Lab2 实验报告

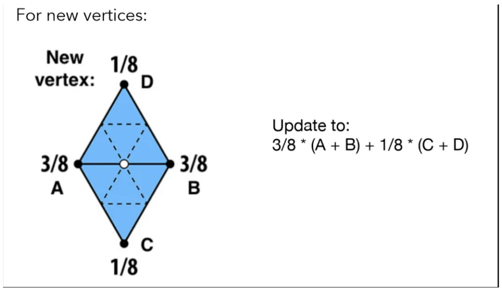
2100012983 刘逸兴

2022 年 12 月 4 日

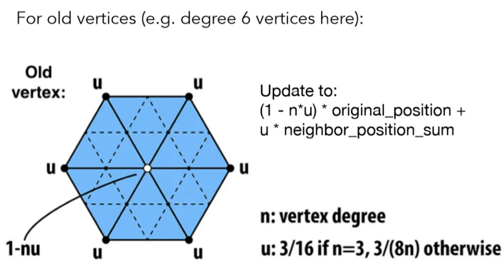
1 Loop Mesh Subdivision

算法大致流程:

1. 遍历每条边，在每条边上添加新顶点。如果边是边界，则新顶点坐标为两端点坐标平均值。否则如图设置坐标。
2. 对于每个旧顶点，如图设置坐标。
3. 按照顺序更新 Indices 数组, 维护 Mesh 结构。
4. 重复上述步骤直到达到迭代次数



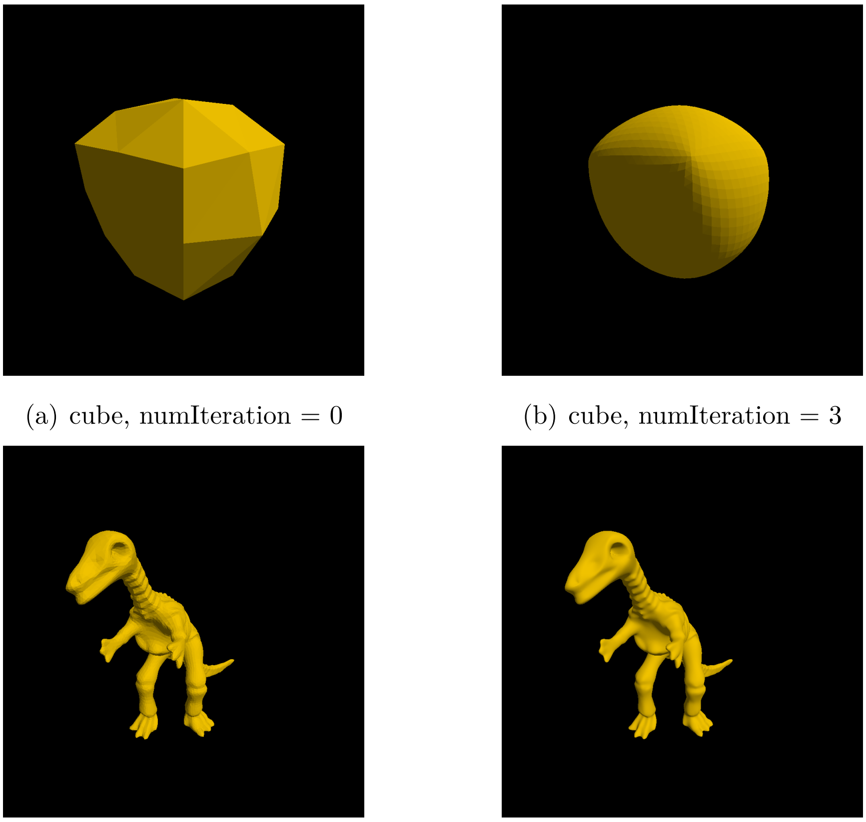
(a) new vertices



(b) old vertices

图 1: Loop Subdivision

得到结果如图:



(a) cube, numIteration = 0
(b) cube, numIteration = 3
(c) dinosaur, numIteration = 0
(d) dinosaur, numIteration = 3

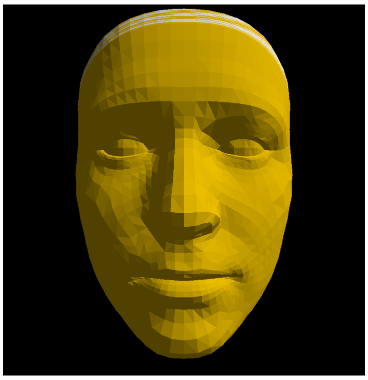
图 2: Mesh Subdivision

2 Spring-Mass Mesh Parameterization

算法大致流程:

1. 初始化边界点上的 uv 坐标，这里我用的是圆边界。对于边界点 $p(x,y,z)$, 初始化坐标为 $u=\frac{1}{2}+\frac{x}{2\sqrt{x^2+y^2}}, v=u=\frac{1}{2}+\frac{y}{2\sqrt{x^2+y^2}}$
2. 新建数组，存储所有的内部点。
3. 按照论文中的方法设置 $\vec{b}, b_i=(\sum_{\text{j 与 i 相连且 j 是边界点}}\lambda_{ij}u_j, \sum_{\text{j 与 i 相连且 j 是边界点}}\lambda_{ij}v_j$
4. 迭代初值均设为 $(0,0)$
5. 利用 Jacobi 迭代求解，其中系数矩阵同样按照论文中的方法设置。

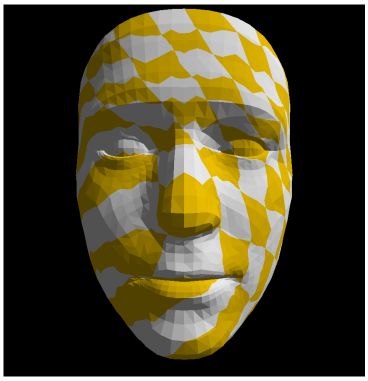
为了方便起见，设 λ_{ij} 为 i 的邻居点的个数的倒数。
得到结果如图:



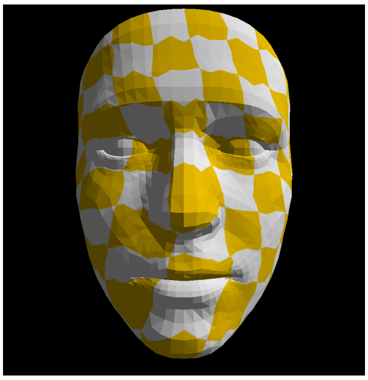
(a) numIteration = 0

(b) numIteration = 25

(c) numIteration = 100



(d) numIteration = 400



(e) numIteration = 1000

图 3: Mesh Parameterization

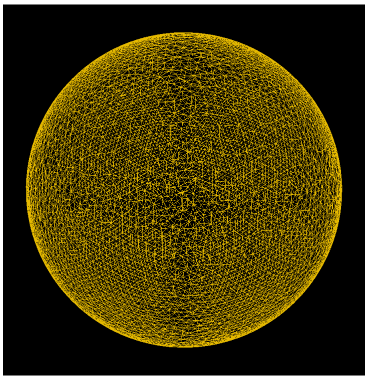
3 Mesh Simplification

算法大致流程:

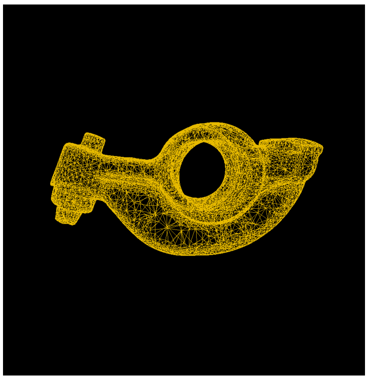
1. 对于每个顶点，按照论文中的方法计算其 Q 矩阵 $Q_i = \sum_{p \in planes(v_i)} (\mathbf{p}^T \mathbf{v})^2$
2. 选取所有合法点对: 相邻顶点或者距离小于阈值的点对
3. 按照论文中的方法计算每对点对的最佳替代点 $\bar{v}$, 以及损失 $cost = \bar{v}^T (Q_1 + Q_2) \bar{v}$
4. 选择 $cost$ 值最小的点对进行替换，同时更新 output 中的各个数组以及存储 Q 矩阵和最佳点对的数组。

5. 不断重复第四步，直到满足迭代终止条件。

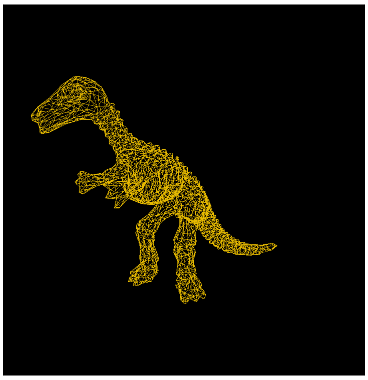
由于没有使用堆结构来维护所有合法点对中 $cost$ 最小的点对，因此该算法时间复杂度较大，同时在第四步更新各个数组的过程中，也没有采取特殊的数据结构，因而常数也较大。因此对于简化比例较小的情况，只挑选了节点数较小的恐龙图来展示最终的结果。当然，由于恐龙图的拓扑结构较为复杂，这一结果也具有较大的意义。



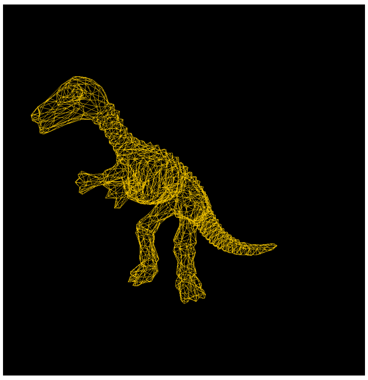
(a) sphere,simplification ratio = 0.8



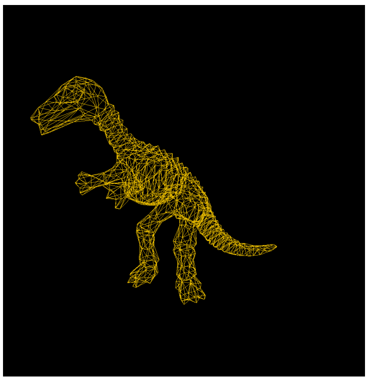
(b) rocker,simplification ratio = 0.8



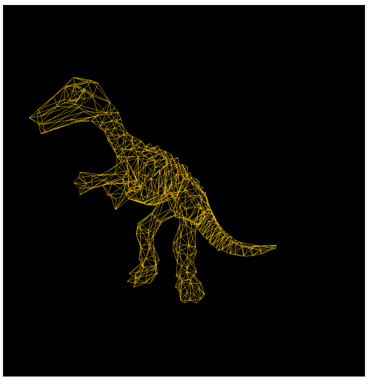
(c) dinosaur,simplification ratio = 0.8



(d) dinosaur,simplification ratio = 0.6



(e) dinosaur,simplification ratio = 0.4



(f) dinosaur,simplification ratio = 0.2

图 4: Mesh Simplification

4 Mesh Smoothing

算法大致流程:

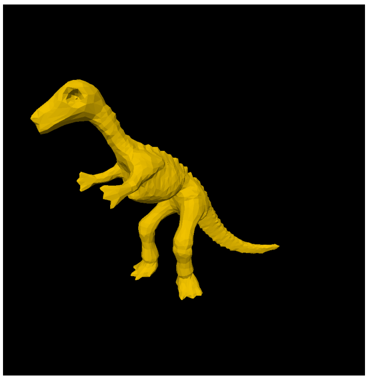
1. 对于每个顶点，计算邻居位置的加权平均
$$v_i^* = \frac{\sum_{j \in N(i)} w_{ij} v_j}{\sum_{j \in N(i)} w_{ij}}$$

2. 其中使用 Uniform Laplacian 时 $w_{ij} = 1$ ；使用 Cotangent Laplacian 时 $w_{ij} = \cot \alpha_{ij} + \cot \beta_{ij}$. 在计算 $\cot \alpha_{ij}$ 和 $\cot \beta_{ij}$ 时，先计算出两个角的余弦值。而经过尝试，直接用两个向量的点乘直接代替余弦值效果更好。

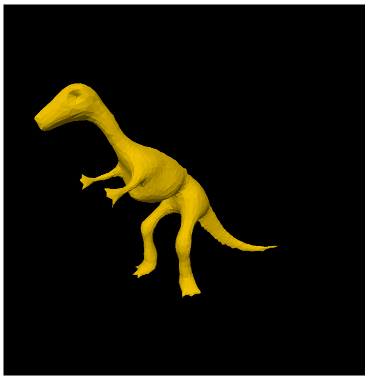
3. 更新顶点：$v_i = (1 - \lambda) v_i + \lambda v_i^*$

4. 重复上述步骤直到达到迭代次数

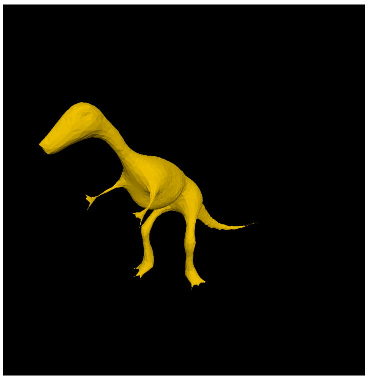
得到结果如图:



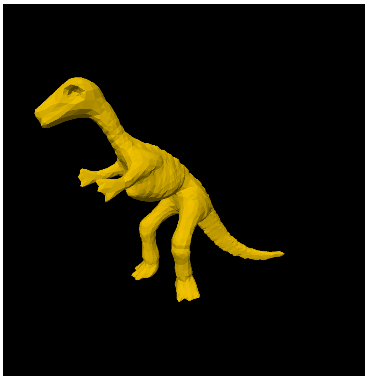
(a) Cotangent, numIteration=1,λ=0.7



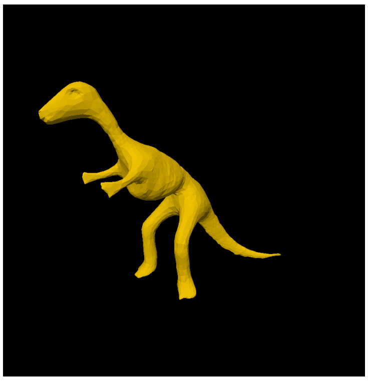
(b) Cotangent, numIteration=5,λ=0.7



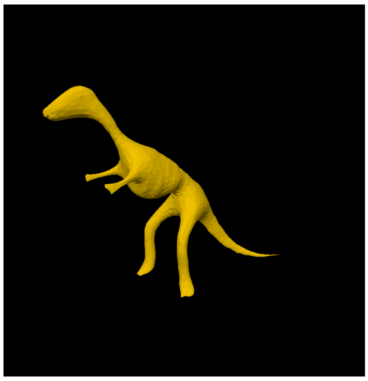
(c) Cotangent, numIteration=10,λ=0.7



(d) Uniform, numIteration=1,λ=0.7



(e) Uniform, numIteration=5,λ=0.7



(f) Uniform, numIteration=10,λ=0.7

图 5: Mesh Smoothing

可以看到使用 Cotangent Laplacian 得到的结果相比与使用 Uniform Laplacian 得到的结果更加尖锐，在爪子、脚趾处结构保留更好，但是手臂、腿明显更为退化。

5 Marching Cubes

算法大致流程:

依次对每一个小立方体执行以下步骤：

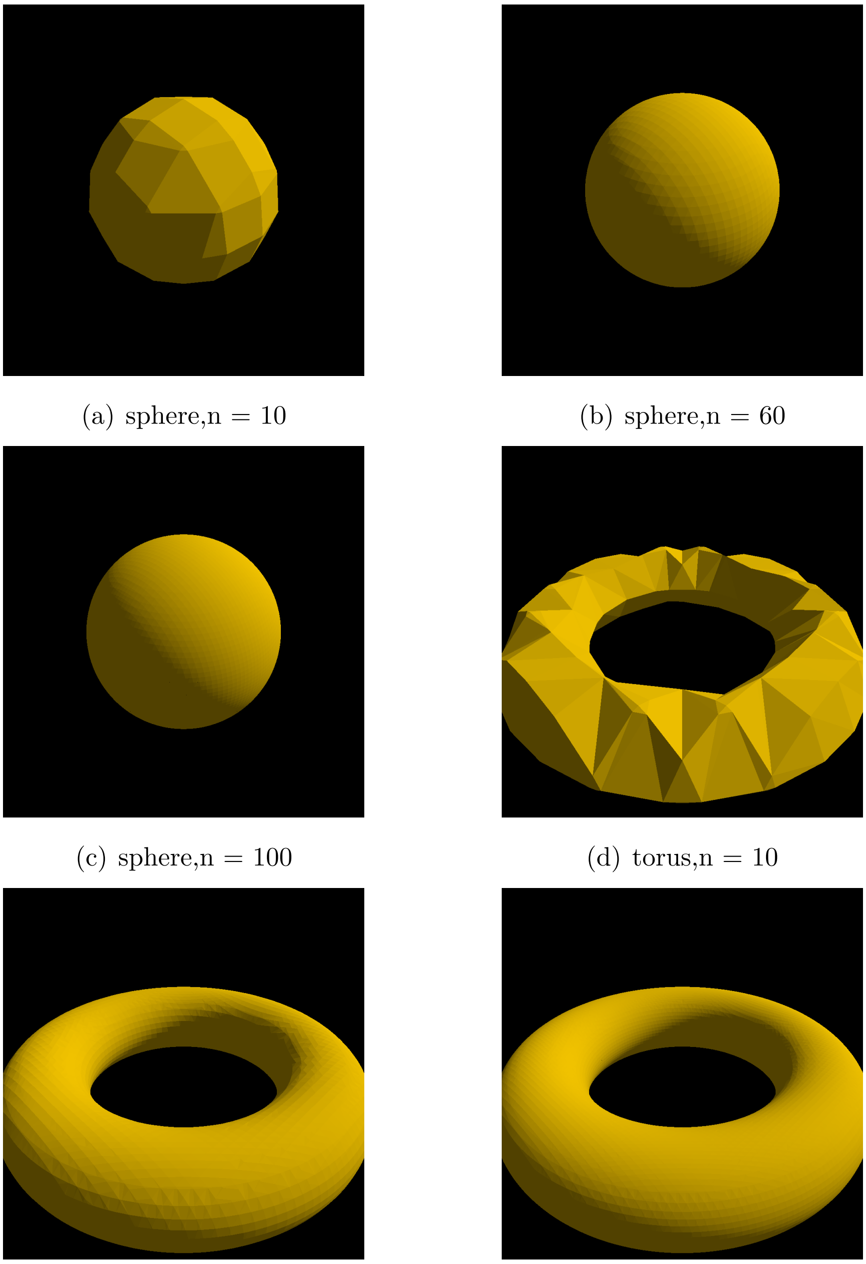
1. 计算该立方体的 $index$: 如果编号为 i 的顶点到隐式表面的有向距离小于等于 0, 则 $index$ 的第 i 位为 1, 否则为 0.

2. 如果 $index = 0or255$, 说明这个立方体与隐式表面没有交点，前进到下一个立方体。

3. 否则找到 $c_EdgeOrdsTable[Index]$ 数组，里面的数字按顺序每三个一组，为隐式表面与立方体相交得到的三角形顶点所在的边，按照线性插值公式 $P = P_0 + \frac{-sdf(P_0)}{sdf(P_1)-sdf(P_0)} \times (P_1 - P_0)$ 得到该顶点的位置.

4. 若该点不在 Positions 数组中，则插入. 更新 Indices 数组，并前进到下一个立方体.

得到结果如图:



(a) sphere,n = 10　(b) sphere,n = 60

(c) sphere,n = 100　(d) torus,n = 10

(e) torus,n = 60　(f) torus,n = 100

图 6: Marching Cubes