第六次作业报告

PB21010362 汪兆辰

2023 年 4 月 29 日

1 算法介绍

为了在网格变形的过程中能够保持局部细节和结构，论文[1] 引入了 laplace 坐标来进行网格编辑. 不同与传统的坐标采用网格顶点的绝对位置，laplace 坐标考虑网格顶点间的相对位置. 对网格上每个点定义 laplace 坐标：

$$\delta_i = \mathcal{L}(v_i) \tag{1}$$

其中 $\mathcal{L}(v_i)$ 是关于顶点 v_i 及其邻接点的函数，根据权重选取的不同有不同的形式. 使用均匀权重时，laplace 坐标表示为：

$$\mathcal{L}(v_i) = v_i - \frac{1}{d_i}\sum_{j\in\mathcal{N}_i} v_j \tag{2}$$

选用 cot 权重时，laplace 坐标表示为：

$$\mathcal{L}(v_i) = v_i - \sum_{j\in\mathcal{N}_i} \frac{\cot\alpha_j + \cot\beta_j}{2} \tag{3}$$

其中 $\mathcal{N}_i$ 为与 v_i 邻接的点的集合，$d_i = |\mathcal{N}_i|$，α_j, β_j 为 $v_i v_j$ 所对的两个角.

得到所有点的 laplace 坐标 $\Delta = LV$ 后，可以在 Δ 上对顶点进行操作，但由于 rank(L)=n-1, 无法直接还原顶点坐标 V，于是考虑固定部分控制点，并定义能量：

$$E(V') = \sum_{i=1}^{n} \|\delta_i - \mathcal{L}(v_i')\|^2 + \sum_{i=m}^{n} \|v_i' - u_i\|^2 \tag{4}$$

只需求解 $E(V')$ 的最小值，即可得到经过操作后的网格. 这可以转化为求解以下方程的最小二乘解：

$$\begin{pmatrix} L \\ c \end{pmatrix} V = \begin{pmatrix} \Delta \\ c' \end{pmatrix} \tag{5}$$

其中 L,V 定义如前文所述，c 为控制点的位置，c' 为控制点的坐标.

2 算法实现

算法的实现部分用 c++ 建立了 laplace 算子矩阵 L，并将计算得到的 L 及其他需要的数据传入 matlab，利用 matlab 脚本计算方程 (5) 的最小二乘解.

2.1 网格变形

利用均匀权重的 Laplace 算子可以通过网格的邻接矩阵经过矩阵运算得出：

$$L = I - D^{-1}A \tag{6}$$

其中 A 为网格的邻接矩阵，D 是 d_i 构成的对角阵，容易从邻接矩阵直接计算出. 由于邻接矩阵是稀疏的，考虑采用 Eigen 库的 SparseMatrix，利用三元组进行赋值. 三元组的建立需要遍历读入网格时得到的顶点间的拓扑关系 meshF，其中每一行为一个三角形面的三个顶点标号，对每个三角形面按顺时针、逆时针各遍历一遍，就得到了网格的邻接矩阵. D 只需对 A 的每一行求和并化为对角阵即可，这样就得到了 Laplace 矩阵 L.

方程 (5) 的建立比较平凡，因为控制点的坐标已经在 P2PVtxIds 中给出，只需取出对应的点填入矩阵方程对应位置即可. 令方程 (5) 为 $Az = b$, z 的最小二乘解可以由：

$$A^TAz = A^Tb \tag{7}$$

给出，z 的前 n 行即为编辑后的网格坐标.

2.2 利用 cot 权重

采用 cot 权重对 laplace 算子赋值选择了直接对空的稀疏矩阵对应位置赋值. 对每个三角形面，可以得到每条边对应的两个角的其中一个，因而考虑对建立的 Cot_Mat 的每个位置赋两次值，这需要允许赋值的位置非空，因此利用 coeffRef 方法，例如：

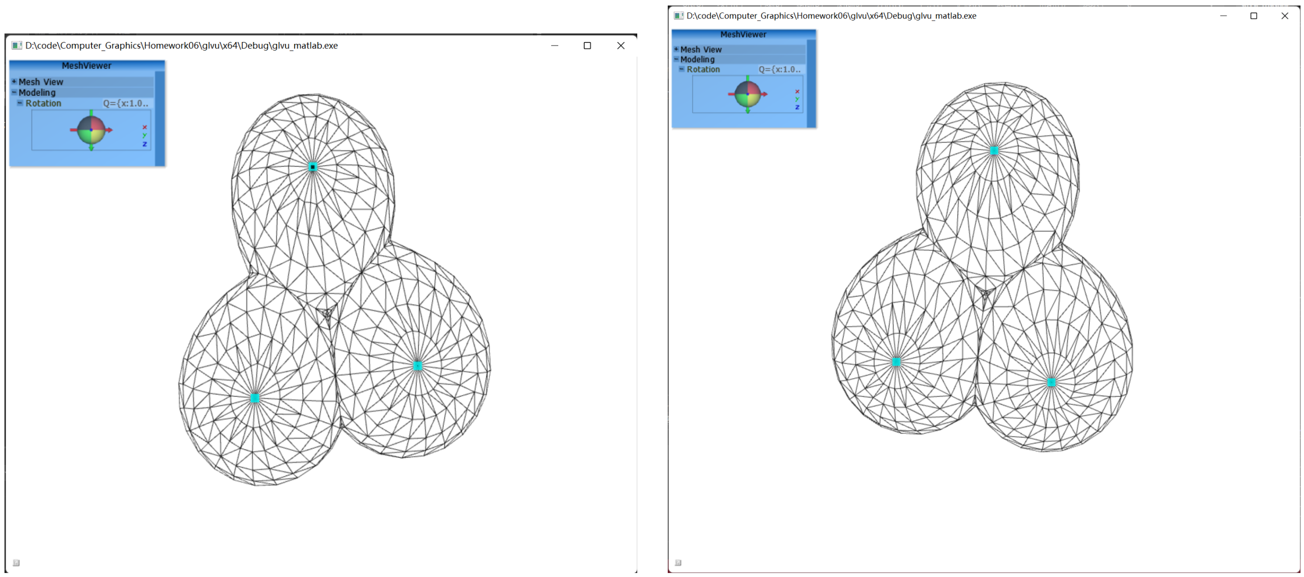
```
Cot_Mat.coeffRef(temp(0,j),temp(0,(j+1)%3))+=Cot(...)/2;
```

注意到没有现有的函数可以直接根据三角形的三个顶点位置给出 cot 值，因而在函数中自定义 lambda 表达式：

```
auto Cot = [X](int i, int j,int k) {
    Eigen::Vector3f ki=X.row(i)-X.row(k);
    Eigen::Vector3f kj=X.row(j)-X.row(k);
    float cosVal = ki.dot(kj) / (ki.norm() * kj.norm());
    float angle = acos(cosVal);
    return 1/tanf(angle);
};
```

该 lambda 表达式值捕获了函数中的顶点坐标 X，输入参数 i,j,k 为三个顶点的标号，返回边 ij 所对的角的 cot 值.

2.3 实现效果



(a) uniform　　　　(b) cot

3 bonus:laplace 坐标下的旋转操作

由于在 laplace 坐标下，旋转操作前后的局部细节无法很好的保持，于是仍然考虑对定义的能量找到最小二乘解，从而确定变换矩阵. 在这里定义了能量：

$$E(V') = \sum_{i=1}^{n} \|T_i(V')\delta_i - \mathcal{L}(v_i')\|^2 + \sum_{i=m}^{n} \|v_i' - u_i\|^2 \tag{8}$$

为了求解这一方程的最小二乘解，将其转化为类似于方程 (5) 的形式：

$$\begin{pmatrix} L \\ c \end{pmatrix} V = \begin{pmatrix} \Delta' \\ c' \end{pmatrix} \tag{9}$$

其中 Δ' 为将控制点旋转后的 laplace 坐标.

在这里旋转是由用户操作返回的四元数组 myRotation 定义的，将其传入 matlab 并调用 quat2dcm 函数，从而得到对应的三维旋转矩阵，再将 Δ 中控制点右乘该矩阵即可，其他求解过程相同. 为了能够实时生成旋转后的图像，需要用到回调函数实时对更新的 myRotation 调用方法，(由于没有看懂老师讲的 TwAddVarCB 里面 setCallBack 和 getCallBack 是什么东西) 考虑使用 GL 中的 glutTimeFunc 来登记回调函数，其中回调函数定义如下：

```
void rotationcallback(int) {
    if (...) {\\myRotation has been updated
        for (int i = 0; i < 4; i++)recRotation[i] = myRotation[i];
        meshDeform();
    }
    glutTimerFunc(1, rotationcallback, 1);
}
```

其中利用 recRotation 来判断用户是否更新了 myRotation.

参考文献

[1] O. Sorkine et al. Laplacian Surface Editing. SGP 2004

[2] https://cloud.tencent.com/developer/article/1745166