目录

[1 系统方案 4](#_Toc86503197)

[2 硬件设计 4](#_Toc86503198)

[2.1 器件 4](#_Toc86503199)

[2.1.1 人工智能控制器 4](#_Toc86503200)

[2.1.2 光电传感器 5](#_Toc86503201)

[2.1.3 红外测距传感器 6](#_Toc86503202)

[2.1.4 Intel第二代神经棒 7](#_Toc86503203)

[2.1.5 电机，驱动器 7](#_Toc86503204)

[2.2 结构设计 8](#_Toc86503205)

[2.2.1 整体结构 8](#_Toc86503206)

[2.2.2 传感器布局 9](#_Toc86503207)

[2.2.3 攻击部件 9](#_Toc86503208)

[2.2.4 上台结构 11](#_Toc86503209)

[2.2.5 底板设计 13](#_Toc86503210)

[3 软件设计 15](#_Toc86503211)

[3.1 总体概述 15](#_Toc86503212)

[3.2 相关技术 17](#_Toc86503213)

[3.2.1 多进程：Python Multiprocessing 17](#_Toc86503214)

[3.2.2 Apriltag 17](#_Toc86503215)

[3.2.3 目标检测：YOLO v5 17](#_Toc86503216)

[3.2.4 图像翻译：DeblurGAN-v2 18](#_Toc86503217)

[3.3 代码分析 18](#_Toc86503218)

[3.3.1 main.py 19](#_Toc86503219)

[3.3.2 kamera.py 19](#_Toc86503220)

[3.3.3 my\_controller.py 20](#_Toc86503221)

[3.3.3.2 def update\_frame 22](#_Toc86503222)

[3.3.3.3 def first\_check 22](#_Toc86503223)

[3.3.3.4 def upstage 22](#_Toc86503224)

[3.3.3.5 def get\_block\_location 22](#_Toc86503225)

[3.3.3.6 def move\_xx & def turn\_\_XX\_\_fast & turn\_XX\_slow 22](#_Toc86503226)

[3.3.3.7 def edge\_detect 23](#_Toc86503227)

[3.3.3.8 def upstage\_direction\_adjust 23](#_Toc86503228)

[3.3.3.9 def find\_energy\_block 24](#_Toc86503229)

[3.3.3.10 def find\_normal\_block 25](#_Toc86503230)

[3.3.3.11 def attack 25](#_Toc86503231)

[3.3.4 Yolov5.py 25](#_Toc86503232)

[3.4 效果展示 26](#_Toc86503233)

[4 总结与心得 27](#_Toc86503234)

[参考文献 28](#_Toc86503235)

对对队 设计报告

设计者：韩帅伊 周添 赖金熠

指导教师：方文倩

武汉大学大学生工程训练与创新实践中心 武汉430072

**摘 要：**

团队以创意之星人工智能版控制器为控制核心，以Solidworks为设计工具，设计了轮式格斗机器人，该机器人能够完成自主上台、判断自身状态、识别能量块、追踪敌方小车，进攻与防守等功能。机器人整体机械框架及结构通过Solidworks设计完成，并以白色高强度光敏树脂为零部件材料3D打印完成。机器人通过红外光电传感器，红外测距传感器和摄像头采集外部信息，其中红外测距传感器配合摄像头追踪敌人；由红外测距传感器和红外光电传感器共同检测周围信息来判断机器人在台上还是在台下、在台下时的朝向、在台上是否靠近边缘等；控制器基于滤波算法、模数转换处理传感器数据，基于目标检测与图像翻译处理图片并将两者数据融合，反馈到电机驱动单元，来控制机器人的运动。机器人运动稳定、目标识别准确率很高、自身状态判断准确。

**关键词：**多进程，YOLOv5，DeplurGANv2，3D打印，力学结构

# 系统方案

格斗机器人采用四轮驱动结构，用白色高强度光敏树脂作为主要材料，通过3D打印制作机身。机器人主体有两层面板，第一层用来放置电机、驱动器；第二层用来放置电池、传感器、控制器等器件。机身前方设计一下半部分是斜三角形，上半部分为弧形的铲子作为攻击武器，具有较好的贴地性和攻击性。后方设计三角形结构，用来实现从后方登台。作品机身分布八个红外光电传感器，五个红外测距传感器和一个摄像头，两个光电传感器置于前方，两个光电传感器置于后方，从上到下检测，用于识别擂台边缘；前，后，左，右各放置一个光电传感器，后方放置三个测距传感器，前，左，右各放置一个测距传感器，正前方放置一个摄像头，配合使用根据返回的参数检测外部环境，判断小车状态，监控敌方位置。控制器中载入团队成员自主编写的代码，根据传感器和摄像头判断机器人自身和周围的不同状态，从而执行不同操作。控制器、驱动器、传感器等零器件均符合统一部件要求。

机器人通过软开关启动之后，直接执行登台操作，如果检测到位于擂台边缘，则会后退；如果不在边缘，则会检测擂台上二维码，和敌方小车的位置，根据敌方小车的方向选择优先进攻中立能量块还是敌方能量块；当除己方能量块都被推下擂台之后，则会调整自身车位朝向，追踪敌方小车，发起进攻。将敌方小车推下台后，会执行守台功能，阻止敌方小车登台；如果机器人自身被击落擂台，则根据传感器返回的数据，调整小车位置，使小车迅速背对擂台，实现快速登台。

# 硬件设计

## 器件

### 人工智能控制器

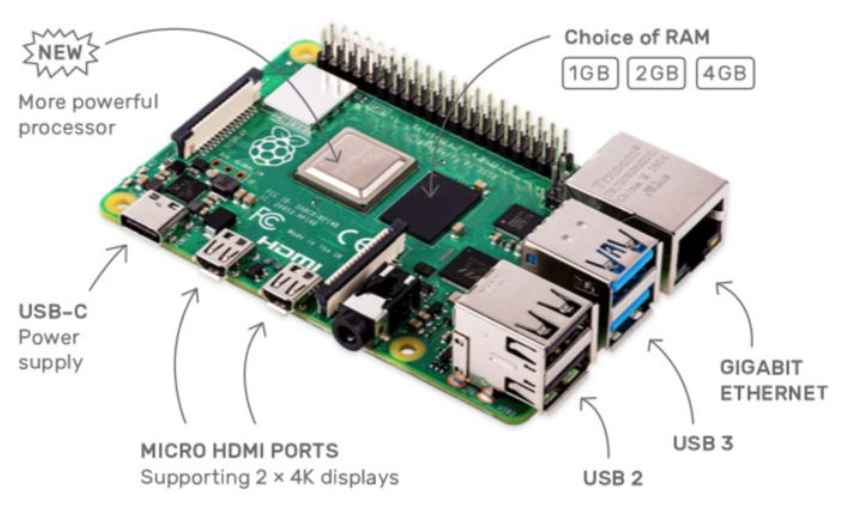
树莓派4B在处理器速度，多媒体性能，内存和连接方面提供了突破性的增长，同时保留了向后兼容性和类似的功耗。对用户来说，树莓派4B提供的桌面性能可与入门级x86 PC系统相媲美。树莓派4B的主要功能包括高性能64位四核处理器，通过一对micro-HDMI端口支持分辨率高达4K的双显示屏，高达4Kp60的硬件视频解码，高达4GB的RAM，双频2.4/5.0 GHz无线局域网，蓝牙5.0，千兆以太网，USB 3.0和PoE功能（通过单独的PoE HAT插件）。双频无线局域网和蓝牙具有模块化合规认证，允许将电路板设计到最终产品中，大大降低了合规性测试，从而降低了成本和上市时间。

图2.1树莓派4B效果图

创意之星人工智能版控制器采用树莓派 4B 作为主控，加装的拓展板是拓展出了多种接口，供电及CDS5516 总线舵机接口，ADC 传感器接口，IO 口，WS2812B 指示灯MPU65001.96 寸彩色液晶等，实现电机控制，传感器读取等。

### 光电传感器

光电传感器是数字量传感器，采用E18-D80NK，是利用被检测物对光束的遮挡或反射,由同步回路选通电路,从而检测物体的有无。物体不限于金属,所有能反射光线的物体均可被检测。光电开关将输入电流在发射器上转换为光信号射出,接收器再根据接收到光线的强弱或有无对目标物体进行探测。工作原理如图2.2所示。多数光电开关选用的是波长接近可见光的红外线光波型,因此也称为红外开关。

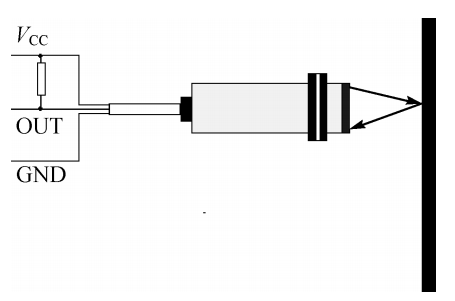


图2.2光电传感器原理图

在本系统中，采用的是漫反射式光电传感器，它是一种集发射器和接收器于一体的传感器,当有被检测物体经过时,物体将光电传感器发射器发射的足够量的光线反射到接收器,于是光电传感器就产生了信号。

无障碍物时，传感器自带灯不亮，输出引脚为高电平（5V），检测到障碍物

时，传感器自带灯点亮，输出低电平，传感器检测范围可以通过旋转传感器上的电位器调节。阈值同样受到环境温度、光线等因素的影响。

在机器人的前方和后方左右各设置一个光电传感器，用来检测前进时是否到达擂台边缘，其实在测试过程中发现后方的光电传感器几乎没用作用，但是为了避免某些特定角度可能导致小车坠台，我们选择保留该部分。其次在前后左右各设置一个传感器用来检测机器人是否接触到障碍物。调整光电传感器阈值，使其检测范围在35~40cm。

### 红外测距传感器

红外测距传感器是模拟量传感器，采用SHARP GP2Y0A41SK，主要是由红外发射器、PSD(位置敏感检测装置)及相关处理电路构成,红外发射器发射一束红外光线,红外光线遇到障碍物被反射回来,通过透镜投射到 PSD 上,投射点和 PSD的中心位置存在偏差值a,根据如图2.3所示的a、b、α三个值就可以计算出 H 的值,并输出相应电平的模拟电压。

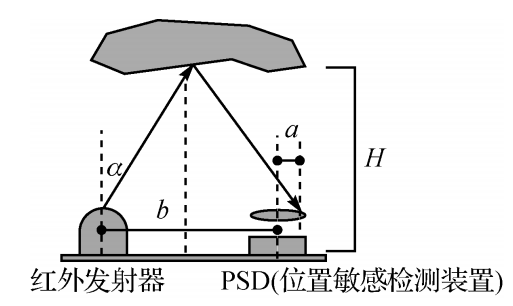


图2.3红外测距传感器原理图

红外测距传感器的工作性能与障碍物的反射角度基本无关，与反射物体的颜色及材质基本无关，但是,有效测量距离是随被测物体材质而不同的,例如对于白纸,最大有效测量距离可达到80cm;但是对于黑色皮革,有效测量距离可能只能达到60~70cm。这是由于不同材质的反射率不同所致。在本次实验中，目标物为棕黄色纸箱，有效距离应在70~80 cm。

物体距离传感器越近，读取得数值越大（测量电压范围 0~5v，对应输出值 0~1023。） 物体距离传感器越远，读取数值越小。可以作为红外光电传感器的补充，可以在更加复杂的环境中准确获知环境的情况。但是红外测距传感器返回值不稳定，跳变很大，需要用滤波算法将周围干扰过滤使其有一个稳定的返回值，在此机器人中主要用在：自主上台对擂台和围栏距离的测量，以便于调整好姿态

成功上台。其次，对擂台上的目标，在光电传感器的基础上得到准确的距离，可以根据不同的情况做出不同的决策，提高整体的稳定性。

### Intel第二代神经棒

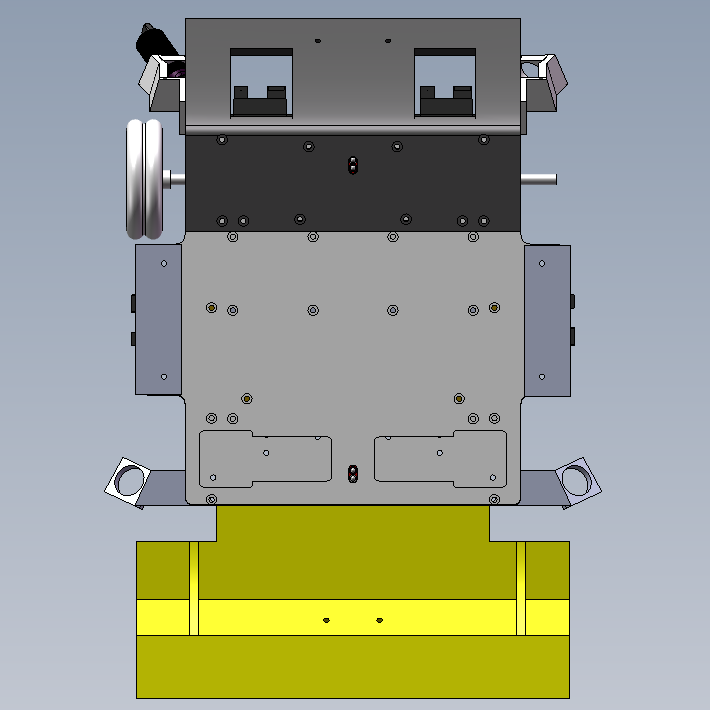
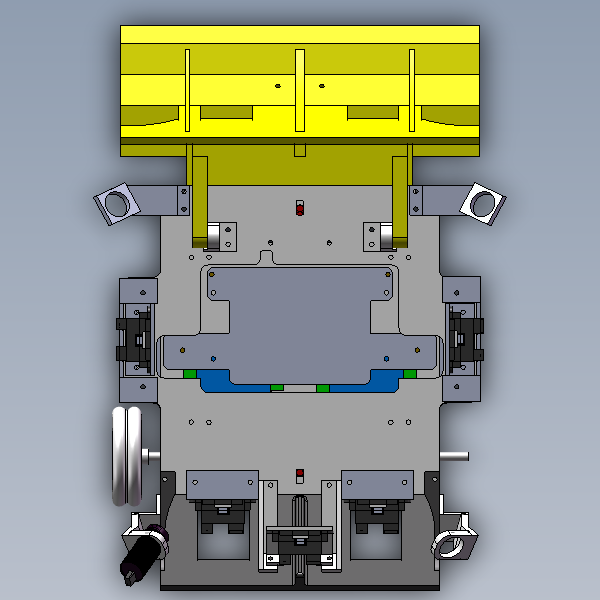
第二代神经计算棒(Neural Compute Stick 2/NCS 2) 尺寸只有72.5×27×14毫米，通过USB 3.0 Type-A接口插入主机，兼容64位的Ubuntu 16.04.3、CentOS 7.4、Windows 10操作系统。NCS 2内置了最新的Intel Movidius Myriad X VPU视觉处理器，集成16个SHAVE计算核心、专用深度神经网络硬件[加速器](https://xiazai.zol.com.cn/netaccelerate_soft_index/netaccelerate_page_1.html)，可以极低的功耗执行高性能视觉和AI推理运算，支持TensorFlow、Caffe开发框架。英特尔NCS 2不需要额外的硬件，因此用户可以快速转换并将计算机训练模型无缝部署到各种各样的设备上，并且无需网络或云端连接。借助英特尔NCS 2，计算机视觉和人工智能可以轻松地部署到物联网和边缘设备原型上。

### 电机，驱动器

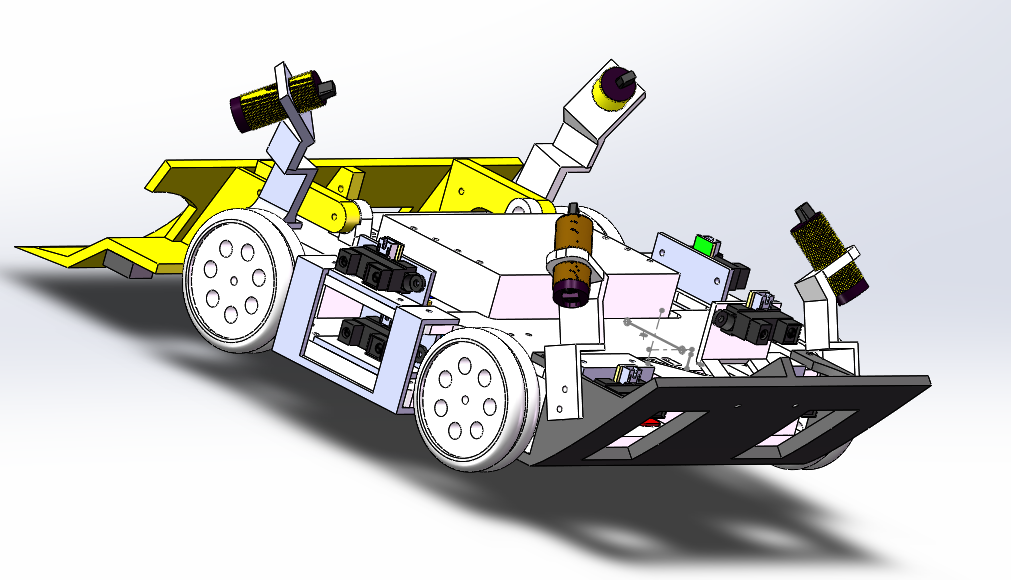
采用博创的BDMC1203驱动器和FAULHABER 234224CR电机为机器人提供动力，同侧的两个车轮并联，始终保持相同的速度，分别控制两侧的车轮转速可以实现前进、倒退、转弯、原地旋转等动作。

## 结构设计

### 整体结构

要在比赛开始后迅速完成登台等一系列操作，机器人应轻巧灵活，前进、后退、转动动作应自如流畅，并具有较快的行进速度。因此我们选择自主设计机器人的整体结构，通过SolidWorks 2019软件上进行三维建模完成设计。模型如图下图2.4、2.5、2.6所示：

2.4 机器人俯视示意图 2.5机器人底部示意图

2.6 机器人侧面整体示意图

机器人整体设计呈赛车式的扁平形态，高度略高于擂台，重心低故而稳定性很高。同时，由于双方机器人在后续攻防过程中将发生激烈的冲撞，进行制约与反制约的较量，因此要求机器人结构有一定的刚度，因此我们选择白色高强度光敏树脂作为打印材料。

其他一些连接或固定用的小构件也是自己根据需要进行专门加工制作。

### 传感器布局

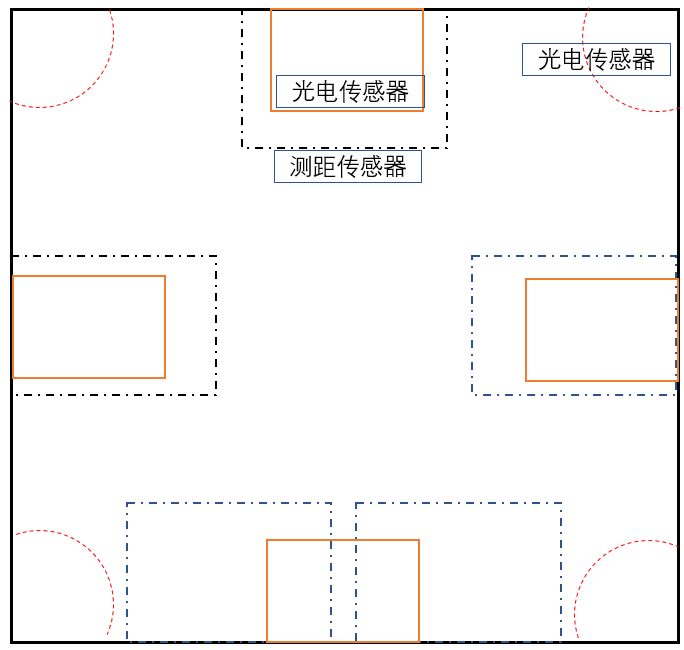
结合不同传感器的特点我们选择红外光电传感器和红外测距传感器，机器人传感器的布局见图2.7。检测是否到达擂台边缘的两个光电传感器安装在机器人的左前方和右前方，左后方，右后方，斜向下指向地面。用于上台和检测目标的光电传感器和测距传感器安装在机器人的前后左右四个方向。

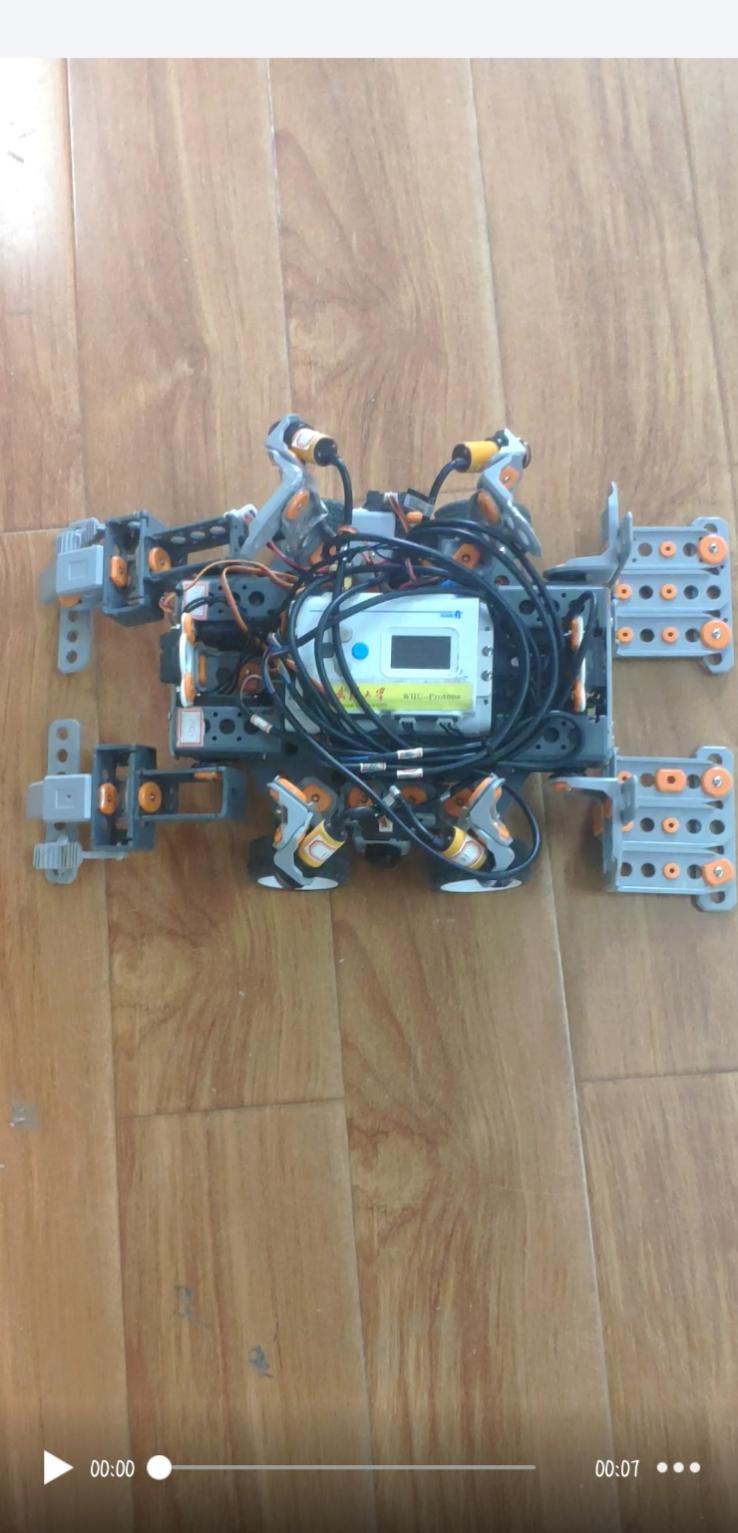
图2.7传感器布局

图中黑色虚线部分代表红外测距传感器，橙色图形代表光电传感器。

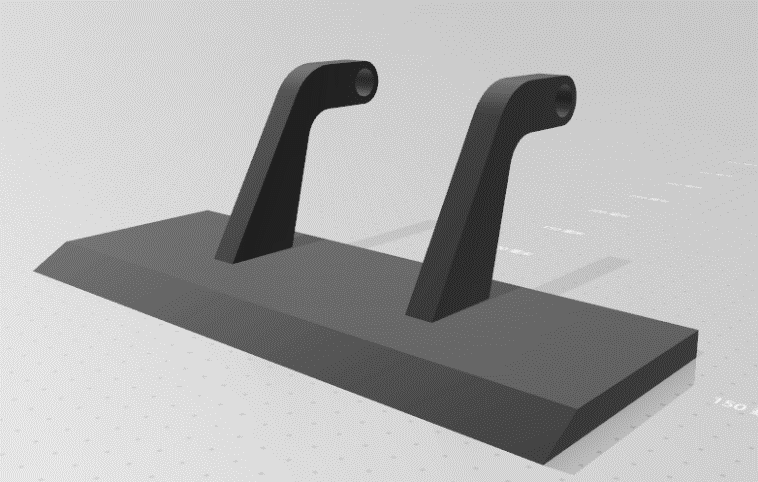
利用擂台和围栏的高度差，同一方向的传感器，红外测距传感器在下方，光电传感器在上方通过传感器不同的返回值判断机器人朝向。同时，由于小车采用后冲登台，因此小车是否正对擂台对登台成功率有着很显著的影响。为了解决小车没有正对擂台，导致登台失败，我们在后面改为双红外测距传感器，利用传感器的差值，调整小车朝向，使小车正对擂台。

### 攻击部件

智能格斗机器人的主要攻击方式是通过从正面或者侧面将敌方机器人铲起，使其离开地面甚至将其撞翻，所以在设计前铲时着重对其贴地性和强度进行考虑

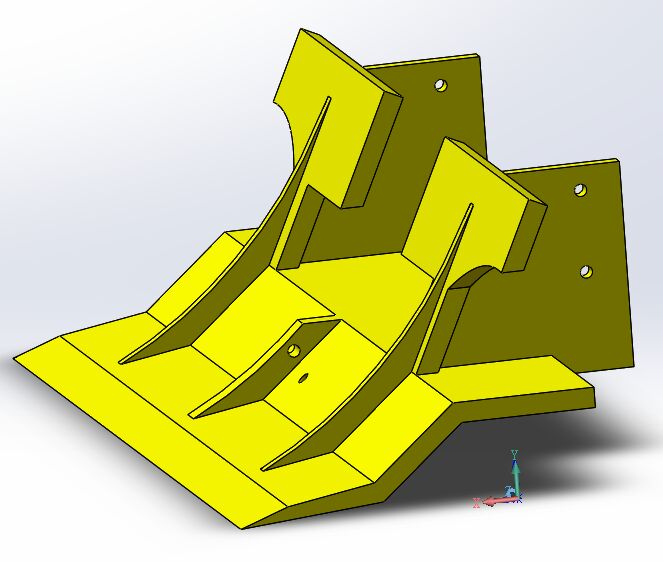
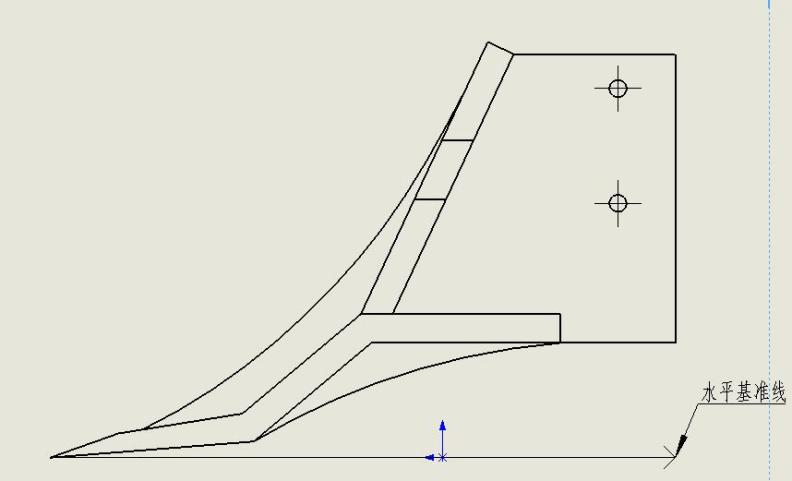
方案一：使用如图2.8所示前方的两个铲子，分别由左右两个舵机控制。机器人漫游时，铲子贴住地面，当遇到目标物体时，机器人会将目标从底部铲起，使其失去与地面的接触，借助自身的动力将目标推向擂台下。这种结构铲子贴地不够，重量轻，容易被对手铲起，且较为脆弱，在对抗结构中易损坏。

2.8 方案一铲子结构

方案二：使用如图2.9所示的可活动的铲子，安装在机器人的正前方。正常状态下铲子会紧贴地面，当遇到目标物体时，机器人会将目标从底部铲起，使其失去与地面的接触，借助自身的动力将目标推向擂台下。这种结构虽然可以使铲子贴紧地面，但是其上部活动结构会对贴地性产生影响，在与敌方小车正面对抗时，如果对方的贴地性也十分良好，对撞产生的力会在我方铲子上面的活动结构处产生一个扭力，使我方铲子轻微反转，贴地性变差。而且这种结构的铲子与地面的接触面积较大，摩擦力较大，阻碍行动的同时会产生磨损，需要频繁更换。

2.9方案二铲子结构

方案三：使用如图2.10,2.11所示的铲子，该设计由方案二铲子改良而来，安装在机器人正前方。设计时，通过借鉴市面上多数的四驱轮式装载机的铲子结构并进行了力学分析后，将铲子角度进行调整，铲子前端斜向下方，减少贴地面积的同时进一步加强了其贴地性，而且减少了铲子重量，有利于提高登台成功率。同时

为了能够更加容易的将敌方铲子铲起，我们将铲子前方设计了一定弧度，可以轻易地将对方铲子进一步铲起。

2.10 方案三铲子结构整体视图 2.11方案三铲子结构侧视图

综合以上分析，方案三更能满足要求，故选择最终选择方案三作为对抗结构。

### 上台结构

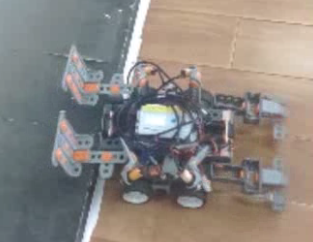
方案一：如图2.12前方和后方各两个铲子，分别用四个舵机控制铲子的运动。前铲和后铲结合上台，当检测到在台下时，首先正对擂台，利用前面两个舵机旋转控制前铲将机器人前部支撑起来，此时驱动前方轮子，然后再让后面两个舵机旋转使后铲将机器人后部支撑起来，这时驱动后方轮子使机器人上台。这种上台方式对动力要求不高，但是速度缓慢，且对手在边缘守台的情况下不适用。

图2.12前后铲子结合上台结构

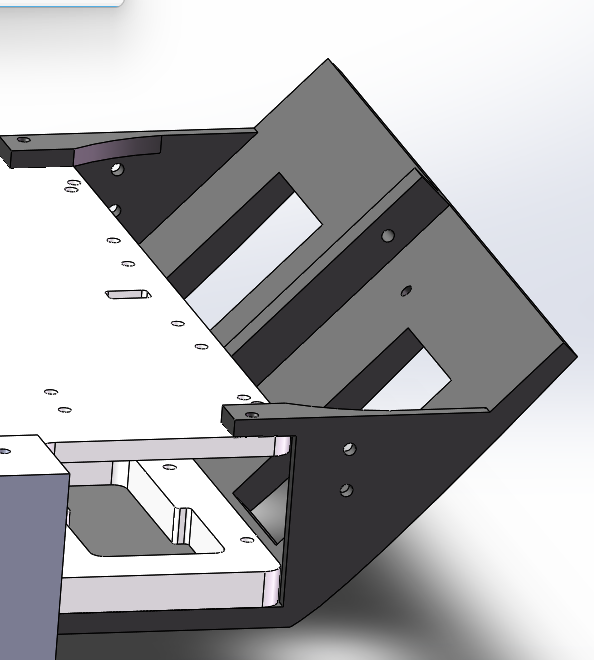
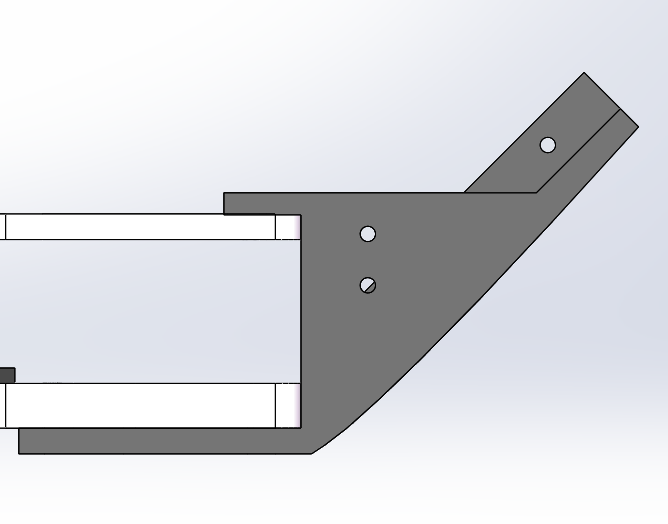
方案二：如图2.13,2.14设计一斜三角铲子，小车先背对擂台，然后加速冲向擂台。完成登台。

图2.13斜面上台结构侧视图图 图2.14斜面上台结构整体视图

斜面上台结构上台分为三个阶段：

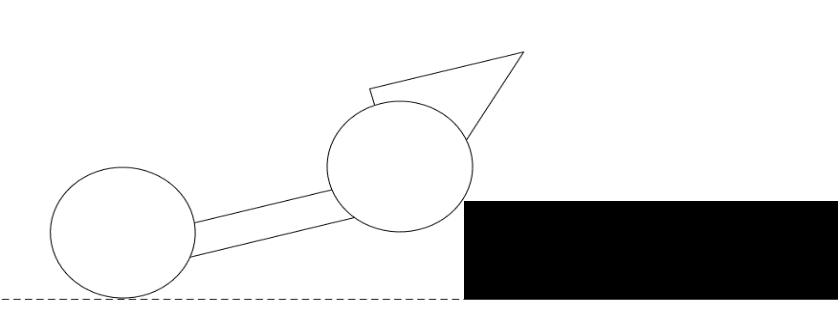
第一阶段，机器人背对擂台，开始加速，当斜面结构接触到擂台边缘时，利用惯性借助斜面结构使机器人上升并使后轮接触擂台，如图2.15；

图2.15上台第一阶段

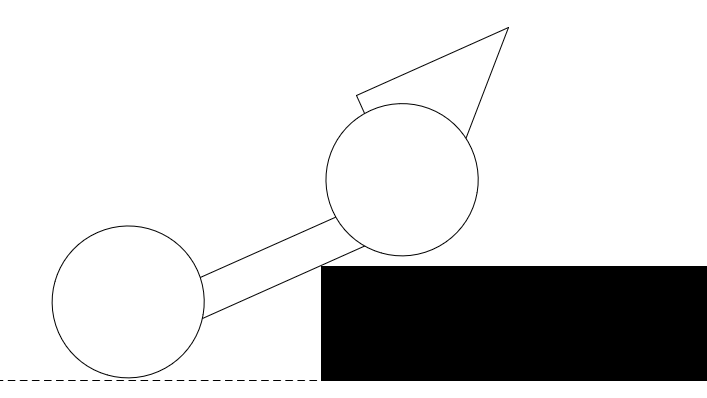
第二阶段，后轮接触擂台，前轮仍在地面，此时机器人获得动力使机器人继续上升直到四轮悬空，如图2.16；

图2.16上台第二阶段

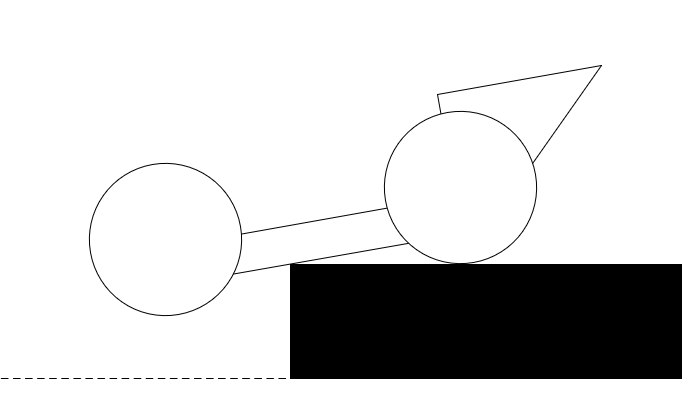
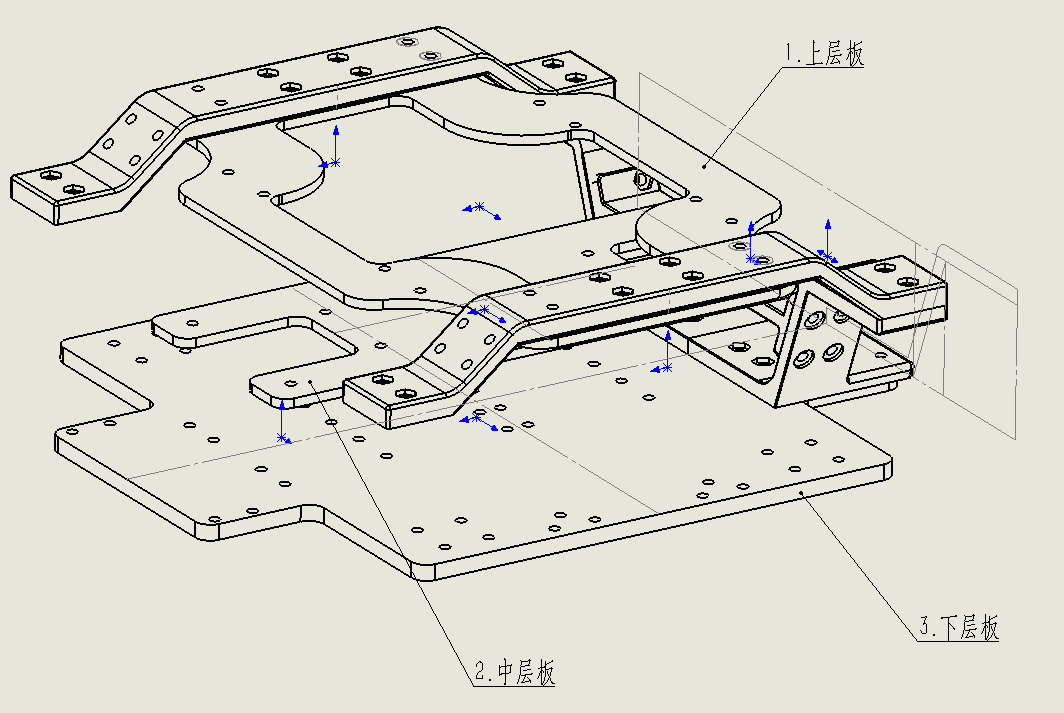
第三阶段，在机器人车轮没有附着时， 利用自身惯性使机器人重心到达擂台之内，进入擂台便可以开始进行漫游和检测目标等动作，如图2.17。

图2.17上台第三阶段

综合以上分析，方案二更能满足要求，故选择方案二作为上台结构。

### 底板设计

方案一：如图2.18所示，底板分为三层，上层板固定前铲和上台铲，中层板固定白壳电池，下层板固定蓝皮电池，电机组，但上层板和中层板之间有很多空间未利用，在增加了摄像头和其他传感器之后高度过高，些许超过规定高度，且重心也偏高导致车身不稳定。结构件过多，连接处不稳固，不易操作。

 2.18 方案一底板设计

方案二：如图2.19所示，底板分为两部分——上底板和下底板。两层底板之间通过电机座连接。下底板如图2.20所示，上底板如图2.21所示，上层底板承载各种传感器及支架，下层底板承载蓝皮电池、电机组，但是走线困难，而且电机组离地距离超过预期，使上台困难。

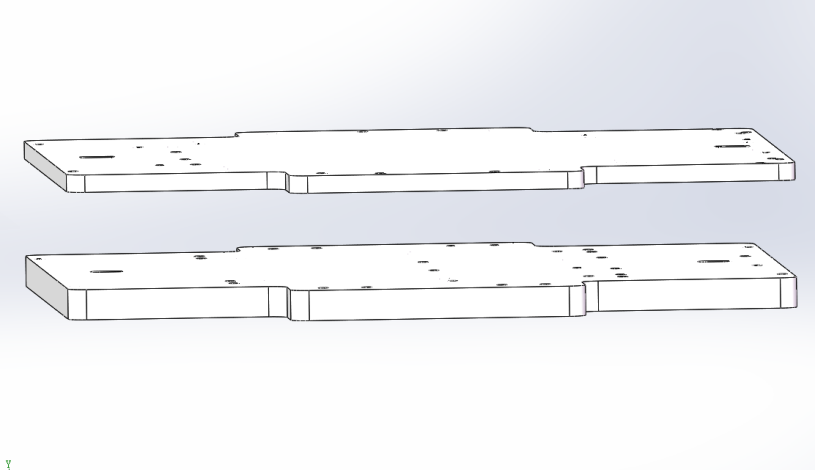
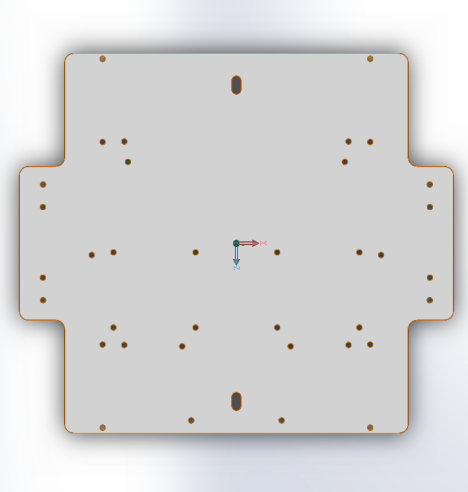
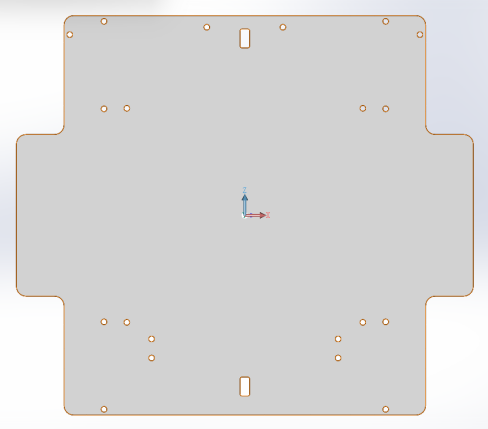
图2.19方案二底板设计

图2.20 方案二上底板 图2.21 方案二下底板

方案三：如图2.22所示，沿用方案二的双层结构，中间通过电机座进行连接，通过将下底板挖出四个电机固定位，使电机更稳固的同时，减少上下底板之间的距离，从而使整体车身高度降低，上底板中间开口，通过加设一个支架的方式来达更好的空间利用率并且通过这种方式，上下两层传感器更方便连接到控制器上，使整体走线整洁美观。

1.下底板：承载电机组、蓝皮电池、下层传感器

2.上底板 ：承载摄像头支架、上层传感器、前方攻击铲、登台板

3.白壳电池支架：支撑白壳电池、控制器

4.电机座：固定电机，连接上下底板

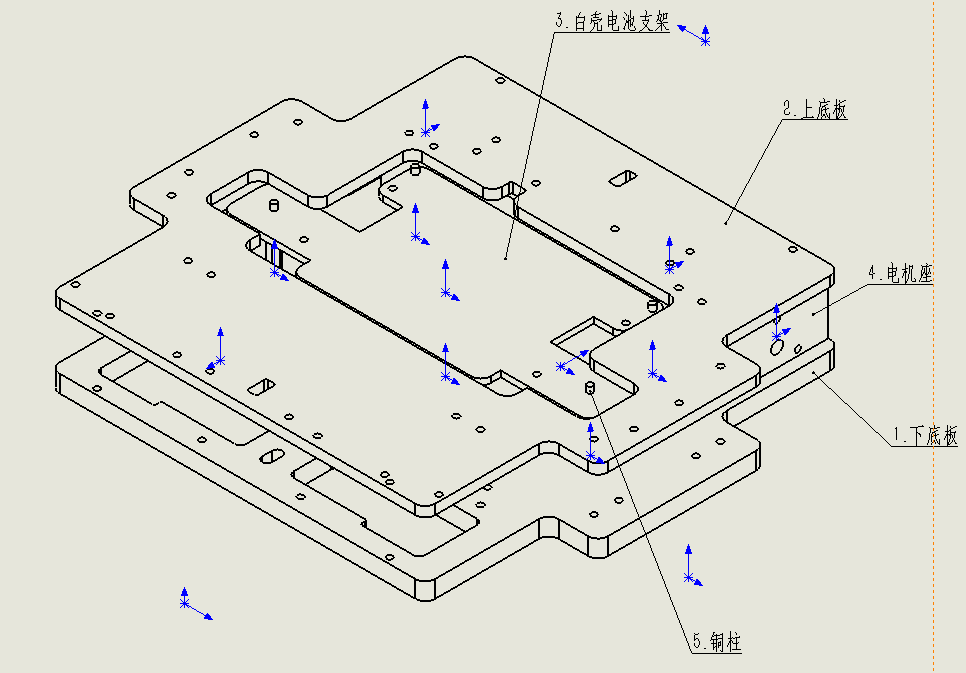
5.铜柱：支撑白壳电池支架

图2.22 方案三底板设计

综合以上分析，方案三更能满足要求，故选择方案三作为底板结构。

# 软件设计

## 总体概述

2021年统一部件轮式自主格斗线上比赛规则，要求参赛选手的小车完成推下能量块、推下普通障碍快、避免推下炸弹快和避免掉下擂台等功能。与线下比赛相比，提高了对小车视觉处理的要求，也更进一步挑战了小车的灵活度。我组在自主制作了优良硬件的基础上，精心设计了小车实现比赛任务的一系列代码，并不断进行改良与优化（小车的总流程图见图3.1）。基于硬件设计，小车代码共分为三个部分：运动部分、视觉处理部分、与传感器交互部分。

在我们处于真实场景进行程序检验的过程中，我们发现，相比单进程的运行方式，多进程的运行方式在灵活性和实时性上具有显著优势。所以我们采取了多进程的程序设计方式。运行主程序后，总共将会产生四个进程，分别为：主管运动的主进程（图3.1红线部分）、图像采集子进程（图3.1绿线部分）、传感器数据获取子进程（图3.1紫线部分）、边缘检测子进程（图3.1黄线部分）。不同进程之间通过共享变量（位于内存）的方式进行同步，共同实现小车的协调运动。

本次比赛相比往届，对小车的视觉处理提出了极大挑战。我们也花费

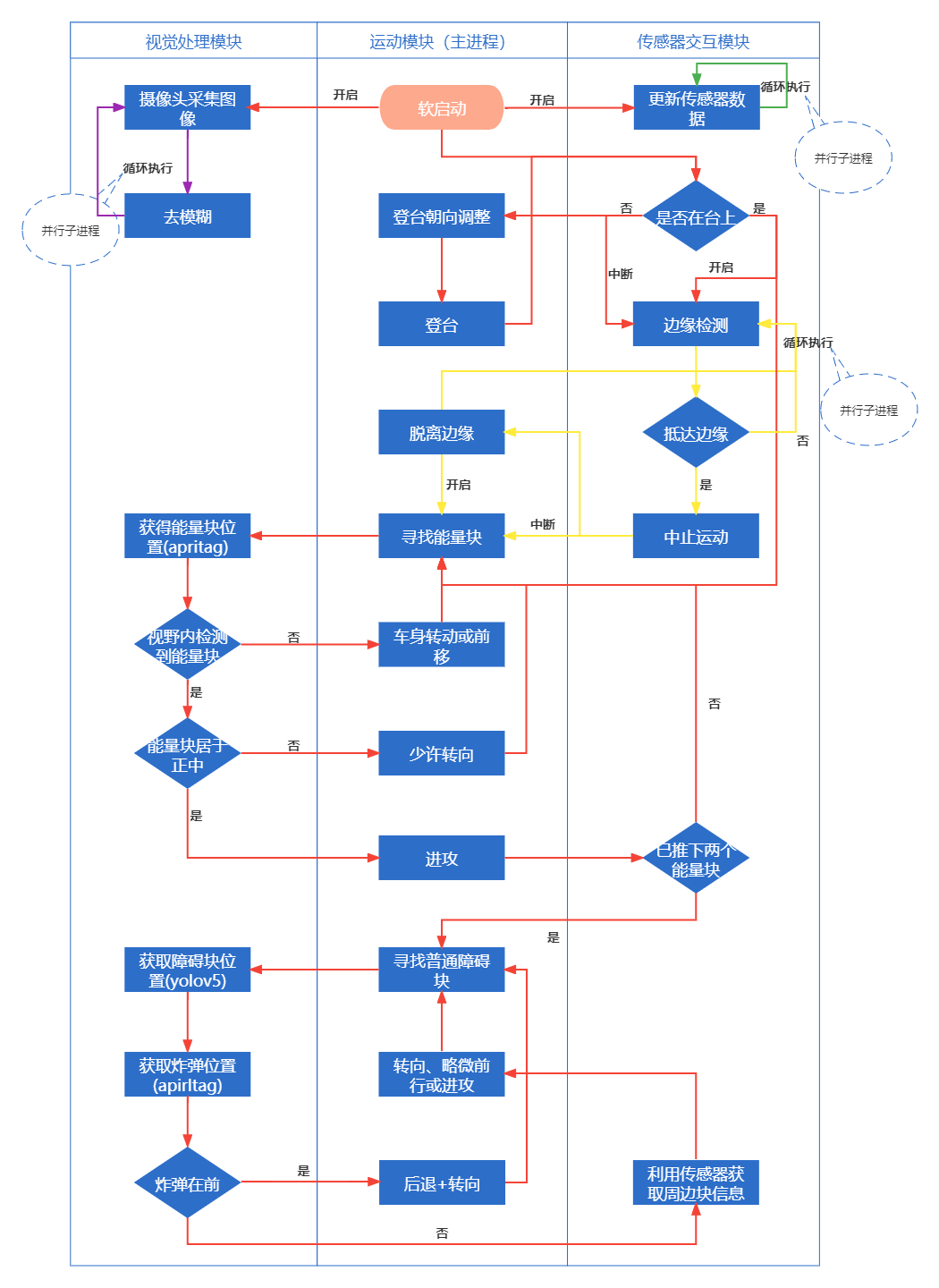
了大量的时间对此进行设计。除了基础的对于Apirltag码的检测外，我们还用到了先进的目标检测模型：YOLO v5[[1]](#footnote-1)，对普通障碍块进行检测。我们成功的将其转化为IR模型，部署于树莓派上，并通过API调用神经棒提高运算力。

图 3.1：算法总流程图。不同的进程通过不同颜色的线条加以区别度。

此外，由于摄像头刷新率较低，导致小车高速转动时，采集到的图像过度模糊，影响了目标检测的准确率。我们采用图像翻译的方法，使用DeblurGANv2[[2]](#footnote-2)模型，对小车采集到的图像进行了去模糊处理，进一步提高了目标检测的准确率。

## 相关技术

### 多进程：Python Multiprocessing[[3]](#footnote-3)

Multiprocessing是python中一个支持使用与threading模块类似的API来产生进程的包。Multiprocessing与Threading相比，虽然使用起来略为繁琐，但是可以实现进程的完全并行，充分利用树莓派处理器的资源，而且可以接受linux的各种信号。我们的代码主要使用Multiprocessing.Process中的Queue和Value类，分别实现视频帧的传输和不同进程之间的同步。

### Apriltag[[4]](#footnote-4)

Apriltag是改进的ARToolkit、 ARTag 。在光照不均的环境下也能具有良好的语义分割效果。

我们的代码使用检测Apirltag的方式检测能量块。

### 目标检测：YOLO v5

YOLO是经典且高效的目标检测算法，通过非滑动窗口、网络结构上一步到位的方式实现目标检测。比起Fast-RNN，SSD，检测准确率不分伯仲，但速度却远远胜出。

YOLO v5 是 YOLO 的最新版本。采用了Mosaic数据增强、自适应锚框计算、自适应图片缩放等技巧，运用了CSP结构，采用了高效的损失函数。比起YOLO v3，YOLO v4这些前辈，YOLO v5网络最小，AP精度也最低，但速度非常快。由于模型非常小，非常适合放入嵌入式设备中。

小车的目标检测只是简单的二分类问题（是否有普通障碍块。能量块和炸弹通过Apirltag码检测）。不需要很高的检测精度，但对速度的要求却极高。如果检测速度过慢，会拖慢小车的反应速度。且高速移动的小车会产生大量的模糊帧。与其运用高精度模型检测这些模糊帧（而我们采用图像翻译的方法），不如在相同时间内检测更多帧，从而降低检测中模糊帧的占比。所以高速的YOLO v5就很有优势了。

### 图像翻译：DeblurGAN-v2

图像翻译是一种主要基于对抗生成网络GAN的技术。与文本翻译类似，图像翻译将原始域的原始图像翻译到目标域的目标图像。自从CycleGAN实现无监督的图像翻译之后，图像翻译的应用范围就被大大拓宽了。去模糊，就是其中一个非常重要的方向。通过对高清图像进行模糊化，就可以得到高清-模糊的成对训练数据。这使训练网络变得可行。

小车在转向寻找能量块的时候，如果转动速度快，那么转动时间也短，但是会产生模糊帧，导致需要一定时间等待画面恢复清晰才能进行检测。如果转动速度慢，虽然不会产生模糊帧，不需要等待时间，但是转动时间却变长了。

为了降低高速转向时的等待时间，我们采用了图像去模糊技术，使用DeblurGAN-v2对模糊帧进行到清晰图片的翻译，从而大大提高了检测精度。

## 代码分析

图3.2：相关代码的截图

运用到的相关代码如图3.2所示：

main.py：程序入口。

kamera.py：设计了一个用于图像采集的camera类，并具备不断刷新缓冲队列，保证帧实时的功能。

my\_controller.py：运动模块、与传感器交互模块的代码。

Yolov5.py：通过openvino运行yolov5的代码。

DeblurGAN-v2：存放DeblurGAN-v2相关代码的文件夹，实现去模糊功能。

接下来将对这些代码分别进行具体分析。

### main.py

流程图如图3.1所示

目标是推下两个能量块、推下普通障碍快、避免推下炸弹快和避免掉下擂台

创建一个MyController类。创建同时打开视频采集进程与传感器数据获取进程→登台→打开边缘检测进程→进入循环：

若当前处于停止状态->寻找能量块。若找到且没有炸弹在正前方，则进攻。

当推下的能量块数量达到2时，退出循环。

进入下一个循环：

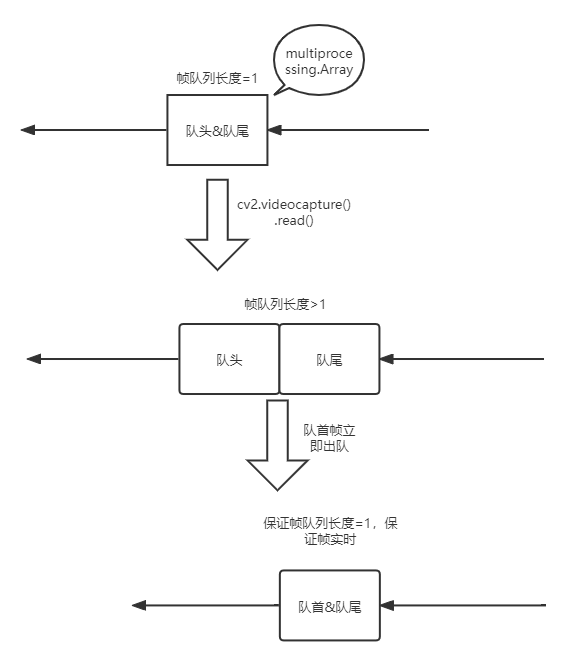
若当前处于停止状态->寻找普通障碍块。若找到且没有炸弹在正前方，则进攻。

### kamera.py

流程如图3.3所示。

Kamera.py提供给一个供其他程序调用的camera类，用来采集图像。一开始，我们采集图像的方式是在每次进行视觉判断之前调用一次cv2.videocapture().read()，但后来发现，这样得到的帧严重滞后。原因是摄像头采集的帧首先会放入一个底层的缓冲队列中。由于摄像头采集帧的速度远远高于程序执行cv2.videocapture().read()的速度，导致缓冲队列越来越长，从队首取得的帧总是滞后帧。

因此，我们设计了一个并行的子进程，不短调用cv2.videocapture().read()，使得底层的缓冲队列不会有累积，再把取得的帧放入我们自设的帧队列中。帧队列属于multiprocessing.array，可用于进程间同步。当帧队列长度大于1时，立刻使队首帧出栈，这样就保证了队列中永远都是实时帧。

图3.3

### my\_controller.py

提供一个MyController类供其他程序使用，导入了小车内置的UpTech包控制运动和获取传感器数据，导入kamera.py创建Camera类获得实时帧。导入yolov5.py通过yolov5鉴别普通障碍块。导入apirltag获取apirltag检测类。

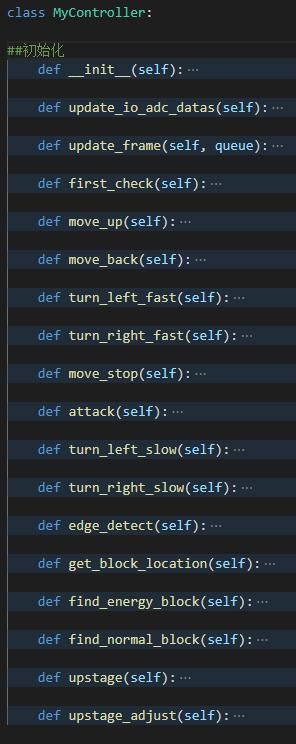
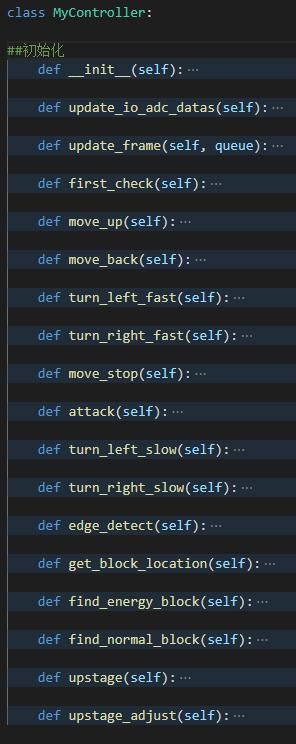
MyController成员函数如图3.4所示：

图 3.4

接下将一一介绍每个成员函数。

#### def update\_io\_adc\_datas：更新传感器数据。

图 3.5：更新传感器数据是一个并行子进程

流程如图3.5所示。最初，我们采用每次执行运动前获取一次传感器数据的方法。这种单进程模式在仅执行移动、转向、攻击等运动时没有问题。但在边缘检

测和调整登台朝向时会产生问题。

因为在小车高速运动时，如果每次执行运动前都要做一次更新操作，就意味着更新操作的时间一起计入了运动时间中。虽然更新数据的时间开销极小，仍会导致小车反应迟缓，而且常常出现慢半拍的情况。

因此，我们为更新传感器数据单独设立了一个子进程，这样不仅能充分利用处理器资源，还能将更新数据的开销独立出来，避免拖慢需要即使反应的边缘检测和调整登台位置模块。

实际上，在将边缘检测独立为一个单独子进程之后，为了实现进程间数值的同步，传感器数值数组必须为共享变量，更新传感器数据也必须为独立为一个并行的子进程。

#### **def update\_frame**

复制帧队列的队尾帧。

#### **def first\_check**

图 3.6

流程如图3.6。软启动函数。

#### **def upstage**

登台函数。如果是遵循线下规则，一局比赛可能进行多次登台。但按照线上规则由于没有外力会将小车推下台，所以只用登一次。

#### **def get\_block\_location**

通过视觉获取能量块、炸弹、普通障碍块位置的函数，会调用apirltag，yolov5。

#### **def move\_xx & def turn\_\_XX\_\_fast & turn\_XX\_slow**

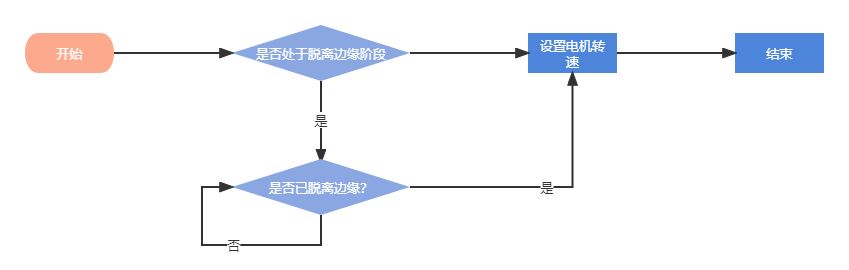
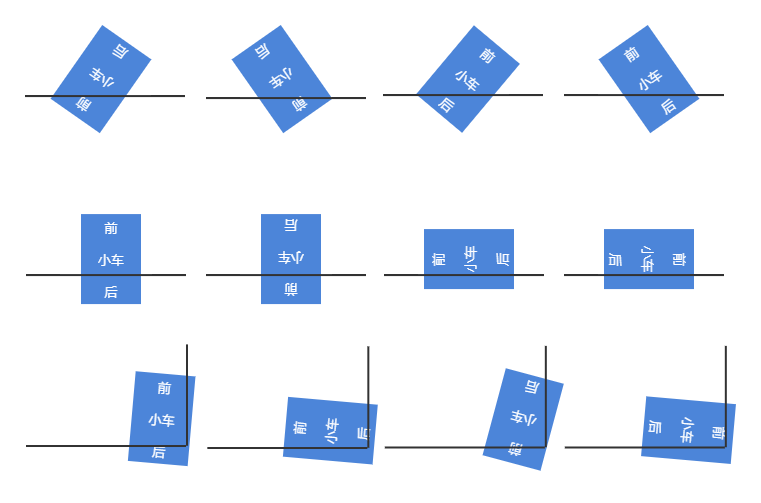
流程如图3.7所示。需要强调的是，如果最开始没有判断是否处于脱离边缘阶段就直接执行运动，可能会使小车掉台。

图3.7

#### **def edge\_detect**

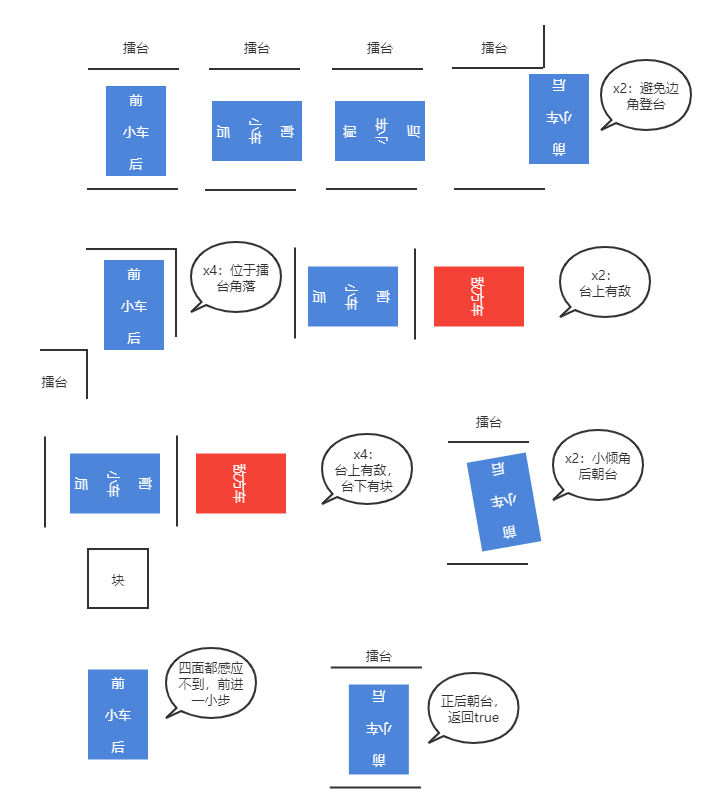
最初，我们采用的是每次循环前执行一次边缘检测的方式，结果和之前提到的一样，由于接触到边缘时，单进程的运行方式并不一定在进行边缘判断，会导致小车出现慢半拍的情况，从而冲下台。所以我们为边缘检测开启了一个并行的子进程，大大降低了小车冲下台的概率。

按照顶部四个斜向下的io传感器开闭情况进行排列组合，总共有16种可能情况。其中位于边缘的12种可能情况如下图所示。有三种情况是不可能出现的：①四个io全灭；②左上和右下io亮，右上和左下io灭；③右上和左下io亮，左上和右下io灭。四个io全亮即代表不位于边缘。我们为每种情况都设计了一种脱离边缘的策略。虽然判断条件众多，但时间开销其实只有2e-5秒~5e-5秒，已经足够灵敏。

图 3.8：边缘检测的12种情况，小车的四个角都有一个检测边缘的io传感器

#### **def upstage\_direction\_adjust**

调整登台角度函数。台下情况如图3.10所示。每种情况均有详细对策。由底层的红外测距传感器(adc)和中层的光电传感器（io）负责。后底部配备双adc传感器，用于避免边角登台和小倾角后朝台的调整。难点在于后底部两个adc传感器检测相同距离返回的值不同，需要进行线性修正。

图3.9：17种调整登台角度的情况。xN代表不同朝向的类似情况。对红外测距传感器的数值有进行线性修正。

#### **def find\_energy\_block**

寻找能量块的函数。流程见图3.1中部，不再重复赘述。

#### **def find\_normal\_block**

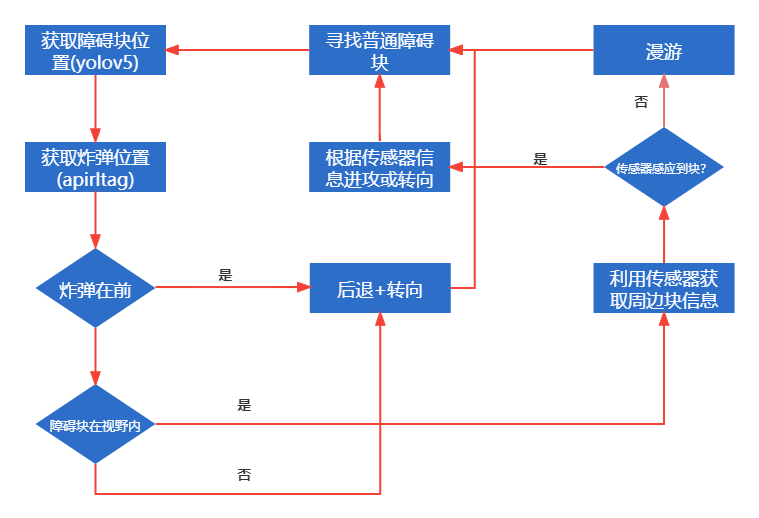
寻找普通障碍块的函数。流程如图3.10所示。

图3.10：寻找普通障碍块的流程。是对图3.1的细化与补充

由于yolov5检测时有少许概率会把画面噪声当作检测目标，因此只用yolov5做分类用。

#### **def attack**

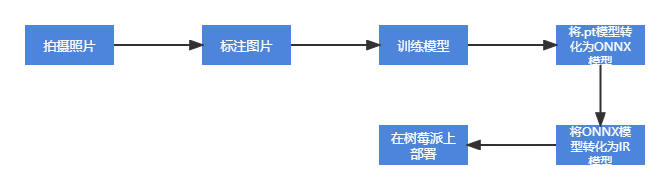
图 3.11

流程如图3.11所示。

### Yolov5.py

流程见图3.12，Yolov5部署到树莓派，需要经过一系列转换。在此之前，我们先拍摄了一系列包含普通障碍块的图片，之后再用labelImg工具进行标注，

建立数据集，再进行模型的训练。训练出相应的.pt模型后，再转化为ONNX模型，接着使用openvino的mo.py进行到IR模型的转换。最后通过调用小车内部的openvino包载入模型，就可以实现对目标的检测了。

图3.12 yolov5 部署

## 效果展示

图3.13 apirltag检测

图3.14 yolotag检测

如图所示。在使用图像翻译的方法之后，小车在apirltag检测与yolotag检测上都变得更为精确了。

# 总结与心得

团队以人工智能控制器为控制核心，采用红外光电传感器、红外测距传感器、驱动器、电机等外设，设计了轮式格斗机器人。该机器人能够实现竞赛的所有要求，且机器人反应速度较快，运动灵活，能够敏锐识别进攻“能量”块并避开“炸弹”块。此外，该机器人还能够实现自主寻敌功能，通过机器学习，机器人已经能够自主辨认敌方小车，将识别与漫游相结合就可以快速寻找到对方的位置发起进攻。而且一旦跌落擂台，可以有效的面对各种可能发生的情况，迅速调整位置重新登台，这样在实际的对抗中便能更快的取得胜利。

本次比赛，对我们团队成员是一个从0到1的过程。我们总结以往的经验，在4到6月份设计并且组装了第一代小车，最终因为结果并不是很理想而放弃。但是队员并没有被眼前的挫折而击败，我们认真总结失败的原因：如何能够最大程度的降低机器人的重心增强其稳定性？如何设计上台结构能够最大限度减小阻力迅速上台？如何布置传感器能够使机器人在登台时能够正对擂台？面对传感器返回数据波动太大有噪声干扰时选取哪种滤波算法……在设计过程中对这些问题仔细推敲，反复斟酌，尽可能使其达到我们设想中的理想状态。在7月又重新设计了第二代小车。

我们以为最简单的代码过程却成了我们最大的难题，机器人的运动状态与我们的设想大相径庭，无法实现上台、无法准确判断自身状态、漫游时一直原地旋转，不能识别能量块……我们针对每个问题，从硬件和软件两方面入手，提出所有可能的解决方案，在接下来的过程中，我们重新调整传感器的位置、仔细观察每个传感器的阈值及灵敏度，将小车重心后移……但是效果仍然一般，陷入焦虑的我们几近奔溃，于是我们暂停了进度，在休息的一天中，我们相互鼓励，积极调整心态。在冷静下来之后，我们有了头绪，最终在更换轮胎之后解决了登台问题。更重要的是我们打破之前的想法，有针对性的从结构到程序做了全新的调整，用一部分的光电传感器取代红外测距传感器，重新开始设计代码……经历了一次又一次的“打破”与“塑造”。

在这不断否定与重塑的过程中，我们面对困难越来越得心应手，也离成功越来越近，同样的我们感到自身也在不断地蜕变，不仅仅是知识的获取，更是心灵上的升华。

这一次的备赛，我们更加深刻的认识到，学术研究本就是一个不断推进的积累过程，我们既需要有不断创新之“破”，也需有追本溯源之“立”。这就是我们这次比赛最大的收获，它将伴随我们之后的学术生活，帮助我们走的更远。而且，通过本次比赛所学的知识为我们接下来进一步研究做了很好的铺垫，我们希望借此次机会，进一步训练我们的格斗机器人，并以机器人充当我们的平台，继续训练我们的识别模型，争取在某一个方面能够取得不错的收获，进而申请专利或者发表论文。

最后，衷心祝愿比赛越办越好，也祝愿我们能够取得不错的成绩

# 参考文献

**[1]** https://docs.ultralytics.com/ （由于无论文，我们只引用了位于github上的代码。）

**[2]** [DeblurGAN-v2: Deblurring (Orders-of-Magnitude) Faster and Better, arXiv:1908.03826](https://arxiv.org/abs/1908.03826) [cs.CV]

**[3]** https://docs.python.org/zh-cn/3/library/multiprocessing.html

**[4]** E. Olson, "AprilTag: A robust and flexible visual fiducial system," 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 3400-3407, doi: 10.1109/ICRA.2011.5979561.

**[5]**胡仁善，闫彩霞，彭卫平等编著. Pro/Eingineer Wildfire 2.0中文版机械设计高级应用实例.北京:机械工业出版社,2005.

**[6]**张云杰主编.Pro/ENGINEER Wildfire 2.0机械设计.北京:中国林业出版社，2006.

1. https://docs.ultralytics.com/ （由于无论文，我们只引用了位于github上的代码。） [↑](#footnote-ref-1)
2. [DeblurGAN-v2: Deblurring (Orders-of-Magnitude) Faster and Better, arXiv:1908.03826](https://arxiv.org/abs/1908.03826) [cs.CV] [↑](#footnote-ref-2)
3. https://docs.python.org/zh-cn/3/library/multiprocessing.html [↑](#footnote-ref-3)
4. E. Olson, "AprilTag: A robust and flexible visual fiducial system," 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 3400-3407, doi: 10.1109/ICRA.2011.5979561. [↑](#footnote-ref-4)