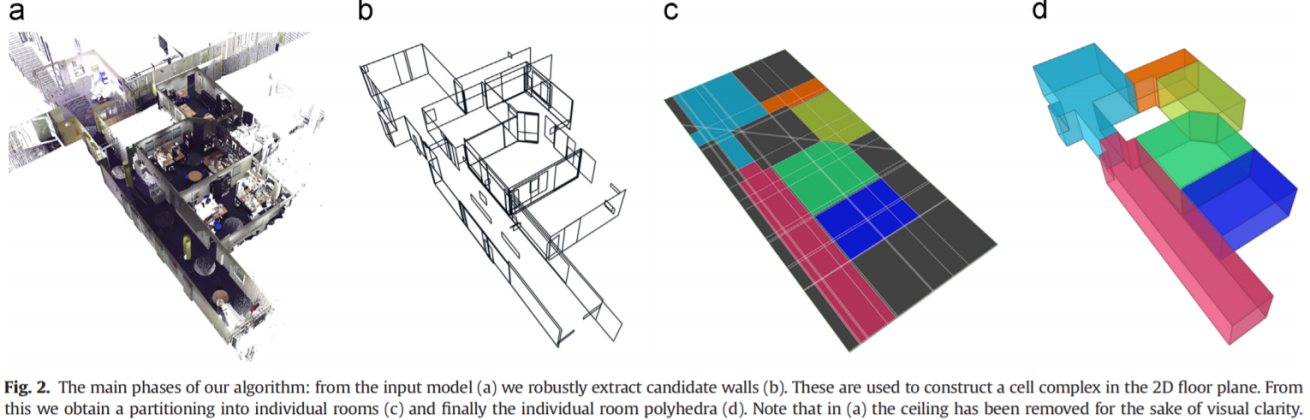
# Indoor Reconstruction Review

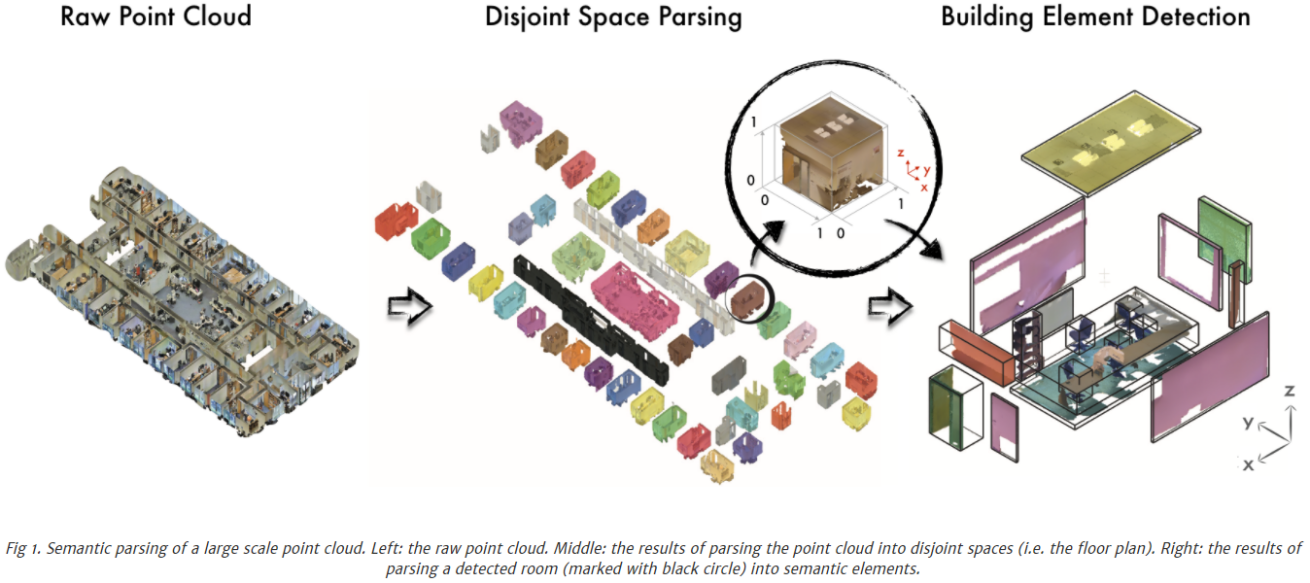
## Indoor Floor Map

### Reconstruction from point cloud



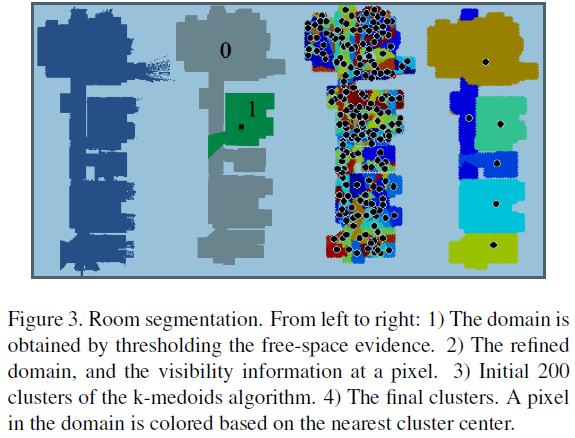
Mura, C., et al. (2014). "Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts." Computers & Graphics **44**: 20-32.

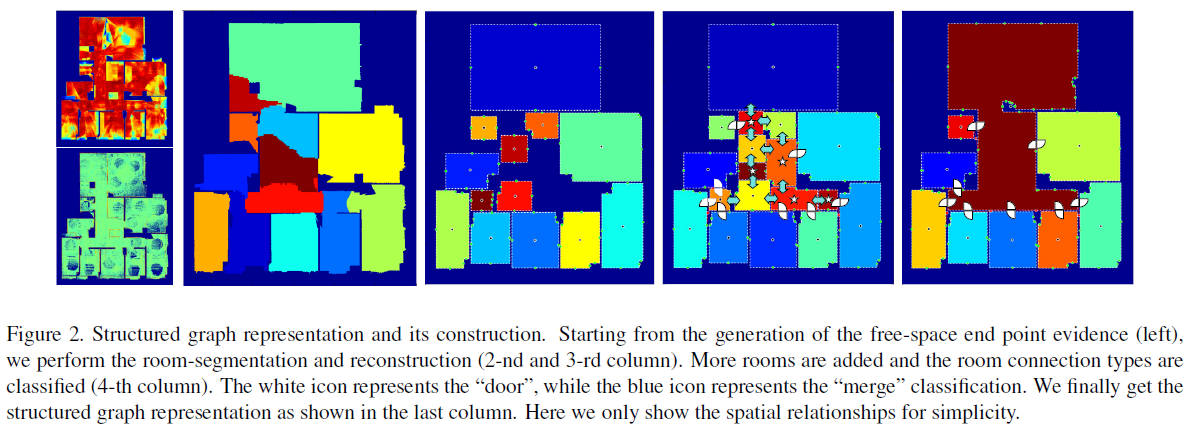
Ochmann, S., et al. (2016). "Automatic reconstruction of parametric building models from indoor point clouds." Computers & Graphics **54**(Supplement C): 94-103.



Armeni, I., et al. (2016). 3d semantic parsing of large-scale indoor spaces. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

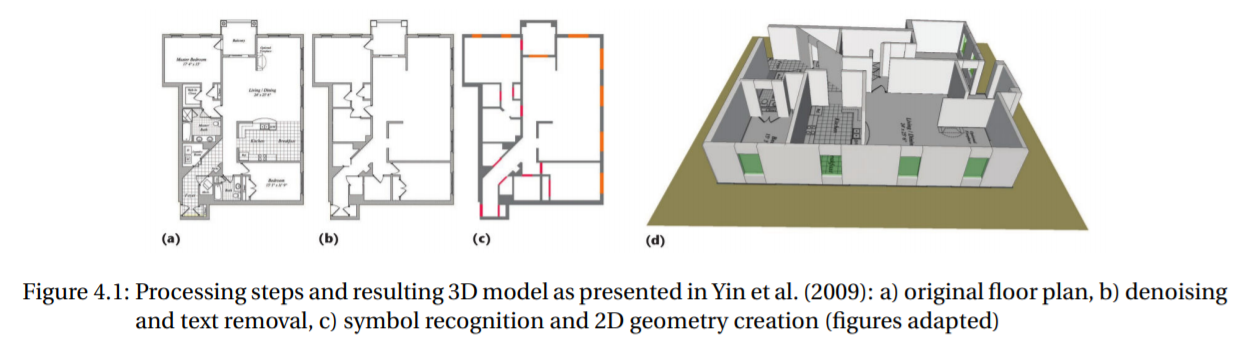
<http://buildingparser.stanford.edu/method.html>





Ikehata, S., et al. (2015). Structured indoor modeling. Proceedings of the IEEE International Conference on Computer Vision.

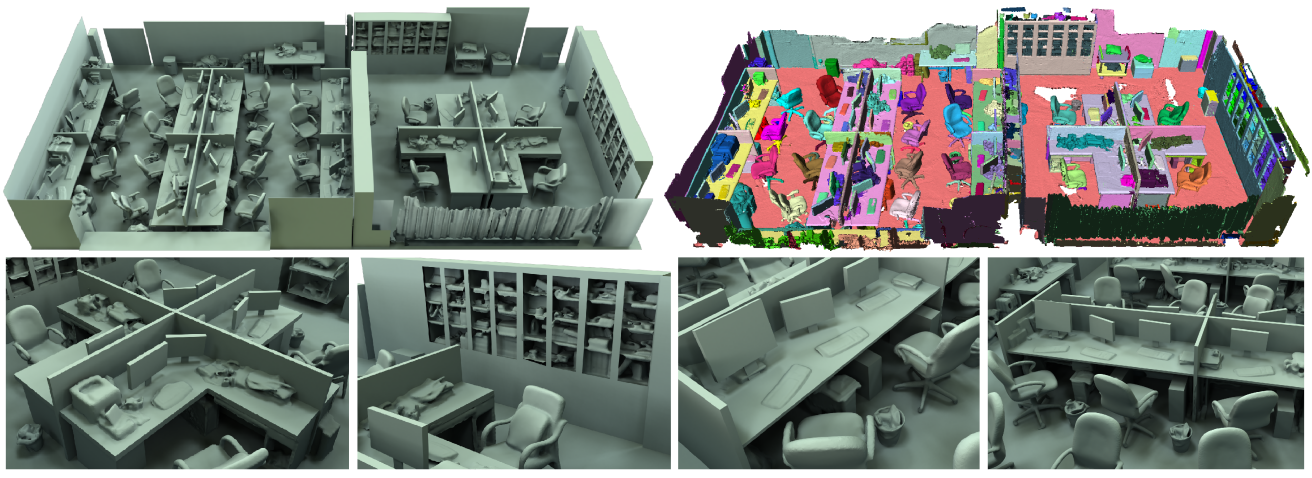
### Reconstruction by reverse-engineering of existing floor maps



Peter, M. (2016). Crowd-sourced reconstruction of building interiors, München: Verlag der Bayerischen Akademie der Wissenschaften.

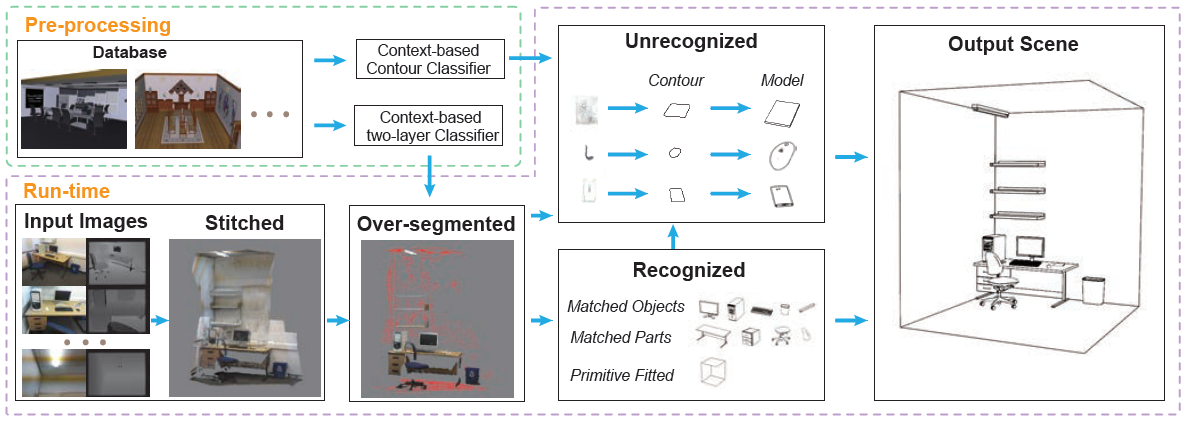
## Indoor 3D Map

### Planes + meshes

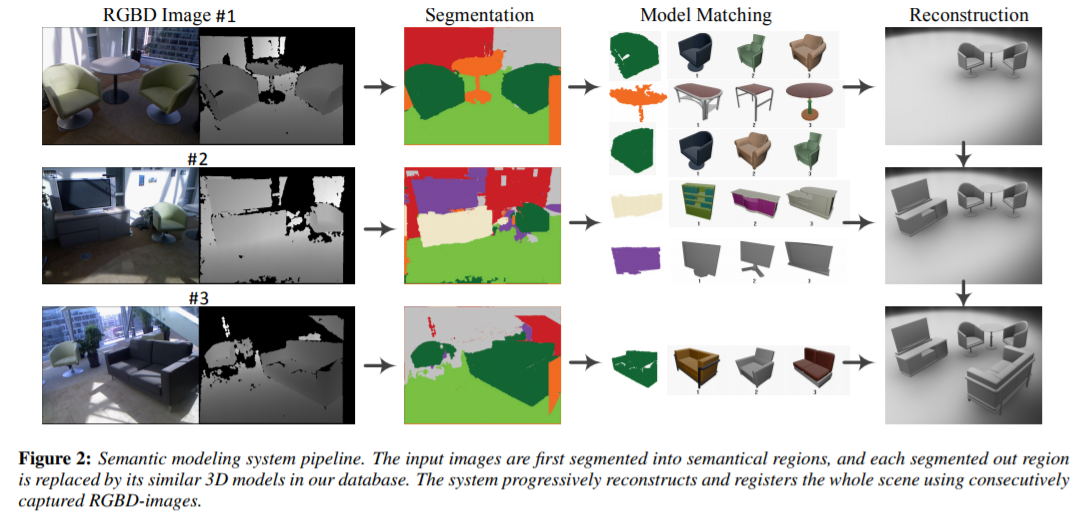


Zhang, Y., et al. (2015). "Online structure analysis for real-time indoor scene reconstruction." ACM Transactions on Graphics (TOG) **34**(5): 159.

### semantic model

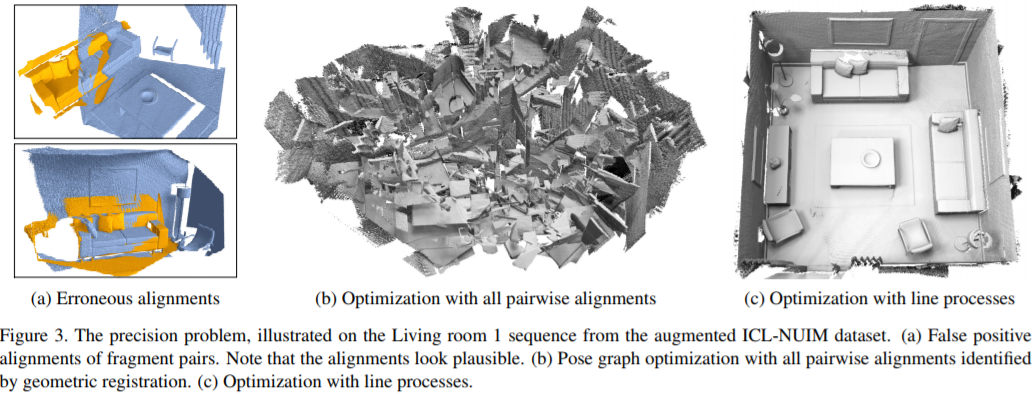


Chen, K., et al. (2014). "Automatic semantic modeling of indoor scenes from low-quality RGB-D data using contextual information." ACM Transactions on Graphics **33**(6).



Shao, T., et al. (2012). "An interactive approach to semantic modeling of indoor scenes with an rgbd camera." ACM Transactions on Graphics (TOG) **31**(6): 136.

## Crowdsensing Mapping



Choi, S., et al. (2015). Robust reconstruction of indoor scenes. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

<https://github.com/qianyizh/ElasticReconstruction>

## Model Standard

### IndoorGML (Static Model)

IndoorGML is an OGC standard for an open data model and XML schema for indoor spatial information. It aims to provide a common framework of representation and exchange of indoor spatial information. It is defined as an application schema of [OGC Geographic Markup Language (GML) 3.2.1](http://portal.opengeospatial.org/files/?artifact_id=20509).

<http://indoorgml.net/>

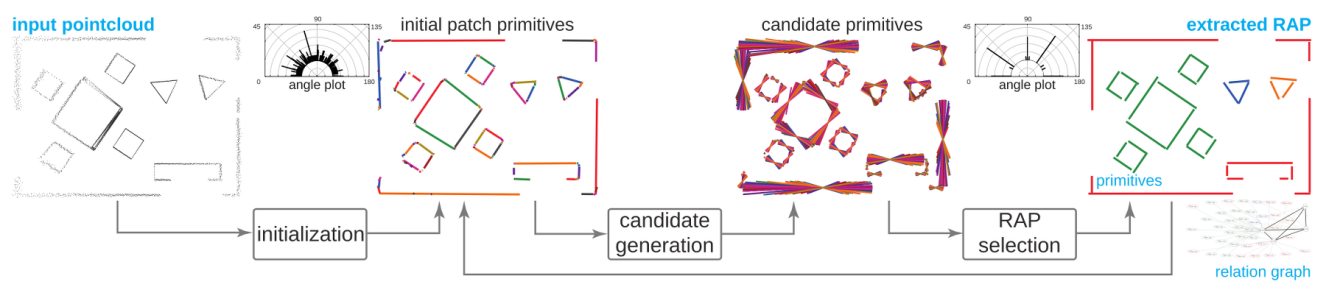
优势在可视化

### BIM (Process Model)

Building information modeling (BIM) is a process involving the generation and management of digital representations of physical and functional characteristics of places. Building information models(BIMs) are files (often but not always in proprietary formats and containing proprietary data) which can be extracted, exchanged or networked to support decision-making regarding a building or other built asset.

优势在语义、数据管理

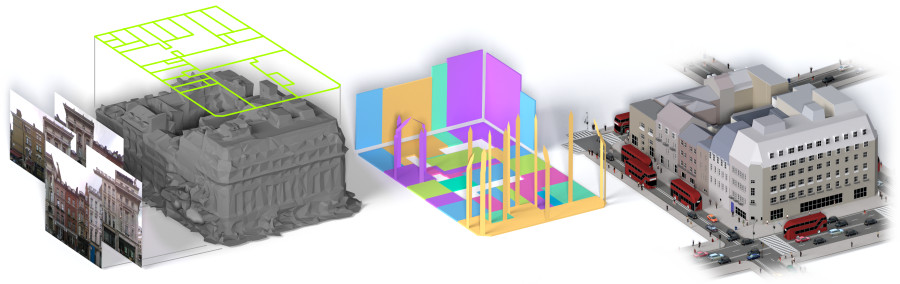
# Outdoor reconstruction



<http://geometry.cs.ucl.ac.uk/projects/2015/regular-arrangements-of-planes/>

Monszpart, A., et al. (2015). "RAPter: rebuilding man-made scenes with regular arrangements of planes." ACM Trans. Graph. **34**(4): 1-12.

<https://github.com/amonszpart/RAPter>

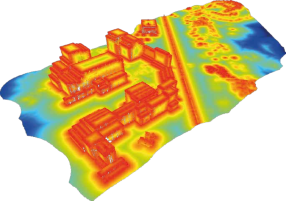
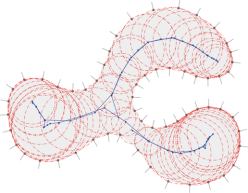


<http://geometry.cs.ucl.ac.uk/projects/2017/bigsur/>

Kelly, T., et al. (2017). "BigSUR: Large-scale Structured Urban Reconstruction."

## Point cloud modelling with the 3D Medial Axis Transform

<https://3d.bk.tudelft.nl/projects/3dsm/>



# Research Plan

## 2017. Oct.

1. Indoor reconstruction developing
   1. floor map and 3D layouts reconstruction by partition-merging method

* Space partition and cell representation
* Adjancency and topology among cells
* Optimized solution for cell merging
* ...
  1. room parsing

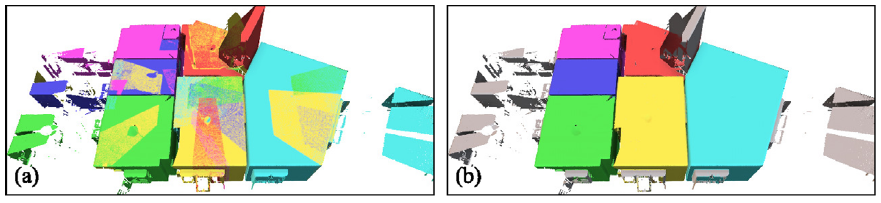
1. Design code structure ok
2. System Testing

1. Research report
2. Collect methods for crowdsoured map update

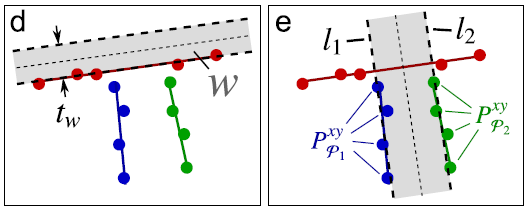
## 10.2 - 10.6

1. Algorithm flow

* Outlier removal (by manual edition)



* Consider the thickness of the wall



* optimization by graph-cut

Oesau, S., et al. (2013). Indoor scene reconstruction using primitive-driven space partitioning and graph-cut. Eurographics workshop on urban data modelling and visualisation.

如何做多个分割

Mean-shift

Graphcut

* 二维上做cell的合并

1. system develop

* Iplcore
* classifier
* utest

1. programming style guide

Refer to PCL, OpenCV and CGAL.

## 10.10 - 10.14

1. 系统设计和代码重构

(1.1) 重要的数据结构直接重定义，在ipl内部采用一套统一的数据结构

Iplcore.h

namespace ipl

{

#if HAVE\_PCL

using pcl::PointCloud;

template <typename PointT>

using iplPointCloud = pcl::PointCloud<PointT>;

template <typename PointT>

using iplPointCluster = pcl::PCLBase<PointT>;

#else

// template <typename PointT>

// class iplPointCloud : public pcl::PointCloud<PointT>

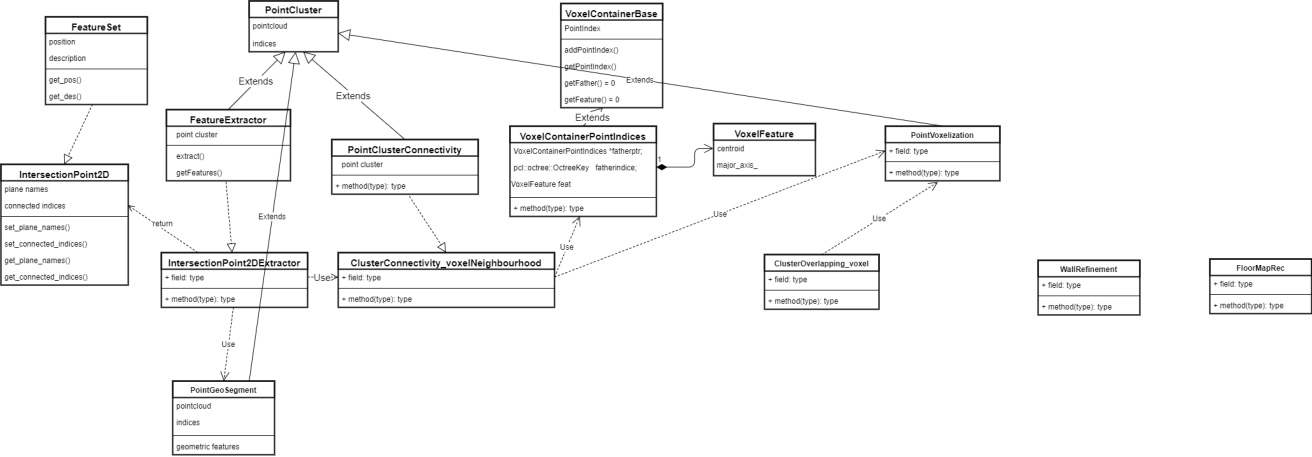
// {

// };

#endif

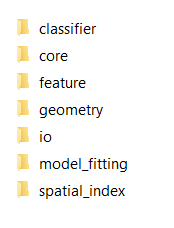
};

(1.2) 定义部分接口类



(1.3) 代码重构

已完成registration外的全部代码重构，有些地方的结构还不满意，需要进一步重构



每个模块都已完成utest

1. 室内模型重建的算法开放

(2.1) 算法模块

* Wall refinement 80%
* Space subdivision by lines 30%
* Extract room layout by graphcut 0%
* Regularization 0%

## 10.16 - 10.21

1. Floormap reconstruction flow chat

* Load wall segments;
* Construct line arrangement;
* Traverse facet to construct a graph
* Use graph-cut to find interior and exterior
* Connect the edges of border facets
* Add inner edges
* Adjust edges

1. the multi-label optimization

构造无向图



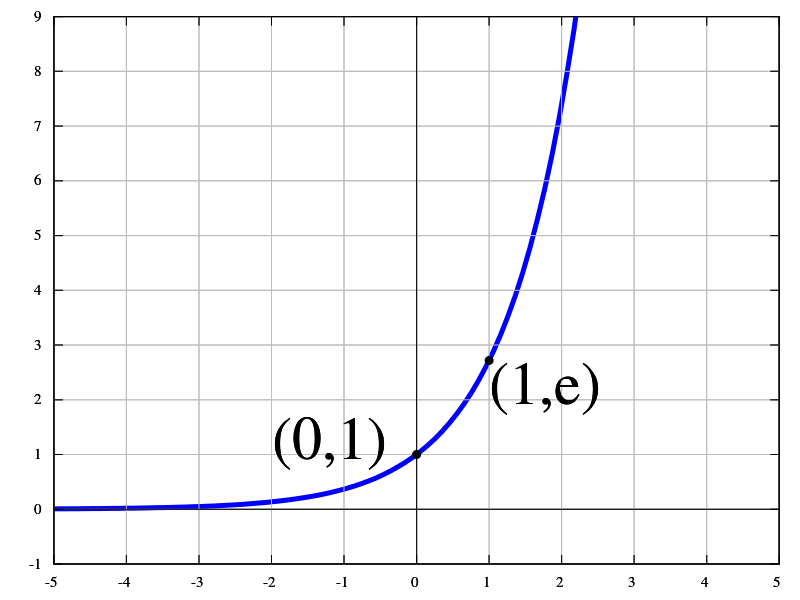
V是空间剖分后得到的facet，E是相邻facet之间的连接边。该图是空间划分后得到的图的简化与对偶表示。







* Unary cost (data cost)





 存在点的面积与总面积之比

* Binary cost (smooth cost)



用于调节两个数据项

1. Building model

* Data structure for building model
* Model I/O

Polygon\_2D

Polyhedral\_3D

1. manual editing

* DOM/DSM
* Model overlapping

extrude

## 10.23 - 10.28

CGAL中三种合并facet的方法

int main ( )

{

// Create an arrangement of four l ine segments .

Arrangement arr ;

Segment s1(Point (1 , 3) , Point (5 , 3));

Halfedge\_handle e1 = arr . insert\_in\_face\_interior( s1 , arr . unbounded\_face( ) ) ;

Segment s2(Point (1 , 4) , Point (5 , 4));

Halfedge\_handle e2 = arr . insert\_in\_face\_interior( s2 , arr . unbounded\_face( ) ) ;

insert (arr , Segment(Point (1 , 1) , Point (1 , 6) ) ) ; // s3

insert (arr , Segment(Point (5 , 1) , Point (5 , 6) ) ) ; // s4

std : : cout << "The␣ i n i t i a l ␣arrangement :" << std : : endl ;

print\_arrangement( arr ) ;

//Remove e1 and i t s incident v e r t ic e s using the member function remove\_edge ( ) .

Vertex\_handle v1 = e1−>source ( ) , v2 = e1−>target ( ) ;

arr . remove\_edge(e1 ) ;

remove\_vertex( arr , v1 ) ;

remove\_vertex( arr , v2 ) ;

// Remove e2 using the free remove\_edge () function .

remove\_edge( arr , e2 ) ;

std : : cout << "The␣ f ina l ␣arrangement : " << std : : endl ;

print\_arrangement( arr ) ;

return 0;

}

// File : ex\_observer . cpp

#include <CGAL/ basic . h>

#include <CGAL/Arr\_observer.h>

#include "arr\_exact\_construction\_segments .h"

#include "arr\_print .h"

// An observer that receives not ifications of face spl i t s and face mergers .

class Faces\_observer : public CGAL: : Arr\_observer<Arrangement> {

public :

Faces\_observer(Arrangement& arr ) : CGAL: : Arr\_observer<Arrangement\_2>(arr ) {}

virtual void before\_split\_face(Face\_handle , Halfedge\_handle e )

{

std : : cout << "−>␣The␣insertion␣of␣: ␣␣[ ␣" << e−>curve ( )

<< "␣ ] ␣␣causes␣a␣face␣to␣ spl i t . " << std : : endl ;

}

virtual void before\_merge\_face (Face\_handle , Face\_handle , Halfedge\_handle e )

{

std : : cout << "−>␣The␣removal␣ of ␣ : ␣␣ [ ␣" << e−>curve ( )

<< "␣ ] ␣␣causes␣two␣faces␣to␣merge. " << std : : endl ;

}

};

int main ( )

{

// Construct an arrangement containing a s ingle face of a diamond shape .

Arrangement arr ;

Faces\_observer obs( arr ) ;

insert\_non\_intersecting\_curve ( arr , Segment(Point(−1, 0) , Point (0 , 1) ) ) ;

insert\_non\_intersecting\_curve (arr , Segment(Point (0 , 1) , Point (1 , 0) ) ) ;

insert\_non\_intersecting\_curve ( arr , Segment(Point (1 , 0) , Point (0 , −1)));

insert\_non\_intersecting\_curve ( arr , Segment(Point (0 , −1), Point(−1, 0) ) ) ;

// Insert a ver t ical segment dividing the diamond into two , and a

// a horizontal segment further dividing the diamond into four .

Segment s\_v(Point (0 , −1), Point (0 , 1));

Halfedge\_handle e\_v = insert\_non\_intersecting\_curve ( arr , s\_v) ;

insert ( arr , Segment(Point (−1, 0) , Point (1 , 0) ) ) ;

print\_arrangement\_size ( arr ) ;

// Remove a portion of the ver t ical segment .

remove\_edge( arr , e\_v) ; // the observer wi l l make a notification

print\_arrangement\_size ( arr ) ;

return 0;

}

//! \file examples/Arrangement\_2/ex\_observer.cpp

// Using a simple arrangement observer.

#include <CGAL/Cartesian.h>

#include <CGAL/Quotient.h>

#include <CGAL/MP\_Float.h>

#include <CGAL/Arr\_segment\_traits\_2.h>

#include <CGAL/Arrangement\_2.h>

#include <CGAL/Arr\_observer.h>

typedef CGAL::Quotient<CGAL::MP\_Float> Number\_type;

typedef CGAL::Cartesian<Number\_type> Kernel;

typedef CGAL::Arr\_segment\_traits\_2<Kernel> Traits\_2;

typedef Traits\_2::Point\_2 Point\_2;

typedef Traits\_2::X\_monotone\_curve\_2 Segment\_2;

typedef CGAL::Arrangement\_2<Traits\_2> Arrangement\_2;

// An arrangement observer, used to receive notifications of face splits and

// face mergers.

class My\_observer : public CGAL::Arr\_observer<Arrangement\_2>

{

public:

My\_observer (Arrangement\_2& arr) :

CGAL::Arr\_observer<Arrangement\_2> (arr)

{}

virtual void before\_split\_face (Face\_handle,

Halfedge\_handle e)

{

std::cout << "-> The insertion of : [ " << e->curve()

<< " ] causes a face to split." << std::endl;

}

virtual void before\_merge\_face (Face\_handle,

Face\_handle,

Halfedge\_handle e)

{

std::cout << "-> The removal of : [ " << e->curve()

<< " ] causes two faces to merge." << std::endl;

}

};

int main ()

{

// Construct the arrangement containing one diamond-shaped face.

Arrangement\_2 arr;

My\_observer obs (arr);

Segment\_2 s1 (Point\_2(-1, 0), Point\_2(0, 1));

Segment\_2 s2 (Point\_2(0, 1), Point\_2(1, 0));

Segment\_2 s3 (Point\_2(1, 0), Point\_2(0, -1));

Segment\_2 s4 (Point\_2(0, -1), Point\_2(-1, 0));

insert\_non\_intersecting\_curve (arr, s1);

insert\_non\_intersecting\_curve (arr, s2);

insert\_non\_intersecting\_curve (arr, s3);

insert\_non\_intersecting\_curve (arr, s4);

// Insert a vertical segment dividing the diamond into two, and a

// a horizontal segment further dividing the diamond into four:

Segment\_2 s\_vert (Point\_2(0, -1), Point\_2(0, 1));

Arrangement\_2::Halfedge\_handle

e\_vert = insert\_non\_intersecting\_curve (arr, s\_vert);

Segment\_2 s\_horiz (Point\_2(-1, 0), Point\_2(1, 0));

insert\_curve (arr, s\_horiz);

std::cout << "The initial arrangement size:" << std::endl

<< " V = " << arr.number\_of\_vertices()

<< ", E = " << arr.number\_of\_edges()

<< ", F = " << arr.number\_of\_faces() << std::endl;

// Now remove a portion of the vertical segment.

remove\_edge (arr, e\_vert);

std::cout << "The final arrangement size:" << std::endl

<< " V = " << arr.number\_of\_vertices()

<< ", E = " << arr.number\_of\_edges()

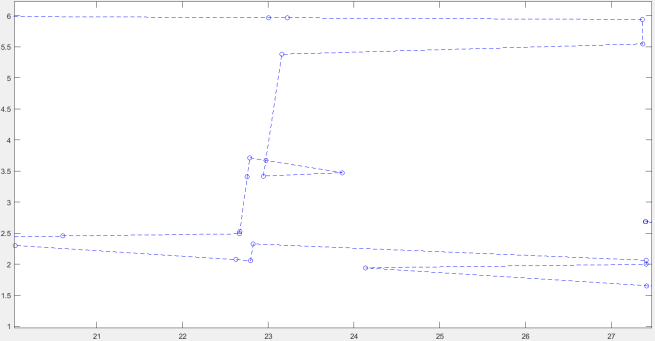
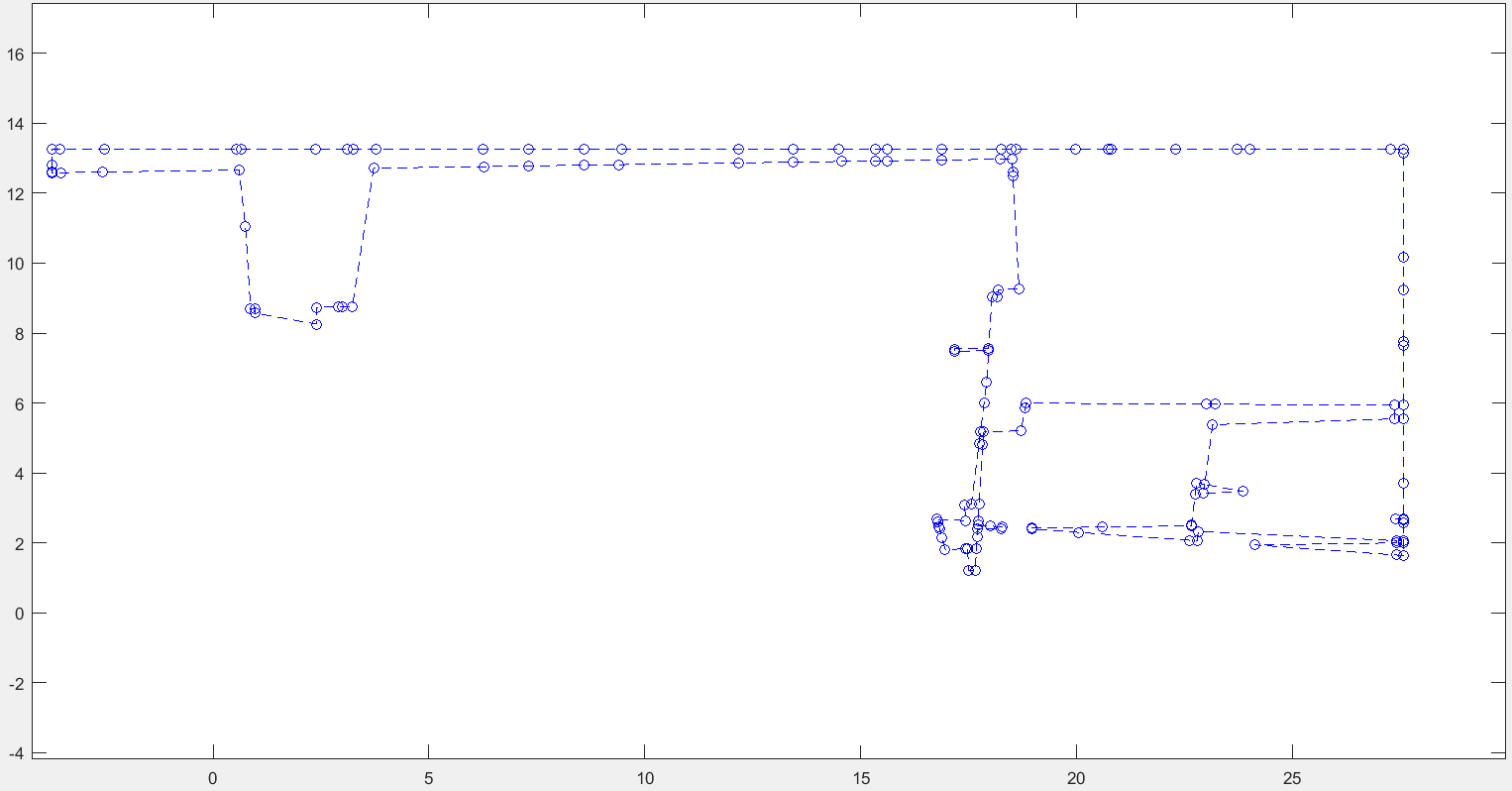
<< ", F = " << arr.number\_of\_faces() << std::endl;

return (0);

}

## 11.14 - 11.18

(1) Self intersection polygon



Arrangement\_on\_surface\_2

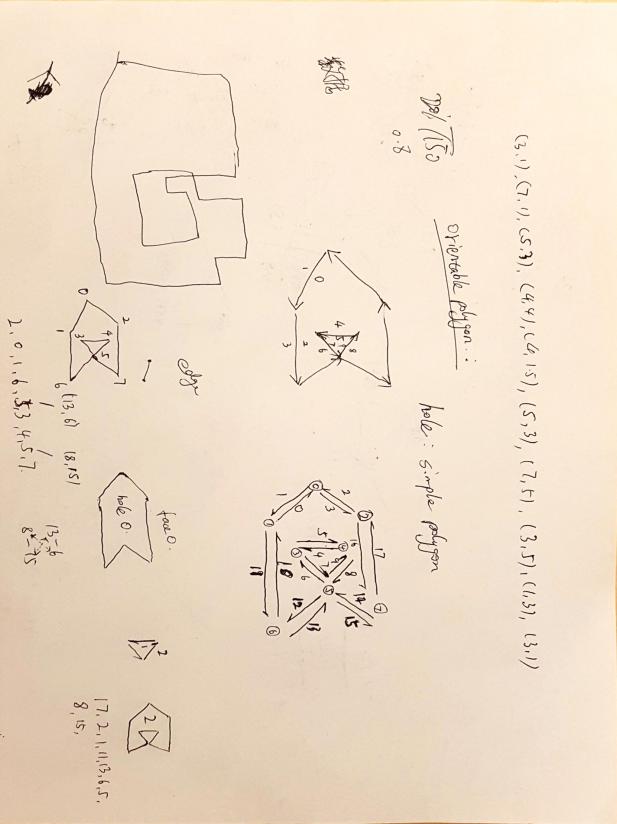
不能直接修改HDS结构，封装在Dcel中，提供Arr\_accessor进行修改，但并没有修改边的连接顺序的接口。

protected:

friend class Arr\_observer<Self>;

friend class Arr\_accessor<Self>;

理想的解决方案



h8->set\_next(h6);

h6->set\_prev(h8);

h13->set\_next(h15);

h15->set\_prev(h13);

替代方案：找到ring边，作为内部hole，多边形直接导出到Polyhedron中

Polyhedron P;

polyhedron\_builder<HalfedgeDS> builder(points\_, flist, hlist.size());

P.delegate(builder);

## 11.20 - 11.25

1. Develop an algorithm to repair self-intersection polygons and test it. This algorithm firstly counts the winding number of a polygon and then if the winding number is odd, which means existing self-intersection, a ring detector is started to find the vertices causing self-intersection, finally the ring is removed and the gap vertices are connected to fill the hole.
2. investigate current 3D modelling workflow and design an optimized flow, write a report which has been submitted to Zhe.
3. integrate developed algorithm to existed flow, the output need to be supported in ESRI software. So some GIS modules should be added to our ipl system. Two opensource libraries GDAL and GEOS are buit from source and integrated to ipl. Develop two test programs to test GDAL output and topology analysis.
4. develop functions to output reconstruction results into ESRI directly. The results have been tested by Xueyang and worked well.

## 11.27 - 12.2

# Processing flow

## utest\_iplclassifier

Input:

raw point cloud \*.pcd

Output:

[output\_dir]/wall\_pts.pcd

[output\_dir]/ceiling\_pts.pcd

[output\_dir]/outlier\_pts.pcd

Description:

* 用法向量区分垂直和水平点
* 用高度直方图统计从水平点中提取ceiling和unclassified

## utest\_model\_fitting

Input:

wall\_pts.pcd

--doRANSAC

Output:

[output\_dir]/eRansac

Description:

Ransac算法提取平面，每个平面保存为独立的点云和平面参数文件

## utest\_wall\_refinement

Input:

[pointcloud\_dir]

[.pcd] file extension

Output:

[filtered file dir]

Description:

* 过滤非显著性的墙面
* 利用PointDistribution\_wall分析候选墙面的点云分布
* float areaTh, float sparseTh, float rmsTh, float maxGirderTh
* 占据面积，稀疏度，拟合中误差，墙面最小高度四个阈值进行判断
* 墙面融合
* 合并由测量误差造成的重影(单面墙)
* 合并双面墙(同一墙体的两个面)

## Utest\_floormapRec

Input:

[pointcloud\_dir]

[.pcd] file extension

Output:

boundary.shp

Description:

* 利用墙面的2D投影对空间进行划分成独立的多边形单元
* 利用图割算法对多边形单元进行合并，提取室内平面图layout
* 导出矢量数据

## Todo

门窗的提取思路：

PointDistribution\_wall

提取轴对齐的分布voxel，

采用图像处理方法

作为墙面上的空洞，从影像中提取最大的空洞，然后进行约束条件下的拟合

http://xcd.blog.techweb.com.cn/archives/91.html

# The optimization algorithm

The target is seperating the facets into indoor or outdoor sets. It belongs to the classical label optimization problem.

the label optimization

构造无向图



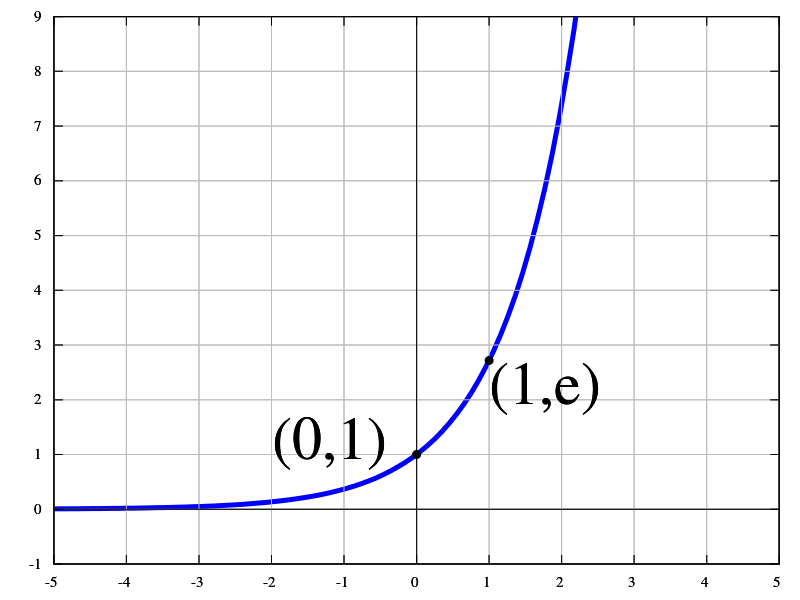
V是空间剖分后得到的facet，E是相邻facet之间的连接边。该图是空间划分后得到的图的简化与对偶表示。







* Unary cost (data cost)





 存在点的面积与总面积之比

* Binary cost (smooth cost)







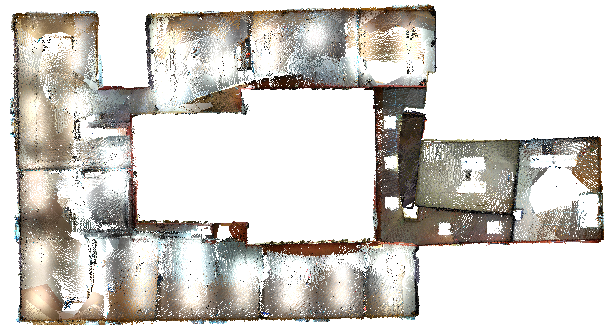
用于调节两个数据项

# Experiments

## Dataset Description

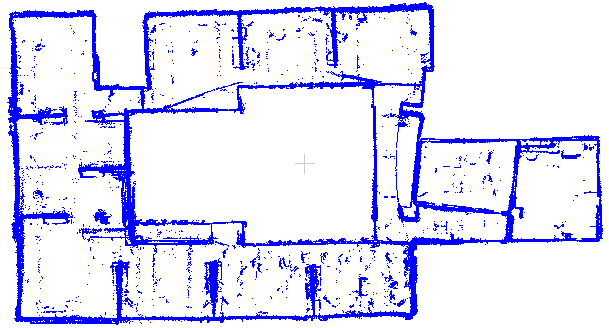
(1) Raw pointcloud

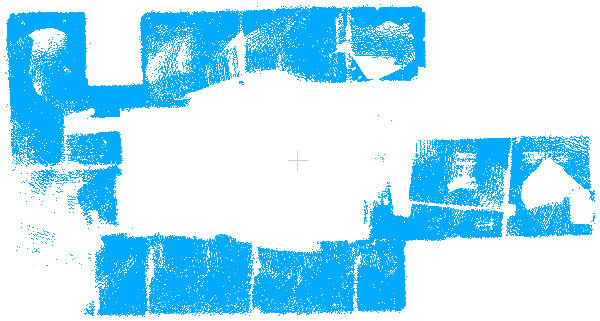
..\sample1\ref\_dataset\ref\_pointcloud\_outlierRemoval.pcd



1. point cloud classification

* ..\sample1\ref\_dataset\result\wall\_pts.pcd
* ..\sample1\ref\_dataset\result\ceiling\_pts.pcd
* ..\sample1\ref\_dataset\result\outlier\_pts.pcd







1. plane fitting

..\sample1\ref\_dataset\result\eRansac\

1. wall refinement

..\sample1\ref\_dataset\result\filtered\

1. floor map reconstruction

..\sample1\ref\_dataset\result\polygons\

## Results Analysis

### Parameter description

(1) Parameters for wall refinement

* float areaTh 最小面积
* float sparseTh 稀疏度(反应空洞大小)
* float rmsTh 平面拟合中误差(反应噪声水平)
* float maxGirderTh 最小高度(梁等结构)
* float vsize voxel格网大小(重影和双面墙)

(2) Parameters for reconstruction

* float gridsize 格网大小(用于计算face和edge的属性)
* double beta\_ 光滑项系数(graph cut中用于平衡face和edge的权重)

### Experiment I：The influence of beta (GraphCut)

float areaTh = 4.0

float sparseTh = 0.4

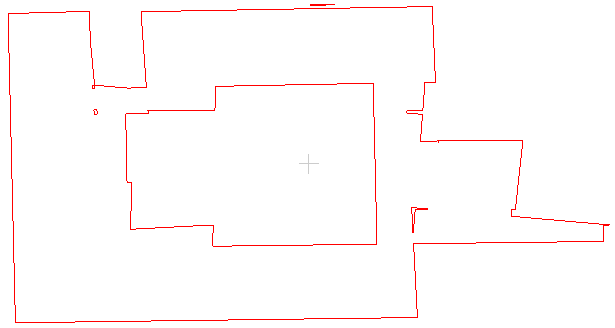
float rmsTh = 0.1

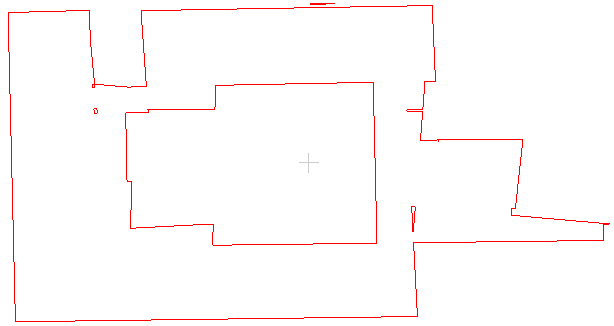
float maxGirderTh = 0.3

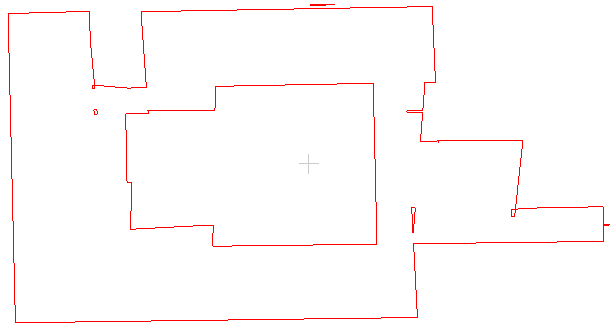
float vsize = 0.2

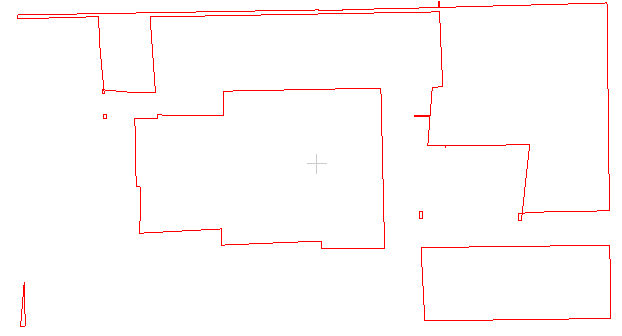
float gridsize = 0.2

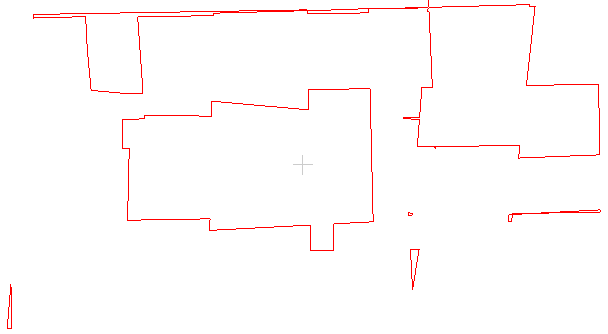
double beta\_ = 0.0 / 0.05 / 0.08/ 0.1 / 1.0











### Experiment II：The influence of wall refinement

float areaTh = 4.0

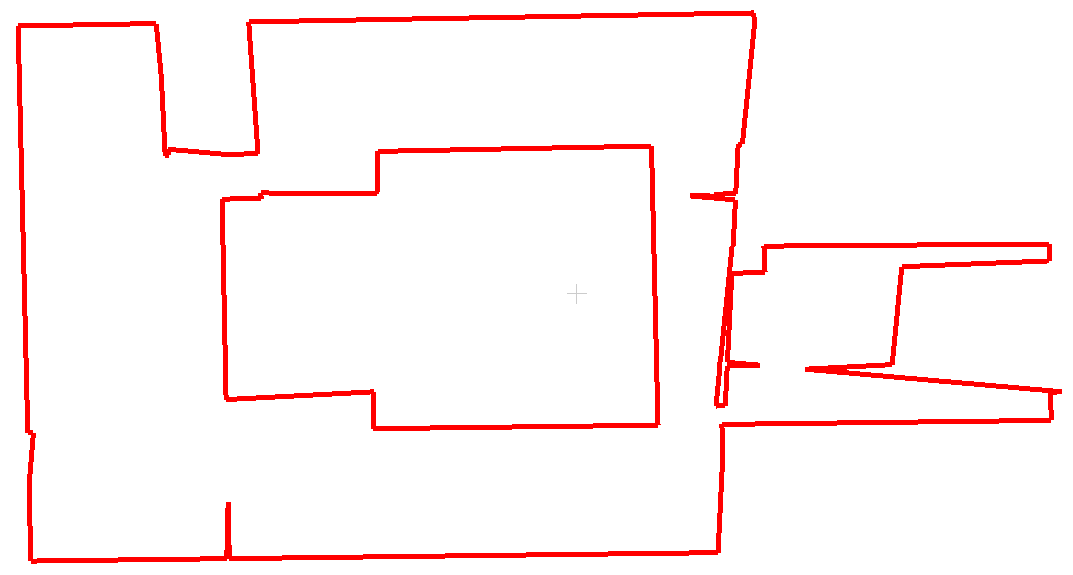
float sparseTh = 0.4

float rmsTh = 0.1

float maxGirderTh = 0.3

float vsize = 0.2

float gridsize = 0.2



### Experiment III: The influence of grid size

float areaTh = 4.0

float sparseTh = 0.4

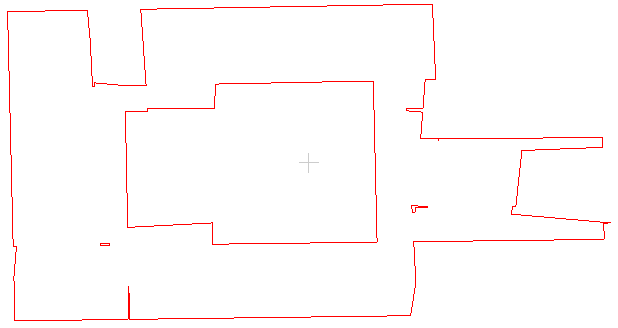
float rmsTh = 0.1

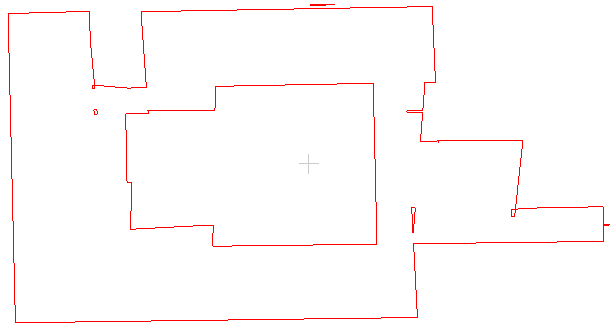
float maxGirderTh = 0.3

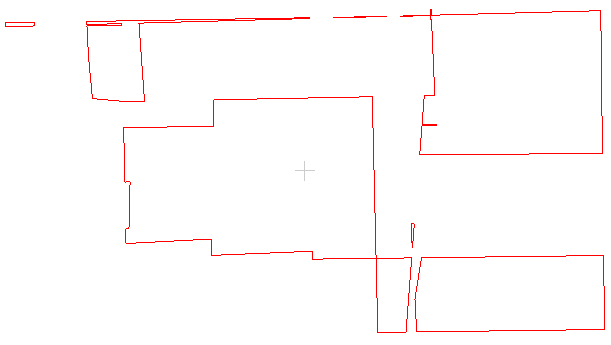
float vsize = 0.2

double beta\_ = 0.08

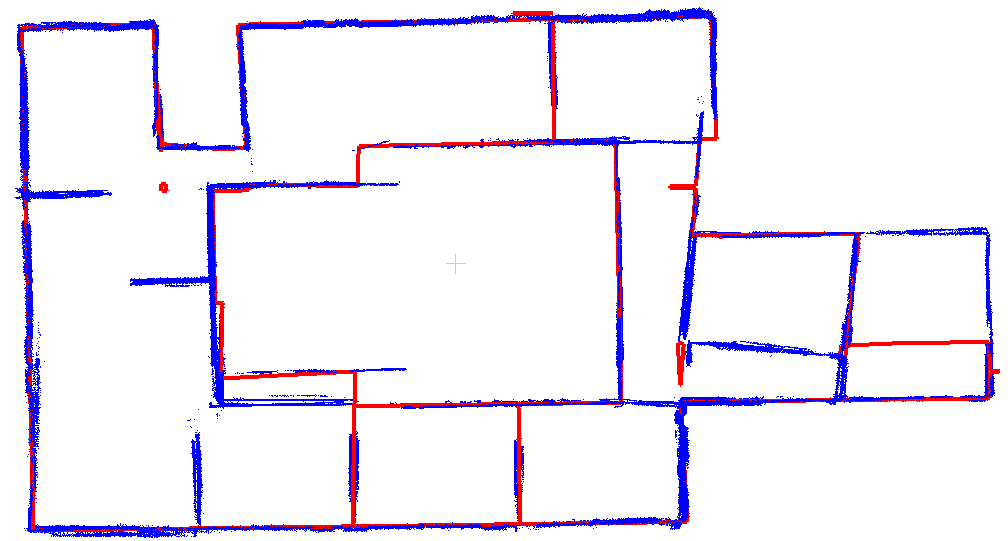
float gridsize = 0.1 / 0.2 / 0.3

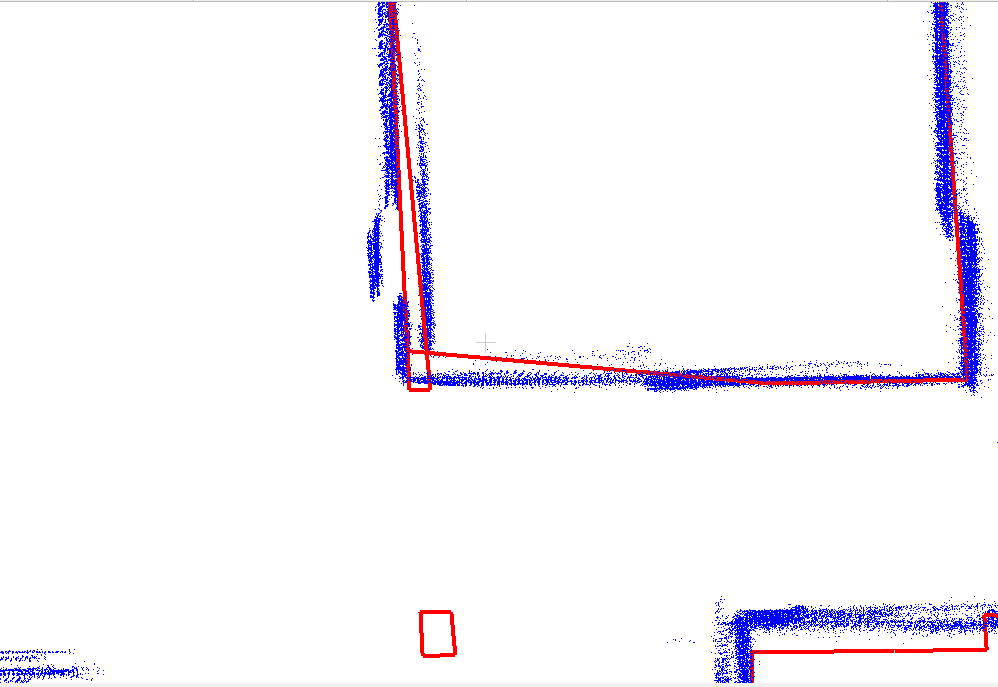


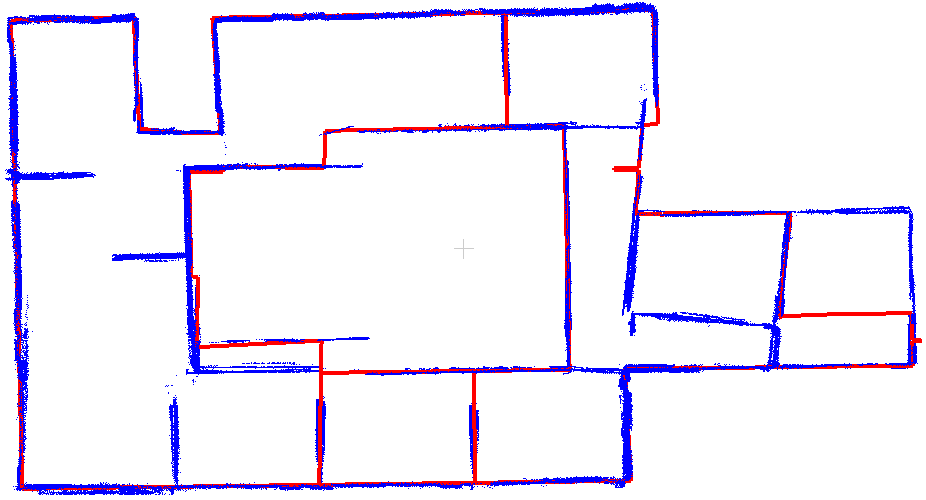


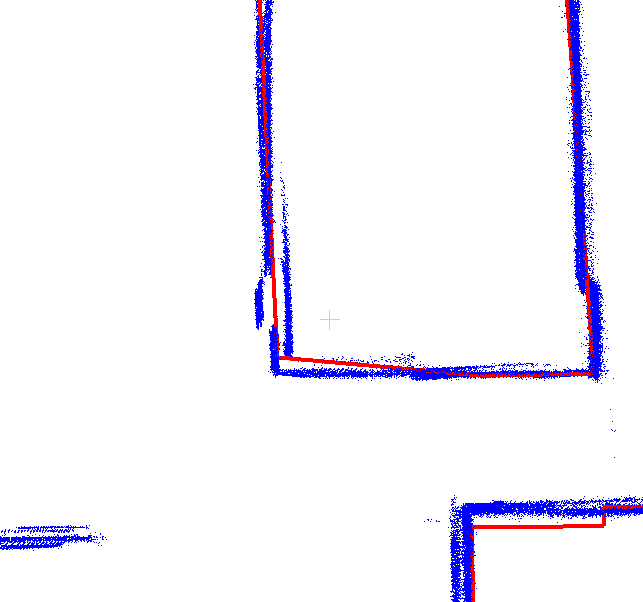


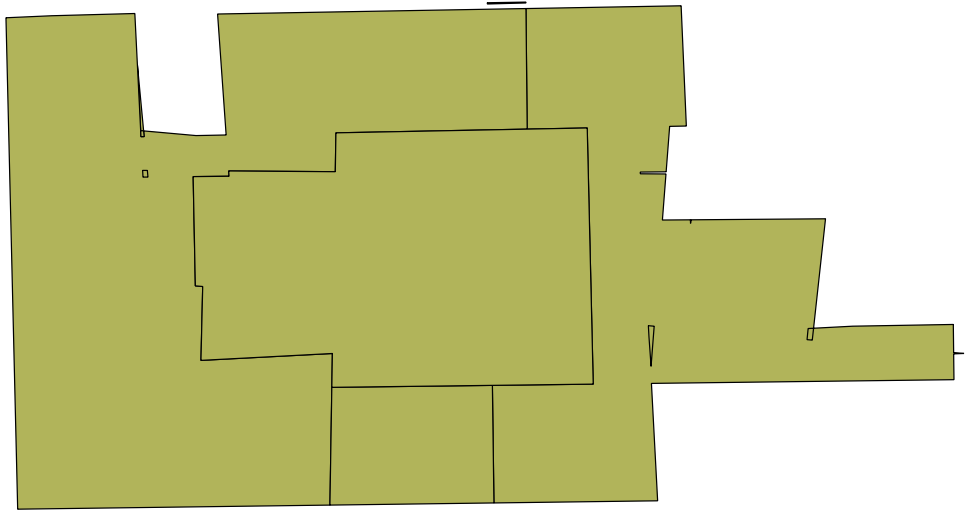
### Experiment IV: eliminate tiny polygons

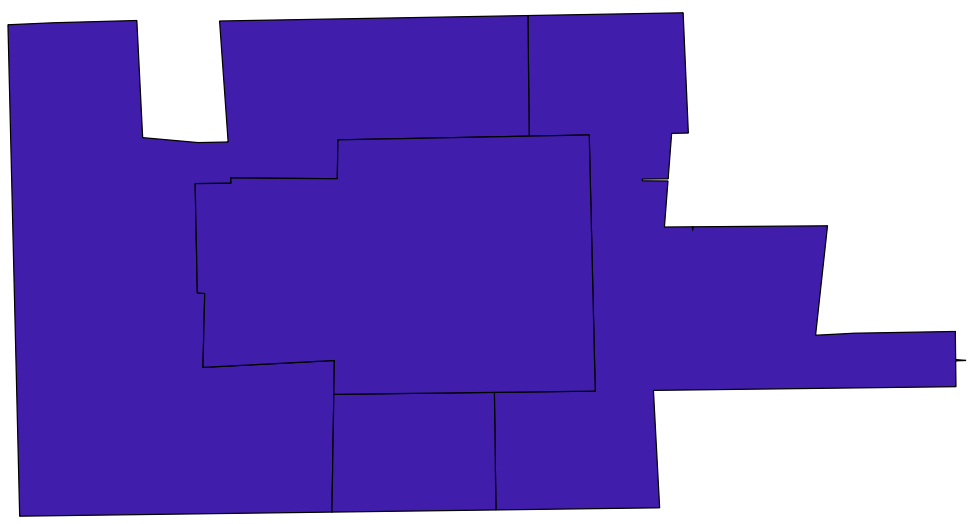




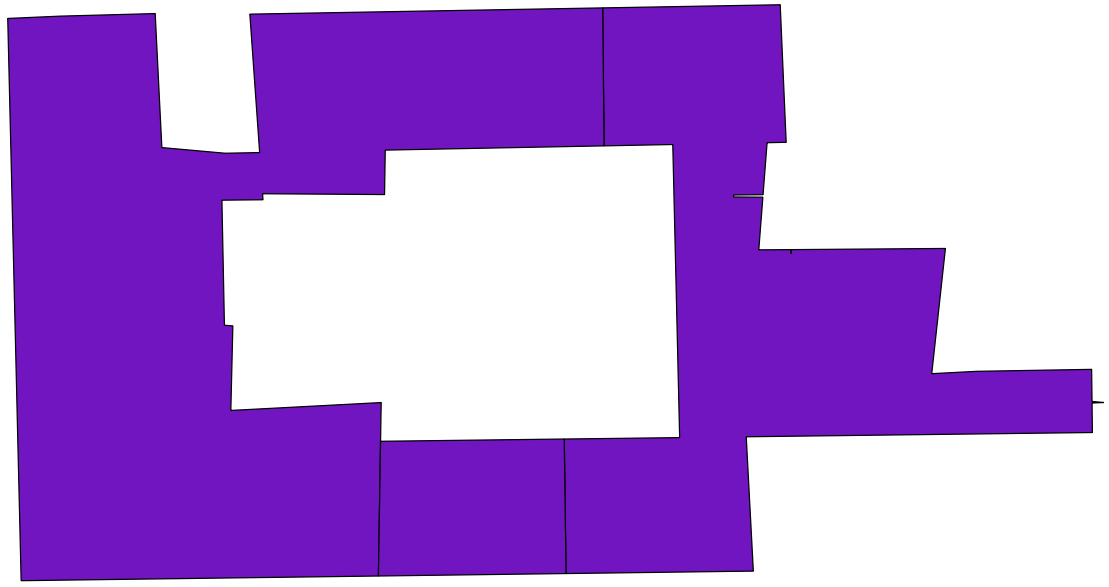






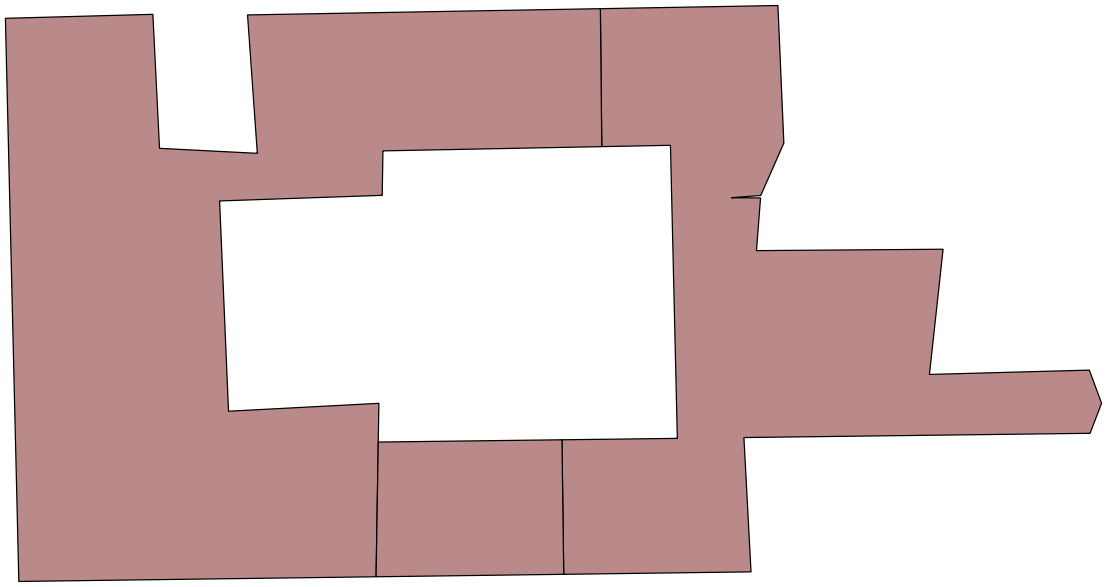


### Experiment V: Polygon simplification



1. Douglas-Peucker algorithm with Topology Preserving

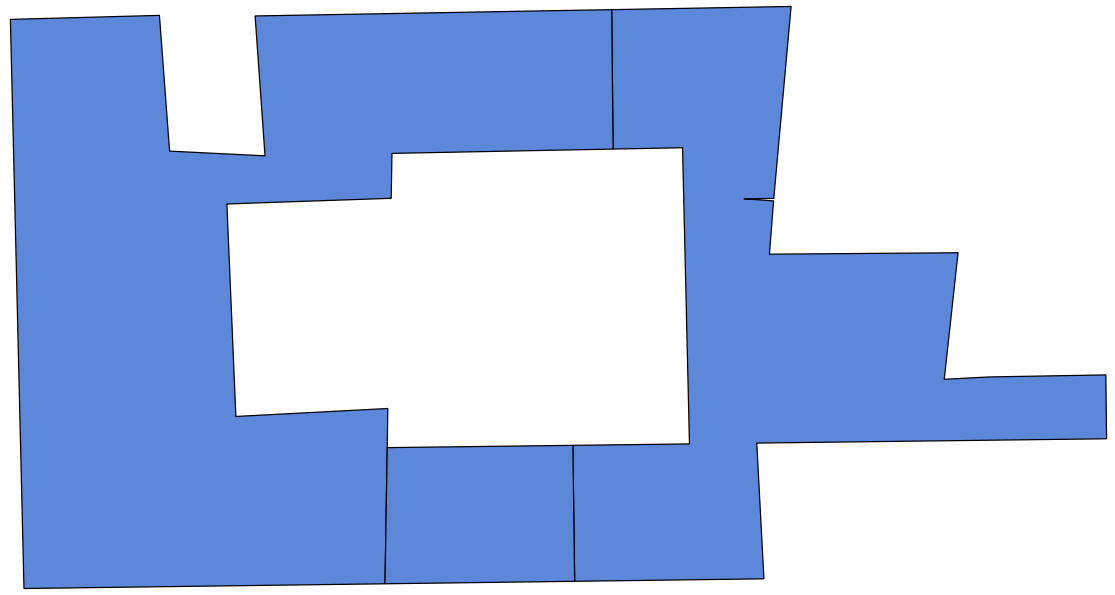
dTolerance = 0.5



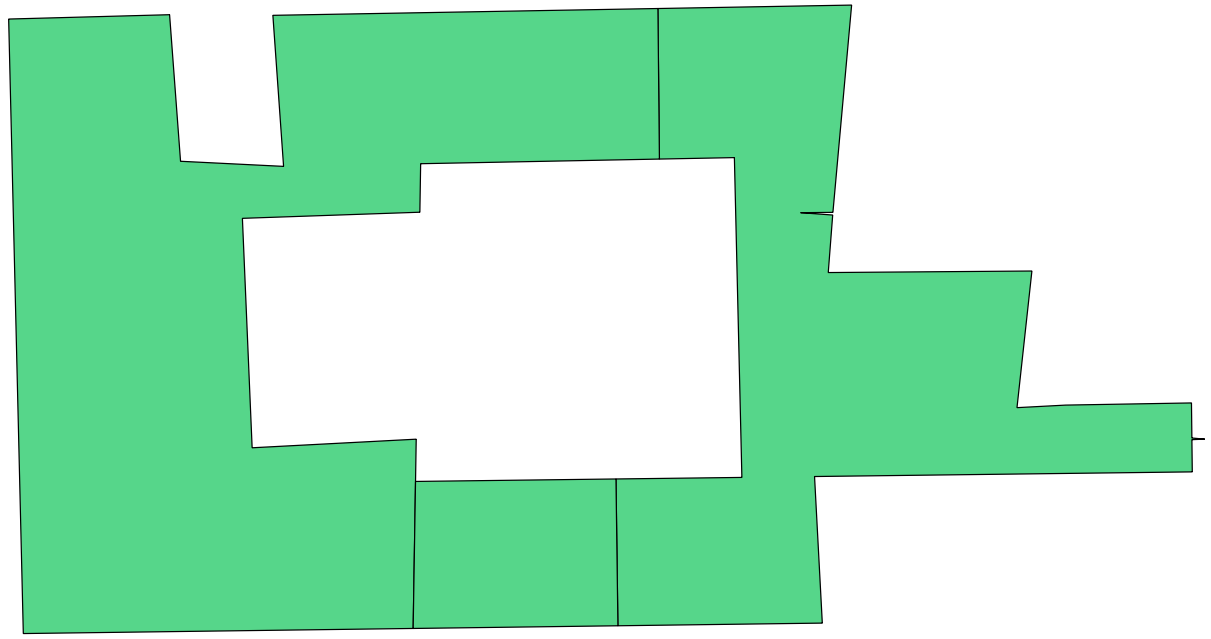
1. polylines simplification based on Dyken et al., “Simultaneous curve simplification”

|  |  |
| --- | --- |
| Squared distance cost | calculates the cost as the square of the distance between the original and simplified polylines |
| Scaled squared distance cost | calculates the cost as a scaled variant of the square of the distance between the original and simplified polylines. |
| Hybrid squared distance cost | calculates the cost as the square of the distance between the original and simplified polylines, possibly scaled based on a factor. |

Squared\_distance\_cost 0.5m



Scaled\_squared\_distance\_cost 0.1



Hybrid\_squared\_distance\_cost cost(FT ratio) = 0.5; Stop = 0.5

Recommended method, is insensitive to parameters, and it is not over-simplified.

