中国科学技术大学

硕士学位论文

基于边长的三维曲面插值

作者姓名： 刘振晔

学科专业： 计算数学

导师姓名： 刘利刚教授 陈仁杰教授

完成时间： 二〇二二年二月二十八日

University of Science and Technology of China

A dissertation for master's degree

3D Mesh Interpolation Based on Edge Length

Author: Zhenye Liu

Speciality: Computational Mathematics

Supervisors: Prof. Ligang Liu, Prof. Renjie Chen

Finished time: February 28, 2022

中国科学技术大学学位论文原创性声明

本人声明所呈交的学位论文，是本人在导师指导下进行研究工作所取得的成果。除已特别加以标注和致谢的地方外，论文中不包含任何他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的贡献均已在论文中作了明确的说明。

作者签名：_____ 签字日期：_____

中国科学技术大学学位论文授权使用声明

作为申请学位的条件之一，学位论文著作权拥有者授权中国科学技术大学拥有学位论文的部分使用权，即：学校有权按有关规定向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅，可以将学位论文编入《中国学位论文全文数据库》等有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。本人提交的电子文档的内容和纸质论文的内容相一致。

控阅的学位论文在解密后也遵守此规定。

☑ 公开 ☐ 控阅（____ 年）

作者签名：_____ 导师签名：_____

签字日期：_____ 签字日期：_____

摘　要

形状插值问题是一个非常重要的计算机图形学问题，被广泛应用于计算机动画等领域，已有几十年的研究历史。而在插值过程中，我们往往会对边长有要求，希望其能在源网格和目标网格中平滑过渡。但是，现有的形状插值算法很多都对边长并无直接要求。即使在对边长有直接要求的算法中，有的方法则还结合了其它量，例如二面角等来得到插值序列；有的方法则计算速度慢且是间接重建的。并且这些方法最后得到的结果的边长往往离我们要求的边长仍有较大距离。因此，需要一种边长误差小且计算速度快的形状插值方法。

我们提出了一种基于边长的在两个具有相同连接关系的三维曲面间插值的方法。该方法相比现有方法，以顶点坐标为变量，直接对边长进行优化。为了快速收敛，该方法使用牛顿法对能量进行优化。由于牛顿法要求二阶海森矩阵为正定矩阵，因此我们还将其近似为正定矩阵。为了进一步减小运算量，加快速度，我们通过多种技巧使之能快速二阶海森矩阵近似为正交矩阵，避免了通常需要的特征值分解这一过程。因此该方法速度快，且相比于已有的其它基于边长的插值方法，具有更小的误差。

关键词：计算机图形学；形状插值

ABSTRACT

Key Words:

目　录

第 1 章　绪　　论

1.1　引言

形状插值问题是一个很经典的图形学问题。假设存在两个输入的形状 S_0 和 S_1，这两个网格存在相同的拓扑，即存在一个双射 $f(x): S_0 \to S_0$。形状插值问题则是要找到一函数 Ψ 使得对于任意时间 $t \in [0,1]$，存在

$$S^t = \Psi(S^0, S^1, t)$$

其中 S^t 是 t 时刻的形状。对于一个形状插值算法的好坏，我们可以从以下几个方面进行判断：

1. 插值。算法生成的形状序列应经过首尾这两个形状，即 $S_0 = \Psi(S^0, S^1, 0)$ 且 $S_1 = \Psi(S^0, S^1, 1)$。

2. 对称性。若将源形状和目标形状交换，则算法生成的形状序列应该与原本相反，即 $\Psi(S^0, S^1, t) = \Psi(S^1, S^0, 1-t)$

3. 平滑。即算法生成的形状的顶点路径应该是光滑的，且算法生成的形状序列整体上也应是光滑的.

4. 有界扭曲。其是为了度量面曲面中的三角形或体网格中的四面体的形变程度。有界扭曲即使指扭曲程度在形变过程中在一个范围内。

5. 自然。即算法生成的形状序列对于人眼来说应该是自然且符合直觉的。

6. 无翻转。算法生成的形状序列应该是无翻转的，即是一个局部单射。

7. 大变形 (large deformation)。大变形插值是指 S_0 要经过较大的变形才能变形成 S_1。因此算法应该生成合理的形状序列以从 S_0 过渡到 S_1。

一个形状插值算法对于上述几点满足的越多，则说明该算法越好。

形状插值可应用于很多方面，广泛应用于动画产业[1] [2] [3]。在文化强国的大背景下，加上中国经济的快速发展，文化产业也在飞速发展。作为文化产业的重要部分，中国的动画产业也在飞速发展。2014 年，中国动画产业产值大约为 1000 亿元，而 2020 年则已经超过了 2000 亿元，增长十分迅速。并且制作动画的软件，例如 Maya，3D Studio Max 等，均包含形状插值算法。因此，形状插值算法非常重要。动画制作者一般首先会绘制多处关键帧，然后使用合适的形状插值算法在各关键帧之间生成插值序列，最终得到了一连续动画。

而形状插值可以基于许多标准，比如角度、变换矩阵、边长等[4]。下一章将会较为详细的介绍各种形状插值方法。

对于主要基于边长的插值方法，例如[5-6]，前者针对面网格，后者则针对体网格。它们最后得到的结果距离期望边长依然有较大距离，不过后者得到的结果

更好。但是后者之所以在边长上难以更进一步，是因为后者的处理对象是体网格，若单纯插值边长，其内部往往如图2.7所示，无法正好粘合。因此，若想使边长进一步优化，内部的边会阻碍这一目标。但是面网格则不会有这一困难。并且考虑到前者方法[5]的插值结果的边长与期望边长的差距并不比后者方法[6]的插值结果小，因此我们猜测关于面网格的方法在边长这一方面仍有很大的进步空间，甚至可能能得到与期望边长几乎一样的插值结果。

从自由度方面来看，我们的猜测也是有道理的。对于一个亏格为 0 的封闭三维面网格，设其边，顶点和面数分别为 e, v, f。则由于每个面都是三角形，因此有 $3f = 2e$。另外由欧拉公式可知 $f + v - e = 2$。因此在已知所有边长的情况下，其自由度为 $3v - e = 3(e - f + 2) - e = 6$，正好对应于刚性变换的 6 个自由度。因此当边长确定时，从自由度的角度来看，存在满足条件的网格且网格各点的位置是确定的。并且 Euler[7]于 1766 年也提出了一个问题：当一个闭曲面没有被撕开时，它不能变形。之后 Cauchy[8]证明了当凸的多面体网格的各个面是刚性时，其整体是刚性的。Gluck[9]之后则证明了大多数亏格为 0 的三角形网格是刚性的。因此从这个角度来看，我们更有理由相信当边长确定时，存在至少一个网格满足边长条件，且网格各点的位置也是确定的。基于以上理由，我们尝试着在面网格上基于边长进行插值。

1.2　章节安排

本文的章节安排主要所示：

第一章是绪论。这章主要介绍了研究背景及形状插值。

第二章是相关工作。这章主要介绍已有的形状插值算法，三角形网格和四面体网格均匀涉猎。该章重点介绍的是之后的对比实验所用的算法。

第三章是算法。这章首先详细介绍了我们提出的基于边长的形状插值算法。之后又进行了各种实验，展示了该算法相较于其它算法的优势之处。

第四章是总结与展望。这章先是总结了我们方法的主要优点，然后指出了算法的不足之处，最后提出了之后的改进方向。

第 2 章　相 关 工 作

形状插值问题研究已经有几十年历史了，Sedeberg 等[10]早在 1993 年就研究了二维多边形插值问题。其相关工作非常多。如之前所说，形状插值可以基于许多标准，比如角度、边长、面积、仿射变换矩阵等。

2.1　基于简单的几何量

Winkler 等[11]利用了二面角和边长进行形状插值。该方法主要是将网格分解成多个层级后再从最底层往上逐层重建。

第一步，如图2.1[11]所示，他们将网格分解成多个层级。每次分解，将上一级大面片分成 m 个小面片。当大面片的三角形数小于 m 时，则不再分解。这里取 $m = 6$。最终会形成一棵树，顶部根节点包含整个网格，而底部节点则只有几个相邻的三角形面片。

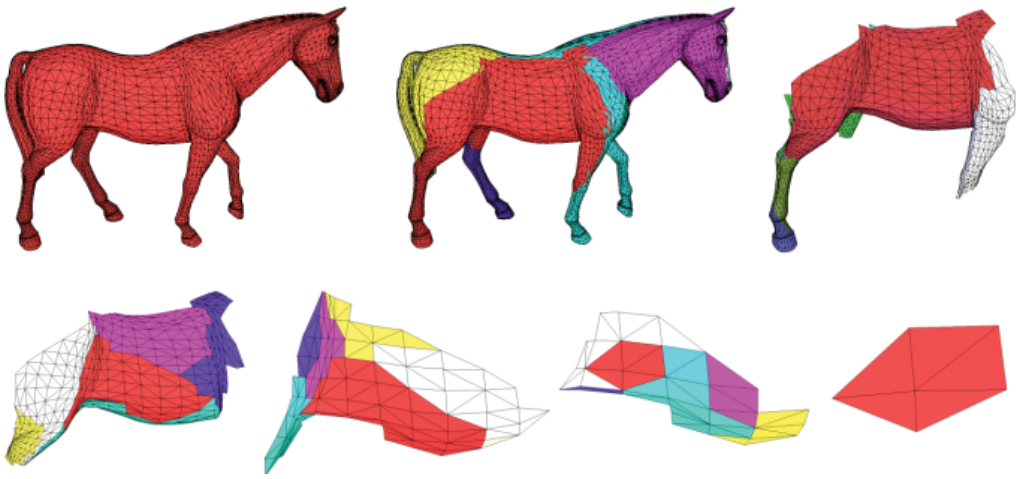

图 2.1　将左上角的网格分解成多个层级

第二步则需要把多个小面片合成一个大面片。首先得将多个小面片初步对齐，以便下一步优化。该方法在这一步使用的是 Williams[12]提出的算法。将多个小面片初步对齐后，该方法则将之合并一个大面片。用 $P^{[k]}$ 表示第 k 个小面片，P 代表大面片。如图2.2[11]所示，假设 $v_i^{[j]}$ 代表 v_i 在第 j 个面片的局部坐标系坐标。因此理论上，v_i 在面片的局部坐标和在右边环状面片中时的全局坐标之间应满足

$$v_i - v_j = v_i^{[k]} - v_j^{[k]} \tag{2.1}$$

其中 $k = 1, ..., 6$。

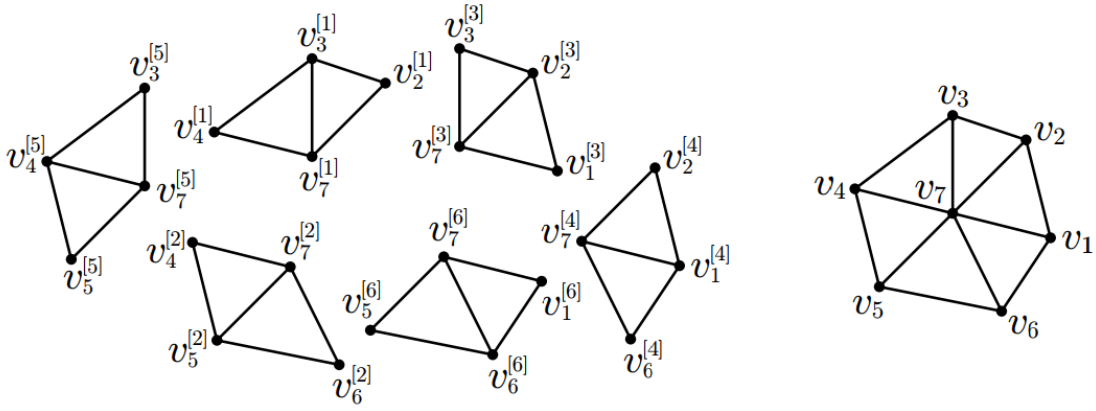

图 2.2　将六个面片 $P^{[1]},...,P^{[6]}$ 合成一个大的环状面片 P。

对于所有这些等式，可以将之表示成矩阵形式，即

$$\boldsymbol{M}v = e \tag{2.2}$$

其中 v 代表大面片 $\boldsymbol{P}$ 上的所有顶点的坐标；$\boldsymbol{M}$ 则是一个稀疏矩阵，对应于上式左边的部分，其系数为 1 或者-1；e 则是各个小面片的所有边，对应于上式右边的部分。接着可以在最小二乘法意义下求解上式，即求解

$$\min_{v} \parallel \boldsymbol{M}v - e \parallel_2^2 \tag{2.3}$$

通过求解上式，我们得到了 v，即面片 $\boldsymbol{P}$ 的各顶点坐标。为了得到更好的结果并且使边长达到我们期望的值，将式2.1替换成

$$v_i - v_j = \frac{v_i^{[k]} - v_j^{[k]}}{\parallel v_i^{[k]} - v_j^{[k]} \parallel} l_{ij}(t) \tag{2.4}$$

其中 $l_{ij}(t)$ 是 t 时刻边 $e_{ij} = [v_i, v_j]$ 的期望长度。然后对接下来的两式做出相应修改，即可得到更好的合并结果。重复上述过程，从最下一层逐渐向上重建，最终得到插值结果。结果如第四章图3.4和图3.5所示。

Wang 等[13]也是利用了二面角和边长进行形状插值。不过，他们并没有将网格分解成多个层次，而是在每个面上建立了一个局部坐标系 f。因此存在一个旋转矩阵 $\boldsymbol{R}_{ij}$ 使得 $f_j = f_i \boldsymbol{R}_{ij}$，其中 f_i, f_j 分别是网格面 i 和面 j 上的正交坐标架。因此，该方法定义该部分能量：

$$E_f(f) = \frac{1}{2} \sum_{e \in E} w_e^{-1} \parallel f_j - f_i R_{ij} \parallel^2 \tag{2.5}$$

其中 $\boldsymbol{E}$ 是边的集合，w_e 是余切权重[14]。每个面有了局部坐标系后，则每个三角形面 $\boldsymbol{T}$ 上的三个点均有相应的局部坐标。并且由于三角形边长是已知的，因此这些局部坐标也是确定的。而每个点在全局坐标系下也有坐标，因此可以定义该

部分能量：

$$E_x(x,f)=\frac{1}{2}\sum_{e=v_mv_n\in E}\sum_{T\supset e}w_{e,T}\parallel x_n-x_m-f_T(a^1_{mn,T},a^2_{mn,T},0)^T\parallel^2 \tag{2.6}$$

其中 v_m,v_n 是相邻两个顶点，x_m,x_n 则是顶点 v_m,v_n 在全局坐标系下的坐标，f_T 是三角形面 T 上的局部坐标系，$a^1_{mn,T},a^2_{mn,T}$ 则分别是顶点 v_m,v_n 在三角形面 T 的局部坐标系下的坐标。最后定义总的能量：

$$E_M(x,f)=wE_ff+E_x(x,f) \tag{2.7}$$

其中 w 是权重，该方法将之设为 1000。上式是一个关于 (x,f) 的二次多项式，因此只需要知道 $R_{ij},a^1_{mn,T},a^2_{mn,T}$ 即可以直接求解出其最小值并得到各顶点的坐标值 x。我们可以通过插值的方法得到的 $R_{ij},a^1_{mn,T},a^2_{mn,T}$。

该方法的主要不足是无法保证 f 是一个正交矩阵。这样进而导致得到的坐标 x 无法使网格的边长等于期望的边长。因此，该方法在边长误差这一指标上表现并不好，不如 Winkler 等[11]提出的方法。第四章的图3.4和图3.5展示了这一点。

Chen 等[15]同时使用边长、角度、二面角进行形状插值。因此其设能量为

$$E_s=\sum_e(l_e-l_e^*)^2 \tag{2.8}$$

$$E_b=\sum_e(\theta_e-\theta_e^*)^2 \tag{2.9}$$

$$E_v=(v-v^*)^2 \tag{2.10}$$

其中 e 为边，l_e 代表边 e 的实际长度，l_e^* 代表边 e 的目标长度；θ_e 代表边 e 对应的二面角的实际角度，θ_e^* 代表边 e 对应的二面角的目标角度；v 代表网格的实际体积，v^* 代表网格的目标体积。

接着，该方法再令总能量

$$E=\lambda E_s+\mu E_b+\upsilon E_v \tag{2.11}$$

其中 λ，μ，υ 是权重系数。

该方法通过使用多个参考模型来构成一个形状空间。给定 k 个参考模型 $M_1,...,M_k$，并且设 $L_e^{(i)}$，$\Theta_e^{(i)}$，$V^{(i)}$ 分别为其边长，二面角和体积。再给定模型 M。接着，该方法令式2.8，2.9和2.10中的各目标参数为

$$\begin{aligned}l_e^*&=L_e+\sum_{i=1}^k\alpha_i(L_e^{(i)}-L_e)\\ \theta_e^*&=\Theta_e+\sum_{i=1}^k\alpha_i(\Theta_e^{(i)}-\Theta_e)\\ v_e^*&=V_e+\sum_{i=1}^k\alpha_i(V_e^{(i)}-V_e)\end{aligned} \tag{2.12}$$

其中 α_i 为权重。当 $k=1$ 且令 $\alpha_i = t$，其中 $t \in [0,1]$ 为时间，则此时式2.12即可用于两个模型间插值。

该方法使用高斯-牛顿法进行优化。为了加快迭代速度并且避免其过早的陷入局部最优，该方法参考文章[16-17]设计了一套多分辨率优化方案。如图2.3所示，该方法首先将前一帧结果 $\boldsymbol{M}$ 进行简化，得到一个约 1000 个顶点的粗网格 $\boldsymbol{C}$。同时将粗网格 $\boldsymbol{C}$ 的顶点作为锚点将 $\boldsymbol{M}$ 转换为光滑模型 $\boldsymbol{B}$。然后在粗网格 $\boldsymbol{C}$ 上进行优化得到初步结果 $\boldsymbol{C}'$。接着利用粗网格锚点间的变换将光滑模型 $\boldsymbol{B}$ 形变为 $\boldsymbol{B}'$。最后利用文章[18]中的方法得到最终结果 $\boldsymbol{M}'$。

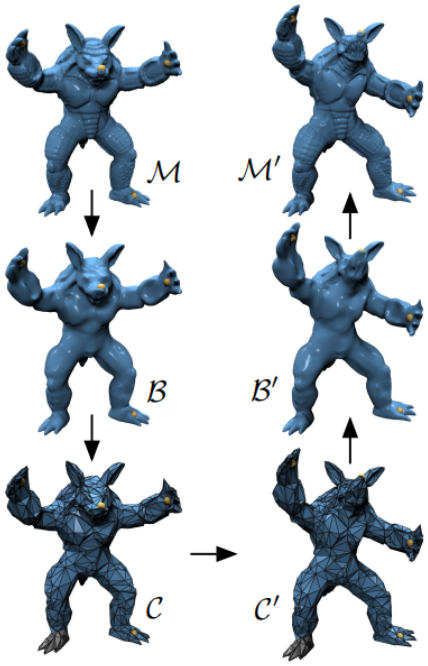

图 2.3　多分辨率优化方案[15]。左边一列是初始模型，右边一列是处理后的模型。上面一行是高分辨率模型，中间一行是光滑模型，下面一行是低分辨率模型。

相比于之前的几种方法都利用了二面角，Rojas 等[5]则提出了一种只利用边长就得到插值结果的方法。该方法首先使用边长及其它信息计算出二面角，然后再进行重建。

如图2.4所示，瓢虫围绕顶点 v 旋转时，在经过棱时需要围绕棱旋转，在经过三角形面时需要围绕面的法线旋转。若在顶点处放置一个正交坐标架，使瓢虫的坐标在该坐标系下一直为 $(1,0,0)$ 且始终面向 $(0,1,0)$。则当瓢虫围绕棱旋转，坐标架相应的围绕自身 x 轴旋转；当经过三角形面时，坐标架相应的围绕自身 z 轴旋转。假设顶点的度为 k，坐标架相应的围绕自身 x 轴旋转对应的旋转矩阵为 X_i，围绕自身 z 轴旋转对应的旋转矩阵为 Z_i，其中 $k=1,2,...,k$。则由于当瓢虫旋转一周时，坐标架也回到初始状态，因此有 $Z_1X_1Z_2X_2...Z_kX_k = I$。其中 I 是单位矩阵. 因此该方法定义顶点 v 处的能量：

$$E_v = \| I - Z_1X_1Z_2X_2...Z_kX_k \|_F \tag{2.13}$$

其中 F 代表 Frobenius 范数。对于网格上的所有点，定义总的能量：

$$E = \sum_{v} E_v \tag{2.14}$$

由于中间时刻 t 的边长是指定的，故该方法利用边长求出每个三角形面的三个角的角度。并且矩阵 Z_i 完全由角度决定，因此矩阵 Z_i 是已知的。而矩阵 X_i 只由相邻两面的二面角 α_i 决定，因此该方法通过优化二面角 α_i 来优化能量。于是该方法使用梯度下降法优化能量 E_i 并得到了对应的二面角 α_i。

获得二面角后，接下来利用它和边长来重建网格。该方法使用两步间接重建网格。第一步是计算顶点 v 处的局部坐标系在全局坐标系下的旋转矩阵 G_v。由于相邻两面的二面角和边长均是已知，因此可以计算出变换矩阵 $M_{v,u}$ 使得

$$G_v M_{v,u} - G_u = 0 \tag{2.15}$$

其中 v, u 是相邻两顶点。在最小二乘意义下求解上式，即能得到 G_v。不过这样得到的 G_v 往往并不是一个正交矩阵，所以也不是一个旋转矩阵。因此该方法使用特征值分解将其近似为一个正交矩阵。第二步则是使用 G_v 重建最终网格。对于相邻顶点 u, v，有

$$v + G_v u_v - u = 0 \tag{2.16}$$

其中 u_v 是顶点 u 在顶点 v 处的局部坐标系下的坐标。可以在最小二乘意义下求解上式，得到顶点坐标 u。

可以看到，该方法仅用边长就得到插值结果。不过该方法也有不足。首先，其计算速度慢。主要有两个原因。第一，该方法使用的是梯度下降法，迭代次数多。第二，每次迭代需要重新计算系数，并且该系数需要通过大量矩阵计算才能得到。其次，其作为一个仅用边长插值的方法，边长误差这一指标仅略好于 Winkler 等[11]提出的方法。第四章的图3.4和图3.5也体现了这一点。之所以会这样，很可能是由于该方法得到二面角后经过两步才得到最终的网格，因此边长的误差会被放大。

除了面网格插值算法外，还有体网格插值算法。Paillé 等[19]就提出了一个插值四面体网格的算法，其基于的是四面体网格的二面角。

由格林公式，易得 $\sum_i n_i a_i = 0$，其中面 i 是一个四面体 t 的第 i 个面，n_i 和 a_i 则分别是三角形面 i 的单位法向量和面积。将上式左右两边点乘法向量 n_j 得到

$$\begin{pmatrix} 1 & -cos\theta_{01} & -cos\theta_{02} & -cos\theta_{03} \\ -cos\theta_{01} & 1 & -cos\theta_{12} & -cos\theta_{13} \\ -cos\theta_{02} & -cos\theta_{12} & 1 & -cos\theta_{23} \\ -cos\theta_{03} & -cos\theta_{13} & -cos\theta_{23} & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.17}$$

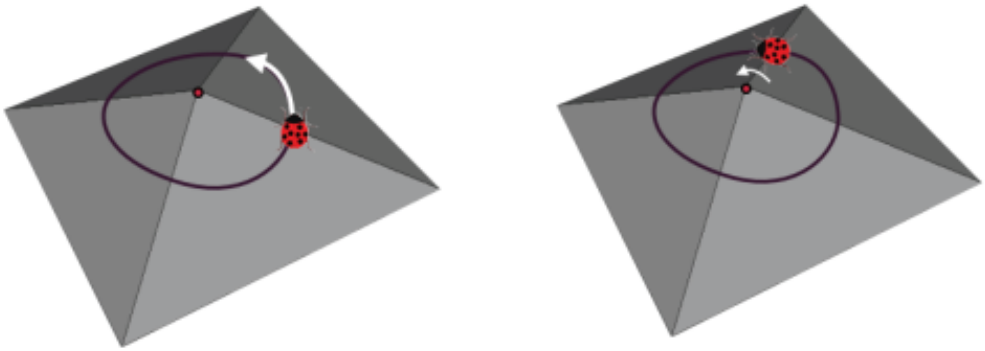

图 2.4　瓢虫围绕一个顶点旋转

我们将左边的 4 阶矩阵定义为 G_t，并且易得

$$\det(G_t) = 0 \tag{2.18}$$

设 C_t 是 G_t 的余子式，可证明其应为正定矩阵。设 t_0 和 t_1 是相邻的两个四面体，面 i 是它们的公共的面，则有

$$cos\phi_{ij}^0 = cos\phi_{ij}^1, j = 0, 1, 2 \tag{2.19}$$

其中 j 代表面 i 的第 j 个角。如图2.5[19]所示，我们定义

$$w = \theta_0 + \theta_1 + \theta_2 - \pi \tag{2.20}$$

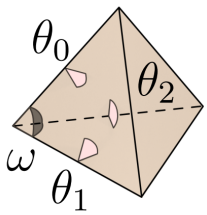

图 2.5

从角度出发，我们定义现有角度与目标角度的差为

$$f(\Theta) = \| \Theta - \hat{\Theta} \|^2 + \| \Omega - \hat{\Omega} \|^2 \tag{2.21}$$

其中 Θ 是包含所有二面角的集合，$\hat{\Theta}$ 则是包含所有的目标二面角的集合；Ω 和 $\hat{\Omega}$ 则是使用式2.20从 Θ 和 $\hat{\Theta}$ 得到的。$\| \Omega - \hat{\Omega} \|^2$ 可以有效的防止尖锐、细长的

四面体。结合之前的限制条件可以定义优化问题为

$$\begin{aligned} \min \quad & f(\Theta) \\ \text{s.t.} \quad & \det(G_t) = 0 \wedge C_t > 0 \qquad \forall t \in T \\ & \phi_c^0 = \phi_c^1 (\text{式}2.19) \qquad \forall c \in C \\ & \sum_{\theta \in \Theta_e} \theta = 2\pi \qquad \forall e \in E \\ & \theta_b \leqslant \theta \leqslant \pi - \theta_b \qquad \theta \in \Theta \end{aligned} \tag{2.22}$$

其中 T 是所有的三角形；C 是所有的四面体的三角形的角；Θ_e 是边 e 周围的所有的二面角；Es 是所有的边；θ_b 是一个正数，用于限制二面角的范围；Θ 是所有的二面角。我们使用软约束优化上述问题，即定义能量为

$$E(\Theta) = \alpha f(\Theta) + \| c(\Theta) \|^2 + \| \min(d(\Theta), 0)) \|^2 \tag{2.23}$$

其中 α 是一个常数，该方法设 $\alpha = 10^{-6}$；$c(\Theta)$ 代表式 2.22中的所有等式限制条件，$d(\Theta)$ 代表式2.22中的所有不等式限制条件。该方法将线性插值得到 Θ 作为其初始化，然后使用 L-BFGS 法进行优化，得到了所有的二面角 Θ。

最后，该方法利用这些二面角分为以下几步进行重建。第一步为重建每一个四面体。对于一个四面体 t，其边长为

$$l_{ij} = b_t \frac{\partial |G_t|}{\partial \theta_{ij}} \tag{2.24}$$

其中 b_t 是一个常数，并且使得四面体 t 的其中一条边长度为 1。文章[20]有该式的证明。

第二步则是全局重建。首先如图2.6所示，v_4 的关系可以表示为

$$\begin{aligned} & v_4 = a_0 v_0 + a_1 v_1 + a_2 v_2 + a_3 v_3 \\ & \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ 1 & 1 & 1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} v_4 \\ 1 \end{pmatrix} \end{aligned} \tag{2.25}$$

其中 v_i 是 3 维向量，a_i 是重心坐标。由于相邻的两个四面体可以放在一个局部坐标系里，而且 a_i 在旋转平移下不变，因此可以计算出 a_i。将所有相邻的两个四面体对应的式2.25写成一个线性系统，得到 $Ax = 0$。其中 A 的各项为 a_i，x 则是顶点坐标。接着只要在最小二乘意义下求解上式即可得到四面体网格的顶点坐标。

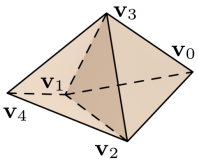

图 2.6　两个相邻的四面体

该方法主要有三点不足。第一是能量函数非线性非凸，难以优化。第二是其重建方法只能重建亏格大于 0 的模型。第三是该方法不能保证全局不自交。

Chen 等[6]提出了一种有界扭曲的四面体网格插值算法。该方法基于一个观察，即插值边长的平方后得到的四面体既是有界扭曲的，截面离散曲率 (Discrete Sectional Curvature) 也较小。对于一个四面体网格，如图2.7所示，四面体网格内部的一条边 e_j 截面离散曲率定义为

$$K_{e_j} = 2\pi - \sum_{t_i \in N(e_j)} \beta_j^i \tag{2.26}$$

其中 $N(e_j)$ 是边 e_j 的一邻域的四面体的集合，β_j^i 则是四面体 t_i 中边 e_j 所对应的二面角。

图 2.7　一条内边周围的四面体[6]。

然后，该方法利用了[21]中的一个定理说明了一事实，即包含所有满足式2.27的 n 阶矩阵 M 的集合是一个凸集。

$$\begin{aligned} \lambda_1(M) &\leqslant \alpha \\ \lambda_n(M) &\geqslant \beta \\ K(M) &\leqslant \gamma \end{aligned} \tag{2.27}$$

其中 $\lambda_1(M)$ 代表矩阵 M 的最大特征值;$\lambda_n(M)$ 代表矩阵 M 的最小特征值;$K(M) = \frac{\lambda_1(M)}{\lambda_n(M)}$;$\alpha, \beta, \gamma$ 则均为常数。式2.27的前两个子式限制了变换的扭曲，使其倾向于成为一个等距变换；最后一个子式则使其倾向于成为一个共形变换。

接着，其又证明了式2.27等价于下式

$$\begin{aligned} M - \lambda_n I &\geqslant 0 \\ \lambda_1 I - M &\geqslant 0 \\ sI - M &\geqslant 0 \\ K^2 M - sI &\geqslant 0 \end{aligned} \tag{2.28}$$

其中 $\lambda_1 \geqslant \lambda_n > 0$，$K \geqslant 1$，$s \in R$ 是一个松弛变量。

对于式2.34中的变换矩阵 J_i，我们可以定义其对称狄利克雷扭曲 (Symmetric Dirichlet Distortion) 为

$$\zeta(J_t) = \sum_{i=1}^{n} \sigma_i^2 + \sigma_i^{-2} \tag{2.29}$$

其中 σ_i，$i = 1, ..., n$ 是矩阵 J_t 的特征值。其作为能量可以有效防止翻转的现象出现。接着其利用上述定义证明了下式

$$\zeta(M) \leqslant \zeta \tag{2.30}$$

等价于

$$\begin{gathered} Tr(M) + Tr(Y) \leqslant \zeta \\ \begin{pmatrix} Y & I \\ I & M \end{pmatrix} \geqslant 0 \end{gathered} \tag{2.31}$$

其中 ζ 是用户自定义的常数，M 是一个正定矩阵，Y 是一个与 M 有关的松弛对称矩阵 (Slack Symmetric Matrix)。

该方法所使用的能量是对称狄利克雷能量，即 $\sum_{t_i} \zeta(M_i)|t_i|$，其中 $|t_i|$ 是四面体 t_i 的体积，M_i 是 t_i 对应的变换矩阵。值得注意的是，这里的 M_i 不是相对于源四面体的变换矩阵，而是相对于参考四面体 (将边长平方线性插值获得的四面体) 的变换矩阵。之所以能这么做，是因为式2.27定义的集合是凸集这一性质保证了参考四面体相对于源四面体是有界扭曲的。是一个凸集其考虑到所有内边 e_j 的截面离散曲率，即 $K_{e_j} = 0$，因此将之作为惩罚项加入到能量中。因此其能量应该是

$$E = \lambda \parallel \vec{K} \parallel^2 + \sum_{t_i} \zeta(M_i)|t_i| \tag{2.32}$$

其中 λ 是权重，$\vec{K}$ 是将所有的 K_{e_j} 组合在一起的向量。利用式2.30等价于式2.31这一事实，再一阶近似 K，则利用论文中方法，该优化问题的第 l 次迭代可被转换

为

$$\begin{aligned}
\min_{\vec{x},S,R} \quad & \lambda_l \parallel \vec{K}_{\vec{x}_l} + J_{\vec{x}_l}\vec{x} \parallel^2 + \sum_{t_i}(Tr(M_i^R) + Tr(R_i))|t_i| \\
\text{subject to} \quad & M_i^S = T_i^S[\vec{x}]_i \\
& M_i^R = T_i^R[\vec{x}]_i \\
& \begin{pmatrix} R_i & I \\ I & M_i^R \end{pmatrix} \geqslant 0 \\
& Tr(M_i^S) + Tr(S_i) \leqslant \zeta_i \\
& \begin{pmatrix} R_i & I \\ I & M_i^R \end{pmatrix} \geqslant 0
\end{aligned} \tag{2.33}$$

其中 λ_l 是权重；$\vec{x}$ 是边长变量；$\vec{x}_l$ 是第 l 次迭代时的边长初始值；S_i, R_i 是松弛对称矩阵；ζ 是用户指定的相对于源模型的对称狄利克雷扭曲，该文章将之设为 $1.01\zeta_i^{S\to T}$；$[\vec{x}]_i$ 代表向量 $\vec{x}$ 中有关 t_i 的部分；T_i^S 代表将源四面体映射到目前的四面体对应的变换矩阵；T_i^R 代表将参考四面体映射到目前的四面体对应的变换矩阵。注意到这里同时存在 M_i^S 和 M_i^R，这是因为有界扭曲是相对于源模型，而对称狄利克雷能量则是相对于参考模型。对于式2.33，我们可以使用文章[22]的算法 3.1 进行优化。事实上，该论文对插值问题的优化进行了一系列处理以加快速度，限于篇幅就不再阐述了。

2.2　ARAP 及相关算法

除了使用二面角，边长, 体积等简单几何量进行插值外，还有些算法基于仿射变换矩阵进行形状插值。Alexa 等[23]提出了 ARAP(As-Rigid-As-Possible) 方法。该方法首先计算出在 t 时的仿射变换矩阵，然后进行重建。

对于源网格和目标网格的一个对应的第 i 个三角形面 T_i^1 和 T_i^2, 存在一个变换矩阵 J_i 和向量 b 使得

$$T_i^2 = J_i T_i^1 + b \tag{2.34}$$

若将这两个三角面的对应的一个顶点平移至原点，则此时 $b = 0$。此时，我们只需插值矩阵 J_i 即可。在时刻 t 时，设此时插值出的变换矩阵为 $A(t)$。最简单的做法自然是令 $A(t) = (1-t)I + tA$。不过使用该矩阵得到的结果不够自然。为了得到更好的效果，可以使用极分解将变换矩阵分解成旋转旋转部分 R 和放缩部分 S，即 $J = RS$。接着分别对旋转部分和放缩部分进行插值，得到了目标的仿射变换矩阵 $A(t) = R(t)((1-t)I + tS)$，其中的 $R(t)$ 是通过对 R 进行旋转角度插值

得到的旋转矩阵。接着我们设能量

$$E = \sum_{f} \| J - A(t) \|_F^2 \tag{2.35}$$

其中 $\|\|_F$ 代表 Frobenius 范数。我们只需将 J 写成有关顶点坐标的表达式后代入。此时式2.35为一个关于顶点坐标的二次表达式，可以直接求解出其最小值及此时的顶点坐标。不过 ARAP 方法也有不足之处，比如可能会出现翻转。

Chao 等[24]在 ARAP 的基础上提出了一种新的插值算法。

给定一个映射 $f : M \to \widetilde{M}$，其中 M 和 $\widetilde{M}$ 是两个网格。则 ARAP 能量也可以写成以下形式[25]：

$$E(f) = \int_M dist(df, SO(n))^2 = \int_M \min_{R \in \mathrm{SO}(n)} |df - R|^2 \tag{2.36}$$

其中 n 为空间维度，SO(n) 是包含所有 n 阶正交矩阵的集合。当 f 固定且其行列式大于 0 时，将其极分解为 $df = RY$，则这个 R 即是上式取到最小值的 R。因此此时有 $df - R = R(Y - I)$。将其分解为迹为 0 的矩阵和常数矩阵两部分，即

$$Y - I = (Y - \frac{\mathrm{tr}(Y)}{n} I) + (\frac{\mathrm{tr}(Y)}{n} - 1)I =: \overline{Y} + yI \tag{2.37}$$

于是我们可以定义能量

$$E_{\alpha,\beta}(f) = \int_M |\beta\overline{Y} + \alpha y I|^2 = \int_M \beta^2 |\overline{Y}|^2 + \frac{\alpha^2}{n}(\mathrm{tr}(Y) - n)^2 \tag{2.38}$$

可以看到，当 $\alpha = 0$ 且 $\beta = 1$ 时，该能量即是 ASAP(as-similar-as-possible) 能量[26]；当 $\alpha = 1$ 且 $\beta = 1$ 时，该能量即是 ARAP 能量。

若 M_0 和 M_1 分别是源网格和目标网格，设 M_t 是时刻 t 时的网格，则该方法定义其为

$$M_t = \mathrm{argmin}_{M_x}(1 - t)d(M_0, M_x) + td(M_x, M_1) \tag{2.39}$$

接着，可以使用能量 $E_{\alpha,\beta}$ 代替 d，即

$$\widetilde{M}_t = \mathrm{argmin}_{M_x}(1 - t)E_{\alpha,\beta}(M_0 \to M_x) + tE_{\alpha,\beta}(M_1 \to M_x) \tag{2.40}$$

其中 $\widetilde{M}_t$ 即是时刻 t 时的插值结果。

Gao 等[27]通过利用相同拓扑模型的数据库指导 ARAP 方法进行插值。其主要思想就是在数据库中寻找一条最短路径将源网格和目标网格连接起来。

由于要寻找最短路径，因此需要定义两个模型间的距离。对于具有相同拓扑的两个模型 S_i 和 S_j，该方法定义它们之间距离为

$$\overline{d}(S_i, S_j) = \sqrt{\frac{\sum_{k=1}^{n} \| S_i[k] - S_j[k] \|^2}{n}} \tag{2.41}$$

其中 $S_i[k]$ 是数据库中第 i 个模型的第 k 个顶点，n 是模型顶点的数量。接着为了将密集的 $\overline{d}(S_i, S_j)$ 分隔开，定义距离度量 $d = F(\overline{d}) = d^2$。接着由于数据库的模型太少，于是该方法先是使用近邻传播算法[28]和 ARAP 插值来获得可靠的模型。然后使用上一步的近邻传播算法再次将新增的模型放入若干个局部线性形状空间中。假设该空间有 n 个新增的模型，则该空间可以用这 n 个模型线性表示。假设 S_s 和 S_t 分别是源模型和目标模型，将插值问题转换为寻找最短路径问题:

$$min \sum_{i=0}^{m+1} \sum_{j=0}^{m+1} f_{ij} x_{ij} \tag{2.42}$$

并且有：

$$\sum_{j=0}^{m+1} x_{ij} - \sum_{j=0}^{m+1} x_{ji} = \begin{cases} 1 & i = 0 \\ -1 & i = m+1 \\ 0 & i = 1, ..., m \end{cases} \tag{2.43}$$

其中 m 为空间数，x_{ij} 为取值 0 或 1 的变量，$x_{ij} = 1$ 代表存在一条路径从空间 C_i 到空间 C_j。并且 $f_{ij} = F(d_{ij}) = \overline{d}(cs_i, cs_j)$，其中 cs_i、cs_j 分别是空间 C_i 到空间 C_j 的参考模型。然后该方法通过交替迭代优化的方法有效的求解式2.42。

该方法获得了 N_R 个被选中空间的参考模型 $\overline{S}_k$，$k = 1, 2, ..., N_R$ 接着利用这些模型指导插值。我们假设 $\hat{S}(t)$ 是 t 时刻的插值结果，于是该方法定义了 $\hat{S}$ 和 $\overline{S}_k$ 间的能量为

$$E_k = \sum_{i=1}^{n} \sum_{j \in N_i} (w_{ij} \parallel (\hat{S}[i] - \hat{S}[j]) - R_k[i](\overline{S}_k[i] - \overline{S}_k[j]) \parallel^2) + \gamma \parallel \hat{S}[i] - \overline{S}_k[i] \parallel^2 \tag{2.44}$$

其中 n 是顶点数，N_i 是顶点 i 的一邻域，w_{ij} 是余切权重[14]，$R_k[i]$ 是 $\hat{S}(t)$ 与 $\overline{S}_k$ 之间在顶点 i 处的最优旋转矩阵，γ 为常数且该方法将之设为 0.001。可以看到该能量基本上就是 ARAP 能量。接着该方法定义 $\hat{S}$ 与所有参考模型之间总的能量为

$$E(t) = \sum_{k=1}^{N_R} \exp(-\epsilon |t - t_k|^2) E_k \tag{2.45}$$

其中 ϵ 为常数，且该方法将之设为 6.0；$t_k = \frac{k-1}{N_R - 1}$。最后再次使用交替迭代的方法求解式2.45。

对于那些源模型和目标模型的拓扑与数据库中的模型的不一样的情况，该方法也有一定办法进行处理。假设 N_S 和 N_T 分别是源模型和目标模型。然后在数据库中寻找这两个模型对应的最近模型 S_S 和 S_T。由于这样得到的 S_S 和 S_T 往往与 N_S 和 N_T 的姿势相差较大，因此使用 ARAP 能量求得更相似的 $\hat{S}_S$ 和

$\hat{S}_T$，即定义能量：

$$E(\hat{S}_S)=\sum_{i=1}^{n}\sum_{j\in N_i}(w_{ij}\parallel(\hat{S}_S[i]-\hat{S}_S[j])-R[i](S_S[i]-S_S[j])\parallel^2)+\lambda\parallel\hat{S}_S[i]-\hat{N}_S[i]\parallel^2 \tag{2.46}$$

其中 λ 为一常数，取值范围为 [0.2,1]。使用与前面类似的交替方法可以求得 $\hat{S}_S$。同理也可以得到 $\hat{S}_T$。将 $\hat{S}_S$ 和 $\hat{S}_T$ 视为源模型和目标模型，接着就可以类似获得参考模型 $\overline{S}_k$。最后该方法利用论文[18]和[29]将 $\hat{S}_S$ 与 $\overline{S}_k$ 之间的变换迁移到 $\hat{N}_S$ 以获得参考模型 $\hat{S}_k$。然后使用之前利用参考模型插值的方法即可得到插值结果。

该方法通过利用数据库，能获得比 ARAP 更好的结果。但是该方法也有不足之处。首先，为了获得好的效果，其需要相当大的数据库。其次，该方法由于利用数据库指导插值，因此产生的结果也往往与数据库的模型相似。最后，对于那些与数据库中模型相差较大的源模型或者目标模型，其可能会产生低质量的结果。

几年后，Gao 等[30]又提出了一种新的数据驱动方法用于形状插值。并且该方法允许用户交互来获得更好的效果。

2.3　其它形状插值方法

Kilian 等[16]提出了 AIAP(as-isometricas-possible) 方法用于形状插值。该方法通过保证网格上两点的测地距离不变来插值，即使形变过程是等距变换。

假设网格 M 形状插值变换为 $\varphi:[0,1]\times M\rightarrow \mathrm{R}^3$，则顶点 p 的路径可以表示为 $p(t)=\varphi(t,p)$。因此 M 在 t 时刻的形变场 (deformation field) 可以定义为

$$X(t):=\left(\frac{d}{dt}p(t)\right)_{p\in M} \tag{2.47}$$

可以证明一个形状插值变换是等距变换当且仅当

$$<X_p(t)-X_q(t),p(t)-q(t)>=0 \tag{2.48}$$

对任何一条边 $e=(p(t),q(t))$ 成立，其中 $<,>$ 为欧几里得空间的向量内积。

不过由于源模型和目标模型的对应边的边长往往是不同的，因此此时变换也不可能是等距的，所以该方法将目标定为使式2.48左边尽量小，这也是 AIAP 这一名字的由来。对于两个形变场 X 和 Y，定义它们的内积为

$$\ll X,Y\gg_M:=\sum_{(p,q)\in M}<X_p-X_q,p-q><Y_p-Y_q,p-q> \tag{2.49}$$

不过上式定义出的内积与一般意义上的内积相比，并不满足正定性。例如当 $X=1$ 时满足 $\ll X,X\gg_M=0$。

为了解决这一问题，该方法加了一项正则项

$$\ll X,Y\gg_M^{L^2}:=\sum_{p\in M}<X_p,Y_p>A_p \tag{2.50}$$

其中 A_p 是点 p 邻域内所有三角形面积和的三分之一。从而定义 X 和 Y 的内积为

$$\ll X,Y\gg_{M,\lambda}:=\ll X,Y\gg_M+\lambda\ll X,Y\gg_M^{L^2} \tag{2.51}$$

其中 λ 为一常数，该方法取为 0.001。

图2.8展示了算法流程。设 M 和 N 分别是源网格和目标网格。该方法首先使用文章[31-32]的方法，同时将网格 M 和 N 的对应边进行简化。每次迭代，将简化具有最小损失和的边。最终得到了两个简化后的模型 M^0 和 N^0。

考虑到输出结果是一个离散的插值序列，该方法因此需要针对离散的插值序列定义了能量。假设 $P_0,P_1,...,P_{n+1}$ 是一个插值序列，$X_0,X_1,...,X_n$ 则是插值序列之间的形变场该方法定义 P_i 的对称能量

$$E(P):=\sum_{i=0}^{n}\left(\ll X_i,X_i\gg_{P_i}+\ll X_i,X_i\gg_{P_{i+1}}\right) \tag{2.52}$$

其是连续能量 $\int\ll X,X\gg_{P(t)}dt$ 的离散化形式。该方法使用了拟牛顿法优化上式。

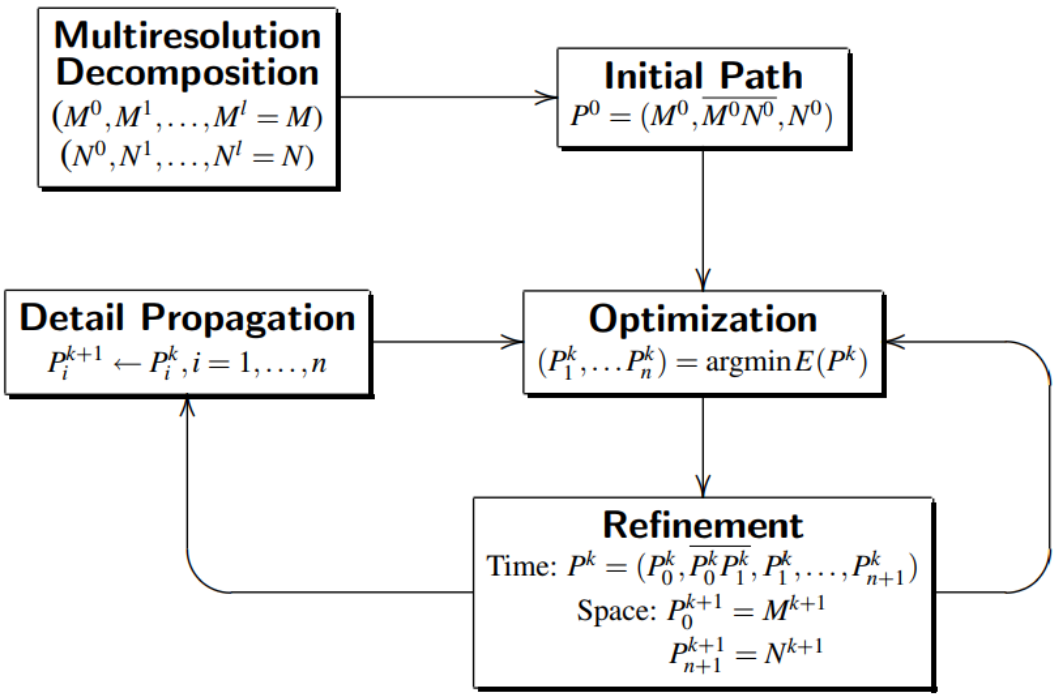

图 2.8　算法流程[16]

优化上述能量之后需要进一步优化以获得更加光滑且精细的插值序列。该方法有两种优化方式。第一种方式是时间优化，即增加更多的中间网格。文章主

要使用简单的线性插值增加新的中间网格，例如将 P_0^k 和 P_1^k 进行线性插值获得网格 $\overline{P_0^k P_1^k}$。第二种方式是空间优化，即增加网格分辨率，从简化后的粗网格变为简化前的细网格。这两种优化方式互相独立，可以交替进行，直到得到首尾分别为 M 和 N 的插值序列方停止。空间优化较为复杂，故进行较为详细的介绍。在第 k 次空间优化中，令第一帧的网格 $P_0^k = M^k$，且最后一帧的网格 $P_{n+1}^k = N^k$。接着该方法求出 $P_i^{k+1}, i = 0, 1, ..., n+1$。首先有 $P_0^{k+1} = M^{k+1}$ 和 $P_{n+1}^{k+1} = N^{k+1}$。然后将新增的细节扩散到中间网格 $P_1^k, ..., P_n^k$。对于任何一个新增到 P_0^k 的点 p，该方法存储其投影点 p' 所在平面的索引，并且计算其重心坐标。除此之外，其还计算其在法向量 N_f 上的坐标值 $< p - p', N_f >$。接着就可以利用重心坐标和法向量的坐标将顶点 p 加在 P_i^k 上，最终得到一细网格，设为 $P_{i,0}^{k+1}$。使用类似的方法也可以从 P_{n+1}^k 得到一细网格 $P_{i,n+1}^{k+1}$。最后，该方法使用线性插值得到最终结果，即令

$$P_i^{k+1} = \frac{n+1-i}{n+1} P_{i,0}^{k+1} + \frac{i}{n+1} P_{i,n+1}^{k+1} \tag{2.53}$$

上述部分已经能得到在时间 $t \in [0, 1]$ 的插值序列。该方法还能进一步得到 $t > 1$ 的形变序列。假设 $P \in \mathrm{R}^{3m}$ 代表网格 M 的所有顶点位置，m 是顶点数，定义

$$F(t, P, \dot{P}) := \ll \dot{P}(t), \dot{P}(t) \gg_{M(t)} \tag{2.54}$$

则插值序列应最小化能量 $\int F dt$。因此，由欧拉-拉格朗日方程得其应该满足等式

$$F - \dot{p}_i^k \frac{\partial F}{\partial \dot{p}_i^k} = C_{i,k} \quad i = 1, ..., m, k = 1, 2, 3 \tag{2.55}$$

其中 $p_i = (p_i^1, p_i^2, p_i^3)$ 代表网格 M 的第 i 个顶点，$C_{i,k}$ 是与 i, k 有关的常数。

假设 M_0 和 X_0 分别代表初始网格和初始方向，我们要利用它获得接下的形变序列 $M_j (j > 0)$。首先，$M_1 = M_0 + \Delta t X_0$ 且 $C_{i,k}$ 可以求得。接着我们需要计算 X_j，即优化下式

$$f(p) = \sum_{i=1}^{m} \sum_{k=1}^{3} (F - \dot{p}_i^k \frac{\partial F}{\partial \dot{p}_i^k} - C_{i,k})^2 \tag{2.56}$$

使用 X_{j-1} 作为 X_j 的初始值，利用拟牛顿法即可求解得到 X_{j+1}。接着令 $M_{j+1} = M_j + \Delta t X_j$。重复上述过程，即可得到 $t > 1$ 时刻的外插 (extrapolation) 结果。

文章最后提到，该方法的主要不足是不能保证不自交，可能会发生穿模的情况。

Kircher 等[33]提出了一种利用物体内部元素的相对位置关系来插值的方法。如图2.9所示，τ 是三角形，而坐标架 D_τ 指的是 τ 的一个局部坐标架，同理坐标架 D_σ 指的是 σ 的一个局部坐标架。因此 D_τ 可以表示为 D_τ=[$u_\tau v_\tau n_\tau$]，其中 u_τ

和 v_τ 是三角形 τ 的两条边，而 n_τ 则是三角形 τ 的单位法向量。$Q_{\sigma\tau}$ 则是将 D_σ 转换为 D_τ 的映射，即 $Q_{\sigma\tau} = D_\tau^{-1} D_\sigma$。假设 D_τ^t 是 t 时刻的坐标架。考虑到 D_τ^t 的

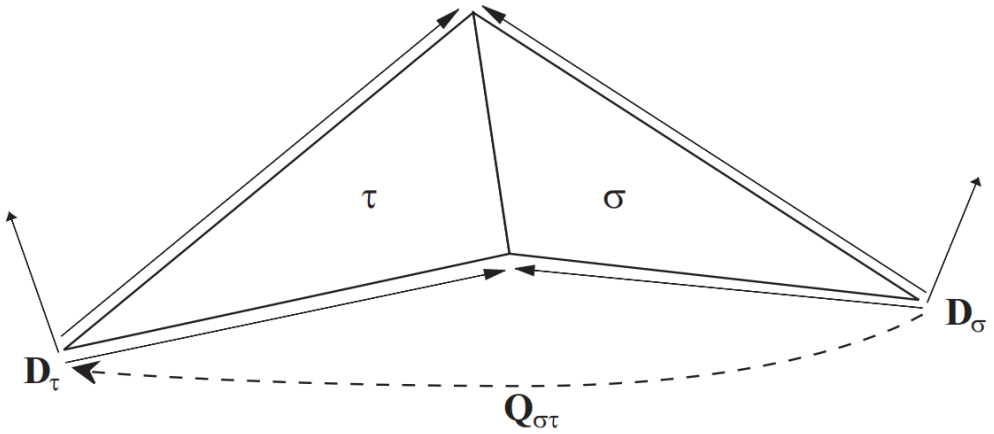

图 2.9　三面体 D_τ 和 D_σ

法向模长为 1 且与剩下两向量垂直，因此我们可以将其极分解为

$$D_\tau^t = R_\tau^t \hat{S}_\tau^t = \text{matrix}(q_\tau^t) \begin{pmatrix} S_\tau^t & 0 \\ 0 & 1 \end{pmatrix} := (q_\tau^t, S_\tau^t) \tag{2.57}$$

其中 q_τ^t 代表一个单位四元数，S_τ^t 则是一个二维的对称矩阵，$\hat{S}_\tau^t$ 则是一个三维的对称矩阵。

接着定义绝对减号 $\ominus$(absolute subtraction) 为

$$(q_\tau^t, S_\tau^t) \ominus (q_\tau^s, S_\tau^s) = (q_\tau^t (q_\tau^s)^{-1}, S_\tau^t - S_\tau^s) \tag{2.58}$$

这里的绝对指的是绝对世界坐标。同时定义绝对加号 $\oplus$ 为

$$w(q_\tau^t, S_\tau^t) \oplus (q_\tau^s, S_\tau^s) = (\text{pow}(q_\tau^t, w) q_\tau^s, w S_\tau^t + S_\tau^s) \tag{2.59}$$

其中 $w \in [0, 1]$ 为权重。然后我们可以定义 blend 操作符为

$$\text{blend}(D^s, D^t, w) = w(D^t \ominus D^s) \oplus D^s \tag{2.60}$$

不过由于该 blend 操作符依赖于三角形在世界坐标的朝向，因此仍然需要改进。给定极分解 (q_σ^t, S_σ^t) 和 (q_τ^t, S_τ^t)，则有

$$Q_{\sigma\tau}^t = (R_\tau^t \hat{S}_\tau^t)^{-1} R_\sigma^t \hat{S}_\sigma^t = (\hat{S}_\tau^t)^{-1} (\text{matrix}((q_\tau^t)^{-1} q_\sigma^t)) \hat{S}_\sigma^t \tag{2.61}$$

因此我们可以通过线性插值 S_σ 和 S_τ，球面线性插值 $(q_\tau^t)^{-1} q_\sigma^t)$ 来插值 $Q_{\sigma\tau}^t$ 和 $Q_{\sigma\tau}^s$。

当我们有 $Q_{\sigma\tau}$ 时，我们可以利用它来还原网格。由 $Q_{\sigma\tau}$ 的定义，可以得到 $Q_{\sigma\tau}^T D_\tau^T - D_\sigma^T = 0$。对于网格的所有相邻三角形，我们可以将之写成矩阵的形式，即存在一个矩阵 $H \in \mathrm{R}^{3p*3m}$ 使得

$$H D^T = 0 \tag{2.62}$$

其中矩阵 D 是将 D_σ^T 排成一列得到的矩阵。接着在最小二乘意义下就可以通过求解式2.62得到 D。

求到 D_σ 后，可以求得顶点位置。考虑到 $D_\sigma = [u_\sigma \quad v_\sigma \quad n_\sigma]$，因此得到了 u_σ 和 v_σ。因此有

$$F_\sigma = [u_\sigma \quad v_\sigma] = [(x_i - x_j) \quad (x_k - x_i)] \tag{2.63}$$

其中 (i, j, k) 是三角形 σ 的三个顶点。像刚才一样，依然可以将所有三角形对应的式2.63写成一个矩阵等式，即 $GX = 0$，其中 X 代表顶点位置。接着在最小二乘意义下就可以通过求解即可得到 X，即重建出插值网格。

第 3 章　基于边长的三维曲面插值

3.1　能量优化

设顶点 i 的坐标为 p_i，则顶点 i 和顶点 j 连成的边 e_{ij} 的长度为 $\|p_i - p_j\|$。我们的目标是优化能量使得最终的曲面的边长等于目标边长。对于 e_{ij}，定义能量

$$E_{ij} = \|\|p_i - p_j\|^2 - l_{ij}^2\|^2 \tag{1}$$

$$E = \sum_{e_{ij}} E_{ij} \tag{2}$$

其中 l_{ij} 是 e_{ij} 的目标边长。

对于一个网格，设其所有边的集合为 E，所有点的集合为 V。设顶点 i 的坐标为 p_i，则顶点 i 和顶点 j 连成的边 e_{ij} 的长度为 $\|p_i - p_j\|$。我们的目标是优化能量使得最终的曲面的边长等于目标边长。因此对于边 e_{ij}，我们定义能量

$$E_{ij} = \|\|p_i - p_j\|^2 - l_{ij}^2\|^2 \tag{1}$$

$$E_l = \sum_{e_{ij} \in E} E_{ij} \tag{2}$$

其中 l_{ij} 是 e_{ij} 的目标边长，E_l 是网格所有边的能量总和。

接下来我们要优化能量 E_l。由于牛顿法收敛速度相比梯度下降法和拟牛顿法更加快，因此我们使用牛顿法，于是需要计算一阶梯度和二阶偏导。设 $p_i = (p_{i1}, p_{i2}, p_{i3})$，则通过计算可得，

$$\frac{\partial E_{ij}}{\partial p_{ik}} = 4(\|p_i - p_j\|^2 - l_{ij}^2)(p_{ik} - p_{jk}), \tag{3a}$$

$$\frac{\partial^2 E_{ij}}{\partial p_{ik}^2} = 8(p_{ik} - p_{jk})^2 + 4(\|p_i - p_j\|^2 - l_{ij}^2), \tag{3b}$$

$$\frac{\partial^2 E_{ij}}{\partial p_{ik} \partial p_{jk}} = -8(p_{ik} - p_{jk})^2 - 4(\|p_i - p_j\|^2 - l_{ij}^2), \tag{3c}$$

$$\frac{\partial^2 E_{ij}}{\partial p_{ik} \partial p_{il}} = 8(p_{ik} - p_{jk})(p_{il} - p_{jl}), \tag{3d}$$

$$\frac{\partial^2 E_{ij}}{\partial p_{ik} \partial p_{jl}} = 8(p_{ik} - p_{jk})(p_{jl} - p_{il}) \tag{3e}$$

其中 $i, j, k, l = 1, 2, 3$ 且 $k \neq l$。观察可得，

$$\frac{\partial^2 E_{ij}}{\partial p_{ik}^2} = -\frac{\partial^2 E_{ij}}{\partial p_{ik} \partial p_{jk}}, \tag{4a}$$

$$\frac{\partial^2 E_{ij}}{\partial p_{ik}\partial p_{il}} = -\frac{\partial^2 E_{ij}}{\partial p_{ik}\partial p_{jl}} \tag{4b}$$

设 H_{ij} 是 E_{ij} 的 hessian 矩阵，则 H_{ij} 是一个 6*6 的对称矩阵。由等式 (4a) 和 (4b) 可得 $H_{ij} = \begin{pmatrix} A_{ij} & -A_{ij} \\ -A_{ij} & A_{ij} \end{pmatrix}$，其中 A 是一个 3*3 的对称矩阵，即 $A_{ij} = 4$

$$\begin{pmatrix} d_{ij}+2(p_{i1}-p_{j1})^2 & -2(p_{i1}-p_{j1})(p_{i2}-p_{j2}) & -2(p_{i1}-p_{j1})(p_{i3}-p_{j3}) \\ -2(p_{i1}-p_{j1})(p_{i2}-p_{j2}) & d_{ij}+2(p_{i2}-p_{j2})^2 & -2(p_{i3}-p_{j3})(p_{i2}-p_{j2}) \\ -2(p_{i1}-p_{j1})(p_{i3}-p_{j3}) & -2(p_{i2}-p_{j2})(p_{i3}-p_{j3}) & d_{ij}+2(p_{i3}-p_{j3})^2 \end{pmatrix} \tag{3.1}$$

其中 $d_{ij} = |p_i - p_j\|^2 - l_{ij}^2$。

令 $H = \sum_{e_{ij}} H_{ij}$。由于 H_{ij} 不一定是一个正定矩阵，所以 H 也不一定是一个正定矩阵。因此我们需要做适当的处理使得 H 是一个正定矩阵。如果直接对 H 进行特征值分解后正定化，则算法复杂度为 $O(|E|^3)$，计算量巨大，因此不太合适。考虑到 $H = \sum_{e_{ij}} H_{ij}$，我们可以直接对 H_{ij} 进行正定化。这样只需要对 $|E|$ 个 6*6 的矩阵进行特征值分解，此时的算法复杂度为 $O(|E|)$，远小于之前的 $O(|E|^3)$ 且依然能保证 H 是一个正定矩阵。由于 H_{ij} 是一个对称矩阵，因此将 H_{ij} 特征值分解后有 $H_{ij} = Q\Lambda Q^T$。其中 Q 为正交矩阵，Λ 为实对角矩阵。接下来我们只需要令 $\widetilde{\Lambda} = \max(0, \Lambda)$ 并用 $\widetilde{\Lambda}$ 代替 Λ，那么就有 H_{ij} 为一个半正定矩阵。

但是由于每次我们迭代都需要重新计算 H 并将其正定化，是算法的主要运算量之一，因此我们希望能够进一步减少其计算量。注意到 $H_{ij} = \begin{pmatrix} A_{ij} & -A_{ij} \\ -A_{ij} & A_{ij} \end{pmatrix}$。这是由于边 e_{ij} 上的顶点 i 和顶点 j 对于对称位置。并且我们可以利用这一特殊性质进一步减少计算量。为此，我们先证明一定理。

定理 3.1　设 $H = \begin{pmatrix} A & -A \\ -A & A \end{pmatrix}$，则 H 是半正定矩阵的充分必要条件是 A 为半正定矩阵。

证明　充分性　若 A 是半正定矩阵，则对任意向量 $x \in R^n$ 有 $x^{\mathrm{T}}Ax \geqslant 0$。对任意向量 $y \in R^n$，设 $y = (y_1, y_2)$，其中 $y_1, y_1 \in R^n$。则有 $y^{\mathrm{T}}Hy = (y_1 - y_2)^{\mathrm{T}}A(y_1 - y_2) \geqslant 0$。因此 H 是半正定矩阵。

必要性　若 H 是半正定矩阵，则对任意向量 $y \in R^n$，设 $y = (y_1, y_2)$，其中 $y_1, y_1 \in R^n$ 有 $y^{\mathrm{T}}Hy \geqslant 0$。又因为 $y^{\mathrm{T}}Hy = (y_1 - y_2)^{\mathrm{T}}A(y_1 - y_2)$，因此 $(y_1 - y_2)^{\mathrm{T}}A(y_1 - y_2) \geqslant 0$。令 $y_2 = 0$，则有 $y_1^{\mathrm{T}}Ay_1 \geqslant 0$。又因为 y_1 是任意的，因此 A 是半正定矩阵。　∎

由上述事实可知我们只需令 A_{ij} 为半正定矩阵即可使得 H_{ij} 为半正定矩阵。因此我们可以像之前一样先将 A_{ij} 特征值分解，然后将中间的对角矩阵小于 0 的

元素改为 0，从而将其半正定化。考虑到特征值分解的计算量为 $O(n^3)$，通过这种操作，理论上可以将这一部分的计算量变为原来的 $\frac{1}{8}$。在实现时我们可以通过调用先有的矩阵计算库来实现，例如 *Eigen* 库。但是我们毕竟特征值分解的计算量为 $O(n^3)$，因此如果能不进行特征值分解直接得到最终结果那将进一步提高速度。

设

$$P_{ij}=\begin{pmatrix}(p_{i1}-p_{j1})^2 & -(p_{i1}-p_{j1})(p_{i2}-p_{j2}) & -(p_{i1}-p_{j1})(p_{i3}-p_{j3})\\ -(p_{i1}-p_{j1})(p_{i2}-p_{j2}) & (p_{i2}-p_{j2})^2 & -(p_{i3}-p_{j3})(p_{i2}-p_{j2})\\ -(p_{i1}-p_{j1})(p_{i3}-p_{j3}) & -(p_{i2}-p_{j2})(p_{i3}-p_{j3}) & (p_{i3}-p_{j3})^2\end{pmatrix},$$

则计算可得 P_{ij} 的三个特征值分别为 0,0 和 $(p_{i1}-p_{j1})^2+(p_{i2}-p_{j2})^2+(p_{i3}-p_{j3})^2$。

设 A_{ij} 的三个特征值分别为 $\lambda_1,\lambda_2,\lambda_3$。注意到 $A_{ij}=4(\|p_i-p_j\|^2-l_{ij}^2)I+8P_{ij}$，因此有

$$\lambda_1=4(\|p_i-p_j\|^2-l_{ij}^2), \tag{3.2}$$

$$\lambda_2=4(\|p_i-p_j\|^2-l_{ij}^2), \tag{3.3}$$

$$\lambda_3=12(p_{i1}-p_{j1})^2+12(p_{i2}-p_{j2})^2+12(p_{i3}-p_{j3})^2-4l_{ij}^2 \tag{3.4}$$

由于 A_{ij} 是对称矩阵，因此存在正交矩阵 Q 和对角阵 D 使得

$$A_{ij}=QDQ^T,$$

其中 $D=\mathrm{diag}(\lambda_1,\lambda_2,\lambda_3)$。设为 $Q=(x_1,x_2,x_3)$，则将上式展开可得

$$P_{ij}=\lambda_1x_1x_1^{\mathrm{T}}+\lambda_2x_2x_2^{\mathrm{T}}+\lambda_3x_3x_3^{\mathrm{T}}$$

由于 A_{ij} 是一个半正定矩阵等价于为 $\lambda_1,\lambda_2,\lambda_3\geqslant 0$, 因此我们只需让 $\widetilde{\lambda_k}=\max(\lambda_k,0)$,k=1,2,3，然后令 $\widetilde{A_{ij}}=\widetilde{\lambda_1}x_1x_1^{\mathrm{T}}+\widetilde{\lambda_2}x_2x_2^{\mathrm{T}}+\widetilde{\lambda_3}x_3x_3^{\mathrm{T}}$。则此时 $\widetilde{A_{ij}}$ 为半正定矩阵。由于 Q 是一个正交阵，即 $QQ^T=I$。因此有 $x_1x_1^{\mathrm{T}}+x_2x_2^{\mathrm{T}}+x_3x_3^{\mathrm{T}}=I$。并且由于 $\widetilde{\lambda_1}=\widetilde{\lambda_2}$, 因此有

$$\widetilde{A_{ij}}=\widetilde{\lambda_1}(I-x_3x_3^{\mathrm{T}})+\widetilde{\lambda_3}x_3x_3^{\mathrm{T}}$$

接下来我们只需计算 x_3。经过计算可得

$$x_3=\frac{(p_{i1}-p_{j1},p_{i2}-p_{j2},p_{i3}-p_{j3})}{\sqrt{(p_{i1}-p_{j1})^2+(p_{i2}-p_{j2})^2+(p_{i3}-p_{j3})^2}}$$

将其代入上式即可得到 $\widetilde{A_{ij}}$。接下来我们用 $\widetilde{A_{ij}}$ 替代 A_{ij} 则此时有 H_{ij} 为半正定矩阵，进而有 H 为半正定矩阵。通过这样的方法，我们可以跳过特征值分解的

步骤，直接得到半正定化后的 H_{ij}。相比之前特征值分解 $O(n^3)$ 的计算量, 此时的计算量变为 $O(n^2)$。考虑到 H_{ij} 是一个三阶矩阵，理论上这一部分的速度将会是原来的三倍。

由于 H 为半正定矩阵，并不能保证是一个正定矩阵，尽管在绝大多数情况下是一个正定矩阵。为了确保 H 是一个正定矩阵，在这里我们可以令 $\widetilde{H} = H + \epsilon I$, 其中 ϵ 为一很小的数，例如 $\epsilon = 10^{-15}$。

将 H 近似为正定矩阵后我们就得到了下降方向 $d = -H^{-1}\nabla E$。令 $E_l = f(x + \alpha d)$, 其中 x 是当前点的坐标，α 是步长。由于 $f(x + \alpha d)$ 是一个关于 α 的四次函数，因此我们可以对 $f(x + \alpha d)$ 求导，并令 α 为满足 $f(x + \alpha d)' = 0$ 的最小正实数。由于当 α 趋于 $+\infty$ 时，$f(x + \alpha d)$ 也趋于 $+\infty$，因此当 α 足够大时有 $f(x + \alpha d)'$>0。并且 $f(x + \alpha d)'$ 在 $\alpha = 0$ 处的导数小于 0，因此这样的 α 总是存在的。

3.2　插值

我们接下来要对源网格和目标网格的对应边长的平方进行线性插值以得到连续的变换。即对时间 $t \in (0, 1), l_t = \sqrt{(1-t) * l_1^2 + t * l_2^2}$, 其中 l_1, l_2 分别为源网格和目标网格的边长。这样插值后得到的结果如图 4.1 所示。可以看到中间的图显示出在 $t = 0.1$ 时刻存在插值结果不够光滑，自然的情况。

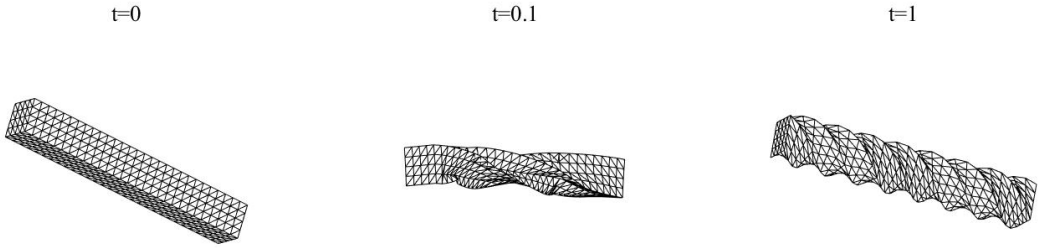

图 3.1　上图是两个木条之间的插值结果。其中左图和右图分别为初始网格和目标网格。中图是 $t = 0.1$ 时的结果。

这是因为只有边长的连续性无法保证得到自然的插值结果，表面可能不够光滑，因此还需要二面角的连续性。于是我们在之前的能量 E_l 的基础上加一项正则项，即令

$$\widetilde{E} = E_l + w\|P_{t_i} - P_i\|^2,$$

其中 t_i 为第 i 帧对应的时间 t，P_t 为 t 时刻所有点的坐标，P_i 为初始化网格，w 为权重。若 w 过大，则会使能量 E_t 项占比过小，使得难以较好的通过优化 $\widetilde{E}$ 来间接优化 E；若 w 过小，则图 4.1 中结果不够光滑的情况依旧会出现。经过测

试，我们最终取 $w = 10^{-3} l_{avg}^2$，其中 l_{avg} 为网格的平均边长。这个的 w 能在我们的测试模型中取得较好的结果。并且这样的 w 能够使得在一个模型放缩一定倍数时，输出的结果也放缩同样的倍数。通过加上这个正则项，只要初始化网格二面角具有连续性，则最终得到的结果的二面角也就具有连续性，

为了得到良好的插值结果，我们需要一个合适的初始化网格。考虑到我们的目标是优化边长，因此我们自然觉得初始化的边长误差越小，结果越好。于是我们准备使用 SQP[6] 这个已经对边长进行较好优化的方法进行初始化。为了验证我们的猜想，我们还是将其与 FFMP[33],ARAP[23],GE[24],ABF[19] 共 4 种方式得到的网格比较。首先，我们先定义一个评价标准

$$L = \sqrt{\frac{1}{|E|} \sum_{e \in E} (\frac{l_{real}}{l_{target}} - 1)^2},$$

其中 E 为所有边长的集合，l_{real} 是我们得到的网格的边长，l_{target} 是目标边长。可以看到实际上 L 就是边长的相对误差的 l_2 范数。

		$L(10^{-3})$						
		airplane	armadillo	bar	bird	bust	camel	cat
FFMP	t=0.25	1.39	0.47	12.42	0.13	3.45	0.96	3.46
	t=0.5	2.50	0.86	4.59	0.18	3.44	1.45	3.85
	t=0.75	1.75	0.62	8.50	0.13	3.44	0.90	2.62
ARAP	t=0.25	2.57	3.35	11.17	1.20	1.73	2.98	6.29
	t=0.5	3.58	4.17	7.46	1.50	2.43	2.21	5.51
	t=0.75	2.36	3.88	5.03	0.96	1.76	1.61	5.55
GE	t=0.25	0.89	0.14	8.50	0.12	1.84	0.43	2.10
	t=0.5	0.65	0.16	9.62	0.16	2.21	0.48	2.09
	t=0.75	0.53	0.15	7.26	0.18	1.93	0.36	1.22
ABF	t=0.25	0.90	0.14	10.81	0.17	1.90	0.63	9.30
	t=0.5	0.59	0.18	13.80	0.23	2.70	0.74	7.50
	t=0.75	0.45	0.16	7.74	0.18	1.98	0.54	3.45
SQP	t=0.25	0.24	0.12	9.83	0.08	1.04	0.33	1.02
	t=0.5	0.29	0.15	8.74	0.10	1.24	0.38	0.81
	t=0.75	0.20	0.13	5.56	0.07	1.00	0.31	0.43

表 3.1　在不同初始化网格下得到的结果的 L 值

如表 4.1 所示，在这五种初始化网格中，SQP 的 L 值更小，得到的结果的总体误差最小。图 4.2 则是多个模型在使用不同初始化网格下得到的 $t = 0.5$ 时刻的结果。从图 4.2 则可以看到 SQP 得到的结果更蓝，对应了其更小的 L 值。并且我们可以看到 SQP 得到的结果更加合理。因此，SQP 得到的结果最好，于是我们选择其作为初始化网格。并且这也确实说明了初始化网格的边长误差越小，结果越好。

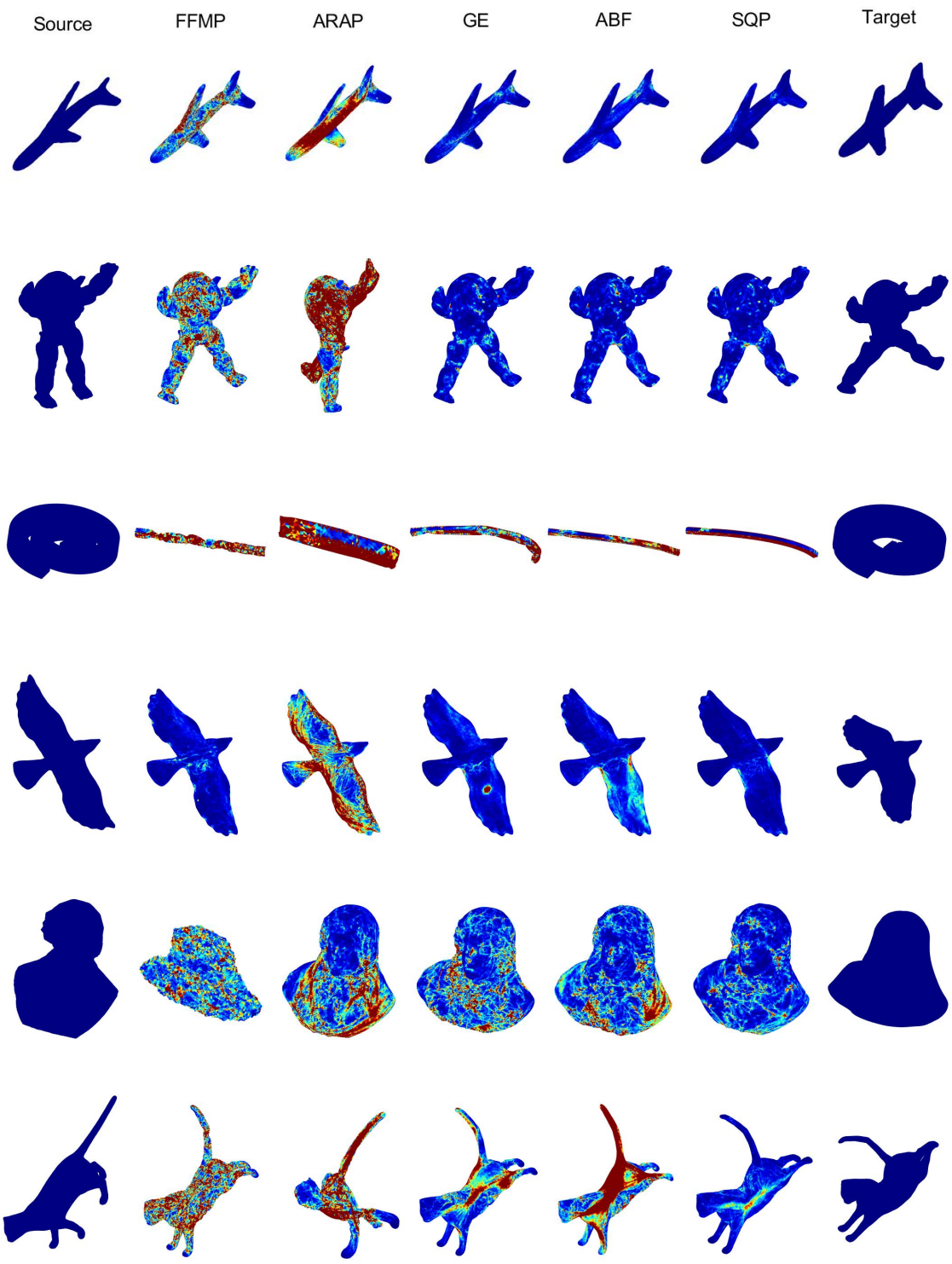

图 3.2　在不同初始化网格下 t=0.5 时得到的结果。从上往下分别是对应于表 4.1 的 fe 飞机 (airplane), 犰狳 (armadillo), 长条 (bar), 鸟 (bird), 半身像 (bust), 猫 (cat) 这 6 个模型。颜色越偏橙红说明实际边长与目标边长的差越大，越偏深蓝色则说明越小。

3.3　算法步骤

我们的算法的主要步骤总结如下所示:

输入：源网格和目标网格

输出：中间时刻 t 的网格

步骤 1:计算源网格和目标网格的边长 l_1 和 l_2,以及 t 时刻的目标边长 l_target。

步骤 2：计算权重 w。

步骤 3：使用 SQP 算法得到 t 时刻的初始化网格。

步骤 4：计算梯度 ∇E 以及此时的能量 E。

步骤 5：计算正交化之后的海塞矩阵 H。

步骤 6：计算能量关于梯度的导数的系数，然后求解步长 α。

步骤 7：更新网格顶点坐标并计算更新后的能量 E_{new}。

步骤 8：比较 E 和 E_{new}，若 E>1.01E_{new} 则重复迭代步骤 3-5，否则停止迭代。

3.4　实验结果

我们使用的系统为 Windows10，CPU 为 E5-2650v2，内存容量为 32GB。为了兼顾运行速度和简易性，所用语言为 matlab 和 C++。为了我们选取了 LSRDF[13]，ELI[5]和 MSGI[11]用于对比。

在实验中，我们使用的网格均是三维空间中的亏格为 0 的封闭网格。这是为了使自由度为 0。对于一个亏格为 0 的封闭三维网格，设其边，顶点和面数分别为 e, v, f。由欧拉公式可知 $f + v - e = 2$。另一方面，假设在网格内部和在边界的边数分别为 e_1 和 e_2, 则有 $e_1 + e_2 = e$ 和 $3f = 2e_1 + e_2$。因此在已知所有边长的情况下，其自由度为 $3v - e = 3(e - f + 2) - e = e_2 + 6$。最后再减去刚性变换的 6 个自由度，则自由度为 e_2，恰好等于网格边界边数。因此为了使结果在理论上具有唯一性，之后我们使用封闭网格当做实验输入。

图 4.3 展示了在 $t = 0.25, 0.5, 0.75$ 时在一些模型下的插值结果。可以看到，这些模型的插值结果是自然且符合直觉的。从最后一列可以看出，输出的网格的边长与期望边长的相对误差大约在 $10^{-}4$ 这一数量级。而在图 4.4 和图 4.5 则进一步展示了我们的方法与选取的三种方法的插值结果的比较。在图 4.4 中，我们的方法得到的网格几乎都是深蓝色的，即边长误差小；而其它三种方法生成的网格则有部分甚至全部都是橙色的，即边长误差大。因此从图 4.4 可以看到我们的方法在这几个模型下的边长误差均是最小的，且整体上误差相近。从图 4.5 左列则可以看到我们的方法的 L 值更小, 即总体上来说我们的方法在各个形变的过程中边长误差均是最小的。并且，除了 Bar 这一模型外，对于剩下四个模型，我们

的方法的边长误差小于其它方法至少一个数量级。这而从图 5 右列则可以看到除了在第一个模型，即 bar 这个模型的某些时刻外，我们的方法在各个形变的过程中最大的相对误差均是最小的，并且也小于其它方法一个数量级。所以无论是从图 4.4，还是从图 4.5 来看，我们的方法均是最好的。

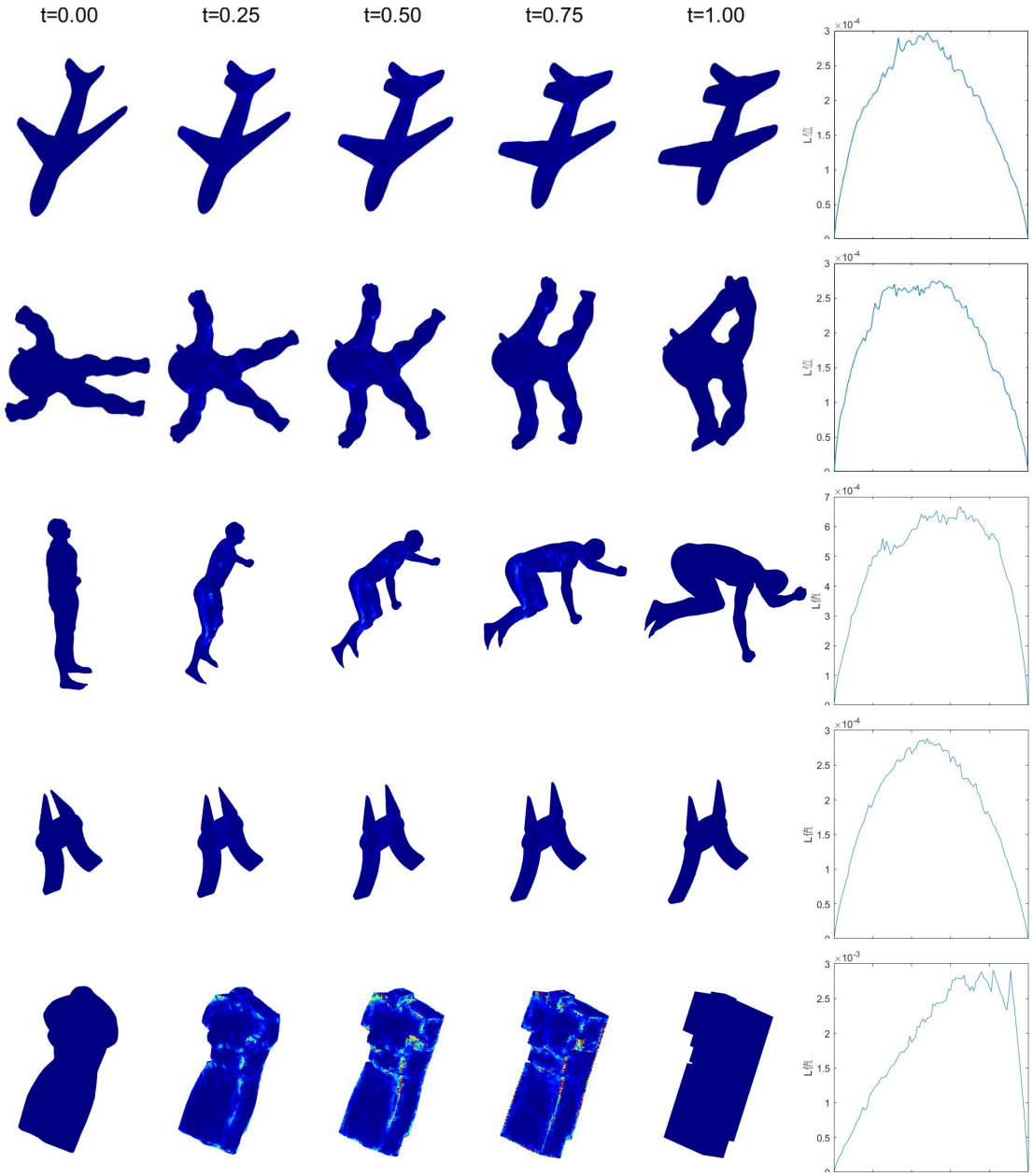

图 3.3　飞机、犰狳、人、钳子和半身像这五个模型的插值结果。其中 t=0.00 和 t=1.00 分别代表源模型和目标模型，其它时刻则为插值得到的结果。最右边是不同时刻的 L 值误差。

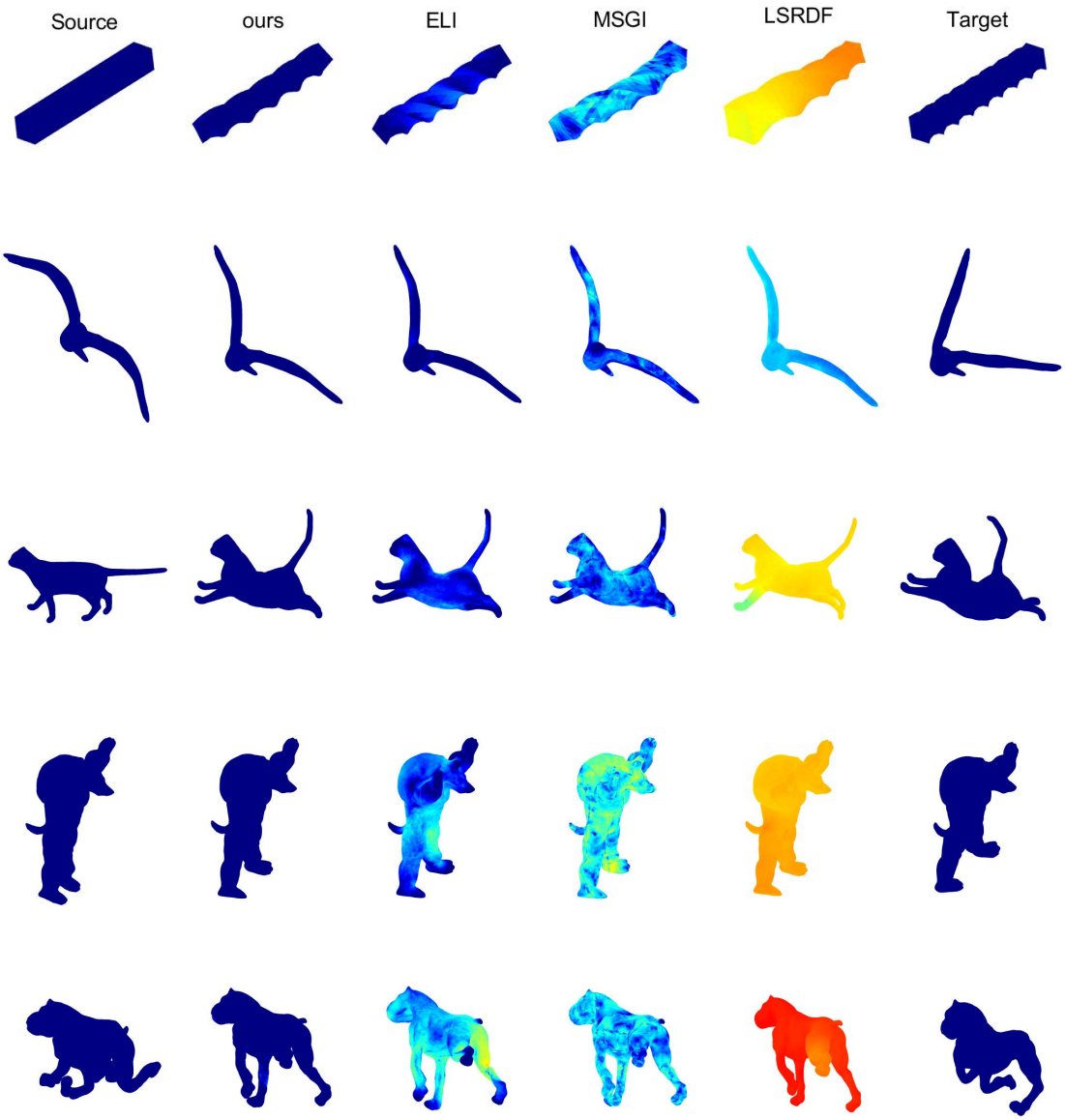

图 3.4　比较四种方法在五个模型下的插值结果。其中中间四列是四种方法在 $t = 0.5$ 时刻的结果。颜色越偏橙红说明实际边长与目标边长的差越大，越偏深蓝色则说明越小。

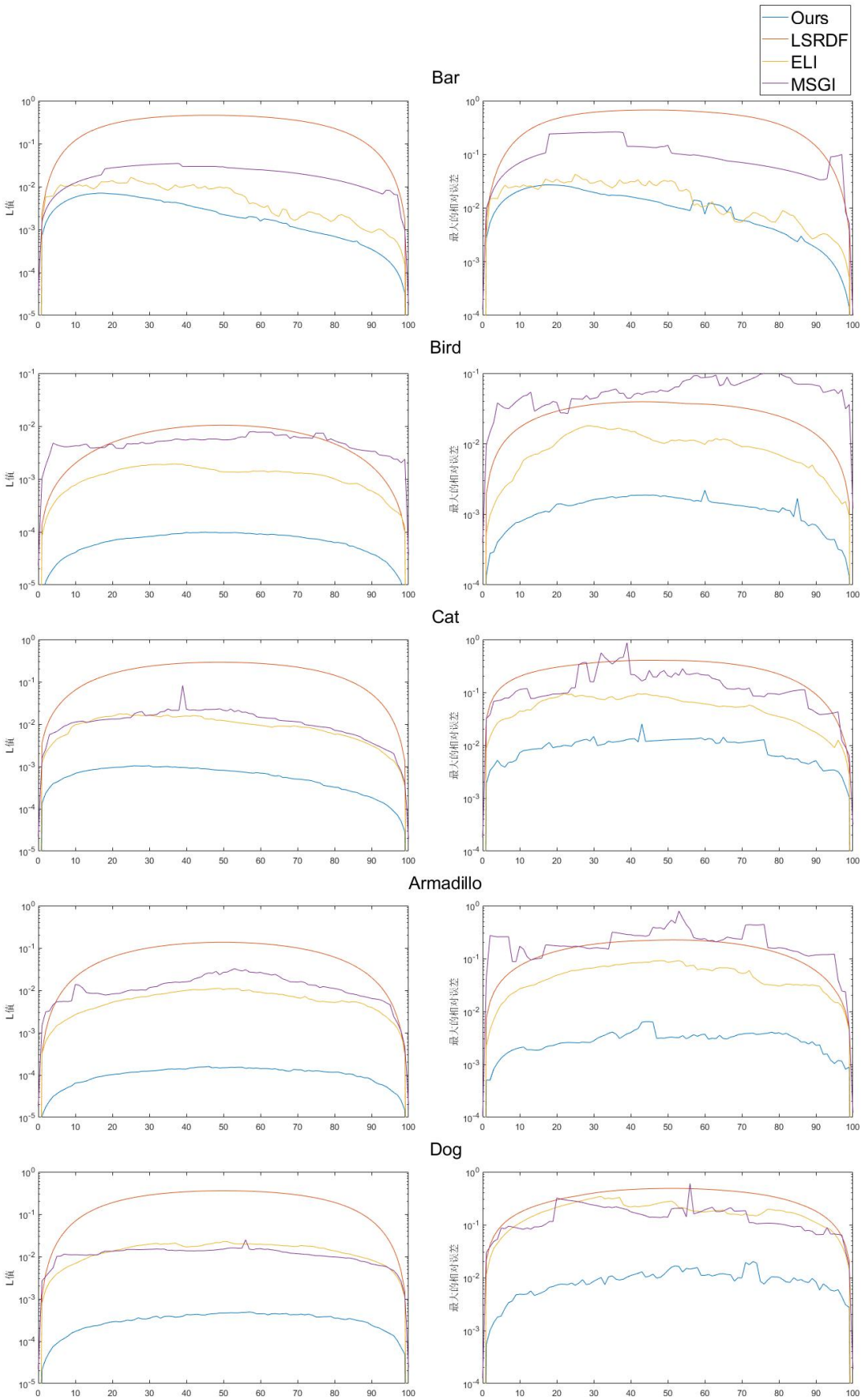

图 3.5　比较四种方法在图三的五个模型下的误差。横轴代表帧数，其中第 0 帧和第 100 帧分别代表源模型和目标模型。左列是四种方法下的 L 值，右列则是最大的相对误差，其中相对误差为 $|\frac{l_{real}-l_{target}}{l_{target}}|$。

第 4 章　总结与展望

4.1　本文总结

形状插值问题一直是计算机图形学的热门问题，并且随着近年来动画行业的快速发展，该问题将愈发重要。本文提出了一种新的基于边长的三维曲面插值算法。在已有的相似算法中，往往由于同时插值二面角和边长，导致自由度不足。最终导致其输出曲面的边长与我们的期望会有较大差距。在这些算法里，与我们最接近的是 ELI[5]，均只使用了边长信息。但是由于我们的方法不需要像 ELI 一样进行多次转换得到最终结果，因此能获得更小的误差。而在另一方面，由于加上了与初始位置之差这一正则化项，因此在初始化较好的情况下，得到的结果依然自然且符合直觉的。从实验也可以看出，我们的方法得到的结果，其边长相比其它方法更加接近我们的期望，误差更小。

4.2　未来展望

不过该方法也有其局限性。其局限性及未来的发展方向主要有两个：

1. 由于正则项包含初始化网格的信息，因此结果的好坏一定程度上依赖于初始化的好坏。当初始化不好时，该方法也难以得到较好的结果。针对这个问题，我们可以尝试寻找并使用更好且更稳定的初始化算法，以此保证该方法尽可能的得到好的结果。甚至于我们可以采取多个初始化算法，并且对于不同模型选择对应的最优算法。

2. 虽然总体上的边长的相对误差相较于其它方法有了较大减少，但是最大的边长的相对误差并没有保证，因此在少数情况下该指标不如其它方法。而且我们最终结果的边长依然与我们期望的长度有一定差距。对此，首先我们可以考虑是否有更好的优化方法并结合更好的初始化使得最终结果的边长等于我们期望的长度。其次，我们可能考虑将最大的边长的相对误差加入到能量中，或者将之作为限制条件，以此来进一步的减少最大的边长的相对误差。

参 考 文 献

[1] 金小刚, 鲍虎军. 计算机动画技术综述[J]. 软件学报, 1997, 8(4): 11.

[2] 周谦. 计算机动画关键帧插补技术综述[J]. 电脑知识与技术：学术版, 2007(1): 2.

[3] CHEN L, DIPAOLA S. An analysis of the current and future state of 3d facial animation techniques and systems[J]. simon fraser university, 2009.

[4] 张智邦, 李桂清, 韦国栋, 等. 形状插值算法综述[J]. 计算机辅助设计与图形学学报, 2015, 27(8): 12.

[5] ROJAS C, TSUI A, HE S, et al. Edge length interpolation[C]//ACM Symposium on Solid and Physical Modeling. 2014.

[6] AHARON I, CHEN R, ZORIN D, et al. Bounded distortion tetrahedral metric interpolation [J]. ACM Transactions on Graphics, 2019, 38(6): 182.1-182.17.

[7] EULER. Recherches sur la courbure des surfaces[Z]. 1766.

[8] CAUCHY A L. Sur les polygones et polyedres[J]. J. Ec. Polytechnique, 1813, 16: 87-99.

[9] GLUCK H. Almost all simply connected closed surfaces are rigid[M]. Geometric Topology, 1975.

[10] SEDERBERG T W, GAO P, WANG G, et al. 2-d shape blending: An intrinsic solution to the vertex path problem[C]//Conference on Computer Graphics and Interactive Techniques. 1993.

[11] WINKLER T, DRIESEBERG J, ALEXA M, et al. Multi-scale geometry interpolation[J]. Computer Graphics Forum, 2010, 29(2).

[12] WILLIAMS J A, BENNAMOUN M. Simultaneous registration of multiple point sets using orthonormal matrices[C]//IEEE International Conference on Acoustics. 2000.

[13] WANG Y, LIU B, TONG Y. Linear surface reconstruction from discrete fundamental forms on triangle meshes[C]//Computer Graphics Forum: volume 31. Wiley Online Library, 2012: 2277-2287.

[14] PINKALL U, POLTHIER K. Computing discrete minimal surfaces and their conjugates[J]. Experimental mathematics, 1993, 2(1): 15-36.

[15] FRÖHLICH S, BOTSCH M. Example-driven deformations based on discrete shells[C]// Computer graphics forum: volume 30. Wiley Online Library, 2011: 2246-2257.

[16] KILIAN M, MITRA N J, POTTMANN H. Geometric modeling in shape space[J]. Acm Transactions on Graphics, 2007, 26(3): 64.

[17] BOTSCH M, SUMNER R, PAULY M, et al. Deformation transfer for detail-preserving surface editing[C]//Vision, Modeling & Visualization. Citeseer, 2006: 357-364.

[18] SUMNER R W, POPOVIĆ J. Deformation transfer for triangle meshes[J]. ACM Transactions

on graphics (TOG), 2004, 23(3): 399-405.

[19] PAILLé G P, RAY N, POULIN P, et al. Dihedral angle-based maps of tetrahedral meshes[J]. ACM Transactions on Graphics (SIGGRAPH 2015 Conf. Proc.), 2015, 34(4).

[20] BARRETT J W. First order regge calculus[J]. Classical and Quantum Gravity, 1994, 11(11): 2723.

[21] CHIEN E, LEVI Z, WEBER O. Bounded distortion parametrization in the space of metrics [J]. ACM Transactions on Graphics (TOG), 2016, 35(6): 1-16.

[22] BONNANS J F, GILBERT J C, LEMARÉCHAL C, et al. Numerical optimization: theoretical and practical aspects[M]. Springer Science & Business Media, 2006.

[23] ALEXA M. As-rigid-as-possible shape interpolatioin[C]//SIGGRAPH2000 Conference Proceedings. 2000.

[24] CHAO I, PINKALL U, SANAN P, et al. A simple geometric model for elastic deformations [J]. ACM transactions on graphics (TOG), 2010, 29(4): 1-6.

[25] GOTSMAN C, LIU L, ZHANG L, et al. A local/global approach to mesh parameterization[J]. Computer Graphics Forum, 2008, 27(5): 1495-1504.

[26] LÉVY B, PETITJEAN S, RAY N, et al. Least squares conformal maps for automatic texture atlas generation[J]. ACM transactions on graphics (TOG), 2002, 21(3): 362-371.

[27] GAO L, LAI Y K, HUANG Q X, et al. A data-driven approach to realistic shape morphing [C]//Computer graphics forum: volume 32. Wiley Online Library, 2013: 449-457.

[28] FREY B J, DUECK D. Clustering by passing messages between data points[J]. science, 2007, 315(5814): 972-976.

[29] YU Y, ZHOU K, XU D, et al. Mesh editing with poisson-based gradient field manipulation [M]//ACM SIGGRAPH 2004 Papers. 2004: 644-651.

[30] GAO L, CHEN S Y, LAI Y K, et al. Data-driven shape interpolation and morphing editing [C]//Computer Graphics Forum: volume 36. Wiley Online Library, 2017: 19-31.

[31] GARLAND M, HECKBERT P S. Surface simplification using quadric error metrics[C]// Proceedings of the 24th annual conference on Computer graphics and interactive techniques. 1997: 209-216.

[32] HOPPE H. Progressive meshes[C]//Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. 1996: 99-108.

[33] KIRCHER S, GARLAND M. Free-form motion processing[J]. Acm Transactions on Graphics, 2008, 27(2).

致　谢

时光如白马过隙，转眼间四年的研究生时光快要结束了。在这宝贵的研究生三年中，老师、学长、同学、亲人均给与我巨大帮助。

首先我要感谢我的导师刘利刚教授和陈仁杰教授。刘老师将我带入了计算机图形学领域，让我领略了其魅力。刘老师他尽职尽责的态度、幽默风趣的谈吐、渊博的学识、废寝忘食的研究热情均深深地激励了我。而陈老师在这三年里一直悉心指导我，在学习和科研方面均给予我极大的帮助，使我能够有能力撰写毕业论文并得以顺利毕业。

我还要感谢我的室友明水根和李常颢。在本科和研究生共七年的生活中大家互相帮助、互相体谅，相处得很愉快。在学习中大家为我解决了很多问题，使我受益良多。

在读期间发表的学术论文与取得的研究成果

已发表论文

1. A A A A A A A A A A
2. A A A A A A A A A A
3. A A A A A A A A A A

待发表论文

1. A A A A A A A A A A
2. A A A A A A A A A A
3. A A A A A A A A A A

研究报告

1. A A A A A A A A A A
2. A A A A A A A A A A
3. A A A A A A A A A A