2018 RoboMaster 夏令营

技术报告

第 三 组

2018.8

目　录

1. 机械部分

第一阶段组内的讨论结果为：
1、舍弃官方车辆，重新做可以 360° 旋转的云台，以躲避对方伤害；
2、夹子设计的夹取域量要尽可能的大，以降低视觉微调的精度。
第一阶段整车的设计结果如图 1 所示，云台如图 2 所示，yaw 轴采用官方下供弹的结构，滑环固定在下供弹出口处。

图 1 第一阶段三维图

图 2 360 云台

夹子的设计如图 3 所示，气缸实现夹取动作，电机实现翻转动作。气缸行程为 125，夹取域量足够大。夹子可以同时夹取地上以及 400 障碍物块的弹药箱（这里有个问题，夹取高处的弹药箱时，由于翻转角度不够，弹丸不能调入弹仓，第一阶段没有考虑到，第二阶段解决），结构简单，模块化设计，拆卸方便。

图 3 夹子整体设计

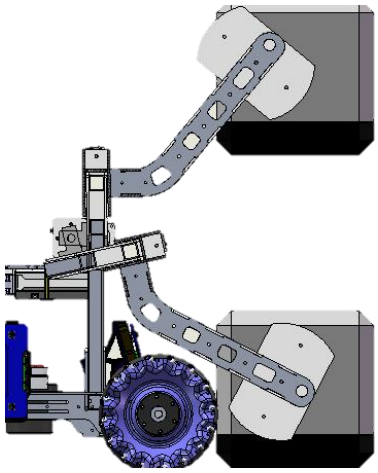


图 4 夹子夹弹药箱

1.1.2 第二阶段

由于我们第一阶段崩了，然后我们组内又讨论了一次方案，主要内容为：
1、放弃 360 云台设计，改造官方车；
2、爪子轻量化设计。

第二阶段整车设计如图 5 所示。爪子的设计方案，依然采取第一阶段的结构。相比于第一阶段，结构更加简单，夹子更加轻，模块化设计，只需拆卸四个螺丝，即可更换。为了解决高处弹药箱弹丸很难倒进弹仓加一个气缸辅助翻转，如图 6 所示。

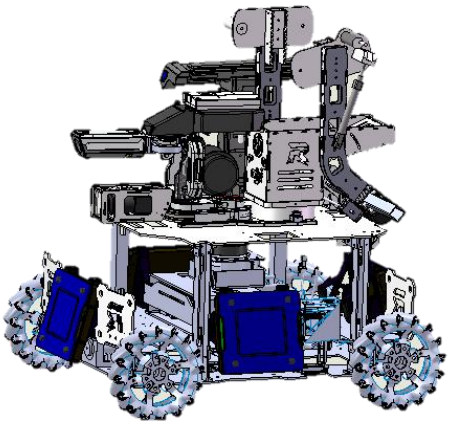


图 5 第二阶段整车模型

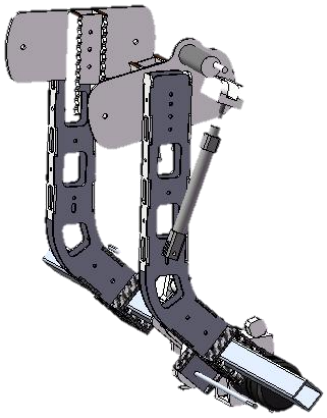


图 6 辅助翻转气缸

爪子可以同时夹取地面上和障碍物上的弹药箱，如图 7 所示。在抓取障碍物上的弹药箱时，需要辅助气缸，将子弹导入到弹仓中，如图 8 所示。

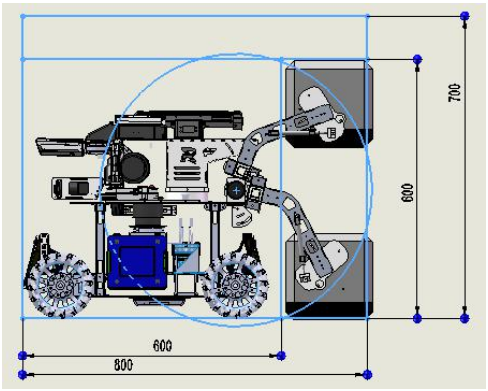


图 7 夹取示意图

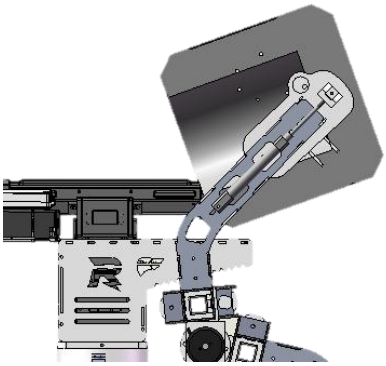


图 8 辅助气缸打开

1.1.3 被舍弃的方案

夹取方案结构设计如图 8 所示，通过链传动带动夹子上下运动，实现地面和障碍物上的弹药箱的夹取；通过顶部链条的倾斜如图 9 所示，实现弹药箱的翻转，使子弹进入弹仓。夹子的夹取和第二阶段方案一致，通过气缸实现夹取。

与之前的方案相比，优点是少了一个辅助气缸的翻转，但是气缸的控制非常简单，所以带来的收益不大；缺点是当电机转速很快时，弯板链容易出现跳动，链传动需要设计张紧机构，还有就是链传动整体较重，对车体重心影响较大。与其带来的收益相比，链传动带来的弊端更大，所以舍弃这种方案。

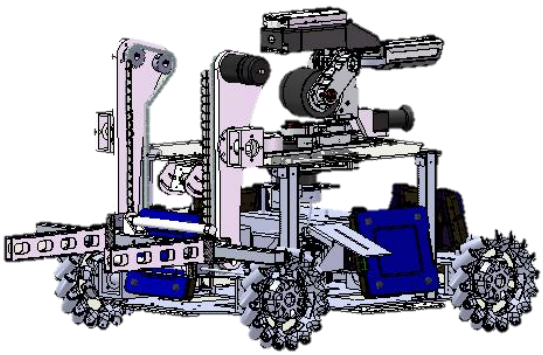


图 8 弯板链夹取

图 9 弯板链侧边

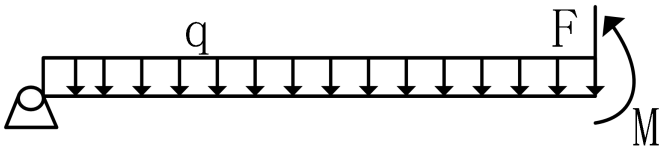
1.2 设计方案（可详解重点或亮点部分）

1.2.1 夹取机构方案设计

夹取机构的设计方案如图 10 所示，其主要组成部分有：翻转电机、夹取气缸（行程 125）、滑车、夹子、光电开关（检测弹药箱）、翻转气缸（辅助翻转）等组成。其主要亮点有 1、结构简单，无需升降机构即可同时夹取地面和障碍物上的弹药箱；2、模块化设计，易拆卸，更换方便；3、夹取域量很大，可以同时夹取两个弹药箱；4、轻量化设计，2mm 的碳板和 2020 的碳方管，算上两个行程 125 的气缸总重大约 600 克左右，对整车重心基本没什么影响。

电机翻转扭矩的计算：

装 50 发 17mm 子弹的弹药箱重为 738g，夹子和气缸自重约为 500g 视为均匀分布的力，最大旋转力臂为 271mm，如下图所示



$$M = 0.5 \cdot 9.8 \cdot 0.271 \cdot 0.5 + 0.738 \cdot 9.8 \cdot 0.271 = 2.63Nm$$

而 3508 的最大输出力矩为 5Nm，持续输出扭矩为 2.8Nm，故电机力矩足够。

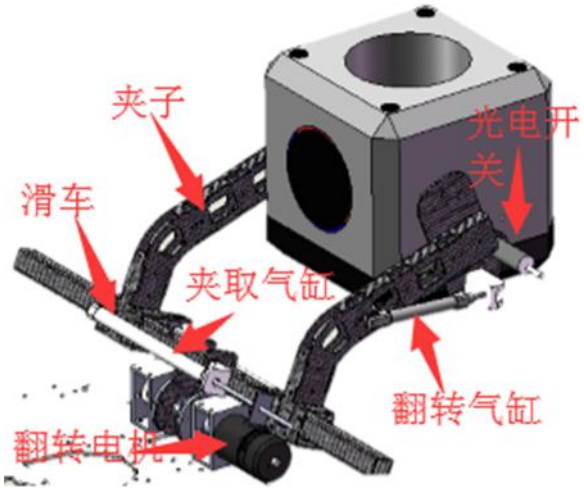


图 10 夹子结构

1.2.2 防撞机构设计

由于场地有很多固定不动的障碍物块，在导航经常崩的情况下，而且战车彼此是看不到的，撞击可能会经常发生，可能会撞掉血，进而改变比赛的局势。故防撞机构的设计以及装甲片的保护是非常有必要而且十分重要的。
防撞机构的设计如图 11 所示，前轮导轮可以有效防止发生车子卡住的情况，装甲片下面的保护板，可以避免装甲因为撞击掉血，这一点比赛过程中，得到了有效的证明。

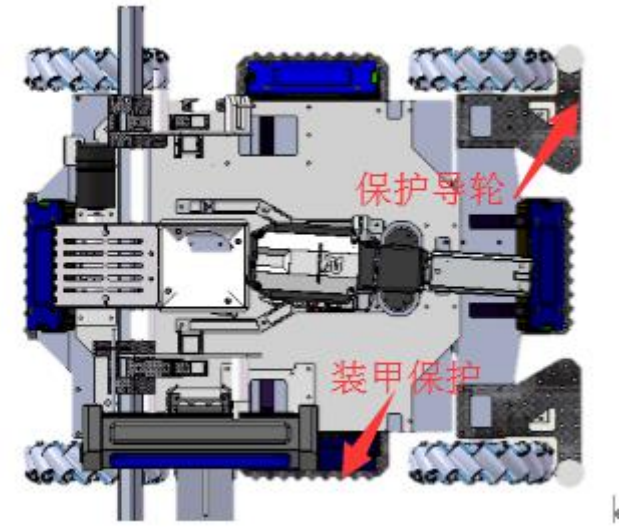


图 11 防撞机构设计

1.3 最新的装配图

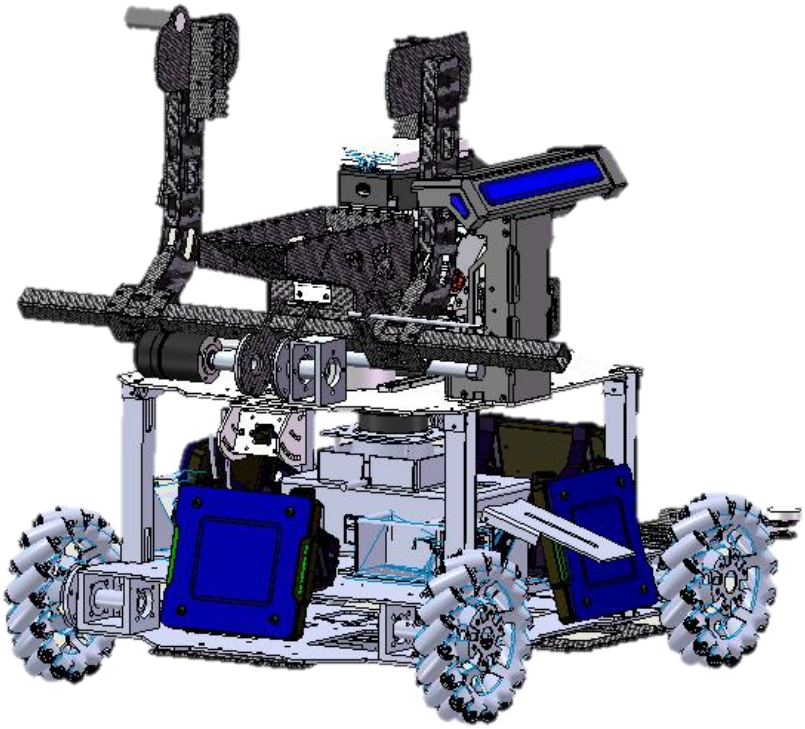


图 12 最新装配图

1.4 还可以优化的方向

1、夹取障碍物上的弹药箱成功率只有 90%，主要问题在于由于夹子和弹药箱接触的摩擦力不够，导致旋转气缸翻转时打滑，尝试过用接触面上粘上砂纸，效果不是很好；
2、当气瓶气压低于 0.4MPa 时，辅助翻转气缸很难推出，现在用的气缸缸径为 10，可以尝试缸径更大的气缸，解决这个问题；
3、气路布置问题，有一辆车迷之漏气；（苟鑫说：这锅好大啊。。）

以上机械部分由：夏令营三组崔家硕编写（爆肝小王子）

2. 嵌入式部分

2.1 整体方案

1 任务要求

此次夏令营的全自动地面机器人需要完成的主要功能包括取弹，自主导航及避障，瞄准射击三个部分。

由于是射击对抗比赛，所以机器人在开局及比赛过程中能够稳定获取弹丸是个队伍需要实现的第一个重要功能。

如何准确运动到达能够获取弹丸的位置，及在比赛过程中通过灵活的走位来打击敌方却躲避敌方的攻击，是此次全自动机器人所要实现的第二个功能。

当在运动过程中发现目标之后，机器人需要能迅速锁定目标，调整云台对目标实现准确打击。因此，通过控制云台、拨盘电机、摩擦轮，实现瞄准射击，是地面机器人的第三个重要功能。

2 方案概述

(1) 取弹：

机器人主控接受妙算通过串口发送的速度来驱动电机，到达摄像头能够稳定识别弹药箱的位置，妙算识别到弹药箱后，通过计算弹药箱相对机器人本体的位姿，给出机器人在 x, y 两个方向的移动速度和 z 方向的转动速度，直到抓取机构到达可以抓取弹药箱的位置。此时主控会放下爪子。并通过查询抓取机构安装的光电传感器检测是否可以抓取。如果满足便控制继电器通过气动机构夹取弹药箱，抬起爪子将弹药倒进弹仓。对于高台的弹药箱，主控会额外控制爪子上另一个旋转气缸，增大弹药箱的旋转角度。

(2) 弹量检测：

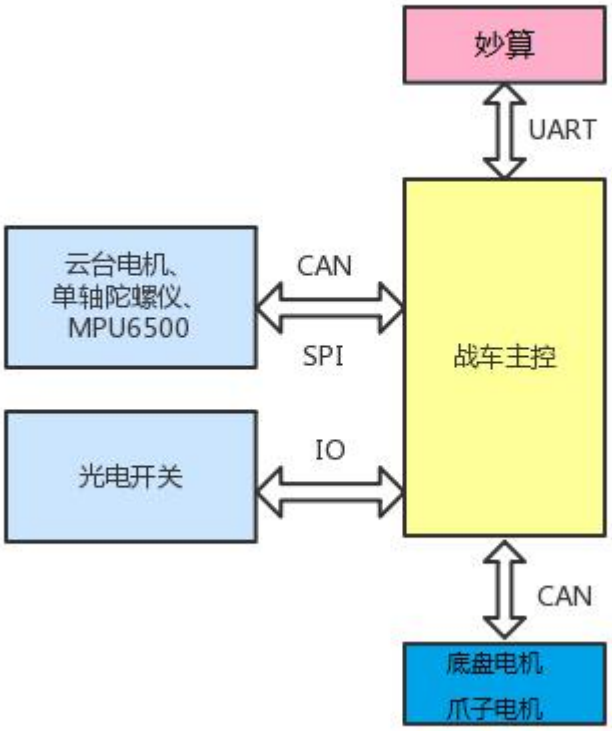
主控通过 CAN 与战车底盘的四个电机进行通信，主控会发送电流信息控制电机转动，电机返回速度位置等信息。因此，通过返回的速度和位置信号，我们可以对底盘进行位置-速度的串级 PID 闭环控制。

对于上层来说，战车只有 x, y 两个方向的移动和 z 方向的转动，而我们实际控制的是四个电机的电流。因此，需要对四个麦轮的转动和战车平移转动进行运动学的正反解。对运动学解算后的数据进行闭环控制，即可实现对战车准确机动的功能。

(3) 瞄准射击：

妙算通过大恒摄像头获取机器人视野内的图像信息，进行装甲片的识别，可以得到装甲板在视野中心的偏差距离，包括 yaw 和 pitch。结合视觉、码盘和单轴陀螺仪的信息，可以对云台进行闭环控制。从而使枪口能快速准确跟随敌方的装甲板。当云台到达可以发弹的适当位置时，便会启动拨盘电机送弹，通过摩擦轮使子弹快速射出，达到瞄准射击的功能。

战车的框图如下：



2.2 底盘控制模块

自动步兵车的底盘控制仍旧以步兵车的地盘控制为支持，在此的基础上发展出了三组的七种特殊底盘模式。如下图所见，结构树右侧的模式为步兵车模式，左侧的模式为自动车的特殊模式。

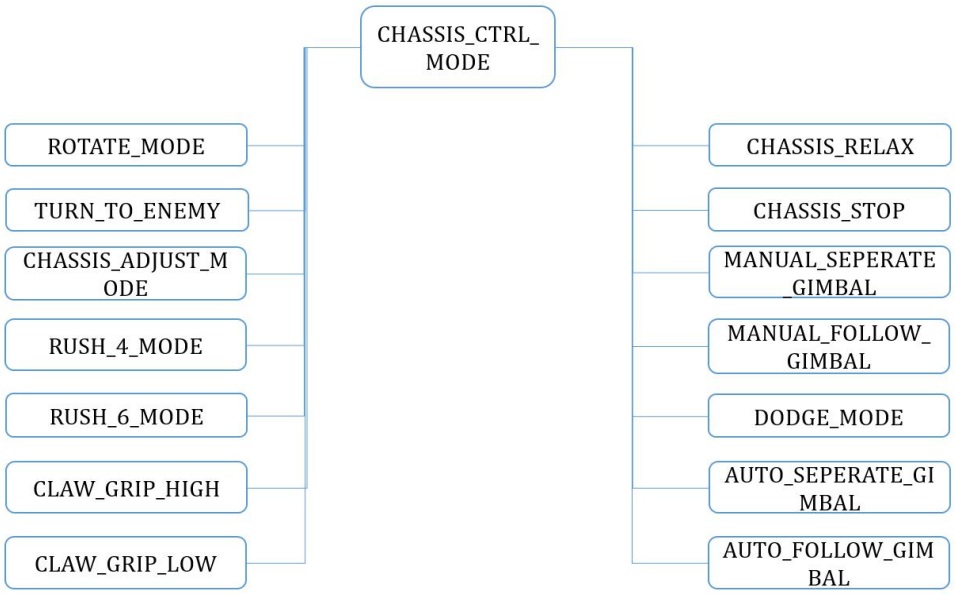


Figure 1Overall Control Flow

这其中自动车模式中最后两种 CLAW_GRIP_HIGH 和 CLAW_GRIP_LOW 模式都将应用于弹药箱的抓取控制流程中，将在 2.4 小节被解释。其他五个模式的详解如下。

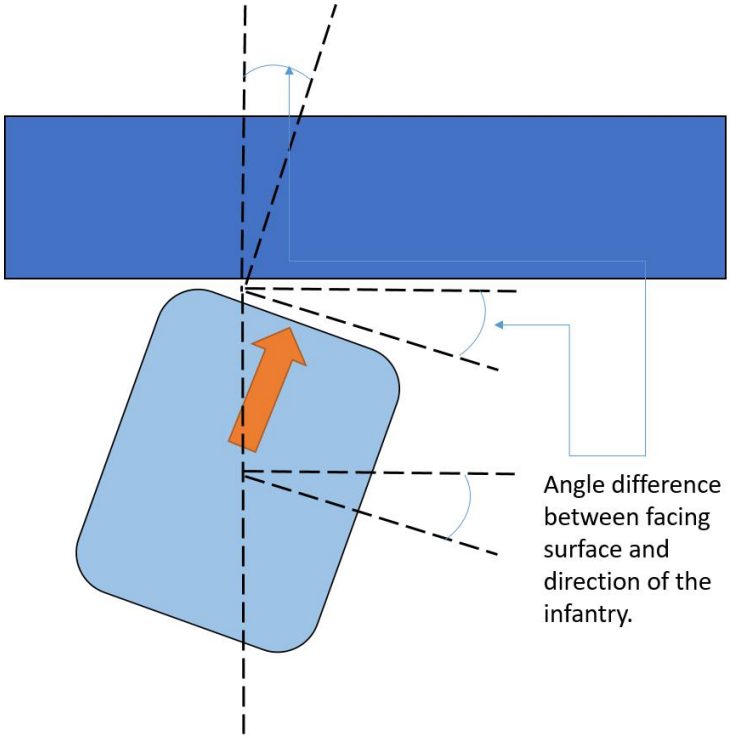
ROTATE_MODE:
Mode Purpose: To rotate the infantry and avoid being recognized by enemy;
Mode Trigger: Receiving PC message about whether enter the rotate mode or not;
Mode Contents: Rotate the infantry chassis by setting *chassis.vx* . A *rotate_side* flag is used to set the direction of rotation.

TURN_TO_ENEMY:
Mode Purpose: To turn the infantry to the direction of armor attacked, so that visual recognition and counterstrike command could run at the first time;
Mode Trigger: Receiving PC message about being attacked;
Mode Contents: Call the *judgement_info* .;*judge_recv_mesg* . *blood_changed_data* . *armor_type* to locate which direction is the enemy; Turn to a certain direction by setting different *chassis.vx* and timer combinations (by defined counts) .

RUSH_6_MODE:
Mode Purpose: To Rush to the central buff zone;
Mode Trigger: Receiving PC message about whether enter this mode and which path to take;
Mode Contents: Run the infantry to the buff zone by timer and encoder.

RUSH_4_MODE:
Mode Purpose: To Rush to the central buff zone;
Mode Trigger: Receiving PC message about whether enter this mode and which path to take;
Mode Contents: Run the infantry to the buff zone by timer and encoder.

CHASSIS_ADJUST_MODE:
Mode Purpose: To adjust the attitude angle of the infantry, especially when it is ready to grip the higher bullet boxes, and make the infantry face vertically to the surface of the gripping point;
Mode Trigger: Receiving PC message about whether to start adjusting;
Mode Contents: The angle of adjusting is relative to the chassis gyro feedback angle calling from *chassis.gyro angle.* This adjusting action is controlled by a PID algorithm, which make the process quick and smooth.



2.3 云台与射击控制模块

自动车的云台控制结构沿用步兵车的架构。在云台的 PID 控制参数和射击的射速射频控制中我们加入了自己的思想。主车和辅车的云台 PID 参数有所区别，其原因是由于两车云台 PITCH 轴传动带紧固程度不同。受自动车的打击反应的稳定性的要求，我们将云台 PID 控制微微 over-dumping，因此其反应力可能不够快速，但对于小范围内运动的装甲块目标打击更为稳定，单位时间内造成的伤害更高。考虑到射速对于伤害没有特殊加成，而一定程度的高射速又有利于提升击打移动装甲板的准确率，我们经过计算将射速固定在 16 米每秒，并以此来对应的在热量控制下提升射频。

2.4 弹药箱抓取控制

自动车的弹药箱抓取控制由上层命令指示开始，由下层条件决定何时结束。当机器人完成视觉微调与位姿微调，上层将发送命令进入抓取模式。具体的抓取流程如下。在抓取过程中我们应用了 PID 控制算法和斜坡控制算法来达到使抓取更为稳定快速的目的。

CLAW_GRIP_LOW:
Mode Purpose: To grip the bullet box on ground positions.
Mode Trigger: Receiving PC message about whether enter this mode;
Mode Contents:
Approaching Action: An infrared sensor is utilized on the claw so that while a determined delay after the sensor brings back the signal of ready for gripping
Catching Action:, the claw will catch the bullet box swiftly and tightly by setting two pneumatic pistons at each side of the box as *set(activated)* or *preset(retrieved)*.
Lift-up Action: And then a motor is controlled by reading its encoder value as feedback and calculating its PID output so that it stops at expected positions quickly and stably.
Put-down Action: Setting the angle reference of PID control of the lift-up motor to its catching position, so that the bullet box is returned to its original position.
Back Action: All the parameters and actions will return to their original status.

CLAW_GRIP_HIGH:
Mode Purpose: To grip the bullet box on higher positions.
Mode Trigger: Receiving PC message about whether enter this mode;
Mode Contents: The overall control strategy of **CLAW_GRIP_HIGH** mode is similar to that of **CLAW_GRIP_LOW** mode. The only difference is that, between the *lift-up action* and the *put-down* action, there is a *rotate action* pouring the bullet from the box by rotating the box by an extra piston on each side while holding it.

2.5 难点与不足

2.5.1 PID 控制

在对于云台实现的专门应对打击的 PID 控制中，未能实现和云台驾驶模式下的 PID 模式有所区分。这导致有时自动车的底盘进入 CHASSIS_FOLLOW_GIMBAL 模式时精度降低。

2.5.2 卡尔曼滤波

我们对于在云台识别到的敌人进行追踪的 TARGET 算法中应用的卡尔曼滤波算法认识不够深刻，如果能进行相应的改进，应该可以增加追踪射击的反应速度和精确度，同时一定程度上避免出现 “射墙”的问题。

以上 2.2 2.3 2.4 2.5 由三组嵌入式刘李正编写。

3. 算法部分

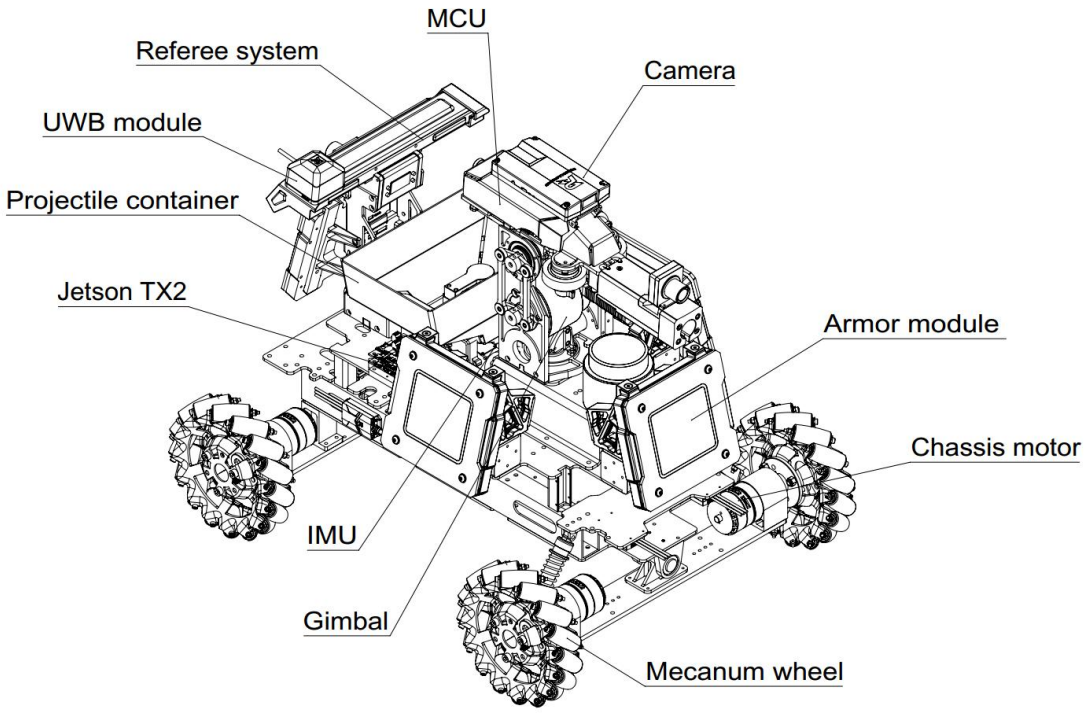
3.1 开发环境介绍

3.1.1 硬件环境

机器人上所使用的计算设备妙算，传感器包括激光雷达，单目摄像头，陀螺仪。妙算作为上位机，运行 ROS 系统，负责策略，导航与路径规划，目标识别等任务。

下位机则是一块 STM32 的主控板，通过接受妙算通过串口发送过来的数据，进行对云台，底盘电机以及摩擦轮等底层设备的控制。

机器人各硬件部分分布如图所示：



3.1.2 算法环境

算法环境搭建

算法环境主要指妙算上所配置的相关的开发工具，此次夏令营所用的开发环境为 ubuntu 上的 Kinect 版的 ROS，根据我们本队对于弹药箱的识别方案，额外配置了 yolo 的依赖环境，同时为了视觉方面的调试，安装了 ROS 上的 OpenCV 包。关于定位和导航的，也都需要安装好对应的功能包。

3.2 整体技术方案概述

本次夏令营机器人对抗赛按功能需求可分为以下个模块：定位、导航、识别弹药箱及装甲片，行为树编写等。具体的实现及方案选择如下：

1， 定位，通过嵌入式发送的 odometer 数据及雷达 scan 获取的信息，结合 UWB 全场定位给出的全局坐标，通过数据融合及 AMCL 算法给出机器人相对于场地的全局位姿。

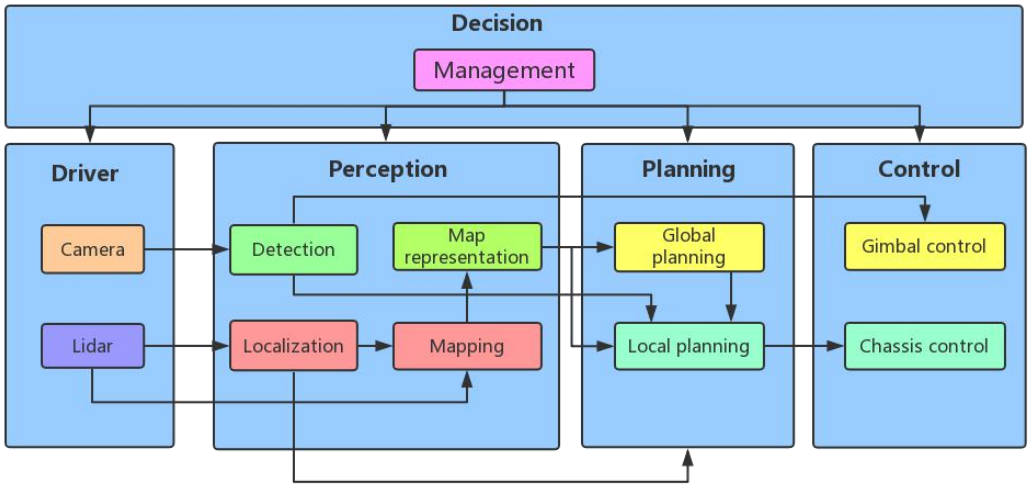
2， 导航，分为全局导航和局部导航。全局导航选用 A-star 算法，局部导航则选用 TEB 算法，实现动态避障的效果。

3， 识别弹药箱及装甲片，采用 yolo 此类深度学习的图像识别方案，给出目标弹药箱相对机器人的位姿信息，以此控制机器人到达合适的位置。至于装甲片的识别，通过 RGB 色彩过滤（经过测试效果较 HSV 更加稳定），

找出红色或蓝色色块，并利用装甲片本身的大小，相对位置，绝对位置信息过滤掉不合理的色块，并解算位姿转换为云台 yaw 轴和 pitch 轴的控制数据，通过串口发送给嵌入式主控。

4， 行为树编写：根据比赛规则及队内方案，通过推测比赛中机器人可能遇到的情况编写对应的行为，通过 ROS 的 action-lib 模块进行编程实现。算法整体框架设计

各模块的关系如下图所示：



a) 算法功能模块说明

3.4.1 定位算法

本次夏令营所用的定位算法主要是 AMCL 最大似然度拟合的方法。该算法对机器人底盘根据码盘、陀螺仪测量得到的里程计数据的准确度要求较高。一旦机器人被障碍物阻塞，底盘打滑空转时，易出现 AMCL 校准偏慢，最终雷达数据匹配失效的情况，此时机器人的定位会出现严重的偏差，完全不能用来定位和导航。因此需要添加进额外的全局定位数据来矫正，所采用的是 UWB 全场定位方案，不过由于 UWB 易受遮挡的影响，及需要被准确的校准。我们队伍通过实验验证添加 UWB 后，虽然对基于 odometer 来定位的数据有不错的纠偏效果，但在动态导航的过程中，却效果不佳，机器人经常出现撞到障碍物的情况。因此最后不采用该方案来矫正。

我们队伍有考虑过采用基于一对正交编码器和陀螺仪的全场定位方案来解决这个问题，该方案较为稳定，但对地面平坦度要求较高，无法适应凸起凹陷较多的场地。不过此次夏令营的场地十分平坦，适合该方案。不过由于经费及时间所限，我们队伍最终放弃该方案。

3.4.1 导航算法

比赛过程中所用到的导航算法主要有两方面的，分为全局路径规划和局部导航。全局路径规划所采用的是 A-star 算法，

A*（A-Star)算法是一种静态路网中求解最短路最有效的方法。公式表示为：f(n)=g(n)+h(n), 其中 f(n) 是节点 n 从初始点到目标点的估价函数， g(n) 是在状态空间中从初始节点到 n 节点的实际代价，函数 g 一般是固定的，就是初始节点到当前节点的距离，一般会选取欧式距离或者曼哈顿距离 h(n)是从 n 到目标节点最佳路径的估计代价。heuristic 函数，启发式函数这里的 h(n)也就是启发式函数，一般来讲，这个 h(n)取两节点间直线距离作为估价值，也就是

h(n) = sqrt((x1-x0)^2+(y1-y0)^2)

在 ROS 的 costmap 中在路径规划中是按照像素格子去计算的，也就是曼哈顿距离。

而局部路径规划所采用的是 teb 算法，teb 算法在运行时，优化由全局路径规划器生成的初始轨迹，以便最小化轨迹执行时间（时间最优目标），与障碍物分离，并遵守诸如满足最大速度和加速度的动力学约束。

以上 3.11-3.4.1 由夏令营三组华南虎虎哥（魏伟和）编写

3.4.2 跟踪射击

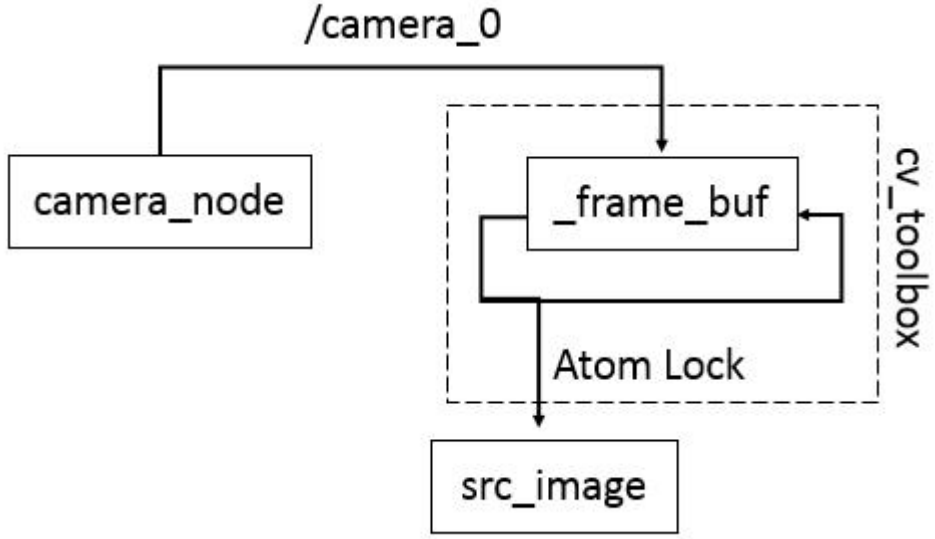
1. 简介

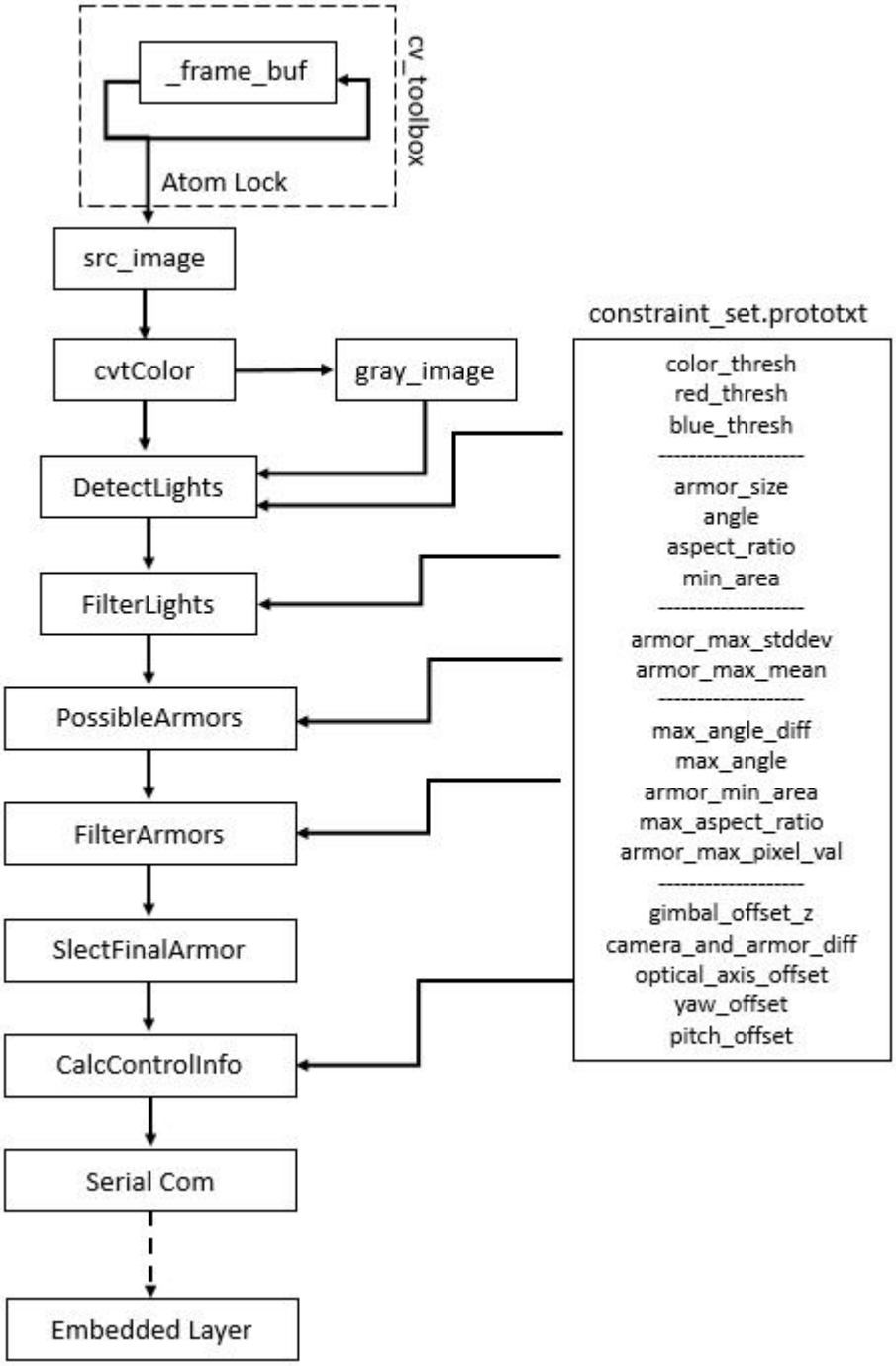
   跟踪射击是 RM 夏令营挑战赛中最为核心的模块之一。决定了是否能在成功获取弹药箱之后具备反击能力。其以装甲板具备固定的视觉识别特征-灯条-作为目标，使用传统计算机视觉方法，构建了一套较为成熟可靠装甲板识别系统。且不同于其他模块，跟踪设计模块将部分需要高实时高传输带宽的任务下放到了嵌入式端，提高系统的鲁棒性，降低了串口通讯负载。
2. 核心算法实现简介

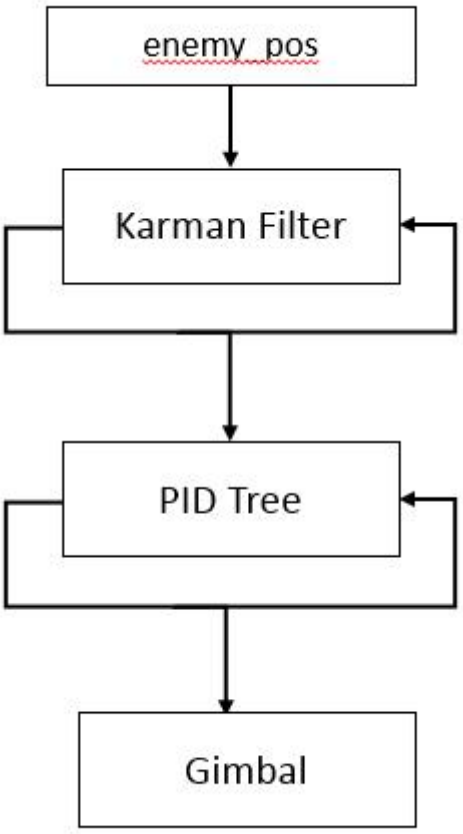
   装甲板视觉模型上是一固定单元，具有两定长度平行 LED 照明灯条，能在纯红色与纯蓝色之前切换，亮度极高，且有较为严格的安装要求。所以大概流程为先找出图像中符合要求的色块，再根据灯条形状与安装要求滤除不符合规则的色块并两两匹配成装甲板模块。由于色彩纯度极高，所以常用相机往往会遇到曝光过度，装甲板颜色在成像结果中变成纯白色的情况。所以在夏令营中我们使用了参数高度可调的大恒光学工业 USB3 相机。通过合理的降低曝光速度与增益，保持灯条整体不过曝，便于后期识别与处理。跟踪模块内置 HSV（Hue, Saturation, Value）色相，饱和度，明度分离，与 RGB 红绿蓝可见光组元分离两种 RGB 二值化方法。由于灯条使用了纯度极高的单色 LED 光照，所以 HSV 上可见其红蓝色相距离较远，饱和度非常高，明度较为稳定。所以可以高效稳定的过滤掉观感上不符合的色块。但是在比赛中，我们采用了较为传统的 RGB 分离，通过合理的调整参数，RGB 分离法在我们的测试中

呈现出了更低的召回率与更精准的定位，同时 RGB 法参数整定在保持全图不过曝下较为简单。当得到过滤后的装甲板模块后，通过已知装甲板模型计算装甲板的相对位姿，通过串口传递给下位机用于目标追踪与预测。下位机通过一个一阶卡尔曼预测，反馈目标 Yaw 与 Pitch 给云台。由于输入，输出，反馈系统存在较大的延迟，如果直接引入 PID 控制目标，可能会造成震荡，但是传统的超前滞后滤波器对于强非线性延迟作用有限，所以射击模块引入了一个一阶卡尔曼滤波器，用于补偿延迟造成的震荡问题，同时预测目标在未来短时间内的移动趋势，提高设计的精度。

3. 识别流程



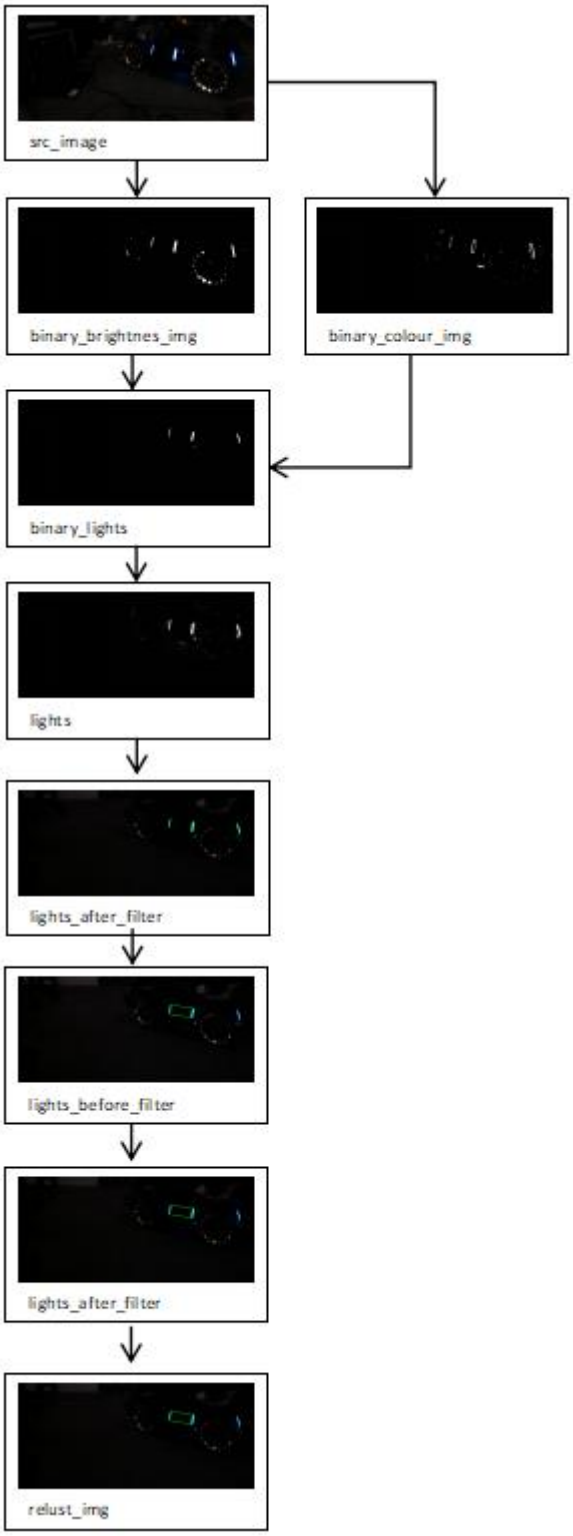

cv_toolbox
_frame_buf
Atom Lock
src_image
cvtColor
gray_image
DetectLights
FilterLights
PossibleArmors
FilterArmors
SlectFinalArmor
CalcControlInfo
Serial Com
Embedded Layer
constraint_set.prototxt
color_thresh
red_thresh
blue_thresh
armor_size
angle
aspect_ratio
min_area
armor_max_stddev
armor_max_mean
max_angle_diff
max_angle
armor_min_area
max_aspect_ratio
armor_max_pixel_val
gimbal_offset_z
camera_and_armor_diff
optical_axis_offset
yaw_offset
pitch_offset



射击模块首先从 cv_toolbox 中取出缓冲区内的 src_image，相较于直接从单一 buffer 缓冲中取出避免了图像撕裂的产生，同时能够减少等待，稳定识别帧率。Ping-pong 缓存与原子锁能保证文件完整性不受读取写入影响，且在轻负载场景下表现良好，实现简单。src_image 后通过 OpenCV 内置函数降维成亮度灰度图，随后用于从场景中检测出特定颜色。DetectLight 函数使用阈值二值法提将符合 HSV 或 RGB 范围的像素点二值为黑色，其他为白色并找出包含对应形状的外框。PossibibleArmor 找出所有可能的装甲板组合，随后 FilterLights 在其中尝试匹配所有可能组成对的装甲板，并滤除不可能存在的情况，如角度差，整体倾角，整体面积，最大长宽比，以及最大的像素距离。去除装甲库中超出其标准差与平均值要求的低置信装甲板。 SelectArmor 找出最大的装甲板（即距离最近），然后应用校准参数 CalcControlInfo，校准摄像头外参后将数据发送给底层嵌入式端。嵌入式端从上位机端取出目标数据，送入卡尔曼滤波器后将数据递交给云台 PID 末端闭环。

4. 测试结果

经过严格的参数整定之后，优化了实际的识别效果，根据赛制重新制定嵌入式端卡尔曼滤波器，与状态转换的参数。根据最后场地实测，装甲检测模块工作稳定正常，实现了既定目标，稳定识别目标并拥有较为精准的在预定距离打击目标的能力。


src_image
binary_brightnes_img
binary_colour_img
binary_lights
lights
lights_after_filter
lights_before_filter
lights_after_filter
relust_img

5. 不足与提高
    1. 卡尔曼滤波器给下端机造成了一定的运算压力，同时卡尔曼本身对于时隙要求不严格，如此对于算力要求较大的算法应该跑在高性能 PC 上。此处设计到的 USB 带宽问题，RM 主控板配备了 USB 接口，以后的设计我建议考虑使用 USB 软串口传输，不仅有硬 CRC 校验，自动重发，且其鲁棒性，易用性，远高于普通异步串行通讯协议（UART）。
    2. 指令的设置中给出了是否启动 neon 的指令，但是实际并没有做实现。由于缺乏 SIMD 的支持，导致断码段运行缓慢，占用了一定的资源。对于 TX2 嵌入式系统而言，可以考虑引入 CUDA 或者 compute shader 等进一步提高运算速度。CUDA9.0 支持了 Uni-Memory，支持 Zero-Copy Access，对于 TX2 显存/运存一体的 SoC，可以规避由于拷贝导致的运算缓慢问题。期望以后加入对于 SIMD 指令或 CUDA 的支持。
    3. 目标运动时的预测打击仅基于卡尔曼滤波器的预测机制，卡尔曼滤波的相应机制对于实际比赛中机器人复杂运动姿态如猫步（狗头 dogged）模式，甚至是简单自旋缺乏抑制性。没有完全利用云台所拥有的带宽。上层目标位置输出不仅存在延迟，更存在着更新速率底下的问题，卡尔曼可以等效成一个低通滤波器，进一步降低了其控制指令的等效带宽。如果能够根据机器人的实际结构与性能构建一个控制学模型，通过视觉的输出更新观测器而不是直接控制输出，理论上可以极强的抑制其底盘的随机摆动对于视觉识别造成的干扰。
    4. 摄像头节点占用了过多的 CPU 资源，导致帧率提升困难。摄像头数据的预处理也可以引入 SIMD 指令，ROS 的原生摄像头预处理太过于复杂，不适合 RM 这种资源受限的环境。可以考虑引入自有数据格式。同时存在着处理冗余的问题。如大恒相机直接读取的结果已经是一个一维数组，可以直接塞进 Sensor_msgs/Image，而原文采用了较为低效的两次转换，产生了资源的浪费。

以上 3.4.2 由韩世豪编写（我笑笑是大哥啊）

附：yolo 简介

（1）核心思想

YOLO 的核心思想就是利用整张图作为网络的输入，直接在输出层回归 bounding box 的位置和 bounding box 所属的类别。

YOLO 检测网络包括 24 个卷积层和 2 个全连接层，如下图所示。

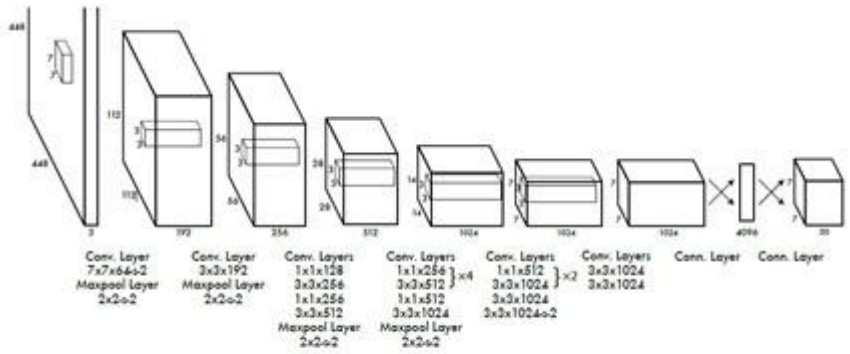


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

其中，卷积层用来提取图像特征，全连接层用来预测图像位置和类别概率值。

YOLO 使用均方和误差作为 loss 函数来优化模型参数，即网络输出的 S*S*(B*5 + C)维向量与真实图像的对应 S*S*(B*5 + C)维向量的均方和误差。

$$loss = \sum_{i=0}^{S^2} coordError + iouError + classError$$

其中，coordError、iouError 和 classError 分别代表预测数据与标定数据之间的坐标误差、IOU 误差和分类误差。

（2）为什么选择 yolo v3

v3 和 v2 的不同点：

loss 不同：作者 v3 替换了 v2 的 softmax loss 变成 logistic loss，而且每个 ground truth 只匹配一个先验框。

anchor bbox prior 不同：v2 作者用了 5 个 anchor，一个折衷的选择，所以 v3 用了 9 个 anchor，提高了 IOU。

detection 的策略不同：v2 只有一个 detection，v3 一下变成了 3 个，分别是一个下采样的，feature map 为 13*13，还有 2 个上采样的 eltwise sum，feature map 为 26*26，52*52，也就是说 v3 的 416 版本已经用到了 52 的 feature map，而 v2 把多尺度考虑到训练的 data 采样上，最后也只是用到了 13 的 feature map，这应该是对小目标影响最大的地方。

backbone 不同：这和上一点是有关系的，v2 的 darknet-19 变成了 v3 的 darknet-53，因为需要上采样，卷积层的数量自然就多了，另外作者还是用了一连串的 3*3、1*1 卷积，3*3 的卷积增加 channel，而 1*1 的卷积在于压缩 3*3 卷积后的特征表明其实用性大大增强。

相比于 yolo v2，yolo v3 准确度更高且实时性更好，最后决定使用 yolo v3。

1. 数据采集

（1）采样方法及采样结果

采样的主要方法是实车实际场地进行采集，之后再手动补充一些特殊角度的照片。采样时考虑了样本的多样性，所以综合考虑光照、距离、角度、清晰度等影响因素后进行了样本的采集。采集得到的结果按照影响因素分类如下：

1）光照

光照不足条件下：

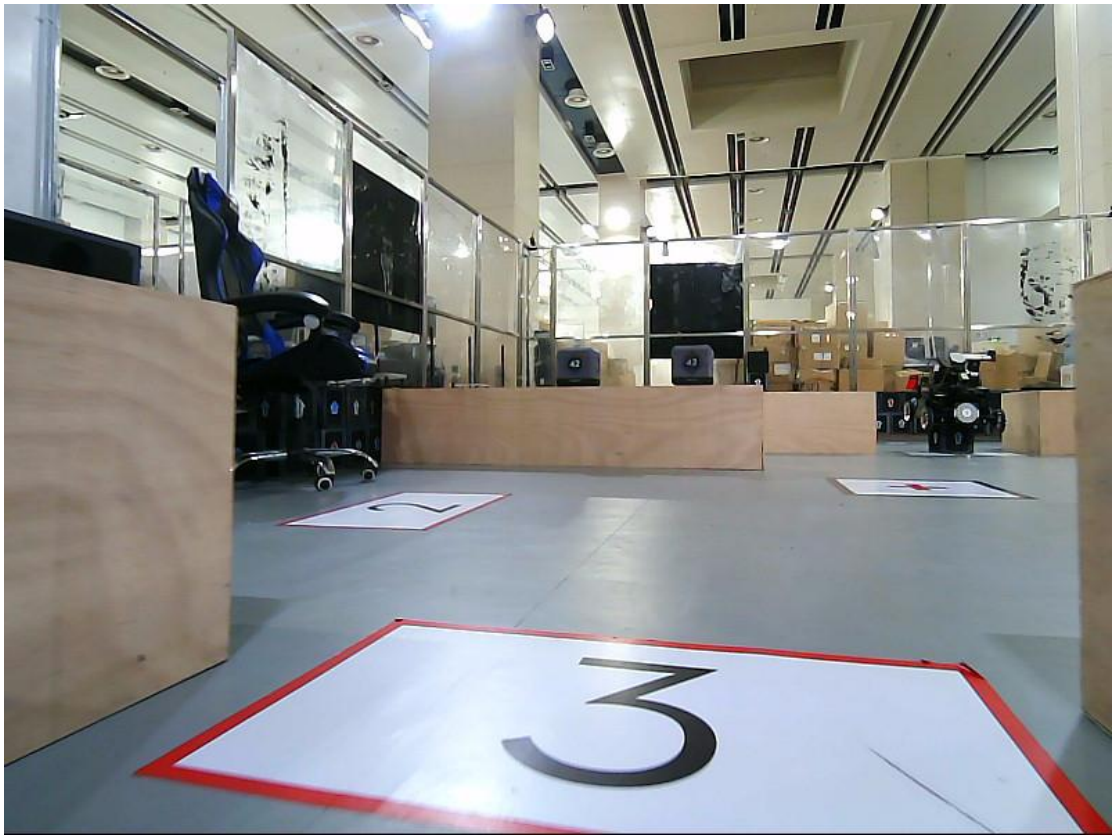
光照充足条件下：

2）距离

距离特别近以至看不全目标：

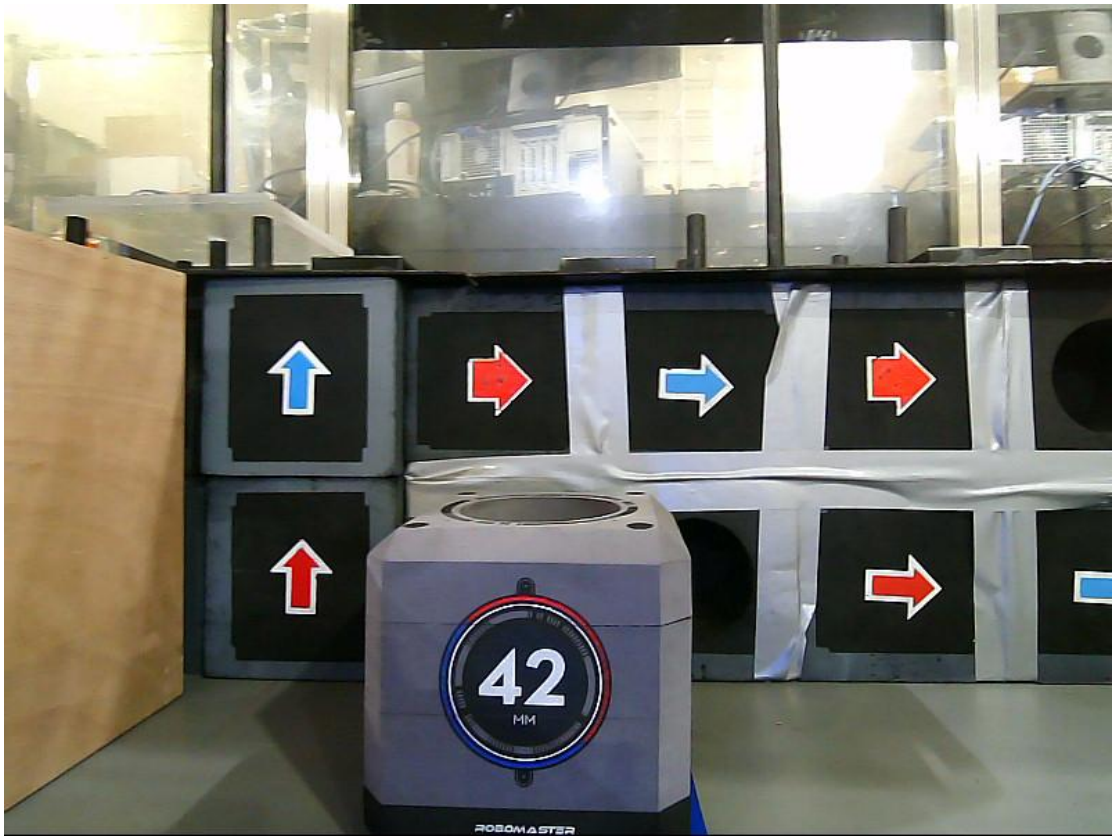
距离较近：

距离较远：

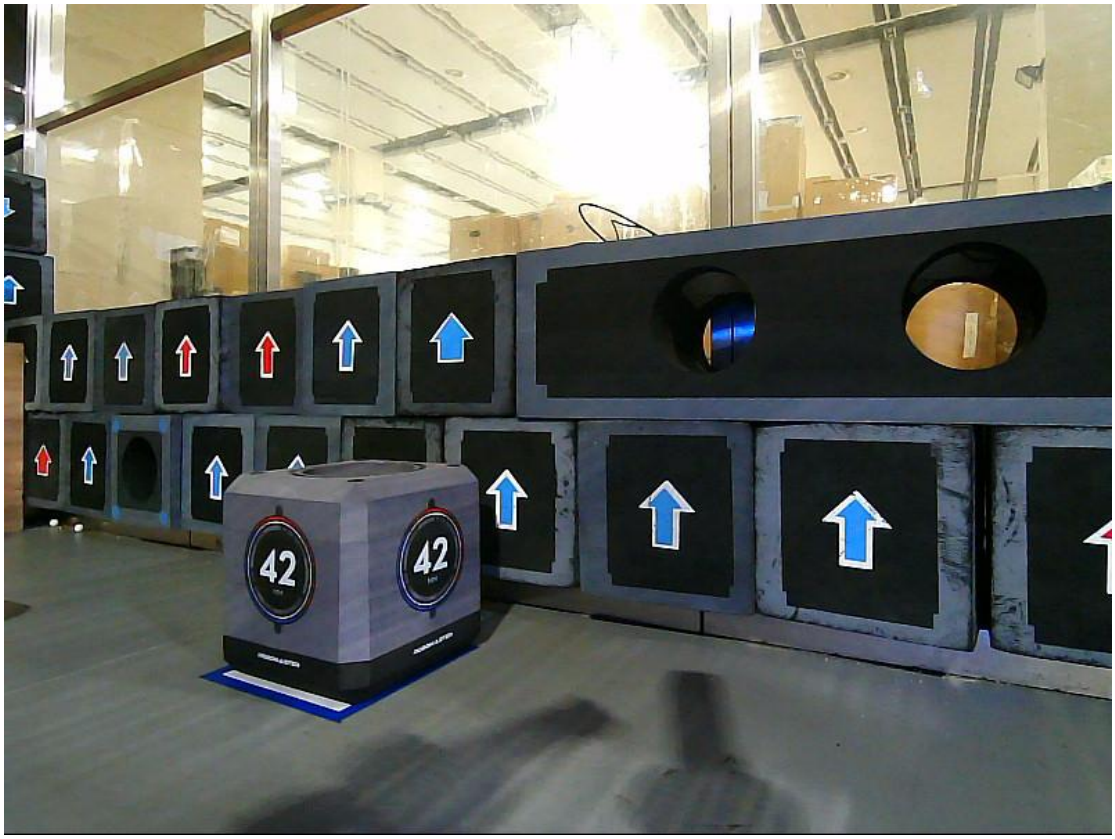


3）角度

正面：



侧 45 度：

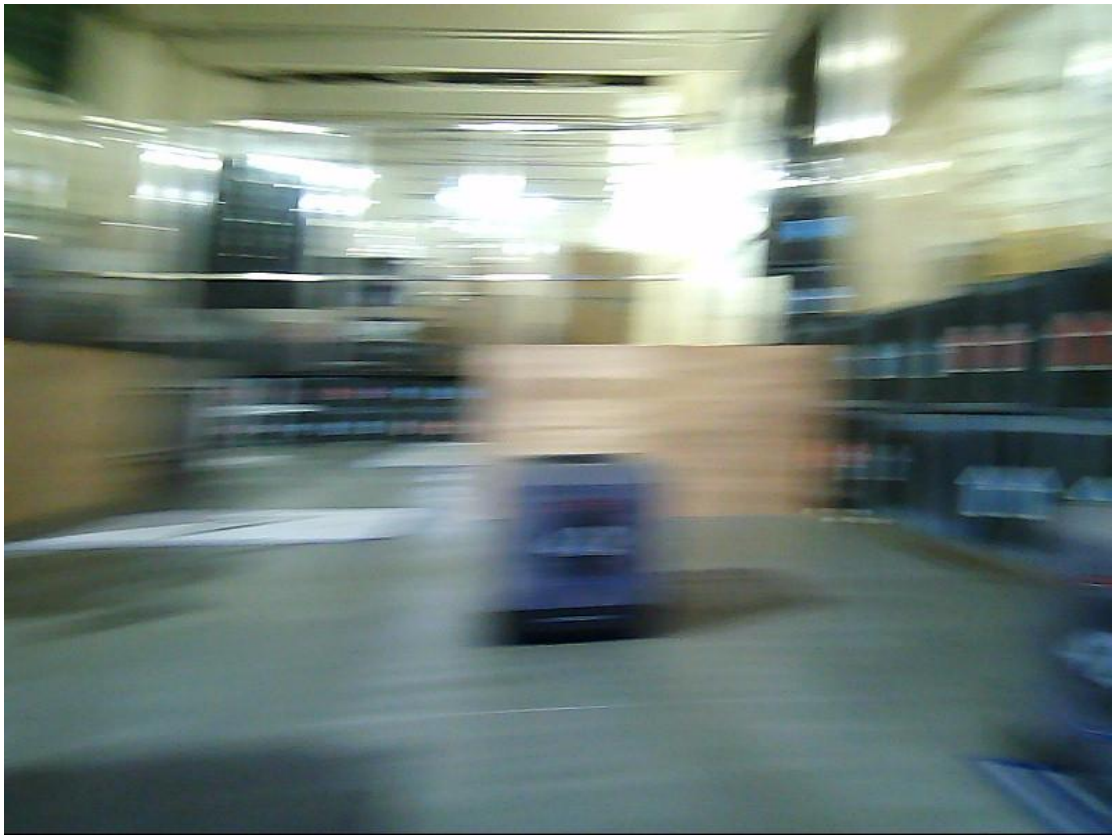


其他特殊角度：

4）清晰度

失焦：

横向模糊：



（2）打标工具 yolo_mark

使用方法：

1）下载 https://github.com/AlexeyAB/Yolo_mark，进行编译；

2）运行检查安装成功

on Windows: x64/Release/yolo_mark.cmd　　//直接运行命令

on Linux: ./linux_mark.sh

3）配置打标相关参数

delete all files from directory x64/Release/data/img　　//删除目录下所有文件

put your .jpg-images to this directory x64/Release/data/img　　//将要标注的图片文件放在目录里

change numer of classes (objects for detection) in　file x64/Release/data/obj.data:　//改类别数

classes= 2　　//改数目

train　= data/train.txt

valid　= data/train.txt

names = data/obj.names

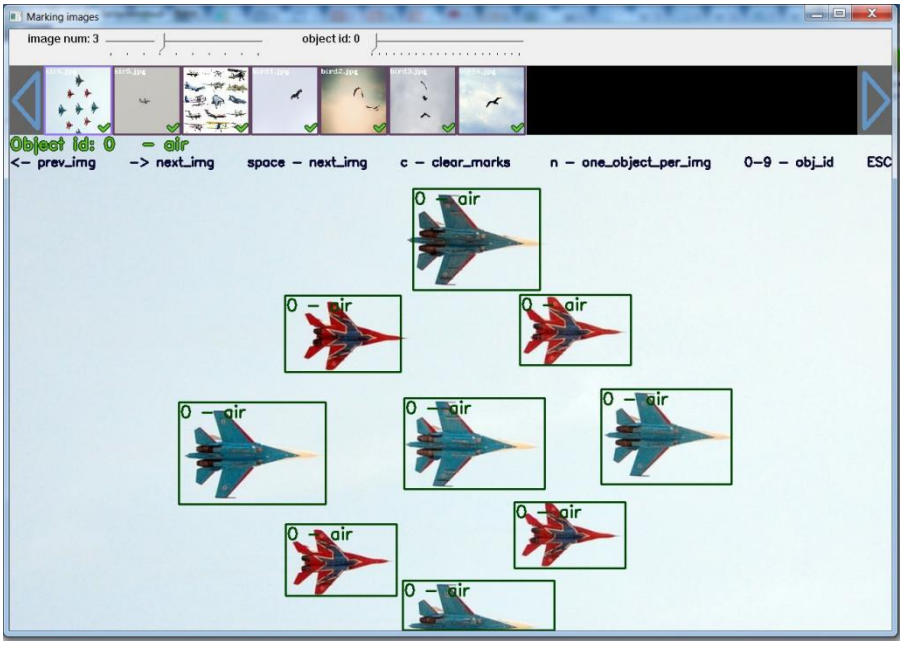
backup = backup/

put names of objects, one for each line in file x64/Release/data/obj.names: //添加类别名称 类似 car bird 等名称

4）运行并进行手动打框

x64\Release\yolo_mark.cmd　//运行

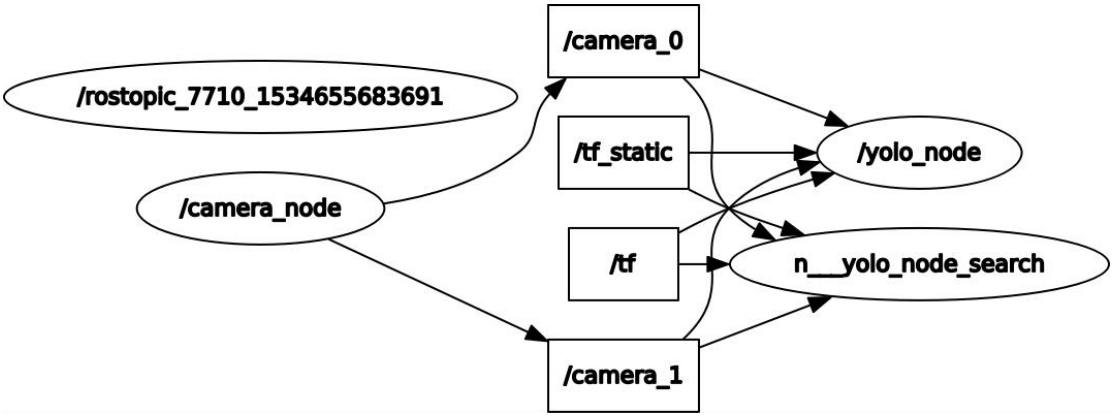
打框完成后效果如下：



2. yolo 在系统中的应用方案

（1）yolo 识别弹药箱流程

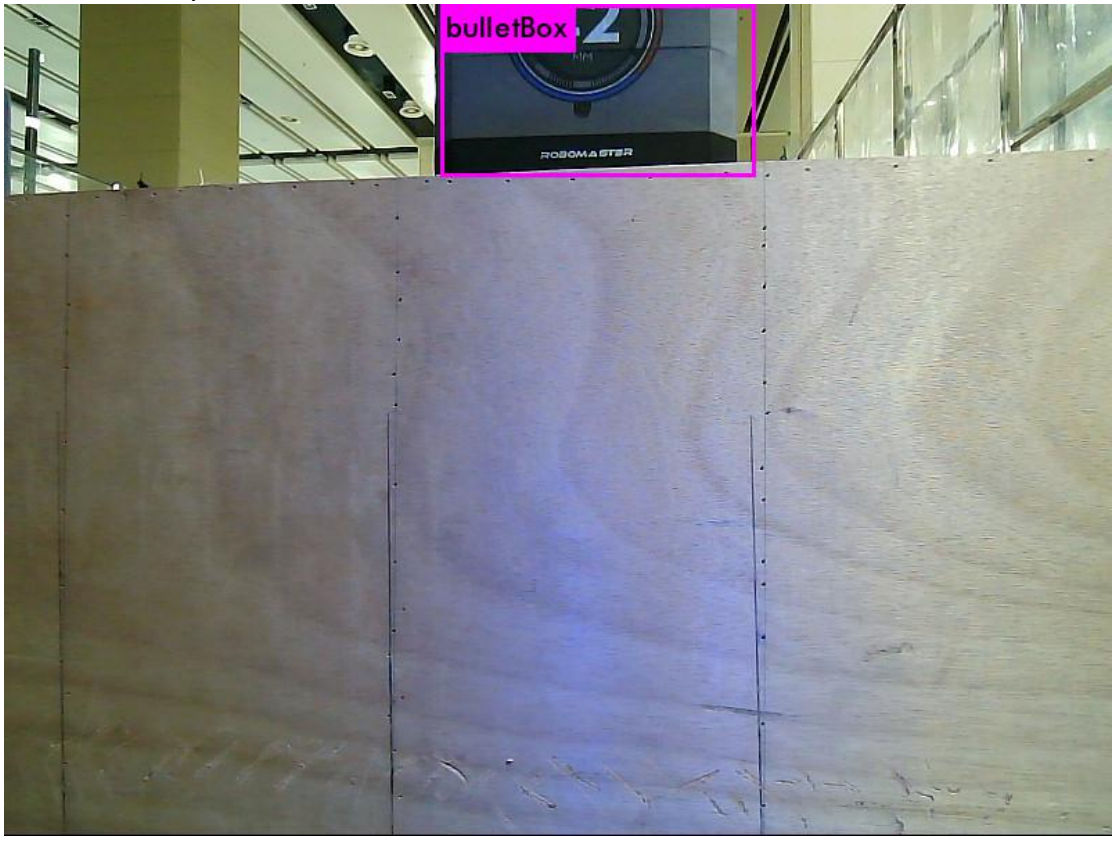
图像信息来源于车体后部的 uvc 摄像头，在摄像头采集到一帧图像后，数据流入到两个 yolo 节点，在其中完成弹药箱识别与位置反馈。其数据流图如下：



（2）识别方案

在弹药箱的检测中，我们使用了分辨率不同的两个 yolo，分别为低分辨率 220 x 220 与 高分辨率 800 x 800，其中低分辨率 yolo 用于抓取弹药箱时的位置微调，高分辨率 yolo 用于弹药箱位置预测。

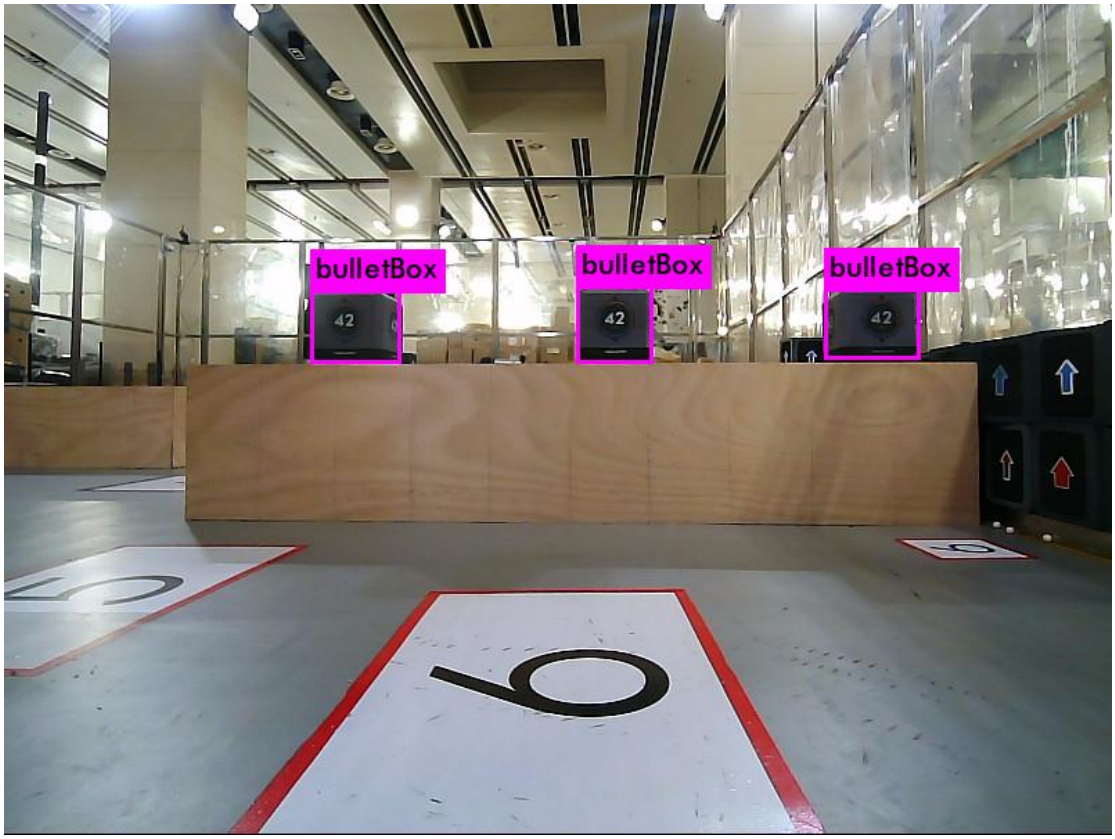
1）低分辨率 yolo 可以分辨出两米范围内可视的弹药箱，效果如下：



将 yolo 框出部分的中心点与图像中心线的偏差 x_offset 近似作为实际空间中弹药箱位置相对车体中心线的偏移量，根据这一偏移量对车体位置进行微调，使其在作夹取动作时车体中轴线近似正对弹药箱，以保证夹取成功。

在实地测试中发现，在微调时，yolo 可能会识别到将要抓取的弹药箱后面背景内的弹药箱，我们通过判断框的宽度大小与弹药箱在 y 轴的偏移量 y_offset 来滤掉这一干扰。

2）高分辨率 yolo 可以分辨出五米范围内可视的弹药箱，效果如下：



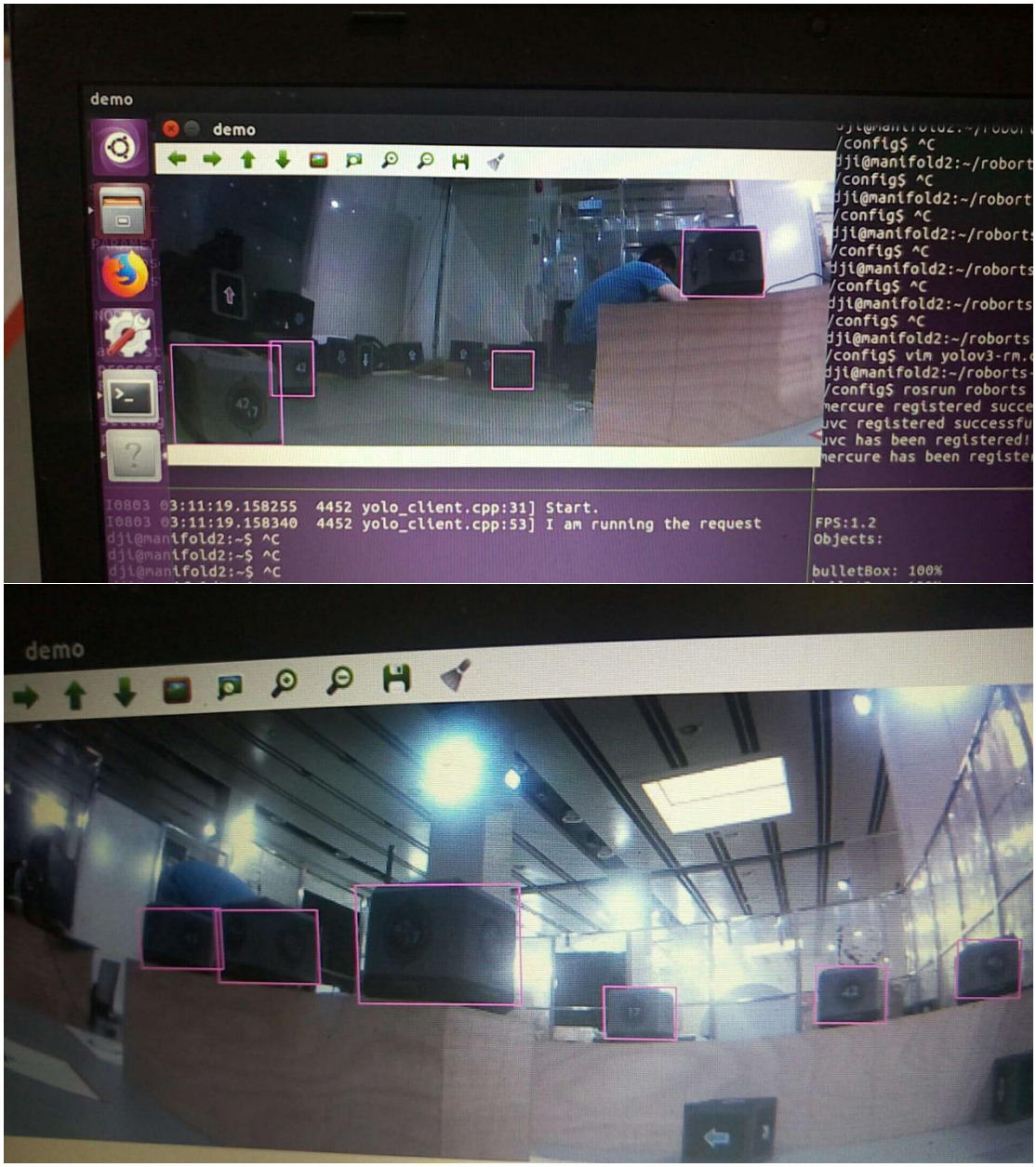
根据固定位置采集到的图像中弹药箱编号与其像素位置的对应关系这一先验信息，可以在比赛中通过视野中弹药箱的像素位置还原出其编号，反馈给决策节点后，步兵可以根据地图信息导航到弹药箱附近，之后进行微调并作抓取。

在图示情形下，在识别之后会将 13、14、15 位置有弹药箱这一信息作为一个话题发布出去，决策节点会接受这一信息。

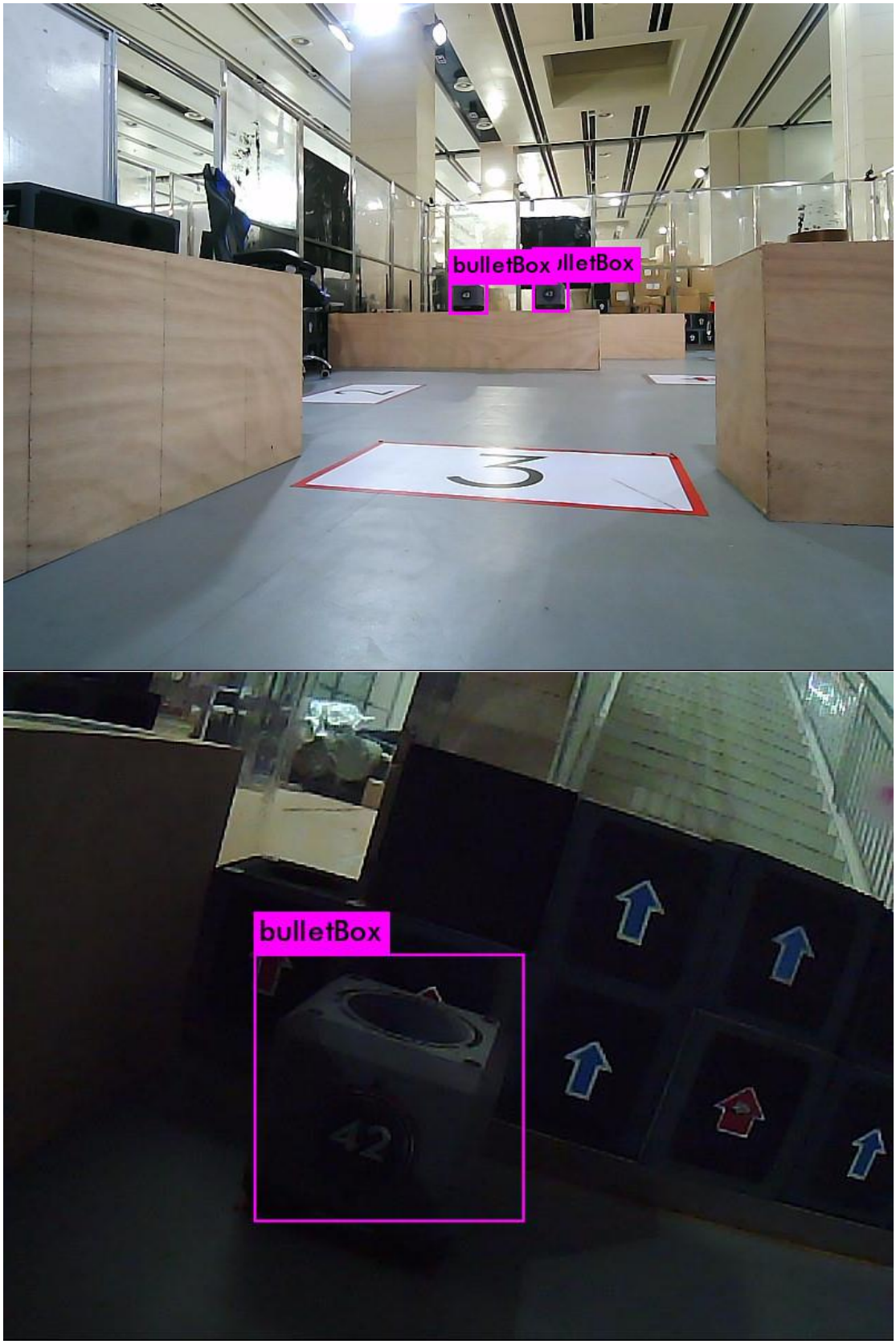
测试结果验证了这一方案的可行性。

3. 识别效果

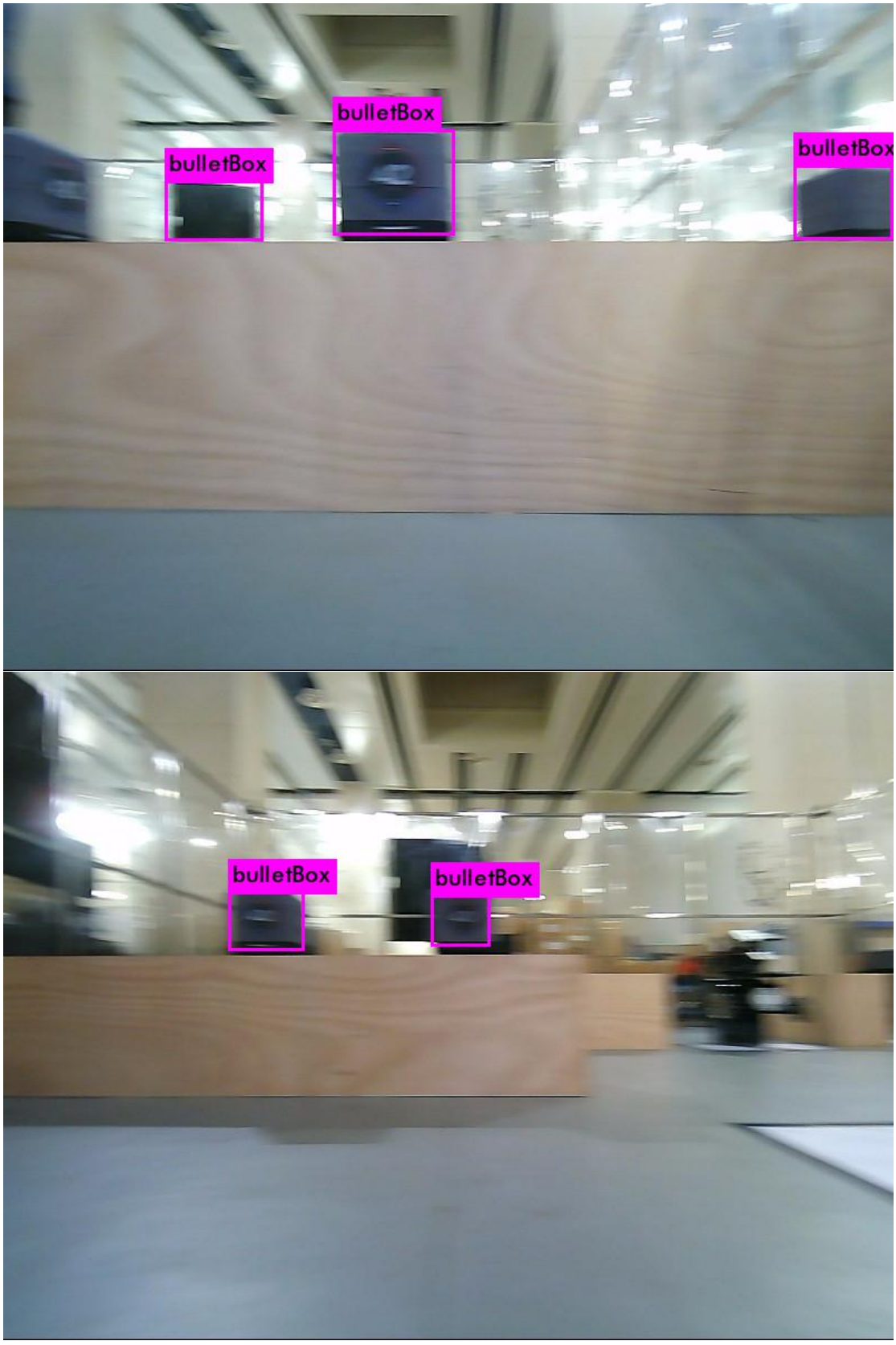
（1）正常情况下的识别效果

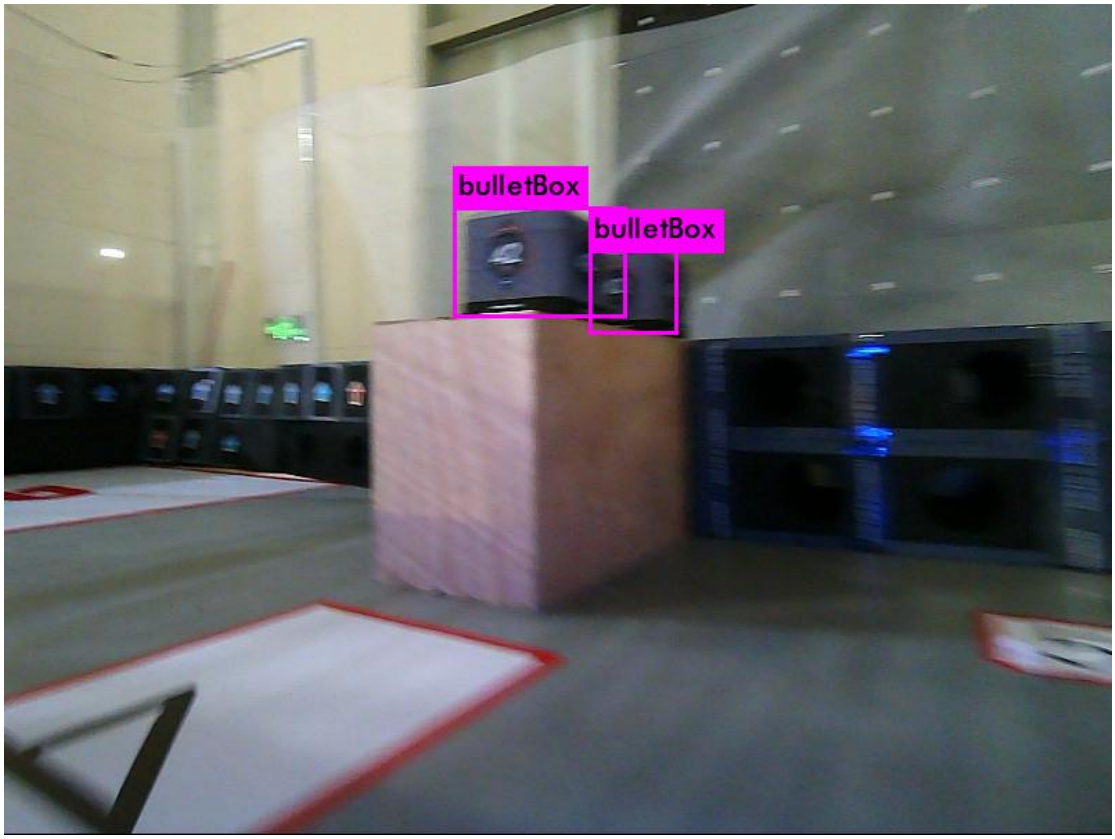


（2）极端条件下的识别效果


bulletBox
bulletBox


bulletBox
bulletBox


bulletBox
bulletBox
bulletBox
bulletBox
bulletBox

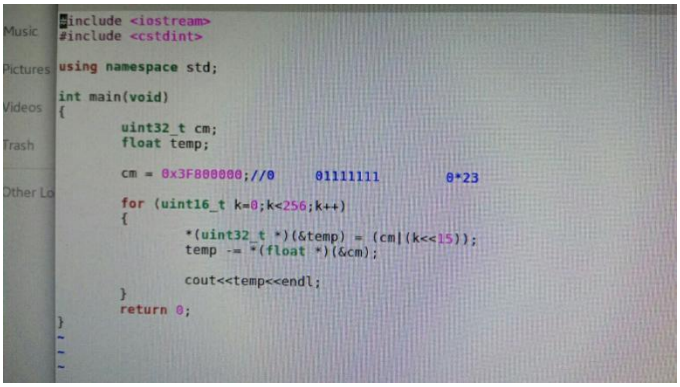


4. 针对 yolo 的几条建议

（1）对于 yolo,其浮点化效率极低，输入图像占用时间耗时 1.2ms+，可以考虑采用多种方法降低耗时，如使用 SIMD 向量化指令

```
.global neon_16tofloat
.syntax unified
.arm
.fpu neon
.type neon_16tofloat, %function
neon_16tofloat:
ldr r3, =0x37000000
vdup.32 q4,r3
loop1:
   vld4.16 {d0,d2,d4,d6}, [r0]!
   vmov.I64 d1, #0  @fill d1 with 0
   vmov.I64 d3, #0
   vmov.I64 d5, #0
   vmov.I64 d7, #0
   vzip.16 d0, d1  @interleave d0 and d1
   vzip.16 d2, d3
   vzip.16 d4, d5
   vzip.16 d6, d7
   vcvt.F32.U32 q0, q0, #16
   vcvt.F32.U32 q1, q1, #16
   vcvt.F32.U32 q2, q2, #16
   vcvt.F32.U32 q3, q3, #16
   vadd.F32 q0, q0 ,q4
   vadd.F32 q1, q1 ,q4
   vadd.F32 q2, q2 ,q4
   vadd.F32 q3, q3 ,q4
   vst4.32 {d0,d2,d4,d6}, [r1]!
   vst4.32 {d1,d3,d5,d7}, [r1]!
   sub r2, r2, #16
   cmp r2, #0
   bgt loop1
bx lr
.size neon_16tofloat, .-neon_16tofloat
.align 2
.global neon_floatto16
.syntax unified
.arm
.fpu neon
.type neon_floatto16, %function
```

或丢失映射均匀性和精度的基础上简化操作，如移位减组合（未呈现 SIMD 化结果）。



（2）内置的 Yolo 网络基于 darknet 框架，其虽然没有多少外部支持，但是也没有为嵌入式平台，尤其是 TX2 上优化。可以尝试转换 Darknet->Keras->caffemodel，使用 TensorRT 加速运行。Yolov3 仅仅需要手工添加一个检测层即可实现。TensorRT 就目前公开数据看（尤其是对于嵌入式平台）优化的效果极佳，而且也有人在 Github 尝试 yolov3-on-TensorRT，但是还没有完成，夏令营时间紧迫就没有继续研究。
（3）以上都写于刷夜后的一个早上，对于内容不负责，我也觉得大概使不了，但是不失为一个优化的方向。

以上 yolo 部分由夏令营三组刘明轩编写（兵哥哥贼酷）

3.4.3 单兵逻辑

简单介绍下接下来会用到的名词：

行为树的节点类型

Selector 节点：将子节点按优先级从高到低执行，直至某一子节点返回成功状态，该 Selector 节点状态即为成功。当优先级低的子节点在执行过程中有某一优先级高的子节点突然满足执行条件，优先级高的子节点将有执行优先权，并会打断所有优先级低的同级子节点的执行。

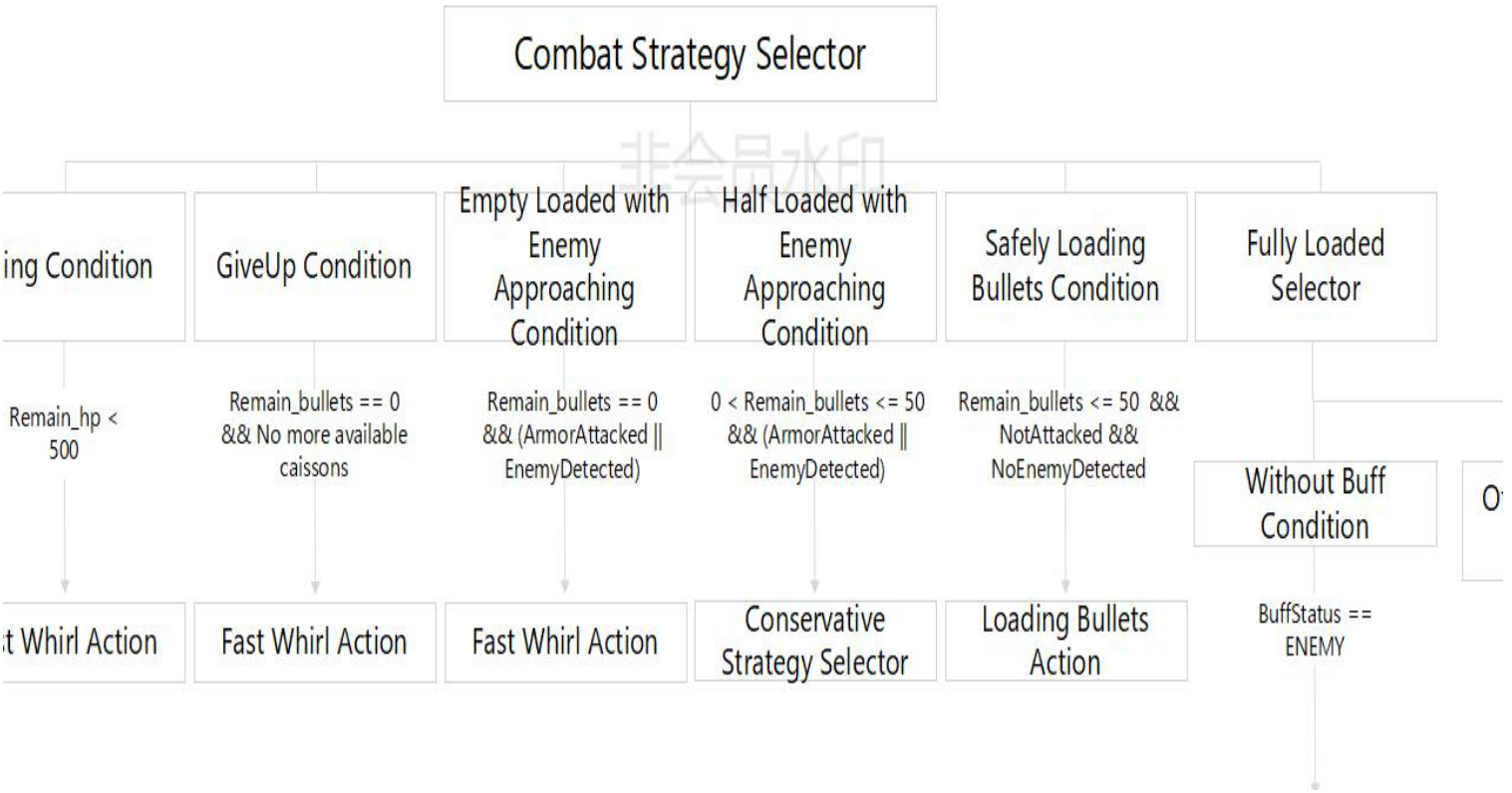
Sequence 节点：将子节点按优先级从高到低执行，只有前一子节点返回成功状态后下一个子节点才会被执行，直至所有子节点执行完毕并返回成功状态，该 Sequence 节点状态即为成功。

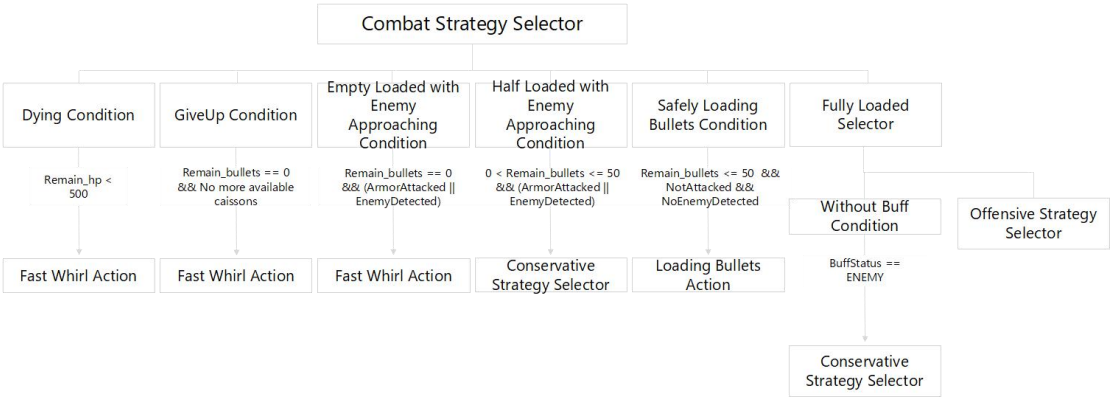
Precondition 节点：判断执行条件；如果不满足则返回失败，如果满足则指向对应的 Action 节点。

Action 节点：包含具体的行为函数。

3.4.3 单兵逻辑

- A. 战略选择决策树 - 普通版本





A. 战略选择决策树 - 普通版本

战略的选择由一个Selector节点完成。各个子节点的具体行为如下，优先级按从高到低排列：

- （1）残血状态：
  - 判断条件：血量 < 500
  - 行为：高速原地自旋，转速为5。
- （2）无弹无补给状态：
  - 判断条件：弹量为零，且已确认指定取弹区无弹可取。无法再恢复战斗力。
  - 行为：高速原地自旋，转速为5。
- （3）无弹且发现敌方靠近状态：
  - 判断条件：弹量为零，且发现敌方或被敌方攻击。无法反击。
  - 行为：高速原地自旋，转速为5。
- （4）有一箱弹且发现敌方靠近状态：
  - 判断条件：弹量大于零小于等于50，且发现敌方或被敌方攻击。可以反击，但战斗力不强。
  - 行为：保守打法。
- （5）弹量储备小于两箱状态：
  - 判断条件：弹量小于等于50。即无弹或只取过一箱弹。
  - 行为：取弹 （详见3.4.4）
- （6）弹量大于等于两箱状态：
  - 6.1 如果敌方获得buff，则采取保守打法（详见3.4.3 D）
  - 6.2 如果敌方未获得buff，则采取激进打法 （详见3.4.3 C）

实现效果：

（1）在弹量不足两箱时，步兵车会前往取弹区取弹，取弹过程中一旦发现敌方靠近或被攻击，会根据现有弹量选择反击或自保。

（2）一旦取完两箱子弹，步兵车会根据敌方是否有buff加持来选择采取保守打法或激进打法。

（3）在打击过程中弹量逐渐下降，在弹量小于50时，一律切换为保守打法。

（4）在弹量小于50且敌方暂无威胁的情况下，步兵车会返回我方取弹区继续补弹。

（5）补弹完成后步兵会重新返回战场进行打击。

（6）如果战斗过程中步兵车受到严重伤害，进入残血状态，步兵车会放弃战斗，高速自旋直至比赛结束。

3.4.3 单兵逻辑

- B. 战略选择决策树 - Rush

B. 战略选择决策树 - Rush版本

Rush战略是在普通战略的基础上变得更激进了一点。战略的选择依然由一个Selector节点完成。各个子节点的具体行为如下，优先级按从高到低排列：

- （1）残血状态：
  - 判断条件：血量 < 500
  - 行为：高速原地自旋，转速为5。
- （2）无弹无补给状态：
  - 判断条件：弹量为零，且已确认指定取弹区无弹可取。无法再恢复战斗力。
  - 行为：高速原地自旋，转速为5。
- （3）无弹且发现敌方靠近状态：
  - 判断条件：弹量为零，且发现敌方或被敌方攻击。无法反击。
  - 行为：高速原地自旋，转速为5。
- （5）无弹状态：
  - 判断条件：弹量小于50。
  - 行为：取弹 （详见3.4.4）
- （6）有弹状态：
  - 6.1 如果我方获得buff，则采取激进打法（详见3.4.3 D）
  - 6.2 如果比赛进行到最后一分钟，我方血量高于2000，则采取激进打法。
  - 6.2 其余情况采取保守打法 （详见3.4.3 C）

实现效果：

（1）在弹量不足一箱时，步兵车会前往取弹区取弹，取弹过程中一旦发现敌方靠近或被攻击，会立即高速自旋进行自保。

（2）一旦取完一箱子弹，步兵车会根据我方是否有buff加持来选择采取保守打法或激进打法。

（3）在打击过程中弹量逐渐下降，在弹量小于50且敌方暂无威胁的情况下，步兵车会回我方取弹区继续补弹。

（4）补弹完成后步兵会重新返回战场进行打击。

（5）如果比赛进行到最后一分钟，我方依然血量保持在2000+，一律切换为激进打法。

（6）如果战斗过程中步兵车受到严重伤害，进入残血状态，步兵车会放弃战斗，高速自旋直至比赛结束。

3.4.3 单兵逻辑

- C. 激进打法

C. 激进打法

激进打法的选择由一个Selector节点完成。各个子节点的具体行为如下，优先级按从高到低排列：

- （1）受到高频伤害状态：
  - 判断条件：每秒伤害 > 600 或无弹
  - 行为：逃跑。
- （2）发现敌人状态：
  - 判断条件：识别到敌人。
  - 行为：解算敌方位姿，进行追击。
- （3）受到打击状态：
  - 判断条件：某一装甲板反馈被打击信息。
  - 行为：转向被打击方向。
- （4）识别到敌方装甲颜色状态：
  - 判断条件：识别到敌方装甲颜色。

实现效果：

（1）发现敌人时会一边追击一边自瞄发弹。

（2）被打击后会迅速转向被打击的方向进行还击。

（3）在战斗过程中如果受到高频伤害则会迅速撤离交火区。

（4）在未发现敌方战车时会深入敌方区域搜索敌方战车。

（5）如果搜索失败则进行常规定点巡逻，等待敌方战车出现。

D. 保守打法

保守打法的选择也由一个Selector节点完成。各个子节点的具体行为如下，优先级按从高到低排列：

- （1）受到高频伤害状态：
  - 判断条件：每秒伤害 > 600 或无弹
  - 行为：逃跑。
- （2）发现敌人状态：
  - 判断条件：识别到敌人。
  - 行为：解算敌方位姿，进行远程吊射。
- （3）识别到敌方装甲颜色状态：
  - 判断条件：识别到敌方装甲颜色。
  - 行为：转向识别到的方向。
- （4）受到打击状态：
  - 判断条件：某一装甲板反馈被打击信息。
  - 行为：转向被打击方向。
- （5）巡逻行为：定点巡逻。
- （6）低速自旋行为：低速自旋，自旋速度2.5。设置低速的原因是保证车在自旋过程中依然能够识别到视线范围内的敌方车辆。

实现效果：

（1）发现敌人时会远程自瞄吊射。

（2）被打击后会迅速转向被打击的方向进行还击。

（3）在战斗过程中如果受到高频伤害则会迅速撤离交火区。

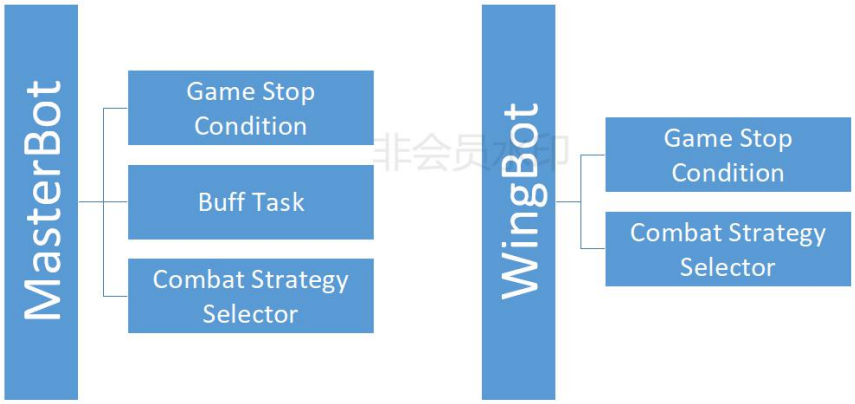
（4）在未发现敌方战车时会进行常规定点巡逻。

（5）如果路径规划失败，无法前往巡逻点，则进行原地低速自旋。

3.4.3 单兵逻辑

- D. 保守打法

3.4.4 多兵作战

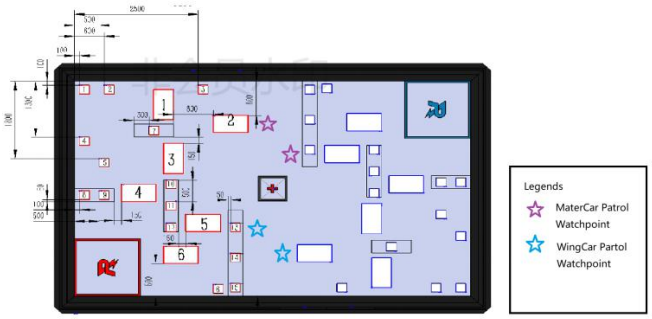


3.4.4 多兵作战

- A. 多兵任务分配
- （1）主车Masterbot：
  - 如果比赛未开始，则原地不动。
  - 在比赛前五秒倒计时阶段，开启雷达扫描功能，确定4-6号随机障碍块的存在与否，据此加载含有4-6号任意障碍块的地图，同时决定比赛开始后抢buff的路线。
  - 比赛一旦开始，若双方都未取得buff，则主车前往buff zone抢buff。到达buff zone后进行原地高速自旋直至buff被激活，保证敌方无法在这段时间内靠近主车或发起打击。
  - 一旦有任意一方获取buff，则buff task立即失效，主车接下来的战略选择由combat strategy selector决定。主车的combat strategy selector为Rush版本。一般表现为先回家取弹，取完一箱立即发起攻击，弹量耗尽后回家继续取弹，取完一箱再返回战场，直至所有弹药箱被取完。（详见3.4.3 B）
- （2）从车Wingbot：
  - 如果比赛未开始，则原地不动。
  - 在比赛前五秒倒计时阶段，开启Yolo识别功能，确定10-15号弹药箱的存在与否。同时开启雷达扫描功能，确定4-6号随机障碍块的存在与否，据此决定比赛开始后取弹药箱的顺序与位置。
  - 比赛一旦开始，从车接下来的战略选择由combat strategy selector决定。从车的combat strategy selector为普通版本。一般表现为先回家取弹，取完两箱发起攻击，弹量耗尽后回家继续取弹，再取一箱即返回战场，直至所有弹药箱被取完。（详见3.4.3 A）

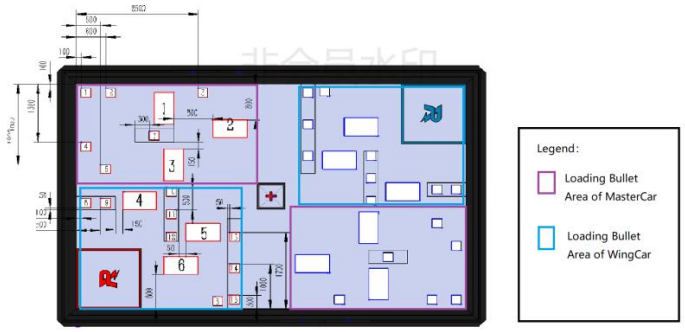
3.4.4 多兵作战

- B. 多兵巡逻埋伏配合
- 两车分别埋伏于中心点两侧，位于敌方战车离开己方阵营前往对方阵营的必经之路上。敌方战车一旦出现便可实现两侧同时打击。



3.4.4 多兵作战

- C. 两车弹仓分配
- 主车完成Buff Task后返回己方阵地来取地面上的弹药箱，外加7号弹药箱（紫色方框）；从车负责取高台上的9-15号弹药箱（蓝色方框）。这种分配方式可保证每车至少有2箱弹药可取，又防止了两车因取弹路线交集发生撞车。



以上 3.4.3-3.4.4 由学妹(钱愉盈)编写 大一学妹贼强

5， 夏令营感想、总结、

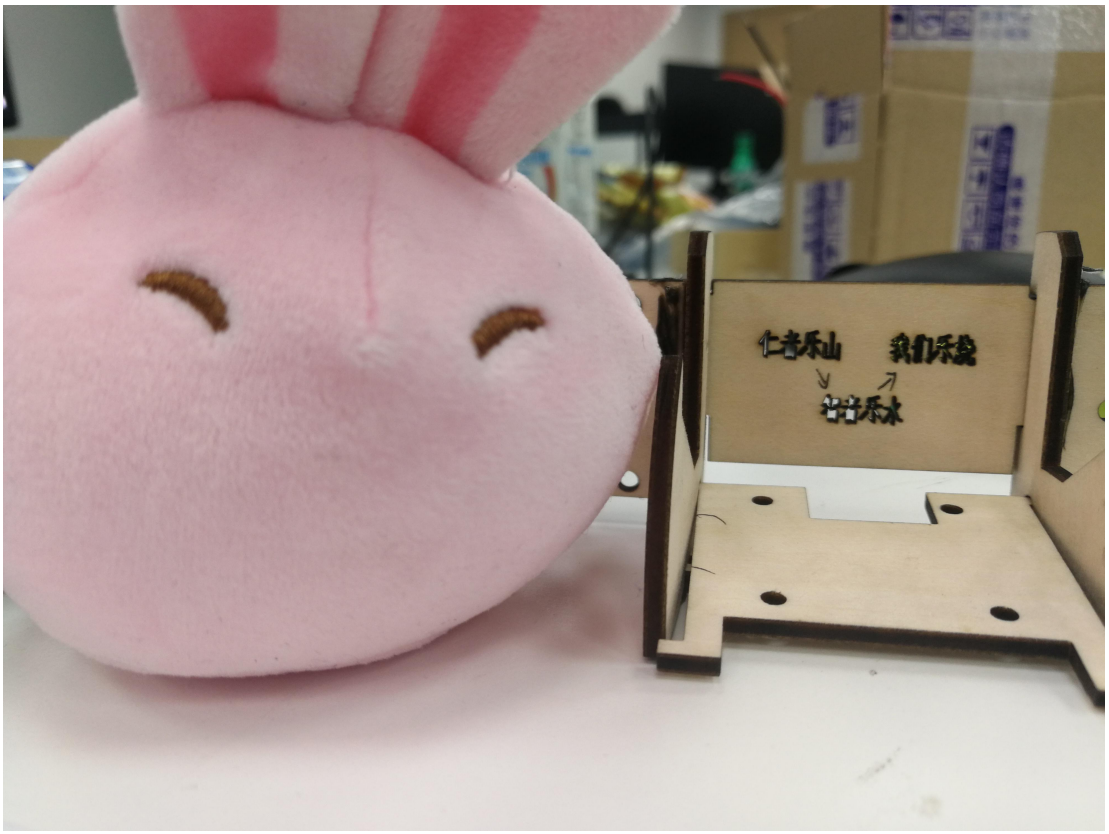
Robomaster2018 夏令营历时 43 天，据说是最长的一届夏令营，但是为什么还是感觉时间不够啊，除了第一周的参观与破冰赛以外。。感觉每天至少都是 2 点回去，每天只有睡醒，来集成的时候才能感受一波深圳夏日的阳光，可能这就是夏天最好的避暑方法了吧，不让你见到太阳。当然 Robomaster2018 夏令营的确是一个很棒的平台，一堆很有难度的技术题目，一个来自世界各地的技术精英团队，一大堆能让你随意使用的价值好几万的物资，再加上自由支配的项目资金和 24 小时开放的加工设备/场地，哇，研发环境的天堂了解下。

回顾我们的夏令营经历，我们团队在第一阶段，没有意识到整个项目的难度，于是采取了很极端的技术方案，还有堪称工作量最大的节点方案。嗯，果不其然，我们翻车了。我们在第一阶段要比赛当天的早上 4 点钟完成了测试（10 次测试 9 次成功），我们以为很稳了，但是在回去休息以后再回来比赛的时候，我们的方案就凉了。因为我们没有考虑到场地的变量，当天的场地和我们测试的时候略有不同，被大妈刚刚拖过的地，让我们的麦轮上的小轮子打滑，产生了累计偏移。跑飞了，夹取失败，大家回来都十分沮丧，但是很快又反应过来，这是一个好事，当时我们的方案的确太激进了，如果我们第一阶段得到了好成绩，我们将继续沿用这种极端的方案，那么在我们第二阶段再根据这个车体进行改装，那么我们就会因为既有程序与我们新车的不匹配而失去我们相对于其他组的优势并且浪费我们第二阶段的测试时间，于是在第二阶段，我们全面摒弃第一阶段的激进方案，重新再进行总方案讨论设计，将研发项目块化，并行开发。加工时间挤压推前为测试留时间（幸苦家硕老哥。为了抢到整块的加工时间，我和他只能每天白天设计方案/审图，晚上彻夜加工，装配。PS：家硕胁迫我说：“必须加

上一句！我们机械大佬说他回去以后要买雕刻机！买激光切割机”嗯，我加了），把时间留给测试和整体算法系统的构建（当然嵌入式和算法也非常给力啊，任务只留给他们一小点冗余量但是依然完成了），总的一个思路，多测试，多拟合系统，让所有的流程走通，但是也留出来准备给我们流程失误的时候。2 天一次方案会议，4 天一次的技术审核会，把我们技术方案潜力完全压榨了出来。虽然最后，我们距离第一名只有一点点血量差距，但是我们为我们能在前期发生这么多的失误（中间因为瞎拿线，没检查端子而插电短路而烧了大概 2w 的东西=。=）的情况下依然能取得如此成绩而自豪。

谈谈关于项目的总结：整个夏令营的题目是个非常难得题目，去年的 icra 会议这么多人做，最后的实现效果，也不太佳。但是我们得在去年 icra 的题目上再加上我们功能，再根据我们的理解将整个流程优化，有难度，放游戏里这也是个世界任务啊，但是我们得在一个月的时间内将他做出来，而且得做好，做实用（2v2 竞赛），当然在有 RM 大佬无时无刻的帮助的情况下，我们将整个比赛所需的东西做出来了，而且个人感觉还不错（感恩硕哥，亚楠哥，向勇哥，李老师，帆哥，一俊（罗一俊永远是我大哥！）董大师，盼姐，仓管磊哥，以及每一位帮助过我们的人，再次感恩给予我们的帮助）。

完美的夏令营，谢谢谢谢谢谢



以上归总总结由苟鑫归总 （RM2018 夏令营三组）

RoboMaster 大赛组委会

邮箱：robomaster@dji.com

官方论坛：http://bbs.robomaster.com

官方网站：http://www.robomasters.com

电话：075536383255（周一至周五 10:00-19:00）

地址：广东省深圳市南山区西丽镇茶光路 1089 号集成电路设计应用产业园 2 楼 202

微信

微博