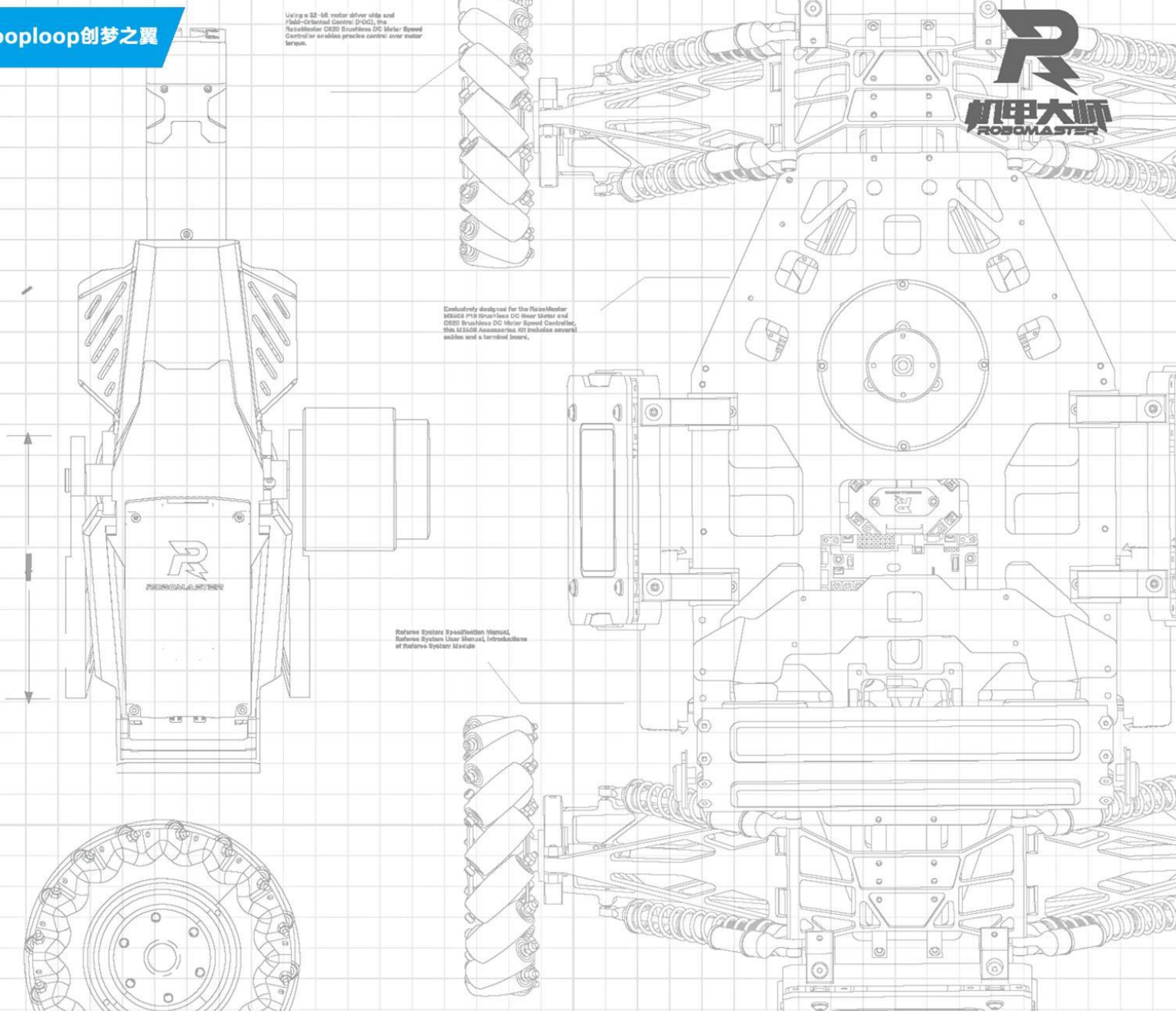
Nooploop创梦之翼



第二十一届全国大学生机器人大赛

ROBOMASTER 2022

人工智能挑战赛

技术方案

目录

内容提要

RoboMaster RMUA 需要参赛队综合运用机械、电控和算法等技术知识，自主研发全自动射击机器人参赛，对综合技术能力要求极高。

本文简述了我队本赛季的总体技术方案。目前部分功能已经实现和通过测试，其余功能也已经处于开发阶段。阅读本文后能够了解我队技术思路以及功能的具体实现方法。

1. 硬件部分

与对抗赛类似，整车采用云台底盘双控制板，主控板云台部分使用 RoboMaster C 型开发板。底盘板需要多个串口（裁判系统通信，调控数据接收，传感器数据接收），拟选取 RoboMaster A 型开发板进行控制。双板之间通过 CAN 总线进行通信。

1.1 传感器

机器人利用传感器进行定位。在场地地图已知的情况下，可以实现路径规划、决策控制以及其他功能。

1.1.1 传感器选型

传感器	型号
激光雷达	RPLIDAR A1（后续考虑升级 A3）
超声波测距模块	电应普超声波模组一代
相机	迈德威视 SUA133GC

1.1.2 传感器选型理由

机器人主要依赖激光雷达进行场地实时定位。RPLIDAR A1 激光雷达可以进行 360°扫描测距，测量距离为 12 米超过场地长度，足够机器人在场上的定位所需，而且该激光雷达性价比较高，故选用该激光雷达。由于安装处有部分遮挡，实际上测量的角度为 10°~350°。

因其使用光磁融合测距，而比赛场上存在透明障碍物，对光的反射性降低，使得激光雷达产生测距误差。为了避免测距失灵，拟选用电应普超声波传感器进行补偿。该超声波模组能够检测透明障碍物，最大测量距离为 450cm，盲区为 3cm，测量角度为 60° 。该传感器一方面能够弥补激光雷达的死区，另一方面能够与激光雷达进行数据融合。为了提高联合定位精度，将超声波模组安装在车体的八个方位角。

反馈控制能提高机器人的运动精度。主控板读取陀螺仪、加速度计、码盘等传感器数据后，利用算法获得四轮的转角、解算机器人的姿态，反馈给控制器。

根据赛事规定和场地条件，机器人可以利用场中视觉标签进行定位，拟选用迈德威视 SUA133GC 相机捕捉图像。该相机最大分辨率可达 1280X1204，综合考虑场地大小和图像清晰度，能够满足自瞄和识别的需求。

1.2 计算设备

1.2.1 计算设备选型

机器人搭载的处理器为 Intel 寒霜峡谷 NUC10 i5 FNH，配备 8G 运行内存。

哨岗电脑为自行组装，CPU 为 Intel i7 11700K，配备 32G 运行内存，高速 USB3.0 接口，显卡选择了 NVIDIA GeForce RTX 3070 Ti，8G 显存。

1.2.2 计算设备选型理由

机器人计算端需要运行多个节点，如自瞄节点、运动规划节点（负责路径规划与导航）、通讯节点、决策节点，其中自瞄节点还要求高帧率。

若自瞄系统以 640*480 的分辨率读取 uint_8 类型图像，保持 200FPS 运行，则每秒仅原图像的读入量就达到了

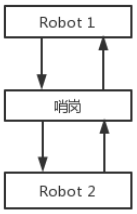
$$640 \times 480 \div 1024 \div 1024 \times 200 \approx 67.38\text{Mb}$$

运动规划节点经过测试，所需运行内存平均在 80Mb 上下浮动。虽然以上两个数字看起来占用内存并不大，但是由于各个节点均需保证高频率运行，如导航节点的高频率（100Hz）向下位机发送底盘运动控制信息、自瞄节点的高帧率（大于 200FPS）识别，此型电脑虽然运行内存上能够胜任计算任务，不过客观来说，1.6GHz 主频还是显得不够高，今后可能考虑更换 NUC8i7BEH，主频高达 2.7GHz，能带来更优秀的算力。

哨岗需要运行两个使用 TensorRT 量化的，GPU 加速的高帧率，高分辨率的目标检测网络。目前虽然没有测出确切的占用内存的数值，但是已知在当前配置下未经 TensorRT 加速的模型推理时 CPU 使用率可达 60%以上，足以证明此配置能够支持所需算力。故所选用的配置显存、运存都较大，CPU、GPU 较好。

1.3 通信链路

两车之间不直接进行通讯，信息均上传哨岗、分析综合后广播。如图所示，两车上传各自的位置信息，哨岗识别、综合上传信息后，广播场上的所有机器人坐标。



这样设计的好处在于所有车进入潜伏状态、交换敌我颜色之后，两车仍然能够选取所广播的四个坐标中最接近自己的坐标，从而得到另外两个敌方机器人的位置，与敌我颜色是否发生交换无关。

预计的广播频率为 30Hz~50Hz 之间（目标检测网络帧率在 30FPS 左右），所有数据包经过编码、截取，大小都在 1Kb 以下。经过测试，现有条件完全能够满足需求，平均在 40Hz 左右。使用的通讯协议是 TCP/IP 协议。比赛要求信息传输具有一定的稳定性，该协议能够保证数据的稳定、保真传输，并且速度较快。

2. 软件部分

系统硬件搭建了机器人端和哨岗端信号和数据流动的通道，软件部分则负责将信号归类、处理，得到有意义的信息，用于规划和控制。

软件主要分为：定位部分、自瞄部分、路径规划与导航部分、决策部分。定位中使用的蒙特卡洛定位算法、视觉标签识别来自于现有库函数，其它算法均由我队自主开发。

2.1 机器人端

机器人端部分功能模块部署在 ROS 系统中。

我队的路径规划、运动控制没有采用 ROS 自带的导航功能包，而是选择自主研发。

2.1.1 感知与定位

传感器模块获取环境信息后，定位系统读取数据后利用融合、滤波算法解算出机器人的姿态。拟选用的定位方法如下：

1)障碍物定位：

启动区围挡近距离围绕距离机器人，并且位于拟选用的超声波测距模块的精准测距范围内，故拟基于八个环绕车身的测距模块数据解算出机器人的初始位姿。这种方法的初始化精度能够达到厘米级精度。

2)雷达定位：

激光雷达的信息是机器人位置信息的主要来源。我们拟采用自适应蒙特卡洛定位（AMCL），该方法利用粒子滤波在已建好的二维地图上计算求出机器人的位置，同时能够解决机器人绑架的问题，计算出里程计的累积漂移（/map_frame -> /odom_frame）。在里程计漂移过大、两方处于对峙状态时，机器人将重返启动区进行里程计重置。

3)误差累计消除：

里程计的误差积累是一个缓慢的过程。在其正常运转情况下，里程计的精度远高于其它方法。我们读取四个轮子的编码器数据，然后结合陀螺仪解算机器人的运动。我队已在自制的车上测试，每百厘米里程计误差稳定在 3%，并且不随时间变化。通过引入一个比例系数，我们成功弥补了这一误差。并且经过长程测试，发现没有大于 1%的累积误差。为了进一步避免比赛时可能出现的刮蹭、轻微碰撞以及打滑的情况，在已有避障措施的基础上，我们结合 AMCL 定位返回的累积漂移。即当累计漂移大于一个阈值时，返回启动区通过初始化重置误差。

4)视觉标签定位：

拟采用的视觉标签识别算法来自于密歇根大学开发的 AprilTag 库，我队移植、修改后可用于比赛中视觉标签的识别和解算。该库优化程度高、占用资源少，可以同时检测多个视觉标签。我们创建了和比赛中视觉标签相容的 TagFamily，能够用于特征匹配。比赛中的视觉标签比 Apriltag 少了外层定位框，我队在修

改了检测方法后，已能够适配视觉标签。经过实际测试，计算速度完全能够达到实时的要求。并且在计算结果的基础上取了当前视野内各个标签的加权平均，加大了距离相机近的、角点提取精确的标签解算出的位姿权值，提高了整体精度。

5)位姿指纹定位：

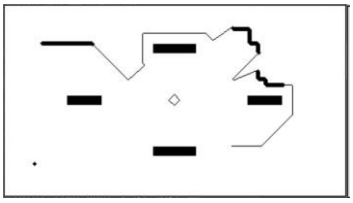
我们还采用结合神经网络的位姿指纹定位算法。将八个超声波测距模块得到的测距信息组成八维向量，称为位姿指纹。备赛阶段将机器人在场地上的每一个位置，每一个姿态反复采集位姿指纹信息，组成详尽、精确的位姿指纹数据库，传入全连接神经网络中进行学习、拟合。将学习得到的模型部署之后，传入实时采集的位姿指纹信息，即可推断出当前的精确位姿，此方法的精确性由海量的预采集位姿指纹保证。

2.1.2 路径规划、避障

为了让机器人在复杂场地中与敌方进行对抗并获胜，需要进行路径规划和避障。

我队的路径规划综合考虑了敌方位置、我方位置、最短路等因素，并且最坏情况下能够在 0.1 秒内规划出一条带有速度信息的运动路径，平均时间在 20ms 左右，比赛时以最大频率运行，以保证规划出的路径具有时效性。我们采用的算法是 A*算法，该算法作为启发式搜索的经典算法之一，能够通过精心设计的启发函数来将各种因素考虑在内，规划出最优路径。缺点是作为 BFS(宽度优先搜索)的变种，虽然有了启发函数的指导，但仍然具有盲目的尝试性，所以我们针对性设计了优化方法：

1)原始全局最优路径生成



在发布终点后，先使用综合了各种因素的估值函数探索规划出一条全局最优路径(如右图，我方机器人从左上角走到右下角，敌方机器人为左下角的黑点，能被打到的地方被加粗标出）。

A*所使用的启发函数 $f(nowx,nowy)$ 为

$$f(nowx,nowy)=Dis(NowPosition,Goal)+k_1*FireCost(State)+k_2\ *\ CooperateBenefit(State)$$

其中，$State$为包括了敌我机器人位置、敌方机器人载弹量与血量的全局状态高维描述向量。$FireCost(State)$ 函数评价了在当前状态下，我方机器人在点 (x,y) 承受的被打击代价。$CooperateBenefit(State)$ 为在该点的协作利益估值函数，目前虽然已经有设计，但暂未融入程序编写，因为其功能与决策树部分的评价函数重合。k_1,k_2为可调的比例系数。

我们认为每一点受击概率$HitCost$符合以敌方机器人(x,y)为中心的二维高斯分布，即

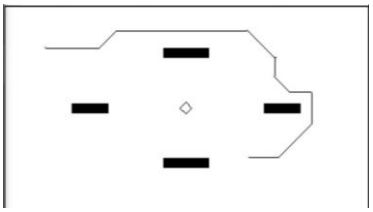
$$HitCost(x,y)=\frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}}e^{-\frac{1}{2(1-\rho^2)}\left[\frac{(x-\mu_1)^2}{\sigma_1^2}-2\rho\frac{x-\mu_1}{\sigma_1}\cdot\frac{y-\mu_2}{\sigma_2}+\frac{(y-\mu_2)^2}{\sigma_2^2}\right]}$$

则被打击代价$FireCost(State)$为：

$$FireCost(State)=k_2\ *\ Threshold_1(BulletLeft)\ *\ Threshold_2(HPLeft)\ *\ HitCost(x,y)$$

其中，两个$Threshold$函数为对剩余血量、子弹的归一化函数。我们认为子弹低于阈值的敌方机器人不具有威胁能力。k_2为可调的比例系数。这样就量化了一个敌方机器人对点(x,y)的威胁程度。

将被打击代价纳入启发函数内，有利于寻找不被打击的路径进行运动。若仅考虑最短路代价，则可能盲目选择通过被敌方火力覆盖的区域，造成不必要的血量损失。



但是不难发现，这条路径由于盲目的尝试、取最优解而走了扭曲的路线，不利于机器人按照这一路线高速运动。所以必须进行二次优化。

2)局部路径优化

我们不难发现一个事实：在敌方打不到我的时候，我可以以最短路算法快速通过当前路径；在敌方能打到我的时候，我更应以最快速度通过当前路径。所以，相邻接敌状态改变的点之间的路径是可以使用最短路优化的。

为了提高程序的模块复用性，使用的算法仍然是 A*算法，不过启发函数是：

$$f(x,y) = dist(x,y,goalx,goaly)$$

其中，$dist(x,y,goalx,goaly)$为：

$$dist(x,y,goalx,goaly) = \sqrt{(goalx-x)^2 + (goaly-y)^2}$$

对这些小段路径进行最短路优化之后得到的路径如右图所示。优化后的路径基本都是直线、易于转弯的折线，非常适合机器人高速通过。如此就实现了对固定障碍块的避障。当然，在编写过程中也使用了内存池等编程技巧优化代码，努力将路径规划这一过程速度提到最快。

但是，仅仅靠上述算法避障是无法满足需求的。

场地的障碍主要分为两种，一是场地上固定的障碍块，二是移动的机器人。下面分别针对两者简述避障策略。

1)对固定障碍块的避障

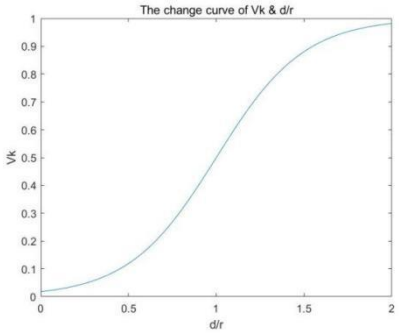
对场地障碍块的避障是通过对预置地图的膨胀腐蚀完成的。其中膨胀腐蚀操作的结构元素要大于等于根号二倍的底盘半径，这样可以避免机器人在转弯的时候刮蹭到障碍块。算法就是上述的 A*改进算法。

2)移动物体的避障

对于移动机器人的避障，仅仅将其当前所在位置设为不可通过区域是不合理的，因为其处于移动状态。我们结合比赛地图的特质，不难发现有很多通道只能容纳一台机器人通过。故我们将比赛地图划分为若干个通道区块，哨岗通过全局的目标检测网络检测到敌我机器人之后，能够解算出其在场地中的实际位置，进而得到他目前是在哪一个通道区块当中，将这些通道称作不可通过通道，并广播给我方机器人，禁止我方机器人通过同一区块。由此即可防止机器人之间的碰撞。

不过想要实现实时避障，最根本的方法还是提高路径规划的频率。考虑到机器人实际移动的速度，如果规划算法能够达到 10 赫兹及以上就基本可以认为是实时。我队并没有采用 ROS 自带的导航包，而是自行开发了基于 A*算法的运动规划，在路径最长的情况下规划时间也仅需 0.1 秒，显然能够达到实时规划的

要求。只要敌方的位置由哨岗广播实时更新，即可实现实时避障。



当然，哨岗也偶尔会有漏检的可能，甚至可能因为漏检而撞上敌方机器人，所以我们在速度规划内设计了根据测距模块返回的障碍距离计算的速度惩罚机制（速度势的另一种体现）。灵感起源于深度学习中的激活函数 sigmoid(x)。将各个由测距模块得到的速度惩罚系数V_k乘以该方向上的原计划速度V_{plan}，最后再进行速度合成，就完成了速度规划层面的避障。右图即为速度惩罚因子随 d/r 的变化曲线。不难发现，当 d>1.5r 的时候，障碍基本对速度没有什么影响了。

不妨设底盘半径为 r，相距前方障碍块距离为 d，则前进速度

$$V_{actual} = V_k * V_{plan}$$

其中速度惩罚因子

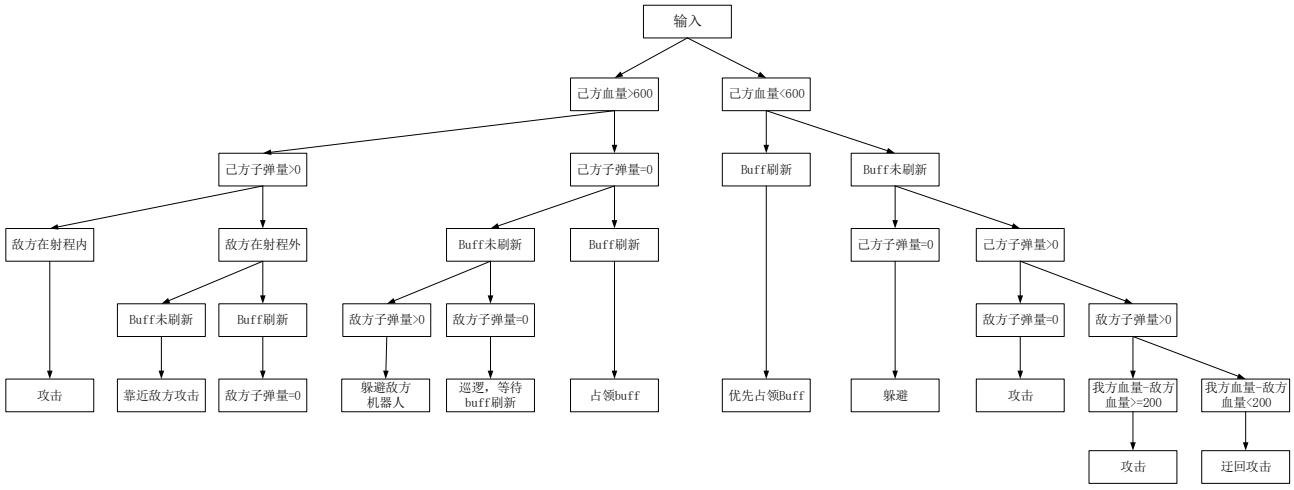
$$V_k = \frac{1}{1 + exp(-4 * (\mathrm{d/r} - 1))}$$

如此则实现的机器人的路径规划、运动避障。

2.1.3 加成惩罚区运动策略

行为树是控制机器人决策流程的分层节点树，在树的范围内，叶子是控制机器人人的实际命令，而形成分支的则是各种类型的效用节点，它们控制机器人沿着树走，以达到最适合这种情况的命令序列。

机器人的决策主要分攻击、靠近敌方、躲避敌方、巡逻/等待 BUFF、迂回攻击。决策的条件主要为己方血量、己方/敌方子弹量、敌方是否在射程外。拟采取决策树结构如图。



2.1.4 云台跟随控制

云台跟踪是为了减少发现目标后的调整角度。实现的方法就是始终将云台指向跟踪敌人的方向，无论其是否出现在视野中。

云台运动分为三个模式：索敌模式、预瞄模式、打击模式。

索敌模式：未锁定装甲板、未锁定预瞄点时的模式，以一定角速度来回搜索视野内敌人。

预瞄模式：根据哨岗广播的敌方坐标，不难算出敌方机器人相对我方机器人的大致角度。经决策树判断后认为需要，就可以令云台锁定在该角度，可以在检测到装甲板之后第一时间进行打击。

打击模式：由自瞄代码搜索目标后发现装甲板，进行位置解算、坐标系变换之后，直接由自瞄代码控制的模式。

默认是索敌模式，有打击目标但是还没有到攻击区域是预瞄模式，进入打击区域、接敌状态下是打击模式。打击模式退出后会进入预瞄模式而不会进入索敌模式，在预瞄模式后停留一段时间之后没有发现敌人才会重新进入索敌模式，以免因为敌方在偶尔躲藏后就直接大幅度摇摆云台搜索敌人，而是停留在预瞄点警戒。所以三个模式是层层递进，可以迅速进入打击状态，但是不会迅速退出打击状态。

2.2 哨岗端

哨岗主要承担两个任务：全局目标检测与定位、向我方机器人发送位置坐标相关信息。具体流程是：接受两车上传的位置信息，与本身目标检测网络得到的位置信息融合之后广播给机器人。

2.2.1 全局感知

哨岗需要对连接到主机上的两个相机同时进行高速实时、高 IOU 的目标检测任务。为了提高响应速度，所部署的目标检测网络必须经过加速，选择的加速工具是 TensorRT，预期单网络帧率能够超过 30FPS。目标检测网络给出检测框之后，认为检测框中心点向下 1/4 边长为机器人的位置中心，由手动选取的场地四个角点进行透视变换，得到机器人的位置中心在场地中的实际位置，与机器人颜色、编号绑定后实时发送给机器人。

不过，由于本赛季潜伏机制的存在，在 135 秒之后，哨岗所发送的机器人位置信息不再与颜色信息绑定，而是直接发送四个位置坐标，各己方机器人选择与自己定位最为相近的位置坐标作为自己的位姿。

2.2.2 敌我识别的方法策略

敌我识别用于锁定装甲板、发射子弹。在本赛季添加潜伏机制之后，优秀的敌我识别是保证胜利的基本条件，否则就会出现自己人打自己人的尴尬情况。

敌我识别主要分两个阶段：潜伏之前和潜伏之后。

1)潜伏之前

潜伏之前的阶段直接使用装甲板光学特征即可判定敌我。对所读入图像进行通道分离、膨胀腐蚀等图形学操作之后，进行灯条筛选、匹配，最后进行 pnp 解算即可获得敌方装甲板位置。目前我队该系统基本趋于完善，能够在地方扭腰的时候精准锁定装甲板。

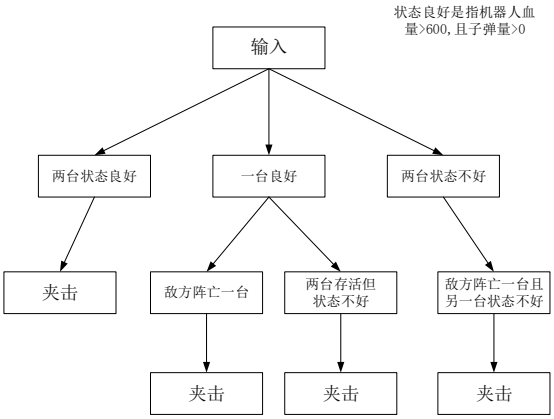
2)潜伏之后

潜伏开始之后两车不再考虑装甲板的颜色特征，而是识别视野内所有的装甲板。但是识别之后不是立即进行打击，而是查看处于同步状态下的由哨岗广播的位置信息，如果当前锁定的装甲板经计算发现与我方另一机器人坐标重合，则不进行打击，否则进行打击。这一方法依赖于两车之间的位置信息同步。

2.3 机器人配合调度

良好的机器人配合调度能够快速建立优势、发动有效打击，是比赛中最为复杂、核心的部分。

目前决策部分使用行为树算法，在设计时考虑了己方和敌方的血量、弹量、buff 激活状态、是否在火力覆盖范围内等信息作为决策系统的输入，根据状态信息变化实时输出机器人行为。



机器人由自身状态可完成自主决策，可独立完成攻击、躲避、巡逻和抢占 buff 点动作。哨岗可根据判断敌我状态进行全局决策，使我方机器人协同作战进行夹击支援动作。

后续可以加入强化学习来更好的调整阈值，使得决策更合理。

2.4 模块调度方案、应急预案

虽然各个节点的代码均反复调试无误，但是比赛中难免会有未考虑到的情况出现，可能导致某个节点崩溃、死循环等等。为了维持整套系统的良好运转，必须设计模块调度及应急处理方案。

每一个功能节点都会在一同个话题上以固定频率发布节点编号信息以证明其处于正常运行状态，同时我们也创建了错误代码。这一话题上的内容由一个模块管理节点监听，当一定时间收不到某一节点信息（或收到错误代码）之后，调用外部脚本对其进行重启，同时通知其余受影响节点进行复位操作。这样可以避免意外导致的节点崩溃而使整个系统无法运行。

2.5 调试工具设计方案

调试工具选择了 ROS 自带的简单、好用并且强大的 rqt 工具。该工具可以监视话题上的数据包、节点间的通讯，也能够手动控制数据包的发送。

节点代码的测试也使用了 Visual Studio 自带的内存检测工具，用以评价节点运行时所需运行内存的大小、有无内存泄露等漏洞，以便提出针对性优化。

3. 总结部分

3.1 自我评价

我队路径规划与导航功能实现并未直接调用开源方案，而是根据比赛实际需求自主开发，即除路径最优外考虑火力代价、友方支持等因素得到合理规划结果。虽然当前方案并未达到开源成熟方案的鲁棒性，但我们将在日后的开发与调试中反复打磨优化技术与工程细节；同时结合开源方案与前沿技术，努力使规划与导航系统更加完善。

3.2 预期功能阐述

预计能够在技术报告审查之前完成上述所有系统的编写、调试。完成一套能够自主决策、智能运动规划、自动寻敌打击的高速、鲁棒的综合系统。

邮箱：hrbeu_rm@163.com
微信公众号：哈工程创梦之翼战队
微博：@哈尔滨工程大学创梦之翼战队
地址：哈尔滨市南岗区南通大街145号哈尔滨工程大学61号楼6004室