施工实习答辩

实习单位：中国建筑科学研究院

指导教师：李寅斌

李钦　2020012872　未央-水木 02

2023 年 9 月 21 日

Contents

1. 容差合并
   - 朴素算法
   - 近邻搜索
   - k-d Tree
2. 模型清洗

Contents

1. 容差合并
   - 朴素算法
   - 近邻搜索
   - k-d Tree
2. 模型清洗

容差合并

距离小于容差 t 的点视为同一点，将其合并为一个点.

输入
- 待合并点集 $\{P_1, \ldots, P_n\}$
- 容差 t

输出
- 合并后的点集 $\{Q_1, \ldots, Q_m\}$

朴素算法

Listing 1. remove_duplicate_points_naive.py

```
19 def point_less(u, v):
20     u_coordinates = rs.PointCoordinates(u)
21     v_coordinates = rs.PointCoordinates(v)
22     return u_coordinates < v_coordinates
23
24
25 removed = set()
26 for u, v in itertools.combinations(P, 2):
27     if u in removed or v in removed:
28         continue
29     if not rs.PointCompare(u, v, tolerance=t):
30         continue
31     removed.add(v if point_less(u, v) else u)
32 Q = [u for u in P if u not in removed]
```

复杂度 $\Theta(n^2)$

近邻搜索

Listing 2. RemoveDuplicatePoints.cs

```
protected override void SolveInstance(IGH_DataAccess DA)
{
    List<Point3d> points = new List<Point3d>();
    double tolerance = 0.01;
    DA.GetDataList("Points", points);
    DA.GetData("Tolerance", ref tolerance);
    points.Sort((Point3d a, Point3d b) => a.CompareTo(b));
    HashSet<Point3d> point_set = new HashSet<Point3d>(points);
    Node3d<Point3d> tree = new Node3d<Point3d>(
        converter: (Point3d pt, out double x, out double y, out double z) =>
        {
            x = y = z = 0.0;
            TreeDelegates.Point3dCoordinates(pt, ref x, ref y, ref z);
        },
        region: new BoundingBox(points)
    );
    tree.AddRange(points);
    for (int i = 0; i < points.Count; i++)
    {
        if (!point_set.Contains(points[i]))
            continue;
        while (true)
        {
            List<Index3d<Point3d>> neighbors = tree.NearestItems(
                locus: points[i],
                groupSize: 2,
                minimumDistance: 0.0,
                maximumDistance: tolerance
            );
            if (neighbors.Count < 2)
                break;
            Index3d<Point3d> neighbor = neighbors[1];
            if (points[i].DistanceTo(neighbor.Item) > tolerance)
                break;
            point_set.Remove(neighbor.Item);
            tree.Remove(neighbor);
        }
    }
    DA.SetDataList("UniquePoints", point_set);
}
```

近邻搜索

1. 创建一个空的 `HashSet`，用于存储最终的结果.
2. 将点集中的所有点添加到 `HashSet` 中.
3. 遍历点集中的每个点，如果该点已经被从 `HashSet` 中移除，则跳过该点；否则，执行以下操作:
   1. 使用 `Node3d` 的 `NearestItems` 方法，查找距离该点在容差范围内的最近的两个点 (包括该点本身).
   2. 如果只找到一个点 (即该点本身)，则说明该点没有重复点，继续下一个点的遍历.
   3. 如果找到两个或更多的点，则说明该点有重复点，取第二个最近的点作为重复点，并从 `HashSet` 和空间划分树中移除该重复点. 然后重复这一步骤，直到找不到更多的重复点为止.
4. 将 `HashSet` 中剩余的点作为输出返回.

k-d Tree

Contents

1. 容差合并
   - 朴素算法
   - 近邻搜索
   - k-d Tree
2. 模型清洗