**松灵bunker底盘搭建 进度**

已经完成：

（1）完成全局地图的构建，包括栅格地图和拓扑地图

（2）完成小车的定位功能，使用eskf算法融合IMU和RTK数据，获 得小车的位姿信息

（3）完成目标点的轨迹规划任务

（4）添加小车的运动控制模型

（5）在gazebo中仿真小车的导航

（6）在gazebo中和相机结合起来，实现停障的功能

（7）各种传感器的驱动调试

后续安排：

1. 跑数据集，实际测试
2. YOLO的道路分割模块和手持信标的跟踪实现
3. 最开始转弯的时候可能会有角度的突变
4. 现在用的是单位速度方向定初始转向角度:能否使用eskf中的imu初始位姿来确定这个角度，以此减小误差？或者让车先沿当前朝向运动一段距离，来确定初始的转角？（这个方法比较危险）
5. 2.录一个只有imu，gps的数据集，遥控跑一跑各种转弯直线的情况，然后分析imu的位姿定位情况

（6）现在小车行驶道路宽度的要求：4脚的宽度，目前最大是6脚的宽度（一脚的宽度大概是30cm）

**导航模块**

（1）全局地图---->拓扑＋栅格地图实现

（2）自身定位---->卡尔曼滤波算法

（3）路径规划

A.全局路径规划---->Dij算法计算

B.局部路径规划---->障碍物，避障的功能（暂未考虑，只做了 停障功能）

（4）运动控制---->将cmd\_vel转换成电机转动的控制指令，实现实际 场景下的小车运动

（5）环境感知---->主要是传感器的使用，结合全局地图，局部地图， 自身定位，运动控制各个模块。

**主要使用的传感器**

RTK传感器---->获取当前时刻小车位置的经纬度坐标

IMU六轴---->获取当前时刻小车3个方向的线加速度和角加速度，用于估计小车的姿态信息

Gemini 335L相机---->识别物体，实现停障功能，道路分割功能

**拓扑地图和栅格地图构建（全局地图）**

（1）在一个数据集地图中利用RTK采集数十个点的经纬度坐标信息，利用坐标转换的计算公式，在东北天坐标系下，以第一个点为原点建立直角坐标系，可以得到这些点的（x，y）。

（2）用有宽度的边将点与点连接，表示可行驶区域，且每个点都会有一个领域半径，来表示可达区域。

（3）栅格地图在拓扑地图的基础上，将地图的外环进行一定的扩展，更好模拟周围的环境。

Coordinate\_converter.h

Topology\_map.h

Grid\_map.h

Coordinate\_converter.cpp

Topology\_map.cpp

Grid\_map.cpp <---path planning / available node

读取yaml文件中的点的经纬度坐标和边的关系还有宽度等信息--->坐标转换的函数--->生成全局的拓扑地图--->dij算法找到节点直接的最短路线--->可视化拓扑地图

栅格地图头文件包含拓扑地图的头文件，可以调用拓扑地图的有关函数，有yaml文件，读取分辨率，extent等参数，在拓扑地图的基础上，构建一张栅格地图。

**路径规划**

（1）先判断初始点和目标点是否在可行驶的范围之内，同时满足才进行下一个步骤

（2）目前小车的起始点设置为（0,0），终点可以自己指定。对于终点的位置，分两步进行判断，第一种是在拓扑点的领域内，那么找到这个拓扑点，用dij算法构建最短的路线，先拓扑点到拓扑点，在到终点（避免回头问题）。第二种是在可行驶的边上，首先这个点到边中线的距离一定小于等于边一半的宽度，且这个点的投影一定在两个拓扑点的连线上。找到这个拓扑点，然后用dij算法得到拓扑点到拓扑点的最短距离，再加上最后一段的距离长度。

navigation.h

Navigation.cpp

类

公共

更新当前的位置

更新目标的位置

导航处理

计算路径

运动到目标位置

目标点回调

实时位置回调

速度回调

路径可视化

更新频率

Ros 定时器

私有

一些参数

当前的位置（x，y）

当前的速度x方向，y方向

更新频率

目标的位置（x，y）

主函数

**自身定位**

机器人导航中的定位，利用里程计定位和传感器结合的方式实现。使用开源的卡尔曼滤波算法，基于IMU（里程计-->累计误差）和RTK（传感器-->信号干扰误差），实时更新小车的位姿信息。

**在gazebo中仿真运动（运动控制）**

仿真环境中使用gazebo的odom话题消息代替实际的ekf\_lat\_lon的话题消息。

栅格地图和拓扑地图的根坐标系都是global，机器人模型的根坐标系是base\_footprint（避免出现模型有一半在平面以下的情况），odom里程计的根坐标系是odom。所以要添加坐标变换，将三个坐标系都统一到global中。

在gazebo中已经自动给出了odom到base\_footprint的tf变换，需要在publishRobotPose函数中添加一个odom到global的动态坐标变换，（这里不能使用静态坐标变换）

自由搭一个小车的模型，添加所有必要的设置。

在rviz中发布终点信息，用点的格式发布，然后小车可以实现从当前点沿路径规划的轨迹运动到终点。

或者直接rostopic pub clickpoint xy的坐标。

目前的运动控制使用的PID控制，只用到了比例，没有用积分和微分去修正误差。

**结合YOLO图像完成停障的功能**

启动YOLO文件的时候，要给一个访问权限，在scripts目录下输入命令chmod +x py文件名字

目前的框架是，YOLO接收到相机的topic，包括深度信息，图像信息，和相机的内参，三个消息必须都有，缺一不可。然后在YOLO训练的模型下，识别图像中的物体是否为障碍物，如果是则心跳包发送1为真，不是则心跳包发送0为假。

模拟1----给cmd\_vel进行测试

通过订阅heartbeat的topic和初始的导航的cmd\_vel，发布新的cmd\_vel\_safe来实现小车运动控制的停止与运动。注意在导航的头文件中，订阅的cmd\_vel要改成cmd\_vel\_safe，并且gazebo的move.xacro的topic也要改成cmd\_vel\_safe。

模拟2----给clickpoint进行测试（重点）

通过订阅heartbeat的topic和终点（x，y），发布新的cmd\_vel\_safe来实现小车的运动控制的停止与运动。此时只用将gazebo的move.xacro的topic改成cmd\_vel\_safe。

在rviz中订阅BoundingBoxes的topic可以显示出障碍物的模型和信息，需要在gazebo模型中添加一个link表示相机自身的坐标系，因为相机自身的坐标系与ros的坐标系不一致。

可以在gazebo中使用一个空世界，行驶的时候在前面放上障碍物，模拟真实的现实情况。创建新的world时，需要sudo gazebo启动gazebo不然保存不了环境。

**运动控制**使用的小车模型是两轮差速模型，目前采用的是移动和旋转结合控制。如果将移动和旋转分开控制，由于小车要实时校准与目标点的角度偏差，很可能出现走一点，停下来，转一下的情况，整个运动不流畅。将移动和旋转结合起来，可能的问题是转弯的半径大，小车会走出安全区域。但我们在栅格地图构建的时候考虑到了地图边界的问题。在栅格地图中添加一个extent=5m作为地图边界的拓展，目前地图的可行驶区域，即终点可选择的区域仍然是3m宽度的道路（实际大小要根据实际的场景进行调整）但有一个5m的拓展之后，小车在3m的边缘转弯的时候就不会离开安全区域（安全区域=3m的可选择终点区域+所有边向外拓展5m的拓展区域）

（5m的拓展区域是安全的，只是我无法选择其中的点作为终点发布）

目前的设想是如果在实际行驶的环境中，总道路的宽度是5m，那么我的width基本就设定是3m，extent设计为1m，总的就是5m，但我可选择的终点范围只在3m内，这样留一个2m的拓展，防止转弯的时候离开安全区域。

**实际上车的测试**

输入各种传感器的消息，输出cmd\_vel给电机，电机控制小车运动

IMU 驱动文件

R70驱动文件

Gemini 335L 相机 驱动文件

Agilex松灵bunker底盘电机驱动文件

Bunker底盘注意事项

车的电量，显示屏上要≥48v，不然得充电

遥控器的电量，4节电池，初始状态下abcd都是最上方。然后长按2个开关键开机。随后拨动b到中间（遥控控制模式），就能遥控控制，左边是前进后退，右边是旋转。

用can接口开发时，b要在最上方，为指令控制模式。

急停开关，正常都是释放弹起状态

不能直接手去推车

充电时要关机充电

**底盘说明**

bunker和scout的使用基本一致

(1)下载对应的sdk文件 github

(2)连接can口，此时遥控器的b要拨到最上面。

$ sudo modprobe gs\_usb

$ sudo ip link set can0 up type can bitrate 500000 //默认都是500000，输完这条指令，正常can 上

//上面会亮起3个红灯，其中的rx是交替闪烁（scout），bunker一样

//注意接线，scout上面蓝线是canl，黄线是canh

//bunker 也是一样的接线,如果3个红灯长亮，则是异常，bunker建议把车关机重启来解决。

//所以不太建议使用逆变器，而用一个独立的电池给工控充电。

$ ifconfig -a

$ sudo apt install can-utils

$ candump can0 //输入完这条指令，正常能看到输出的数据，就没有问题了

(3)launch启动底盘驱动的文件，成功启动之后，会有日至输出的提示。

scout是这样的

[INFO] [1755689626.598352724]: Working as scout mini: 1

[INFO] [1755689626.598907542]: Working as scout omni: 0

Start listening to port: can0

Detected protocol: AGX\_V2

Start listening to port: can0

[INFO] [1755689626.658551651]: Using CAN bus to talk with the robot

terminate called after throwing an instance of 'std::system\_error'

what(): Resource deadlock avoided

bunker的launch文件要做修改，将bunker\_robot\_base.launch中的mini那行改成true

将bunker\_base.launch中的remap后面的smooth改成/cmd\_vel这样就不用改话题了，就是使用默认的cmd\_vel

(4)启动导航的文件包，3个launch文件即可，rostopic list查看所有的消息

(5)rostpic pub -r 10 /cmd\_vel 发布速度消息，小车能动，就代表测试成功。

**imu说明**

(1)sudo chmod 666 /dev/ttyUSB0,给usb口访问的权限,为了避免与r60冲突，改launch文件为usb1

1. roslaunch yesense\_imu run\_without\_rviz.launch 启动这个节点，静态坐标变换已经添加，具体数值可根据实际情况调整

**335L相机说明(可以参照以前写的那版说明)**

(1)dmesg | grep -i usb，查看相机的端口号，在multi的那个launch文件中修改

(2)同样是在这个文件中，修改num数量为要用的数量，并添加静态坐标变换，具体数值根据装车的位置决定

(3) roslaunch orbbec\_camera multi\_camera.launch 启动文件

(4)在导航文件对应的yolo文件启动前要先进入scripts文件目录，然后$ chmod +x (py文件名)yolo\_3d\_bbox.py yolo\_obstacle.py yolo\_v8.py

xiaomi1@xiaomi1-XM22AL5S:~/navi\_eskf/src/yolov8\_ros/src/Yolov8\_ros/yolov8\_ros/scripts$ chmod +x yolo\_3d\_bbox.py

xiaomi1@xiaomi1-XM22AL5S:~/navi\_eskf/src/yolov8\_ros/src/Yolov8\_ros/yolov8\_ros/scripts$ chmod +x yolo\_obstacle.py

(5)注意修改对应的py文件的相机的topic名称

**R60说明**

(1)接一个充电线，用电池充电，接一个com口的线，要有转接口，接到工控。

(2)cutecom指令，查看usb口，默认是0口

(3)使用命令从串口读取nmea格式的GPS字符段并发布话题/nmea\_sentence（/nmea\_msgs/Sentence格式）：

rosrun nmea\_navsat\_driver nmea\_topic\_serial\_reader \_port:=/dev/ttyUSB0 \_baud:=115200 //报警告不要紧

使用nmea\_navsat\_driver驱动中的节点接收话题/nmea\_sentence（/nmea\_msgs/Sentence格式）并发布/fix话题（/NavSatFix格式）：

rosrun nmea\_navsat\_driver nmea\_topic\_driver //报警告不要紧。

(4)然后rostopic echo /fix 到室外查看有没有数据 或者rostopic echo /nmea\_sentence 正常到室外就有数据了

**导航模块**

最后启动--->共2个launch文件

在模拟环境中，使用的是两轮小车，而实际的环境中是履带式小车。受履带式小车的物理特性的影响，小车运动控制的约束距离，约束角度等参数要适当增大，还有速度方向单位化的判断条件也要适当放宽松。避免实际环境中出现小车一直原地打转的情况。

**小车运动轨迹的设计（旧）**

对于1中每个标定的地图点，都会有个小领域，这个领域的半径r=1.5m，用于判定车是否在这个领域内。

对于车行使过程中的偏移角度，可以设定一个偏差的范围-5～5的角度范围。

(1)小车的初始位置到最近的标定点的运动

先判断小车的初始位置是否在某个地图点的领域中，如果不在任何一个点的领域中，则返回消息，小车位置不合法。如果在，则找到最近的那个地图点。然后给小车给一个微小的运动，v=0.1m/s，t=0.5s，这样就能知道小车的初始速度方向，接着调整速度方向，让速度方向与小车当前位置与目标点的连线方向基本重合，这个角度偏差满足在设定的范围内就可以，最后前进到我的目标点。(小车初始位置的姿态不重要)

(2)标定点到离终点最近的标定点的运动

这一环节的运动涉及到多条可选择的线路，可能是2-3-4-5，可能是2-4-7-5，利用A\*算法或Dijkstra算法，计算出最短的行驶路径。当小车行驶到每个标定点的时候，都要对此刻行驶方向与两个标定点连线的角度进行判定，如果这个角度偏差大于设定的偏差范围，则要停止直线运动，先调节行驶方向的角度，然后再运动。如果角度偏差在设定的偏差范围内，则继续运动到下个点的位置。

(3)从标定点到终点的运动

与(1)基本一致