1 几何处理

1.1 离散微分几何

我们总是用离散的网格[1]来描述连续的曲面, 但网格定义出的几何形状只能给出 $\mathcal{C}^0$ 连续性. 如何将连续曲面的各种微分量应用在离散的情况下就属于**离散微分几何**研究的内容. 常见的各种微分量有法向量, 曲率, 拉普拉斯算子等.

1.1.1 局部平均区域

我们可以用下面的方法刻画三角网格顶点 $\boldsymbol{x}_i$ 处的微分量 $\boldsymbol{f}$:

$$\boldsymbol{f}(\boldsymbol{x}_i) = \left(\iint_{\Omega(\boldsymbol{x}_i)} \boldsymbol{f} \mathrm{d}S \right) \Big/ \left(\iint_{\Omega(\boldsymbol{x}_i)} \mathrm{d}S \right)$$

即在 $\boldsymbol{x}$ 的某个邻域 $\Omega(\boldsymbol{x})$ 的邻域上该微分量的平均值. 邻域的选择直接影响离散微分量的结果和准确度. 邻域越大, 结果越平滑, 但也会丢失更多的细节. 邻域越小, 结果越接近真实值, 但也对噪声更敏感. 常见的邻域选择有以下几种:

1. **重心单元 (barycentric cell)**: 把与顶点 $\boldsymbol{x}$ 相邻的所有三角形的两边的中点和重心连起来形成的多边形区域.
2. **泰森多边形单元 (Voronoi cell)**: 把与顶点 $\boldsymbol{x}$ 相邻的所有三角形的外心连起来形成的多边形区域.
3. **混合泰森多边形单元 (mixed Voronoi cell)**: Vorinoi cell 在钝角三角形中取的点可能超出三角形外, 因此在钝角三角形中用边的中点代替外心, 其余情形不变.

1.1.2 法向量

三角形内部点的法向量可以唯一地定义为三角形所在平面的法向量. 对于顶点 $\boldsymbol{x}$ 处的法向量 $\boldsymbol{n}(\boldsymbol{x})$, 可以通过对顶点周围三角形的法向量加权平均得到:

$$\boldsymbol{n}(\boldsymbol{x}) = \frac{\sum\limits_{T \in \Omega(\boldsymbol{x})} \alpha_T \boldsymbol{n}(T)}{\sum\limits_{T \in \Omega(\boldsymbol{x})} \alpha_T \left|\left| \boldsymbol{n}(T) \right|\right|}$$

其中 $\Omega(\boldsymbol{x})$ 是与顶点 $\boldsymbol{x}$ 邻接的三角形构成的几何,α_T 和 $\boldsymbol{n}(T)$ 是三角形 T 的权重和法向量. 常见的权重有以下几种:

1. **常数权重**: $\alpha_T = 1$. 这在计算上最简单, 但对于一些不规则网格效果不好.
2. **面积权重**: $\alpha_T = A_T$. 以三角形面积 A_T 加权, 比常数权重更合理, 但对过于不规则的网格效果也不好.
3. **角度权重**: $\alpha_T = \theta_T$, 其中 θ_T 是顶点 $\boldsymbol{x}$ 在三角形 T 中的对角. 这样加权效果最好, 但计算上也最复杂.

[1]在本节, 我们主要以三角网格作为研究对象.

1.1.3 梯度

梯度是非常重要的微分量, 在计算拉普拉斯算子, 网格参数化等方面有重要应用.

对于一个在三角网格上定义的标量函数 f, 我们可以通过 f 在三角形顶点上的值通过重心插值的方式得到三角形内部的值. 不妨设三角形的顶点为 $\boldsymbol{x}_a, \boldsymbol{x}_b, \boldsymbol{x}_c$, 则其内部任意一点 $\boldsymbol{x}$ 的值为:

$$f(\boldsymbol{x}) = \alpha(\boldsymbol{x})f(\boldsymbol{x}_a) + \beta(\boldsymbol{x})f(\boldsymbol{x}_b) + \gamma(\boldsymbol{x})f(\boldsymbol{x}_c)$$

其中 α, β, γ 是点 $\boldsymbol{x}$ 在三角形中的重心坐标, 在 **Lecture 3 绘图**一节中已经讲过其定义方式. 我们在那时已经证明重心坐标是线性的, 因此 f 的梯度 ∇f 也是定值. 现在来证明之.

证明. 不妨假定 $\boldsymbol{x}_a, \boldsymbol{x}_b$ 和 $\boldsymbol{x}_c$ 在三角形 T 中对应的顶点分别为 A, B 和 C, $\boldsymbol{x}$ 对应 T 内的一点 P. 以 $\alpha(\boldsymbol{x})$ 为例, 根据重心坐标与三角形面积的关系, 有

$$\alpha(\boldsymbol{x}) = \frac{A_{\triangle PBC}}{A_{\triangle ABC}} = \frac{\overline{BP} \cdot \overline{BC} \cdot \sin\angle PBC}{2A_T} = \frac{\left|\left|\overrightarrow{BP} \times \overrightarrow{BC}\right|\right|}{2A_T} = \frac{(\boldsymbol{x} - \boldsymbol{x}_b) \cdot (\boldsymbol{x}_c - \boldsymbol{x}_b)^\perp}{2A_T}$$

其中 $(\boldsymbol{x}_c - \boldsymbol{x}_b)^\perp$ 表示该向量在平面上逆时针旋转 $90°$ 后的向量. 于是

$$\nabla\alpha(\boldsymbol{x}) = \frac{(\boldsymbol{x}_c - \boldsymbol{x}_b)^\perp}{2A_T}$$

于是

$$\nabla f(\boldsymbol{x}) = \frac{(\boldsymbol{x}_c - \boldsymbol{x}_b)^\perp}{2A_T} f(\boldsymbol{x}_a) + \frac{(\boldsymbol{x}_a - \boldsymbol{x}_c)^\perp}{2A_T} f(\boldsymbol{x}_b) + \frac{(\boldsymbol{x}_b - \boldsymbol{x}_a)^\perp}{2A_T} f(\boldsymbol{x}_c)$$

这表明 $\nabla f(\boldsymbol{x})$ 与 $\boldsymbol{x}$ 无关, 即在三角形内部为常值. □

1.1.4 离散拉普拉斯算子

定义 1.1 拉普拉斯算子 **拉普拉斯算子 (Laplace Operator, Laplacian)** 是一个二阶微分算子, 记号为 Δ 或 ∇^2, 定义为函数梯度的散度:

$$\Delta f = \operatorname{div}(\operatorname{grad} f)$$

在 n 维欧氏空间中, 对于一个标量函数 $f(x_1, \cdots, x_n)$, 拉普拉斯算子可以表示为:

$$\Delta = \sum_{i=1}^{n} \frac{\partial^2}{\partial x_i^2}$$

我们知道, 对于一维函数, 其二阶导数表示该函数的凹凸程度; 而 Laplacian 是多维函数在各个正交的方向上的二阶导数之和, 也就刻画了该函数在某一点整体的凹凸程度.

我们也可以用梯度和散度的几何意义理解 Laplacian. 梯度作为向量场衡量了函数在各点的变化方向, 再对其求散度就可以得到函数在该点附近梯度的情况. 如果梯度场流入某一点更多, 此时该点的 Laplacian 为正, 表明此处函数内凹 (类似于山谷); 反之, 如果梯度场流出某一点更多, 此时该点的 Laplacian 为负, 表明此处函

数外凸 (类似于山峰). 当梯度场流入和流出的量近似时, 该点的 Laplacian 接近零, 表明此处函数比较平坦[2](类似于山坡).

Laplacian 对于刻画曲面性质有重要作用. 例如, 我们可以用 Laplacian 计算由参数方程 $\boldsymbol{S}(u,v)$ 在三维空间中定义的曲面 S 的平均曲率[3]:

$$H = -\frac{\Delta \boldsymbol{S}}{2\boldsymbol{n}}$$

在离散情况下, 我们可以得到如下结论.

定义 1.2 离散拉普拉斯算子 在三角形网格中, 函数 f 在顶点 $\boldsymbol{x}_i$ 处的**离散拉普拉斯算子 (Discrete Laplace Operator)** 定义为:

$$\Delta f(\boldsymbol{x}_i) = \sum_{\boldsymbol{x}_j \in \Omega(\boldsymbol{x}_i)} w_{ij} \left(f(\boldsymbol{x}_j) - f(\boldsymbol{x}_i) \right)$$

其中 $\Omega(\boldsymbol{x}_i)$ 是与顶点 $\boldsymbol{x}_i$ 邻接的顶点集合, w_{ij} 是权重.

定义 1.3 均匀拉普拉斯算子 **均匀拉普拉斯算子 (Uniform Laplace Operator)** 的权重定义为:

$$w_{ij} = \frac{1}{|\Omega(\boldsymbol{x}_i)|}$$

其中 $|\Omega(\boldsymbol{x}_i)|$ 是邻接顶点的个数.

定义 1.4 余切拉普拉斯算子 **余切拉普拉斯算子 (Cotangent Laplace Operator)** 的权重定义为:

$$w_{ij} = \frac{1}{2} \left(\cot \alpha_{ij} + \cot \beta_{ij} \right)$$

其中 α_{ij} 和 β_{ij} 是与边 $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ 相邻的两个三角形中该边所对的角.

我们现在来证明上述定义 (尤其是余切 Laplacian) 的合理性[4].

证明. 我们首先推导散度定理在二维情形下 (即格林公式) 的向量表述[5]. 对于一个在区域 Ω 上定义的向量场 $\boldsymbol{v}(x,y) = (P(x,y), Q(x,y))$, 设 $\boldsymbol{n}(x,y) = (a,b)$ 为 $\partial\Omega$ 上 (x,y) 处的单位外法向量, 此处的单位切向量 $\boldsymbol{t}(x,y) = (\cos\alpha, \cos\beta)$. 于是有

$$\boldsymbol{k} = \boldsymbol{n} \times \boldsymbol{t} = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ a & b & 0 \\ \cos\alpha & \cos\beta & 0 \end{vmatrix} = (a\cos\beta - b\cos\alpha)\,\boldsymbol{k}$$

[2]这里的平坦不是指函数值变化不大, 而是指函数值的变化速率变化不大, 类似于一次函数.

[3]对于一个曲面 S 上某一点 P, 过 P 的法向量做平面可以与 S 交出一条曲线 C. 改变平面的方向所得的 C 中在 P 处的曲率有极大值 k_1 和极小值 k_2, 称作曲面 S 在点 P 处的**主曲率** (Principal Curvature). 曲面 S 在 P 处的**平均曲率** (Mean Curvature) 定义为 $H = (k_1 + k_2)/2$. 以后也许还会用到**高斯曲率** (Gaussian Curvature), 定义为 $K = k_1 k_2$.

[4]依照下面的推导, 余切 Laplacian 的系数的归一化与邻域面积有关, 但实际实现时往往会直接进行归一化操作; 并且在计算得到的余切值显著较大时还需要进行修正.

[5]这在高等数学习题中亦有相应的习题, 此处再推导一遍.

其中 $\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}$ 分别为 x,y,z 轴的单位向量. 解上述方程从而表明 $\boldsymbol{n}=(\cos\beta,-\cos\alpha)$. 于是就有

$$\oint_{(\partial\Omega)^+}\boldsymbol{v}\cdot\boldsymbol{n}\mathrm{d}s=\oint_{(\partial\Omega)^+}(P\cos\beta-Q\cos\alpha)\mathrm{d}s=\oint_{(\partial\Omega)^+}(P\mathrm{d}y-Q\mathrm{d}x)$$
$$\xlongequal{\text{格林公式}}\iint_\Omega\left(\frac{\partial P}{\partial x}+\frac{\partial Q}{\partial y}\right)\mathrm{d}S=\iint_\Omega\operatorname{div}\ \boldsymbol{v}\mathrm{d}S=\iint_\Omega(\nabla\cdot\boldsymbol{v})\mathrm{d}S$$

现在回到对于网格的讨论来. 对于顶点 $\boldsymbol{x}_i$ 的一个邻域 $\Omega(\boldsymbol{x}_i)$, 根据散度定理有

$$\iint_{\Omega(\boldsymbol{x}_i)}\Delta f\mathrm{d}S=\iint_{\Omega(\boldsymbol{x}_i)}\nabla(\nabla f)\mathrm{d}S=\int_{\partial\Omega(\boldsymbol{x}_i)}\nabla f\cdot\boldsymbol{n}\mathrm{d}s$$

其中 $\boldsymbol{n}$ 表示给定点处的法向量. 我们分别考虑边界[6]$\partial\Omega(\boldsymbol{x}_i)$ 在 $\boldsymbol{x}_i$ 邻接的各个三角形 T_m 中的一段 $\partial\Omega(\boldsymbol{x}_i)\cap T_m$, 设 T_m 的顶点为 $\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k$.

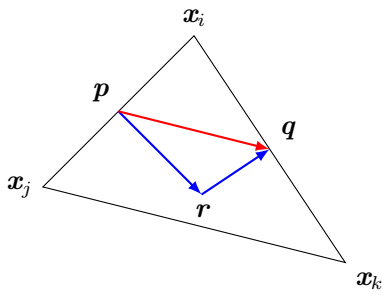

图 1: $\partial\Omega(\boldsymbol{x}_i)$ 在 T_m 中的一段 (图中的蓝色有向线段)

在梯度一节中我们已经证明 ∇f 在给定三角形上是定值, 于是依照上图有

$$\int_{\partial\Omega(\boldsymbol{x}_i)\cap T_m}\nabla f\cdot\boldsymbol{n}\mathrm{d}s=\nabla f\cdot\int_{\partial\Omega(\boldsymbol{x}_i)}\boldsymbol{n}\mathrm{d}s=\nabla f\cdot\left(\int_{PR}\boldsymbol{n}\mathrm{d}s+\int_{RQ}\boldsymbol{n}\mathrm{d}s\right)$$

而对于有向线段 PR 而言, 其方向向量为 $\boldsymbol{t}=\dfrac{\boldsymbol{r}-\boldsymbol{p}}{||\boldsymbol{r}-\boldsymbol{p}||}$, 于是

$$\int_{PR}\boldsymbol{n}\mathrm{d}s=\int_{PR}\left(-\boldsymbol{t}^\perp\right)\mathrm{d}s=\frac{(\boldsymbol{p}-\boldsymbol{r})^\perp}{||\boldsymbol{r}-\boldsymbol{p}||}\cdot\int_{PR}\mathrm{d}s=(\boldsymbol{p}-\boldsymbol{r})^\perp$$

同理有 $\displaystyle\int_{RQ}\boldsymbol{n}\mathrm{d}s=(\boldsymbol{r}-\boldsymbol{q})^\perp$. 于是

$$\int_{\partial\Omega(\boldsymbol{x}_i)\cap T_m}\nabla f\cdot\boldsymbol{n}\mathrm{d}s=\nabla f\cdot\left((\boldsymbol{p}-\boldsymbol{r})^\perp+(\boldsymbol{r}-\boldsymbol{q})^\perp\right)=\nabla f\cdot(\boldsymbol{p}-\boldsymbol{q})^\perp$$

按照**局部平均区域**一节中所述的 $\Omega(\boldsymbol{x}_i)$ 的取法,$\boldsymbol{p}$ 和 $\boldsymbol{q}$ 分别为各自边的中点. 于是

$$\int_{\partial\Omega(\boldsymbol{x}_i)\cap T_m}\nabla f\cdot\boldsymbol{n}\mathrm{d}s=\frac{1}{2}\nabla f\cdot(\boldsymbol{x}_j-\boldsymbol{x}_k)^\perp$$

现在再将[7]上一节中推导的梯度公式变形可得

$$\nabla f=(f_j-f_i)\frac{(\boldsymbol{x}_i-\boldsymbol{x}_k)^\perp}{2A_m}+(f_k-f_i)\frac{(\boldsymbol{x}_j-\boldsymbol{x}_i)^\perp}{2A_m}$$

[6]这里默认取正向边界方向. 后文的顶点选取的顺序也依照正向边界的方向.

[7]为了简洁考虑, 记 f 在 $\boldsymbol{x}_i$ 处的值为 f_i, 其余类似

代入上式可得

$$\begin{aligned}\int_{\partial\Omega(\boldsymbol{x}_i)\cap T_m}\nabla f\cdot\boldsymbol{n}\mathrm{d}s &= \frac{1}{4A_m}\left[(f_j-f_i)\left(\boldsymbol{x}_i-\boldsymbol{x}_k\right)^{\perp}\left(\boldsymbol{x}_j-\boldsymbol{x}_k\right)^{\perp}+(f_k-f_i)\left(\boldsymbol{x}_j-\boldsymbol{x}_i\right)^{\perp}\left(\boldsymbol{x}_j-\boldsymbol{x}_k\right)^{\perp}\right]\\ &= \frac{1}{4A_m}\left[(f_j-f_i)\left(\boldsymbol{x}_i-\boldsymbol{x}_k\right)\cdot\left(\boldsymbol{x}_j-\boldsymbol{x}_k\right)+(f_k-f_i)\left(\boldsymbol{x}_i-\boldsymbol{x}_j\right)\cdot\left(\boldsymbol{x}_k-\boldsymbol{x}_j\right)\right]\end{aligned}$$

上面的式子中恰好存在边的点乘与面积之比, 因此考虑用三角函数表示. 事实上, 对于任意的 $\triangle ABC$ 总有

$$\frac{\overrightarrow{AB}\cdot\overrightarrow{AC}}{A_{\triangle ABC}}=\frac{\overline{AB}\cdot\overline{AC}\cdot\cos A}{\frac{1}{2}\overline{AB}\cdot\overline{AC}\cdot\sin A}=2\cot A$$

于是把这一结论运用到上式即可得

$$\int_{\partial\Omega(\boldsymbol{x}_i)\cap T_m}\nabla f\cdot\boldsymbol{n}\mathrm{d}s=\frac{1}{2}\left[(f_j-f_i)\cot\gamma_k+(f_k-f_i)\cot\gamma_j\right]$$

其中 γ_j 和 γ_k 分别是顶点 $\boldsymbol{x}_j$ 和 $\boldsymbol{x}_k$ 处的角. 现在, 我们将 $\partial\Omega(\boldsymbol{x}_i)$ 在所有邻接三角形中的部分累加起来, 即可得

$$\int_{\partial\Omega(\boldsymbol{x}_i)}\nabla f\cdot\boldsymbol{n}\mathrm{d}s=\frac{1}{2}\sum_{j\in\Omega(i)}\left(\cot\alpha_{ij}+\cot\beta_{ij}\right)\left(f_j-f_i\right)$$

其中 α_{ij} 和 β_{ij} 是与边 $(\boldsymbol{x}_i,\boldsymbol{x}_j)$ 相邻的两个三角形中该边所对的角. 最后, 我们只需要将上式除以邻域 $\Omega(\boldsymbol{x}_i)$ 的面积 $A(\boldsymbol{x}_i)$ 即可得到余切 Laplacian 的定义, 即

$$\Delta f\left(x_i\right)=\frac{1}{2A(\boldsymbol{x}_i)}\sum_{j\in\Omega(i)}\left(\cot\alpha_{ij}+\cot\beta_{ij}\right)\left(f_j-f_i\right)$$

在实际应用中, 由于 $A(\boldsymbol{x}_i)$ 是定值, 因此我们通常令权重 w_{ij} 为

$$w_{ij}=\frac{1}{2}\left(\cot\alpha_{ij}+\cot\beta_{ij}\right)$$

后进行归一化处理.

对权值进行简化, 我们可以得到均匀 Laplacian 的定义. 均匀 Laplacian 的计算量较小, 但没有恰当的几何意义, 在很多时候也经常导致问题. □

1.2 网格平滑

随着三维扫描和曲面重建技术的发展, 得到实体表面的多边形网格表示已经不是难事, 但所得到的表面往往包含噪声. 为了将其转化为更光滑的网格以便后续处理, 我们需要用到**网格平滑**技术.

定义 1.5 网格平滑 **网格平滑 (Mesh Smoothing)**(又称为**网格降噪 (Mesh Deinoising)** 通过) 对多边形网格进行处理, 以减少噪声, 从而得到更光滑, 性质更优良的网格表示.

一般而言, 噪声所在的位置都是曲率较大, 形状比较尖锐的地方. 因此, 网格平滑的目标是减少这些高频噪声, 同时尽可能地保留网格的整体形状和细节.

我们在本节主要介绍**拉普拉斯平滑 (Laplacian Smoothing)**.

我们用扩散流的数学模型来描述网格平滑的过程. 考虑热量传导的物理模型, 设 $f(\boldsymbol{x},t)$ 表示时间 t 时刻位置 $\boldsymbol{x}$ 处的温度, 则扩散方程为:

$$\frac{\partial f(\boldsymbol{x},t)}{\partial t}=\lambda\Delta f(\boldsymbol{x},t)$$

在网格中, 高频的噪声就对应于杂乱分布的高温点. 随着扩散的进行, 热量分布趋于均匀, 对应顶点构成的表面也趋于平滑.

现在, 我们需要将扩散流公式在时间和空间上进行离散化, 以便在计算机中处理. 在空间上的 Laplacian 离散化已经在前面一节中介绍, 即

$$\frac{\partial f\left(\boldsymbol{x}_i,t\right)}{\partial t}=\lambda\Delta f\left(\boldsymbol{x}_i,t\right)=\lambda\sum_{\boldsymbol{x}_j\in\Omega(\boldsymbol{x}_i)}w_{ij}\left(f\left(\boldsymbol{x}_j,t\right)-f\left(\boldsymbol{x}_i,t\right)\right)$$

而在时间上的离散化, 可以将 t 划分为步长为 h 的若干个间隔, 用有限差分的思想对微分方程进行近似:

$$\frac{\partial f\left(\boldsymbol{x}_i,t\right)}{\partial t}=\frac{f\left(\boldsymbol{x}_i,t+h\right)-f\left(\boldsymbol{x}_i,t\right)}{h}$$

于是

$$f\left(\boldsymbol{x}_i,t+h\right)=f\left(\boldsymbol{x}_i,t\right)+h\frac{\partial f\left(\boldsymbol{x}_i,t\right)}{\partial t}=f\left(\boldsymbol{x}_i,t\right)+h\lambda\sum_{\boldsymbol{x}_j\in\Omega(\boldsymbol{x}_i)}w_{ij}\left(f\left(\boldsymbol{x}_j,t\right)-f\left(\boldsymbol{x}_i,t\right)\right)$$

在网格简化中, 我们考虑的 "温度" 就是顶点的空间坐标. 于是, 在每次时间间隔为 h 的第 k 次迭代中, 每个顶点 $\boldsymbol{x}_i$ 的位置按照如下的方式更新:

$$\boldsymbol{x}_i^{(k+1)}\leftarrow\boldsymbol{x}_i^{(k)}+\lambda\Delta\boldsymbol{x}_i^{(k)}$$

其中 λ 为**平滑参数**即为前面推导中的 $h\lambda$.

在两种 Laplacian 下, 上述迭代方法可以得到不同的平滑效果. 均匀 Laplacian 下, 每个顶点都向其邻居的均值移动, 这种方法简单且计算效率高, 但容易导致网格收缩和细节丢失. 余切 Laplacian 下, 每个顶点向平均曲率的方向移动, 也被称作**平均曲率流 (Mean Curvature Flow)**. 这种方法能够更好地保留网格的形状和细节, 但是计算复杂度较高.

1.3 网格编辑

定义 1.6 网格编辑 **网格编辑 (Mesh Editing)** 是操纵和修改网格表面的几何形状, 同时能够保留原始网格几何细节的操作.

回顾我们对二维图像进行 Poisson 编辑的过程, 这可以描述为下面的最小化问题:

$$\arg\min_f\iint_\Omega\left|\left|\nabla f-\nabla g\right|\right|^2\mathrm{d}S,\quad\text{s.t.}\quad f|_{\partial\Omega}=f^*|_{\partial\Omega}$$

类似的思想在三维情形下的应用就是 Laplacian 网格编辑. 它通过尽可能使得编辑后网格上的 Laplace 微分坐标与原始网格上的 Laplace 微分坐标保持一致保留网格的细节. Laplace 微分坐标在本质上就是对网格顶点施加均匀 Laplacian 的结果, 即

$$\mathcal{L}(\boldsymbol{v}_i) = \boldsymbol{v}_i - \frac{1}{N_i}\sum_{j\in\Omega(i)}\boldsymbol{v}_j$$

1.4 网格简化

在图形学中, 渲染效率与渲染质量需要兼顾. 同样的物体在近处时需要使用精度较高, 面数较多的模型以保证真实; 而在远处时由于物体在屏幕上占据的像素较少, 使用精度较低, 面数较少的模型也能保证视觉效果. 因此, 我们需要对高精度的网格进行简化以提高渲染效率. 这时就需要用到**网格简化**算法来生成不同**细节层次 (Level of Detail, LoD)** 的模型用于不同情况下的渲染.

定义 1.7 网格简化 (Mesh Simplification) 通过对多边形网格进行处理, 以减少其面数, 从而得到更简洁, 渲染效率更高的网格表示.

网格简化一般可以通过移除顶点或坍缩边实现, 实际应用中后者更容易实现, 也更常用. 边坍缩算法的核心是设计方法找出哪些边的坍缩对网格形状的影响最小, 以及确定坍缩后新顶点的位置, 从而在简化时尽量保留网格的整体形状.

我们在本节主要介绍**二次误差度量 (Quadric Error Metrics, QEM)** 方法[8]. 对于一次边坍缩, 考虑将 $\boldsymbol{v}_1, \boldsymbol{v}_2$ 收缩为 $\boldsymbol{v}$. 为了度量收缩操作与原网格的差异, 我们可以用简化后的点与原网格中 $\boldsymbol{v}_1, \boldsymbol{v}_2$ 邻接的所有面的距离之和来表示.

定义 $\text{plain}(\boldsymbol{v}_i)$ 表示所有与顶点 $\boldsymbol{v}_i$ 邻接的面的集合, 其中的元素 $\boldsymbol{p} = \begin{bmatrix} a & b & c & d \end{bmatrix}^{\mathrm{t}}$ 表示对应平面的方程为 $ax + by + cz + d = 0$, 并且 $a^2 + b^2 + c^2 = 1$. 于是优化目标为:

$$\boldsymbol{v}^* = \arg\min_{\boldsymbol{v}} \sum_{\boldsymbol{p}\in\text{plain}(\boldsymbol{v}_1)\cup\text{plain}(\boldsymbol{v}_2)} d_{\boldsymbol{v},\boldsymbol{p}}^2$$

在这里, 为了计算方便, 我们将 $\boldsymbol{v}$ 扩展为具有四个分量的向量 $\boldsymbol{v} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^{\mathrm{t}}$, 其中前三个分量表示 $\boldsymbol{v}$ 的位置. 于是点 $\boldsymbol{v}$ 到平面 $\boldsymbol{p}$ 的距离为:

$$d_{\boldsymbol{v},\boldsymbol{p}}^2 = \left(\boldsymbol{v}^{\mathrm{t}}\boldsymbol{p}\right)^2 = \boldsymbol{v}^{\mathrm{t}}\boldsymbol{p}\boldsymbol{p}^{\mathrm{t}}\boldsymbol{v}$$

令 $\boldsymbol{K}_{\boldsymbol{p}} = \boldsymbol{p}\boldsymbol{p}^{\mathrm{t}}$, 则有

$$\boldsymbol{v}^* = \arg\min_{\boldsymbol{v}} \boldsymbol{v}^{\mathrm{t}} \left(\sum_{\boldsymbol{p}\in\text{plain}(\boldsymbol{v}_1)\cup\text{plain}(\boldsymbol{v}_2)} \boldsymbol{K}_{\boldsymbol{p}} \right) \boldsymbol{v}$$

为了简化计算, 我们将上式近似为

$$\boldsymbol{v}^* = \arg\min_{\boldsymbol{v}} \boldsymbol{v}^{\mathrm{t}} \left(\sum_{\boldsymbol{p}\in\text{plain}(\boldsymbol{v}_1)} \boldsymbol{K}_{\boldsymbol{p}} + \sum_{\boldsymbol{p}\in\text{plain}(\boldsymbol{v}_2)} \boldsymbol{K}_{\boldsymbol{p}} \right) \boldsymbol{v}$$

[8] Garland M, Heckbert PS. Surface simplification using quadric error metrics. In: Proceedings of the 24th annual conference on computer graphics and interactive techniques. 1997, p. 209-16.

实际上至多只会重复以 $\boldsymbol{v}_1, \boldsymbol{v}_2$ 为顶点的两个三角形面, 但将误差转化为仅与顶点相关的信息, 因而简化了计算. 令 $\boldsymbol{Q}_i = \sum_{\boldsymbol{p}\in\mathrm{plain}(\boldsymbol{v}_i)} \boldsymbol{K}_{\boldsymbol{p}}$, 于是就有

$$\boldsymbol{v}^* = \arg\min_{\boldsymbol{v}} \boldsymbol{v}^{\mathrm{t}} \left(\boldsymbol{Q}_1 + \boldsymbol{Q}_2\right) \boldsymbol{v}$$

现在来考虑这一二次型的最小值何时取到. 令 $\boldsymbol{Q} = \boldsymbol{Q}_1 + \boldsymbol{Q}_2$, 将上述式子展开可得

$$\boldsymbol{v}^{\mathrm{t}} \boldsymbol{Q} \boldsymbol{v} = q_{11}x^2 + 2q_{12}xy + 2q_{13}xz + 2q_{14}x + q_{22}y^2 + 2q_{23}yz + 2q_{24}y + q_{33}z^2 + 2q_{34}z + q_{44}$$

并且要求

$$\frac{\partial \boldsymbol{v}^{\mathrm{t}} \boldsymbol{Q} \boldsymbol{v}}{\partial x} = \frac{\partial \boldsymbol{v}^{\mathrm{t}} \boldsymbol{Q} \boldsymbol{v}}{\partial y} = \frac{\partial \boldsymbol{v}^{\mathrm{t}} \boldsymbol{Q} \boldsymbol{v}}{\partial z} = 0$$

于是可以得到如下线性方程组:

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

令 $\boldsymbol{Q}'$ 为前面的矩阵, 则有 $\boldsymbol{Q}'\boldsymbol{v}^* = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^{\mathrm{t}}$. 如果 $\boldsymbol{Q}'$ 可逆, 就有

$$\boldsymbol{v}^* = \left(\boldsymbol{Q}'\right)^{-1} \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^{\mathrm{t}}$$

如果 $\boldsymbol{Q}'$ 不可逆, 这就说明我们无法找出使误差最小的 $\boldsymbol{v}$. 此时, 我们可以简单地选择 $\boldsymbol{v}_1, \boldsymbol{v}_2$ 以及它们的中点 $\frac{\boldsymbol{v}_1 + \boldsymbol{v}_2}{2}$ 中使得误差 $\boldsymbol{v}^{\mathrm{t}} \boldsymbol{Q} \boldsymbol{v}$ 最小的点作为优化的结果 $\boldsymbol{v}^*$. 需要注意的是, 这里的 $\boldsymbol{Q}'$ 仅用于求解 $\boldsymbol{v}^*$ 使用. 计算误差仍然使用 $\boldsymbol{Q}$.

于是, QEM 算法的基本流程为:

1. 初始化每个顶点的误差矩阵 $\boldsymbol{Q}_i$.
2. 对每组合法的 (可收缩而不改变网格拓扑结构的) 点对 $(\boldsymbol{v}_1, \boldsymbol{v}_2)$, 计算其误差矩阵 $\boldsymbol{Q} = \boldsymbol{Q}_1 + \boldsymbol{Q}_2$ 以及最优收缩点 $\boldsymbol{v}^*$ 和误差 $\mathrm{err}(\boldsymbol{v}^*) = \left(\boldsymbol{v}^*\right)^{\mathrm{t}} \boldsymbol{Q} \boldsymbol{v}^*$.
3. 将所有点对按误差从小到大存入优先队列中.
4. 重复以下操作直到达到预定的顶点数:
   a. 从优先队列中取出误差最小的点对 $(\boldsymbol{v}_1, \boldsymbol{v}_2)$, 并将其坍缩为 $\boldsymbol{v}^*$, 在网格结构中实现这一操作.
   b. 考虑所有坍缩前与 $\boldsymbol{v}_1, \boldsymbol{v}_2$ 相邻的顶点 $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n$. 首先, 由于 $\boldsymbol{u}_i$ 相邻的点已经更新为 $\boldsymbol{v}^*$, 因此其相邻的面已经发生改变, 因此需要重新计算 $\boldsymbol{u}_i$ 的误差矩阵. 然后考虑所有与 $\boldsymbol{u}_i$ 相邻的顶点 $\boldsymbol{w}_{i1}, \cdots, \boldsymbol{w}_{im}$, 对于顶点对 $(\boldsymbol{u}_i, \boldsymbol{w}_{ij})$, 由于 $\boldsymbol{u}_i$ 的误差矩阵发生变化, 因此需要重新判断点对是否合法, 并重新计算收缩点和误差, 最后将其更新到优先队列中.