**1 代码及功能文件对应说明**

|  |  |  |
| --- | --- | --- |
| 编号 | 文件名 | 对应功能 |
| 1 | CloudRegister.cpp | 入口文件，封装对外提供的接口 |
| 2 | CADModel.cpp | 图纸解析及3D建模，包含输出各项排序功能 |
| 3 | CloudSegment.cpp | 点云粗分割及粗配准 |
| 4 | TransformOptimize.cpp | 点云精配准,平面方程计算,图纸关联 |
| 5 | CloudBorder.cpp | 点云门洞边框检测 |
| 6 | CloudRefine.cpp | 根据墙面及门洞边框进行多余部分点云裁剪精修 |
| 7 | CalcBayAndDepthMeasure.cpp | 开间以及进深测量 |
| 8 | CalcCorner.cpp | 阴阳角测量 |
| 9 | CalcHoleMeasure.cpp | 门洞测量 |
| 10 | CalcNetHeight.cpp | 净高及极差计算 |
| 11 | CalcRootFlatness.cpp | 地面平整度计算 |
| 12 | CalcSquareness.cpp | 方正度计算 |
| 13 | CalcWallFlatness.cpp | 墙面平整度 |
| 14 | CalcWallVerticality.cpp | 墙面垂直度 |
| 15 | CalcHoleMeasure.cpp  funHelper.cpp  GeometryUtils.cpp | 底层基础计算函数 |

**2 接口说明**

**2.1 输入接口说明**

图纸解析细节参见前期提供的示例说明，针对各个子项输入顺序，除地面输入顺序为逆时针外，

墙面，门洞，横梁输入顺序均为顺时针，以正对方向的左下角为起点，参见如下示意图

1|————————|2

| |

| |

0|————————| 3

**2.2 输出接口说明**

输出各子项边框均调整为以正对方向的左下角为起点，顺时针输出，与上图相同

墙面顺序以最大采光面为起点，顺时针输出

各测量接口设置参数说明可参见CloudRegister.h说明，可调整部分参数均已设置默认值

输出默认由三部分组成

1 每项测量的具体测量数字calcMeassurment\_t.value

2 每项测量计算使用的点云范围框 calcMeassurment\_t.rangeSeg，以多个线段方式构成

3 std::vector<seg\_pair\_t> 每项测量指标可视化辅助输出线段信息

**3 各模块设计说明**

**3.1 CADModel**

该模块与最终输出排序相关，点云各项输出顺序在完成与图纸的映射后，与图纸顺序完全保持，例如墙面门洞边框顺序，最大采光面顺时针排序，如果后期需要调整各项输出顺序请在该模块完成相关排序功能，点云输出自动调整。reSortWall函数对应现在的墙面输出排序功能，地面及顶部对应边框排序以墙面排序为准。

**3.2 各测量指标计算**

各测量指标计算方式以接口为准，本文档只描述通用计算部分逻辑和底层通用函数

为了计算提速以及逻辑清晰，所有计算指标均使用以点云检测出的墙/门洞/地面/顶部来计算获取需要计算的点云范围边框，然后再只获得原始的局部点云根据不同的计算方式进行细节指标计算。

以开间计算为例，通过点云地面边框，通过相互投影的方式计算二二平行的且有overlap线段完成开间多个区域划分的计算(参见下面的示意图，投影计算的A1A2,B1B2即为一个计算区间)，随后将门洞门框投影到地面边框，计算出墙面在X或者Y方向有点云的区域范围，以此为基础再计算时间点云范围框，最终使用过滤出的局部点云块完成计算。

A1 A2

------------------------ WallA

| |

| |

--------------------------- WallB

B1 B2

常用基础函数说明：

groupDirectionIndex(horizenSeg, rootBorder, vecVerticalIndex, vecHorizenIndex);

将边框线段rootBorder分为与horizenSeg平行或者垂直的二组数据

std::tie(hasOverlap, s1Pt, e1Pt, s2Pt,e2Pt) = calcOverlap(toSeg,calcSeg);

计算二个平行的线段是否有ovelap,如果hasOverlap为true，则后面4个点有效，分别对应上图中的A1A2B1B2四个点

auto vecPt = createRulerBox(seg, index, thickness, calcHalfPara \* 2);

计算获取局部点云3D尺子边框, seg为尺子边框的中心线， index代表退化的方向索引，例如一面墙的XY二个轴，如果X轴是主要变化方向，那个Y轴就是退化轴，index为1反之为0，thickness退化轴的范围及墙的厚度，calcHalfPara \* 2，尺子的宽度，长度已经由前面的中心线指定

item.rangeSeg = calcBoxSegPair(vecPt);

根据计算出的顶点创建可视化使用的线段

auto vecTmp = getRulerCorners(vecPt);

auto pCloud = filerCloudByConvexHull(pWall, filerPt);

根据尺子过滤出计算需要的点云

**3.3 点云粗分割及粗配准**

输入：原始点云，相关CADModel

输出：分割完成的各墙体对应点云，粗配准的变换举证，处理之后整体模型已进行粗对齐

主要流程：

calibrateDirectionToAxisZ()

对点云坐标的Z轴进行较正，使得天花板点云片为水平状态（平行于XOY平面）；

recordModelBoundingBox()

计算CADModel的AABB包围盒，为后续的裁剪与点云范围选择作依据；

alignCloudToCADModel()

将点云对齐到CAD坐标下：

将包含点云主体形状的点云取出（sliceMainBody）

检测竖直面（平行于OZ）作为墙体候选面，投影到水平面获得2D墙面线段

将投影墙面线段与CAD图的墙面线段（即俯视图）进行匹配（computeSegmentAlignCandidates，chooseTransformByHoles），获得2d变换

以天花板点云高度对齐CAD图天花板即获得高度变换，与以上2d变换组合即获得3d变换

使用3d变换变换点云以粗对齐到CAD坐标

segmentByCADModel()

点云已对齐到CAD坐标系，则可以根据CAD各面的位置将点云各面进行分割：

由于数据量的关系，首先分割稀疏点云，对CAD模型的每个面，先根据面的位置和法向框选目标点云，随后进行平面检测，并进行错误面片的祛除，从而获得稀疏点云中各CAD面的点云（segmentCloudByCADModel）

以所获得的各CAD面对应的稀疏点云面作为聚类中心，将原始稠密点云聚类即获得分割结果（segmentByCADModel）；

调试：

1. 启用预处理宏VISUALIZATION\_ENABLED重新编译运行，即能看到原始点云，粗对齐，分割的可视化窗口，以快速确定问题位置。
2. CloudSegment.cpp重要参数：

alignCloudToCADModel()

auto planes = detectRegionPlanes(cloud, 3./ 180.\* geo::PI, 1., 500); 500决定了稀疏后墙面至少包含的内点数目；

constexpr double MATCH\_THRESH = 0.15; 点云与CAD图纸对齐之后的最大匹配误差；

segmentCloudByCADModel(thecloud)

SLICE\_HALF\_THICKNESS， NORMAL\_CHECK， GROWTH\_ANGLE，三个参数分别是切分厚度、法向偏差、增长法向偏差，决定了粗选墙体的范围；

auto planes = detectRegionPlanes(slice, slicenormals, GROWTH\_ANGLE, 1., slice->size()/ 4); slice->size()/ 4意味着所检测到的墙面片应当至少包含粗选墙体点云25%的点；

if (ratio > 0.7f) { 0.7f表示所检测到的小墙面片应当有70% 都在CAD所表示的墙面实体范围内（例如面片基本分布在门洞范围，肯定是不能要的）；

segmentByCADModel()

constexpr float ON\_PLANE\_CHECK\_THRESH = 0.05f； 稠密点云最大允许的与粗分割面的距离；

**3.4 点云精配准**

getModelPlaneCoeff(cadModel, center);

计算每个墙面方程及根据要求修改平面方程法向，满足计算点到平面的凹凸符号定义

matchCloudToMode();

根据粗对齐的结果完成和图纸的一一关联

downSampling();

optimize(transform);

使用降采样后的点云计算全局变换最优解，地面点云不参与计算，策略为顶部优先

transformCloud(transform);

根据求解出的变换矩阵，将原始所有点云进行变换，统计变换后地面点云到平面的距离中值，重新计算垂直方向的移动T，完成底部对齐

//get plane coeff with input Cloud

Eigen::Vector3d newCenter = transform.block<3, 3>(0, 0) \* center + transform.block<3, 1>(0, 3);

getCloudPlaneCoeff(newCenter);

根据变换后的点云位置重新计算每个平面方程

**3.5 点云边框获取**

点云边框分为墙面外边框和门洞边框

3.5.1墙面外边框

a.外边框的获取主要是通过计算三个相邻点云平面的交点，再将交点和cad线段对应起来而得到的

b.主要代码在CloudRegister.cpp的calcAllCloudBorder函数里：

pushNeedData 获取各墙面的点云平面参数和cad各顶点，以备后面使用

groupPlanesBySamePt 根据cad顶点是否重合，可以知道哪三个墙面是相邻墙面，会共享顶点

interSectionOf3Planes 根据三个相邻墙面的点云平面参数计算实际共享的顶点，即点云三个平面的交点

findNearestSegOfNodeOrCloudPt 依次对每个墙面的每一个外边框cad线段，先在点云三个平面的交点中找到对应的两个点作为外边框线段端点，如果找不到符合条件的（点距离过大或方向偏差过大），则直接在点云里面找最符合条件的两个点作为外边框线段端点（也要满足点距离阈值和方向偏差阈值，如果不满足会有找不到的情况）

c.可视化：

cadSegs-plane-\*pcd是各墙面的cad边框（包含外边框和门洞边框）

cloudSegs-plane-\*pcd是计算得到的各墙面的点云边框（包含外边框和门洞边框）

（\*是指代具体的墙面编号，定位时最好只观察某一个墙面）

3.5.2门洞边框

a.门洞边框的获取主要是通过检测点云直线，然后获取直线交点，再将交点和cad线段对应起来而得到的，会依赖于3.5.1获取的点云外边框

b.主要代码在CloudBorder.cpp中：

prepareData 保存入参，做些预处理（将一面墙点云投影到点云平面，并稀疏化），以备后面使用

getOverlapOfOuterSegAndHoleSeg 计算得到与门洞重合的点云外边框线段，保留下来参与后续计算

detectLinesInCloudWall 在稀疏化后的点云上先得到点云boundaries，然后再检测线段

removeOuterSideLines 在上一步输出的线段中识别并去除外边框线段，保留内边框线段

addCandidateSegs 往上一步输出的线段中加入getOverlapOfOuterSegAndHoleSeg的与门洞重合的点云外边框线段，得到后续参与匹配的检测线段

matchLineToCadHoleSeg　将上一步输出的线段和cad门洞线段做匹配，一个cad门洞线段可匹配多个检测线段；匹配主要依据方向，距离，重合部分的点数，以及点云是否在同一侧来判断的，这里涉及到的可视化是area-hole\*pcd（获取带匹配的检测线段周围的局部点云）和belongLines-\*pcd（seg及其匹配上的线段）

getIntersectionNodeOfLines 在相邻的两个cad seg所属的检测直线中，两两直线求交点，

findNearestSegOfNodeOrCloudPt 依次对每个墙面的每一个门洞边框cad线段，先在上一步输出的交点中找到对应，如果找不到符合条件的（点距离过大或方向偏差过大），则直接在点云里面找对应（也要满足点距离阈值和方向偏差阈值，如果不满足会有找不到的情况）

c.可视化：

origLine-\*pcd是检测到的原始线段

leftLine-\*pcd 是在origLine的基础上，简单识别并去除外边框线段后留下的线段

findLine-\*pcd 是在leftLine的基础上，加上与门洞重合的外边框线段得到的线段

vecNodes-\*pcd是输出的直线交点，可以叠加cadSegs-plane-\*pcd观察

（\*是指代具体的墙面和门洞编号，定位时最好只观察某一个墙面或某一个门洞）