# 网格简化实验报告

基科 11 刘心宇 2011012135

2014 年 9 月 25 日

## 目 录

# 1 实验目的

实现基于二次误差度量的网格简化算法。[1]

# 2 实验内容

导入三角网格模型（obj 文件），按一定比例进行简化，并输出。

# 3 实验原理

Hoppe 的边收缩 (edge collapse) 操作[2,3] 可推广为一般的顶点合并变换来描述 $(v_1,v_2)\to v$，其含义是将场景中的两个顶点 $(v_1,v_2)$ 移到一新的位置 $v$，将连向 $(v_1,v_2)$ 的所有边都连向 $v$，并删除所有退化的边和面片。

## 3.1 点对合并

其中点对 $(v_1,v_2)$ 合并的原则

1) $(v_1,v_2)$ 为某一表面上的相邻点，即 $(v_1,v_2)$ 为一条边，如图 1

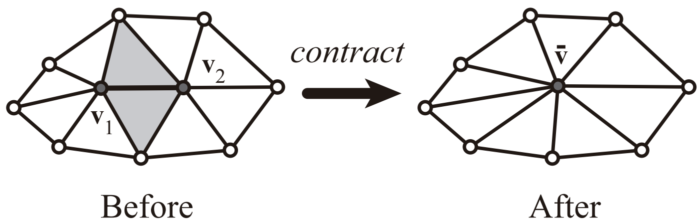



图 1: Edge contraction

2) $||v_1-v_2||<t$，$t$ 为用户给定的阈值参数，如图 2

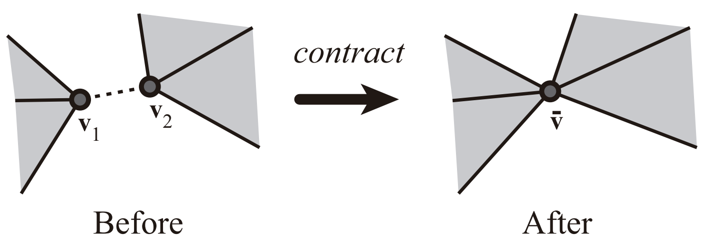



图 2: Non-edge contraction

## 3.2 二次误差度量

Garland 和 Heckbert 引进了二次误差度量来刻划每一个顶点移动后引起的误差。对表面上的每一个顶点均有许多三角面片与之相邻，记 $plane(v_a)$ 为这些三角形所在的平面方程所构成的集合，即

$$plane(v_a)=\left\{(a,b,c,d)\;\middle|\;\begin{array}{l}ax+by+cz+d=0,(a^2+b^2+c^2=1)\\ \text{is coefficients of the adjacent plane of } v_a\end{array}\right\} \tag{3.1}$$

则我们采用如下的二次函数来度量 $v_a$ 移动到 $v$ 时产生的误差

$$\Delta(v_a\to v)=\sum_{p\in plane(v_a)}(pv^T)^2 \tag{3.2}$$

其中 $v=(x,y,z,1)$ 为齐次坐标。展开公式 (3.2)得到

$$\Delta(v_a\to v)=\sum_{p\in plane(v_a)}(pv^T)^2=v(\sum_{p\in plane(v_a)}K_p)v^T=vQ(v_a)v^T \tag{3.3}$$

式 (3.3)中

$$K_p = p^T p = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix} \tag{3.4}$$

这样，对每一顶点 $v_a$，在预处理时，我们均可按上述方法计算矩阵 $Q(v_a)$，进而就可对其移动进行误差度量了。但由于每次合并时，需同时移动两点，故必须考虑同时移动多个顶点后形成的误差。

### 3.3 多点移动的误差

Garland 和 Heckbert 简单地采用加法规则来刻划多点移动而形成的误差，对点对合并 $(v_1, v_2) \to v$，其误差为 $\Delta(v) = \Delta(v_1 \to v) + \Delta(v_2 \to v) = v(Q(v_1) + Q(v_2))v^T = vQv^T$, 其中。因而，应选取 $v$ 使误差达到最小。

由极值的性质知，$v$ 满足系统方程

$$\frac{\partial \Delta(v)}{\partial x} = \frac{\partial \Delta(y)}{\partial y} = \frac{\partial \Delta(z)}{\partial z} = 0$$

即

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} v = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.5}$$

若式 (3.5)有唯一解，则其解为 $v$ 的最优解。否则，用伪逆技术求 $v$。若伪逆技术失败，则简单地选取 $v$ 为 $(v_1, v_2)$ 或者 $\frac{v_1 + v_2}{2}$ 中的任何一个。

## 4 编程实现

### 4.1 算法描述

基于二次误差度量的网格简化算法如下

1) 计算所有顶点的二次误差度量矩阵 $Q$

2) 选择所有符合 3.1中条件的顶点对 $P_{ij} = (v_i, v_j)$。

3) 计算每一个顶点对 $P_{ij}$ 最优的目标顶点 $\overline{v_{ij}}$。将误差 $C_{ij} = \overline{v_{ij}}^T (Q_i + Q_j)\overline{v_{ij}}$ 作为收缩该顶点对 $P_{ij}$ 的代价（cost）

4) 按照顶点对的代价 $C_{ij}$，生成最小堆（heap）[4]

5) 收缩代价最小的顶点对 $P_{ij}$，移除多余的顶点，更新包含 $v_i$ 和 $v_j$ 的顶点对的代价 $C_{im}, C_{nj}$。反复执行，直到满足简化条件

### 4.2 文件结构

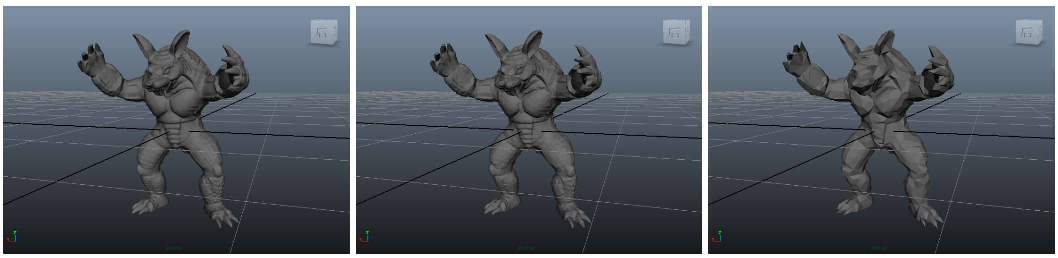
| | |
|---|---|
| .\SimpleObject.h | 定义辅助类，包括误差矩阵 QMat，堆 Heap 等 |
| .\SimpleObject.cpp | 实现网格简化算法的主要代码 |
| .\XYTime.cpp | 计时模块 |
| .\Main.cpp | 测试用例 |

//网格模型解析（助教提供，有部分改动）
.\SimpleObject.h
.\Vec3f.h
.\SimpleObject.cpp
.\Vec3f.cpp

### 4.3 使用说明

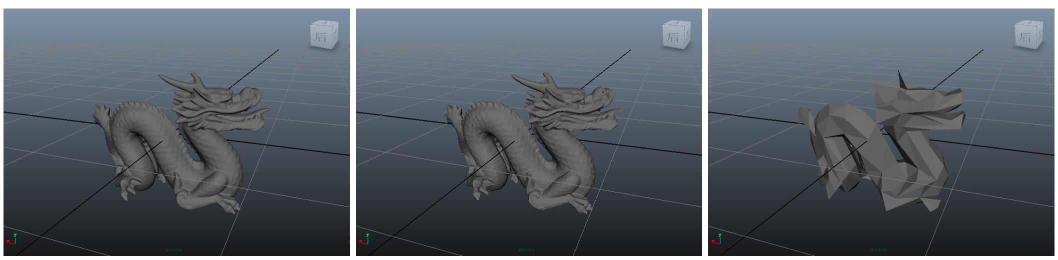
```
<inputfile> <option>
available parameters
-r <rate> : Set the coefficient of simplification
-h        : Show help
```

## 5 效果演示



(a) ×1.0 46398 faces　　(b) ×0.2 3.973s　　(c) ×0.05 4.112s

图 3: 怪兽



(a) ×1.0 209227 faces　　(b) ×0.05 107.7s　　(c) ×0.002 107.9s

图 4: 龙

## 6 实验总结

本次实验实现了基于二次误差度量的网格简化算法，了解了网格简化的基本思想，也实现了非边的顶点对的合并（阈值设为整个模型尺寸的 0.1 倍）。

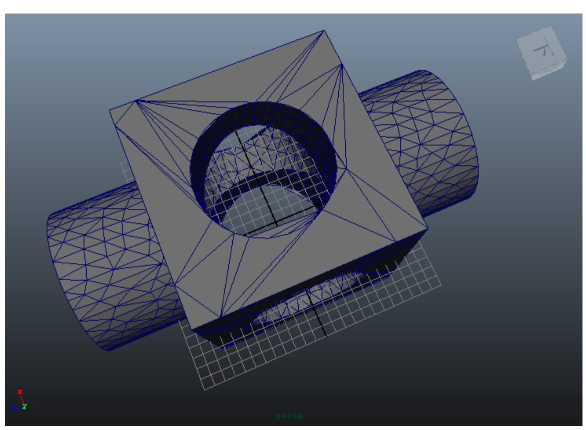


图 5: 存在的问题

当然出现了一些问题，浮点误差导致的 bug，堆中的边收缩和模型中的边收缩没有同步导致的 bug 等。除了已经解决的问题，还有一些问题没有解救，例如在比较特殊的模型中，边收缩会导致一些奇怪的现象，如图 5，在中间圆形的边缘，效果不是很好，还没有得到很好的解决。有关网格简化，还有一些更好的方法，希望以后可以继续学习。最后感谢老师和助教的帮助。

## 参考文献


[1] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/ Addison-Wesley Publishing Co., 1997.

[2] H. Hoppe. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108. ACM, 1996.

[3] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 19–26. ACM, 1993.

[4] C. E. Leiserson, R. L. Rivest, C. Stein, and T. H. Cormen. *Introduction to algorithms*. The MIT press, 2001.