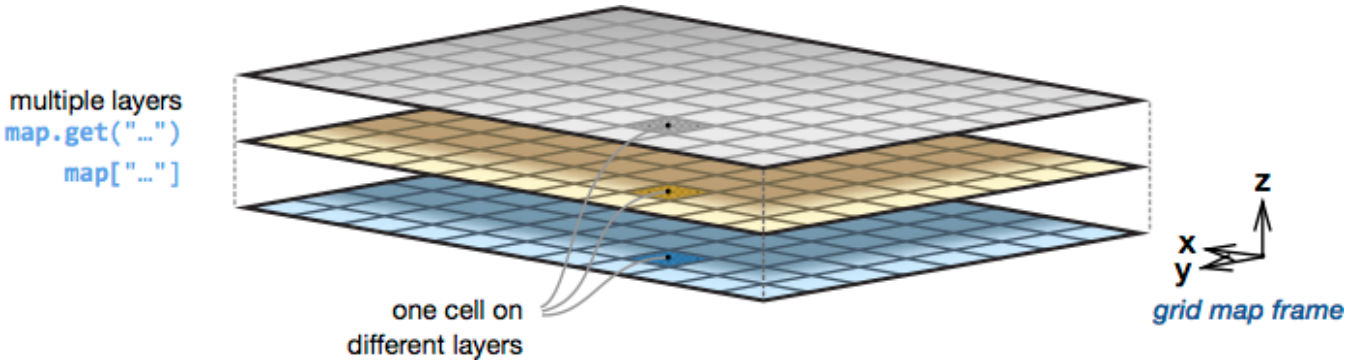
栅格静态障碍物精度提升

1、现状

栅格地图用的是开源代码库grid_map: https://github.com/ANYbotics/grid_map/tree/master



目前栅格地图接入了视觉fs、超声波、激光雷达fs，栅格的分辨率为0.1m，而传感器输出障碍物位置精度小于0.1m，在栅格地图中输出静态障碍物就带来了障碍物位置的精度损失。

特别是在泊车空间受限的场景下，对障碍物位置的精度要求很高，最好就是对下游尽量输出实时障碍物。

传感器精度(远距离和fov边缘误差会变大)：

激光雷达：3cm

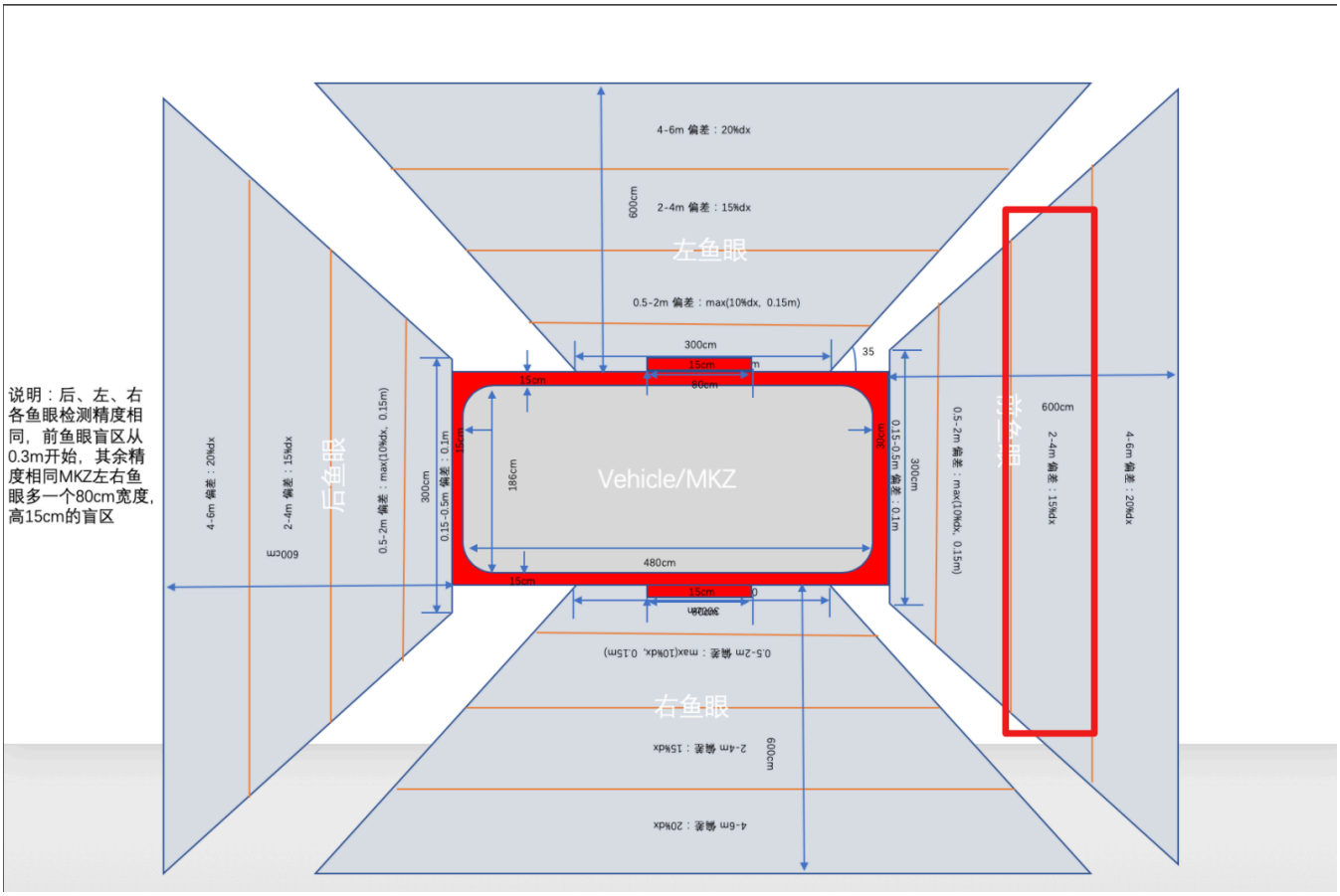
超声波：3-10cm

✧ 【详细测试结果】

➢ 关键指标测试结果

序号	车号	指标名称		测试结果	通过标准	测试是否通过	备注
1	ANP042（量产改装车）	20cm阀值	障碍物坐标准确率	99.74%	98%	测试通过	
2			障碍物坐标召回率	100%	96%	测试通过	
3		10cm阀值	障碍物坐标准确率	99.33%	/	/	指标不做要求
4			障碍物坐标召回率	100%	/	/	指标不做要求
5		5cm阀值	障碍物坐标准确率	87.42%	/	/	指标不做要求
6			障碍物坐标召回率	93.29%	/	/	指标不做要求

视觉fs：10-20cm



2、精度提升方案

方案一、栅格分辨率提升到0.05m

优点：提升障碍物位置输出精度，改动量很小

缺点：1、栅格数量增加4倍，内存增大、栅格更新时长增加，CPU增加

2、栅格更加稀疏，影响对障碍物形状的拟合检测

方案二、每一个图层再多增加对应的两个图层

对于用于存放传感器检测障碍物数据的图层，再额外增加两个图，分别用于存放坐标x和y值。

例如视觉创建freespace图层，再增加两个图层freespace_x和freespace_y。

优点： 1、改动少

缺点：1、栅格地图的图层增加，内存会增长

方案三、修改grid_map源码，修改存储数据的结构体

当前代码中只用到源码中的grid_map_core和 grid_map_cv两部分。开源协议是BSD 3-Clause license.

master 16 branches 21 tags

mgaertneratanybotics and CI Merge branch 'fix/grid_map_core/reduce_even_... d6f0912 10 days

grid_map	Merge remote-tracking branch 'origin/release-22.07'
grid_map_core	Merge branch 'fix/grid_map_core/reduce_even_more_clang_warnings
grid_map_costmap_2d	Merge remote-tracking branch 'origin/release-22.07'
grid_map_cv	Merge remote-tracking branch 'origin/release-22.07'
grid_map_demos	Merge remote-tracking branch 'origin/release-22.07'
grid_map_filters	Merge remote-tracking branch 'origin/release-22.07'
grid_map_loader	Merge remote-tracking branch 'origin/release-22.07'
grid_map_msgs	Merge remote-tracking branch 'origin/release-22.07'
grid_map_octomap	Merge remote-tracking branch 'origin/release-22.07'
grid_map_pcl	Merge remote-tracking branch 'origin/release-22.07'
grid_map_ros	Merge remote-tracking branch 'origin/release-22.07'
grid_map_rviz_plugin	Merge remote-tracking branch 'origin/release-22.07'
grid_map_sdf	Merge remote-tracking branch 'origin/release-22.07'
grid_map_visualization	Merge remote-tracking branch 'origin/release-22.07'
.gitignore	Merge branch 'anymal_research/rsl/fix/grid_map_pcl' into 'master'
LICENSE	Merge branch 'feature/cleanup_licenses' into 'master'
README.md	Merge branch 'anymal_research/rsl/feature/grid_map_sdf' into 'master

源码中的多层栅格数据 C++ | 收起

```cpp
//! Grid map data stored as layers of matrices.
 using Matrix = Eigen::MatrixXf;
 std::unordered_map<std::string, Matrix> data_;  //二维矩阵存储float类型数据

 //create  grid map layer,  10m*10m的栅格 对应的 size = 10m/0.1m, 10m/0.1m
for (auto& layer : layers_) {
   data_.insert(std::pair<std::string, Matrix>(layer, Matrix()));
 }
 for (auto& data : data_) {
   data.second.resize(size_(0), size_(1));
 }

```

将float类型数据改造成一个结构体

C++ | 收起

```cpp
struct DataInfo {
    EIGEN_MAKE_ALIGNED_OPERATOR_NEW
    float data;
    Eigen::Vector3f real_point;
}

using Matrix = std::vector<std::vector<DataInfo>>;
 std::unordered_map<std::string, Matrix> data_;  //二维矩阵存储结构体类型数据

  //create  grid map layer,  10m*10m的栅格 对应的 size = 10m/0.1m, 10m/0.1m
 for (auto& layer : layers_) {
    data_.insert(std::pair<std::string, Matrix>(layer, Matrix()));
  }
  for (auto& data : data_) {
    data.second.resize(size_(0));
    for (auto i : size_(0)) {
        data.second[i].resize(size_(1));
    }
  }
```

优点： 1、可以根据业务需求，设计出符合要求的栅格地图

2、栅格的更新保持不变，CPU部分增长很少

3、在栅格地图上的应用算法部分维持不变

缺点：1、栅格地图所占内存会增加

2、需要对源码进行一定程度的修改, 需要修改set\get\clear\reset等接口，改动量较大

结论：

方案二、每一个图层再多增加对应的两个图层

3、融合精度提升

3.1、单层栅格地图数据源融合

栅格地图太大，无法对每个cell进行缓存多个历史数据帧进行融合，或遍历周围数据进行均值滤波等操作，空间和时间复杂度太高。

将采用栅格地图cell中的值+实时观测值做加权，来提供障碍物位置的稳定性。

1、栅格cell被击中过，即占用率occ > 0.0

根据占用栅格的占用概率和观测值点的置信度计算加权值。

$$x_f = x + k * (z - x)$$

$$k = confidence/(occ + confidence)$$

对于无法提供置信度的K:

$$k = 1.0 - occ$$

2、栅格cell未被击中过，或者占用率occ < 0.0，将观测数据赋值过去，不做融合。

$$x_f = z$$

3.1、多层栅格地图多传感器数据源融合

不同传感器观测的静态障碍物在栅格中形成不同的层。视觉FS障碍物为prob层，超声波组点障碍物为ultra_point层，激光雷达为lidar层。

在栅格地图中所有传感器观测到的障碍物位置cell都是同一个，如果同位置cell的障碍物被多个传感器观测到，就可以考虑从不同层同位置cell中的数据进行融合。

同一个cell位置三种传感器都有击中：

$$x = k1 * x_p + k2 * x_u + k3 * x_l$$

$$k1 + k2 + k3 = 1.0$$

$$K1 < k2 < k3$$

K1 K2 K3可以根据传感器精度来设置，。

同一个cell位置有两种传感器有击中：（比如视觉和超声波）

$$x = k1 * x_p + k2 * x_u$$

$$k1 + k2 = 1.0$$

同一个cell位置只有一种传感器击中：（比如视觉）

$$x = x_p$$

收益：障碍物融合后，稳定性更好。同时输出障碍物个数会降低。

不足：无法确定到栅格障碍物传感器来源，问题分析较难。