解决的什么问题，怎么解决的，重点在哪，创新点在哪，一定要说清楚

奖项：2019.4 第八届上海交通大学“思源杯”交通科技大赛二等奖

参赛项目：基于路侧激光雷达阵列的无人驾驶车辆定位系统开发

项目做了啥：

个人贡献：双激光雷达标定。利用一个圆盘，两个激光雷达打在上面，分别提取两个激光雷达坐标系下面圆盘的圆心，则找到一组对应点。移动圆盘多次，得到多组对应点，利用SVD奇异值分解，得到旋转平移矩阵。

2019.10 “以创新.致未来美的开放式创新大赛2019上海站”特等奖

参赛项目：DeepWind厨房空调

个人贡献：参与调试

2020.4-2020.11

安徽海博智能科技无人矿车研发

团队：实验室团队负责采矿平台的无人驾驶以及仿真平台搭建。采矿车分为三个阶段：卸料口到采矿平台、采矿平台的规划控制、采矿平台到卸料口。本团队主要负责采矿平台的规划控制，当无人矿卡寻迹到采矿平台的时候。

**个人贡献**：

融合融合激光雷达和组合导航数据， 实现了特征稀少的矿区点云地图构建， 并生成了 opendrive 格式的高精度地图用于导航。最开始考虑到矿区特征稀少，所以直接将组合导航数据作为激光雷达的位姿估计，将所有点云进行融合，得到点云地图。不过

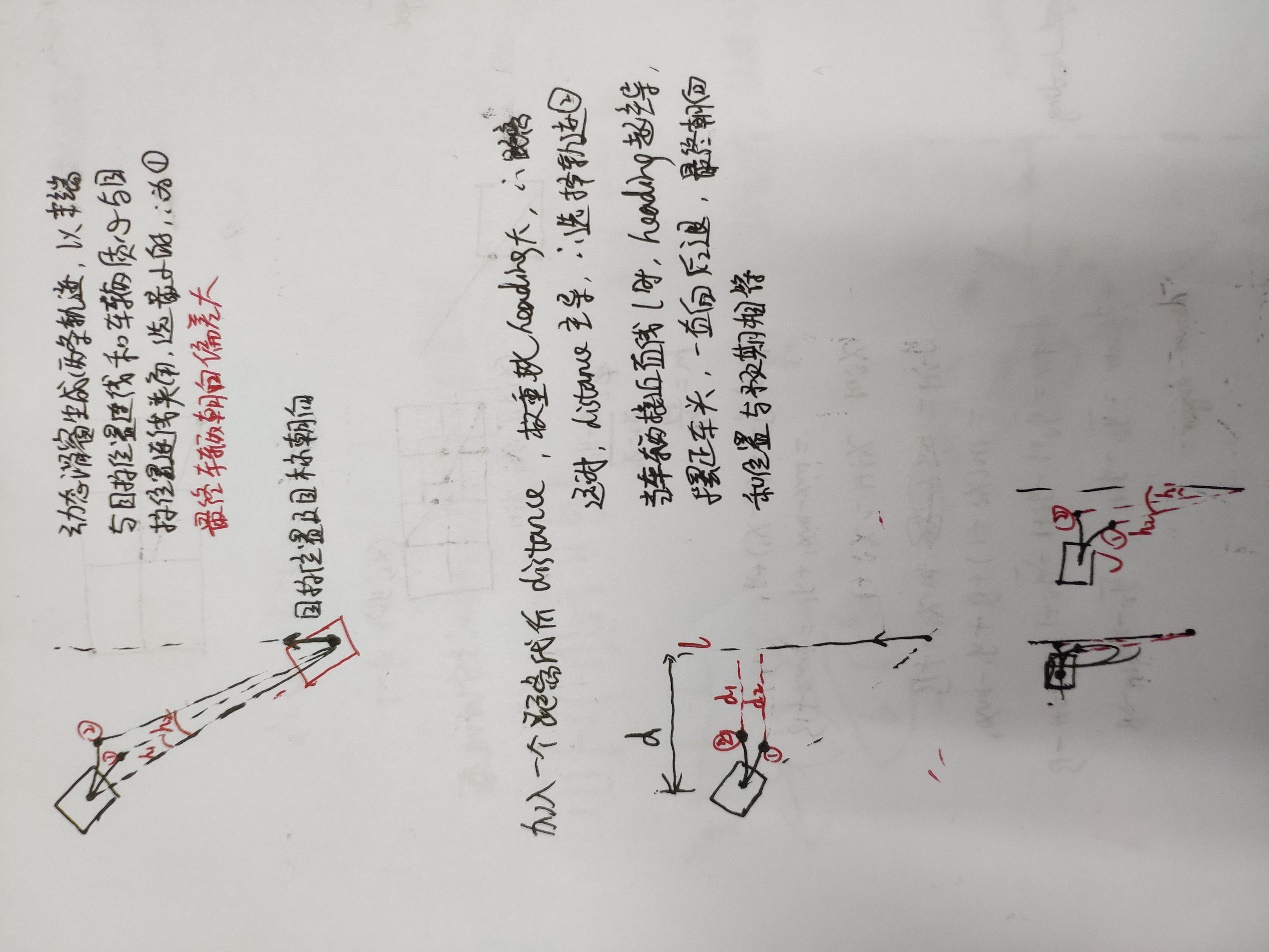
利用激光雷达建立栅格地图，供dwa路径规划使用。这里使用的是高程差算法建图，可以避免在地面不平以及车辆抖动较大的情况下拟合路面，地面点去除更加干净。因为矿区车辆振动频率较高，10Hz扫描频率的激光雷达在一圈内可能都会抖出波浪形，导致用RANSAC拟合平面误差较大，有很多地面点不能去除，如果阈值调高又会将低矮的障碍物滤除

（中间遇到的问题就是矿卡运行时粉尘很大，导致激光雷达误识别，开始想利用粉尘变化很快，利用聚类和ICP匹配来实现灰尘剔除，如果相邻帧相邻类的物体匹配误差大，则表示为灰尘，后来发现不行，因为短时间里面灰尘变化也没那么大，如果间隔几帧检测延时又太大，最后融合了毫米波雷达数据，将粉尘的干扰去除，具体做法是首先将两种传感器标定到车辆坐标系，然后对激光雷达去除地面点，聚类，如果类中检测到毫米波雷达目标，则表示此类有真实障碍物，因为灰尘不会被毫米波检测到）。

利用激光雷达和组合导航写了一个简易的RTK-SLAM，将激光雷达数据注册到惯导坐标系下，生成高精度地图。

改进了DWA局部路径规划实现矿车在装料平台的路径规划，并能精准地将矿车倒至指定位置和指定角度（误差5°以内），解决了传统DWA算法无法精准控制车辆在终点位置的朝向问题。因为传统的DWA算法是用在可以原地旋转的小车，所以对终点朝向的要求不高，而我们的乘用车不能原地旋转，所以需要解决这个问题。

论文里面给的DWA算法只有heading代价和与障碍物的距离代价，但是我们这里考虑场地没有障碍物，所以不进行避障，所以与障碍物的距离代价写了，但没用。但是我们发现heading代价最终只能让车辆到达那个指定位置，但是朝向偏差大。最后我们就想，能不能让heading和最终的朝向一开始就一样呢？所以就加了一个车辆与目标位置朝向的直线的距离。即让这个距离权重大一些，在开始距离较远时，将车辆拉到这个直线附近，然后此时这个距离较近，距离代价较小，heading起作用，则可以让车辆在这条直线附近将车头摆正，最后停到指定位置。



2019.10：参加苏州智能驾驶技术创新大赛，获得环卫车组应用创新奖。

自己主要负责车辆控制和反馈CAN消息的编码和解析，激光雷达利用高程差算法构建栅格地图。我们主要是使用的组合导航进行定位，加上纯跟踪寻迹，以及利用激光雷达识别路沿，保证车辆能够对路沿进行贴边清扫。所以我们需要提前采集路径数据，加上路沿识别进行对车辆定位进行纠正，保证贴边清扫。因为我们采集的路径数据就是靠近路边的，所以也会对路沿的识别进行一定限制，防止误识别，偏离道路太远。

2020.12：

1. 负责伯镭无人驾驶车辆的智能化改造，主要负责车辆控制和反馈CAN信息的编码和解析，基于高程差算法的激光雷达占据栅格地图的生成，RRTStar算法的局部路径规划避障。

利用激光点云数据和高程差算法建立平面占据栅格地图，用于路径规划和AEB。高程差算法可以防止车辆抖动太大，或者地面不平，导致RANSAC拟合地面不准，导致地面点去除不完整，造成刹车。

并且我也根据点云有时会因为阳光反射在地面上出现少量噪声，导致出现误判的情况，改进了高程差算法，只会当超过高程差阈值区域的点云对数量超过一定数目时，才判定此栅格为障碍物占据。具体做法是先一次遍历，找出每个栅格的激光点云的z值最小值，然后二次遍历栅格，记录比z值最小值大height\_diff阈值的点云数量HigherThanMinZNum。最后统计每个栅格的HigherThanMinZNum,如果大于一定数量，则表示此栅格被占据。

3.改进栅格地图构建策略配合基于目标的RRT星路径规划算法实现循迹中的局部路径规划

循迹过程中，如果循迹路径在车前5米以内出现障碍物，则触发AEB，如果5-9米出现障碍物，则以14米处点为重点，进行RRT星路径规划。车辆在循新的路径的时候，如果路径上再次遇到障碍物，则再次触发RRT\*路径规划。RRT星路径规划的速度非常快，在10ms以内，所以可以进行快速的路径规划。RRT算法也能引导算法向终点搜索（随机点有一定概率选到终点）

RRT\*算法过程：

1）首先随机选点，为了能快速搜索到终点，所以给定一定概率选择终点

2）在已经搜索得到的树结构中为随机点找到一个最近点，得到连线，并在连线上离最近点step\_size处得到新的节点，然后判断两个点连线在栅格地图中是否有碰撞（**Bresenham**算法），如没有碰撞，则生成新的树枝节点newNode。

2）为newNode重新选择父节点：找到newNode周围radius范围的圆内的所有点，计算以他们为父节点后的路径代价，选择一个代价更小的父节点，这样可以防止路径太多小的曲折，不光滑

3)重布线随机树：判断newNode周围radius的圆内的点是否能以newNode作为父节点，减少各个点的路径代价

4）结束规划：当newNode与goal的距离小于step\_size,且连线无障碍物时，则路径规划完成。

***遇到的问题***：传统的栅格地图会多次触发RRT星的重规划：

改进栅格地图：在进行路径规划时，为了不考虑规划路径的车辆模型，首先将栅格地图以车辆的尺寸进行膨胀，则可以将车辆的路径规划当成点来进行。为了解决障碍物稍微运动，就会重新触发RRT\*算法的弊端。我在原来膨胀的基础上再膨胀一圈，作为路径规划的边界。

首先有一个inscribed的小圈和一个planning的大圈。触发RRT\*路径规划的条件是循迹或者已经规划好的路径点连线在inscribed圈内，然后路径规划时节点是否有障碍物的判断是首先以planning圈为标准，则可以将规划路径远离障碍物，障碍物少量运动不会重复触发RRT\*算法，如果规划不了路径，则再以inscribed圈为标准进行规划。

***后续改进：***

针对路径规划时候激光雷达检测路沿的距离较近，RRT\*规划可能超出路沿，我们可以使用路沿的高精度点云地图制作栅格地图来作为规划限制

用了哪些算法和数据结构：树，kd树来搜索

2021华为软件精英挑战赛 上合赛区32强二等奖

题目大概意思是我们是公有云提供商，有很多类型服务器（不同的core和memory，价格，每天能耗）可以购买，给用户提供不同类型的虚拟机服务。而且虚拟机还分单双节点。我们如何在用户虚拟机请求给定的情况下，购买充足服务器满足用户所有需求，但是成本最低。这个成本包含服务器购买成本和服务器运行能耗成本。并且每天还有一定次数可以将部署好的虚拟机迁移到其他服务器上去，以便能将服务器尽可能满，空出一些服务器，减少能耗成本。所以本题的关键点在于服务器的购买，虚拟机在服务器上的部署，虚拟机的迁移。

服务器的选择：

在这个比赛中，服务器的选择是比较重要的。我们首先对服务器数据进行了分析，发现服务器的购买价格以及每天的能耗都是core和memory的二元函数，即购买价格和core、memory基本处于一个平面上，所以我们进行了最小二乘拟合，得到表达式a1\*core+b1\*memory=serverCost, a2\*core+b2\*memory=powerCost。而我们讨论中想到当服务器用的天数多的时候，每天能耗的价格占主导因素，而服务器用的天数少的时候，服务器的购买价格占主导因素。我们开始设定了一个剩余天数的阈值，当小于250天的时候，利用serverCost选服务器，否则用powerCost选阈值，发现少了几千万，证明是有效果的。但是参数越少越好，最好弄成自适应的。所以我们用拟合得到的表达式T\_days-(a1+b1)/(a2+b2)作为分界线，之前的用powerCost，之后的用serverCost。

虚拟机的分配：

这其实可以看成一个背包问题，或者装箱问题。每个服务器可以看成一个背包，最大容量就是最大的core和memory，如果能够将每个背包尽可能装满，则能耗肯定最低，因为就可以有空闲的服务器不增加能耗。所以我们写了01背包的动态规划代码，这里有两个限定条件，所以状态为3个，core、memory、服务器数量。最简答的01背包的只有两个状态量，dp[i][j]=max(dp[i-1][j],dp[i-1][j-w[i-1]]+val[i-1])表示前i个物品在背包容量为j的情况下能装的最大价值。而这里dp[i][v][u]=max(dp[i-1][v][u],dp[i-1][v-core[i-1]][u-memory[i-1]]+core[i-]+memory[i-1])表示前i个虚拟机在服务器core容量为v，memory容量为u的情况下能装下的最大的core和memory和。我们对三维状态量优化成了两维（威少补充：咋做的）

但是效果不太好，因为虚拟机的请求是一定的，则肯定是前面的背包装得满，后面就没法装得满（威少再补充一下原因），并且这种方法还很耗时，所以最终我们摒弃了这种算法。转而使用比较朴素的first fit，best fit这类方法。最开始我们使用best fit算法耗时太长，但是后面我们更改了服务器的存储结构，将其用set存储，排序方式按照资源剩余率排序，剩的少的在前面，则虚拟机部署时从前往后找到的第一个服务器就是剩余容量差不多能装上的。不过set其实设计初衷就不能直接修改其中的值，只有删除和插入节点才会触发重排，并且只重排插入删除节点相关的节点，这样才能达到logn的时间复杂度。最终我们对于需要修改的服务器先将存在set中的指针删除，然后更改容量，然后再插入。

虚拟机的迁移：

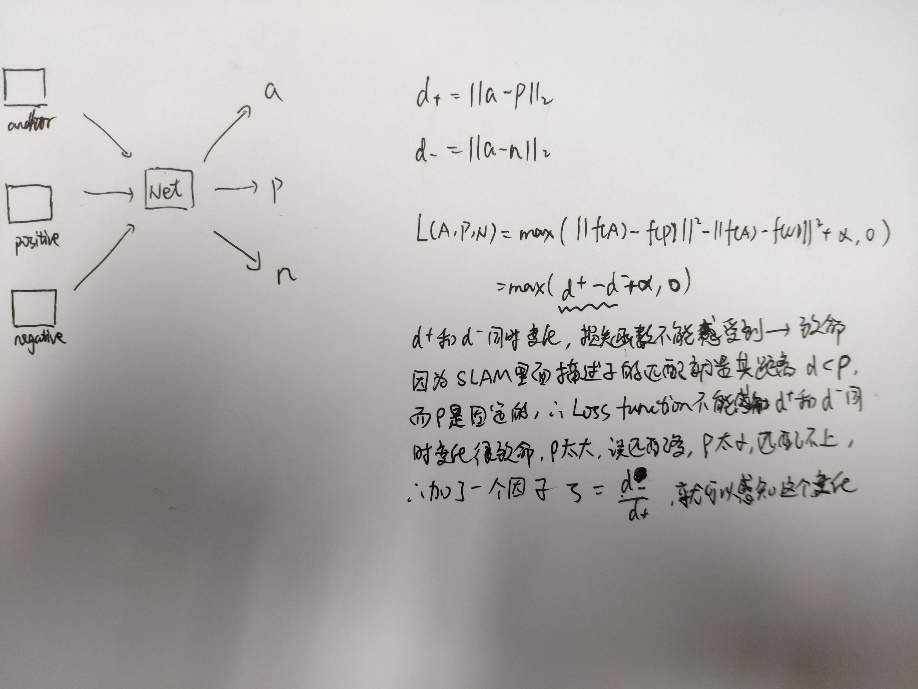
我主要负责迁移部分代码的编写，后面对于服务器的购买优化我也做了一些工作，提升了好几个名次。迁移的目的就是将每个服务器尽可能装满，然后空出一些服务器，这样就可以减少能耗成本。最开始我们是将所有的服务器按照部署的虚拟机的数量排序，多的在前，然后将后面服务器上的虚拟机往前面迁移。但是因为O(n^2)的复杂度，所以速度比较慢。后来我们在迁移之前将剩余率<0.04的服务器排除，这样虽然还是O(n^2)的复杂度，但是时间常数可以降低不少。后面我们又发现可以将当天需要删除的虚拟机先迁移到后面，将当天不删除的迁移到前面，这样每天之后删除的vm部署的服务器就很可能空出来，这样又提升了一些分数。

最终我们初赛22名，复赛12名，夺得上合赛区32强二等奖。

2020.09：二进制描述子

自己利用triplet loss得到深度学习描述子，并进行二进制化，得到了精度和效率较高的描述子，并利用ORB-SLAM2框架，将自己设计的描述子应用到SLAM系统中，建图定位精度提高。

难点：损失函数设计。深度学习描述符的设计其实已经有很多了，但是不适用于SLAM系统，因为他们没有考虑到描述子的尺度问题。现在描述子生成一般使用三元组损失triplet loss, 而里面有一个正样本和anchor样本描述子距离与负样本和anchor样本描述子距离的差，所以两者同时变化的时候，损失不变，这对于SLAM系统来讲比较致命。因为我们需要对两张图的描述子进行匹配，所以需要计算相似描述子的距离，距离小于一定阈值则属于同一个描述子。比如这里我们要将anchor样本和正样本的描述子聚成一类，则根据他们两者的距离来进行判断。而因为尺度问题，损失函数里面是没法确定这个距离的大小的。所以这个阈值就没办法很好的确定。因此必须要让损失函数观测到这个变化，于是我在损失函数里面加了一个尺度因子。这样当两者距离变化时，损失函数能观测到这个变化。



评价指标：

简历中自己还需要好好理解的项目：ORB-SLAM2，LOAM、cartographer

图优化和滤波器SLAM优缺点：滤波器SLAM只能估计xt，如果前面估计错误，就无法纠正；而图优化估计的是x0:t,整条路径的位姿，后面可以利用回环检测对于前面估计的位姿进行修正。

图优化原理：非线性最小二乘。节点表示位姿，边表示两个位姿空间约束，形成环之后就产生了误差。找到各节点的位姿最优值，让里程计预测与观测的误差最小(x0:t)

cartographer：