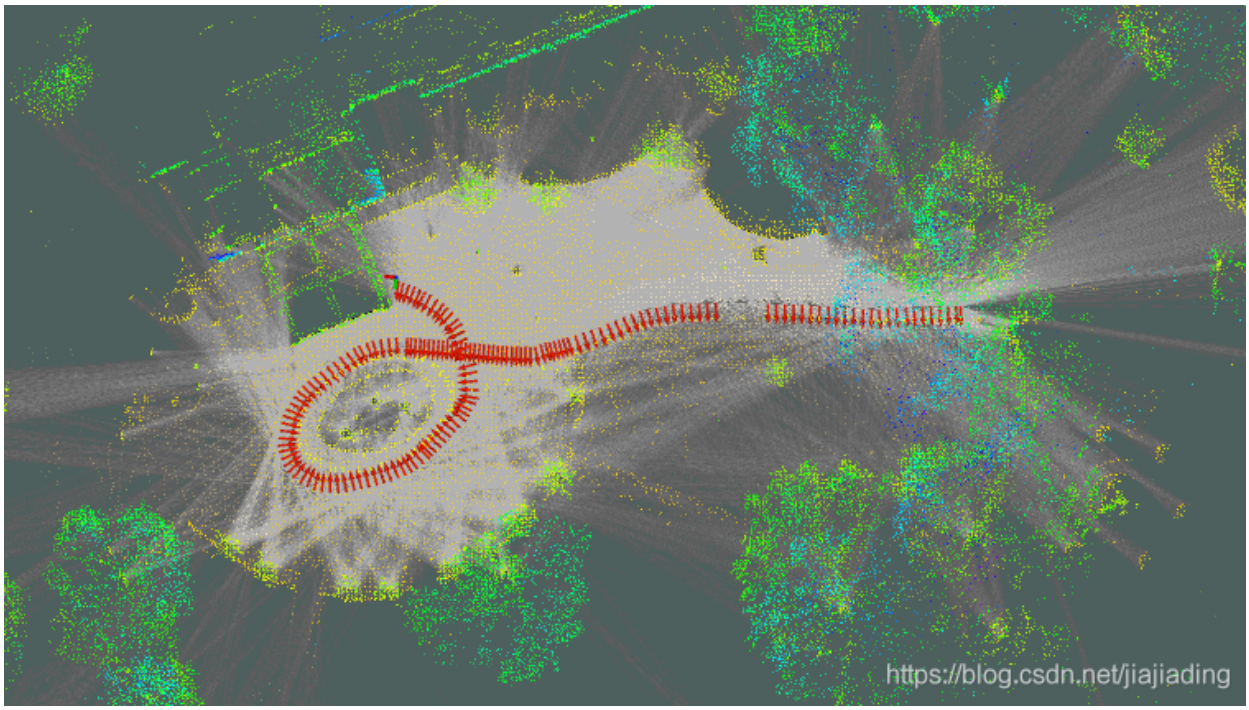
lego-loam 同步构建2d栅格导航地图

基于目前移动机器人的应用可知，目前3d slam存储的主要为点云地图，由于其特征点比2D激光器数据更加丰富，因此用于后期的定位具有更好的抗干扰性和鲁棒性。但是用于导航的基本路径规划功能，目前仍主要依赖于2d栅格地图。

其中16年开源的cartographer的3dslam则同步发布了2d map格式，而存储的点云也是基于stream自定义格式，而不是传统的点云地图。因此定位时可直接使用3d定位结果，2d地图结果进行导航。

本文参考cartographer中2d 栅格概率更新的功能，在lego-loam开源代码中实现其2d栅格地图的同步创建，同时2d地图可自动剔除slam过程中的移动物体（注：lego-loam 创建的3d点云地图，没有剔除）；其效果图如下：



操作步骤如下：

3d点云预处理

在imageProjection.cpp3d点云分割代码中，根据分割后的结果，将地面上的点以及高过机器人高度的所有点云进行剔除。并计算同一个水平扫描ID下的距离值，并保存。如此可获取投影水平面的2D scan message格式。

```
1  for (size_t j = 0; j < _horizontal_scans; ++j) {
```

```
float min_range = 1000;
size_t id_min = 0;
for (size_t i = 0; i < _vertical_scans; ++i) {
size_t Ind = j + (i)*_horizontal_scans;
float Z = _full_cloud->points[Ind].z;
if ((_ground_mat(i, j) != 1) &&
(Z > 0.4) && (Z<1.2) &&
(_range_mat(i, j)<40)) { // 地面上点云忽略，过高过矮的点忽略， 过远的点忽略
 if(_range_mat(i, j) < min_range) { // 计算最小距离
 min_range = _range_mat(i, j);
 id_min = Ind;
 }
 }
 }
 if (min_range<1000) {
 _scan_msg->push_back(_full_cloud->points[id_min]);
 }
 }
```

keypose保存

类似于基本图优化结构和lego-loam存储keypose轨迹序列，并同步记录每个2d点云的笛卡尔坐标。

```
// 将极坐标转换为直角坐标系
for(int i = 0; i < _scan_msg->points.size(); ++i) {
scan_points.emplace_back(_scan_msg->points[i].x, _scan_msg->points[i].y);
}

// 定义新的scan格式，每一束光采用直角坐标
std::shared_ptr<slam::LaserScan> laser_scan(new slam::LaserScan(scan_points));
laser_scan->setId(_scans.size()); // 第一帧激光不做处理，仅记录并放入优化器顶点中
// laser_scan->setPose(Eigen::Vector3f(0, 0, 0));
 laser_scan->setPose(pose); //记录初始激光帧位置，用于slam建图初始坐标（即创建地图坐标系）
 laser_scan->transformPointCloud(); //根据激光位置，计算每个点的在map的位置
 _scans.push_back(laser_scan); //收集每帧激光
```

```
}
```

根据闭环条件更新2d map

闭环条件由lego-loam 3d slam 触发和后端优化，根据3d位置更新 2d投影位置。

```
// 若存在闭环处理，则需要对位姿进行修正，将历史的的位姿用优化后的数据进行更新
void MapOptimization::correctPoses() {
 if (aLoopIsClosed == true) {
 recentCornerCloudKeyFrames.clear();
 recentSurfCloudKeyFrames.clear();
 recentOutlierCloudKeyFrames.clear();
 // update key poses
 int numPoses = isamCurrentEstimate.size();
 for (int i = 0; i < numPoses; ++i) {
  cloudKeyPoses3D->points[i].x =
  isamCurrentEstimate.at<Pose3>(i).translation().y();
  cloudKeyPoses3D->points[i].y =
  isamCurrentEstimate.at<Pose3>(i).translation().z();
  cloudKeyPoses3D->points[i].z =
  isamCurrentEstimate.at<Pose3>(i).translation().x();

  cloudKeyPoses6D->points[i].x = cloudKeyPoses3D->points[i].x;
  cloudKeyPoses6D->points[i].y = cloudKeyPoses3D->points[i].y;
  cloudKeyPoses6D->points[i].z = cloudKeyPoses3D->points[i].z;
  cloudKeyPoses6D->points[i].roll =
  isamCurrentEstimate.at<Pose3>(i).rotation().pitch();
  cloudKeyPoses6D->points[i].pitch =
  isamCurrentEstimate.at<Pose3>(i).rotation().yaw();
  cloudKeyPoses6D->points[i].yaw =
  isamCurrentEstimate.at<Pose3>(i).rotation().roll();
  // 更新 2d 投影位置
  _scans[i]->setPose(Eigen::Vector3f(cloudKeyPoses6D->points[i].z,cloudKeyPo
ses6D->points[i].x,cloudKeyPoses6D->points[i].pitch));
  _scans[i]->transformPointCloud();
  }

  aLoopIsClosed = false; // 修正完成
```

```
  }
}
```

构建和2d map

已知每个时刻的2d绝对位置和对应的range_scan的中所有point笛卡尔坐标，基于cartographer中概率地图的生成和更新，从而构建2d栅格地图。 其中bresenham为经典的画线法，用于更新无障碍栅格，而range_scan的端点用来更新障碍栅格。其具体理论可详看:
cartographer 代码思想解读（4）- probability grid地图更新1（https://blog.csdn.net/jiajiading/article/details/108615628）
cartographer 代码思想解读（5）- probability grid地图更新2（https://blog.csdn.net/jiajiading/article/details/108625446）

```
for(const std::shared_ptr<LaserScan>& scan : scans) {
Eigen::Vector2f start = getMapCoords(scan->getPose());
const PointCloud& point_cloud = scan->getTransformedPointCloud();
for(const Eigen::Vector2f& point : point_cloud) {
Eigen::Vector2f end = getMapCoords(point);
std::vector<Eigen::Vector2i> points;
bresenham(start[0], start[1], end[0], end[1], points);

int n = points.size();
 if(n == 0) {
 continue;
 }

 for(int j = 0; j < n - 1; ++j) {
 int index = getIndex(points[j][0], points[j][1]);
 if(value_[index] + log_odds_free_ >= log_odds_min_) {
 value_[index] += log_odds_free_;
 }
 }

 int index = getIndex(points[n - 1][0], points[n - 1][1]);
 if(value_[index] + log_odds_occupied_ <= log_odds_max_) {
 value_[index] += log_odds_occupied_;
 }
```

```
25    }
26    }
```

总结

目前3d slam主要目的是用于移动机器人的后期定位使用，而SLAM主要用于一个新环境的第一次配置。因此，3d定位对应的2d栅格地图十分必要，本文简单理解就是已知定位建图的功能。