雷达站技术报告

郑煜，袁宇恒
上海交通大学

目录

1 整体思路，设计思路以及战术定位

在本赛季，雷达站主体功能主要分为三点。

1.1 多方位的视野

第一点是最基本功能即为云台手提供多方位的视野，使得云台手得到获知战场的整体信息，同时视野图像也能提供给雷达站程序作进一步的自动分析。该功能技术实现主要依靠机械结构设计，传感器的选择及安装以及如何通过自定义 UI 进行显示。

1.2 为战场敌方车辆定位

第二点即为战场敌方车辆定位，也为雷达站最核心的功能之一。该功能于比赛中除了能给到操作手 UI 上实时的小地图信息以外，更重要的是其为雷达站进一步的自动分析和决策提供了可能，其精度决定了分析和决策的可靠性。该功能实现主要依靠神经网络给出车辆的图像预测框以及对激光雷达点云和图像预测框信息融合来得到车辆相对于相机的位置，此外为了准确得到车辆的世界坐标，相机相对于世界坐标系的位姿估计也是一项十分重要的工作。同时，为了提高程序的鲁棒性，我们还设计了一系列后处理方式来处理误识别的信息。

1.3 雷达自动决策

第三点为雷达自动决策。该项功能决定了雷达站这个兵种的上限，本赛季我们的设计思路主要就其防御功能来考虑，即预警，该功能在赛场上能提供给操作手视野盲区的信息，助其及时反应来避免被袭击。其分为两大块，飞镖预警及车辆预警。飞镖预警我们主要采用了传统方法对发射飞镖头进行检测，车辆预警我们主要根据第二部分得到的位姿及车辆位置信息，采取了图像反投影检测和直接位置检测两种方式来检测进入感兴趣区域的敌方车辆，并通过自定义 UI 和车间通信来反馈预警信息。

2 机器人主要技术参数

技术参数及传感器型号如下

表 1: 雷达站技术参数

主要参数	数据
重量（含支架）	6.3kg
总体尺寸（不包含支架）	245mm*250mm*195mm
激光雷达数量	1
相机数量	3

表 2: 雷达站传感器类型

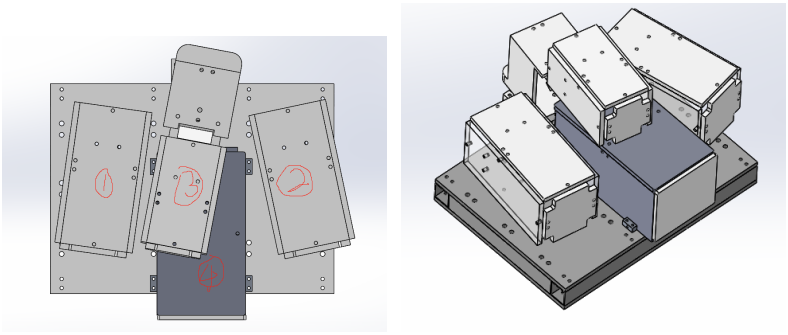
传感器	使用型号
左/右相机	MV-SUA630C-T 手动变焦镜头 4-18mm 3mp 1/1.8”
上相机	MV-UBS31GM 长焦镜头 35mm 5mp 2/3”
激光雷达	Livox Mid70

3 机械结构

雷达整体的机械结构主要分为两个部分，一个是上云台，另一个是支架。

3.1 上云台

整体结构图如图1(a)及1(b)



(a) (b)

图 1: 上云台上视图

如图1(a)所示, 上云台的底层结构为层状结构，两块玻纤板之间加上铝方管以加固，减少晃动。底盘下面有一块连接件，与购买的相机支架相连。再往上看，图中①②③相机传感器的安装位置，④为雷达传感器安装位置。其中①②位置安装方式相同，均采用铜柱增高（55mm) 的方式安装，左右相机朝向均与激光雷达朝向（平台中轴线方向）成一夹角，以满足相机视野对赛场的全覆盖，其中左相机夹角 8 度，右相机夹角 12 度，其中左相机夹角较小，是由于雷达基座于赛场偏左位置。而③号相机传感器安装方式有所不同。为了使该相机可以左右上下移动，在这里用了螺栓制作了两根丝杆。

首先来看上下移动部分,3D 草图如图2

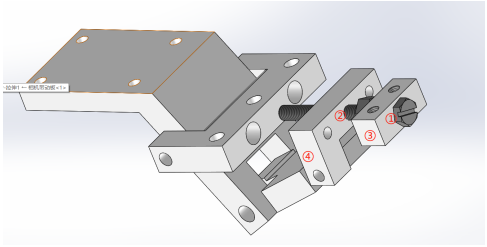


图 2: 上下移动部分 3d 草图

①号孔与②号孔同心，用一根 M4*45 的螺栓相连，其中①号孔无螺纹②号孔有螺纹，且两孔之间有防松螺母（紧贴一号孔），这起到了一个限位作用。由于物块③是固定的，物块④是滑动的，在旋转螺栓时，带螺纹的滑块④便会滑动, 进而实现上下滑动。左右滑动也是利用了这样的结构。

左右滑动结构图3

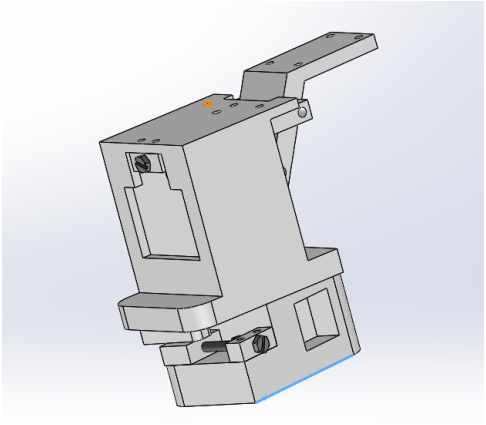


图 3: 左右滑动结构

3.2 雷达支架

雷达支架采用的是普通的相机支架。

图 4: 雷达支架

雷达支架底部使用的是磁吸，通过吸力与地面固定，如图5(a)。

(a) 同支架　　(b) 近处

图 5: 雷达磁吸

4 算法

雷达站的核心是算法，即如何用程序实现整体思路中所述的各项功能，主要基于机器人自传感器读入到后处理的各个阶段来阐述。

首先，图6为雷达站主程序的流程图（出于图像美观性考虑，仅展示了雷达站核心处理部分，未将多进程独立相机图像以及飞镖预警部分加入其中）

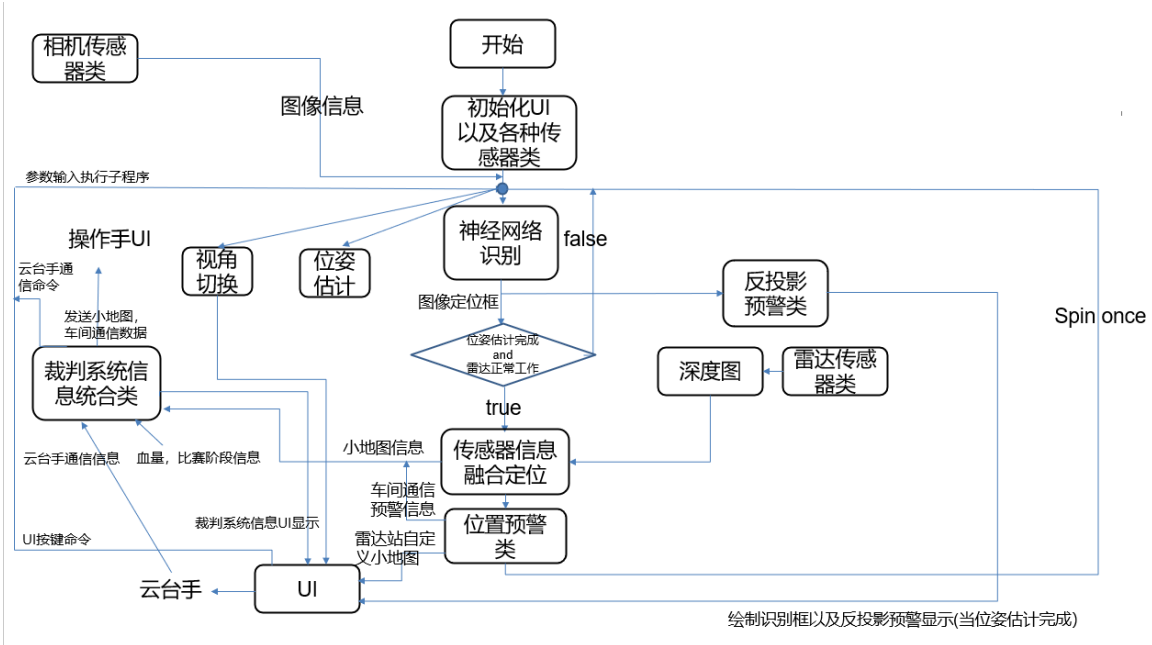


图 6: 雷达主程序流程图

下面我将就各个部分进行详细说明

4.1 传感器类

4.1.1 相机传感器

雷达站运算平台端和传感器端中间采用长电缆连接，故存在相机断连的可能性，且在雷达站程序启动时，传感器与雷达站运算平台不一定已经连接上，考虑到这些问题，设计一个相机

传感器类来处理这些问题是十分有必要的。该类基于相机厂商提供的 demo 驱动程序设计，其在程序启动但未检测到相机连接时会检测到这个异常，并在主程序每次循环中不断尝试启动相机，当首次启动相机时，会提供操作 UI 界面供人工选择曝光和相机增益等参数并记录下该参数。后续当检测到相机异常掉线，该类会捕捉该异常，然后同样在每次循环中尝试启动相机，当然，后续启动可能在比赛进行中出现，无法人工操作，故而会自动读取上次设置好的相机参数然后启动。

4.1.2 雷达传感器

我们在比赛只使用了一个激光雷达，故而雷达传感器类可以基于全局类来设计，但我们同时要对雷达和相机信息进行融合，这就出现了雷达对多个相机要提供接口的情况，故而雷达传感器类设计是雷达点云信息全局处理加反投影深度图对象化处理。

由于采用了 Livox 雷达提供的基于 ROS 的 demo 驱动，我们雷达传感器类实际上是订阅 Livox 雷达的 ROS 驱动点云节点，通过一个线程不断接收雷达点云信息。由于 livox 雷达是基于非重复式扫描设计，其点云需要一定的积分时间，我们设计了一个队列来接收，该队列存储的实际上是投影到相机平面上点云（在投影相机平面外的点云则去除）的像素平面坐标数组，每个队列对应一张深度图，实际上便为各个投影点在相机坐标系的 z 坐标矩阵，深度图初始全部位置值为 nan，代表全部位置无信息。队列长度上限设置为 200，结合 ros 点云发送频率 10Hz，实际上我们深度图上保留了 20s 的点云，由于我们主要需要深度信息，物体运动产生的点云结构畸变对于赛场整体的深度信息影响不大，而更长的积分时间能够使得深度图更加稠密，故而我们保留了较长时间段内的全部点云。以上过程具体公式为

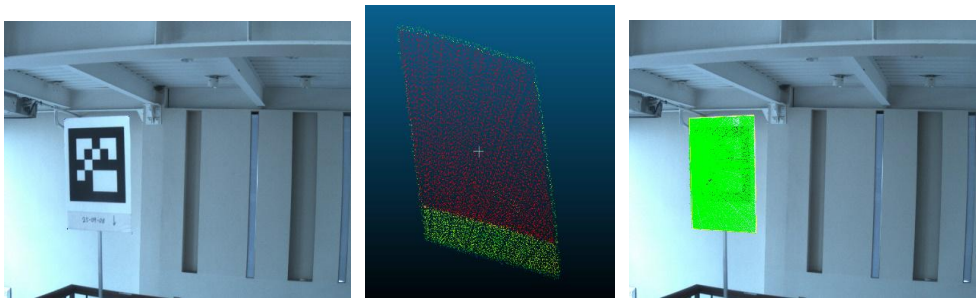
$$z_c \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K_c \begin{pmatrix} R_l^c & t_l^c \end{pmatrix} \begin{pmatrix} x_l \\ y_l \\ z_l \\ 1 \end{pmatrix} \qquad D(u,v) := \min\{z_c, D(u,v)\}$$

更新策略上，在点云离队时，根据 pop 出的投影点像素平面坐标数组，将深度图上其所有投影点位置的值置为 nan，新点云加入时，将其投影点像素平面坐标数组入队，然后将各个投影点的 z 坐标值和深度图对应位置的值作比较，取各点整体的最小值，来更新深度图，以保证新出现遮挡物体（通常会使得对应位置其观测者更近）的深度能被准确测量。

以上工作即阐释了对象化处理的思路，即为每一个相机提供一个点云投影队列以及点云投影需要的参数[1]。其中雷达和相机联合标定是一个比较有挑战性的工作。我们尝试了很多方法，比如基于 ICP 使得雷达和相机在空间中不断移动每次移动会有雷达到雷达和相机到相机的变换矩阵，然后迭代优化雷达到相机的变换矩阵。最终我们参考于 livox 官方开源的标定方法，基于 Apriltag 定位系统，通过在相机图像定位到 Apriltag 标定板四点（先定位标定板内部四点，然后 PnP 定位相机位姿，反投影定位外部四点），然后将点云中的标定板四点找到（我们是采用了 CloudCompare 工具将点云中标定板截出，然后基于预先得到的相机与雷达位姿（可能不准确）先将点云投影到相机平面上，点云投影区域为一个凸四边形，只需找到其最小外接矩形，并

[1]主要是相机内参以及雷达和相机位姿标定参数

计算离外接矩形四个角点最近的点即为点云中标定板的四点，这样不失为一种更为快捷的半自动化方式），这样得到多组四点对应关系便可采用 PnP 方式计算相机和雷达位姿。经试验，如图7(a)7(b)7(c)，这样得到的位姿反投影效果也是不错的，不失为一种较为有效且简单的手动标定方式。



(a) camera　　(b) 点云中切出的标定板　　(c) 投影效果

图 7: 雷达相机联合标定

4.2 主程序类

雷达站主程序类作为对于传感器类，预警类以及子进程统一调度而设计。其主方法 spin once 是整个雷达站主程序的主循环函数，包含读取相机图像，神经网络预测，以及将图像预测框送入预警类（预警类中集成了相机雷达信息融合定位模块）进行后处理。为便于外部通信输入，主循环会在每次进入 spin once 函数前，检查各个外部命令的置位符，若置位为真，则会在 spin once 前执行相应的子程序命令，包括相机位姿估计，主视频源切换，打开子进程相机画面，开/关视频录制等，其中位姿估计以及打开子进程相机画面将在后面做进一步阐述。

4.3 雷达站主循环

下面对主循环的各个主要步骤进行阐述。

4.3.1 多传感器说明

值得一提的是，前面机械部分提到，我们在实际比赛采用了一个激光雷达以及左右两侧各一个相机的设计来得到完整的视野（由于采用了变焦镜头，焦距约 14mm 以放大视野中车辆来满足神经网络对于尺度的要求，FOV 相应变小，故需要双相机补充视野）故而在算法设计上需要考虑如何对双相机得到的信息进行结合，下面会做详细阐述。

4.3.2 神经网络预测

通过相机传感器类得到双相机各一帧图像后，将图像送入神经网络进行预测。由于雷达站项目对帧率的要求并不高，我们采用了双网络预测的设计，主要是为了克服在雷达站整体视野下装甲板尺度较小，受背景干扰较大的问题，若网络输入图片的视野内只有少量车辆，则与自

瞄装甲板任务的输入图像类似，则可迁移自瞄的深度学习网络来完成雷达站任务。经试验，经过该方法处理，装甲板预测的准确度和召回率均有显著提高。

具体实现为，第一层采用 YOLOv5s 网络，使用 DJI ROCO 数据集，2019 年中国科学院大学开源的 icra 数据集以及自行制作数据集进行训练，预测 car,watcher,base 三个标签，实际最后主要使用预测出的 car 标签，其中输入尺寸为 640*640。在第一个网络预测出 car 标签的 bounding box 后，我们将原始图像对应于 bounding box 的 ROI 取出，作为第二个网络的输入。第二层网络采用修改了 Detect 层的 YOLOV5s 网络，使其输出为装甲板四点及装甲板类别，输入尺寸为 256*256。以上过程如图8所示。

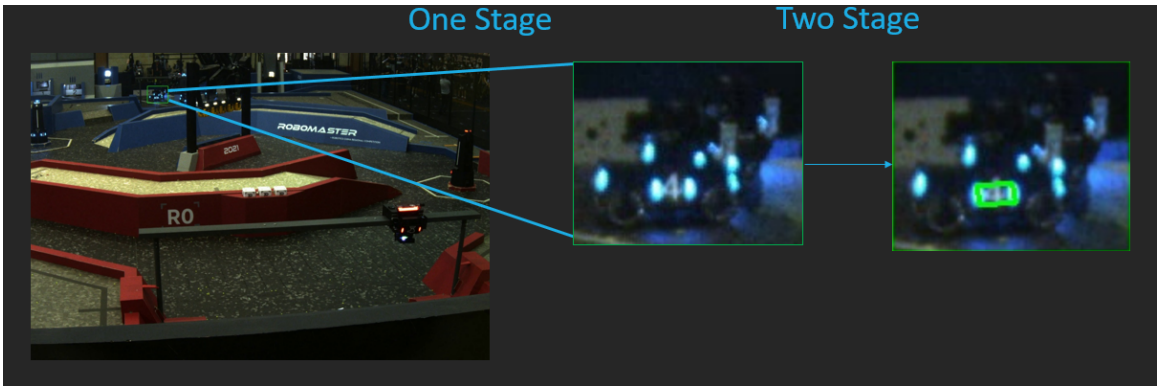


图 8: 双层神经网络设计

我们对每个 ROI 经过第二层网络输出的装甲板种类统计频率，选择频率最高的种类作为对应车辆 bounding box 的装甲板类别，并将所有 ROI 预测出的装甲板合并输出。当一 ROI（在对应置信度阈值限制下）无装甲板输出时，通过亮度阈值，加 HSV 阈值筛选，来粗略分辨该区域车辆敌我信息（即只输出为红或者蓝车辆），若该方案仍不可分辨，则只输出该区域为车辆（未知装甲板编号）。

对于上述合并输出的装甲板，还需做一次装甲板去重，这是由于在后续处理中，将使用装甲板四点的 bounding box 作为各个 id 车辆的图像定位框，其相比于车辆的 bounding box 的优势将在信息融合定位部分中进一步阐述。由于是各个 id 车辆，而在比赛中车辆 id 具有唯一性，则对于每个 id 的定位框也只需要唯一一个，假以此先验，装甲板去重只需要遍历各个 id，然后取所有预测出的装甲板中置信度最高的一个便可。该方案同时也可去除一些误识别的装甲板，因为通常来说若该 id 的装甲板在视野中出现，其置信度相比误识别肯定是要高的。

4.3.3 相机位姿估计

从流程图中，可以看到，在进行传感器融合定位前，除了需要雷达传感器已然连接，还需要完成相机位姿估计，该工作也是进行反投影预警的前置条件。在主循环中，若位姿估计未完成，则会跳过信息融合阶段，只通过反投影预警类绘制车辆 bounding box 显示在自定义 UI 上，该操作也使得位姿估计工作能在程序启动后进行，使得在位姿估计未能完成的情况下，前置功能能够正常运行。

本赛季我们尝试了许多方式来进行相机位姿估计，如自动识别或手动标定赛场提供的 R0 及 B0 标签等，但都没有达到理想的效果。由于我们位姿估计采用的是 PnP 方式，查阅资料后了解

到，PnP 方式估计精度与各个标定点实际空间距离成正相关，故而采用了大范围手动标定的方式，达到了理想的效果。具体方案为，使用 PnP 四点法估计位姿，四点分别为敌方基地顶端，（敌）我方前哨站顶端，以及我方 R0/B0 定位标签的上方两点，这些位置的空间坐标可通过规则手册及小地图等比例缩放计算得到。

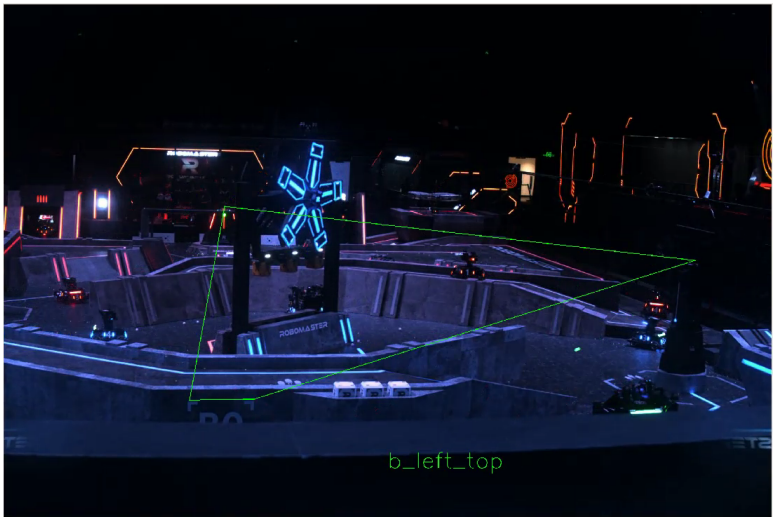


图 9: 位姿估计 UI 及四点

4.3.4 反投影预警类

反投影预警无需用到激光雷达信息，故而可以在 fusion 前进行，且可以通过反投影预警类对图像预测框做一些后处理。首先叙述反投影预警的机理，其前置条件是已知相机位姿。我们在比赛前根据比赛规则手册提供的信息以及通过小地图等比例缩放计算，可以得到感兴趣位置的空间坐标，为方便起见我们规定感兴趣区域在 BEV 下为凸四边形且至多只有一对边高度不同，这样根据位姿，相机标定信息，我们可以将感兴趣区域四点投影到图片中，其也为一凸四边形。我们只需判断敌方各个车辆 id 对应的装甲板框任意一点落于对应凸四边形内便认为该目标出现于感兴趣区域中，触发自定义 UI 中对应区域闪烁，若为车辆预测框（如基于 HSV 预测得到的框以及下面将提到的基于 IOU 预测得到的框），这可取框的 $(\frac{1}{3}w, \frac{3}{5}h, \frac{1}{3}w, \frac{1}{5}h)$[2]作为装甲板位置，如图10，其中红色即为预测框。我们试验下来该位置对于装甲板位置估计效果极佳，并可用于后续融合定位[3]。

点落于凸四边形判断可采用以下办法，即顺时针遍历凸四边形四点，对于每一个点，计算以该点和预判断点为端点向量和以该点和沿顺时针该点下一点为端点的向量的外积，如 $\vec{AP} \times \vec{AB}$，若这些外积同号，则可说明点在凸四边形内，该过程如图11所示。

图12为反投影飞坡区域预警效果。

装甲板预测网络在部分情形下无法预测出装甲板种类及位置，在这种情况下，除了上述 HSV 判断敌我外，我们在反投影预警类内对图像预测框增加一后处理过程，即基于 IoU 的预测器。我们尝试使用一些追踪器进行目标追踪，来达到补帧的效果，但追踪器一旦跟丢，会造成一段时

[2]分别为左上点 (x_0, y_0)，以及框大小 (w', h')（以原始框的左上角为坐标原点）

[3]当然，我们只在我们网络预测出的车辆预测框上测试过，但这种先验应该是泛用的

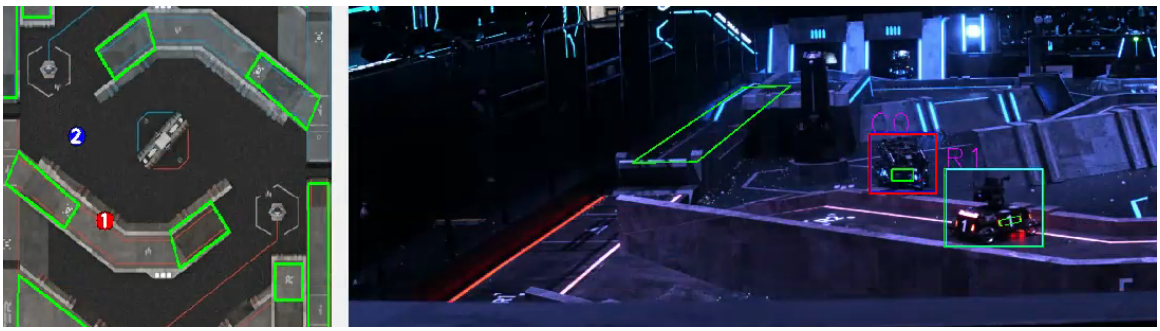


图 10: 装甲板框估计

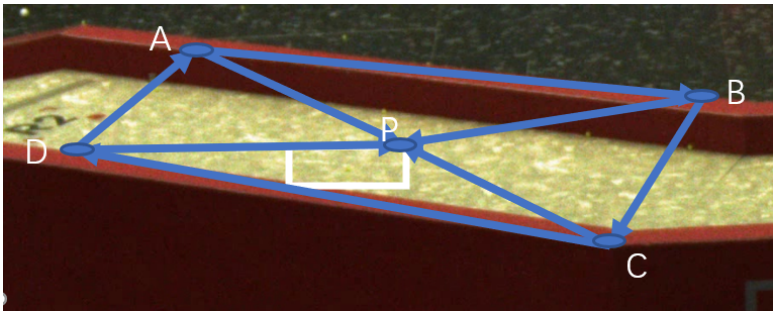


图 11: 凸四边形内点判断

间内的误识别，且难以通过后处理来排除。故而我们弃用了追踪器。我们考虑到第一层神经网络对于一定范围内甚至遮挡目标有着稳定的预测效果，可以通过存储一帧车辆预测框[4]，如图13，在对于每一帧预测结束，对每一个没有预测出装甲板框的车辆 id 检查上一帧是否有对应 id 的车辆预测框，若存在，则对计算该预测框和当前帧所有车辆预测框的 IoU，在一定阈值[5]滤除后，取车辆预测框中 IoU 最大者作为当前帧该 id 的车辆预测框，并通过上述装甲板位置估计方式来估计其装甲板位置，如上述图10。

对于当前帧基于该方法预测出的车辆预测框是否存储作为下一帧预测依据，我们最终是没有存储，这是由于在测试中发现，若一车辆 A 在某一次被预测后，其预测框为另外一辆车 B 的预测框，此时，这样预测产生的误差很小，因为 IoU 判断的先验下，两辆车的位置十分接近，可若将该预测框存储，并在之后的一段时间内，车辆 B 没有移动，而车辆 A 进入遮挡区域导致其目标丢失，此时，存储的 A 车 id 绑定框会一直将其位置预测为 B 车位置，直到 A 车出现并可预测出装甲板，或 B 车快速移动导致 IoU 预测失效，这将导致长时间的误识别。

值得一提的是，我们反投影预警类是对每个相机单独设置的，且飞镖预警仅在对应右相机的预警类中设置。故而预警也是在对应图像上单独报警，对于感兴趣区域重叠问题，可以对比两个图像中同一预警区域四点落在图像内数量，多者保留预警区域，少者去除该预警区域。这种方式虽能减少预警判断，但可能云台手在看某一相机视频源时，重叠区域报警在另一个相机图像中导致预警未能显示，故而重叠区域不去重也是可以的。

[4]这些预测框均通过第二层网络预测信息得到其对应装甲板编号

[5]即若最大 IoU 小于阈值，则全部结果舍弃，对应 id 不做预测

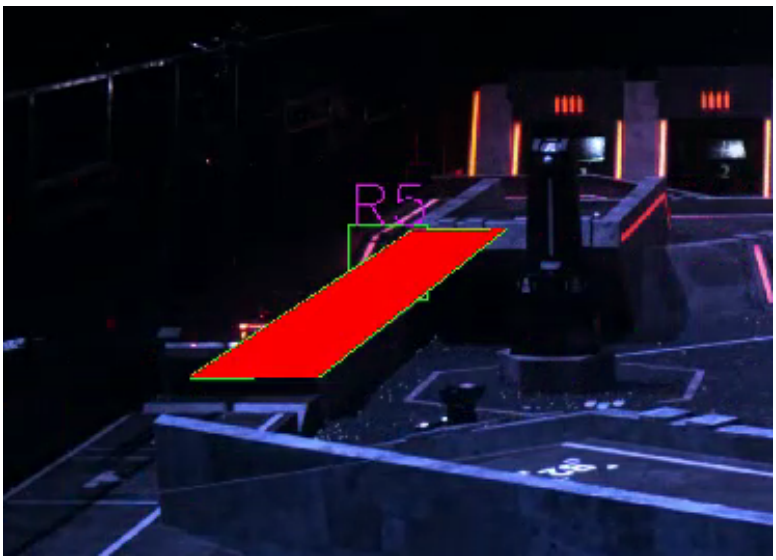


图 12: 飞坡反投影预警

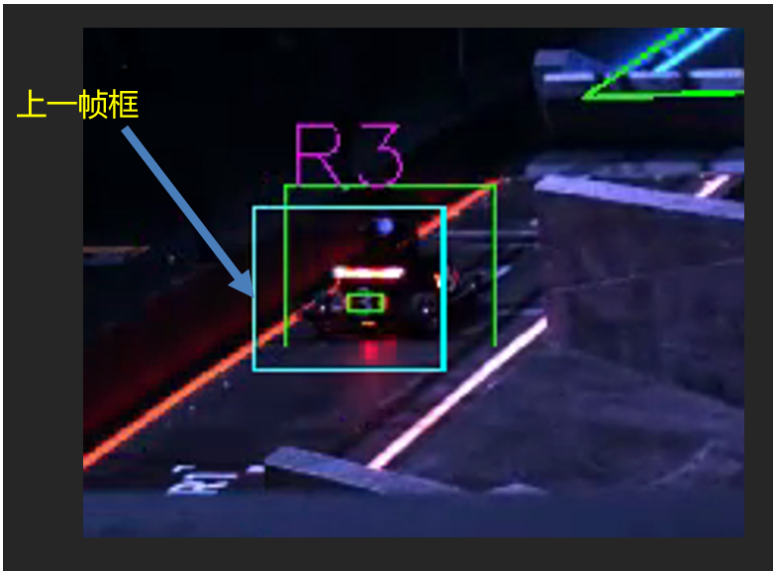


图 13: 存储一帧，计算 IoU

4.3.5 飞镖预警类

反投影预警类绑定了一个子类，即飞镖预警类，因为飞镖预警的工作也需要反投影划定感兴趣区域，我们将飞镖预警的感兴趣区域分为两部分，一个是飞镖检测区域，其被划至了空中来排除能量机关转动引发的误识别，另一个是飞镖发射架闸门开启指示灯闪烁检测 · 区域。我们将飞镖预警分为两阶段检测，第一阶段，飞镖发射架闸门开启指示灯闪烁，通过亮度阈值过滤以及帧差法进行检测（值得一提的是，能量机关转动也会被检测到，故而需要云台手判断），当检测到则提醒云台手通过车间通信手动开启第二阶段，并同时指示哨兵进入反导准备阶段。第二阶段为飞镖检测阶段，该阶段将感兴趣区域移动至空中，通过亮度阈值，HSV 区域过滤（防止其他光源干扰）以及帧差法检测区域中出现的飞镖头，如图14, 然后自动通过车间通信发送飞镖发射信号给哨兵指示其反导。其中，由于反投影区域可能是凸四边形，帧差法取该凸四边形的 bounding box 取原图像 ROI 做帧差，是否落于区域的判断同样采用上述思路，各个阶段检测

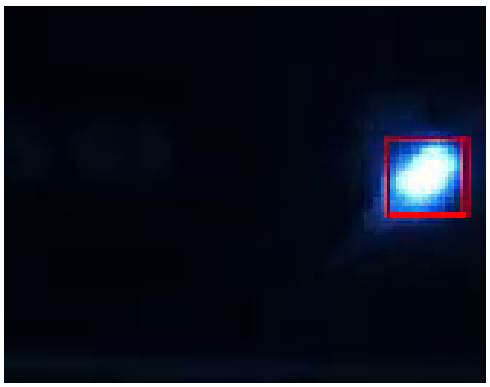


图 14: 检测飞镖头

出的识别框为帧差法检测出边缘的 bounding box。

4.3.6 传感器信息融合定位及位置预警类

当完成位姿估计和雷达正常工作时，便进入信息融合定位阶段，这一部分全部整合在位置预警类中，其包含三个阶段，取雷达深度图计算各个装甲板世界坐标位置，对位置做后处理（包含 z 轴突变平滑，以及简单位置预测器预测），然后对各个敌方车辆位置做位置预警。

首先是第一阶段，取雷达深度图计算各个装甲板世界坐标位置。该步骤基于上述投影得到的雷达深度图，其即为相机坐标系 z 坐标矩阵，其各个像素点和图像各个像素点一一对应（当然在忽略联合标定误差前提下），故而可直接对装甲板定位定位框对应深度图 ROI 取均值[6]作为该框中心点对应相机坐标系中点的 z 坐标值，再转换至世界坐标系，便得到对应 id 车辆在世界坐标系中位置，取 (x, y) 作为小地图（BEV）中坐标值。全过程如图15所示。

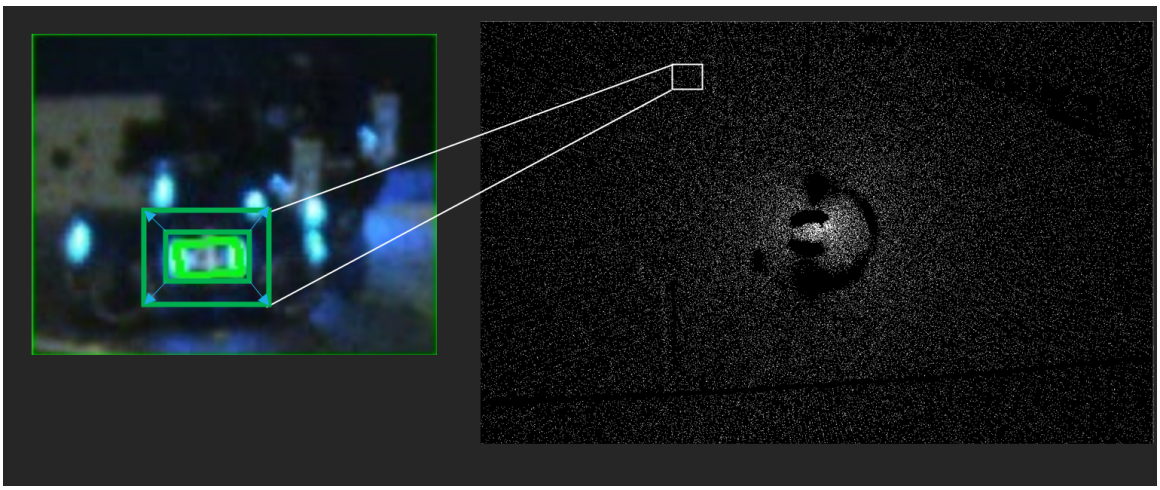


图 15: 深度图对应

[6] *nan* 均值，即忽略 *nan* 元素后对剩余元素取均值

位置计算过程具体公式为,

$$\hat{z}_c = \frac{1}{N_{ROI}} \sum_{ROI} D(u,v) \qquad \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} = \begin{pmatrix} R_c^w & t_c^w \end{pmatrix} \begin{pmatrix} \hat{z}_c K_c^{-1} \begin{pmatrix} u_{center} \\ v_{center} \\ 1 \end{pmatrix} \\ 1 \end{pmatrix}$$

实际上，实际测试中我们发现直接一一对应会在装甲板较小时受到标定误差噪声的干扰效果不佳，最终我们采用的方式是将装甲板 bounding box 以中心为基准扩大一倍，测试下来效果极佳，由于扩大带来的背景干扰将在后续阐述处理办法。

对于上述方法的得到的坐标，由于背景干扰，可能会有误差值。我们主要依靠先验知识，对一种误差进行校正处理，即 z 坐标突变误差。这种误差的产生一般是由于相机视角中一些遮挡区域，这些区域通常来说会有遮挡与非遮挡区域的交界线，在这些敌方，深度是突变的，故而在做 ROI 平均时，交界两侧的值均参与了平均，使得测距出现误差，车辆位置被前移。

前面在神经网络预测部分提到我们采用了装甲板预测框而不是车辆预测框来和深度图对应，同样是由于深度的突变。若使用车辆预测框，则在图像上方区域，由于单位尺寸深度变化值较大，会有较多的远处背景的深度加入均值运算，导致深度偏大，然后车辆位置被后移，采用装甲板框可以有效减缓该种后移的误差。

我们可以由世界坐标系中 z 坐标的突变来判断这一误差（指前移误差）的产生，即存储一帧预测点的世界坐标系 z 坐标，对于每一帧，将当前计算得到的世界坐标系 z 值和上一帧同一 id 的世界坐标系 z 值比较，若突变超过一定阈值[7]则将当前 z 值转换为前一帧 z 值，具体方法为，利用相机位姿估计得到的相机在世界坐标系中位置，得到相机到当前帧物体位置向量，然后计算上一帧 z 值和相机坐标 z 值作差再除以相机到物体向量 z 值得到缩放系数，向量乘以缩放系数再加上相机世界坐标，即得到调整后物体世界坐标。具体公式如下，

$$\begin{pmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \end{pmatrix} = \begin{pmatrix} R_c^w & t_c^w \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \qquad radio = \frac{z_{last} - z_{cam}}{\hat{z} - z_{cam}} \qquad \begin{pmatrix} \hat{x}' \\ \hat{y}' \\ z_{last} \end{pmatrix} = radio \cdot \left(\begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} - \begin{pmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \end{pmatrix} \right) + \begin{pmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \end{pmatrix}$$

由于我们采用了两个相机，而对象坐标只有一个，故而要在基于上述处理办法，增加一个合并逻辑，如图16所示，其中在做 z 轴突变调整时，需要基于各自的相机坐标。

实际上也可以采用两个相机同级预测结果[8]作平均，我们采用固定相机输出的原因主要是在测试中发现右相机的估计较左相机要好不少。

4.3.7 Location Prediction

流程图中，location prediction 还没有提到，其即为一个匀直预测器，我们存储两帧车辆位置，当当前车辆 id 没有任何定位框估计时，若其前两帧都估计出了位置，则该帧位置由前一帧位置加上乘上一个缩放系数的前两帧位置偏移向量。并对预测进行计数，当一个目标预测次数超过

[7]我们在测试中加入了两个先验限制来避免一些误差的发生，即只检测正突变和对前一帧 z 值设定上界，实际上即只对由地面突变至高地的情况作校正

[8]如同为神经网络预测装甲板或同为 IoU 预测

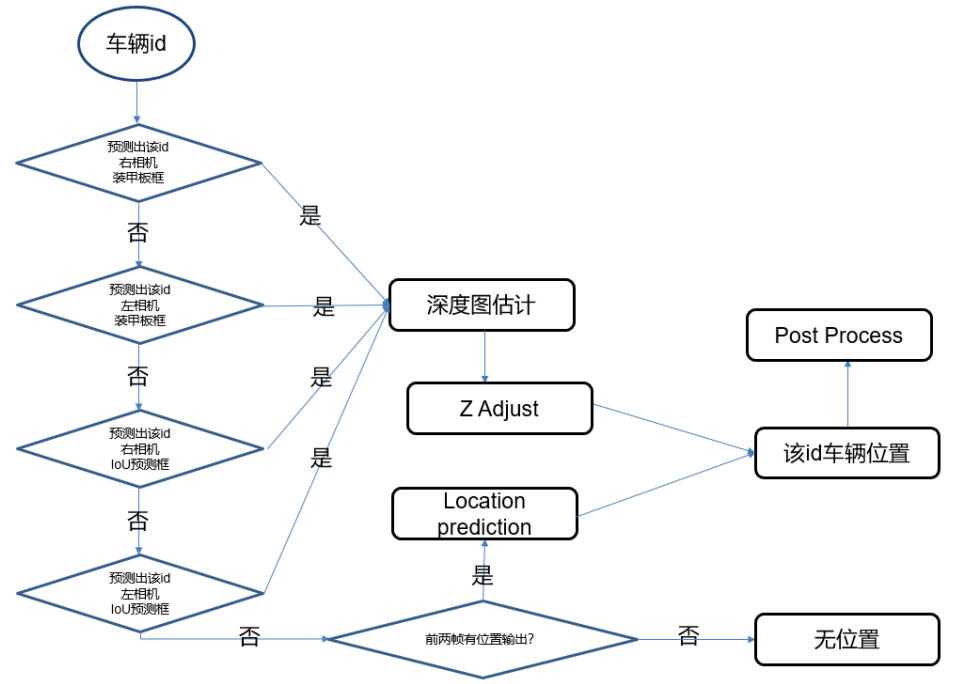


图 16: 合并逻辑

阈值则停止预测。实际上，匀直预测对于雷达站大多数目标消失情况是无法处理的，只能处理突变遮挡区域[9]，其还有一个功能是对 UI 显示做一些平滑处理[10]。

4.3.8 Post Process

位置预测后的在算法上后处理即位置预警。这是一个比较简单的功能，在地图上划定一系列凸四边形预警区域，用与上述相同的凸四边形内点检测方式便可获知敌方车辆是否出现在感兴趣区域内。

4.3.9 多进程独立相机图像

在 python 中多线程若使用 opencv 来显示图像的话，在创建窗口时无法显示并使得主线程阻塞，而若不采用并行显示图像，对于一个帧率要求高的相机视野来说是不可行的。故而，我们采用了 python 多进程来显示独立相机的图像。多进程之间主要采用队列进行通信。具体流程如图17，其中 task 默认为 0，等待信息是指在队列为空时，对队列执行出队操作会阻塞进程。

4.3.10 与裁判系统通信

通过 USB 转 TTL 进行和裁判系统主控模型的串口通信，以读取裁判系统信息和发送信息到裁判系统。

小地图通信基于官方协议。当前面算法中得到小地图信息后，对算法得到的小地图位置执行变换，设原来位置为（x,y)，变换为（x，15+y)，这是由于世界坐标系原点定义到赛场左上方，

[9]即车辆径直开过一小片遮挡区，如雷达站视野前方哨兵区域

[10]即缩放系数调得比较小，来让目标延迟消失

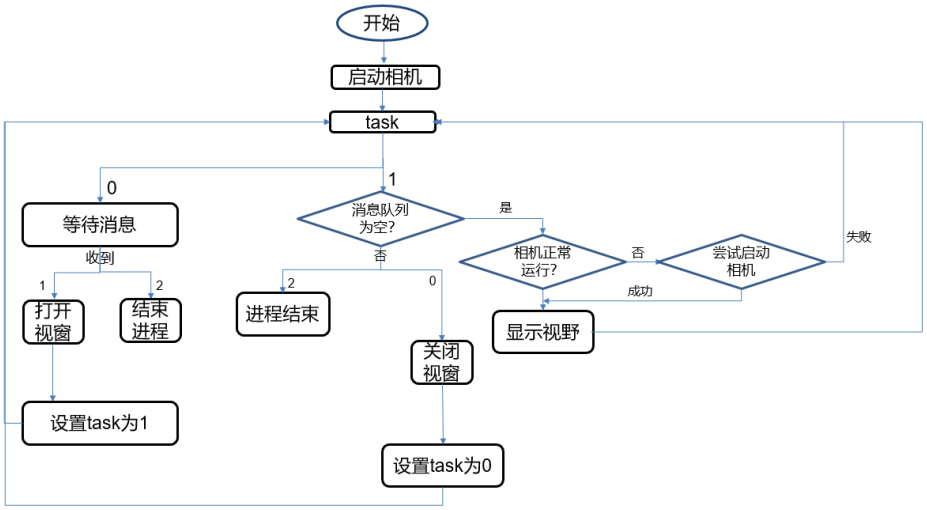


图 17: 多进程相机流程图

而小地图坐标原点定义在赛场左下方。预警信息发送基于车间通信协议，经由 numpy 库将位置转化为 float32 类型，然后发送。由于协议规定每轮发送一辆车信息，故而用全局变量控制车辆发送顺序，从 1 到 5 轮换。其中，算法处理时，对于未预测到位置的车辆 id，会将其位置置 0，发送上则对置 0 的 id 执行不发送处理。

对于车间通信预警，对于上述两个预警类，我们会在其触发预警信息时，同步发送预警给裁判系统。包定义内容 ID 判断预警区域，对于飞坡预警区域，己方与敌方高地双侧坡预警区域，算法主程序在发送预警时，会同时发送预警区域内敌方车辆 id，通过二进程位编码置于包内容数据段。预警会判断发送间隔，同一预警信息发送间隔需大于 2s 以保证小地图通信频率，且由于算法可能会误识别灰色装甲板车辆。基于裁判系统读取的敌方血量信息，若预警区域内均为血量为 0 的车辆则不发送预警。

5 UI 人机交互

5.1 雷达站自定义 UI

如图18所示，自定义 UI 基于 PyQt5 设计，可适应自由缩放。

5.1.1 基于裁判系统控制优化控制

基于裁判系统读取，雷达站可获知比赛开始与结束信息，在比赛开始自动将 UI 全屏化并在未开始视频录制的时候执行开始录制命令，结束时将 UI 小窗化并结束视频录制。并会计算结束局数，在局数达到设置上限时自动结束程序。此外若位姿估计未能完成，在比赛开启时雷达站会自动读取上一次对应相机和阵营的位姿信息以保证雷达站主功能的实现。此外，通过裁判系统读取的血量信息会显示在雷达站主 UI 上的裁判系统反馈区域，并可根据血量变化判断血量上限变化以供云台手更好地分析比赛局势。

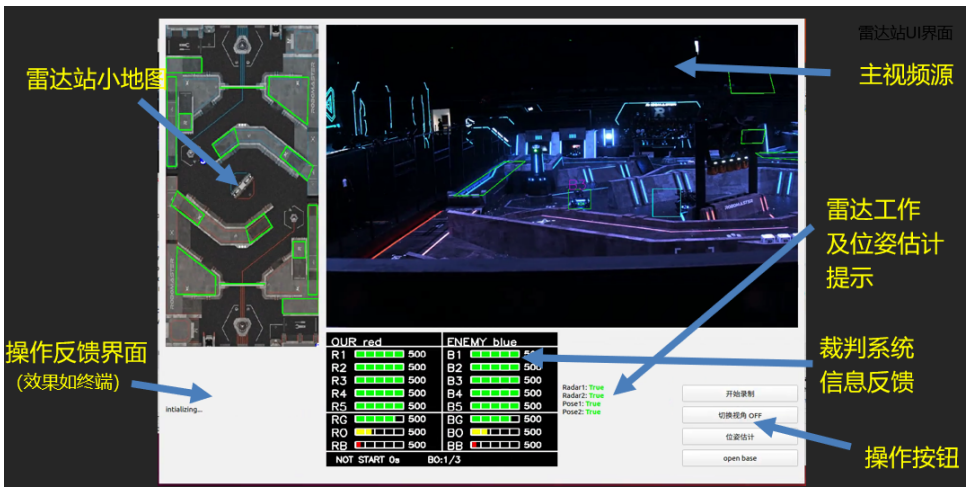


图 18: 雷达站自定义 UI

5.1.2 雷达站小地图

基于操作手 UI 小地图设计，通过小地图绘制类管理，上述位置预警类会将位置信息传递给小地图绘制类然后在地图上绘制敌我车辆位置。小地图绘制类会根据设定的我方阵营信息自动调整小地图朝向，以使得雷达站小地图与视野图像吻合。预警框会被绘制在小地图上，当位置预警发生时，对应预警框会闪烁（基于 opencv 填充凸多边形函数）三次，并主视频源左下角会闪烁红灯以提醒云台手关注小地图预警, 如图19。

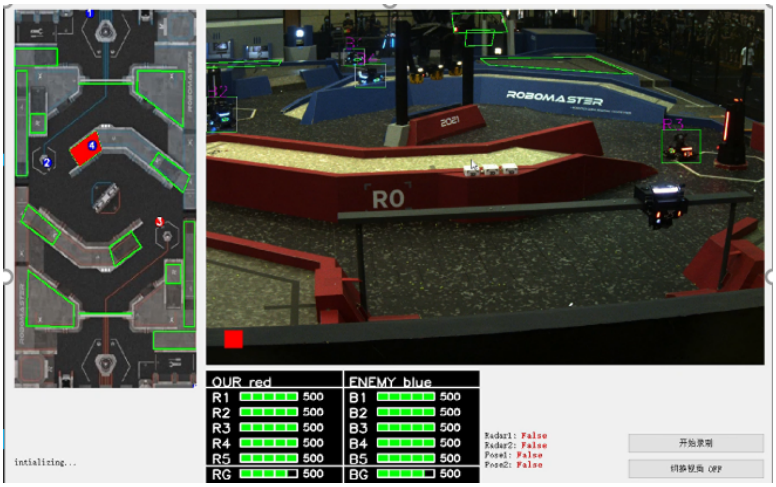


图 19: 红灯闪烁

5.1.3 主视频源

主视频源主要显示反投影预警类输出，其上为车辆预测框标注，装甲板框标注以及各个反投影区域，当反投影区域产生预警时会和小地图类一样，闪烁三次。上述提到的飞镖预警框也绘制在右相机的图像中，并在第一阶段预警时，在左下角以黄灯闪烁预警，如图20。

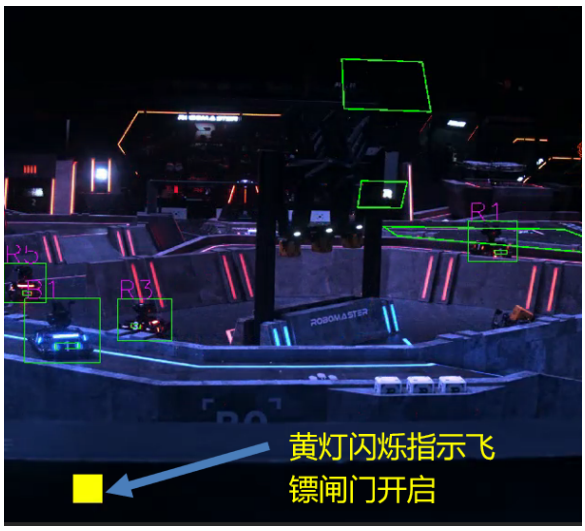


图 20: 黄灯闪烁

5.1.4 操作反馈界面

主要反馈飞镖预警发送信息，相机未正常工作信息，按钮操作反馈信息，以及主程序结束信息。

5.1.5 雷达工作及位姿估计提示

由于雷达是接收另一 ros 驱动程序 topic 发送的 message，故而在 message 没有发送时，雷达站主程序也能正常工作，故而需要通过判断接收 message 的子线程是否有接收到 message 来检测雷达是否正常工作。位姿估计在程序启动后进行，故而也需要提示雷达站上场队员位姿估计是否已经完成。

5.1.6 操作按钮

提供给雷达站上场队员执行雷达站准备工作，后三个按钮都会将对应置位符置为 true，当主程序执行 spin once 前，会判断置位符执行对应子程序（包括子进程消息发送），并将置位符恢复。其中 open base 即为打开/关闭子进程相机视野，当其为开启时，UI 会小窗化并在右侧开启视窗显示子进程相机图像。视野切换即切换主视频源的相机源。位姿估计即会执行位姿估计程序，位姿估计程序会显示左相机和右相机的实时视频，操作者可在视频中进行四点标定，并通过键盘操作确认标定或撤销。使用实时视频源主要是为了防止某帧图像标定目标被遮挡现象。云台手通过车间通信也能执行视野切换以及打开关闭子进程相机功能。接收对应车间通信键位信息后，雷达站主程序会直接调用对应按钮触发函数，相当于按下对应按键效果。

5.2 操作手 UI

如图21车辆在接收到雷达站通过车间通信发送的预警信息后，会在操作手 UI 上基于官方 UI 绘制协议绘制如图预警效果，其中左侧为飞坡预警区，右侧前后分别为敌方和我方高地双侧坡。在收到预警时，预警区域会变黄并持续数秒，且会在其中显示在预警区域中敌方车辆 id。

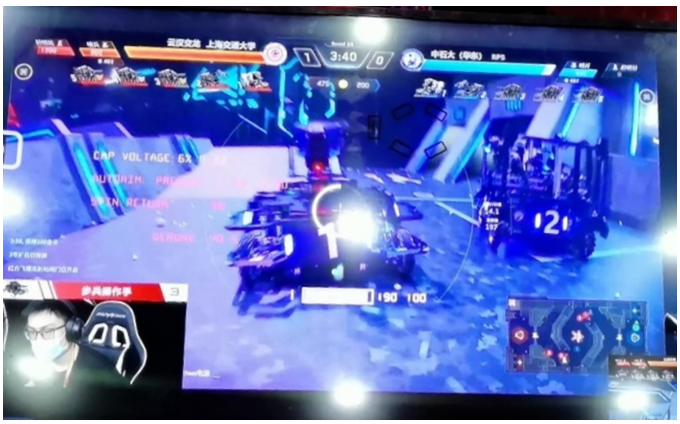


图 21: 操作手 UI 雷达预警部分

6 测试

雷达站由于在实验室没有十分标准场地，我们对于算法的测试主要基于分区赛所拍摄的视频和雷达点云录制。基于视频测试，我们对于算法做了很多改进，如上述预测框缓存问题发现，装甲板框放大一倍取 ROI，感兴趣区域划分和预警测试，z 轴突变问题发现以及相机位姿估计方案确定和测试。