语义引导的稠密重建

一.环境配置推荐

- Ubuntu 18.04
- CGAL 5.0
- OPENCV 3.4.0
- eigen 3.3.4
- boost 1.65.1
- colmap 3.6
- c++ 11
- cmake 3.10.2
- 程序内中文注释为GB2312格式
- 本程序为CPU版本，未支持GPU

二.程序配置参数

数据文件夹内结构：

```
+-- data_directory
| +-- configSL.txt  程序运行参数配置文件
| +-- bundler
|   +-- cameras.txt //**输入**:相机内外参数
|   +-- SfM_Final.out //**输入**:SfM获得的结果
| +-- images //**输入**：原始RGB图像
| +-- labels //**输入**：原始语义图
| +-- undis  //畸变矫正后的RGB图像
| +-- undis_labels //畸变矫正后的语义图
| +-- sparse.ply //运行过程保存的稀疏点云
| +-- sparse_bound.ply //手动设定边界后的稀疏点云（不设定边界的话可用sparse.ply拷贝重命名）
| +-- interfiles //运行过程中间保存的一些文件
|   +-- camera_pose_neighbors.txt //相机参数位姿+邻域图像选择结果
| +-- SenseCar.db  //运行过程中产生的数据库文件
| +-- depthmap  //运行过程中产生的深度图
```

```
| +-- depthmap_filtered //运行过程中过滤后的深度图
| +-- dense_output //**输出**：稠密点云
```

configSL.txt文件内各项参数说明

```
data_path=/home/xxx/Data/Test // 数据路径，需设定

log_name=log  //日志文件名，需设定。不设定则不输出日志文件

db_name=Test.db // 数据库文件，需设定

image_folder=images // 图像文件夹名，默认值为images

sparse_bound_name=sparse_bound.ply  //稀疏点云边界文件ply，默认值为sparse_bound.ply

bCreatDB=0  // 1：name  创建数据库并写入图像文件名
            // 2：name+GPS  创建数据库并写入图像文件名，同时写入图像GPS信息

bReadFromFiles=0  // 1：bundler 从out读取相机SfM信息
                  // 2：colmap 从colmap文件夹读取相机SfM信息

bSparseBound=0  // 1：从sSparseBoundName.ply裁剪(convex hull)
                // 2：从sSparseBoundName.ply裁剪(bounding box)
                // 3：根据相机sfm自动裁剪(convex hull)
                // 4：根据相机sfm自动裁剪(bounding box)
                // 5：根据相机GPS自动裁剪(convex hull)
                // 6：根据相机GPS自动裁剪(bounding box)
                // 7：从sSaprsePloygonBoundName.txt裁剪

bUndistortImage=0 // 1. 图像畸变矫正

bUndistortLabel=0 // 1. 语义分割图畸变矫正

bNeighborSelection=0  // 1. 选择邻域图像组

bDepthComputation=0  // 1. 深度图计算

bDepthFilter=0  // 1. 深度图过滤

bDepthFusion=0  // 1. 深度图融合

bDenseBound=0   // 1：从sDenseBoundName.ply裁剪
                // 2：从sDensePloygonBoundName.txt裁剪

bColorEstimation=0   // 1. 点云着色

bPointsLabeling=0    // 1：点云语义标注，使用点云原始颜色
                     // 2：点云语义标注，使用label_color_table.txt设置颜色

threads=8       // 多线程并行运行参数，根据本机cpu核心数量进行设置
```

```
scale=0.6        // 图片分辨率裁剪压缩系数，1=不裁剪，0.6=按照原来长宽的0.6倍进行裁剪

ncc=0.6          // NCC低分过滤门槛，小于0.6以下的像素点被认为是不可靠的匹配，要被滤除

half_window_size=5     // NCC半窗口大小

window_size_step=2      // NCC窗口内遍历步长

depth_diff_threshold=0.01 // 重投影误差门槛，误差大于0.01的要被滤除

num_max_views=6    // 邻域图像数量上限

num_min_views=2    // 邻域图像数量下限

angle_threshold=5      // 邻域图像选择时两图像夹角最小值

num_consis_filter=2    // 深度图过滤重投影误差需要最小的图片支持数量

num_consis_fuse=1      // 深度图融合时重投影误差需要最少的图片支持数量

lidar_depthmap=0       // 是否读入激光雷达数据，1读入雷达 ， 0无雷达

max_points=300000000  // 生成稠密点云最大点数

color_points_sample=200000000 // 稠密点云上色最大点数，影响生成点云大小（可设为0~1之间的系数）

color_scale=0.5        // 点云上色点数比例
```

三.程序运行说明

语义引导的稠密重建程序主体部分以V1.6.2为基础，更新了以下两个文件：

- 算法程序：/Algo/Algorithms.cpp
- 基本数据结构和函数声明：/base.h

运行可在以下两种方式中任选一种：

1.命令行模式

```
//首先切换到程序根目录
cd build
cmake ..
make -j8
./3DMapSL /数据集目录地址/configSL.txt
```

2.Visual Studio Code平台

- 确认已安装c/c++插件
- 在.vscode文件夹launch.json文件修改 "cwd"="数据集目录地址"
- 点击F5即可运行

输出的稠密点云.ply文件在/dense_output/文件夹下，结果以运行时间命名。

四.总结及展望

对语义导向的稠密重建各个子模块实际效果进行分析：

- 邻域视角选择：对特定场景有效，主要是岔路口。
- 深度图初始化：全随机初始化最差，网格初始化和语义初始化效果差不多。
- 深度图传播：自适应窗口对于弱纹理区域有明显提升效果，NCC公式中的语义因子效果一般。
- 深度图补全：因为语义分割通常能准确地划分出路面区域，所以路面补全效果明显。

本方法依赖于语义分割的质量。对于室外道路场景，SOTA的方法对路面分割准确率一般在98%-99%以上，所以对于路面的提升效果也是最好的。 其他区域语义分割准确率可能是80%-90%之间，少量的外点会影响细节质量，所以NCC评分公式中添加语义因子有可能带来负面效果。 对于室内场景语义分割质量通常较差，本方法目前提升空间有限。鉴于目前语义分割精度的影响，相对来说语义值更适宜作为区域性的软约束信息，不适宜直接参与单点像素的计算。

以后的发展方向可以考虑将语义约束引入端到端MVS神经网络或者深度图优化中去。近三年以MVSNet为代表性的稠密重建方法展现出了强劲的发展势头，2018年~2020年之间有超过15篇MVSNet方向的顶会paper，与之对应传统方法的MVS不足5篇。神经网络方法计算深度图速度通常能达到传统方法的10-100倍以上，部分场景下有超越传统方法的准确度和完整度。但是截至2021年2月却没有基于语义的MVSNet出现，因此这里是一个值得关注的创新点。

如有任何问题请联系作者：吕明哲 lmzo@vip.163.com