# 基于边收缩的二次误差网格简化

计73 王焱 2017050024

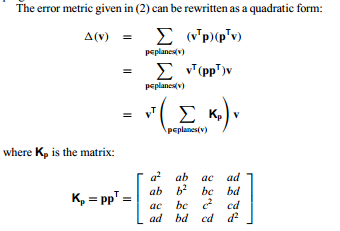
本次实验是对一篇97年的SIGGRAPH论文*Surface Simplification Using Quadric Error* *Metrics*的算法实现，输入一个obj模型，按照给定的简化比，通过计算二次误差选择边收缩进行网格简化，输出一个简化后的模型obj文件。

## 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=vTQv，Q为了便于计算取的是两点的误差矩阵之和Q­1+Q2，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：

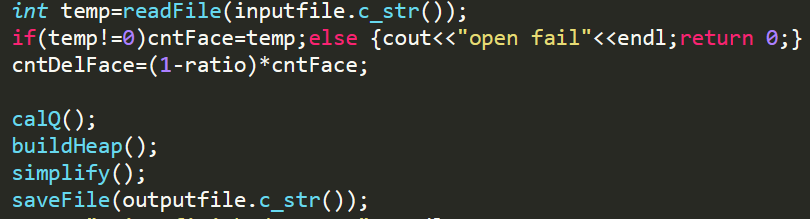


式中就是该点的误差矩阵。

## 二、实现过程

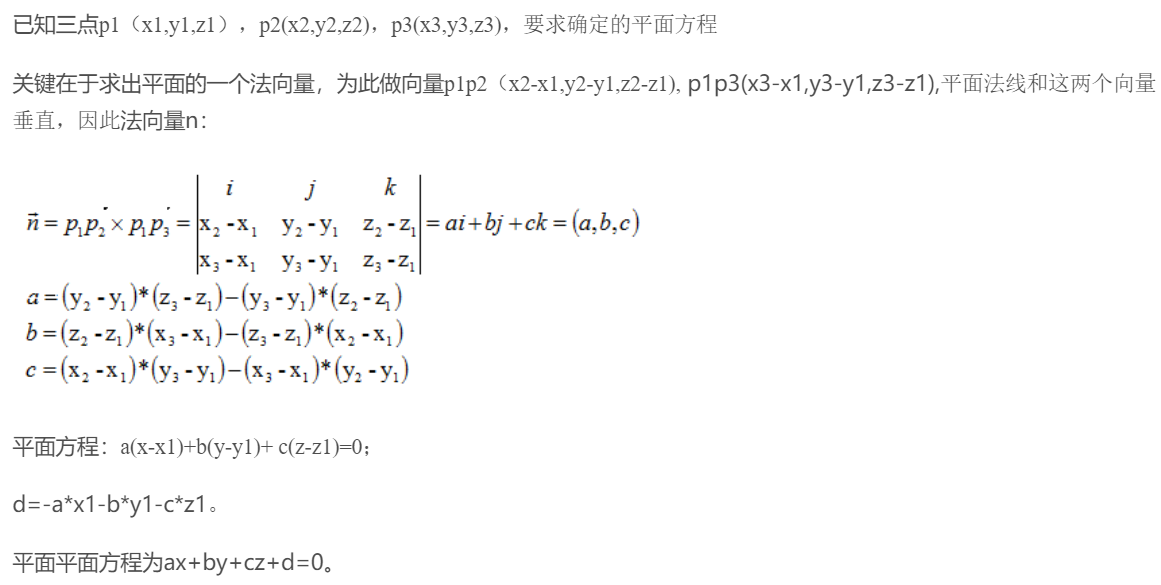
代码中我定义了顶点类Vertex、收缩点类Pair和4阶矩阵类Matrix4，来存储数据。核心函数有以下几个：



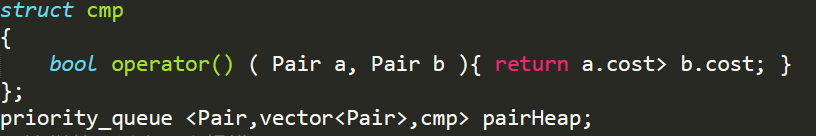


首先调用readfile(inputfile.c\_str())读入obj文件，判断不同标签，将顶点存入Vertex类的vertex数组中，面片信息存储为点的邻接关系即Vertex的connectV成员中的元素。

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。



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

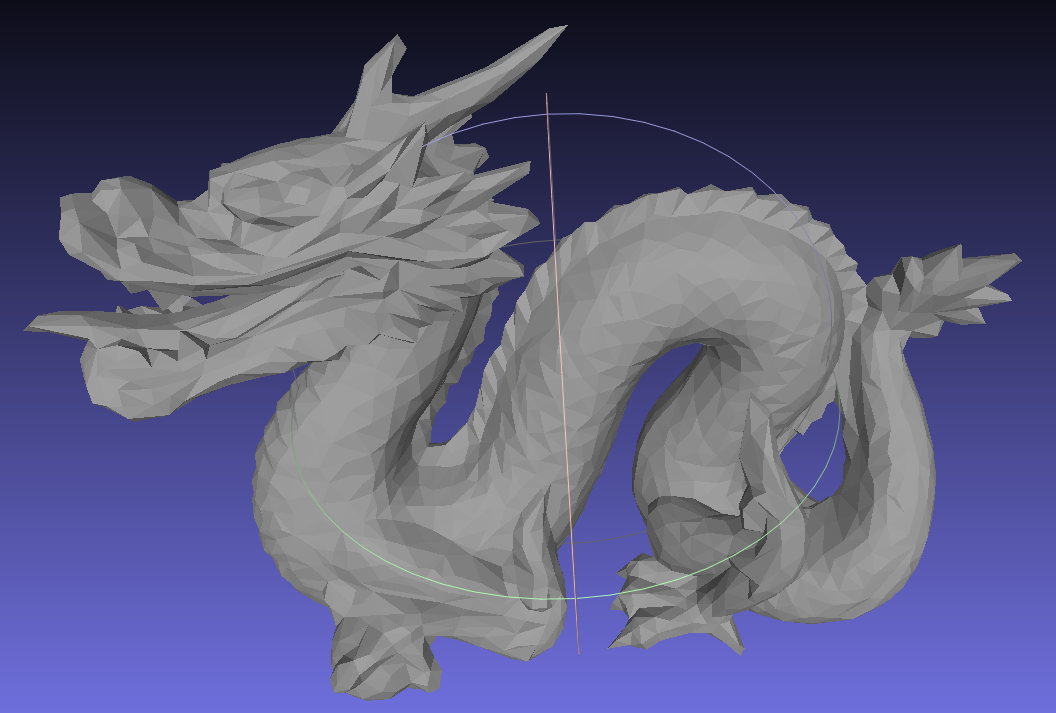
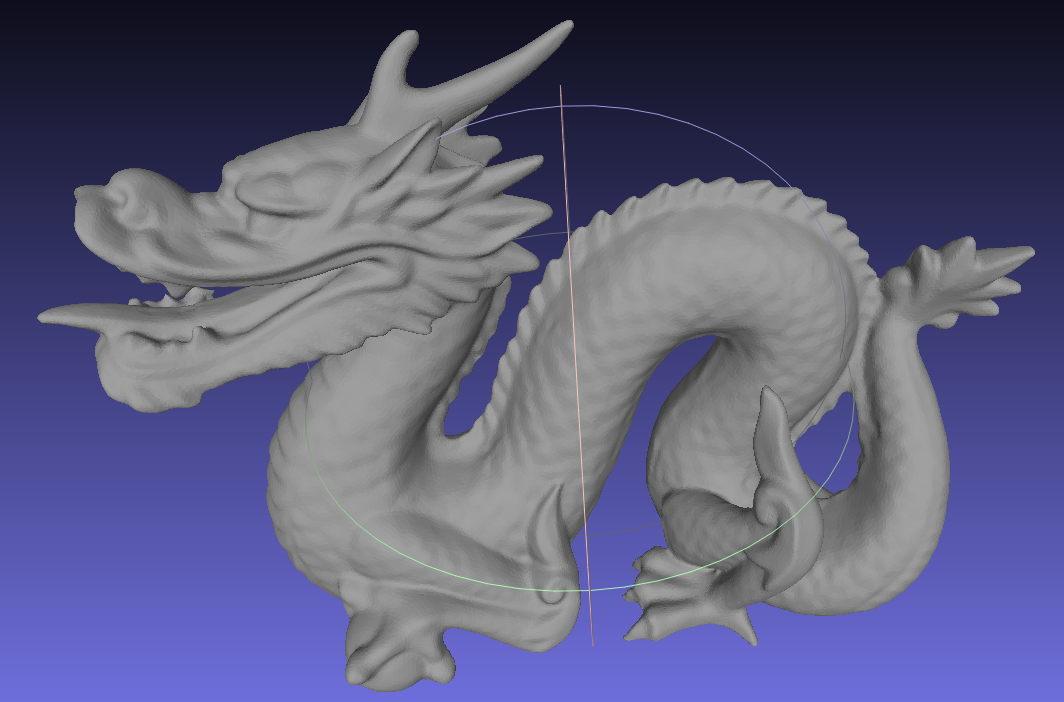
遍历所有的边，调用calculateBest(pair)函数，计算最佳收缩位置v及其收缩代价，push进堆。这里涉及到的计算主要是高斯消元法解方程组。

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。

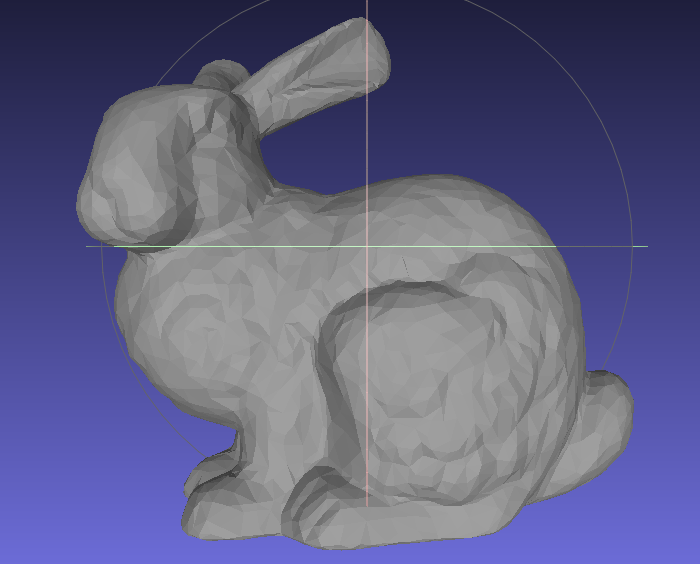
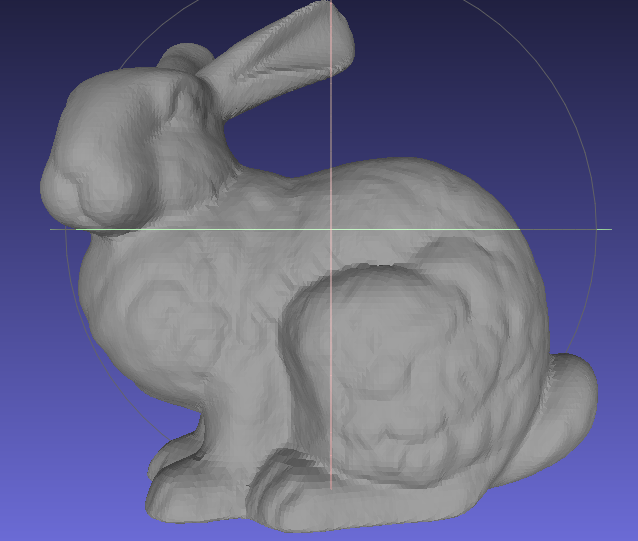
最后按照obj文件格式将堆中有效（未被删除）点和边写入文件。

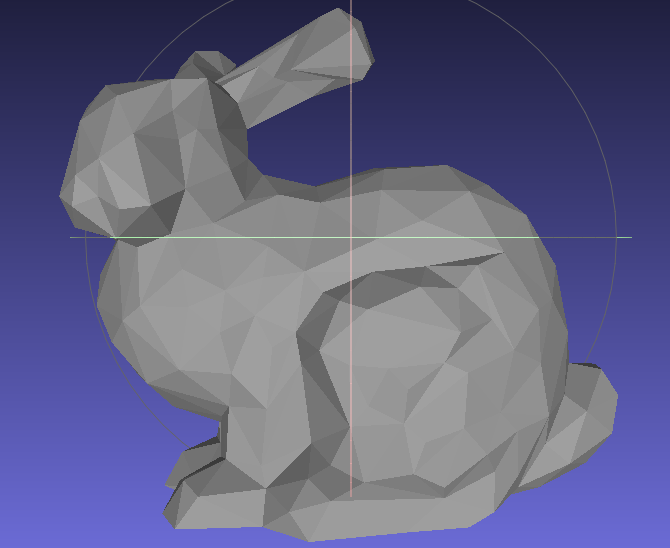
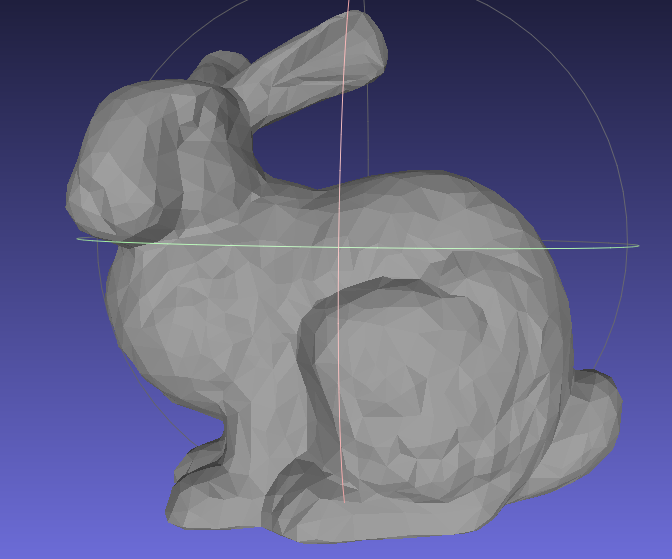
## 三、成果展示

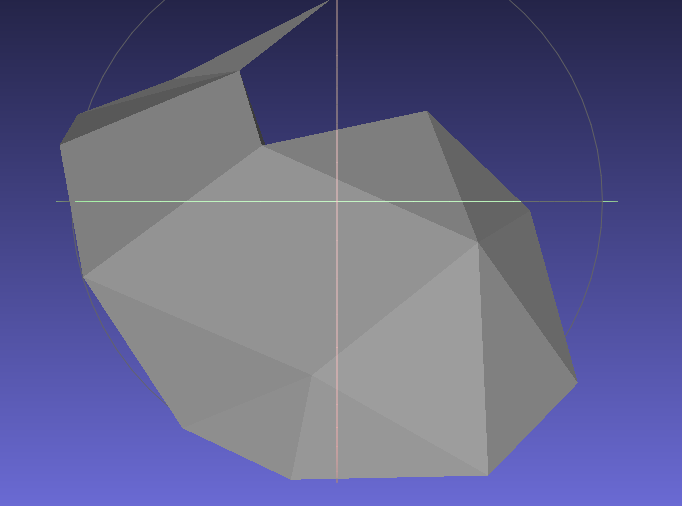
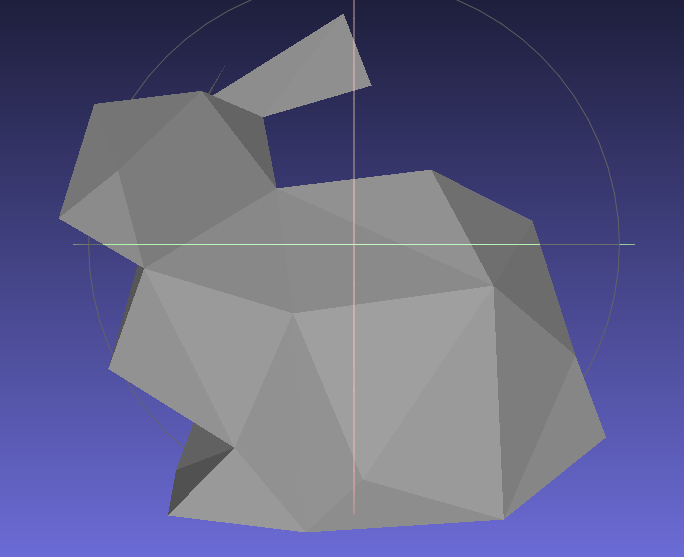
原模型和检查的0.03简化比的dragon.obj



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。