数据结构大作业答辩

赵文亮

自动化系

zhaowl16@mails.tsinghua.edu.cn

2018 年 1 月 25 日

内容

大作业二：多视图三维重建

实现步骤

1. 表面点获取
2. 表面点法向获取
3. 颜色信息获取
4. 泊松重建

表面点获取优化

根据代码运行时间可知，原始的 getModel 算法十分耗时（本机运行 6.6s）。该算法的缺点在于：

- getModel 和 getSurface 函数重复，进行了两次三层循环
- 空间中很多点其实是没有必要判断的，全部判断浪费时间

优化算法

1. 取消 getModel 步骤，直接一步 getSurface 获取表面点。
2. 首先通过循环找到一个表面点（在查找的过程中顺便标记 voxel）
3. 从这一点开始做 BFS，将途中的内部点入队

优化后时间：1.8s

表面点获取优化

进一步优化

- 在 BFS 的过程中仍然有许多内部点入队，而我们希望的是尽可能是表面点参与 BFS
- 定义函数 totalInside 判断某一点是否全在 visual hull 内部（指该点以及其八个角位于 visual hull 内部）
- totalInside 的点在 BFS 中不入队

优化后时间：1.2s

表面点获取优化

进一步优化

- 寻找第一个表面点的过程仍需遍历
- 借用解决散列冲突中“双向平方试探”思路快速找到第一个表面点：
  1. 从中心开始
  2. 每次使用双向平方试探的规则向两侧试探

由于能够快速找到第一个表面点，运行时间进一步被优化：0.54s

法向信息获取优化

- 原代码中使用 innerList 保存表面点周围的内部点
- 事实上只需使用一个变量对内部点求和，最终取平均即可

优化后运行时间减少 0.5s

颜色信息优化

优化思路

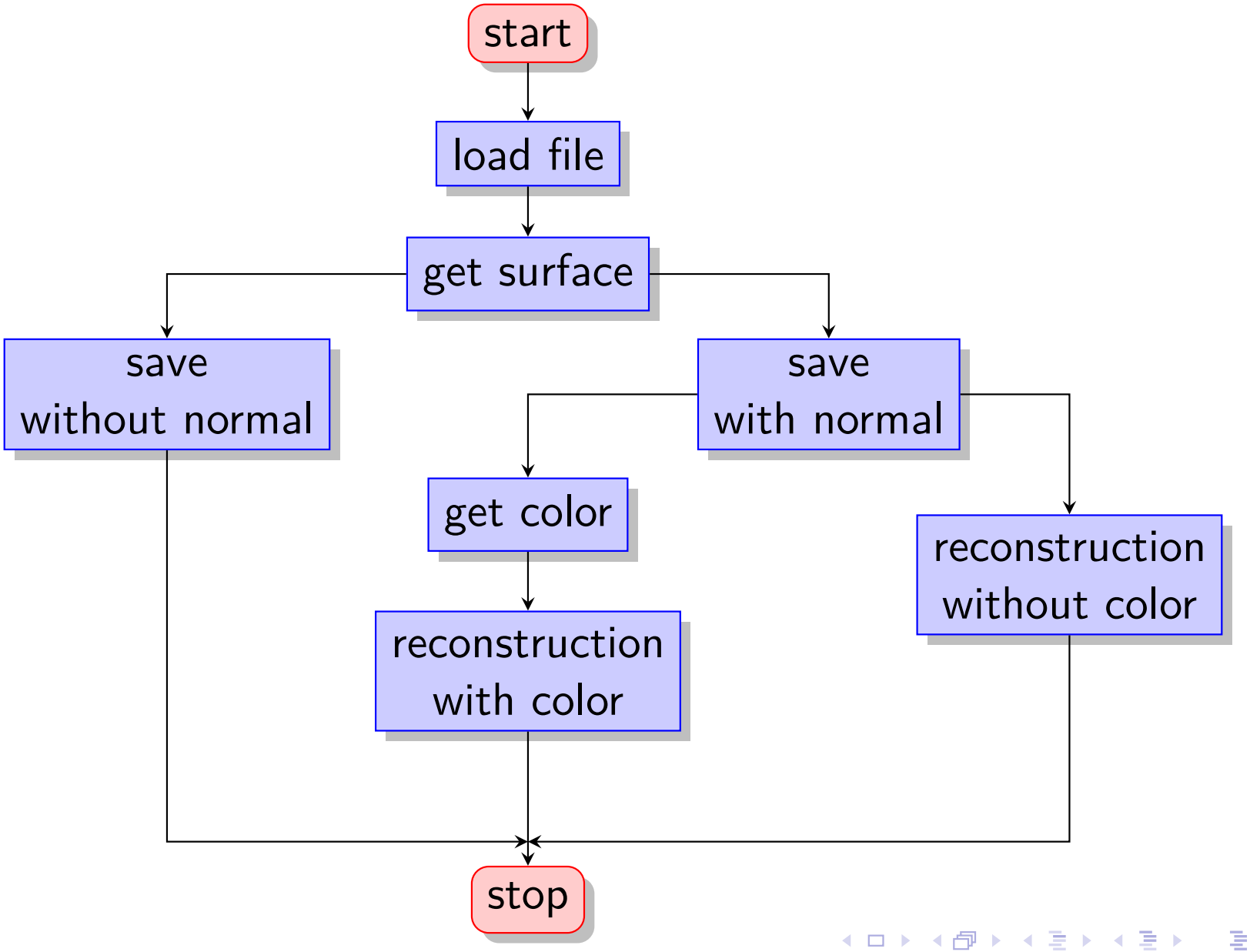
- 读取图片时同时读取原图
- 通过 getColor 函数，读取所有表面点的颜色信息
- 调用 savePly 将表面点的颜色信息和法向信息保存在 ply 文件
- 使用泊松重建得到带颜色的模型

颜色信息优化

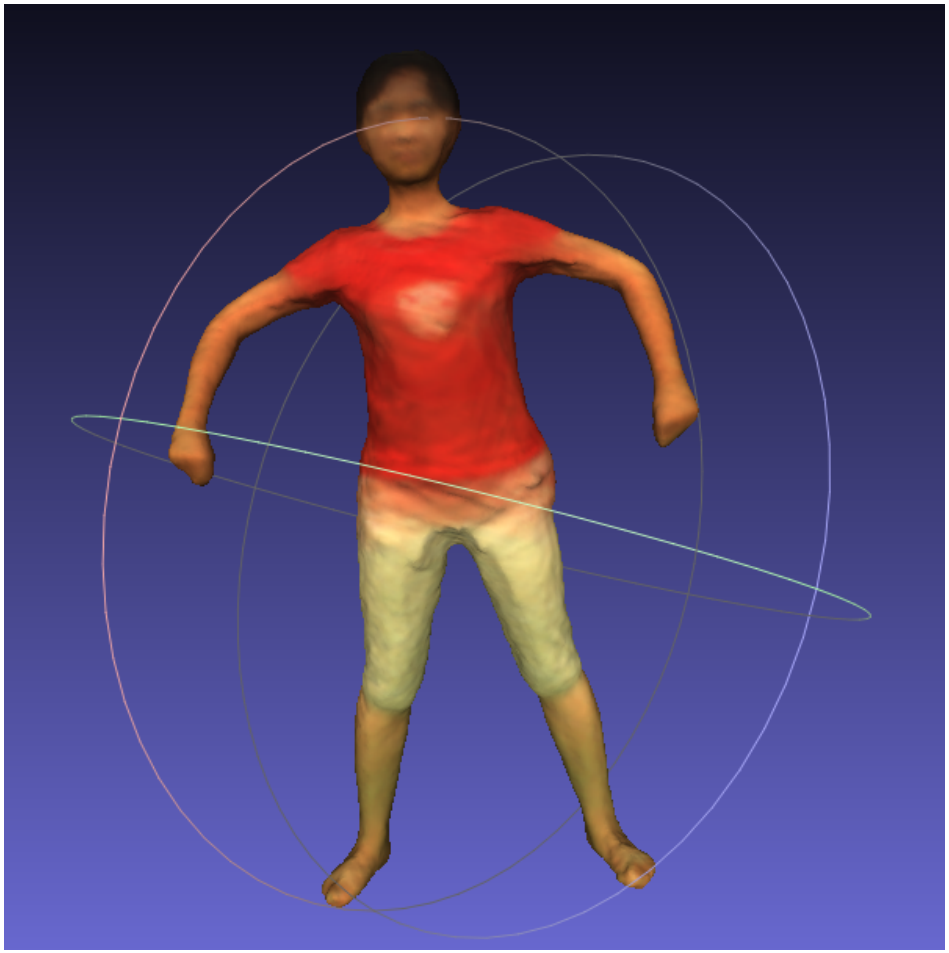
getColor 函数的思路如下

```
void Model::getColor(const Point & p){
    cv::Vec3f color = cv::Vec3f(0, 0, 0);
    for (int i = 0; i < prejectionCount; i++){
    //对每一个投影
        if (在投影范围内){
            color += color_from_this_prejection;
            count++;
        }
    }
    color /= count;//计算颜色平均值
    m_colorMap[p] = color;//保存在hash表
}
```

多线程优化


start
load file
get surface
save
without normal
save
with normal
get color
reconstruction
with color
reconstruction
without color
stop

重建结果



局限

- 有些部分颜色有些暗淡
- 面部能看见五官轮廓，但不是十分清晰
- 表面纹理仍有优化的空间

参考文献

[1] Peng Song, Xiaojun Wu, and Michael Yu Wang. Volumetric stereo and silhouette fusion for image-based modeling. *Visual Computer*, 26(12):1435–1450, 2010.