Chapter 1

空间划分法优化 YinSet 布尔运算中的三角形求交

已有算法使用的两个 YinSet 中所有的三角形两两求交, 该文档在不改变三角形求交算法接口的前提下, 设计使用空间划分法减少计算三角形求交的次数来提高 YinSet 布尔运算的运行效率.

以下实现类和函数都在名字空间 YSB 中.

1.1 算法接口: 契约

调用函数

void TriangleIntersection::operator()(const std::vector<Triangle>& inputA, const std::vector<Triangle>& inputB, Real tol)

输入

inputA, inputB 分别是两个 YinSet 边界上的所有三角形.

输入条件

一个 YinSet 内的三角形不交或交于边和顶点.

输出

TriangleIntersection 的成员变量 vector<pair<vector , vector<pair<int, int>>>> resultA, resultB;

inputA 中第 i 个三角形 inputA[i] 与其他所有三角形的交线段存放在 resultA[i].first, 所有重合三角形存放在 resultA[i].second. resultA[i].first 中的线段保存了是哪两个三角形相交得到. resultB 类似.

1.2 代码实现

1.2.1 Cuboid

模板: template<int T>

T 表示空间坐标使用的数据类型.

功能

三维空间中的长方体, 用于将三角形区分为与长方体有公共部分和没有公共部分两类.

数据成员

1. Point<T, 3> vertex[8]: 长方体的 8 个顶点.
2. Rectangle<T, 3> face[6]: 长方体的 6 张面.

函数成员

1. int contain(Point<T, 3> p, Real tol = TOL) const:
   输入: 一个点 Point p.
   输出: 长方体是否包含 p.
2. vector<Cuboid<T>> divide() const:
   输出: 长方体 8 等分得到的 8 个长方体.
3. void intersect(const Triangle<T, 3>& tri, vector<int>& res, Real tol = TOL) const:
   输入: 一个点 Triangle tri.
   输出: 8 等分得到的长方体分别与 tri 是否相交.

1.2.2 OctreeNode

功能

树节点, 存储节点包含的三角形, 空间划分树确定后计算所有叶节点中的三角形求交.

数据成员

1. Cuboid val: 节点对应的长方体.
2. vector<OctreeNode*>: 8 等分的子长方体对应的节点.
3. vector<int> tris[2]: 包含的 YinSet 中的三角形.

1.2.3 OctreeTriangleIntersection

模板: template<int T>

T 表示空间坐标使用的数据类型.

功能

计算布尔运算时的三角形求交.

数据成员

1. public TriangleIntersection<T>: 继承三角形求交的接口.
2. OctreeNode<Cuboid<T»* root: 计算区域的空间划分树的根节点.
3. int deep: 空间划分树的最大深度, 默认为 log(n).

函数成员

1. void initOctree(const vector<Triangle<T, 3>>& inputA, const vector<Triangle<T, 3>>& inputB, int depth, Real tol):
   输入: 计算求交的所有三角形和树的最大深度.
   输出: 空间划分树的根节点 root, 树内每个节点存储了与节点对应的长方体有公共部分的三角形.
2. virtual long testNum(OctreeNode<Cuboid<T>>* r) const:
   输入: 空间划分树的一个节点 r.
   输出: 节点 r 中需要计算的三角形求交次数.
3. void dfsConstructTree(OctreeNode<Cuboid<T>>* r, int depth, const std::vector<Triangle<T, 3>>& inputA, const std::vector<Triangle<T, 3>>& inputB)
   输入: 空间划分树的一个叶节点, 剩余深度.
   输出: 若没有达到最大深度并且 testNum() 不返回 0, 构造当前节点对应长方体的 8 等分作为叶节点, 并根据相交关系将当前节点包含的三角形分配到叶节点中. 最后对叶节点递归调用函数.
4. int pruneTree(OctreeNode<Cuboid<T»* r):
   输入: 空间划分树.
   输出: 以减少计算三角形求交数量为目标对树进行剪枝.
5. void calTest(OctreeNode<Cuboid<T>>* r, const vector<Triangle<T, 3>>& inputA, const vector<Triangle<T, 3>>& inputB, Real tol):
   输入: 空间划分树, 求交的三角形.
   输出: 对每个叶节点中的三角形进行求交.
6. void edgeCal(const vector<Triangle<T, 3>>& input, Real tol, int id):
   输入: 一个 YinSet 的三角形.
   输出: 三角形之间在边上的相交关系.
7. void intersect(int iA, int iB, const vector<Triangle<T, 3>>& inputA, const vector<Triangle<T, 3>>& inputB, Real tol):
   输入: 求交的两个三角形.
   输出: 在三角形对应的 resultA, resultB 中新增求交结果.