基于边收缩的二次误差网格简化

计 73 王焱 2017050024

本次实验是对一篇 97 年的 SIGGRAPH 论文 *Surface Simplification Using Quadric Error Metrics* 的算法实现，输入一个 obj 模型，按照给定的简化比，通过计算二次误差选择边收缩进行网格简化，输出一个简化后的模型 obj 文件。

一、算法思路

核心思想就是给一个 pair 收缩到某点 v 定义了一个 cost 值，即收缩代价，按这个代价值来维护一个小根堆，迭代的从堆中弹出 pair 进行收缩，并将新生成的 pair 按计算出的 cost 插入到堆中，直到达到给定的简化比为止。

其中对于 pair 的选取，我在算法中取的是网格里有边的两点，除了这种取法，还可以按照两点的距离来选取。定义一个收缩的 Cost=v^TQv，Q 为了便于计算取的是两点的误差矩阵之和 Q_1+Q_2，v 为收缩后的一个点，可通过求导等于零计算其最佳位置。

在原始网格模型中，每个顶点可以认为是其周围三角片所在平面的交集，也就是这些平面的交点就是顶点位置，定义顶点的误差为顶点到这些平面的距离平方和：

The error metric given in (2) can be rewritten as a quadratic form:

$$\begin{aligned}\Delta(\mathbf{v}) &= \sum_{\mathbf{p}\in\text{planes}(\mathbf{v})} (\mathbf{v}^{\mathsf{T}}\mathbf{p})(\mathbf{p}^{\mathsf{T}}\mathbf{v}) \\ &= \sum_{\mathbf{p}\in\text{planes}(\mathbf{v})} \mathbf{v}^{\mathsf{T}}(\mathbf{p}\mathbf{p}^{\mathsf{T}})\mathbf{v} \\ &= \mathbf{v}^{\mathsf{T}}\left(\sum_{\mathbf{p}\in\text{planes}(\mathbf{v})} \mathbf{K}_{\mathbf{p}}\right)\mathbf{v}\end{aligned}$$

where $\mathbf{K}_{\mathbf{p}}$ is the matrix:

$$\mathbf{K}_{\mathbf{p}} = \mathbf{p}\mathbf{p}^{\mathsf{T}} = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

式中$\sum K_p$就是该点的误差矩阵。

二、实现过程

代码中我定义了顶点类 Vertex、收缩点类 Pair 和 4 阶矩阵类 Matrix4，来存储数据。核心函数有以下几个：

```
void calculateBest(Pair& pair)
```

```
int temp=readFile(inputfile.c_str());
if(temp!=0)cntFace=temp;else {cout<<"open fail"<<endl;return 0;}
cntDelFace=(1-ratio)*cntFace;

calQ();
buildHeap();
simplify();
saveFile(outputfile.c_str());
```

首先调用 readfile(inputfile.c_str())读入 obj 文件，判断不同标签，将顶点存入 Vertex 类的 vertex 数组中，面片信息存储为点的邻接关系即 Vertex 的 connectV 成员中的元素。

其次调用 calQ()计算所有顶点的误差矩阵。遍历得到一个顶点周围所有三角面片，参考以下求解方法，计算平面方程的 a,b,c,d 四个参数。

已知三点p1（x1,y1,z1），p2(x2,y2,z2)，p3(x3,y3,z3)，要求确定的平面方程

关键在于求出平面的一个法向量，为此做向量p1p2（x2-x1,y2-y1,z2-z1), p1p3(x3-x1,y3-y1,z3-z1),平面法线和这两个向量垂直，因此法向量n:

$$\vec{n} = \dot{p_1p_2} \times \dot{p_1p_3} = \begin{vmatrix} i & j & k \\ x_2-x_1 & y_2-y_1 & z_2-z_1 \\ x_3-x_1 & y_3-y_1 & z_3-z_1 \end{vmatrix} = ai+bj+ck = (a,b,c)$$

$$a = (y_2-y_1)*(z_3-z_1)-(y_3-y_1)*(z_2-z_1)$$
$$b = (z_2-z_1)*(x_3-x_1)-(z_3-z_1)*(x_2-x_1)$$
$$c = (x_2-x_1)*(y_3-y_1)-(x_3-x_1)*(y_2-y_1)$$

平面方程：a(x-x1)+b(y-y1)+ c(z-z1)=0;

d=-a*x1-b*y1-c*z1。

平面平面方程为ax+by+cz+d=0。

然后利用上面对于误差矩阵的定义及推导，可计算出误差矩阵。这里误差矩阵的存储，我采用顶点类中的矩阵类指针进行存储。

然后调用 buildHeap()开始建堆。利用 C++STL 标准库中的优先队列来实现小根堆

```
struct cmp
{
    bool operator() ( Pair a, Pair b ){ return a.cost> b.cost; }
};
priority_queue <Pair,vector<Pair>,cmp> pairHeap;
```

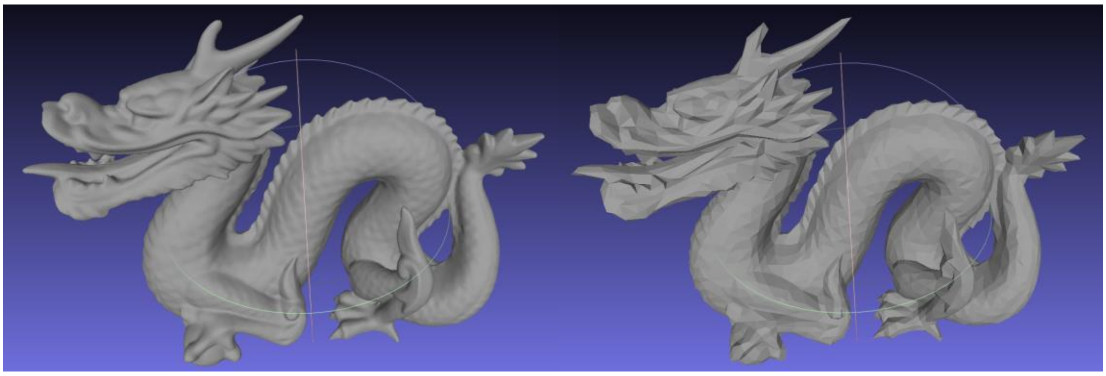
遍历所有的边，调用 calculateBest(pair)函数，计算最佳收缩位置 v 及其收缩代价，push 进堆。这里涉及到的计算主要是高斯消元法解方程组。

最后调用 simplify()进行迭代收缩直到到达指定简化比次数或堆中为空。每次从堆中 pop 得到一个有效的收缩边，对其进行收缩。将两个点的所有临接点都转移给收缩后的点，具体就是原来与两点连接的边记为删除，与对应的计算产生的与收缩后的点连接的边的收缩位置和代价，push 进堆。

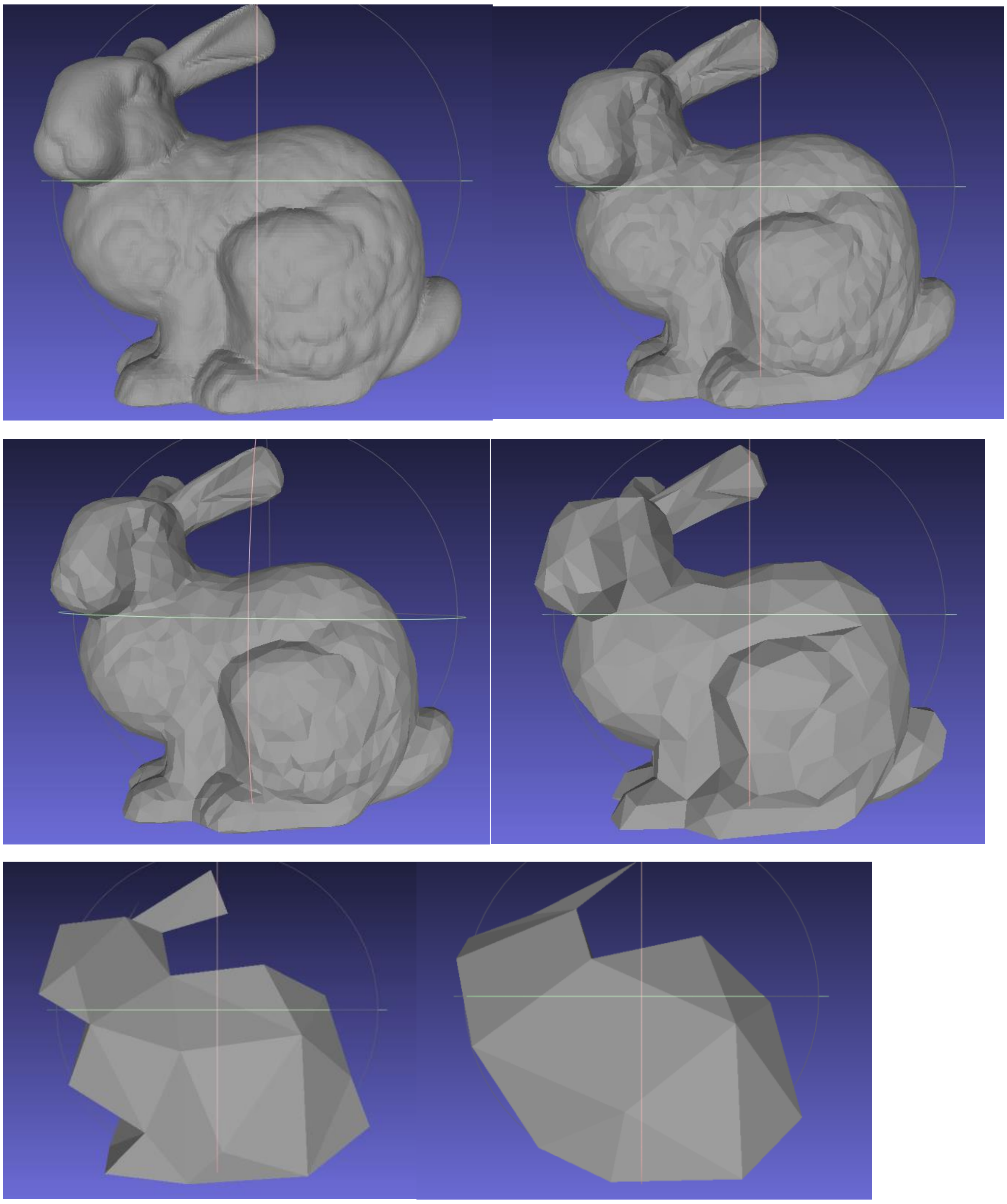
最后按照 obj 文件格式将堆中有效（未被删除）点和边写入文件。

三、成果展示

原模型和检查的 0.03 简化比的 dragon.obj



对同一模型简化比为 0.1，0.05，0.01，0.001，0.0005 时的简化模型



四、实验总结

算法的整体思想上比较容易理解，可以拆分成几个明确的步骤，涉及到一些数学运算，需要编程实现，但是都比较经典的方法，我查资料学习了一下。这次出 bug 的地方主要是有两处，一个是在简化操作时要注意的是，需要将点的关系都更新完毕，才能计算 cost 值，另一个是高斯消元法，需要仔细模拟高斯消元法的过程。

这是我第一次论文实现的编程体验，我首先读了论文，学习算法思想步骤，然后有又参考了网上其他人写的一些代码，也有不同的具体实现方法，综合这些之后，又回归到论文本身有了更深的体会，才写出了自己的代码。感觉这种从算法的原论文出发进行学习，较之以前的各种只从其他的博客教程学习要更透彻一些，总之是一次较好的体验。