

数据：

1. 整理出来的采集的点云数据 ./data/rawdata

Dataset	Frame Number
Bedroom	101
Childroom	60
Kitchen	59
Office_1	42
Office_2	63

2. 采集工具：小推车、电脑、Kinect v1，如右图所示。



3. 采集算法以及配置

- a) Voxel hashing [1]
- b) 具体电脑配置

品名	服务器配置	数量
主板	Asus Z97-K	1
cpu	INTEL I7 4790K 原盒	1
散热器	超频三铁塔豪华版	1
内存	Kingston 8g ddr3 1866 骇客神条	2
硬盘 1	三星 250gb 850 evo	1
硬盘 2	Seagate 2tb dm003	1
显卡	Asus GTX1080Ti-11G	1
机箱	拓普龙工控 4U 中长，并合适扩展风扇	1
电源	全汉金甲 900 800W 额定	1
键鼠	罗技 mk200 套件	1

代码：

1. 算法代码

- a) 分层分割平面 ./code/CPlanefitting_seg_v2.4
输入：点云数据 *.ply
输出：平面分割过后的点云数据 *.idtree, *.label.arma。
*.idtree 数据格式如下，
N

```
[
Node_Index, Plane_child_Num, Object_child_Num, Other_index
[Plane_child_index]
[Object_child_index]
]
```

N 为树结点个数，之后一共有 $3*N$ 行。每 3 行内包含的是一个树结点的分层分割信息。

Node_Index 为该结点编号，Plane_child_Num 为该结点的平面儿子结点个数，Object_child_Num 为该结点的物体儿子结点个数，Other_index 为剩余其他点云集合的结点的编号。

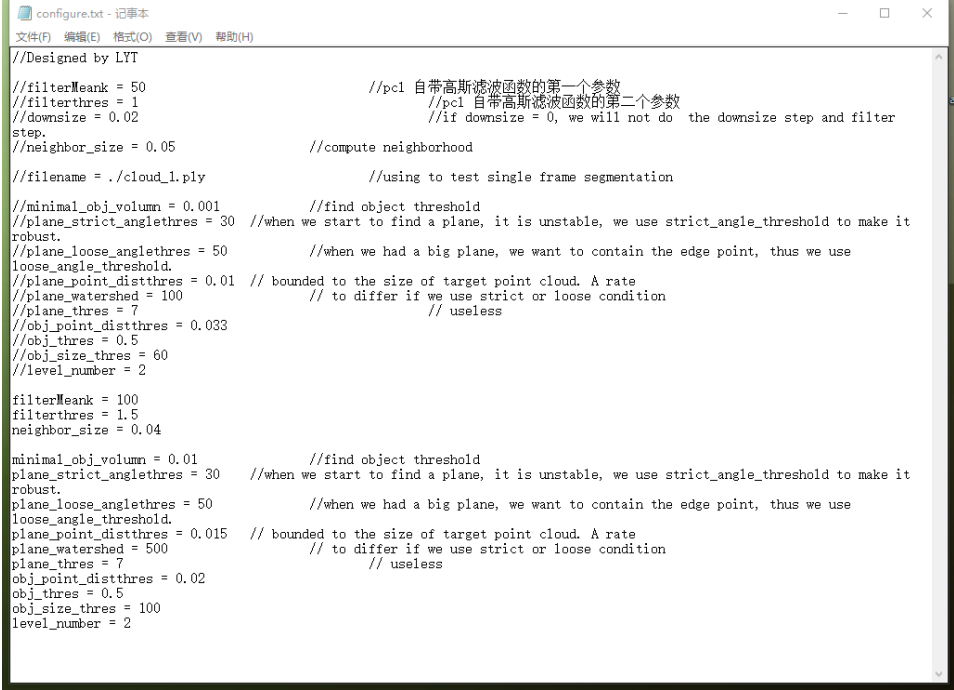
举例说明如下：

```
1 40
2 0 3 7 11
3 1 2 3
4 4 5 6 7 8 9 10
5 1 0 0 0
6
7
8 2 0 0 0
9
10
11 3 0 0 0
```

第 1 行 40 代表一共有 40 个树结点，每个树结点占三行。从第 2 行到第 4 行分别代表 0 号结点有 3 个平面儿子结点，编号为 1,2,3；有 7 个物体儿子结点，编号为 4-10，剩余其他的点云集合的结点编号为 11。

配置文件：configure.txt

里面包括了参数如下：



```
configure.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
//Designed by LYT
//filterMeank = 50 //pcl 自带高斯滤波函数的第一个参数
//filterthres = 1 //pcl 自带高斯滤波函数的第二个参数
//downsize = 0.02 //if downsize = 0, we will not do the downsize step and filter
step.
//neighbor_size = 0.05 //compute neighborhood
//filename = ./cloud_1.ply //using to test single frame segmentation
//minimal_obj_volumn = 0.001 //find object threshold
//plane_strict_anglethres = 30 //when we start to find a plane, it is unstable, we use strict_angle_threshold to make it
robust.
//plane_loose_anglethres = 50 //when we had a big plane, we want to contain the edge point, thus we use
loose_angle_threshold.
//plane_point_distthres = 0.01 // bounded to the size of target point cloud. A rate
//plane_watershed = 100 // to differ if we use strict or loose condition
//plane_thres = 7 // useless
//obj_point_distthres = 0.033
//obj_thres = 0.5
//obj_size_thres = 60
//level_number = 2
filterMeank = 100
filterthres = 1.5
neighbor_size = 0.04
minimal_obj_volumn = 0.01 //find object threshold
plane_strict_anglethres = 30 //when we start to find a plane, it is unstable, we use strict_angle_threshold to make it
robust.
plane_loose_anglethres = 50 //when we had a big plane, we want to contain the edge point, thus we use
loose_angle_threshold.
plane_point_distthres = 0.015 // bounded to the size of target point cloud. A rate
plane_watershed = 500 // to differ if we use strict or loose condition
plane_thres = 7 // useless
obj_point_distthres = 0.02
obj_thres = 0.5
obj_size_thres = 100
level_number = 2
```

b) 协同分割算法 ./code/CoAna

输入：点云数据*.ply，分层分割结果*.idtree，*.label.arma

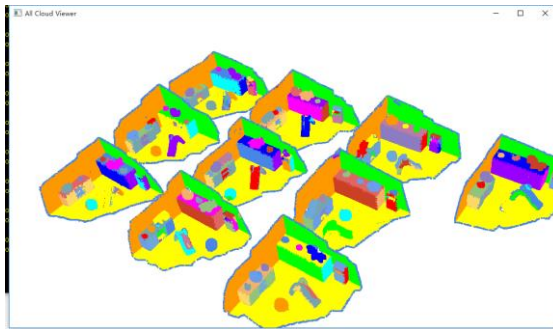
输出：新的分层分割结果*.idtree, *.label.arma。可以用显示代码 2.a)或者 2.b)看结果。

配置参数如下：

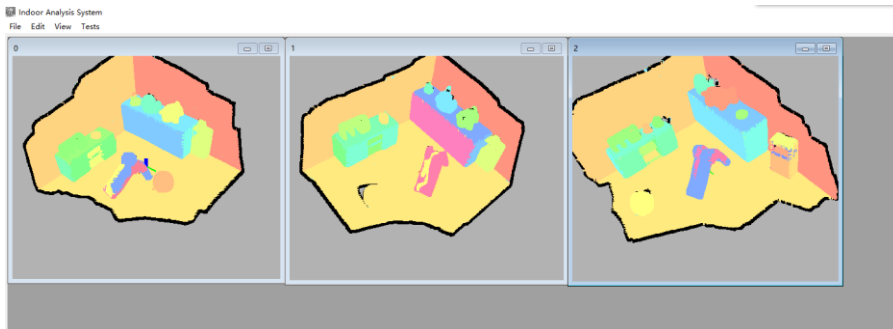
```
SampleNum = 1024*1024;  
BinNum = 1024;  
VecNum = 64;  
ColBinNum = 4;  
clu_thres = 0.85;  
iteration_thres = 5;  
d_thres = 0.15;
```

2. 显示代码

a) 同时显示多个场景./code/DemoProgram



b) 显示结果工具 ./tool/align_annotation



具体操作在下文“工具”中会说明。

3. 别人代码

a) Nießner (Voxel hashing)[1]算法 ./code/ VoxelHashing-master

该算法用于采集点云数据。

配置文件详细见./zParametersDefault.txt

输入：RGBD stream

输出：融合后的点云数据 point cloud, single frame, *.ply

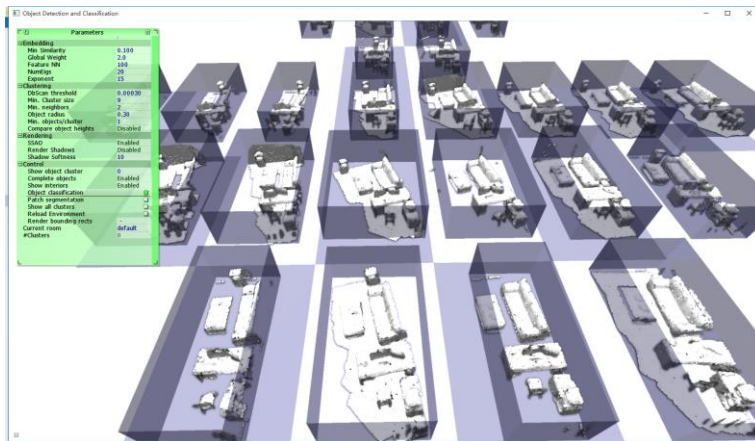
b) Mattausch et. [2] 算法 ./code/objectdetection

该算法用来跑协同分析结果的对比，至少需要电脑配置为 CPU i7, RAM 8G, GTX 960.

算法配置文件详细见./environments/point.env

输入：multiple point clouds

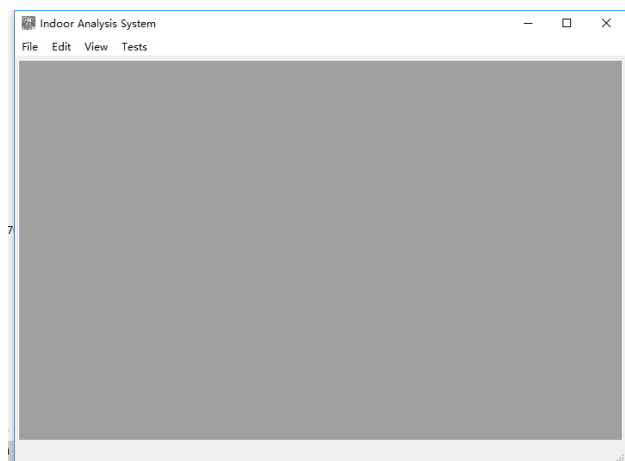
输出：结果存储在*.label.arma



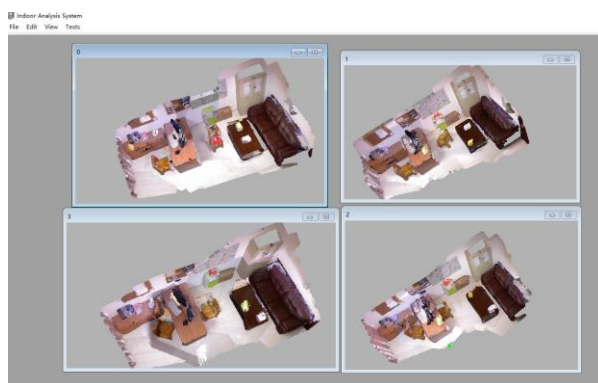
附：该 VS 程序需要调用 Matlab

工具：

1. 预处理（将地面法向转换向上的）工具 `./tool/align_annotation`
打开运行文件：`./Dev_Runtime/runExpIA.bat`

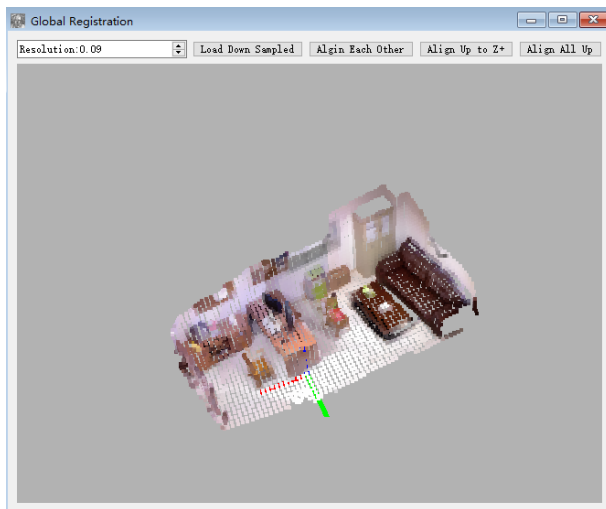


读入数据：File/Open Inputs



操作 1：将所有帧点云粗糙手动调整地面法向向上
Step1: Edit/Global Align

Step2: Load Down Sample



Step3: Annotation

操作 2: 匹配所有帧点云 align Each Other

操作 3: 将所有帧点云调整为地面法向向上, 并且地面的高度为 0。 Align All Up

存储数据: File/Save Aligned

2. 制作 ground truth 工具 ./tool/align_annotation

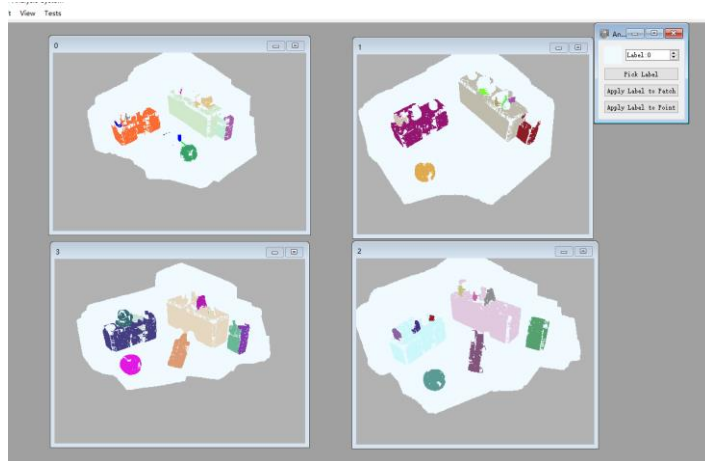
打开软件和读入数据同上操作。

操作 1: 输入分割文件 File/Load Labels

操作 2: 查看结果 View/Custom View



操作 3: 修改分割结果 Edit/Annotator



使用 Annotator 来手动修改 label 结果。

输出结果：File/Save Labels

引用：

[1] Nießner M, Zollhöfer M, Izadi S, et al. Real-time 3D reconstruction at scale using voxel hashing[J]. ACM Transactions on Graphics (TOG), 2013, 32(6): 169.

[2] Mattausch O, Panozzo D, Mura C, et al. Object detection and classification from large-scale cluttered indoor scans[C]//Computer Graphics Forum. 2014, 33(2): 11-21

附录：

*.label.arma 数据格式：N 个点的点云数据对应了 N 个数字。每个数字按次序代表该点属于哪一个区域，举例如下：

点云数据 example.ply 有 3 个点(0,0,0), (0,0,1), (0,1,0)，那么在 example.label.arma 中对应的为

1

1

2

表示第一个点(0,0,0)和第二个点(0,0,1)属于第 1 类

第三个点(0,1,0)属于第 2 类。