香港科技大学 ENTERPRIZE 战队 Robomaster 2025 赛季雷达站算法开源报告

郭子霖 (Fallengold) *　　周思宇 †

2025 年 8 月 4 日

摘要

本项目为 RoboMaster 2025 比赛设计了一种雷达站单目相机的目标检测与跟踪系统，旨在实现场地内机器人的精准识别、定位与轨迹跟踪，为战术决策提供实时支持。主要贡献包括：首先，构建了涵盖复杂场景的专用数据集，并通过创新训练策略提升深度学习模型的鲁棒性，实现高精度机器人检测；其次，提出投影定位法，提供精确三维坐标；最后，自研匹配跟踪系统，确保遮挡或快速移动下稳定的身份识别与轨迹跟踪。我们在 Robomaster 2025 复活赛中验证了此雷达站算法在真实场景中的优异表现，最大程度发挥了雷达站在赛场中的辅助作用。

关键词：目标检测，投影，匹配，跟踪算法

*联系邮箱：zguobd@connect.ust.hk
†联系邮箱: szhoubx@connect.ust.hk

目录

图目录

1 简介综述

雷达站（Radar）作为一种较新的兵种，其技术实现仍有广阔的探索空间。随着裁判系统不再返回密集的兵种的标记状态（除非识别进度大于 >100%），雷达站的算法愈发依赖于传感器与目标检测的精度。目前主流的开源方案，如厦门理工学院的单目视觉方案 [1] 和哈尔滨工业大学（深圳）的多传感器后融合方案 [2]，均已展现出卓越的性能。然而，在实际应用中我们发现，这些方案的视觉处理部分大多直接依赖于双层神经网络的检测结果，缺乏完善的追踪算法，导致其在复杂多变的比赛环境下适应性有时略显不足。此外，引入激光雷达等额外传感器虽能提升性能，但也带来了赛场真实数据采集的挑战，显著增加了算法的复杂度与调试难度。

因此, 针对以上所述等技术难点, 结合实际备赛情况, 我们决定保留单目相机方案, 并提出了以下几点创新的解决思路:

1. 我们创新性地提出了全新的投影方法，相较于厦门理工 [1] 多层透视变换的方案能够更加精准的定位目标。

2. 我们优化了东北大学 [3] 的三层神经网络识别方案, 将 *ResNet* 替换为更加轻量级的 *MobileNet*[4] 图案识别网络, 在维持识别性能的同时大大降低了时间复杂度。

3. 其次，我们利用官方直播图片以及各学校开源的雷达第一视角视频构建了一个针对 RoboMaster 比赛环境的高质量数据集，涵盖了多种复杂场景，包括不同光照条件、遮挡情况以及机器人形态变化。通过结合创新的模型训练策略，该数据集显著提升了基于深度学习的目标检测模型的鲁棒性和泛化能力，使其能够在复杂环境中实现高精度的机器人识别。

4. 最后，我们自主研发了一套基于匈牙利匹配算法的匹配与跟踪系统，确保在机器人快速移动或者装甲板被遮挡导致无法识别的情况下仍能维持稳定的机器人身份识别和流畅的轨迹跟踪。

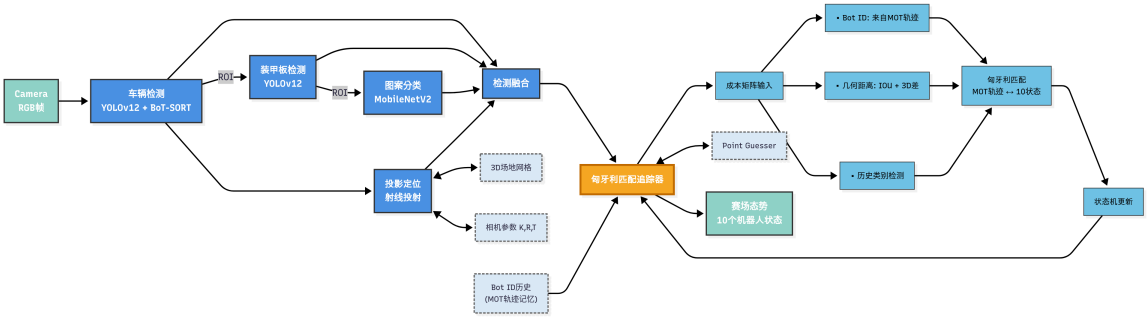

图 1: 雷达站算法概览。整个流程可分为检测算法以及追踪算法, 其中检测 (*Detector*) 使用三层神经网络以及 *BOT-Sort* 多目标 (*MOT*) 追踪算法提取场上的机器人轨迹以及装甲板信息; 追踪算法 (*Tracker*) 则使用了匈牙利匹配 (*Kuhn-Munkres Algorithm*) 来为每个机器人检测状态分配检测结果, 同时负责维护更新相应的机器人综合追踪状态。

2 实现方法

2.1 数据集制作

我们的数据集来源主要为官方公布的图片（见附录），以及各个学校开源的雷达站视角。对于每一张图片，均可以进行两层图片的标注，即：基于原始图片标注机器人、基于机器人标注装甲板。而最后标注出来的装甲板部分，则可以进行裁剪 (*crop*) 以适用于图案分类任务。

图 2: 我们从官方的图片直播中获取了大量图片带机器人遮挡的原始数据并进行了人工标注。相比于在训练时候引入遮挡 (如随机 mask 掉一部分) 等数据增强, 我们的实验表明这这种天然的遮挡训练集能大大增加模型对赛场复杂情况下的适应性。

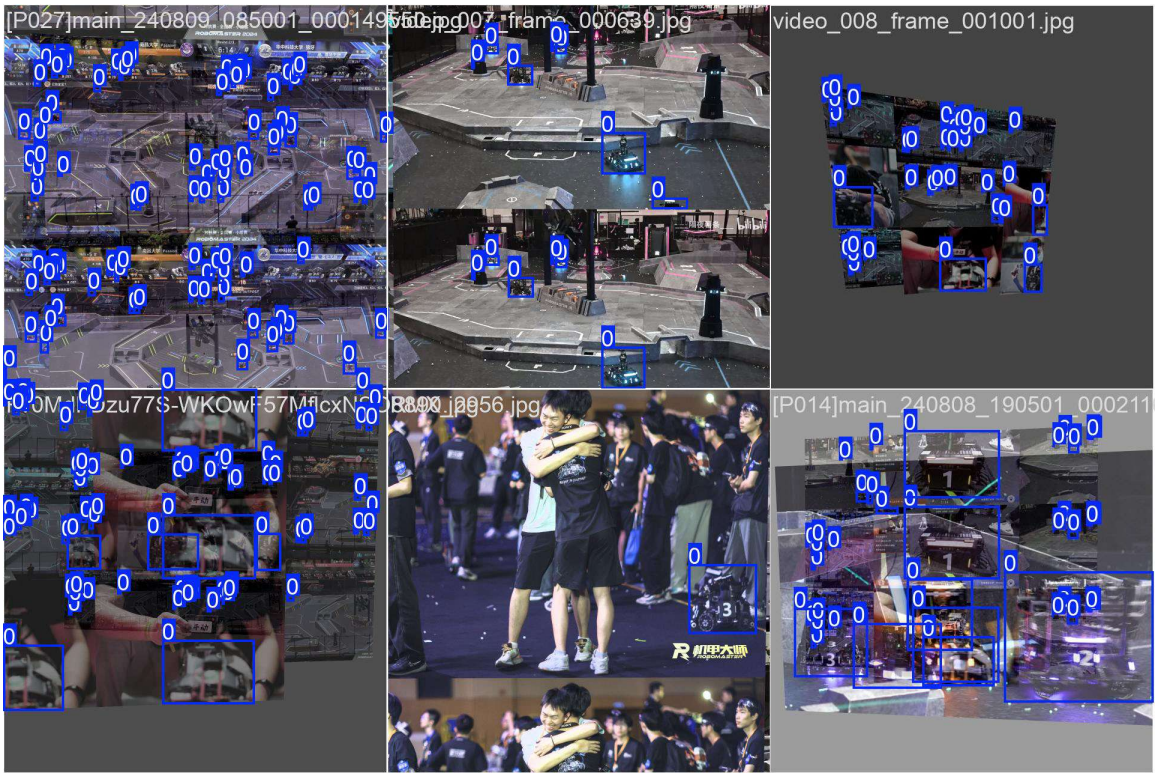

图 3: 我们的数据集还涵盖了多个学校开源的第一视角视频、官方比赛视频回放以及在 RM 社区开源的训练集。对于前两者, 我们抽帧选择性地进行了人工标注; 对于后者, 我们手动对数据进行了检查与筛选, 确保数据集的质量。

为了提高标注效率，我们参考了开源标注工具 *LabelMe* [5]，基于 **Qt** 框架开发了自定义的标注工具，并集成了**自动标注功能**。该方案采用模型与人工协同的交互式标注策略：初始阶段由人工标注少量样本以训练初步模型；随后，利用该模型对未标注数据进行预标注，再由人工进行校正与完善。标注完成的数据被加入训练集，用于迭代优化模型性能。这一过程形成了“模型辅助标注 → 人工修正 → 数据增益 → 模型提升”的正反馈循环，显著提升了标注效率与数据质量, 用极少的人力在短期内生产了大量高质量标注数据。

图 4: 自研标注工具快速标注

对于每一层神经网络的训练，我们均收集了大概 2-3 万张图片作为训练集，实战效果可以在准备阶段就有概率识别到对方在基地的五辆车。

2.2 Detector 检测网络

我们的目标检测分为三层：

- **基于相机图像检测机器人**：使用了更高版本的模型 **YOLO(You Only Look Once) v12**[6]，相较于更早的版本 v5、v8 等，更高版本的 YOLO 因为注意力机制的改进，对于更小的物体有更高精度的检测 [7]，这也与我们的使用场景契合。具体模型的大小我们使用了 s 模型，模型输入大小扩大为 1280×1280, 提高对远处场景 (敌方基地) 车辆识别的准确性。

- **基于机器人图像检测装甲板**：使用了 **YOLO v12**，模型大小使用 n 模型，输入大小为 192×192, 输出装甲板的实际锚框以及其颜色和存活状态 *(蓝; 红; 死亡)*。

- **基于装甲板图像进行分类**：为了保证更快的推断速度，我们使用了 **MobileNet-v2**[4] 这种轻量级网络，图像输入尺寸为 64×64。对于装甲板图案分类这样的

简单任务，可以轻松达到 99.99% 以上的准确率。

2.3 投影位置计算

为了将相机图像中的二维像素坐标 (u, v) 转换为场地中的三维世界坐标 (X_w, Y_w, Z_w)，本系统采用了一种基于**射线投射 (Ray Casting)** 的几何投影方法。该方法物理意义明确，精度高，且能自然地处理非平面场地的定位问题。

整个计算流程分为以下几个步骤：

2.3.1 相机成像模型与坐标系

首先，我们定义相关坐标系：

- **世界坐标系 (World Coordinate System)**: 我们以官方提供的场地模型内部的坐标系为准定义世界坐标系。
- **相机坐标系 (Camera Coordinate System)**: 以相机光心为原点，Z_c 轴沿光轴方向（指向视野前方），X_c 和 Y_c 轴与成像平面平行。
- **图像坐标系 (Image Coordinate System)**: 位于成像平面上，原点在图像左上角，单位为像素。

相机的成像过程可以用一个**针孔相机模型**来描述，并通过一个 3×4 的投影矩阵 $\mathbf{P}$ 将世界坐标 $\mathbf{X}_w = [X_w, Y_w, Z_w, 1]^T$ 映射到图像坐标 $\mathbf{x} = [u, v, 1]^T$：

$$\lambda \mathbf{x} = \mathbf{P}\mathbf{X}_w \tag{1}$$

其中，λ 是一个尺度因子，投影矩阵 $\mathbf{P}$ 可以分解为内参矩阵 $\mathbf{K}$ 和外参矩阵 $[\mathbf{R}|\mathbf{T}]$：

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{T}] \tag{2}$$

- $\mathbf{K}$ 是 3×3 的内参矩阵，包含焦距 f_x, f_y 和主点坐标 c_x, c_y。
- $\mathbf{R}$ 是 3×3 的旋转矩阵，$\mathbf{T}$ 是 3×1 的平移向量，共同描述了世界坐标系到相机坐标系的变换。

2.3.2 射线投射与求交

对于一个无畸变的像素点 $\mathbf{x} = [u, v, 1]^T$，其在相机坐标系下的方向向量 $\mathbf{d}_c$ 可以通过相机内参矩阵 $\mathbf{K}$ 的逆矩阵求得：

$$\mathbf{d}_c = \mathbf{K}^{-1}\mathbf{x} \tag{3}$$

将射线的原点（相机光心）和方向从相机坐标系转换到世界坐标系：

$$\mathbf{O}_w = -\mathbf{R}^T\mathbf{T} \tag{4}$$

$$\mathbf{d}_w = \mathbf{R}^T\mathbf{d}_c \tag{5}$$

其中 $\mathbf{O}_w$ 是相机光心，$\mathbf{d}_w$ 是射线方向。

本系统预先加载了一个经过**预处理**的官方场地 3D 三角网格模型。射线投射的核心是计算从 $\mathbf{O}_w$ 出发，沿 $\mathbf{d}_w$ 方向的射线与这个 3D 网格的交点。设射线上任意一点 $\mathbf{P}(t)$ 的参数方程为：

$$\mathbf{P}(t) = \mathbf{O}_w + t \cdot \mathbf{d}_w, \quad t > 0 \tag{6}$$

使用 Open3D 等高效的射线投射引擎，利用八叉树 (Octree) 求解该射线与场地网格的第一个交点，返回距离参数 t_{hit}。如果射线与网格相交（即 $t_{hit} < \infty$），则交点的 3D 世界坐标 $\mathbf{X}_w$ 为：

$$\mathbf{X}_w = \mathbf{O}_w + t_{hit} \cdot \mathbf{d}_w \tag{7}$$

此坐标 (X_w, Y_w, Z_w) 即为图像中像素点 (u, v) 在真实场地中的精确位置。

总结：该方法通过精确的相机标定和场地建模，将一个复杂的 2D 到 3D 映射问题，转化为一个清晰的几何求交问题，从而实现了高精度、鲁棒的定位。

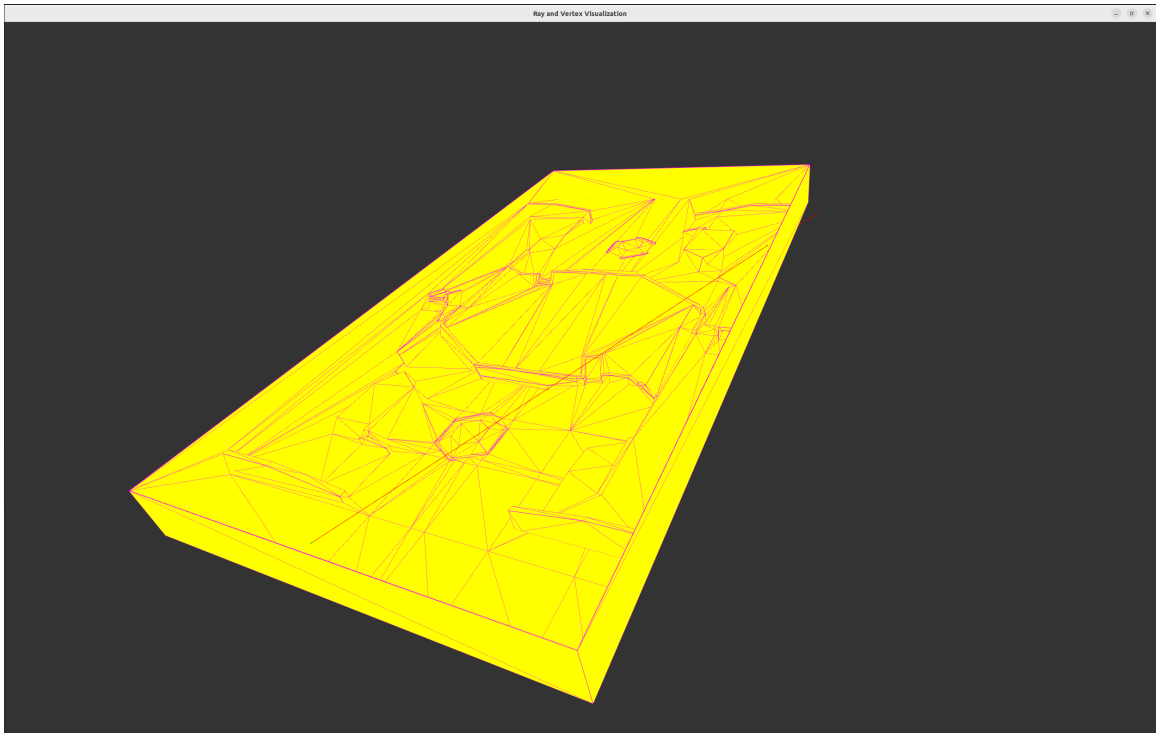

图 5: 射线投影法 (Ray-cast) 的示意图。可以看到从相机光心 (绿点) 处射出一条红色射线, 与场地相交于蓝点处, 该点即为机器人所在的场地坐标。

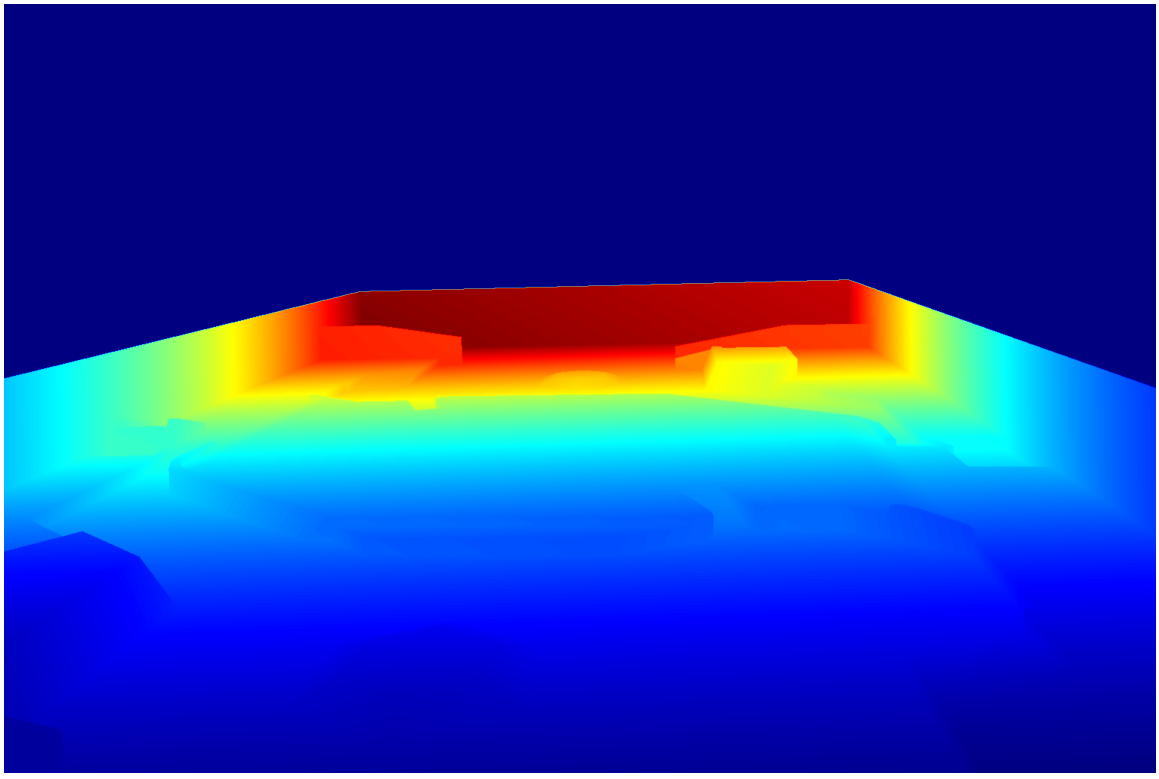

图 6: 通过对每个像素使用射线投射法计算投射距离, 我们可以渲染出在相机第一视角下的深度图。

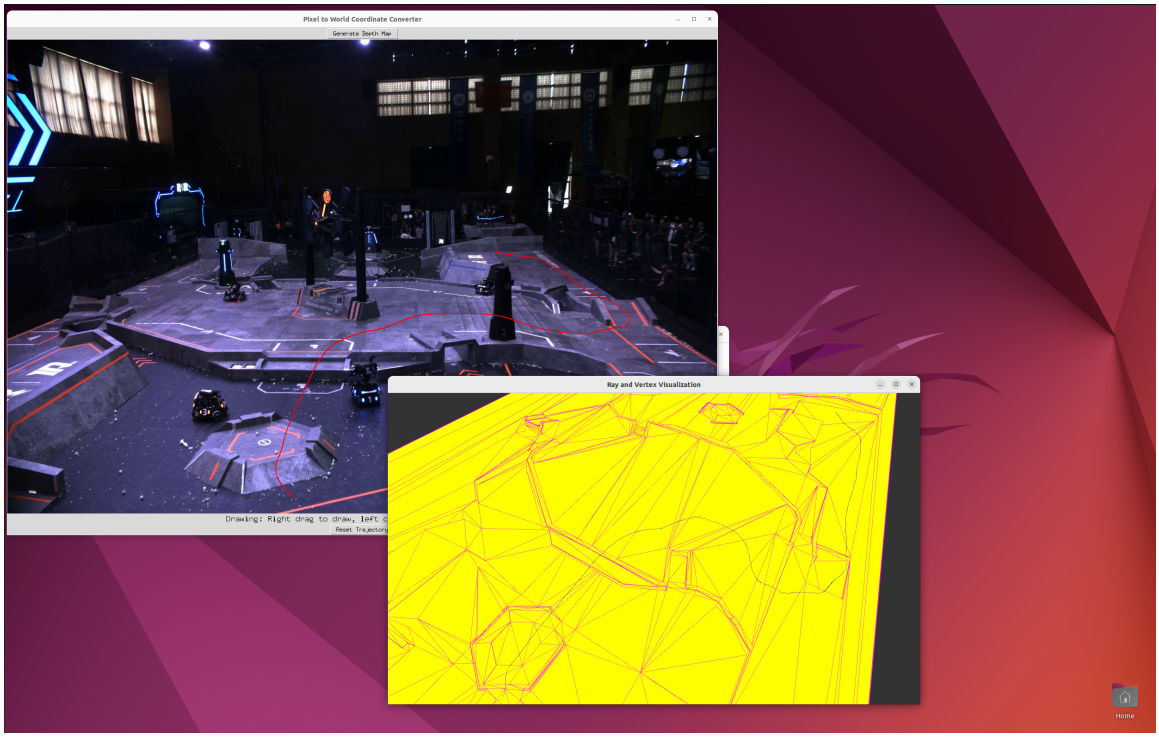

图 7: 射线投射投影变换效果展示。该投影算法可以将像素空间中绘制的一条轨迹 (红色) 较为精确的转化到场地世界坐标下的轨迹 (蓝色)。

2.3.3 标定算法

为了实现相机与场地之间的精确空间对齐，我们采用 **PnP（Perspective-n-Point）** 问题求解方法，具体使用 OpenCV 中的 *solvePnP* 算法来估计相机的外参，即旋转矩阵 **R** 和平移向量 **T** 。标定过程中，我们在场地的三维模型中选取了 6 个具有明确几何意义的关键点（如角落点或对称结构点），这些点在物理空间中的三维坐标已知且精确。当相机固定就位后，我们在采集的图像中手动标定这 6 个点对应的二维像素坐标。

利用这组 3D-2D 点对匹配关系，*solvePnP* 通过最小化重投影误差，高效求解出从世界坐标系到相机坐标系的刚体变换参数 $(\mathbf{R}, \mathbf{T})$ 。选用 6 个非共面点能够在保证计算稳定性的同时满足 PnP 问题的最小解条件，有效避免了解的不确定性。标定完成后，我们通过重投影验证：将三维关键点经估计的 **R** 和 **T** 投影回图像平面，检查其与人工标注点的偏差，确保平均重投影误差小于 8 个像素，满足后续定位与测量的精度需求。

2.3.4 与传统透视变换方法的对比

在单目视觉定位领域，一种常见的方法是**透视变换 (Perspective Transformation)**。该方法通过在平面上选取四对已知的世界坐标和像素坐标，计算出一个 3×3 的单应性矩阵 (Homography Matrix) $\mathbf{H}$，从而实现平面内的 2D 到 2D 映射：

$$\lambda \mathbf{x} = \mathbf{H}\mathbf{X}_{2d} \tag{8}$$

其中 $\mathbf{X}_{2d} = [X_w, Y_w, 1]^T$ 是世界平面上的点，$\mathbf{x} = [u, v, 1]^T$ 是对应的像素点。

然而，RoboMaster 比赛场地并非一个单一的平面。场地内存在多层结构，如高地、台阶和斜坡。厦门理工学院的开源方案提出了一种改进的透视变换方法：将场地划分为多个水平层（例如，低地、中央高地、梯形高地），为每一层分别标定一个单应性矩阵。在实际定位时，系统会分别计算三次透视变换，从而推断当前机器人的所处的水平层，并应用该层的透视变换来计算其 2D 坐标。

尽管该方法在一定程度上解决了多层场地的问题，但它仍存在固有的局限性：

- **精度依赖于分层假设**：该方法假设每一层都是完美的平面。然而，实际的台阶和斜坡是三维的，简单的 2D 映射无法精确描述这些非平面区域的几何关系，导致定位误差。
- **存在模糊区域**：在不同层的交界处（如台阶边缘），机器人究竟属于哪一层存在模糊性。选择错误的变换矩阵会直接导致坐标计算错误。
- **无法获取真实高度 (Z 坐标)**：透视变换本质上是 2D 到 2D 的映射，无法提供目标在垂直方向上的真实高度信息。
- **标定繁琐**：每一层的标定总共需要四个点，如果标定三层则共计需要标定 12 个点，在紧张的三分钟准备时间里标定容易犯错。

相比之下，本系统提出的**射线投射法 (Ray-cast)** 具有显著优势：

- **物理模型精确**：射线投射法基于真实的 3D 几何模型，直接模拟了光线传播的物理过程。只要 3D 网格模型足够精确，计算出的交点坐标就是理论上最准确的解。

- **天然处理非平面**：无论是平坦的地面、垂直的台阶还是倾斜的坡道，只要它们被正确地建模到 3D 网格中，射线都能与之精确求交，无需进行复杂的分层和判断。

- **输出完整 3D 坐标**：该方法直接输出 (X_w, Y_w, Z_w) 三维坐标，高度信息可能可以被后续的追踪算法所利用。

- **适应性好, 操作简单**: 只需要导入特地场地模型便可以完美兼容该场地下的任何投影变换, 无需重新调整任何参数; 标定简单, 适合三分钟准备阶段操作。

综上所述, 射线投射法克服了传统透视变换在处理复杂 3D 场地时的局限性, 为雷达站提供了一种更精确、更鲁棒的定位解决方案。

2.4 Tracking 追踪算法

在 RoboMaster 机器人视觉系统中，准确的身份识别与稳定的目标追踪是实现战术决策与自动瞄准的关键基础。然而，现有开源方案大多停留在“单帧检测”层面，仅依赖两阶段检测流程：第一阶段检测机器人整体（车体），第二阶段识别其装甲板上的红蓝标识以判断阵营与兵种。这类方法虽在理想条件下表现良好，但在实际比赛中面临诸多挑战。

现有开源工作的局限性　目前主流的开源实现普遍未引入多目标追踪（MOT）机制，而是直接对每一帧的检测结果进行独立分类与输出。这种“帧间无关”的处理方式存在明显缺陷：

- **缺乏时序一致性**：由于未维护目标的身份记忆，同一机器人在不同帧中可能被赋予不同的检测结果，导致轨迹跳变、身份混淆；且我们注意到现有工作大多无法很好处理检测器在某一帧检测到多个相同的装甲板 ID 的情况.

- **对低置信度检测敏感**：无法处理装甲板被遮挡的情况；

- **无法处理死亡状态**：机器人被击毁后，其装甲板类别可能变为“死亡”状态，但现有工作未能很好通过检测来区分机器人“存活”与“死亡”的状态，也未将此信息用于身份维持；

- **无运动建模**：缺乏卡尔曼滤波等运动预测机制，无法在检测丢失时进行合理外推，导致轨迹频繁中断。

这些限制使得现有单目视觉方案在高动态、强对抗的比赛场景中鲁棒性较差。

本系统的追踪算法设计 为克服上述问题，本系统自主研发了一套基于匹配机制的多目标追踪算法，并结合 RoboMaster 比赛的特定先验知识进行了创新性改进。

2.4.1 与传统追踪算法的对比与借鉴

DeepSORT[8]、ByteTrack[9] 和 BotSORT[10] 是当前多目标追踪（MOT）领域的主流算法，均遵循“检测-追踪”范式：首先通过目标检测器获取每帧中的边界框，随后利用运动模型（如卡尔曼滤波）预测已有轨迹的状态，并通过数据关联将检测结果与现有轨迹进行匹配。匹配过程通常基于构建代价矩阵（融合 IoU、外观特征或置信度等信息），并采用匈牙利算法求解最优一一对应关系，避免重复匹配。

在系统设计中，我们充分借鉴了这些成熟算法的核心思想：

1. **运动模型的继承**：我们采用二维卡尔曼滤波器对每个机器人轨迹进行状态预测与更新，用以估计其在图像中的边界框位置。该设计与 DeepSORT[8] 和 BotSORT[10] 一致，能够有效平滑轨迹、抑制检测噪声与短暂抖动。

2. **匹配框架的沿用**：我们同样采用匈牙利算法作为数据关联的核心求解器，在检测轨迹（*bot-track*）与预设机器人状态之间寻找全局最优匹配，确保匹配的唯一性与稳定性。

- **关于 *bot_id* 的来源与利用**：本系统中使用的 *bot_id* 并非由我们自行生成，而是借助现成的多目标追踪算法（如 Bot-SORT）在检测序列上运行所输出的短期 track ID。这些 ID 本身融合了外观特征（ReID 向量）与运动信息，具备一定的跨帧一致性。我们将其视为一种高置信度的短期身份线索，作为匹配过程中的辅助依据。在实际运用中，我们可以通过调用 *Ultralytics YOLO Model* 的 *track* 方法来进行 *MOT* 追踪与 *bot_id* 提取。

然而，与通用 MOT 算法不同，我们的应用场景具有**强先验约束**——比赛双方共 **10 个固定兵种**（红蓝各 5 类），且在整个对战过程中兵种数量保持不变。这一关键前提使我们能够对追踪范式进行根本性重构：

- **静态匹配结构**：传统 MOT 算法需动态管理轨迹的初始化、确认、丢失与删除，以应对目标的出现与消失。而我们的系统预先维护一个包含 10 个机器人状态的固定列表 $\mathcal{S}$，每一帧仅需将当前检测到的 *bot-track* 与这 10 个预设状态进行匹配，无需创建新轨迹或销毁旧轨迹。这不仅简化了系统逻辑，也显著提升了匹配的鲁棒性与可解释性。

- **"软外观模型"的创新**：尽管我们借用了现成追踪算法提供的 *bot_id* 作为短期身份线索，我们额外提出了一种基于**多源信息融合**的"软外观模型"，将身份匹配问题转化为一个综合置信度评估过程。具体而言，对于第 i 个机器人状态与第 j 个 *bot-track* 的匹配，我们不仅利用 *bot_id* 对应的装甲板检测历史类别分布，还融合了多种上下文信息，包括时空一致性（如 3D 位置距离）、几何重叠度（如 IoU）、以及 *bot_id* 与机器人状态上一帧的关联状态，构建其综合匹配置信度。

综上所述，我们并非完全摒弃传统追踪算法的成果，而是在其基础上，结合任务先验知识，构建了一个面向固定数量目标的级联匹配追踪框架。该框架既吸收了 DeepSORT 等算法在运动建模与数据关联上的成熟经验，又通过静态匹配结构与基于历史类别的"软外观"建模，实现了在特定场景下的高效、稳定追踪。

2.4.2 匈牙利匹配算法

匈牙利算法（Hungarian Algorithm），又称库恩-曼克莱斯算法（Kuhn-Munkres Algorithm），是解决二**分图最优匹配**问题的经典算法。在多目标追踪中，它被用来解决"将哪些检测框分配给哪些追踪轨迹"的问题。

该问题可以被建模为一个成本矩阵 $\mathbf{C}$，其中 C_{ij} 表示将第 i 个轨迹与第 j 个检测结果进行匹配的成本。匈牙利算法的目标是找到一组匹配对，使得所有匹配对的成本总和最小。该算法的时间复杂度为 $O(N^3)$，其中 N 是轨迹和检测结果中数量较多者。它能保证找到全局最优解，是多目标追踪中实现数据关联的可靠方法。

2.4.3 追踪状态定义

本系统为每个机器人定义了四种追踪状态，构成一个状态机，用于管理其生命周期：

- *INACTIVE* (非活跃): 机器人的初始状态或已丢失的状态。此状态下的机器人不参与运动预测。
- *TENTATIVE* (暂定): 机器人已被初步检测到，但尚未得到充分确认。需要连续命中几次才能转为已确认状态。
- *CONFIRMED* (已确认): 机器人已被稳定追踪，是系统中最可靠的状态。此状态下的机器人会进行运动预测，并参与高优先级的匹配。
- *LOST* (丢失): 机器人曾被确认，但已连续多帧未匹配成功。系统仍会对其进行预测，希望能在后续帧中重新捕获。若持续丢失，则会降级为 *INACTIVE*。

状态间的转换由匹配结果和计数器（*hit_count*, *miss_count*）控制。

2.4.4 匈牙利匹配追踪器

我们的匈牙利匹配追踪器将运动信息与创新的“历史类别置信度”相结合，构建了一个更适应 RoboMaster 场景的多维度匹配系统。其核心流程如下：

Algorithm 1: 匈牙利匹配追踪算法

Require: 输入图像序列 $I_{1:T}$, 阵营信息 F

Ensure: 更新后的机器人状态列表 $\mathcal{S}$

▷ *Step 1: 初始化*

for all $i, \texttt{state}_i \in \mathcal{S}$ **do**
    $\texttt{state}_i\texttt{.state} \leftarrow \mathsf{INACTIVE}$
    $\texttt{state}_i\texttt{.hit\_count} \leftarrow 0$
    $\texttt{state}_i\texttt{.miss\_count} \leftarrow 0$
    $\texttt{state}_i\texttt{.id} \leftarrow i$
    InitKalmanFilter($\texttt{state}_i\texttt{.kalman\_filter}$)
end for
$\texttt{detector} \leftarrow$ Detector()
$t \leftarrow 0$
while $t < T$ **do** ▷ *Step 2: 运行检测*

$\mathtt{D}_t \leftarrow \textsc{RunDetection}(\mathtt{detector}, I_t)$ ▷ 维护 *bot_id* 的类别历史

for all $d_j \in \mathtt{D}_t$ **do**

if $d_j.\mathtt{bot\_id} \notin \mathtt{bot\_id\_history}$ **then**

$\mathtt{bot\_id\_history}[d_j.\mathtt{bot\_id}] \leftarrow \textsc{NewBotIdTrack}(d_j.\mathtt{bot\_id})$

end if

$\textsc{Update}(\mathtt{bot\_id\_history}[d_j.\mathtt{bot\_id}], d_j.\mathtt{class\_id})$

end for

$\textsc{RemoveTimeoutBotIds}(\mathtt{bot\_id\_history})$

▷ *Step 3:* 状态预测（仅 *CONFIRMED* 和 *LOST*）

for all $\mathtt{state} \in \mathcal{S}$ **do**

if $\mathtt{state.state} \in \{\mathsf{CONFIRMED}, \mathsf{LOST}\}$ **then**

$\mathtt{state.car\_box} \leftarrow \textsc{Predict}(\mathtt{state.kalman\_filter})$

$\mathtt{state.pos\_3d} \leftarrow \textsc{PixelToWorld}(\mathtt{state.car\_box})$

end if

end for

▷ *Step 4:* 构建成本矩阵

$N \leftarrow |\mathtt{D}_t|, M \leftarrow |\mathcal{S}|$

$\mathtt{C} \leftarrow \mathbf{0}_{M \times N}$ ▷ 初始化成本矩阵

for all $i, \mathtt{state}_i \in \mathcal{S}$ **do**

for all $j, d_j \in \mathtt{D}_t$ **do**

$\mathtt{IOU}_{ij} \leftarrow \textsc{ComputeIOU}(\mathtt{state}_i, d_j)$

$\mathtt{HistConf}_{ij} \leftarrow \textsc{GetConfidence}(\mathtt{history[d\_j.bot\_id]}, \mathtt{state}_i.\mathtt{id})$

$$\mathtt{BotID}_{ij} \leftarrow \begin{cases} 1.0 & \text{if } \mathtt{state}_i.\mathtt{bot\_id} = d_j.\mathtt{bot\_id} \\ 0.0 & \text{otherwise} \end{cases}$$

$\mathtt{pos\_dist} \leftarrow \|\mathtt{state}_i.\mathtt{pos\_3d} - d_j.\mathtt{pos\_3d}\|$

$\mathtt{max\_field\_dist} \leftarrow \mathsf{MAX\_FIELD\_DIST}$ ▷ 场地最大距离，可预计算

$\mathtt{PosScore}_{ij} \leftarrow \max\left(1.0 - \frac{W_4 \cdot \mathtt{pos\_dist}}{\mathtt{max\_field\_dist}}, -1.0\right)$

$\mathtt{S}_{ij} \leftarrow W_1 \cdot \mathtt{HistConf}_{ij} + W_2 \cdot \mathtt{IOU}_{ij} + W_3 \cdot \mathtt{BotID}_{ij} + \mathtt{PosScore}_{ij}$

```
        end for
    end for
                                        ▷ Step 5: 匈牙利匹配与低置信度匹配过滤
    matches_temp ← HUNGARIAN(C)
    final_matches ← ∅
    for all (i, j) ∈ matches_temp do
        if state_i.state = INACTIVE then
            threshold ← INACTIVE_COST_THRESHOLD
        else
            threshold ← COST_THRESHOLD
        end if
        if C_ij < threshold then
            APPEND(final_matches, (i, j))
        end if
    end for
                                                    ▷ Step 6: 状态更新
    for all (i, j) ∈ final_matches do
        state ← state_i, det ← d_j
        if state.state = INACTIVE then
            state.state ← TENTATIVE
            state.hit_count ← 0
        else if state.state = TENTATIVE then
            state.hit_count ← state.hit_count + 1
            if state.hit_count ≥ HIT_COUNT_THRESHOLD then
                state.state ← CONFIRMED
                RESETFILTER(state.kalman_filter, det.car_box)
            end if
        else if state.state ∈ {CONFIRMED, LOST} then
            state.bot_id ← det.bot_id
```

```
            if state.bot_id ≠ det.bot_id then
                ResetFilter(state.kalman_filter, det.car_box) ▷ 处理 ID 跳跃
            else
                UpdateFilter(state.kalman_filter, det.car_box)
            end if
            state.state ← CONFIRMED, state.miss_count ← 0
        end if
    end for
    for all state ∈ S \ final_matches do
        if state.state = TENTATIVE then
            state.hit_count ← state.hit_count − 1
            if state.hit_count ≤ 0 then
                state.state ← INACTIVE
            end if
        else if state.state = CONFIRMED then
            state.state ← LOST, state.miss_count ← 0
        else if state.state = LOST then
            state.miss_count ← state.miss_count + 1
            if state.miss_count ≥ MISS_COUNT_THRESHOLD then
                state.state ← INACTIVE
                state.bot_id ← −1
                ResetFilter(state.kalman_filter)
            end if
        end if
    end for
    t ← t + 1
end while
return S
```

为更清晰地理解该算法，我们对其关键组件进行分块阐述。

1. 检测器输出与数据结构定义 本系统依赖一个融合检测与短期追踪的 *MOT* 级联检测器，其每帧输出为一组 *bot-track* 级别的检测结果。每个检测结果由 *SingleDetectionResult* 类封装，包含以下关键字段：

- *class_id*: 装甲板识别类别;
    1. **-1**: 当前检测框内没有发现装甲板。
    2. **0-4**: 代表 B1, B2, R3, R4, R7。
    3. **5-9**: 代表 R1, R2, R3, R4, R7。
    4. **10-14**: 代表阵亡的机器人 1,2,3,4,7 号。
- *car_box*: 机器人整体边界框（$[x1, y1, x2, y2]$），用于运动建模;
- *pos_3d*: 通过相机标定与几何模型估计的机器人在世界坐标系下的 3D 位置;
- *bot_id*: 由外部追踪器（如 DeepSORT）赋予的短期轨迹 ID，具备一定跨帧一致性。

这些信息共同构成了数据关联的基础输入。

系统为每个 *bot_id(机器人轨迹)* 维护一个 *BotIdTrack* 实例，用于记录其历史类别分布与置信度演化。该类通过滑动窗口（默认长度 30）维护 *class_id(机器人兵种类别)* 的时间序列，并提供 *get_class_id_exponent_confidence* 方法，计算加权历史类别分布：

$$\text{HistConf}_{ij} = \text{GetConfidence}(j, i), \tag{9}$$

其中权重采用指数衰减（衰减因子 $\tau = 0.5$），近期帧影响更大。此外，该机制将 “死亡” 状态（*class_id* 10–14）映射回对应 “存活” 类别（如对阵亡 1 号机器人的检测仍给予其较高与兵种状态 R1 与 B1 的匹配权重），并为未知类别（-1）均匀分配置信度

2. 机器人状态管理 系统预先初始化 10 个 *TrackingState* 实例 *(以下都简称**”状态”**)*，对应红蓝双方共 10 个固定兵种。每个状态维护以下信息：

- *state*: 状态机（*INACTIVE*, *TENTATIVE*, *CONFIRMED*, *LOST*）；
- *kalman_filter*: 基于边界框的二维卡尔曼滤波器，用于预测与更新位置；
- *bot_id*: 当前关联的短期轨迹 ID，用于跨帧身份一致性判断；
- *pos_3d*: 从 *car_box* 通过 *pixel_to_world*(**2.3** 所论述的投影变换) 转换得到的世界坐标；
- *hit_count* / *miss_count*: 用于状态跃迁的计数器（实际运用阈值分别为 3 和 10）。

3. 匹配逻辑与代价构建 算法的核心在于构建一个融合多维度线索的匹配代价矩阵 $C \in \mathbb{R}^{M\times N}$，其中 $M = 10$ 为机器人状态数，$N = |D_t|$ 为当前帧检测数。

每项代价 C_{ij} 由综合得分 S_{ij} 取负得到：

$$C_{ij} = -S_{ij} \tag{10}$$

$$S_{ij} = W_1 \cdot \mathrm{HistConf}_{ij} + W_2 \cdot \mathrm{IoU}_{ij} + W_3 \cdot \mathbb{I}(b_i = b_j) + \max(1.0 - \| \frac{W_4 \|\mathbf{p}_i - \mathbf{q}_j\|}{\max_{p,q\in Fields} \|\mathbf{p} - \mathbf{q}\|}, -1.0) \tag{11}$$

其中各项含义如下：

- $\mathrm{HistConf}_{ij}$：*BotIdTrack* 提供的历史类别匹配置信度，作为“软外观”模型；
- IoU_{ij}：预测边界框与检测框的交并比，衡量几何一致性；
- $\mathbb{I}(b_i = b_j)$：指示函数，若当前状态的 *bot_id* 与检测一致则为 1，否则为 0；
- $\|\mathbf{p}_i - \mathbf{q}_j\|$：预测 3D 位置 $\mathbf{p}_i$ 与检测 3D 位置 $\mathbf{q}_j$ 的欧氏距离，体现时空连续性。

该加权融合机制使系统在装甲板无法稳定识别时仍能通过多线索协同完成稳定匹配。

4. 匈牙利匹配与状态更新 匹配过程采用匈牙利算法求解最优一一对应。为控制误匹配，引入动态阈值过滤机制：

- 若状态为 *CONFIRMED/LOST*，由于我们可以认为在前几帧存在成功匹配的兵种有较大概率在这一帧也有匹配, 因此可以使用较宽松的 *COST_THRESHOLD*；

- 若为 *INACTIVE* 或 *TENTATIVE*, 使用更严格的 *INACTIVE_COST_THRESHOLD*, 以防在没有足够确信度的情况下进行匹配。

成功匹配的状态的处理与状态机更新:

- *INACTIVE* → *TENTATIVE*：启动身份确认流程，重置 *hit_count*；
- *TENTATIVE*：*hit_count* 递增，达到 *HIT_COUNT_THRESHOLD*（默认 3）后转为 *CONFIRMED*；
- *CONFIRMED*/*LOST*：更新 *bot_id* 与基于边界框的卡尔曼滤波器；若 *bot_id* 发生跳变，则重置滤波器以应对身份漂移；

未匹配状态的处理与状态机更新:

- *TENTATIVE*：*hit_count* 减 1，归零后转为 *INACTIVE*；
- *CONFIRMED*：转为 *LOST*，*miss_count* 归零；
- *LOST*：*miss_count* 递增，超限（默认 10）后转为 *INACTIVE* 并重置状态。

综上所述，本追踪器通过固定目标池、多源信息融合与状态机控制，在 RoboMaster 这类目标数量固定、遮挡频繁的场景中实现了高鲁棒性的身份维持与轨迹平滑。

2.4.5 猜点逻辑

我们的猜点得分思路借鉴了成都大学 Ultra 战队的开源思路 [11]，使用了速度的和到目标位置矢量的**余弦相似度 (Cosine Similarity)**，以及指数级衰减的**欧式距离 (Euclidean Distance)** 得分。

相较于成都大学使用最后两帧坐标来预测速度，我们直接对机器人基于地图上的二维坐标进行卡尔曼滤波（不同于前面基于检测框的卡尔曼滤波，以下算法将使用 $kalman\_filter\_2d$ 指代），使用跟踪的速度能够一定程度上增加鲁棒性。而剩余大部分内容与成都大学开源思路相同，在此放出融合了基于地图二维坐标的卡尔曼滤波的伪代码：

Algorithm 2: 基于运动趋势和距离的猜点预测算法

Require: 丢失的追踪状态 `track`, 阵营颜色 `ref_color`

Ensure: 最优猜测点 `best_point`

▷ *Step 1:* 获取兵种对应的候选猜测点

`robot_name` ← MapNameById(`name_id_convert`, `track.class_id`)

`guess_points` ← LoadGuessPoints(`robot_name`)

if `ref_color` = BLUE **then**

  `center_x`, `center_y` ← 14.0, 7.5 ▷ 场地中心坐标

  `mirrored_points` ← ∅

  for all $(x, y) \in$ `guess_points` **do**

    `mirrored_x` ← $2 \times$ `center_x` $- x$

    `mirrored_y` ← $2 \times$ `center_y` $- y$

    Add(`mirrored_points`, (`mirrored_x`, `mirrored_y`))

  end for

  `guess_points` ← `mirrored_points`

end if

▷ *Step 2:* 获取卡尔曼滤波器状态

`pos_2d`, `vel_2d` ← GetState(`track.kalman_filter_2d`)

`last_pos` ← (`pos_2d`[0], `pos_2d`[1])

`v_vec` ← (`vel_2d`[0], `vel_2d`[1])

▷ *Step 3:* 计算每个候选点的评分

`scores` ← ∅

for all `point` $\in$ `guess_points` **do** ▷ 计算位置向量

  `d_vector` ← (`point`[0] − `last_pos`[0], `point`[1] − `last_pos`[1])

▷ 计算余弦相似度

  `dot_product` ← `v_vec`[0] · `d_vector`[0] + `v_vec`[1] · `d_vector`[1]

  `v_norm` ← $\sqrt{\texttt{v\_vec}[0]^2 + \texttt{v\_vec}[1]^2}$

  `d_norm` ← $\sqrt{\texttt{d\_vector}[0]^2 + \texttt{d\_vector}[1]^2}$

  `cos_sim` ← $\frac{\texttt{dot\_product}}{\texttt{v\_norm} \cdot \texttt{d\_norm} + \epsilon}$ ▷ $\epsilon = 10^{-8}$, 避免除零

▷ 计算距离评分

distance ← d_norm

d_score ← exp(−distance · D_FACTOR)

▷ 综合评分

score ← COS_FACTOR · cos_sim + (1 − COS_FACTOR) · d_score

APPEND(scores, (point, score))

end for

▷ *Step 4:* 选择最高评分的点

SORTDESCENDING(scores, by key: pair ↦ pair[1])

best_point ← scores[0][0]

return best_point

需要注意的是，由于我们预先假设了我们为红方，因此预先输入的盲点均为己方为红方的考虑情况，如果己方为蓝方（对应 $ref\_color ==' blue'$），则需要绕场地中心进行镜像操作。

下一步便是与上面的匹配机制进行融合。很直观的思路便是当目标状态为超出了最大的 **LOST 阈值 (LOST Threshold)** 时，将状态转化为 INACTIVE 并且启用猜点。

LOST 阈值（MISS_COUNT_THRESHOLD）触发时，将状态转为 INACTIVE 并启用猜点机制。

Algorithm 3: 猜点激活与状态重置逻辑

Require: 处于 LOST 状态且未匹配的追踪对象 track

Ensure: 激活猜点机制并重置追踪状态

if track.state = LOST **then**

track.miss_count ← track.miss_count + 1

if track.miss_count ≥ MISS_COUNT_THRESHOLD **then** ▷ 默认阈值为 5

▷ *Step 1:* 确定阵营颜色用于镜像

$$ref\_color \leftarrow \begin{cases} \text{BLUE} & \text{if } \texttt{faction} = \text{FACTION.BLUE} \\ \text{RED} & \text{otherwise} \end{cases}$$

▷ *Step 2:* 调用猜点算法生成预测位置

```
        guess_point ← PredictPoint(point_guesser, track, ref_color)
        track.guess_point ← guess_point
        track.is_start_guess ← TRUE
                                                ▷ Step 3: 重置身份关联信息
        track.bot_id ← −1                                  ▷ 解除 bot_id 绑定
                                                ▷ Step 4: 重置卡尔曼滤波器
        Reset(track.kalman_filter)
        Reset(track.kalman_filter_2d)
                                                ▷ Step 5: 更新追踪状态
        track.is_active ← FALSE
        track.state ← INACTIVE
    end if
end if
```

由于我们没有使用猜点超时的逻辑，因此退出猜点状态的逻辑为目标重新被检测到并激活，否则目标将一直处于被猜点的状态。

3 硬件要求

- **GPU 运算端**：推荐配置 RTX3060 以上。实测笔记本 RTX5070 和笔记本 RTX4060 均能完成约 10fps 的运算。
- **相机配置**：推荐分辨率较大的相机，如海康 cs200:

图 8: 海康 cs200 相机，分辨率 5472 × 3648

4 性能评估与展示

图 9: 最高单场 1912.1s 易伤

图 10: 局均易伤时间，局均额外伤害复活赛第一

5 总结与展望

5.1 项目总结

本项目成功开发了一套完整的 RoboMaster 2025 雷达站目标跟踪系统，在数据集构建、模型训练、算法设计和系统集成等方面取得了显著成果：

1. **高质量数据集构建**:
    - 从官方图片直播中提取并标注了大量真实比赛场景数据。
    - 针对遮挡等复杂场景进行专项标注。
    - 不仅标注机器人存活状态下的装甲板数据，还标注了死亡状态下的装甲板数据。

2. **先进模型架构**:

    - 成功部署 *YOLOv12* 作为核心检测引擎，相比旧版本 YOLO 提高对小物体的检测精度。
    - 使用 *MobileNet-v2* 作为额外的装甲板分类器, 显著降低了装甲板的误识别率。

3. **创新算法设计**:

    - 射线投影 3D 定位算法简洁高效。
    - 融合匈牙利匹配的追踪算法能够在装甲板被遮挡等复杂赛场条件下综合全局以及历史信息完成稳定的追踪。

5.2 未来展望

1. **模型性能提升**:

    - 引入更先进的 Transformer 架构，如 DETR 系列模型。
    - 探索多模态融合（视觉 + 激光雷达），提升复杂环境适应性。

2. **与哨兵信息的融合**:

    - 可以结合哨兵返回的己方机器人坐标，在匈牙利匹配对己方机器人引入额外的匹配权重，从而显著加强对方机器人匹配的准确度。

5.3 不足

在实际的测试中，我们的算法还是会有以下的缺陷:

- 由于依赖检测框来确定投影的位置，对于刚刚在高地后面的机器人会出现预测和实际位置有较大的偏差，接近于允许的最大偏差极限。
- 依赖标定点的准确度，同样的视频，不同的标定结果可能会导致不同的投影点，尤其是对于高地后的机器人，可能会在高低边缘和高低后面很大一部分距离波动。从而影响后续的追踪等一系列计算。
- 数据量的收集与模型的训练成本较大，数据准备时间较长，模型训练时间较久。对于资源短缺的小队伍可能需要更多的精力。

6 鸣谢

本研究的完成离不开众多个人与团队的无私支持与贡献，在此谨致以最诚挚的谢意。

首先，衷心感谢计算机视觉领域的开源社区，特别是 YOLO 项目团队以及卷积神经网络（CNN）技术的先驱者们。正是他们卓越的开创性工作，为本方案的深度学习模型奠定了坚实的基础。

特别感谢 *ENTERPRIZE* 战队的全体成员，是你们的默默支持与共同努力，为本套雷达站算法的研发提供了不可或缺的环境与动力。感谢薛旭朗（Jack）、钟梓洋和袁鸣煊在数据集的收集与标注工作中付出的巨大努力。同时，由衷感谢蒋益诚学长在技术方向上给予的宝贵指导。

我们还要向所有开源雷达站技术的团队致以崇高的敬意。感谢厦门理工 *PFA* 战队、哈尔滨工业大学（深圳）南工骁鹰战队、东北大学 *TDT* 战队以及成都大学 *Ultra* 战队的无私分享，我们从你们的杰出工作中汲取了丰富的灵感与经验，受益匪浅。

最后，感谢佛山大学醒狮战队以及港科大（广州）*PnX* 战队的同学们，你们不仅无偿提供了宝贵的区域赛赛场第一视角数据，更在交流中给予了我们深刻的启发。同时，也感谢成都大学 *Ultra* 战队、南理工 *Alliance* 等战队公开分享的雷达站第一视角视频，这些资源对本研究的验证与优化起到了关键作用。

7 后记

非常感谢各位对香港科技大学 ENTERPRIZE 战队开源项目的关注与支持，欢迎通过以下方式联系我们，强烈推荐加入官方开源交流 QQ 群：

- **作者邮箱**: zguobd@connect.ust.hk
- **作者微信**: guozilin200429
- **香港科技大学 ENTERPRIZE 战队开源交流 QQ 群**: 581184202
- **战队微信公众号**: HKUST ENTERPRIZE (ID: gh_855d709c046e)
- **战队 Bilibili 官号**: 港科大 ENTERPRIZE 战队 (UID: 634988052)

若您在机器人设计过程中受益于我们的项目，恳请在**开源引用中注明我们的项目来源**。你们的支持是我们不断前行的动力。让我们携手共建一个鼓励创新、协作与超越的 RM 开源生态。

参考文献

[1] 厦门理工学院 PFA 战队, *RM2024 赛季-单目相机雷达站视觉算法开源*, https://bbs.robomaster.com/article/43538?source=4, Accessed: 2024-09-11, 2024.

[2] 哈尔滨工业大学（深圳）南工骁鹰战队, *RM2024 雷达站算法开源*, https://bbs.robomaster.com/article/55223?source=4, Accessed: 2025-02-08, 2024.

[3] 东北大学 TDT 战队, *RM2024 雷达站算法开源*, https://github.com/T-DT-Algorithm-2024/T-DT-2024-Radar, Accessed: 2024-11-12, 2024.

[4] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, *Mobilenets: Efficient convolutional neural networks for mobile vision applications*, 2017. arXiv: 1704.04861 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1704.04861.

[5] K. Wada, *Labelme: Image polygonal annotation with python*, If you use this software, please cite it as below., 2021. DOI: 10.5281/zenodo.5711226. [Online]. Available: https://github.com/wkentaro/labelme.

[6] Y. Tian, Q. Ye, and D. Doermann, "Yolov12: Attention-centric real-time object detectors," *arXiv preprint arXiv:2502.12524*, 2025.

[7] Topiew, *YOLO v11 vs. YOLO v10 vs. YOLO v8: Which is Best for Object Detection on COCO, OBB-Dota V1 Dataset*, https://www.topview.ai/blog/detail/yolo-v11-vs-yolo-v10-vs-yolo-v8-which-is-best-for-object-detection-on-coco-obb-dota-v1-dataset, Accessed: 2025-04-05, 2024.

[8] N. Wojke and A. Bewley, "Deep cosine metric learning for person re-identification," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2018, pp. 748–756. DOI: 10.1109/WACV.2018.00087.

[9] Y. Zhang, P. Sun, Y. Jiang, *et al.*, "Bytetrack: Multi-object tracking by associating every detection box," 2022.

[10] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, "Bot-sort: Robust associations multi-pedestrian tracking," *arXiv preprint arXiv:2206.14651*, 2022.

[11] 成都大学 Ultra 战队, *RM2025-单目雷达站开源*, https://bbs.robomaster.com/article/759225?source=1, Accessed: 2025-04-05, 2025.

附录

- RM 赛事官方拍摄图片：https://bbs.robomaster.com/article/54980?source=5
- 2019 大疆开源目标检测数据集（包含机器人，装甲板）：https://bbs.robomaster.com/article/6751