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： 二〇二二年三月二十一日

University of Science and Technology of China

A dissertation for master's degree

3D Mesh Interpolation Based on Edge Length

Author: Liu Zhenye

Speciality: Computational Mathematics

Supervisors: Prof. Ligang Liu, Prof. Renjie Chen

Finished time: March 21, 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，能将一个几何模型光滑的变换到目标模型。其被广泛应用于计算机动画等领域。该问题已有几十年的研究历史，相关工作非常多。其中基于边长进行插值的方法占有相当一部分。

我们提出了一种基于边长的插值方法。其在两个兼容（具有相同连接关系）的三维网格之间插值。该方法主要有以下几个优点。首先，一般的插值边长的形状插值算法还往往需要同时插值其它几何量，例如二面角。而我们只使用了边长作为插值量，因此得到的结果与期望边长的误差更小。而其它只利用边长作为插值量的算法也因为需要先计算出二面角再进行间接重建，导致边长误差变大，也不如我们的方法。其次，与一般线性插值边长的做法不同，我们线性插值的是边长平方。因为有文章已经证明了这样能保证插值结果是有界扭曲的。不过该文章是应用在四面体网格上的。因此我们将其迁移应于三维网格上。并且我们将其结果的表面作为初始化网格可以取得快速收敛的效果。最后，我们使用牛顿法对能量进行优化，并且利用多种技巧将海森矩阵快速近似为正交矩阵，避免了通常需要的特征值分解这一步骤。同时因为直接以顶点位置作为变量，不需要间接重建。因此相比于已有的只基于边长的插值方法，该方法在速度方面也更快。

关键词：计算机图形学；形状插值；变形

ABSTRACT

Shape interpolation is a very important computer graphics problem, which can smoothly transform a geometric model to the target model. It is widely used in computer animation and other fields. This problem has been studied for decades, and there is a lot of related work. The interpolation method based on edge lengths occupies a considerable part.

We propose a shape interpolation method based on edge lengths. It interpolates between two compatible(with the same connection relationship) 3D meshes. This method has the following advantages. First, the shape interpolation algorithm that interpolates the edge lengths usually needs to interpolate other geometric quantities at the same time, such as dihedral angles.Instead,we only use edge lengths as the interpolation quantity. So the error between the result and the expected edge lengths is smaller. Other algorithms that only use the edge lengths as the interpolation quantity need to calculate the dihedral angles first and then reconstruct the mesh indirectly, resulting in a larger edge lengths error.So the result is not as good as our method. Secondly, unlike the general linear interpolation method of edge lengths, we linearly interpolate the square of the edge lengths.Because there is an article that proves that this can ensure that the interpolation result has bounded distortion.This article is applied to tetrahedral meshes. So we apply it to 3D meshes. And we take the surface of the result as the initial mesh to achieve rapid convergence. Finally, we use Newton's method to optimize the energy, and use several tricks to quickly approximate the Hessian matrix with an orthogonal matrix. This avoids the normally required step of eigenvalue decomposition.At the same time, the vertex position is directly used as a variable. So there is no need for indirect reconstruction. Therefore, compared with the existing interpolation methods based only on edge lengths, this method is also faster in speed.

Key Words: Computer Graphics; Shape Interpolation; Morphing

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，存在一个双射 $f(x): S_0 \to S_1$。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。其是为了度量面网格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，即局部是一个单射。

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。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[1-3]。近些年来，中国的动画产业在飞速发展。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。而制作动画的软件，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。

如之前所说的，衡量一个形状插值算法的好坏的一个重要标准是插值结果是否是有界扭曲的。Chen 等[5]提出了一种有界扭曲的四面体网格插值算法。该算法的有界扭曲既是等距变换意义上的，又是共形变换意义上的。其主要的理论

依据是有界的共形扭曲和等距扭曲所构成的回拉度量 (pullback metric) 空间是凸的[6]。而回拉度量可以写成边长平方的线性表示。因此可以通过线性插值边长的平方来从理论上保证是有界扭曲的。还有虽然线性插值边长平方后得到的四面体网格由于内边的邻二面角和一般不等于 2π，不能直接嵌入到 R^3 空间中，但可以证明其与 2π 的差是有界的且实验显示其很接近 2π。因此最后得到的插值结果的边长与目标边长相差不大。

受此启发，我们也可以线性插值边长的平方，将其迁移应用到三维网格中。由于其已经是一个很好的线性插值了边长平方的算法，并且体网格表面也是一个三维网格，因此我们将其作为初始化网格可以获得快速收敛的效果。本方法就是用其初始化顶点坐标。

形状插值问题的相关工作非常多，其中插值边长的方法占有相当一部分。不过这些方法往往需要同时插值其它几何量，如文章[7-8]均是同时插值了边长与二面角，文章[9]则同时插值了边长，二面角与体积。但这些方法的边长误差均较大。事实上，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。因此当各边边长确定时，自由度恰好为 0。那么若还需要同时插值其它几何量，自由度不足，那么边长误差自然较大。而只插值边长的方法则比较少，近些年只有文章[10]。该方法的一个不足是为了结果的鲁棒性分了两步进行重建，导致边长误差在重建过程中放大。为解决这一问题，我们直接使用了顶点位置作为变量定义能量。

因此，我们的方法既避免文章[10]的缺点，直接使用顶点坐标作为变量，省去了重建过程，能得到边长误差更小的结果。而且，像文章[5]一样，插值的是边长平方，能够从理论上保证插值序列是有界扭曲的。

1.2　章节安排

本文的章节安排主要所示：

第一章是绪论。这章主要介绍了形状插值问题及研究背景、思路。

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有涉猎。其中主要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的工作，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[11]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[8]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。

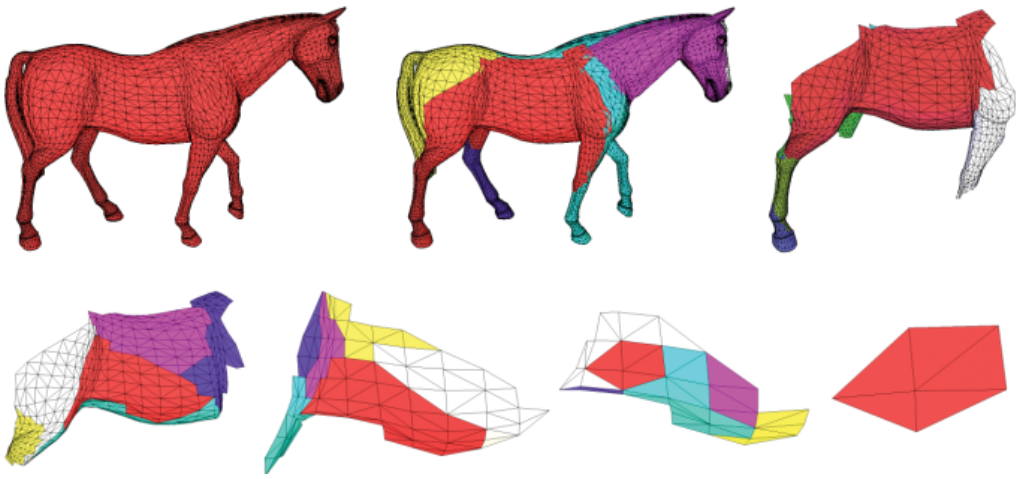

图 2.1　将左上角的网格分解成多个层级[8]

第二步则需要把重建好的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 $\boldsymbol{P}^{[k]}$ 表示第 k 个小面片，$\boldsymbol{P}$ 代表大面片。如图2.2所示，假设 $v_i^{[j]}$ 代表顶点 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 $k = 1, \cdots, 6$。

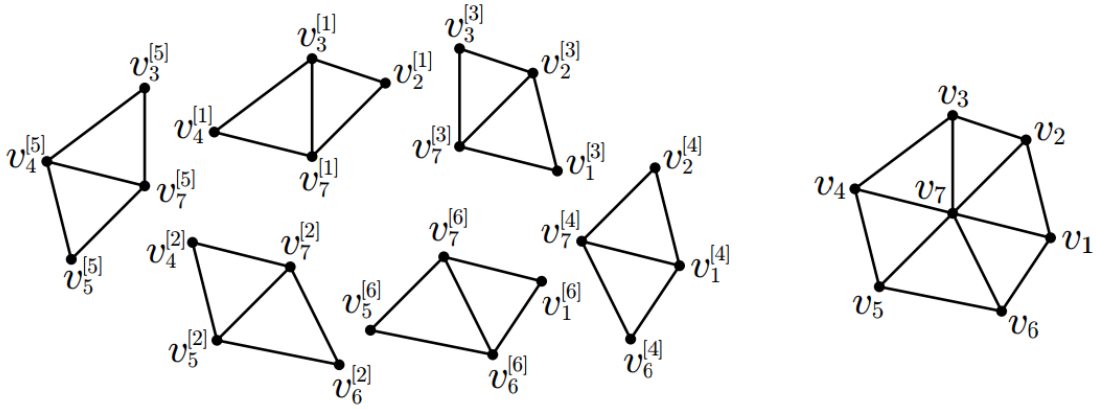

图 2.2　将六个面片 $P^{[1]},\cdots,P^{[6]}$ 合成一个大的环状面片 $P^{[8]}$

对于所有这些等式，可以将之表示成矩阵形式，即

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

其中向量 $\boldsymbol{v}$ 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，其元素为 1 或者-1；$\boldsymbol{e}$ 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

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

通过求解上式，我们得到了 $\boldsymbol{v}$，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更接近期望值，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[7]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 分别是网格相邻面 i 和面 j 上的正交坐标架。因此，该方法定义该部分能量：

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

其中 $\boldsymbol{E}$ 是边的集合，w_e 是余切权重[13]。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：

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

其中 v_m,v_n 是相邻两个顶点，$\boldsymbol{x_m},\boldsymbol{x_n}$ 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 $\boldsymbol{T}$ 上的局部坐标系，$a^1_{mn,T},a^2_{mn,T}$ 则分别是边 e 在局部坐标系 f_T 下的 x 和 y 坐标。最后定义总的能量：

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

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

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 $\boldsymbol{x}$ 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[8]提出的方法。第四章的图3.4和图3.5展示了这一点。

Chen 等[9]同时使用边长、体积、二面角进行形状插值。因此其设能量为

$$\begin{aligned}E_s&=\sum_e(l_e-l_e^*)^2\\E_b&=\sum_e(\theta_e-\theta_e^*)^2\\E_v&=(v-v^*)^2\end{aligned} \tag{2.8}$$

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

$$E=\lambda E_s+\mu E_b+\upsilon E_v \tag{2.9}$$

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 $M_1,\cdots,M_k$，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 $\boldsymbol{M}$。接着，该方法令式2.8中的各目标参数为

$$\begin{aligned}l_e^*&=L_e+\sum_{i=1}^k\alpha_i(L_e^{(i)}-L_e)\\\theta_e^*&=\Theta_e+\sum_{i=1}^k\alpha_i(\Theta_e^{(i)}-\Theta_e)\\v_e^*&=V_e+\sum_{i=1}^k\alpha_i(V_e^{(i)}-V_e)\end{aligned} \tag{2.10}$$

其中 α_i 为权重。当 $k=1$ 且令 $\alpha_1=t$ 时，其中 $t\in[0,1]$ 为时间，则此时式2.10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 [14-15] 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 [16] 中的方法得到最终结果 $\boldsymbol{M}'$。

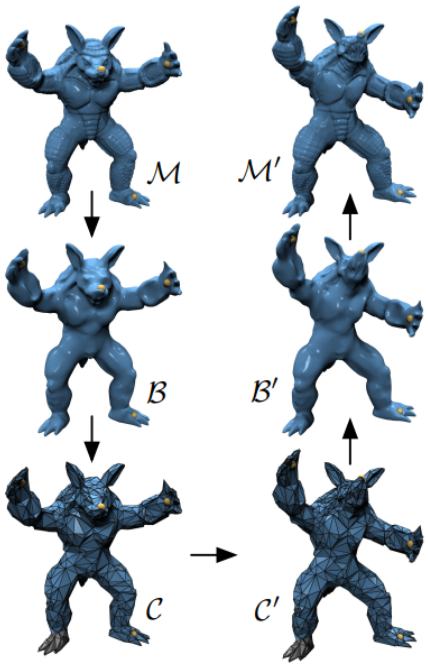

图 2.3　多分辨率优化方案[9]。左边一列是初始模型，右边一列是处理后的模型。上面一行是高分辨率模型，中间一行是光滑模型，下面一行是低分辨率模型

相比于之前的几种方法都利用了二面角，Rojas 等[10]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 $\boldsymbol{X}_i$，围绕自身 z 轴旋转对应的旋转矩阵为 $\boldsymbol{Z}_i$，其中 $k=1,2,\cdots,k$。则由于当瓢虫旋转一周时，坐标架也回到初始状态，因此有 $\boldsymbol{Z}_1\boldsymbol{X}_1\boldsymbol{Z}_2\boldsymbol{X}_2\cdots\boldsymbol{Z}_k\boldsymbol{X}_k=\boldsymbol{I}$。其中 $\boldsymbol{I}$ 是单位矩阵. 因此该方法定义顶点 v 处的能量：

$$E_v=\| \boldsymbol{I}-\boldsymbol{Z}_1\boldsymbol{X}_1\boldsymbol{Z}_2\boldsymbol{X}_2\cdots\boldsymbol{Z}_k\boldsymbol{X}_k \|_F \tag{2.11}$$

其中 F 代表 Frobenius 范数。

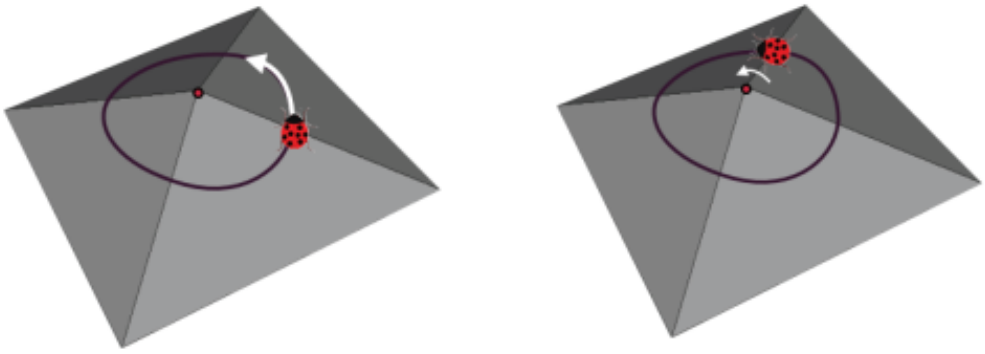

图 2.4　瓢虫围绕一个顶点旋转[10]

对于网格上的所有点，定义总的能量：

$$E = \sum_{v} E_v \tag{2.12}$$

由于中间时刻 t 的边长是插值得到的，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 $\boldsymbol{Z_i}$ 完全由角度决定，因此矩阵 $\boldsymbol{Z_i}$ 是已知的。而矩阵 $\boldsymbol{X_i}$ 只由相邻两面的二面角 α_i 决定，因此式2.12是一个以二面角 α_i 为变量的能量。于是该方法接下来使用拟牛顿法优化能量 E 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 处的局部坐标系与全局坐标系之间的旋转矩阵 $\boldsymbol{G}_v$。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 $\boldsymbol{M}_{v,u}$ 使得

$$\boldsymbol{G}_v\boldsymbol{M}_{v,u} - \boldsymbol{G}_u = 0 \tag{2.13}$$

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 $\boldsymbol{G}_v$。不过这样得到的 $\boldsymbol{G}_v$ 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 $\boldsymbol{G}_v$ 重建最终网格。对于相邻顶点 u, v，有

$$v + \boldsymbol{G}_v u_v - u = 0 \tag{2.14}$$

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[8]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间接重建才得到最终的网格，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[17]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。该方法首先提出了四面体网格的二面角应满足的三个条件。

第一，由格林公式，易得 $\sum_i a_i \boldsymbol{n}_i = 0$，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

$$\begin{pmatrix} 1 & -\cos\theta_{01} & -\cos\theta_{02} & -\cos\theta_{03} \\ -\cos\theta_{01} & 1 & -\cos\theta_{12} & -\cos\theta_{13} \\ -\cos\theta_{02} & -\cos\theta_{12} & 1 & -\cos\theta_{23} \\ -\cos\theta_{03} & -\cos\theta_{13} & -\cos\theta_{23} & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.15}$$

我们将左边的 4 阶矩阵定义为 $\boldsymbol{G}_t$，并且易得

$$\det(\boldsymbol{G}_t) = 0 \tag{2.16}$$

设 $\boldsymbol{C}_t$ 是 $\boldsymbol{G}_t$ 的余子式，并且可证明其应为正定矩阵[18]，即 $\boldsymbol{C}_t > 0$。

第二，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，则应满足

$$\cos\phi_{ij}^0 = \cos\phi_{ij}^1, j = 0,1,2 \tag{2.17}$$

其中 j 代表面 i 的第 j 个角。

第三，对于四面体网格内部的一条边 e，其相邻四面体对应的二面角 θ 之和应为 2π，即

$$\sum_{\theta \in \Theta_e} \theta = 2\pi \tag{2.18}$$

其中 Θ_e 是边 e 周围的所有的二面角。

接着，我们定义能量函数。如图2.5所示，我们定义

$$w = \theta_0 + \theta_1 + \theta_2 - \pi \tag{2.19}$$

从角度出发，我们定义能量函数为

$$f(\Theta) = \| \boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}} \|^2 + \| \boldsymbol{\Omega} - \hat{\boldsymbol{\Omega}} \|^2 \tag{2.20}$$

其中 $\boldsymbol{\Theta}$ 是包含所有二面角的向量，$\hat{\boldsymbol{\Theta}}$ 则是包含所有的目标二面角的向量；$\boldsymbol{\Omega}$ 和 $\hat{\boldsymbol{\Omega}}$ 则是使用式2.19从 $\boldsymbol{\Theta}$ 和 $\hat{\boldsymbol{\Theta}}$ 得到的。$\| \boldsymbol{\Omega} - \hat{\boldsymbol{\Omega}} \|^2$ 这一项可以有效的防止尖锐、

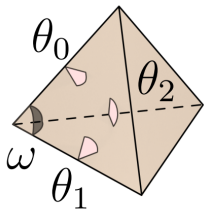

图 2.5　四面体[17]

细长的四面体出现，令插值结果更加平滑。结合之前的三个限制条件和能量函数可以定义优化问题为

$$\begin{aligned} \min \quad & f(\boldsymbol{\Theta}) \\ \text{s.t.} \quad & \det(\boldsymbol{G}_t) = 0 \wedge \boldsymbol{C}_t > 0 \qquad \forall t \in \boldsymbol{T} \\ & \phi_c^0 = \phi_c^1(\text{式}2.17) \qquad \forall c \in \boldsymbol{C} \\ & \sum_{\theta \in \Theta_e} \theta = 2\pi \qquad \forall e \in \boldsymbol{E} \\ & \theta_b \leqslant \theta \leqslant \pi - \theta_b \qquad \theta \in \boldsymbol{\Theta} \end{aligned} \tag{2.21}$$

其中 $\boldsymbol{T}$ 是所有的三角形；$\boldsymbol{C}$ 是所有的四面体的三角形面的角；Θ_e 是边 e 周围的所有的二面角；$\boldsymbol{E}$ 是所有的边；θ_b 是一个正数，用于限制二面角的范围；$\boldsymbol{\Theta}$ 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

$$E(\boldsymbol{\Theta}) = \alpha f(\boldsymbol{\Theta}) + \| c(\boldsymbol{\Theta}) \|^2 + \| \min(d(\boldsymbol{\Theta}), 0)) \|^2 \tag{2.22}$$

其中 α 是一个常数，该方法设 $\alpha = 10^{-6}$；$c(\boldsymbol{\Theta})$ 代表式 2.21中的所有等式限制条件，$d(\boldsymbol{\Theta})$ 代表式2.21中的所有不等式限制条件。该方法将线性插值得到的 $\boldsymbol{\Theta}$ 作为其初始化，然后使用拟牛顿法进行优化，得到了所有二面角 $\boldsymbol{\Theta}$。

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

$$l_{ij} = b_t \frac{\partial |\boldsymbol{G}_t|}{\partial \theta_{ij}} \tag{2.23}$$

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 [18] 有式2.23的证明。

第二步则是全局重建。首先如图2.6所示，$\boldsymbol{v}_4$ 的关系可以表示为

$$\begin{aligned} & \boldsymbol{v}_4 = a_0\boldsymbol{v}_0 + a_1\boldsymbol{v}_1 + a_2\boldsymbol{v}_2 + a_3\boldsymbol{v}_3 \\ & \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} \boldsymbol{v}_0 & \boldsymbol{v}_1 & \boldsymbol{v}_2 & \boldsymbol{v}_3 \\ 1 & 1 & 1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{v}_4 \\ 1 \end{pmatrix} \end{aligned} \tag{2.24}$$

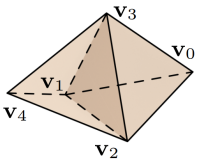

图 2.6　两个相邻的四面体[17]

其中 $\boldsymbol{v}_i$ 是顶点坐标，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2.24写成一个线性系统，得到 $\boldsymbol{A}\boldsymbol{x}=0$。其中 $\boldsymbol{A}$ 的各项为 a_i，$\boldsymbol{x}$ 则是所有顶点坐标。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 $\boldsymbol{x}$。

该方法主要有两点不足。第一是能量函数非线性非凸，难以优化。第二是该方法不能保证插值结果全局不自交。

Chen 等[19]提出了一种应用于平面网格的有界共形扭曲算法。一般插值边长的算法都是线性插值边长，但是该方法则是线性插值边长的平方。这主要是因为其发现了这样能保证插值结果是有界共形扭曲的。

因此对于时间 t，令目标边长为

$$l_t=\sqrt{(1-t)l_1^2+tl_2^2} \tag{2.25}$$

其中 l_1 和 l_2 分别为源模型和目标模型的边长。

接着便使用算法[20]将其共形映射回二维平面。该方法考虑两个黎曼度量 g 和 $\hat{g}$ 在光滑情形下若满足 $\hat{g}=\mathrm{e}^{2u}g$ 则是共形等价的，其中 u 是光滑函数，便将其离散化。即若两个离散度量 l 和 $\hat{l}$ 满足 $\hat{l}_{ij}=\mathrm{e}^{(u_i+u_j)/2}$，则称它们是（离散）共形等价的。其中 u_i 是定义在顶点 v_i 的常数值，v_i 和 v_j 则是相邻点。设 $\lambda_{ij}=2\ln l_{ij}$，则上式可写为

$$\hat{\lambda}_{ij}=\lambda_{ij}+u_i+u_j \tag{2.26}$$

给定一个三角形 $t_{ijk}\in T$ 和对应的边长 l_{ij},l_{jk},l_{ki}，可计算出在点 v_i 处的角为 α_{jk}^i。同理，对于 $\hat{l}_{ij}$，也有对应的 $\hat{\alpha}_{jk}^i$。

设能量为

$$\begin{aligned}E(u)&=\sum_{t_{ijk}\in T}(f(\hat{\lambda}_{ij},\hat{\lambda}_{jk},\hat{\lambda}_{ki}))-\frac{\pi}{2}(u_i+u_j+u_k)+\frac{1}{2}\sum_{v_i\in V}\hat{\Theta}_i u_i\\ f(\hat{\lambda}_{ij},\hat{\lambda}_{jk},\hat{\lambda}_{ki})&=\frac{1}{2}(\hat{\alpha}_{jk}^i+\hat{\alpha}_{ki}^j+\hat{\alpha}_{ij}^k)+\Pi(\hat{\alpha}_{jk}^i)+\Pi(\hat{\alpha}_{ki}^j)+\Pi(\hat{\alpha}_{ij}^k)\end{aligned} \tag{2.27}$$

其中 $\hat{\Theta}_i$ 是期望的顶点 v_i 的所有邻角的和；$\Pi(x)=-\int_0^x\ln|2\sin t|\mathrm{d}t$，为 Milnor-Lobachevsky 函数[21]。之所以这样定义，是因为 $\partial_{u_i}E=\frac{1}{2}(\hat{\Theta}_i-\sum_{v_i\in t_{ijk}}\hat{\alpha}_{ki}^j)$。因此当能量的梯度等于 0 时，各顶点的所有邻角的和恰好为期望值。将内点的 $\hat{\Theta}_i$ 设为 2π，就能保证结果能嵌入一个二维平面中。

优化上述能量后我们得到了 $u^* = \text{argmin}_u E(u)$。进而可以计算出边长和夹角。最后利用这两个信息进行重建即可得到插值结果。

几年后，Chen 等[5]将上篇文章从二维平面网格扩展到四面体网格。该方法的有界扭曲既包含有界共形扭曲，又包含有界等距扭曲。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

$$K_{e_j} = 2\pi - \sum_{t_i \in N(e_j)} \beta_j^i \tag{2.28}$$

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[5]

然后，该方法利用了文章 [6] 中的一个定理，即包含所有满足式2.29的 n 阶矩阵 $\boldsymbol{M}$ 的集合是一个凸集。

$$\begin{aligned} \lambda_1(\boldsymbol{M}) &\leqslant \alpha \\ \lambda_n(\boldsymbol{M}) &\geqslant \beta \\ K(\boldsymbol{M}) &\leqslant \gamma \end{aligned} \tag{2.29}$$

其中 $\lambda_1(\boldsymbol{M})$ 代表矩阵 $\boldsymbol{M}$ 的最大特征值；$\lambda_n(\boldsymbol{M})$ 代表矩阵 $\boldsymbol{M}$ 的最小特征值；$K(\boldsymbol{M}) = \frac{\lambda_1(\boldsymbol{M})}{\lambda_n(\boldsymbol{M})}$；$\alpha, \beta, \gamma$ 则均为常数。式2.29的前两个子式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2.29等价于下式

$$\begin{aligned} \boldsymbol{M} - \lambda_n \boldsymbol{I} &\geqslant 0 \\ \lambda_1 \boldsymbol{I} - \boldsymbol{M} &\geqslant 0 \\ s\boldsymbol{I} - \boldsymbol{M} &\geqslant 0 \\ K^2\boldsymbol{M} - s\boldsymbol{I} &\geqslant 0 \end{aligned} \tag{2.30}$$

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

对于式3.1中的变换矩阵 $\boldsymbol{J}$，我们可以定义其对称狄利克雷扭曲 (symmetric Dirichlet distortion) 为

$$\zeta(\boldsymbol{J}) = \sum_{i=1}^{n} \sigma_i^2 + \sigma_i^{-2} \tag{2.31}$$

其中 σ_i，$i=1,\cdots,n$ 是矩阵 $\boldsymbol{J}$ 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

$$\zeta(\boldsymbol{M}) \leqslant \zeta \tag{2.32}$$

等价于

$$\begin{aligned} &\mathrm{tr}(\boldsymbol{M}) + \mathrm{tr}(\boldsymbol{Y}) \leqslant \zeta \\ &\begin{pmatrix} \boldsymbol{Y} & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{M} \end{pmatrix} \geqslant 0 \end{aligned} \tag{2.33}$$

其中 ζ 是用户自定义的常数，$\boldsymbol{M}$ 是一个正定矩阵，$\boldsymbol{Y}$ 是一个与 $\boldsymbol{M}$ 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 $\sum_{t_i}\zeta(\boldsymbol{M}_i)|t_i|$，其中 $|t_i|$ 是四面体 t_i 的体积，$\boldsymbol{M}_i$ 是 t_i 对应的变换矩阵。值得注意的是，这里的 $\boldsymbol{M}_i$ 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2.29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。除此之外其考虑到所有内边 e_j 的截面离散曲率，即 $\boldsymbol{K}_{e_j}=0$，因此将之作为惩罚项加入到能量中。因此其能量是

$$E = \lambda \parallel \vec{K} \parallel^2 + \sum_{t_i}\zeta(\boldsymbol{M}_i)|t_i| \tag{2.34}$$

其中 λ 是权重，$\vec{K}$ 是将所有的 $\boldsymbol{K}_{e_j}$ 组合在一起的向量。利用式2.32等价于式2.33这一事实，再一阶近似 $\boldsymbol{K}$ 项，则利用论文中方法，该优化问题的第 l 次迭代可被转换为

$$\begin{aligned} \min_{\vec{x},S,R} \quad & \lambda_l \parallel \vec{K}_{\vec{x}_l} + \boldsymbol{J}_{\vec{x}_l}\vec{x} \parallel^2 + \sum_{t_i}(\mathrm{tr}(\boldsymbol{M}_i^R) + \mathrm{tr}(\boldsymbol{R}_i))|t_i| \\ \underset{i=1,\cdots,n}{\text{subject to}} \quad & \boldsymbol{M}_i^S = \boldsymbol{T}_i^S[\vec{x}]_i \\ & \boldsymbol{M}_i^R = \boldsymbol{T}_i^R[\vec{x}]_i \\ & \begin{pmatrix} \boldsymbol{R}_i & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{M}_i^R \end{pmatrix} \geqslant 0 \\ & \mathrm{tr}(\boldsymbol{M}_i^S) + \mathrm{tr}(\boldsymbol{S}_i) \leqslant \zeta_i \\ & \begin{pmatrix} \boldsymbol{R}_i & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{M}_i^R \end{pmatrix} \geqslant 0 \end{aligned} \tag{2.35}$$

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；$\boldsymbol{S}_i, \boldsymbol{R}_i$ 是松弛对称矩阵；ζ_i 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；$\boldsymbol{T}_i^S$ 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2.35，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 $\boldsymbol{J}_i$ 和向量 $\boldsymbol{b}$ 使得

$$T_i^2 = \boldsymbol{J}_i T_i^1 + \boldsymbol{b} \tag{2.36}$$

若将这两个三角面的对应的一个顶点平移至原点，则此时 $\boldsymbol{b} = \boldsymbol{0}$。此时，我们只需插值矩阵 $\boldsymbol{J}_i$ 即可。在时刻 t 时，设此时插值出的变换矩阵为 $\boldsymbol{A}(t)$。最简单的做法自然是令 $\boldsymbol{A}(t) = (1-t)\boldsymbol{I} + t\boldsymbol{A}$。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 $\boldsymbol{R}$ 和放缩部分 $\boldsymbol{S}$，即 $\boldsymbol{J} = \boldsymbol{R}\boldsymbol{S}$。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 $\boldsymbol{A}(t) = \boldsymbol{R}(t)((1-t)\boldsymbol{I} + t\boldsymbol{S})$，其中的 $\boldsymbol{R}(t)$ 是通过对 $\boldsymbol{R}$ 进行旋转角度插值得到的旋转矩阵。接着我们设能量

$$E = \sum_f \| \boldsymbol{J} - \boldsymbol{A}(t) \|_F^2 \tag{2.37}$$

其中 $\|\|_F$ 代表 Frobenius 范数。接着将 $\boldsymbol{J}$ 写成有关顶点坐标的线性表示并代入。此时式2.37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]：

$$E(f) = \int_M \mathrm{dist}(\mathrm{d}f, SO(n))^2 = \int_M \min_{\boldsymbol{R}\in \mathrm{SO}(n)} |\mathrm{d}f - \boldsymbol{R}|^2 \tag{2.38}$$

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 $\mathrm{d}f = \boldsymbol{R}\boldsymbol{Y}$，则这个 $\boldsymbol{R}$ 即是上式取到最小值的 $\boldsymbol{R}$。因此此时有 $\mathrm{d}f - \boldsymbol{R} = \boldsymbol{R}(\boldsymbol{Y} - \boldsymbol{I})$。将其分解为迹为 0 的矩阵和常数矩阵两部分，即

$$\boldsymbol{Y} - \boldsymbol{I} = (\boldsymbol{Y} - \frac{\mathrm{tr}(\boldsymbol{Y})}{n}\boldsymbol{I}) + (\frac{\mathrm{tr}(\boldsymbol{Y})}{n} - 1)\boldsymbol{I} =: \overline{\boldsymbol{Y}} + y\boldsymbol{I} \tag{2.39}$$

于是我们可以定义能量

$$E_{\alpha,\beta}(f)=\int_{M}|\beta\overline{\boldsymbol{Y}}+\alpha y\boldsymbol{I}|^2=\int_{M}\beta^2|\overline{\boldsymbol{Y}}|^2+\frac{\alpha^2}{n}(\mathrm{tr}(\boldsymbol{Y})-n)^2 \tag{2.40}$$

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 $\boldsymbol{M}_0$ 和 $\boldsymbol{M}_1$ 分别是源网格和目标网格，设 $\boldsymbol{M}_t$ 是时刻 t 时的网格，则该方法定义其为

$$\boldsymbol{M}_t=\mathrm{argmin}_{\boldsymbol{M}_x}(1-t)d(\boldsymbol{M}_0,\boldsymbol{M}_x)+td(\boldsymbol{M}_x,\boldsymbol{M}_1) \tag{2.41}$$

接着，可以使用能量 $E_{\alpha,\beta}$ 代替 $d(\boldsymbol{M}_0,\boldsymbol{M}_x)$ 和 $d(\boldsymbol{M}_x,\boldsymbol{M}_1)$，即

$$\widetilde{\boldsymbol{M}_t}=\mathrm{argmin}_{\boldsymbol{M}_x}(1-t)E_{\alpha,\beta}(\boldsymbol{M}_0\rightarrow\boldsymbol{M}_x)+tE_{\alpha,\beta}(\boldsymbol{M}_1\rightarrow\boldsymbol{M}_x) \tag{2.42}$$

其中 $\widetilde{\boldsymbol{M}_t}$ 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 $\boldsymbol{S}_i$ 和 $\boldsymbol{S}_j$，该方法定义它们之间距离为

$$\overline{d}(\boldsymbol{S}_i,\boldsymbol{S}_j)=\sqrt{\frac{\sum_{k=1}^{n}\parallel\boldsymbol{S}_i[k]-\boldsymbol{S}_j[k]\parallel^2}{n}} \tag{2.43}$$

其中 $\boldsymbol{S}_i[k]$ 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 $\overline{d}(\boldsymbol{S}_i,\boldsymbol{S}_j)$ 分隔开，定义距离度量 $d=F(\overline{d})=\overline{d}^2$。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 k 个模型，则该空间可以用这 k 个模型线性表示。假设 $\boldsymbol{S}_s$ 和 $\boldsymbol{S}_t$ 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:

$$\min\sum_{i=0}^{m+1}\sum_{j=0}^{m+1}f_{ij}x_{ij} \tag{2.44}$$

并且有：

$$\sum_{j=0}^{m+1}x_{ij}-\sum_{j=0}^{m+1}x_{ji}=\begin{cases}1 & i=0\\ -1 & i=m+1\\ 0 & i=1,\cdots,m\end{cases} \tag{2.45}$$

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 $\boldsymbol{C}_i$ 到空间 $\boldsymbol{C}_j$。并且 $f_{ij}=F(d_{ij})=\overline{d}(cs_i,cs_j)$，其中 cs_i 和 cs_j 分别是空间 $\boldsymbol{C}_i$ 到空间 $\boldsymbol{C}_j$ 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2.44。

通过求解式2.44获得了 N_R 个被选中空间的参考模型 $\overline{S}_k$，$k = 1, 2, \cdots, N_R$。接着该方法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

$$E_k = \sum_{i=1}^{n} \sum_{j \in N_i} \left(w_{ij} \parallel (\hat{S}[i] - \hat{S}[j]) - R_k[i](\overline{S}_k[i] - \overline{S}_k[j]) \parallel^2 \right) + \gamma \parallel \hat{S}[i] - \overline{S}_k[i] \parallel^2 \tag{2.46}$$

其中 n 是顶点数，N_i 是顶点 i 的一邻域，w_{ij} 是余切权重[13]，$R_k[i]$ 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。可以看到该能量就是 ARAP 能量加上一正则项。接着该方法定义 $\hat{S}$ 与所有参考模型之间总的能量为

$$E(t) = \sum_{k=1}^{N_R} \exp(-\epsilon |t - t_k|^2) E_k \tag{2.47}$$

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2.47。

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：

$$E(\hat{S}_S) = \sum_{i=1}^{n} \sum_{j \in N_i} \left(w_{ij} \parallel (\hat{S}_S[i] - \hat{S}_S[j]) - R[i](S_S[i] - S_S[j]) \parallel^2 \right) + \lambda \parallel \hat{S}_S[i] - \hat{N}_S[i] \parallel^2 \tag{2.48}$$

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 [16] 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[14]提出了 AIAP(as-isometric-as-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 $\varphi : [0,1] \times M \to \mathrm{R}^3$，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

$$X(t) := \left(\frac{d}{dt}p(t)\right)_{p \in M} \tag{2.49}$$

可以证明一个形状插值变换是等距变换当且仅当

$$\langle X_p(t) - X_q(t), p(t) - q(t)\rangle = 0 \tag{2.50}$$

对任何一条边 $e = (p(t), q(t))$ 成立，其中 $\langle , \rangle$ 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2.50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

$$\langle\!\langle X, Y\rangle\!\rangle_M := \sum_{(p,q)\in M} \langle X_p - X_q, p - q\rangle\langle Y_p - Y_q, p - q\rangle \tag{2.51}$$

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 $\langle\!\langle X, X\rangle\!\rangle_M = 0$。

为了解决这一问题，该方法加了一项正则项

$$\langle\!\langle X, Y\rangle\!\rangle_M^{L^2} := \sum_{p\in M} \langle X_p, Y_p\rangle A_p \tag{2.52}$$

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

$$\langle\!\langle X, Y\rangle\!\rangle_{M,\lambda} := \langle\!\langle X, Y\rangle\!\rangle_M + \lambda\langle\!\langle X, Y\rangle\!\rangle_M^{L^2} \tag{2.53}$$

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 $P_0, P_1, \cdots, P_{n+1}$ 是一个插值序列，$X_0, X_1, \cdots, X_n$ 是插值序列之间的形变场。该方法定义 P_i 的对称能量

$$E(P) := \sum_{i=0}^{n} \left(\langle\!\langle X_i, X_i\rangle\!\rangle_{P_i} + \langle\!\langle X_i, X_i\rangle\!\rangle_{P_{i+1}}\right) \tag{2.54}$$

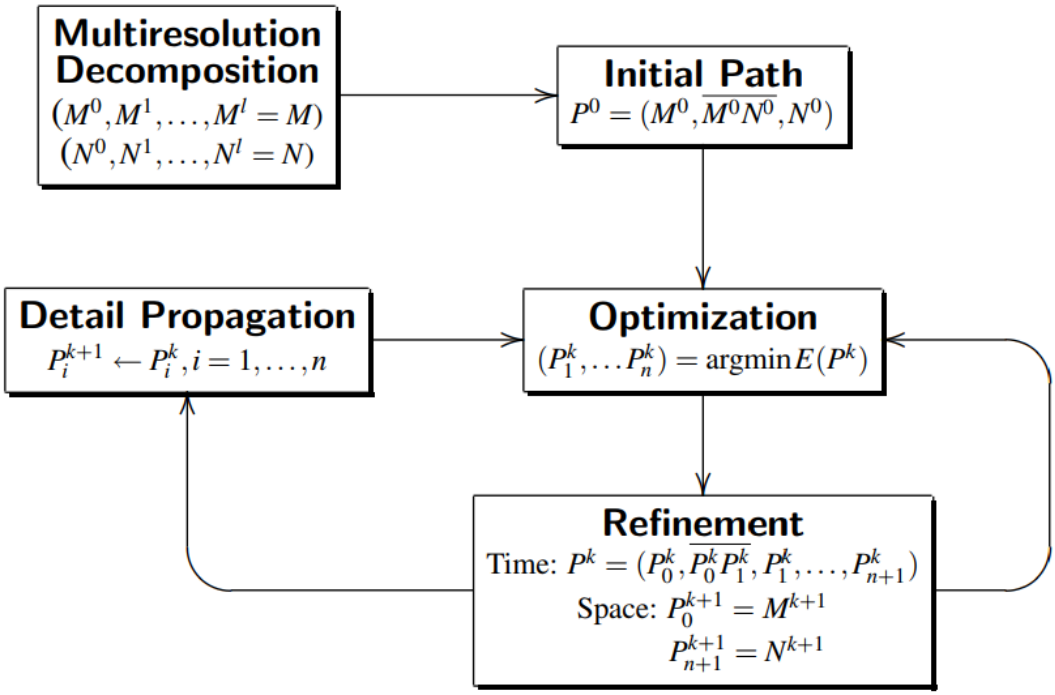

图 2.8　算法流程[14]

其是连续能量 $\int \langle\!\langle X, X \rangle\!\rangle_{P(t)} \mathrm{d}t$ 的离散化形式。该方法使用了拟牛顿法优化上式。

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 $P_i^{k+1}, i = 0, 1, \cdots, n+1$。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 $P_1^k, \cdots, P_n^k$。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 $\langle p - p', N_f \rangle$。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

$$P_i^{k+1} = \frac{n+1-i}{n+1} P_{i,0}^{k+1} + \frac{i}{n+1} P_{i,n+1}^{k+1} \tag{2.55}$$

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

$$F(t, P, \dot{P}) := \langle\langle \dot{P}(t), \dot{P}(t) \rangle\rangle_{M(t)} \tag{2.56}$$

其中 $\dot{P}$ 代表对 P 求导。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

$$F - \dot{p}_i^k \frac{\partial F}{\partial \dot{p}_i^k} = C_{i,k} \quad i = 1, \cdots, m, k = 1, 2, 3 \tag{2.57}$$

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

$$f(p) = \sum_{i=1}^{m} \sum_{k=1}^{3} (F - \dot{p}_i^k \frac{\partial F}{\partial \dot{p}_i^k} - C_{i,k})^2 \tag{2.58}$$

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。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 $\boldsymbol{D_\tau}$ 指的是 τ 的一个局部坐标架，同理坐标架 $\boldsymbol{D_\sigma}$ 指的是 σ 的一个局部坐标架。因此 $\boldsymbol{D_\tau}$ 可以表示为 $\boldsymbol{D_\tau} = [\boldsymbol{u_\tau} \quad \boldsymbol{v_\tau} \quad \boldsymbol{n_\tau}]$，其中 $\boldsymbol{u_\tau}$ 和 $\boldsymbol{v_\tau}$ 是三角形 τ 的两条边，而 $\boldsymbol{n_\tau}$ 则是三角形 τ 的单位法向量。$\boldsymbol{Q_{\sigma\tau}}$ 则是将 $\boldsymbol{D_\sigma}$ 转换为 $\boldsymbol{D_\tau}$ 的映射矩阵，即 $\boldsymbol{Q_{\sigma\tau}} = \boldsymbol{D_\tau^{-1} D_\sigma}$。

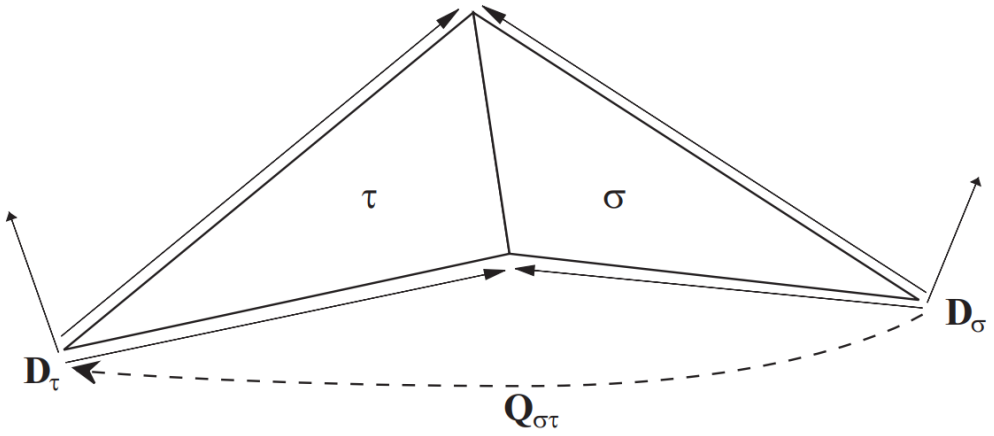

图 2.9　三面体 $\boldsymbol{D_\tau}$ 和 $\boldsymbol{D_\sigma}$ [33]

假设 $\boldsymbol{D}_{\boldsymbol{\tau}}^{\boldsymbol{t}}$ 是 t 时刻的坐标架。考虑到 $\boldsymbol{D}_{\tau}^{t}$ 的法向模长为 1 且与剩下两向量垂直，因此我们可以将其极分解为

$$D_{\tau}^{t}=R_{\tau}^{t}\hat{S}_{\tau}^{t}=\operatorname{matrix}(q_{\tau}^{t})\begin{pmatrix}S_{\tau}^{t} & 0\\ 0 & 1\end{pmatrix}:=(q_{\tau}^{t},S_{\tau}^{t}) \tag{2.59}$$

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

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

$$(\boldsymbol{q}_{\boldsymbol{\tau}}^{\boldsymbol{t}},\boldsymbol{S}_{\boldsymbol{\tau}}^{\boldsymbol{t}})\ominus(\boldsymbol{q}_{\boldsymbol{\tau}}^{\boldsymbol{s}},\boldsymbol{S}_{\boldsymbol{\tau}}^{\boldsymbol{s}})=(\boldsymbol{q}_{\boldsymbol{\tau}}^{\boldsymbol{t}}(\boldsymbol{q}_{\boldsymbol{\tau}}^{\boldsymbol{s}})^{-1},\boldsymbol{S}_{\boldsymbol{\tau}}^{\boldsymbol{t}}-\boldsymbol{S}_{\boldsymbol{\tau}}^{\boldsymbol{s}}) \tag{2.60}$$

这里的绝对指的是绝对世界坐标。同时定义绝对加号 $\oplus$ 为

$$w(\boldsymbol{q}_{\boldsymbol{\tau}}^{\boldsymbol{t}},\boldsymbol{S}_{\boldsymbol{\tau}}^{\boldsymbol{t}})\oplus(\boldsymbol{q}_{\boldsymbol{\tau}}^{\boldsymbol{s}},\boldsymbol{S}_{\boldsymbol{\tau}}^{\boldsymbol{s}})=(\operatorname{pow}(\boldsymbol{q}_{\boldsymbol{\tau}}^{\boldsymbol{t}},w)\boldsymbol{q}_{\boldsymbol{\tau}}^{\boldsymbol{s}},w\boldsymbol{S}_{\boldsymbol{\tau}}^{\boldsymbol{t}}+\boldsymbol{S}_{\boldsymbol{\tau}}^{\boldsymbol{s}}) \tag{2.61}$$

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

$$\operatorname{blend}(\boldsymbol{D}^{\boldsymbol{s}},\boldsymbol{D}^{\boldsymbol{t}},w)=w(\boldsymbol{D}^{\boldsymbol{t}}\ominus\boldsymbol{D}^{\boldsymbol{s}})\oplus\boldsymbol{D}^{\boldsymbol{s}} \tag{2.62}$$

不过由于该 blend 操作符依赖于三角形在世界坐标的朝向，因此需要改进。给定极分解 $(\boldsymbol{q}_{\boldsymbol{\sigma}}^{\boldsymbol{t}},\boldsymbol{S}_{\boldsymbol{\sigma}}^{\boldsymbol{t}})$ 和 $(\boldsymbol{q}_{\boldsymbol{\tau}}^{\boldsymbol{t}},\boldsymbol{S}_{\boldsymbol{\tau}}^{\boldsymbol{t}})$，则有

$$\boldsymbol{Q}_{\boldsymbol{\sigma\tau}}^{\boldsymbol{t}}=(\boldsymbol{R}_{\boldsymbol{\tau}}^{\boldsymbol{t}}\hat{\boldsymbol{S}}_{\boldsymbol{\tau}}^{\boldsymbol{t}})^{-1}\boldsymbol{R}_{\boldsymbol{\sigma}}^{\boldsymbol{t}}\hat{\boldsymbol{S}}_{\boldsymbol{\sigma}}^{\boldsymbol{t}}=(\hat{\boldsymbol{S}}_{\boldsymbol{\tau}}^{\boldsymbol{t}})^{-1}(\operatorname{matrix}((\boldsymbol{q}_{\boldsymbol{\tau}}^{\boldsymbol{t}})^{-1}\boldsymbol{q}_{\boldsymbol{\sigma}}^{\boldsymbol{t}}))\hat{\boldsymbol{S}}_{\boldsymbol{\sigma}}^{\boldsymbol{t}} \tag{2.63}$$

因此我们可以通过线性插值 $\boldsymbol{S}_{\boldsymbol{\sigma}}$ 和 $\boldsymbol{S}_{\boldsymbol{\tau}}$，球面线性插值 $(\boldsymbol{q}_{\boldsymbol{\tau}}^{\boldsymbol{t}})^{-1}\boldsymbol{q}_{\boldsymbol{\sigma}}^{\boldsymbol{t}}$) 来插值 $\boldsymbol{Q}_{\boldsymbol{\sigma\tau}}^{\boldsymbol{t}}$ 和 $\boldsymbol{Q}_{\boldsymbol{\sigma\tau}}^{\boldsymbol{s}}$ 得到 $\boldsymbol{Q}_{\boldsymbol{\sigma\tau}}^{\boldsymbol{t}}$。

当我们有 $\boldsymbol{Q}_{\boldsymbol{\sigma\tau}}$ 时，我们可以利用它来还原网格。由 $\boldsymbol{Q}_{\boldsymbol{\sigma\tau}}$ 的定义，可以得到 $\boldsymbol{Q}_{\boldsymbol{\sigma\tau}}^{\boldsymbol{T}}\boldsymbol{D}_{\boldsymbol{\tau}}^{\boldsymbol{T}}-\boldsymbol{D}_{\boldsymbol{\sigma}}^{\boldsymbol{T}}=\boldsymbol{0}$。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 $\boldsymbol{H}\in\mathbf{R}^{3p\times3m}$ 使得

$$\boldsymbol{H}\boldsymbol{D}^{\boldsymbol{T}}=\boldsymbol{0} \tag{2.64}$$

其中矩阵 $\boldsymbol{D}$ 是将 $\boldsymbol{D}_{\boldsymbol{\sigma}}^{\boldsymbol{T}}$ 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2.64得到 $\boldsymbol{D}$。

求到 $\boldsymbol{D}_{\boldsymbol{\sigma}}$ 后，可以求得顶点位置。考虑到 $\boldsymbol{D}_{\boldsymbol{\sigma}}=[\boldsymbol{u}_{\boldsymbol{\sigma}}\quad\boldsymbol{v}_{\boldsymbol{\sigma}}\quad\boldsymbol{n}_{\boldsymbol{\sigma}}]$，因此得到了 u_{σ} 和 v_{σ}。因此有

$$\boldsymbol{F}_{\boldsymbol{\sigma}}=[\boldsymbol{u}_{\boldsymbol{\sigma}}\quad\boldsymbol{v}_{\boldsymbol{\sigma}}]=[(\boldsymbol{x}_{\boldsymbol{i}}-\boldsymbol{x}_{\boldsymbol{j}})\quad(\boldsymbol{x}_{\boldsymbol{k}}-\boldsymbol{x}_{\boldsymbol{i}})] \tag{2.65}$$

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2.65写成一个矩阵等式，即 $\boldsymbol{GX}=\boldsymbol{0}$，其中 $\boldsymbol{X}$ 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 $\boldsymbol{X}$，即重建出插值网格。

第 3 章　基于边长的三维曲面插值

3.1　插值的几何量

首先, 我们先说明一下为何线性插值边长的平方能保证得到的结果有界扭曲的。对于一个三角形面，设其在源网格和目标网格分别为 t_1 和 t_2，则存在一个变换矩阵 $\boldsymbol{J}$ 使得

$$t_2 = \boldsymbol{J} t_1 \tag{3.1}$$

因此对于三角形 t_2 的任意一条边长 e_2，有 $|e_2|^2 = |\boldsymbol{J}e_1|^2 = (\boldsymbol{J}e_1)^T(\boldsymbol{J}e_1) = e_1^T\boldsymbol{J}^T\boldsymbol{J}e_1 = e_1^T\boldsymbol{M}e_1$，其中 $\boldsymbol{M} = \boldsymbol{J}^T\boldsymbol{J}$。因此 $|e_2|^2$ 可以展开成 $\boldsymbol{M}$ 的一次多项式。故线性插值 $\boldsymbol{M}$ 等价于线性插值边长的平方。故只需证明线性插值 $\boldsymbol{M}$ 得到的结果是有界扭曲的即可。设 $\boldsymbol{M}_t = (1-t)\boldsymbol{I} + t\boldsymbol{M}$，其中 $t \in [0,1]$ 是时间。则可以证明存在 $\alpha \in [0,1]$ 使得 $c(\boldsymbol{M}_t) \leqslant (1-\alpha)c(\boldsymbol{I}) + \alpha C(\boldsymbol{M}) = (1-\alpha) + \alpha C(\boldsymbol{M})$，其中 $C(\boldsymbol{M}) = \frac{\lambda_{max}}{\lambda_{min}}$，$\lambda_{max}$ 和 λ_{min} 分别是 $\boldsymbol{M}$ 的最大特征值和最小特征值。其详细证明可见文章 [19] 中的附录 A。这可以说明线性插值边长平方可以保证共形变换的扭曲是有界的。

除此之外，还可以证明对称狄利克雷能量关于矩阵 $\boldsymbol{M}$ 是光滑且凸的[5]。这可以说明线性插值边长平方可以保证等距变换的扭曲也是有界的。

因此由于线性插值边长的平方有着保证有界扭曲的优势，我们的方法对源网格和目标网格的对应边长的平方进行线性插值。即令

$$l_t = \sqrt{(1-t)l_1^2 + tl_2^2} \tag{3.2}$$

其中 l_1, l_2 分别为源网格和目标网格的边长，t 为时间。

3.2　能量优化

对于一个网格，设其所有边的集合为 $\boldsymbol{E}$，所有点的集合为 $\boldsymbol{V}$。设顶点 i 的坐标为 $\boldsymbol{p_i}$，则顶点 i 和顶点 j 连成的边 e_{ij} 的长度为 $\|\boldsymbol{p_i} - \boldsymbol{p_j}\|$。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

$$\begin{aligned} E_{ij} &= |\|\boldsymbol{p_i} - \boldsymbol{p_j}\|^2 - l_{ij}^2|^2 \\ E_l &= \sum_{e_{ij} \in E} E_{ij} \end{aligned} \tag{3.3}$$

其中 l_{ij} 是 e_{ij} 的目标边长，$\boldsymbol{E}_l$ 是网格所有边的能量总和。

接下来我们要优化能量 $\boldsymbol{E}_l$。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 $\boldsymbol{p_i} =$

(p_{i1}, p_{i2}, p_{i3})，则通过计算可得，

$$\begin{aligned}
\frac{\partial E_{ij}}{\partial p_{ik}} &= 4(\|\boldsymbol{p_i} - \boldsymbol{p_j}\|^2 - l_{ij}^2)(p_{ik} - p_{jk}) \\
\frac{\partial^2 E_{ij}}{\partial p_{ik}^2} &= 8(p_{ik} - p_{jk})^2 + 4(\|\boldsymbol{p_i} - \boldsymbol{p_j}\|^2 - l_{ij}^2) \\
\frac{\partial^2 E_{ij}}{\partial p_{ik}\partial p_{jk}} &= -8(p_{ik} - p_{jk})^2 - 4(\|\boldsymbol{p_i} - \boldsymbol{p_j}\|^2 - l_{ij}^2) \\
\frac{\partial^2 E_{ij}}{\partial p_{ik}\partial p_{il}} &= 8(p_{ik} - p_{jk})(p_{il} - p_{jl}) \\
\frac{\partial^2 E_{ij}}{\partial p_{ik}\partial p_{jl}} &= 8(p_{ik} - p_{jk})(p_{jl} - p_{il})
\end{aligned} \tag{3.4}$$

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

$$\begin{aligned}
\frac{\partial^2 E_{ij}}{\partial p_{ik}^2} &= -\frac{\partial^2 E_{ij}}{\partial p_{ik}\partial p_{jk}} \\
\frac{\partial^2 E_{ij}}{\partial p_{ik}\partial p_{il}} &= -\frac{\partial^2 E_{ij}}{\partial p_{ik}\partial p_{jl}}
\end{aligned} \tag{3.5}$$

设 $\boldsymbol{H_{ij}}$ 是 E_{ij} 的海森矩阵，则 $\boldsymbol{H_{ij}}$ 是一个 6×6 的对称矩阵。由等式3.5可得 $\boldsymbol{H_{ij}} = \begin{pmatrix} \boldsymbol{A_{ij}} & -\boldsymbol{A_{ij}} \\ -\boldsymbol{A_{ij}} & \boldsymbol{A_{ij}} \end{pmatrix}$，其中 $\boldsymbol{A_{ij}}$ 是一个 3×3 的对称矩阵，即

$$\boldsymbol{A_{ij}} = 4\begin{pmatrix} d_{ij} + 2q_1^2 & -2q_1q_2 & -2q_1q_3 \\ -2q_1q_2 & d_{ij} + 2q_2^2 & -2q_3q_2 \\ -2q_1q_3 & -2q_2q_3 & d_{ij} + 2q_3^2 \end{pmatrix} \tag{3.6}$$

其中 $d_{ij} = \|\boldsymbol{p_i} - \boldsymbol{p_j}\|^2 - l_{ij}^2$，$q_k = p_{ik} - p_{jk}$，$k = 1, 2, 3$。

令 $\boldsymbol{H} = \sum_{e_{ij}} \boldsymbol{H_{ij}}$。由于 $\boldsymbol{H_{ij}}$ 不一定是一个正定矩阵，所以 $\boldsymbol{H}$ 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 $\boldsymbol{H}$ 是一个正定矩阵。如果直接对 $\boldsymbol{H}$ 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 $\boldsymbol{H} = \sum_{e_{ij}} \boldsymbol{H_{ij}}$，我们可以直接对 $\boldsymbol{H_{ij}}$ 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 $O(|E|^3)$，并且依然能保证 $\boldsymbol{H}$ 是一个正定矩阵。由于 $\boldsymbol{H_{ij}}$ 是一个对称矩阵，因此将 $\boldsymbol{H_{ij}}$ 特征值分解后有 $\boldsymbol{H_{ij}} = \boldsymbol{Q\Lambda Q}^{\mathrm{T}}$。其中 $\boldsymbol{Q}$ 为正交矩阵，$\boldsymbol{\Lambda}$ 为实对角矩阵。接下来我们只需要令 $\widetilde{\boldsymbol{\Lambda}} = \max(0, \boldsymbol{\Lambda})$ 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 $\boldsymbol{H}$ 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 $\boldsymbol{H_{ij}} = \begin{pmatrix} \boldsymbol{A_{ij}} & -\boldsymbol{A_{ij}} \\ -\boldsymbol{A_{ij}} & \boldsymbol{A_{ij}} \end{pmatrix}$。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 $\boldsymbol{H} = \begin{pmatrix} \boldsymbol{A} & -\boldsymbol{A} \\ -\boldsymbol{A} & \boldsymbol{A} \end{pmatrix}$，则 $\boldsymbol{H}$ 是半正定矩阵的充分必要条件是 $\boldsymbol{A}$ 为半正定矩阵。

证明　充分性　若 $\boldsymbol{A}$ 是半正定矩阵，则对任意向量 $\boldsymbol{x} \in \mathbf{R}^n$ 有 $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} \geqslant 0$。对任意向量 $\boldsymbol{y} \in \mathbf{R}^{2n}$，设 $\boldsymbol{y} = (\boldsymbol{y_1}, \boldsymbol{y_2})$，其中 $\boldsymbol{y_1}, \boldsymbol{y_2} \in \mathbf{R}^n$。则有 $\boldsymbol{y}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{y} = (\boldsymbol{y_1} - \boldsymbol{y_2})^{\mathrm{T}}\boldsymbol{A}(\boldsymbol{y_1} - \boldsymbol{y_2}) \geqslant 0$。因此 $\boldsymbol{H}$ 是半正定矩阵。

必要性　若 $\boldsymbol{H}$ 是半正定矩阵，则对任意向量 $\boldsymbol{y} \in \mathbf{R}^{2n}$, 设 $\boldsymbol{y} = (\boldsymbol{y_1}, \boldsymbol{y_2})$，其中 $\boldsymbol{y_1}, \boldsymbol{y_2} \in \mathbf{R}^n$ 有 $\boldsymbol{y}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{y} \geqslant 0$。又因为 $\boldsymbol{y}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{y} = (\boldsymbol{y_1} - \boldsymbol{y_2})^{\mathrm{T}}\boldsymbol{A}(\boldsymbol{y_1} - \boldsymbol{y_2})$，因此 $(\boldsymbol{y_1} - \boldsymbol{y_2})^{\mathrm{T}}\boldsymbol{A}(\boldsymbol{y_1} - \boldsymbol{y_2}) \geqslant 0$。令 $\boldsymbol{y_2} = \boldsymbol{0}$, 则有 $\boldsymbol{y_1}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{y_1} \geqslant 0$。又因为 $\boldsymbol{y_1}$ 是任意的，因此 $\boldsymbol{A}$ 是半正定矩阵。 ∎

由上述事实可知我们只需令 $\boldsymbol{A_{ij}}$ 为半正定矩阵即可使得 $\boldsymbol{H_{ij}}$ 为半正定矩阵。因此我们可以像之前一样先将 $\boldsymbol{A_{ij}}$ 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 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。但是此时仍然需要进行特征值分解，因此如果能不进行特征值分解并且直接得到最终结果那将进一步提高速度。

设

$$\boldsymbol{P_{ij}} = \begin{pmatrix} q_1^2 & -q_1q_2 & -q_1q_3 \\ -q_1q_2 & q_2^2 & -q_3q_2 \\ -q_1q_3 & -q_2q_3 & q_3^2 \end{pmatrix} \tag{3.7}$$

则计算可得 $\boldsymbol{P_{ij}}$ 的三个特征值分别为 0,0 和 $q_1^2 + q_2^2 + q_3^2$。

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

$$\begin{aligned} \lambda_1 &= 4(\|p_i - p_j\|^2 - l_{ij}^2), \\ \lambda_2 &= 4(\|p_i - p_j\|^2 - l_{ij}^2), \\ \lambda_3 &= 12(q_1^2 + q_2^2 + q_3^2) - 4l_{ij}^2 \end{aligned} \tag{3.8}$$

由于 $\boldsymbol{A_{ij}}$ 是对称矩阵，因此存在正交矩阵 $\boldsymbol{Q}$ 和对角阵 $\boldsymbol{D}$ 使得

$$\boldsymbol{A_{ij}} = \boldsymbol{Q}\boldsymbol{D}\boldsymbol{Q}^{\mathrm{T}} \tag{3.9}$$

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

$$\boldsymbol{A_{ij}} = \lambda_1 \boldsymbol{x_1} \boldsymbol{x_1}^{\mathrm{T}} + \lambda_2 \boldsymbol{x_2} \boldsymbol{x_2}^{\mathrm{T}} + \lambda_3 \boldsymbol{x_3} \boldsymbol{x_3}^{\mathrm{T}} \tag{3.10}$$

由于 $\boldsymbol{A_{ij}}$ 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 $\widetilde{\boldsymbol{A_{ij}}} = \widetilde{\lambda_1} \boldsymbol{x_1} \boldsymbol{x_1}^{\mathrm{T}} + \widetilde{\lambda_2} \boldsymbol{x_2} \boldsymbol{x_2}^{\mathrm{T}} + \widetilde{\lambda_3} \boldsymbol{x_3} \boldsymbol{x_3}^{\mathrm{T}}$。则此时 $\widetilde{\boldsymbol{A_{ij}}}$ 为半正定矩阵。由于 $\boldsymbol{Q}$ 是一个正交阵，即 $\boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}} = \boldsymbol{I}$。因此有 $\boldsymbol{x_1} \boldsymbol{x_1}^{\mathrm{T}} + \boldsymbol{x_2} \boldsymbol{x_2}^{\mathrm{T}} + \boldsymbol{x_3} \boldsymbol{x_3}^{\mathrm{T}} = \boldsymbol{I}$。并且由于 $\widetilde{\lambda_1} = \widetilde{\lambda_2}$, 因此有

$$\widetilde{\boldsymbol{A_{ij}}} = \widetilde{\lambda_1}(\boldsymbol{I} - \boldsymbol{x_3} \boldsymbol{x_3}^{\mathrm{T}}) + \widetilde{\lambda_3} \boldsymbol{x_3} \boldsymbol{x_3}^{\mathrm{T}} \tag{3.11}$$

接下来我们只需计算 $\boldsymbol{x_3}$。经过计算可得

$$\boldsymbol{x_3} = \frac{(p_{i1} - p_{j1}, p_{i2} - p_{j2}, p_{i3} - p_{j3})}{\sqrt{(p_{i1} - p_{j1})^2 + (p_{i2} - p_{j2})^2 + (p_{i3} - p_{j3})^2}} \tag{3.12}$$

将其代入上式即可得到 $\widetilde{\boldsymbol{A_{ij}}}$。接下来我们用 $\widetilde{\boldsymbol{A_{ij}}}$ 替代 $\boldsymbol{A_{ij}}$ 则此时有 $\boldsymbol{H_{ij}}$ 为半正定矩阵，进而有 $\boldsymbol{H}$ 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 $\boldsymbol{H_{ij}}$。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 $\boldsymbol{H_{ij}}$ 是一个三阶矩阵，理论上这一部分所需的计算量是之前的 1/3。

由于 $\boldsymbol{H}$ 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 $\boldsymbol{H}$ 是一个正定矩阵，在这里我们可以令 $\widetilde{\boldsymbol{H}} = \boldsymbol{H} + \epsilon \boldsymbol{I}$，其中 ϵ 为一很小的数，例如 $\epsilon = 10^{-15}$。

将 $\boldsymbol{H}$ 近似为正定矩阵后我们就得到了下降方向 $d = -\boldsymbol{H}^{-1} \nabla E$。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。

3.3　插值

使用上一节的算法插值后得到的结果如图3.1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。

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，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

$$\widetilde{E} = E_l + w \| \boldsymbol{P_{t_i}} - \boldsymbol{P_i} \|^2 \tag{3.13}$$

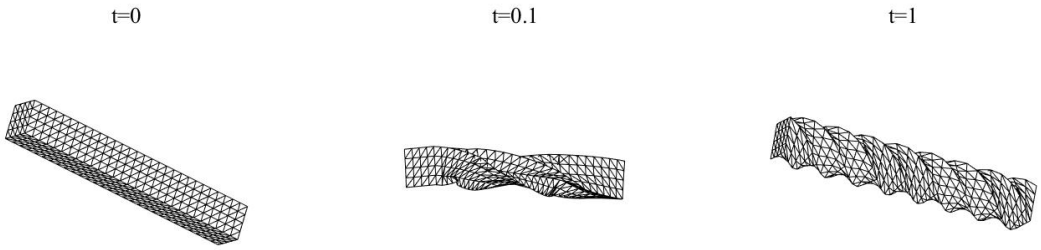

图 3.1　上图是两个长条之间的插值结果。其中左图和右图分别为初始网格和目标网格。中图是 $t = 0.1$ 时的结果

其中 t_i 为第 i 帧对应的时间 t，$\boldsymbol{P}_{t_i}$ 为 t_i 时刻所有点的坐标，$\boldsymbol{P_i}$ 为初始化网格，w 为权重。若 w 过大，则会使能量 $\boldsymbol{E}_t$ 项占比过小，使得难以较好的通过优化 $\widetilde{\boldsymbol{E}}$ 来间接优化 $\boldsymbol{E}$；若 w 过小，则图3.1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 能够使得在两个模型放缩一定倍数时，输出的结果也放缩同样的倍数。加上这个正则项后，只要初始化网格二面角的具有连续性，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的平方，因此我们自然觉得初始化的边长平方的误差越小，结果越好。于是我们准备使用 SQP[5] 这个已经对边长平方进行较好优化的方法进行初始化。为了验证该方法确实更加优秀，我们还将其与 FFMP[33]、ARAP[23]、GE[24] 和 ABF[17] 共 4 种方式得到的网格比较。当然，这些方法的插值边长部分也相应改为线性插值边长的平方。

首先，我们先定义一个评价标准

$$L = \sqrt{\frac{1}{|E|}\sum_{e\in E}(\frac{l_{real}}{l_{target}} - 1)^2} \tag{3.14}$$

其中 $\boldsymbol{E}$ 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 $\boldsymbol{L}$ 就是边长的相对误差的 l_2 范数。

如表3.1所示，在这五种初始化网格中，SQP 的 $\boldsymbol{L}$ 值更小，说明插值结果的总体误差最小。图3.2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3.2则可以看到 SQP 得到的结果更蓝，对应了其更小的边长误差。并且我们可以看到 SQP 得到的结果也很合理。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。

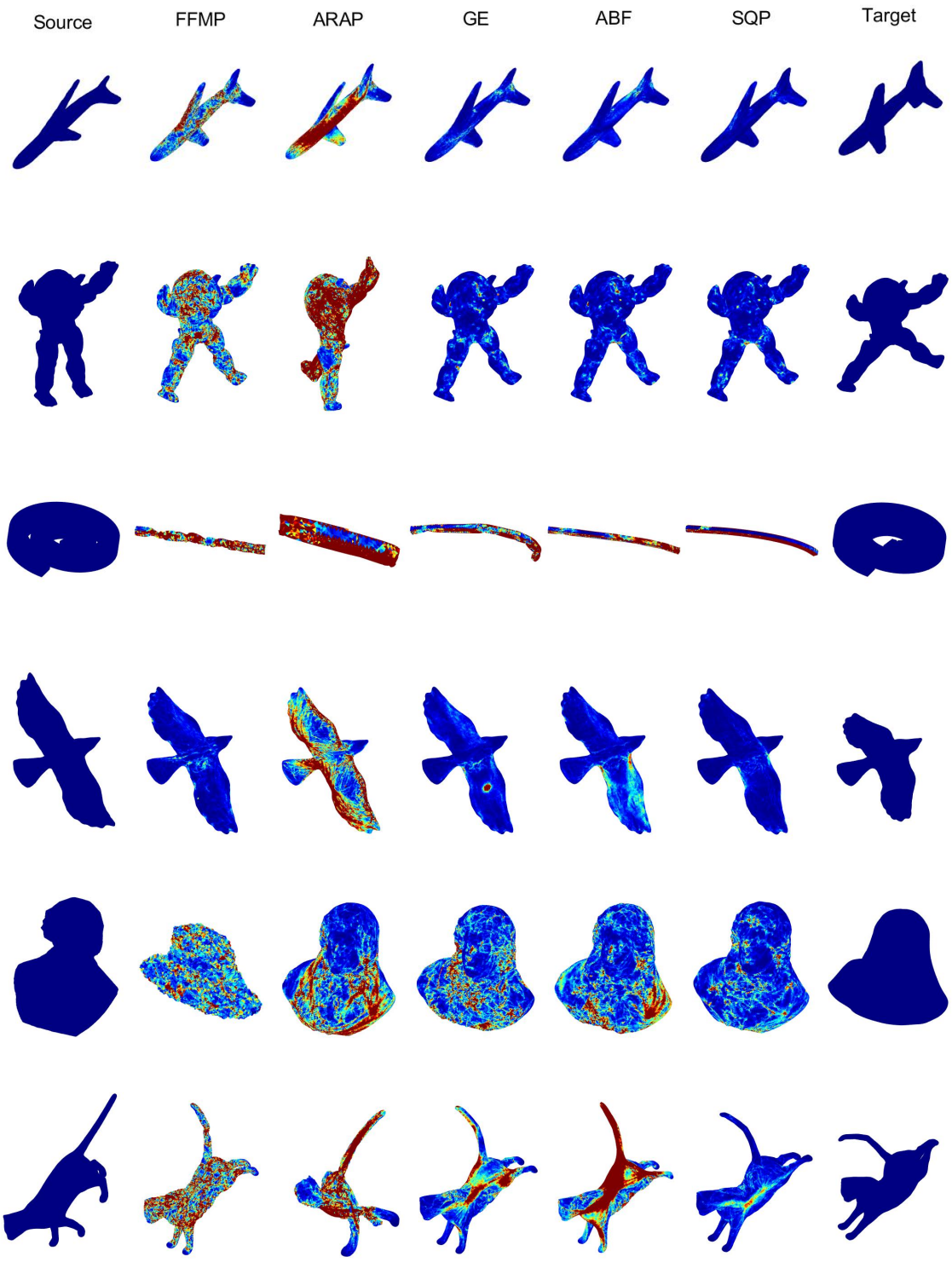

图 3.2　在不同初始化网格下 $t = 0.5$ 时得到的结果。从上往下分别是对应于表3.1的飞机 (airplane)、犰狳 (armadillo)、长条 (bar)、鸟 (bird)、半身像 (bust) 和猫 (cat) 这 6 个模型。颜色越偏橙红说明实际边长与目标边长的差越大，越偏深蓝色则说明越小。

表 3.1　在不同初始化网格下得到的结果的 L 值

		$L(10^{-3})$						
		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

3.4　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} 和权重。

步骤 2：使用 SQP 算法得到 t 时刻的初始化网格。

步骤 3：计算梯度 $\nabla \boldsymbol{E}$ 以及此时的能量 $\boldsymbol{E}$。

步骤 4：计算正交化之后的海塞矩阵 $\boldsymbol{H}$。

步骤 5：计算最优步长 α。

步骤 6：更新网格顶点坐标并计算更新后的能量 $\boldsymbol{E}_{new}$。

步骤 7：比较 $\boldsymbol{E}$ 和 $\boldsymbol{E}_{new}$，若 $\boldsymbol{E}$>1.01$\boldsymbol{E}_{new}$ 则重复迭代步骤 3-5，否则停止迭

代。

3.5　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，所用语言主要为matlab，辅以C++。我们选取了 LSRDF[7]，ELI[10]和 MSGI[8]用于对比。

在实验中，我们使用的网格均是三维空间中的封闭网格。使用封闭网格时为了能使用 SQP 算法进行初始化，因为四面体网格的表面是封闭网格。

图3.3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。

图3.4和图3.5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。在图3.4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3.4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3.5左列则可以看到我们的方法的 $\boldsymbol{L}$ 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3.5右列则可以看到除了在第一个模型，即长条 (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3.4，还是从图3.5来看，我们的方法均是最好的。不过从图3.5也可以看出我们的方法的 $\boldsymbol{L}$ 值这一曲线不够光滑，说明其边长误差不够稳定。ELI 和 MSGI 这两种算法也有这种现象的出现，但 LSRDF 没有。这很可能是因为 LSRDF 的能量函数是一个二次函数，可求解出其全局最优点，并且只需要一步即可嵌入到 R^3。而我们的方法和 ELI 均是非线性优化，无法保证得到全局最优点，MSGI 则需要逐层重建。因此误差都不稳定。

图3.6则是验证我们的方法是否能够保证结果是有界扭曲的，同时还与其它三种方法进行比较。验证有界扭曲主要有两个指标。第一个是衡量共形扭曲的，即 $\sqrt{\lambda_{max}/\lambda_{min}}$。其中 λ_{max} 和 λ_{min} 分别是 $\boldsymbol{J}\boldsymbol{J}^{\mathrm{T}}$ 的最大特征值和最小特征值，$\boldsymbol{J}$ 是仿射变换矩阵。第二个是用于衡量等距扭曲的对称狄利克雷能量，在二维情形下即 $\lambda_{max} + \lambda_{min} + \lambda_{max}^{-1} + \lambda_{max}^{-1}$。从图3.6可以看到无论是共形扭曲还是对称狄利克雷能量，我们的结果的曲线始终是光滑且单调递增的。LSRDF 方法在长条 (bar) 模型下的结果曲线既不单调递增也不光滑，说明其难以保证结果是有界扭曲的。对于 MSGI 方法，猫 (cat) 等模型下的共形扭曲和对称狄利克雷能量在中间某些

时间均超过了目标网格的，因此其无法保证结果是有界扭曲的。ELI 这一方法在狗 (dog) 等模型下的共形扭曲曲线则既不单调递增也不光。以上结果说明我们的方法是能有效的保证结果是有界扭曲的，并且在这一方面比其它三种方法均要出色许多。

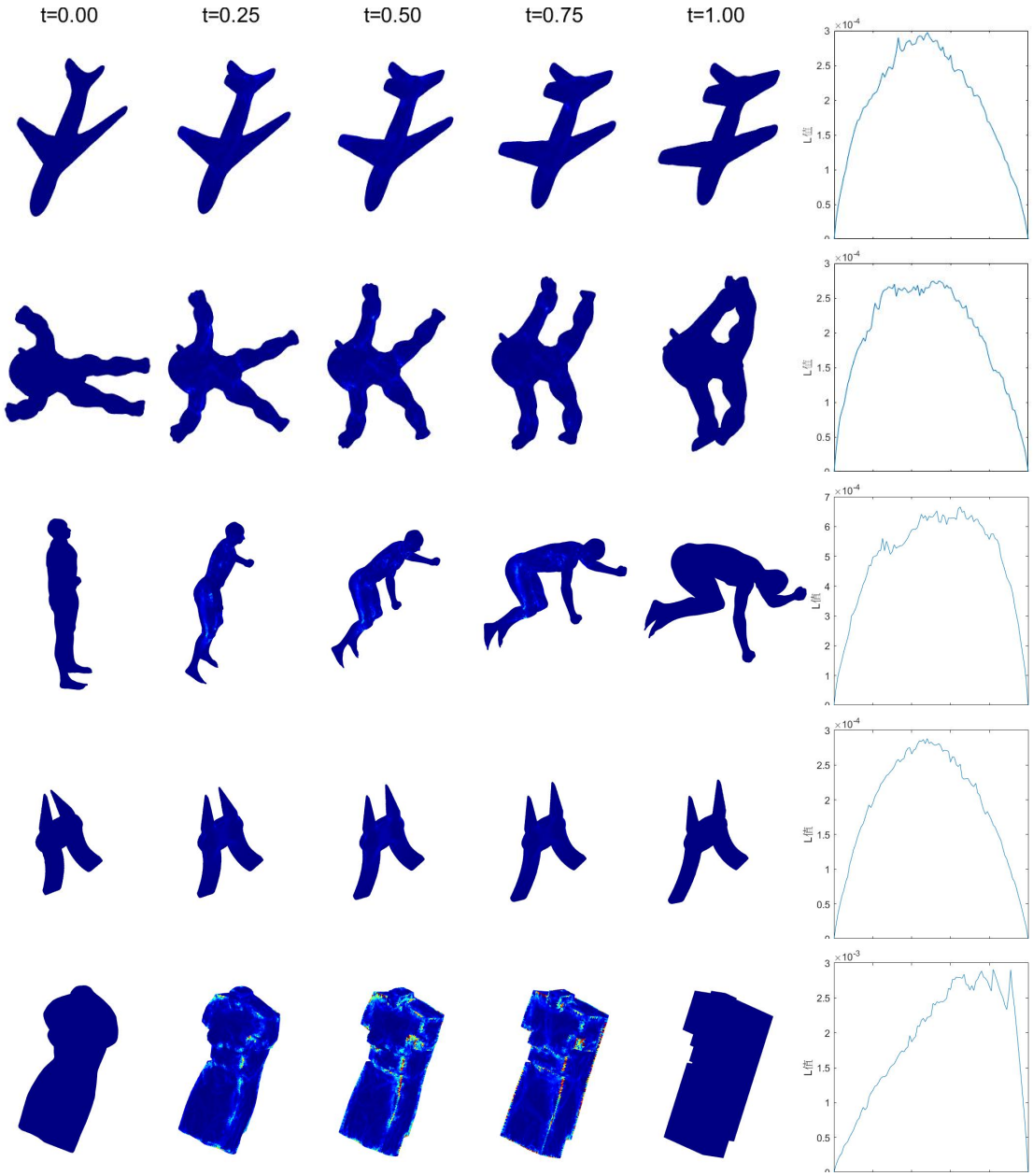

图 3.3　从上到下依次是飞机、犰狳 (armadillo)、人、钳子和半身像这五个模型的插值结果。其中 $t = 0.00$ 和 $t = 1.00$ 分别代表源模型和目标模型，其它时刻则为插值得到的结果。最右边是不同时刻的 L 值误差

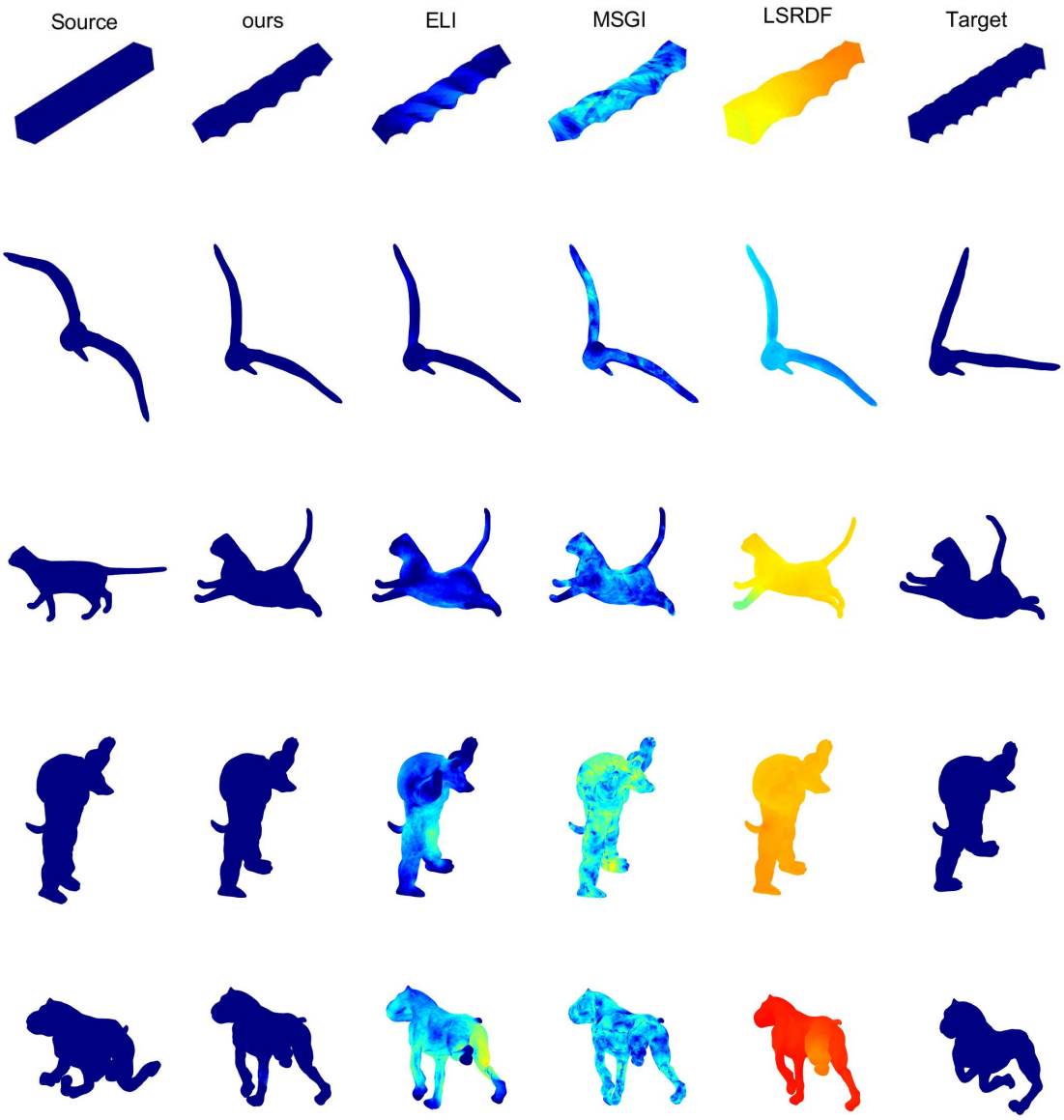

图 3.4　比较四种方法在五个模型下的插值结果，从上到下依次是长条 (bar)、鸟 (bird)、猫 (cat)、犰狳 (armadillo) 和狗 (dog)。其中中间四列是四种方法在 $t = 0.5$ 时刻的结果。颜色越偏橙红说明实际边长与目标边长的差越大，越偏深蓝色则说明越小

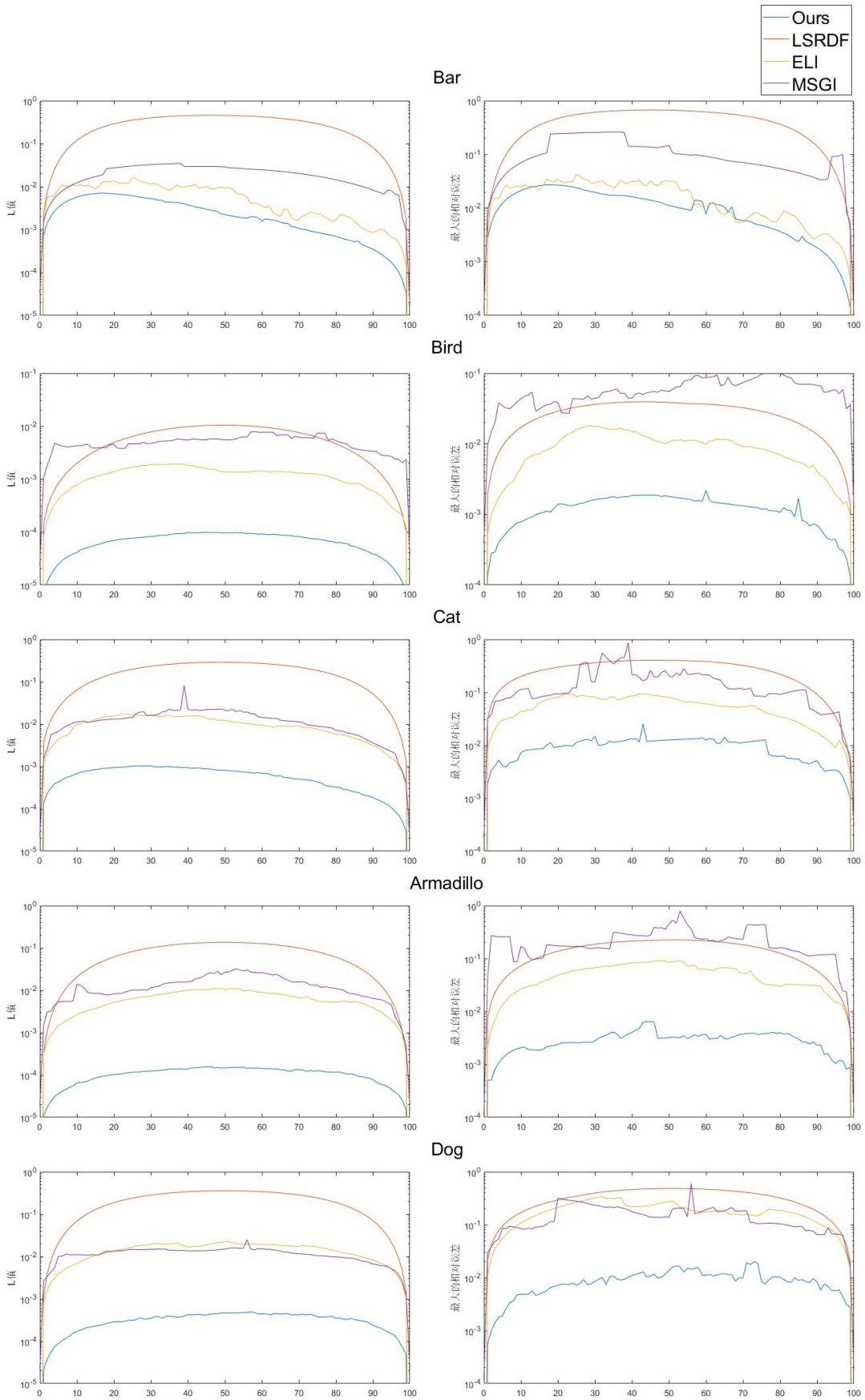

图 3.5　比较四种方法在图3.4的五个模型下的误差程度。横轴代表帧数，其中第 0 帧和第 100 帧分别代表源模型和目标模型。左列是四种方法下的 L 值，右列则是最大的相对误差，其中相对误差为 $|(l_{real}-l_{target})/l_{target}|$

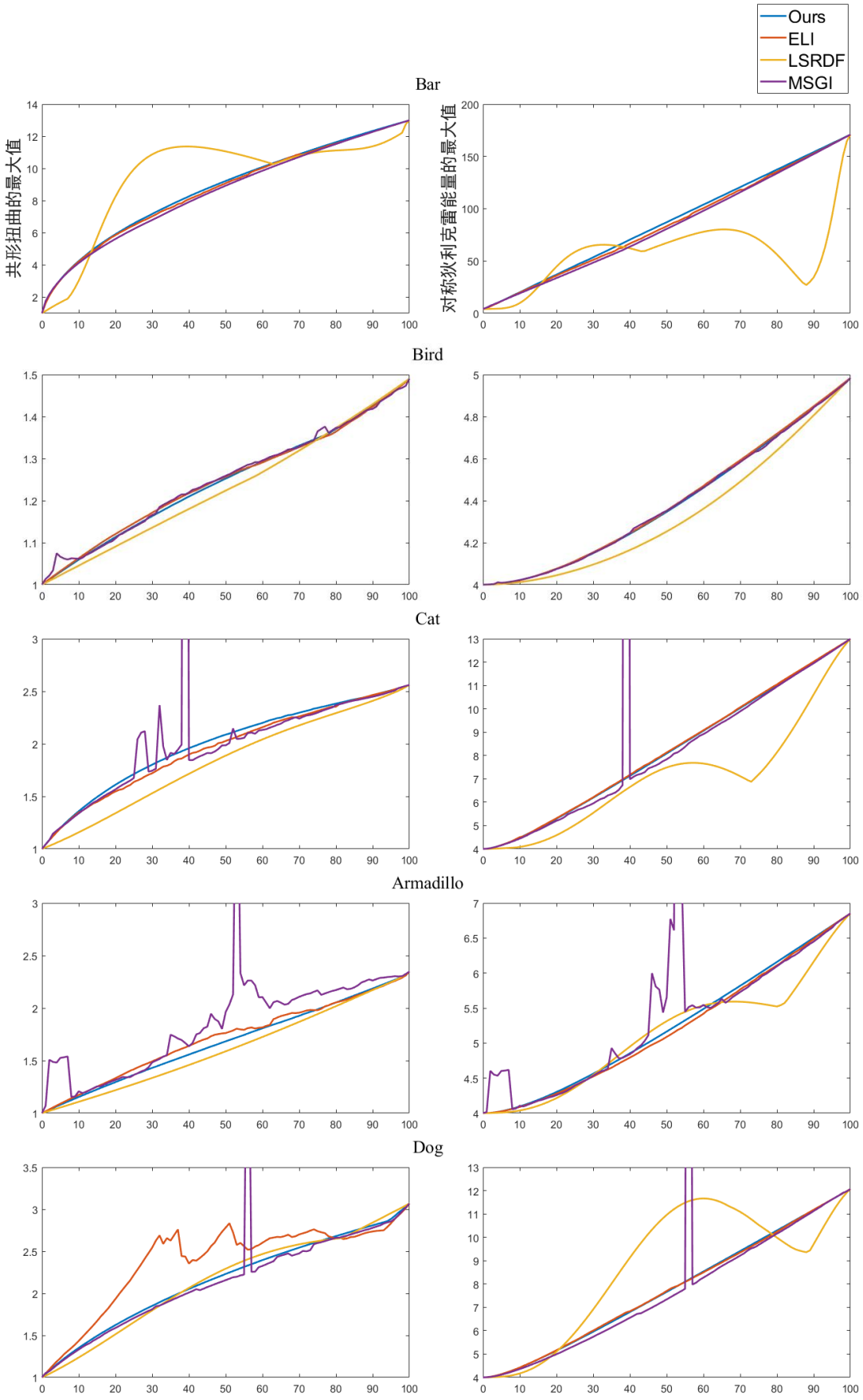

图 3.6　比较四种方法在图3.4的五个模型下的扭曲。横轴代表帧数，其中第 0 帧和第 100 帧分别代表源模型和目标模型。左列是四种方法下所有面的共形扭曲的最大值，右列则是所有面的对称狄利克雷能量的最大值

第 4 章　总结与展望

4.1　本文总结

我们提出了一种新的基于边长的形状插值算法。该方法应用于两个兼容的三维网格。我们先是解释了为何线性插值边长平方能够保证插值结果是有界扭曲的。其次，我们使用了顶点坐标作为变量定义了能量函数，避免了像文章[10]一样的间接重建。随后，我们介绍了本方法的优化算法。我们使用了牛顿法作为我们的优化方法，并且为此将海森矩阵近似为正定矩阵。为了加快速度，我们利用了多种技巧，使得我们能够直接计算出近似后的正定矩阵，无需特征值分解。最后我们进行了实验，与方法[7-8,10] 相比较，证明了我们的方法在边长误差上的表现更为优异。

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误差相较于其它方法有了较大减少，但是最大的边长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，我们可以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] LIU C. An analysis of the current and future state of 3d facial animation techniques and systems [D]. School of Interactive Arts & Technology-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] 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.

[6] 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.

[7] 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.

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

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

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

[11] 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.

[12] WILLIAMS J A, BENNAMOUN M. Simultaneous registration of multiple point sets using orthonormal matrices[C]//2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100): volume 4. IEEE, 2000: 2199-2202.

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

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

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

[16] SUMNER R W, POPOVIĆ J. Deformation transfer for triangle meshes[J]. ACM Transactions on graphics (TOG), 2004, 23(3): 399-405.

[17] PAILLÉ G P, RAY N, POULIN P, et al. Dihedral angle-based maps of tetrahedral meshes[J]. ACM Transactions on Graphics (TOG), 2015, 34(4): 1-10.

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

[19] CHEN R, WEBER O, KEREN D, et al. Planar shape interpolation with bounded distortion[J]. ACM Transactions on Graphics (TOG), 2013, 32(4): 1-12.

[20] SPRINGBORN B, SCHRÖDER P, PINKALL U. Conformal equivalence of triangle meshes [M]//ACM SIGGRAPH 2008 papers. 2008: 1-11.

[21] MILNOR J W. Hyperbolic geometry: the first 150 years[J]. Bulletin of the American Mathematical Society, 1982, 6(1): 9-24.

[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, COHEN-OR D, LEVIN D. As-rigid-as-possible shape interpolation[C]// Proceedings of the 27th annual conference on Computer graphics and interactive techniques. 2000: 157-164.

[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 Graph-

ics (TOG), 2008, 27(2): 1-13.

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