第七次作业报告

PB21010362 汪兆辰

2023 年 5 月 5 日

1 算法介绍

ARAInterp 算法给出了一种从网格初始状态到末状态的插值过程. 考虑网格的每个三角形 T 对应的原始网格中三角形 $P_\mathcal{T}$ 与变换后的网格中三角形 $Q_\mathcal{T}$ 间的变换过程:

$$A_\mathcal{T} P_\mathcal{T} + l = Q_\mathcal{T} \tag{1}$$

其中 l 表示平移变换，在这里忽略为 0，并且给出：

$$P_\mathcal{T} = \begin{pmatrix} p_i^x & p_i^y \\ p_j^x & p_j^y \\ p_k^x & p_k^y \end{pmatrix}; Q_\mathcal{T} = \begin{pmatrix} q_i^x & q_i^y \\ q_j^x & q_j^y \\ q_k^x & q_k^y \end{pmatrix}; D = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{pmatrix} \tag{2}$$

那么此处的 $A_\mathcal{T}$ 可以表示为:

$$A_\mathcal{T} = P_\mathcal{T}^* Q_\mathcal{T} \tag{3}$$

其中 $P_\mathcal{T}^* = (DP_\mathcal{T})^{-1} D$.

对变换矩阵 A 作 SVD 分解，可以得到：

$$A_\mathcal{T} = R_\alpha \Sigma R_\beta = R_\alpha \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} R_\beta, \quad s_x, s_y > 0 \tag{4}$$

从而可以将 A 分解为旋转矩阵 R_γ 与对称阵 S 的乘积，其中：

$$R_\gamma = R_\alpha R_\beta; S = R_\beta^T \Sigma R_\beta = \begin{pmatrix} s_x & s_h \\ s_h & s_y \end{pmatrix} \tag{5}$$

旋转过程实际是对从初始到末尾状态的权重插值，而权重按照设置是均匀随时间变化的，因此也可以看作是对时间的插值：

$$A_\gamma(t) = R_{t\gamma}((1-t)I + tS) \tag{6}$$

其中 $R_{t\gamma}$ 是对旋转的角度插值，只需将 R_γ 对应的旋转角度乘以 t 并转换为旋转矩阵即可.

然而实际上，我们并不能用这里得到的 $A_{t\gamma}$ 直接作为变换. 因为这将破坏网格的结构，使之变为离散的三角形，为此我们仍然需要求解一个最小二乘问题：

$$E(t) = \sum_{\mathcal{T}=1}^{M} a_{\mathcal{T}} \left\| P_{\mathcal{T}}^{*} V_{\mathcal{T}} - A_{\mathcal{T}}(t) \right\|^2 \tag{7}$$

在这里 $V_{\mathcal{T}}$ 是要求的顶点坐标，$a_{\mathcal{T}}$ 是每个三角形的面积. 可以将其化为求解矩阵方程：

$$(\mathbb{P}^T W \mathbb{P})\mathbb{V} = \mathbb{P}^T W \mathbb{A} \tag{8}$$

其中

$$\mathbb{P} = \begin{pmatrix} \mathbb{P}_1^T \cdots \mathbb{P}_M^T \end{pmatrix}^T \tag{9}$$

$$\mathbb{A} = \begin{pmatrix} A_1^T \cdots A_M^T \end{pmatrix}^T \tag{10}$$

$$W = diag \begin{pmatrix} a_1 \cdots a_M \end{pmatrix} \tag{11}$$

并定义 $\mathbb{P}_{\mathcal{T}}$ 为只有第 i,j,k 列的稀疏矩阵：

$$\mathbb{P}_{\mathcal{T}} = \begin{pmatrix} \cdots (P_{\mathcal{T}}^*)_{11} \cdots (P_{\mathcal{T}}^*)_{12} \cdots (P_{\mathcal{T}}^*)_{13} \cdots \\ \cdots (P_{\mathcal{T}}^*)_{21} \cdots (P_{\mathcal{T}}^*)_{22} \cdots (P_{\mathcal{T}}^*)_{23} \cdots \end{pmatrix} \tag{12}$$

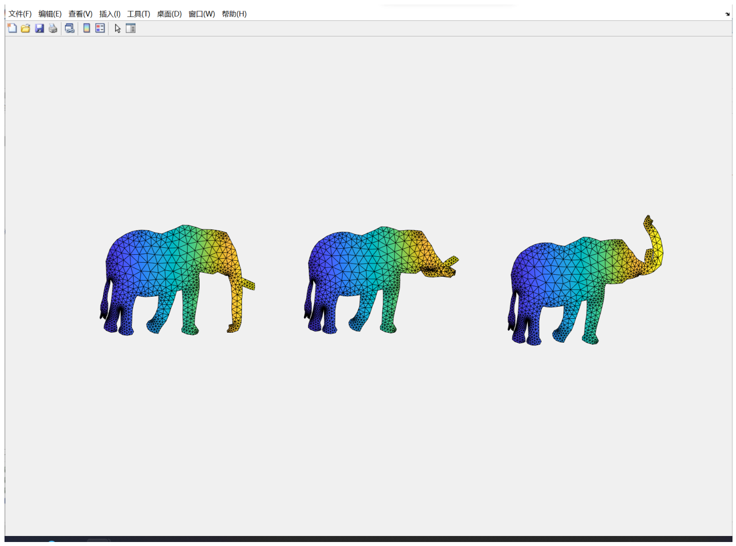
为了约束变换在原位置附近，在方程中添加一个控制点，原方程变为：

$$\begin{pmatrix} \mathbb{P}^T W \mathbb{P} & C^T \\ C & 0 \end{pmatrix} \mathbb{V} = \begin{pmatrix} \mathbb{P}^T W \mathbb{A} \\ D \end{pmatrix} \tag{13}$$

式中 $C = \begin{pmatrix} 1 & 0 & 0 & \cdots \end{pmatrix}$,D 为第一个顶点的坐标. 由于矩阵的尺寸变化，最终计算的 V 只取得到 $\mathbb{V}$ 的前 N 行.

2 算法实现

算法的编程实现几乎就是将上面所讲的翻译了一遍，没有特殊的地方，而且实现的效果为动态过程，这里也不方便展示. 但是注意到网格变换的过程中有很不自然的过程量，如图：这也是 bonus 部



分要解决的旋转一致性问题.

3 bonus: 旋转一致性

没时间，摆了。

参考文献

[1] M. Alexa et al. As-Rigid-As-Possible Shape Interpolation. SIGGRAPH 2000

[2] Baxter et al. Rigid shape interpolation using normal equations. Symposium on Nonphotorealistic animation and rendering 2008