固定网格的边界，但是对于任意一个模型来说，往往选取边界不是一件容易的事情。边界的选取与固定会影响到参数化的结果，如图 6.4所示。

6.3　保角映射

对于保角度映射（Conformal Mapping），引用之前的例子：假设参数空间 (u,v) 上存在一个小圆，使用保角映射将它变换到 3 维坐标空间之后，其沿着 u 和 v 方向的切向量 $\mathbf{x}_u$ 和 $\mathbf{x}_v$ 的模长应该同样地是相等并且 $\mathbf{x}_u$ 和 $\mathbf{x}_v$ 正交。对于此变化的逆变化，即从 3 维坐标空间变换为 2 维参数空间也同样成立。

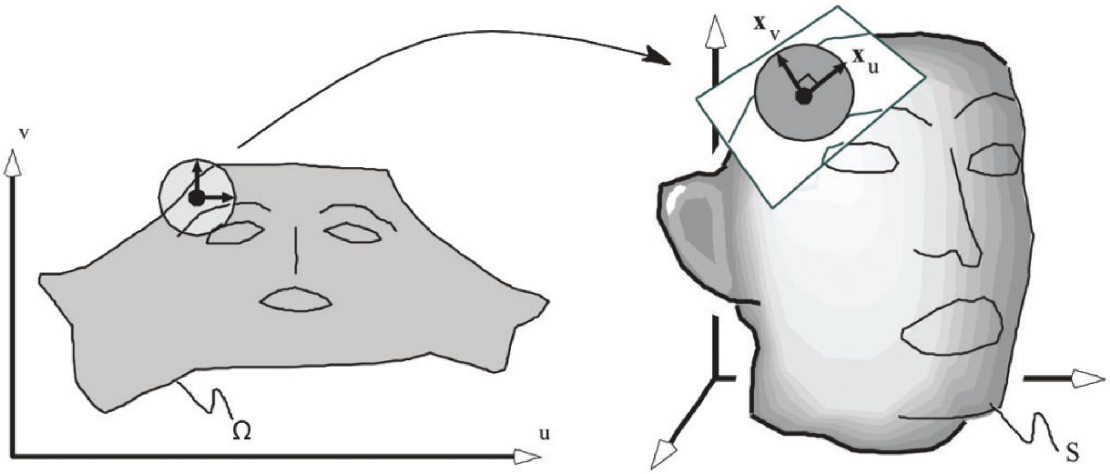
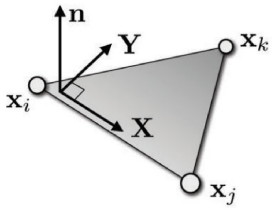


图 6.5　保形参数化映射示意图。



$$\begin{aligned}\mathbf{X} &= \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|}\\ \mathbf{n} &= \frac{\mathbf{X} \times (\mathbf{x}_k - \mathbf{x}_i)}{\|\mathbf{X} \times (\mathbf{x}_k - \mathbf{x}_i)\|}\\ \mathbf{Y} &= \mathbf{n} \times \mathbf{X}\end{aligned}$$

图 6.6　三角形局部 X, Y 基。

6.3.1　三角形梯度

与之前讨论梯度时的情况不同，参数化的函数是一个从 3 维曲面到平面的映射，如图 6.5所示。为了定义这个函数的梯度，对于每一个三角形，选择一个顶点，并以该顶点为原点建立一组正交基。在这样一组基向量下，函数的梯度的定义是：

$$\nabla u = \begin{bmatrix} \partial u/\partial X \\ \partial u/\partial Y \end{bmatrix} = \underbrace{\frac{1}{2A_T}\begin{bmatrix} Y_j - Y_k & Y_k - Y_i & Y_i - Y_j \\ X_k - X_j & X_i - X_k & X_j - X_i \end{bmatrix}}_{=\mathbf{M}_T} \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix}, \tag{6.6}$$

其中对于每一个三角形矩阵 M 是一个常数，A 代表了三角形的面积。由于保角映射的逆映射同样也是保角的，并且参数空间下 u 和 v 分量的基向量是垂直的，所以：

$$\nabla v = \mathbf{n} \times \nabla u. \tag{6.7}$$

6.3.2　最小平方保角映射

上述公式也可用矩阵的形式来表示：

$$\nabla v = (\nabla u)^{\perp} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \nabla u, \tag{6.8}$$

其中，垂直符号代表向量沿着顺时针的方向以法向量为轴旋转得到的新向量。

然后将之前求得的梯度带入到上式，得到：

$$\mathbf{M}_T \begin{pmatrix} v_i \\ v_j \\ v_k \end{pmatrix} - \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \mathbf{M}_T \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{6.9}$$

在连续集上，黎曼几何阐述了一个事实：任何的曲面都能够进行保角参数化。但是，在分片线性函数表示的三角形网格中，只有**可展曲面**才能进行保角参数化。对于一般的曲面（非可展曲面），上式无法成立，即梯度向量 u 和 v 不一定垂直，但是要尽可能地让它们垂直，所以将上式改为最小二乘的形式：

$$E_{\text{LSCM}} = \sum_{T=(i,j,k)} A_T \left\| \mathbf{M}_T \begin{pmatrix} v_i \\ v_j \\ v_k \end{pmatrix} - \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \mathbf{M}_T \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix} \right\|^2. \tag{6.10}$$

上面的方程被称为**保角能量**，注意到保角能量的值不会受到参数空间的平移和旋转变换的影响，所以并不存在唯一的最小解，一般在实际应用的时候会至少固定两个点 (u, v) 的位置，然后再去求其最小值。

6.3.3　保角映射和谐波映射

在之前进行曲面平滑的过程中，为了让曲面变得更平滑，在边界固定的情况下考虑最小化曲面的面积。由于求曲面的面积的方程中包含了平方根和第一基本型，是高度非线性化的，为了简化运算引入了**狄利特雷能量**（Dirichlet's Energy）的概念。而在保角映射中使用到了保角能量 (Conformal Energy) 的概念，当保角能量为 0 时说明该参数化的过程是保角的。

而上述的 3 个量存在以下关系：

$$\underbrace{\int_\Omega \sqrt{\det(\mathbf{I})}\mathrm{d}A}_{\text{曲面面积}} = \underbrace{\frac{1}{2}\int_\Omega \|\mathbf{x}_u\|^2 + \|\mathbf{x}_v\|^2 \,\mathrm{d}A}_{\text{狄利特雷能量}} - \underbrace{\frac{1}{2}\int_\Omega \left\|\mathbf{x}_v - (\mathbf{x}_u)^{\perp}\right\|^2 \mathrm{d}A}_{\text{保角能量}}. \tag{6.11}$$

上式容易推导，只需将等号两边的积分展开就能证明。

曲面平滑的过程中最小化了狄利特雷能量，而在上面的保角映射的过程中使用了最小二乘的思想来最小化保角能量。因为这两个量之间只相差了曲面面积，所以从某种角度上来说，这两种方法是等价的。

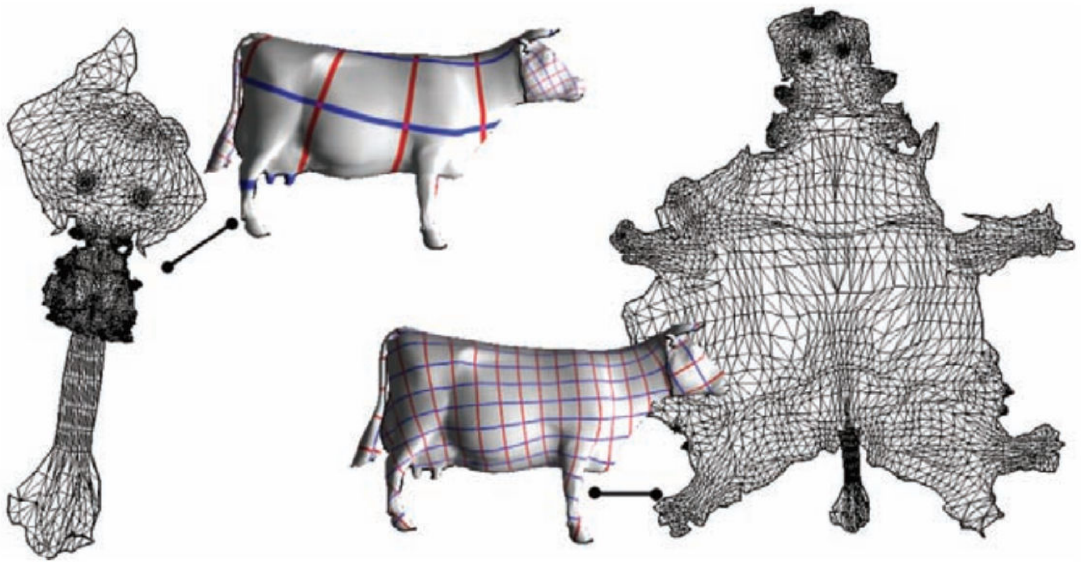


图 6.7　对于高斯曲率较大的曲面，采用保形方法可能会产生高度失真（左图）。ABF 方法及其变体可以更好地平衡失真并给出更好的结果（右图）。

6.3.4　基于几何方法的保角映射

以上所讨论的都是基于分析的方法，基于分析的方法是易于实现的，因为它需要的只是最小化一个二次的方程。但是上面的方法均需要固定至少两个顶点，而这将可能会导致变形。

对于一个高斯曲率较大的曲面，上述保角变换的方法往往会造成诸如图 6.7左图那样的变形，而用户期望的结果一般是像右图那样的。

下面介绍一种**基于角度的拍平化算法**（Angle-Based Flattening，简称 ABF）。由于参数空间是一个二维的三角剖分（Triangulation），并且由某一个顶点周围所有的角唯一地定义，所以可以重新定义解决参数化的问题的思路，即通过角来寻找顶点的坐标 (u, v)。

ABF 方法是通过最小化下面的能量来寻找参数化的结果的：

$$E_{\text{ABF}}(\boldsymbol{\alpha}) = \sum_{T\in\mathcal{F}}\sum_{k=1}^{3}\left(\frac{\alpha_k^T - \beta_k^T}{\beta_k^T}\right)^2, \tag{6.12}$$

其中 α 与 β 项表示与顶点 k 相邻的，处于三角形 T 中角度的大小，α 表示的参数化平面上的角度而 β 表示的是原来 3D 网格上对应位置的角度。

为了保证得到的角度 α 所定义的三角剖分是合法的，需要对其施加一定的限制条件：

- 同一个三角形的内角和必须为 180°：

$$\forall T \in \mathcal{F}: \quad \alpha_1^T + \alpha_2^T + \alpha_3^T = \pi. \tag{6.13}$$

- 对于处于内部的顶点，其周围相邻的角的度数之和为 360°：

$$\forall v \in \mathcal{V}_{\text{int}} : \sum_{(T,k)\in v^*} \alpha_k^T = 2\pi. \tag{6.14}$$

其中，$\mathcal{V}_{\text{int}}$ 表示非边界上的点的集合，v^* 表示与顶点 v 相邻角度的集合。

对于处于内部的顶点，其周围相邻的角对应所处的三角形中的下一个角的正弦值的积等于其周围相邻的角对应所处的三角形中的上一个角的正弦值的积：

$$\forall v \in \mathcal{V}_{int} : \prod_{(T,k)\in v^*} \sin \alpha_{k\oplus 1}^T = \prod_{(T,k)\in v^*} \sin \alpha_{k\ominus 1}^T. \tag{6.15}$$

6.4　基于形变分析的方法

进行纹理映射时，如何减小形变是一个非常重要的问题。

两个切向量从参数空间变换到曲面上时，两向量的夹角一由用下式计算：

$$\frac{\mathbf{w}_1^T \mathbf{w}_2}{\sqrt{\mathbf{w}_1^T \mathbf{w}_1} \cdot \sqrt{\mathbf{w}_2^T \mathbf{w}_2}} = \frac{\overline{\mathbf{w}}_1^T \mathbf{I}(\mathbf{u}) \overline{\mathbf{w}}_2}{\sqrt{\overline{\mathbf{w}}_1^T \mathbf{I}(\mathbf{u}) \overline{\mathbf{w}}_1} \cdot \sqrt{\overline{\mathbf{w}}_2^T \mathbf{I}(\mathbf{u}) \overline{\mathbf{w}}_2}}. \tag{6.16}$$

当第一基本型是单位矩阵的倍数或者奇异值相同时，向量的角度得到了保留。当所有的顶点都满足该要求时，这个变换是**保角**的。此时，圆形变换之后还是圆形，但是它的半径可能发生改变。

曲面某区域面积表达为：

$$\int_U \sqrt{\det(\mathbf{I})} \mathrm{d}A. \tag{6.17}$$

当第一基本型行列式的值为 1 或者所有奇异值相乘结果为 1 时，圆形的面积在变换后不会发生变化，此时这个时称变换是**保面**的。

当上述条件同时满足，即第一基本型是单位矩阵或者奇异值为 1 时，则称变换是**等距**的。因为在这种变换中角度和面积都没有发生变形，这是一种较为理想的情况。只有一些特殊的曲面，如**可展曲面**才能满足，因为这种曲面的高斯曲率处处为零。

6.4.1　分段线性曲面的度量属性

如何计算第一基本型？在进行保角映射的时候，我们定义了参数空间下的梯度，利用它可以得到此映射下的雅可比矩阵

$$\mathbf{J}_T = \begin{bmatrix} \partial u/\partial X & \partial v/\partial X \\ \partial u/\partial Y & \partial v/\partial Y \end{bmatrix} = [\mathbf{u}_X, \mathbf{u}_Y]. \tag{6.18}$$

利用雅可比矩阵，可以得到第一基本型：

$$\mathbf{I}_T = \mathbf{J}_T^T \mathbf{J}_T. \tag{6.19}$$

并且，对于每一个三角形 T，雅可比矩阵 $\mathbf{J}$ 和第一基本型 $\mathbf{I}$ 都是常量。

接下来介绍两种具体的方法：最等距的表面参数化和专用信号参数化。

6.4.2 最等距的表面参数化

最等距的表面参数化（Most Isometric Parameterization of Surfaces，简称 MIPS）是一种基于形变分析的方法。这种方法的主要思想是减小变换过程中小椭圆的两个轴长 δ_1 和 δ_2 的比值：

$$E_2(\mathbf{J}_T) = \|\mathbf{J}_T\|_2 \left\|\mathbf{J}_T^{-1}\right\|_2 = \sigma_1/\sigma_2. \tag{6.20}$$

对上式进行最小化比较复杂，因此可以将上式中的 2 范数改成 Frobenius 范数，然后将其化简可以得到：

$$E_{\text{MIPS}}(\mathbf{J}_T) = \|\mathbf{J}_T\|_F \left\|\mathbf{J}_T^{-1}\right\|_F = \frac{\operatorname{trace}(\mathbf{I}_T)}{\det(\mathbf{J}_T)}, \tag{6.21}$$

其中第一基本型的迹可看作是狄利特雷能量，而雅可比矩阵行列式的值可以看作是 3 维空间和参数空间下三角形面积之比。

6.4.3 专用信号参数化

专用信号参数化（Signal-Specialized Parameterization）是一种基于纹理映射的方法。一般我们将纹理从参数空间映射到曲面上，而纹理变形的程度可以通过观察纹理上的某一个点在某一个方向上被拉伸的程度。

对于某一个特定的三角形，一般使用下式表示其在所有方向上平均被拉伸的程度：

$$E_{\text{stretch}}(T) = \sqrt{\left((1/\sigma_1)^2 + (1/\sigma_2)^2\right)/2}. \tag{6.22}$$

将网格上所有三角形按照面积进行加权组合，可得：

$$E_{\text{stretch}}(\mathcal{S}) = \sqrt{\frac{\sum_T A_T E_{\text{stretch}}(T)}{\sum_T A_T}}. \tag{6.23}$$

第七章　网格重划分

网格重划分（Remeshing）的定义如下：

- 输入一个 3D 网格，通过计算得到另一个和输入大致相同且满足一定质量要求网格。

曲面的网格重划分目标有以下两点：

1. 根据需求减少曲面的复杂度；
2. 改善曲面的质量。

曲面的质量指非拓扑属性，包括采样密度，正则性，大小，方位，对齐性，以及曲面网格的形状。

7.1　局部结构

网格的局部结构说的是网格元素的种类，形状，方位以及分布情况。

元素的种类

最为常用的两个种类是**三角形网格**和**四边形网格**。四边形网格可以通过在每一个四边形中插入一条对角线，轻易地转换成三角形网格。反过来，如果要把三角形网格转换为四边形网格，可以使用重心划分的方法：将三角形的重心和每一条边的中点连接，此时一个三角形被划分成了 3 个四边形。还有另一种方法：将三角形的重心和每一个顶点连接，舍弃网格上原来所有三角形的边。

元素的形状

网格的元素可以分为**各向同性**或者**各向异性**两类。如图 7.1所示，各向同性的元素的形状通常在各个方向上一致。理想情况下，当某个三角形（四边形）的元素是或者近似是等边三角形（正方形）时，就说这个元素是各向同性的。对于三角形的元素来说，可以通过其外接圆的半径和三边中最短的一条边的比值来度量它具有的各向同性的强度。

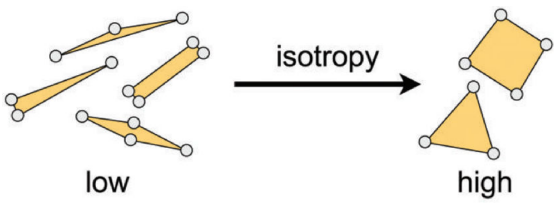


图 7.1　各向同性示意图。右图比左图更强。

各向异性的元素在网格曲面上各个方向的形状往往都不同，通常这些元素的都朝向主曲率的方向。这种元素往往能够更好的表现几何体的结构特征。它的另一个优势在于，相对于各向同性，得到同样质量的网格其使用的元素的个数更少。

元素的密度

在一个均匀分布的网格中，网格元素均匀地分布在整个模型上。在一个不均匀或者自适应分布的网格中，每个区域分布的元素的数量都不同，例如比较小的元素通常会较多地处于曲率较大的区域。通过调整，这种不均匀或者自适应分布的网格能够用更少的元素更好地近似出原来的网格。

元素的对齐性和方位

在网格重划分的过程中，一些尖锐突出的区域通常会受到影响产生走样，而这些区域的切线不连续。为了避免此情况，需要将网格元素与它们对齐。

7.2　全局结构

当三角形网格上非边界上的顶点周围顶点的数量为 6 或者边界上的顶点周围顶点的数量为 4 时，称这个顶点是**正则的**。同理对于四边形网格来说，这两个值对应分别是 4 和 3。相对于正则，其它的顶点则是非正则。网格的全局结构可以根据正则顶点的数目被分为非正则，半正则，高度正则和正则四种。

7.3　一致性

所有网格重划分算法都会在曲面上或者曲面附近计算点的位置。其中大部分算法甚至还会进行额外的迭代来修正顶点的位置以改善网格的质量。所以在网格重划分的过程中一个关键的问题就是要保证计算前后顶点的一致性。

以下有几种解决方案：

- 全局参数化：将输入的模型整个参数化到一个二维参数域上，然后在这个参数域对采样点的分布和位置进行调整，最后再将其还原到三维空间中；
- 局部参数化：算法只保留某个点局部的参数化信息。当采样点变换的时候，再计算那个点的局部参数化信息；
- 投影：将采样点投影到输入模型上对应最近的顶点、边或者三角形上。

全局参数化的计算相对来说比较耗时，尤其是当模型被切开成一个圆盘时。而当采样点离曲面太远时，直接进行投影则可能导致重影现象。通过将对采样点的移动限制在切平面上使得采样点不会原来曲面太远，以解决上述问题。局部参数化方法相对稳定，并且效果较好，不过需要不断的跟踪记录并重新计算局部的参数化信息。

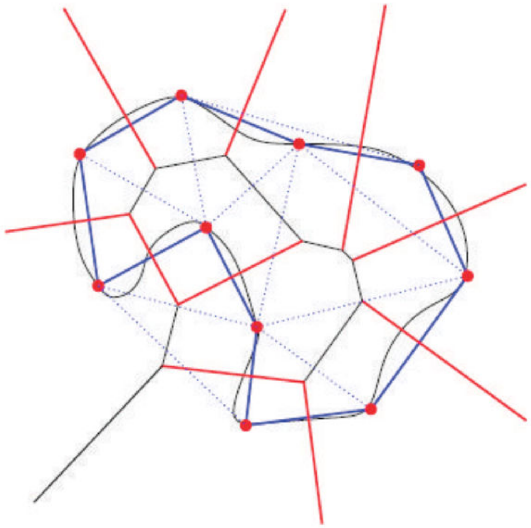


图 7.2　德洛内三角剖分。

7.4　沃洛诺伊图和德洛内三角剖分

沃罗诺伊图（Voronoi Diagrams）是由一组特定点将空间分割成的不同的区域。而每一个区域都只包含这些点中的一个，并且该区域内任意点到该特定点的距离小于任意点到空间中其它特定点的距离。其中这些被分割的区域称作沃罗诺伊区域。

以下为数学上的定义。给定任意维空间 $\mathbb{R}^d$ 上的一个点的集合 $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$，点 $\mathbf{p}_i$ 的沃罗诺伊区域 $V(\mathbf{p}_i)$：

$$V(\mathbf{p}_i) = \left\{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x} - \mathbf{p}_i\| \leq \left\|\mathbf{x} - \mathbf{p}_j\right\|, \forall j \neq i\right\}. \tag{7.1}$$

沃罗诺伊图可看作空间 $\mathbb{R}^d$ 上的一个划分，因为该空间上任意一个点必定属于某一个沃罗诺伊区域。

与该空间中的任意两个点 $\mathbf{p}_i$ 和 $\mathbf{p}_j$ 等距的点组成的区域被称作平分面（Bisector），所有的平分面都是该空间上的仿射子空间。例如，在二维空间上平分面是一条线，三维空间上平分面是一个面。

从平分面的定义可发现，沃罗诺伊区域也可以被定义为由一些平分面的半空间（Half-Space）相交所围成的闭合区域。因为凸集相交仍然是凸的，所以沃罗诺伊区域是一个凸集。不过，在靠近整个空间 $\mathbb{R}^d$ 边缘的区域，闭合组成沃罗诺伊区域的平分面里就会包含 $\mathbb{R}^d$ 的外壳部分。

沃罗诺伊图的对偶结构被称为**德洛内三角剖分**（Delaunay Triangulations）。通过连接沃罗诺伊区域内的顶点可以得到其对应的对偶结构，如图 7.2所示。

德洛内三角剖分的三角形 $(\mathbf{p}, \mathbf{q}, \mathbf{r})$ 与沃罗诺伊区域 $V(\mathbf{p})$，$V(\mathbf{q})$，$V(\mathbf{r})$ 相交得到的顶点对偶；德洛内三角剖分的边 $(\mathbf{p}, \mathbf{q})$ 与沃罗诺伊区域 $V(\mathbf{p})$，$V(\mathbf{q})$ 相交得到的边对偶；德洛内三角剖分的顶点 $\mathbf{p}$ 与沃罗诺伊区域 $V(\mathbf{q})$ 对偶。

可以发现，德洛内三角剖分和沃罗诺伊图在许多局部和全局的属性上存在对偶关系，如：

- 对于 k 维空间上的某个点集 $\mathcal{P}$ 上的德洛内三角剖分，其上的任何一个 k 单纯形（k-simplex），即在 k 维空间上，由 $k+1$ 个点组成的多面体，如 2 维空间上的三角形，3 维空间上的四面等的外界圆的内部都不包含点集上其它的点。

- 对于二维空间上的某个点集 $\mathcal{P}$ 上的德洛内三角剖分，这种剖分能够最大化所有三角形中最小的角。

在进行德洛内三角剖分时，还可以对其加以限制，例如在二维空间上可以用闭合的平面曲线来进行限制，在三维空间上用闭合曲面来进行限制。如图 7.2所示，蓝色的曲线是作为限制的闭合曲线，红色的直线是与之相交的沃罗诺伊图的边。

7.5　三角网格网格重划分

7.5.1　贪心算法网格重划分

贪心算法网格重划分（Greedy Remeshing）的主要思想是对三维德洛内三角剖分的更新（Refining）和提取（Filtering）。

在更新的过程中，输入网格上的点会被选中，并插入到德洛内三角剖分的三角形中。而点的位置则是网格曲面与德洛内三角剖分对应的沃罗诺伊图的边的交点。也就是说，沃罗诺伊图的边被用来当作输入网格曲面的“探针”。

在提取的过程中，更新德洛内三角剖分使其被限制在曲面上，即选取德洛内三角剖分的面，与这些面对偶的沃罗诺伊图的边与该曲面相交。

该算法会涉及以下概念：

- **曲面德洛内球**（Surface Delaunay Ball）曲面德洛内球是一个位于输入网格曲面中心的球，包围一个德洛内三角剖分中的一个特定的面。一个中心位置为 **c**，半径为 r，包围了面 f 的曲面德洛内球可以记作：

$$B_f = B\left(\mathbf{c}_f, r_f\right) \tag{7.2}$$

- **中轴**（Medial Axis）给定 n 维空间上的一个集合 $\mathcal{O}$，它的中轴 $M(\mathcal{O})$ 是一系列点的集合，以这些点为中心的超球面与集合 $\mathcal{O}$ 的边界相切的点的个数至少为 2。

- **Medial ball** 中心在中轴上的球，其内部被集合 $\mathcal{O}$ 包含并且它的包围球面与 $\mathcal{O}$ 的边界相交，称这样的球为中轴球（Medial Ball）。

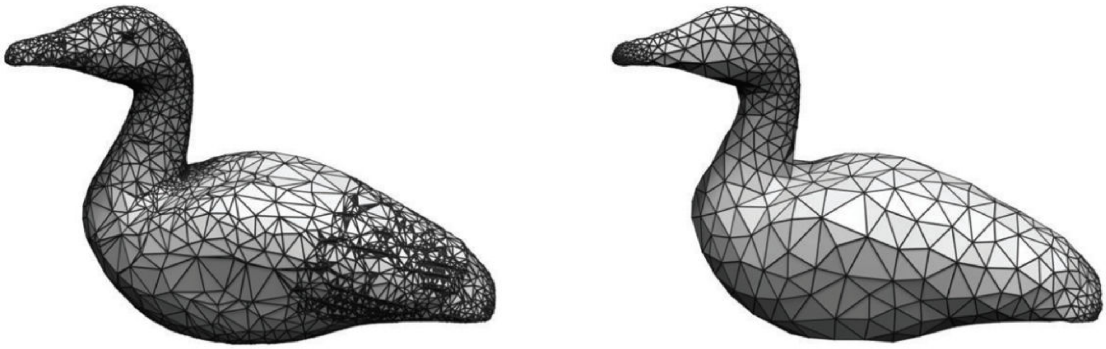


图 7.3　左图为原模型，右图为算法处理后的模型。

- **范围/局部特征尺寸**（Reach/Local Feature Size）集合 $\mathcal{O}$ 内的点 **x** 到集合 $\mathcal{O}$ 上的中轴的距离称为范围或局部特征尺寸。

有了以上定义便可准确地描述提取过程，即调整带限制的德洛内三角剖分，使得所有的曲面德洛内球的半径都小于局部范围，并保证算法能够正确终止还需要确保范围大于 0。

算法需要一个点集 $\mathcal{P}$，$\mathcal{P}$ 的德洛内三角剖分 Del($\mathcal{P}$)，带限制条件的德洛内三角剖分 $\mathrm{Del}_{\mathcal{S}}(\mathcal{P})$，以及 $\mathrm{Del}_{\mathcal{S}}(\mathcal{P})$ 中“不好”的面的列表 L。

“不好”的面的定义为：假定曲面德洛内球 $B_f = B(\mathbf{c}_f, r_f)$ 满足 $r_f > \psi(\mathbf{c}_f)$，其中 ψ 是定义在 $\mathcal{S}$ 上的函数，ψ 满足下列条件：存在 $\mathcal{S}$ 上的一个点 **x**，使得

$$\psi(\mathbf{x}) \geq \psi_{\text{inf}} > 0. \tag{7.3}$$

初始化时点集 $\mathcal{P}$ 选取 $\mathcal{S}$ 中每个连通区域上足够近的三个点，然后执行以下算法。

当 ψ 满足，其中 $\epsilon = 0.2$，$\rho = reach$，上述算法在有限次迭代之后终止，并且算法输出的结果：带限制的三维德洛内三角剖分与输入的网格曲面相互同胚。

$$\psi \leq \epsilon \cdot \rho. \tag{7.4}$$

上述算法由于大量涉及到求直线和三角形面求交的过程，所以采用八叉树的数据结构进行加速。贪心算法网格重划分的优点在于其结果保证不会出现自相交，且无需局部或者全局的参数化信息。

7.5.2　变分算法网格重划分

为了使用尽量少的顶点得到质量尽可能高的网格，则需采用变分算法网格重划分（Variational Remeshing）方法。

- **优化的标准是什么？**一些和形状、三角大小等相关的几何量。

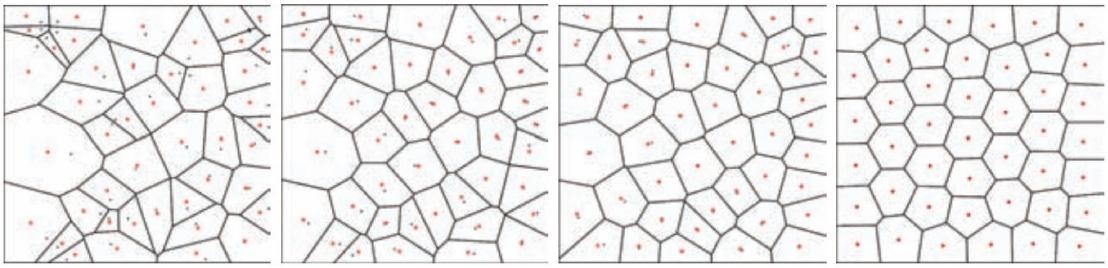


图 7.4　从左到右：原始的沃罗诺伊划分；经过一次迭代后的划分；经过三次迭代的划分；Centroidal Voronoi Tessellation。

- **自由度是多少？**尽可能地少。

变分算法网格重划分的主要思想是将一系列的点尽可能平均地放置到输入的网格之上。以 2D 的情况为例，如果要平均地将一系列点放置到一个平面上，其中一个方法是通过构建一个**重心沃洛诺伊镶嵌**（Centroidal Voronoi Tessellation，简称 CVT）来实现。给定一个边界域 Ω，如果存在一个被 Ω 限制的沃罗诺伊划分，其上的任意一个沃罗诺伊区域中的顶点刚好是这个区域的重心，那么称这个这个划分为 CVT。

求一个沃罗诺伊区域 V_i 的重心 $\mathbf{c}_i$ 的方法如下：

$$\mathbf{c}_i = \frac{\int_{V_i} \mathbf{x} \cdot \rho(\mathbf{x}) \mathrm{d}\mathbf{x}}{\int_{V_i} \rho(\mathbf{x}) \mathrm{d}\mathbf{x}} \tag{7.5}$$

$\rho(\mathbf{x})$ 为密度函数，通常取一个常数。

变分算法通常需要首先定义一个能量函数，然后通过不断地迭代最小化这个能量函数。该能量函数定义为：

$$E\left(\mathbf{p}_1, \ldots, \mathbf{p}_n, V_1, \ldots, V_n\right) = \sum_{i=1}^{n} \int_{V_i} \rho(\mathbf{x}) \left\|\mathbf{x} - \mathbf{p}_i\right\|^2 \mathrm{d}\mathbf{x} \tag{7.6}$$

当 $\mathbf{p}_i$ 是对应沃罗诺伊区域 V_i 上的重心时，上式的能量函数取最小值。

采用洛伊德算法（Lloyd's Algorithm），即通过不断迭代建立 CVT 的方式求解上述问题。给定一个密度函数 ρ 和一个点集 $\mathbf{p}_i$，算法由下面三个步骤组成：

1. 根据点集 $\mathbf{p}_i$ 建立沃罗诺伊划分；
2. 计算每一个沃罗诺伊区域的重心 $\mathbf{c}_i$，然后将 $\mathbf{p}_i$ 移动到 $\mathbf{c}_i$ 的位置；
3. 重复执行 (1)(2) 直到满足收敛条件。

为了能够在 3D 网格上应用同样的算法，我们先将网格进行保角参数化，然后在参数空间上应用洛伊德算法。参数化的过程中会导致三角形变形，此时密度函数 ρ 被用来抵消这种变形。通过在多个带边界限制的沃罗诺伊图上分别应用洛伊德算法，网格上一些诸如褶皱、夹角的特征能够得到保留。

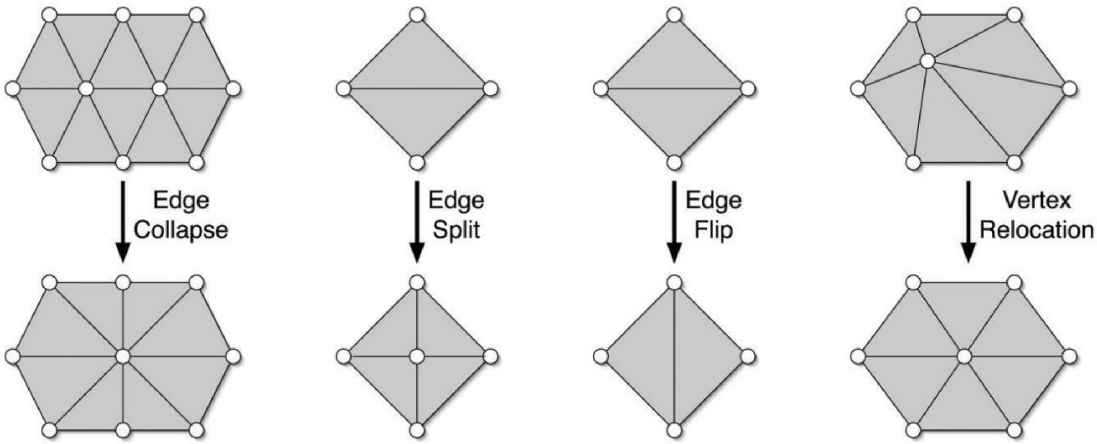


图 7.5　网格的基本操作

7.5.3　增量算法网格重划分

增量算法网格重划分（Incremental Remeshing）相较于变分算法网格重划分来说，实现起来更加简单。它无需进行参数化，且无需进行划分。

算法首先输入一个目标边长，然后根据这个输入对网格中较长的边进行**划分**操作，对较短的边进行**融合**操作，并且会移动顶点的位置，直到所有边的长度和输入的目标边长大致相当。

我们通过输入的长度得到一个区间 $[low, high]$。如果边长在区间左侧，则认为该边太短需要进行融合操作；如果边长在区间的右侧则认为这条边太长，需要进行划分操作。

$split\_long\_edges(high)$ 函数会遍历当前网格中所有的边，如果其长度大于 high，那么我们就从这条边的中点对其进行划分操作。操作之后，和这条边相邻的两个三角形都会被一分为二。

$collapse\_short\_edges(low, high)$ 函数同样对当前网格的所有边进行遍历，对长度小于 low 的边进行融合操作。值得注意的是，在进行融合操作时，需检查经过该操作是否会产生长度大于 high 的边。如果不产生，那么我们进行此次的融合操作，否则不进行。

$equalize\_valences()$ 函数对边进行 Flip 操作来调整各个顶点的价（Valence），即与该顶点相邻的顶点的个数，使其尽量地接近目标价。函数会遍历当前网格的所有边，并尝试对其进行**翻转**操作，然后比较 Flip 操作前后和这条边相邻的两个三角形上的四个顶点其价和目标值的偏差，如果偏差没有变小那么撤销之前的翻转操作，即对这条边再进行一次翻转操作。

$target\_val(v)$ 函数接受一个顶点做为输入，如果该顶点是边界上的点那么函数返回 4，如果是内部的点则返回 6。

$tangential\_relaxation()$ 函数对当前的网格进行反复的平滑过滤。假定 **p** 是

网格上的某一个点，n 是该点的法线量，然后使用下面的方法计算出点 **q**：

$$\mathbf{q} = \frac{1}{|\mathcal{N}_1(\mathbf{p})|} \sum_{\mathbf{p}_j \in \mathcal{N}_1(\mathbf{p})} \mathbf{p}_j. \tag{7.7}$$

将 **q** 向 **p** 点的方向投影，得到 **p** 的新位置：

$$\mathbf{p}' = \mathbf{q} + \mathbf{n}\mathbf{n}^T(\mathbf{p} - \mathbf{q}). \tag{7.8}$$

$project\_to\_surface()$ 函数将顶点投影到原曲面上。
为了保证输入模型的特征，在进行上面的算法的时候需要加入一些限制：

- 角上的顶点对应的一个或两个特征边必需被保留，并且不对其进行任何可以改变其几何或者拓扑结构的操作；
- 对一个特征边进行划分操作将得到两个特征边和一个特征点；
- 对特征边进行 $tangential\_relaxation()$ 函数的时候，只能在这条边的方向上进行；
- 对处于特征边的点进行融合操作时，只能在沿着这条边的方向上进行该操作；
- 不能对特征边进行翻转操作。

7.6　四边形网格网格重划分

生成四边形网格的思路主要有下面几种：

- 四边形化：通过放置一些列的点，然后将其四边形化。不过缺点在于缺乏对边朝向性和对齐性的控制；
- 转化：首先生成三角形网格或者多边形网格，然后将其转换为四边形网格。转换的具体方法有：合并与一条边相邻的两个三角形，切分多边形网格等。这种方法能够简单地控制边的朝向性和对齐性；
- 基于曲线的采样：通过放置一系列与方向场相切的曲线来生成一个曲线网络，这样网络上的每一个交点正好是生成的顶点。此方法能够很好的控制边的朝向性和对齐性，不过生成的四边形网格上可能会有 T-Junction，所以并不完全是一个四边形网格；
- 等值线法（Contouring）：一个能得到纯四边形网格的方法简述如下。计算两个标量函数，然后将经过选取的一系列等值带入这两个函数中从而得到四边形网格的一些列小四边形。

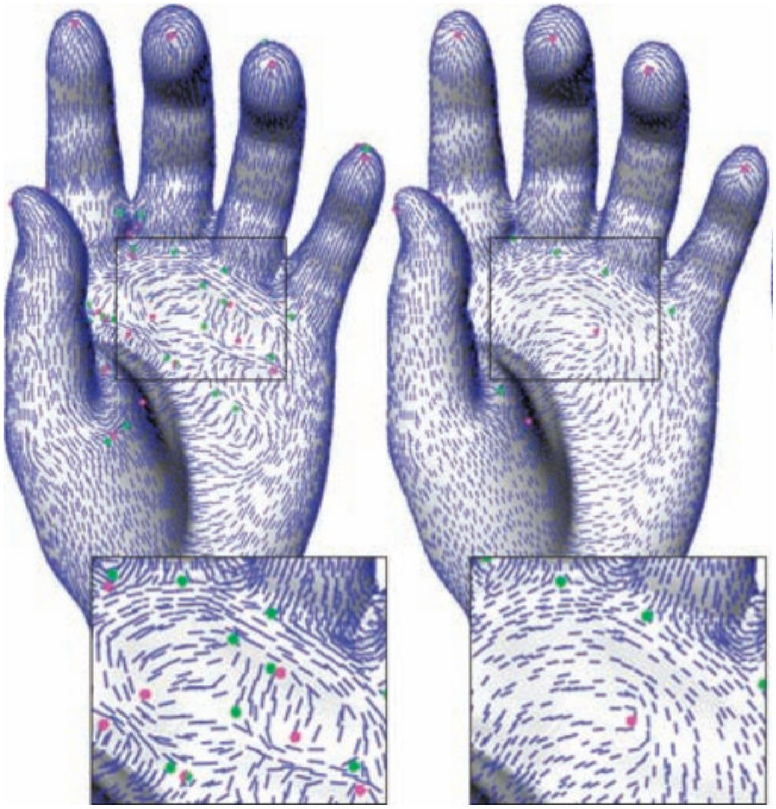


图 7.6　左图：初始主方向场；右图：平滑后的主方向场；图中彩色的点为脐点。

7.6.1　曲线

这里我们讨论后两种生成四边形网格的方法（基于曲线的采样和等值线法）。该方法包含三个主要步骤：

- 首先计算每个顶点的曲率张量来还原出一个连续的模型。计算完张量之后，丢弃掉其中法向量的分量。然后通过计算离散保角参数化得到了一个二维分段张量场，然后对其使用高斯核函数做卷积运算，得到一个平滑的曲率方向场，并且提取出张量场中的脐点。

$$\mathbf{C} = \mathbf{PDP}^{-1}. \tag{7.9}$$

- 第二步是在参数空间上上进行重新采样。对于各向异性的区域，建立由一些列曲率线（沿着主曲率的方法）组成的网格；而对于各向同性的区域则使用普通的顶点采样的方法。

- 最后一步通过第二部采样出来的点获得边，在各向异性的区域我们将曲率线拉直以得到边，在各向同性的区域我们通过德洛内三角剖分来获得边。最终得到的多边形网格中既包含三角形也包含了四边形。

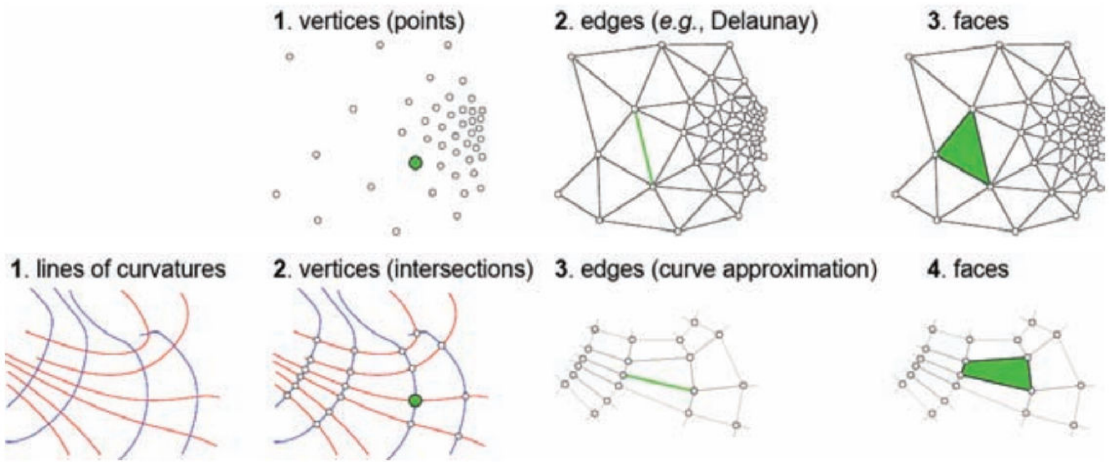


图 7.7　基于点的采样与基于曲率的采样的对比。

对于平坦的区域，由于曲率线之间的距离很大，可能不会产生任何的多边形。因此当曲率线进入一个较为平坦的区域的时候，算法从其上的一个采样点开始沿着测地线（Geodesic Curve）的方向延长，直到其进入较为弯曲的区域。