



# 2017 RoboMaster 夏令营 技术报告

2017.8

# 目录

1 机械部分 .....	4
1.1 设计动机 .....	4
1.2 设计需求 .....	4
1.3 设计方案 .....	4
1.3.1 底盘机构 .....	5
1.3.2 云台设计 .....	5
1.3.3 飞机取弹方案设计 .....	10
1.3.4 战车接弹方案设计 .....	11
1.4 方案的优点与不足 .....	13
1.4.3 战车取弹方案优点与不足 .....	13
1.4.4 战车接弹方案优点与不足 .....	13
2 嵌入式部分 .....	15
2.1 整体方案 .....	15
2.2 运动学解算方法 .....	15
2.3 云台与底盘控制方案 .....	16
2.4 串口高效读取 .....	17
2.5 难点与不足 .....	18
3 算法部分 .....	19
3.1 开发环境介绍 .....	19
3.1.1 硬件环境 .....	19
3.1.2 算法环境 .....	19
3.2 整体技术方案概述 .....	20
3.2.1 技术原理介绍 .....	20
3.3 算法整体框架设计 .....	22

3.4 算法功能模块说明 .....	23
3.4.1 定位算法 .....	23
3.4.2 导航算法 .....	23
3.4.3 跟踪射击 .....	23
3.4.4 单兵逻辑 .....	24
3.4.5 多兵作战 .....	26
3.4.6 控制 .....	26
3.4.7 二维码识别 .....	27
3.4.8 弹舱识别 .....	28
3.4.9 装甲板识别 .....	29
3.5 测试结果 .....	29
3.6 可优化方案 .....	29
3.5 测试结果 .....	30
3.6 可优化方案 .....	30
4 夏令营感想、总结 .....	32

# 1 机械部分

## 1.1 设计动机

无人机从神符立柱上抓取弹丸，并通过机械装置转移给步兵，步兵通过挤压子弹，击打对方。

## 1.2 设计需求

比赛的功能需求

- a. 在神符立柱上抓取子弹
- b. 无人机准确投弹至步兵车弹舱
- c. 弹舱里的子弹能够高效准确地被送进弹管

各需求的优先级

c. > a. = b.

大概的设计时间

- a. 两周
- b. 一周
- c. 两周

## 1.3 设计方案

详细的机械方案

稳定性分析

### 1.3.1 底盘机构

#### 如激光雷达、摄像头等传感器如何安放

- a. 关于激光雷达地安放，我们延续了官方的设计思路，在预留的雷达空间里用碳板加工了雷达的基座零件，通过下方的铝槽走线，减少对雷达构图的干扰。
- b. 关于摄像头的安放，我们与电控同学讨论后决定将摄像头安装在 YAW 轴旋转中心的正上方，同时为摄像头安装亚克力制的防护固定装置。

#### 有悬挂和无悬挂的区别，为什么要这样设计？

官方车采用的是前联合悬挂，后无悬挂。我们认为：悬挂的目的一是在于减缓麦克纳姆轮的微小振动对云台的影响，二是在遭遇梯台之时减缓中心的偏移。而这次比赛是在平地进行，故独立悬挂设计对系统的稳定性贡献作用不大，同时，更改底盘悬挂系统需要大量的时间进行稳定性测试，以保持车身的水平等问题，故我们采取沿用官方悬挂模式。

### 1.3.2 云台设计

#### 如何防止卡弹

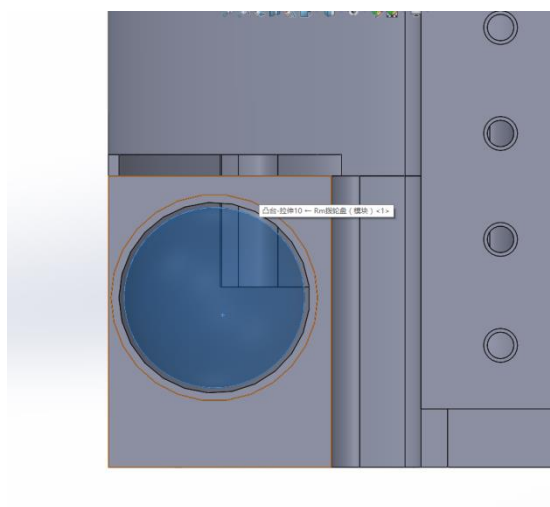
经过分析，我们认为官方提供的拨弹器存在三个问题：

- A. 拨弹器存在出弹速度不均匀现象
- B. 拨轮与铝制挡片存在对心卡弹
- C. 拨弹器存在空弹问题

针对这些问题我们提出了以下的分析和解决方案：

#### A1. 分析：

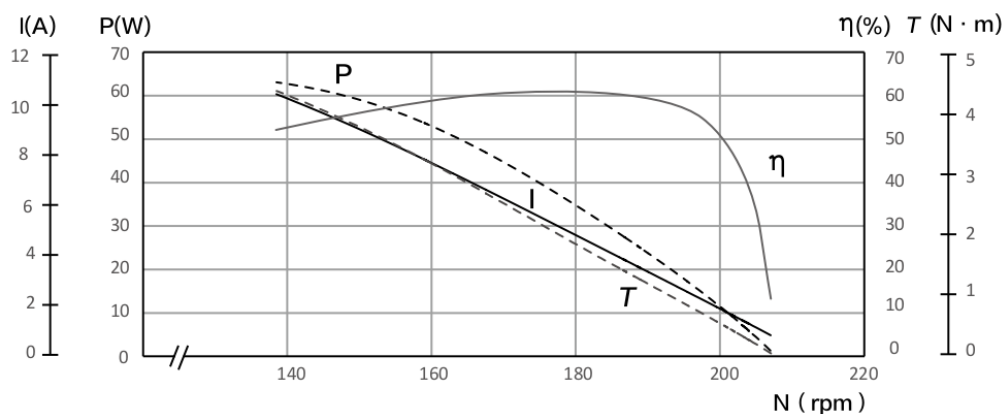
拨弹器出弹速度不均匀是由于子弹磨损较大，表面粗糙容易产生滑动摩擦，且拨弹叉的下部边缘位与拨盘底部距离为 10mm，而子弹半径为 8.5mm 拨盘底部高于底层子弹球心 1.5mm（如图 1.3.2.1）。故与底盘会产生一个与拨弹叉边缘对心，与拨弹器内壁相切的内应力，导致其与拨弹器形成不定时的较大摩擦力扭矩（超过  $4N \cdot M$ ）而限制了拨弹器的速度。



(图 1.3.2.1)

在测试过程中，还出现过与底部塑性形变卡弹。通过查阅 RM2006 电机参数（如图 1.3.2.2）得知：在转速为 180 rpm 时拨弹效率最高，此时测试子弹的预期对应的扭矩约为  $2N \cdot m$ ，故通过减少摩擦力，变滑动摩擦为转动摩擦，可以使力矩降至  $2N \cdot m$  以下以提高拨弹器效率。

## 电机参数 性能曲线



(图 1.3.2.2)

### A2. 解决方法:

我提出的解决方案是垫高 4.71 rad (约  $270^\circ$ ) 底部 2.2mm (如图 1.3.2.4) 使拨弹叉下底面高于底层子弹球心 0.7mm。使之成为对心受力，减少了滑动摩擦的

发生，同时进入拨弹器末端时，子弹高度低于 20mm 不与上端的分层片在入口钳制住子弹。经过长期测试，这种方案比较好地解决了出弹速度不均匀问题。

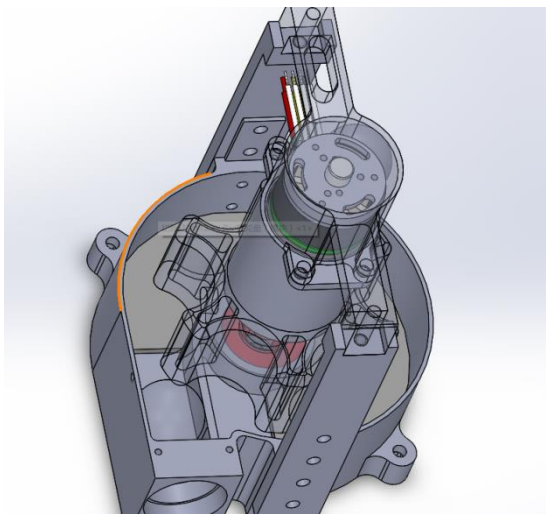


图 1.3.2.3



图 1.3.2.4

### B1. 分析：

子弹与铝制分弹片存在对心卡弹问题，是由于子弹从第二层下落到第一层需要非固定的时间和空间，下落轨迹与拨弹器的转速和子弹下落的初始位置有关。

理论计算为：①  $H=1/2*9.8*T^2$  ②  $S=w*r*t$

故可知，不同转速存在一一映射的落弹初始位置，使子弹球心与铝制分流片在出弹口重合，故我认为**刚性分弹**并不可取。子弹卡弹问题的**核心**是：子弹没有在预定的时间运动到相应的位置上，分弹机构应该是在子弹进入管道道前，把子弹下拉到底层或者抬升到上层。经过测试表明：使用弹性分弹片。

\* 我们在第一次机械技术报告时**最先测试成功并提交了解决方案**，与其他组机械同学交流时发现：其他组解决方案一为遮挡部分落弹区域，防止在对应弹频区间的可能卡弹性，我认为这种方案是牺牲部分的拨弹面积实现的，造成了的空弹问题。二为采用电控方式，通过 Rm2006 电机检测到卡弹时，通过反转拨弹叉，解决卡弹问题。

### B2. 解决方案

针对卡弹问题，我进行了长期设计，测试，论证可以归纳为 2 个版本。第一版设计是在拨弹器的侧壁上加工 2 个相切与 54mm 拨弹叉外边缘，直径 M2 的小

孔，贯穿一根选型 0.8mm 的多股钢丝（如图 1.3.2.5），钢丝末端通过 0.8\*7\*4 的拉簧链接，并采用内径 2mm 的限位杯士固定（如图 1.3.2.6）。经过长期测试论证：钢丝能够有效把在下落过程中的子弹快速归位，同时通过链接在外部的拉簧削弱这一过程产生的振动，并且弹簧位于两层子弹空隙中，不会同子弹产生长时间接触。经过 3 次 3500 发共计超过 10000 发集中实际打弹测试，验证了此方案的可靠性。

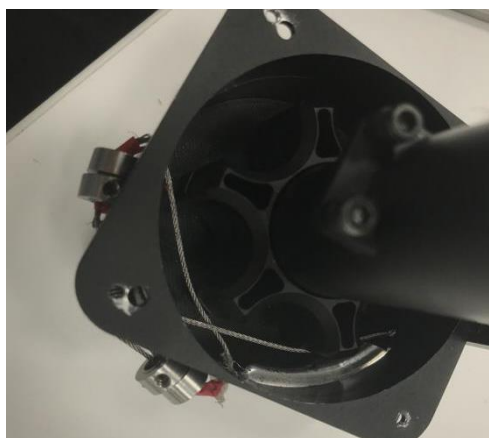


图 1.3.2.5



图 1.3.2.6

考虑到弹性悬挂钢丝的方案的使用寿命，第二版设计，沿用了之前分析的思路，采用弹性拨片分弹，此方案用宽度为 4mm 与拨弹器同心的 1mm 碳纤维板（如图 1.3.2.7）替代了原有的铝制分弹片，在子弹进入弹道前，把子弹下拉到底层

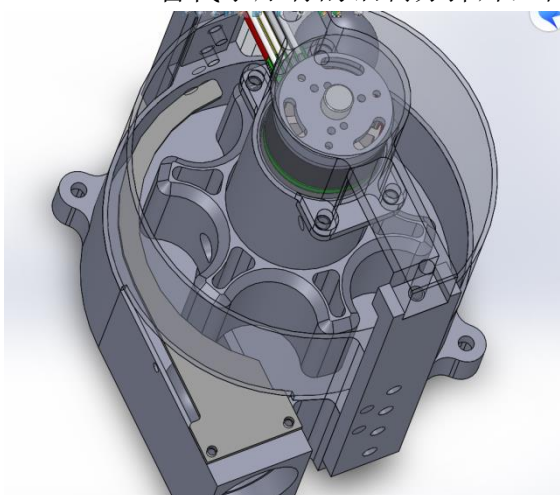


图 1.3.2.7a

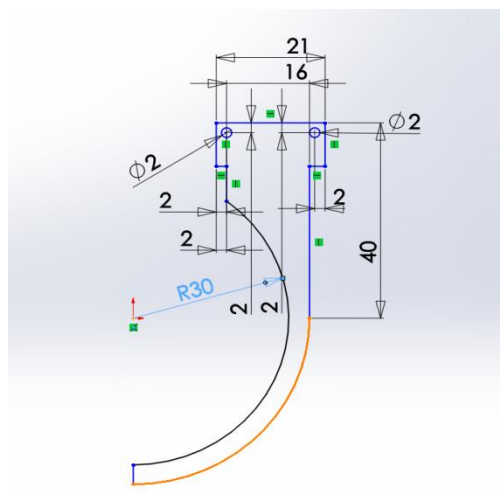


图 1.2.3.7b

或者抬升到上层。经过 2 次 4500 发打弹测试，初步验证了此方案的稳定性。



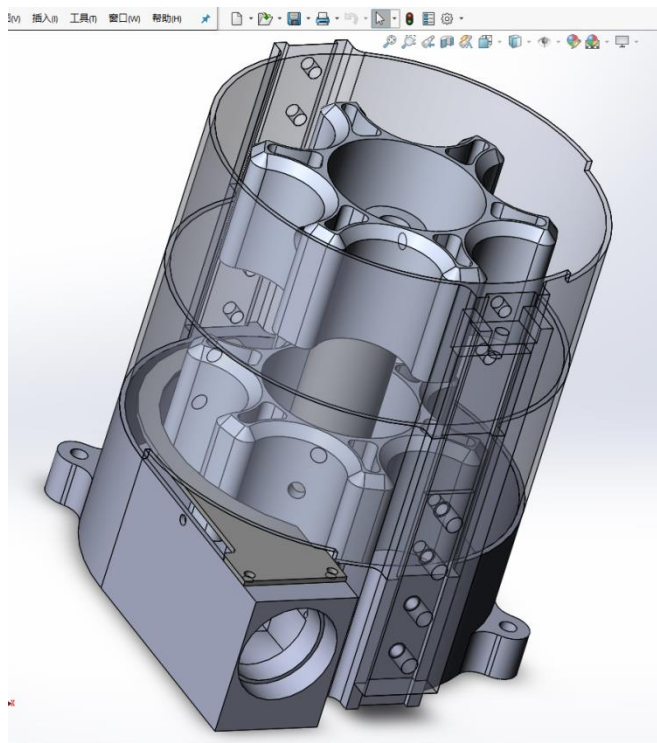
### C1. 分析:

拨弹器空弹问题，是由于拨弹叉转速在 120 rpm 以上时，子弹无法依次进入拨弹器。我认为：**拨弹器的作用是把子弹从弹舱里的无序状态，整理为拨弹器的有序状态，核心是为子弹排序。**官方设计的二层弹舱就是为了高效排序，但随着拨弹器转速远高于 120 rpm 时，子弹无法被拨弹叉捕获。

在计算机算法里，有一种排序方法叫做“**冒泡排序**”，其原理是通过遍历数组比较元素大小，swap 最大元素至末端，同时逐级递减排序数量。受到此排序算法启发，我认为可以在子弹落入拨弹叉之前通过机构给予其一定的 YAW 轴方向的初速度，使其在落入

### C2. 解决思路:

根据这种原理我设计了一种“**双层差速**”的拨弹机构（由于加工条件限制，并没有实现，以下是示意图）

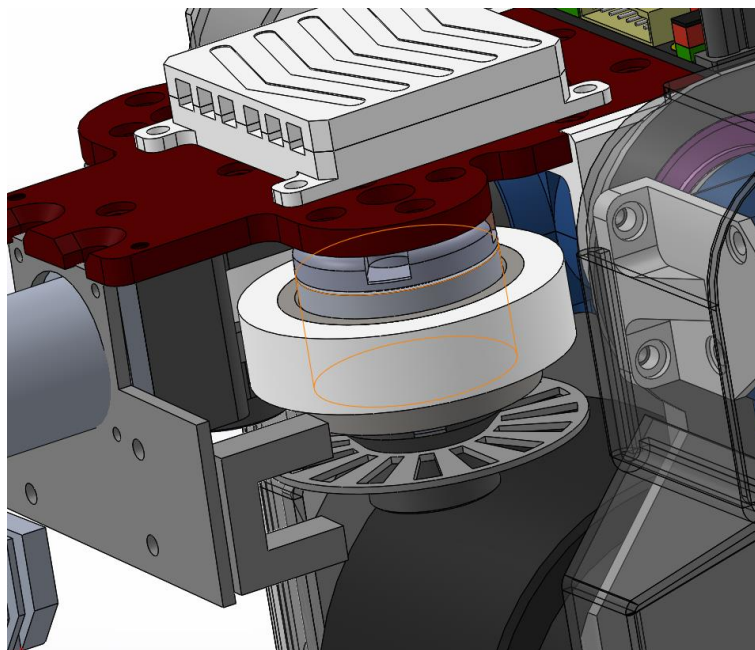


底层拨弹叉给定参量 13000，上层拨弹叉给定参量 6500，两层拨弹叉间相距 18mm。此方案，可以使上层落弹更为高效。同时为子弹提供了 1/2 的惯量，与下层拨弹叉的转速相对差从 13000 降到了 6500，更容易使子弹从中层掉入下层，并不会

造成上层拨弹叉与下层拨弹叉存在卡弹问题，此方案提高了拨弹效率。

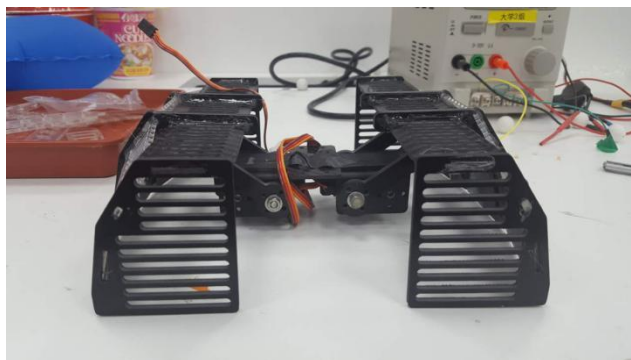
### 如何提高子弹一致性

经过测试，我们认为官方提供的闭环控制在高速高频射击时，弹道在可接受误差内表现良好。最后也取消了我们设计的光电栅的闭环控制。

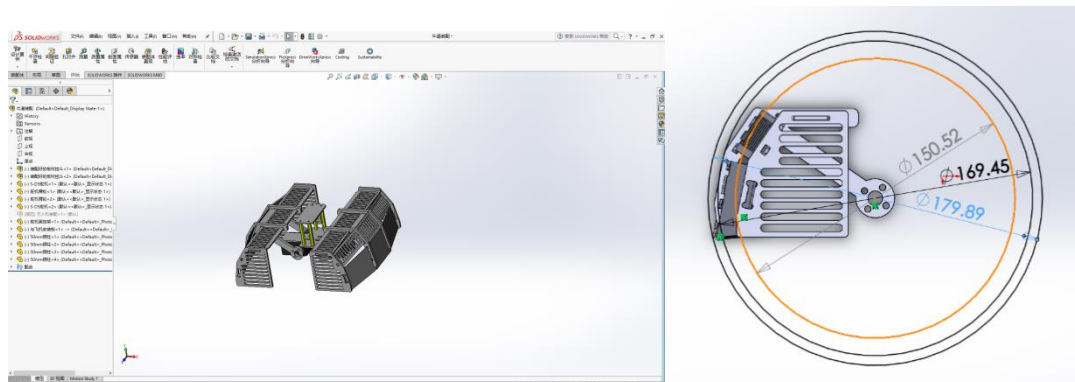


### 1.3.3 飞机取弹方案设计

我们组总共提出了三种设计包括：摩擦轮，涵道，机械爪。并都做出了实物进行测试，在空间，时间，耗电量，电控难度的综合比较后，最终我们选择了机械爪的方案。（图片）



取弹机构的驱动装置是两个对称布置的舵机，机械爪由两只挖斗组成。将飞机放置在停机坪上，挖斗张开时正贴合弹丸上表面。合拢时挖斗底部和停机坪上表面留有间隙，确保不会将飞机重心抬高。



挖斗的形状是一个四分之一圆三条弦线的连线，通过拟合一个圆弧，减少挖掘过程中阻力。

最终效果为一次取弹用时一秒以内，在弹丸铺满时可以取到约 300 颗。第二次约 200 颗，第三次约 150 颗。第一次取弹后的取弹量和飞机降落位置处的子弹分布有较大关系。

### 1.3.4 战车接弹方案设计

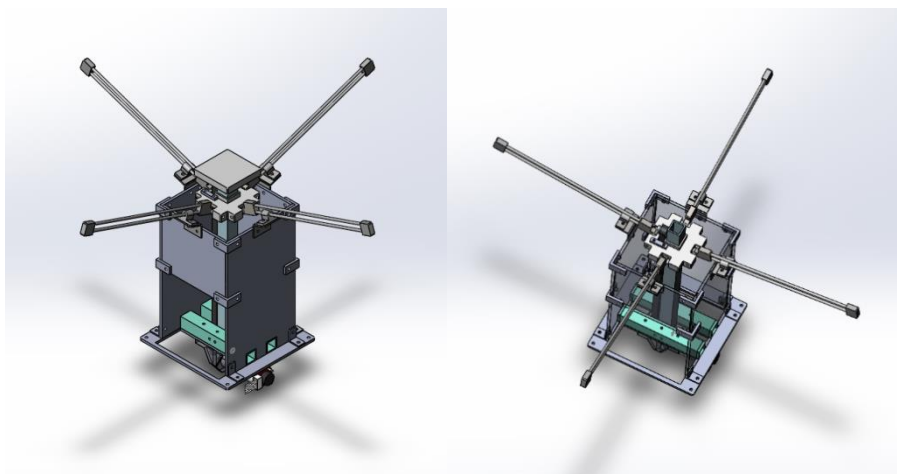
接弹方案一，伸展机构

整个机构主要包括：导向铝管，升降块，四套滑块杆件，驱动同步带及其电机和框架。

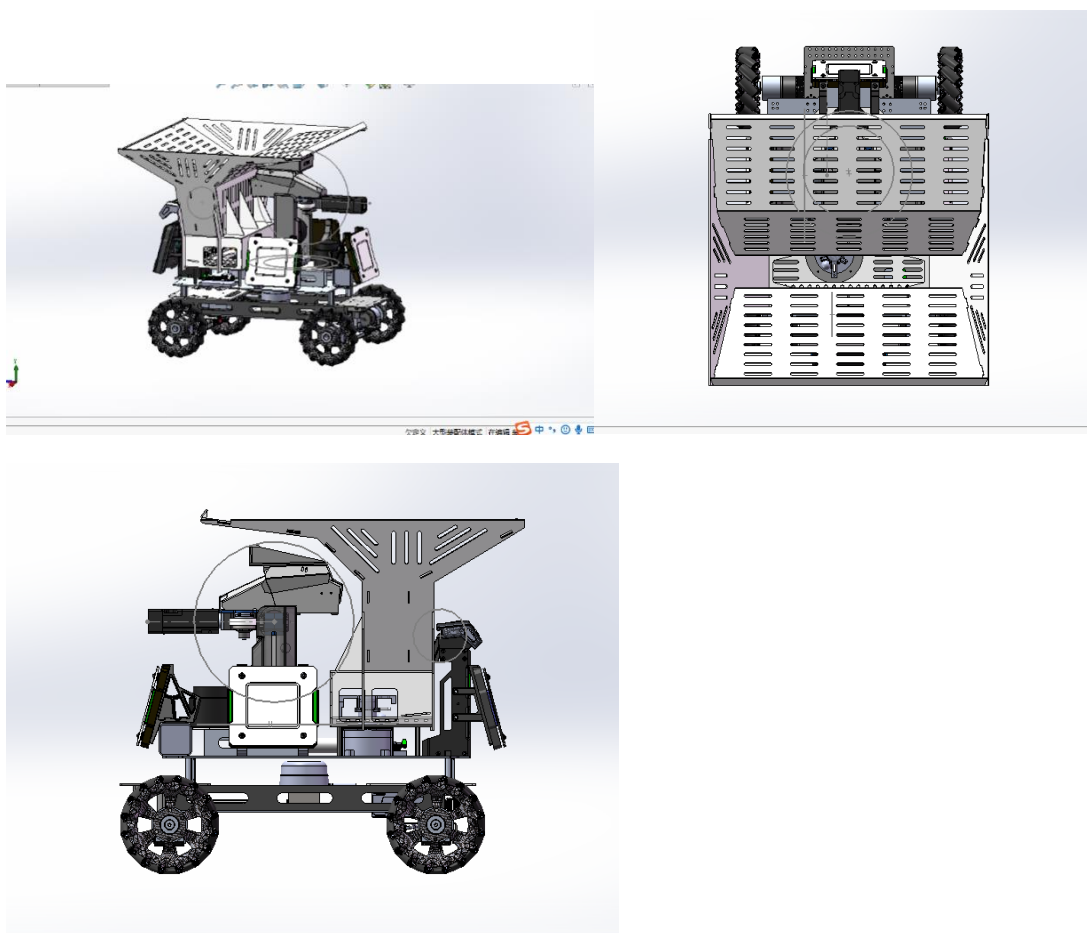
全机构只有一个自由度，由同步带轮加电机驱动。伸展前尺寸为 130mmX130mmX220mm。

展开后的对角线长度为 420mm，接弹面积为 300mmX300mm。





接弹方案二，定尺寸接弹装置



在高度极限内，机构尺寸为 500mmX500mm

## 1.4 方案的优点与不足

### 1.4.1 底盘机构的优点与不足

优点：对底盘改动较小，保证了其稳定性。

缺点：3v3 时，雷达在相近高度，可能存在轻度干涉。

### 1.4.2 云台设计的优点与不足

#### A. 改进型拨弹器

优点：较好解决了卡弹问题，经长期测试验证了其可靠性。

能够保证子弹高效落入拨弹舱，基本解决了空弹问题。

缺点：在其产品化上，其寿命区间有待测试。

在其体量有改进空间。

#### B. 摩擦轮转速闭环控制

优点：能够精确控制摩擦轮转动速度，一定程度上解决了子弹掉速问题、

缺点：电路控制过于复杂，没有容错机制。

### 1.4.3 战车取弹方案优点与不足

优点：1.取弹效率高，取弹量大，最大取弹效率为 300 颗每次，取弹几乎不花费时间。

2.质量轻，整套机构质量为 380 克。

3.模块化，便于移植。近需将飞机架高 30mm，不需要多余的改装。

4.落弹面积较小，轨迹近乎竖直，便于战车接弹。

缺点：1.第二次及以后取弹数量受飞机停留位置处弹量分布影响，取弹数量随次数递减。

2.机械爪开口处若卡弹在飞行过程中会有几颗子弹掉落。

### 1.4.4 战车接弹方案优点与不足

伸展机构

优点：1.机构变形前体积较小，展开后，面积体积比很大。

2.展开过程中，接弹面扫过的体积很小，不易发生干涉。

3.驱动方式简单，用一个电机加同步带即可完成。

缺点：1.导向铝管和升降块之间没有做润滑，有一定的摩擦阻力。

- 2.展开面积受机构初始高度限制，仍然需要进一步优化。
- 3.机构初始顶端面积较小，较难布置二维码等视觉识别标记。

#### 定尺寸接弹装置

优点：1.接弹面积大。

2.制作维修方便。

缺点：1.体积较大，质量较大。

2.在车运动的过程中可能会和墙壁障碍物发生碰撞。



## 2 嵌入式部分

### 2.1 整体方案

在官方代码的基础上根据自己需求修改。

#### 1、整体视图

修整了官方代码，去掉不必要的逻辑，状态标志统一进行切换，刻意的保持了云台和底盘的逻辑框架一致，减小 bug 出现几率。注释统一用中文表达，避免中式英语。

#### 2、云台与底盘

云台与底盘分开控制，底层控制部分保留了官方 PID 运算函数，为方便调试设立了不同的模式。保留了调试用的各个整型变量，以便使用 J-Scope 进行观察。

#### 3、通讯方案

对于步兵车，整机状态机主要跑在上位机，上位机通过串口持续将信息发送至嵌入式控制板，以保证状态的一致性。我们认为官方的串口通信足够高效，如果需要优化则需要实现分层的协议栈，裁判系统也需修改。

### 2.2 运动学解算方法

#### 正逆运动学解算方法

首先是与上层进行沟通，明确了一个车体的坐标系，并根据坐标系矫正了车的  $V_x$ 、 $V_y$ 、 $V_w$  的方向与在速度矩阵里的符号，为麦轮的速度解算做好准备。

我们可以得到由编码器测出的速度，所以测量了车轮转速与其编码器转速之间的比率关系（1 比 20），并明确了轮子的线速度与轮子半径之间的关系；并且找明了各运算元的量纲单位，最后结合四个麦轮运动的合成，测量出车体中心坐标系距离麦轮的  $a+b$  的值，得出在车体 X 轴平动、Y 轴平动、yaw 轴自转的值。

具体步骤如下：

①计算轮子轴心的转速（由编码器和轮子转速比例确定）。

②计算与地面接触的小轱辘的速度（根据轴心速度、轮子半径分解为垂直轱辘方向和平行轱辘方向速度），最后解算得到的合  $V_x$ 、 $V_y$ 、 $V_w$  只与平行于轱辘方向

的速度有关。

③根据 4 个麦轮的速度合成我们所需的合  $V_x$ ,  $V_y$ ,  $V_w$ , 实际上便是在麦轮的速度矩阵上确定正确的系数, 以便达到准确的控制。

而在实际的计算上我们先根据 CSDN 上的资料求得麦轮的逆运动方程, 反解出正向运动的系数发送给上层。

## 2.3 云台与底盘控制方案

### 1、云台控制简述

云台分为松弛、自动射击、遥控器控制三个模式; 其中自动射击模式中, 如果图像没有检测到目标, 则以底盘角度为基准左右晃动进行搜索, 直到搜索到目标, 则以云台自身陀螺仪角度为基准追着目标打击; 遥控器控制下, 以陀螺仪角度为基准进行运动。

云台 yaw 轴位置环和速度环都为 pd 控制, 而且 p、d 值都比较大以确保快速到达目标, 基本上不会出现超调震荡现象。

云台 pitch 轴限位在  $-15^\circ$  到  $2^\circ$  之间, 免得云台意外抬得太高而使视觉丢失目标。Pitch 轴调的比较软, 因为作为反馈的编码器机械角的抖动, 调硬了会造成轻微的震动。

为了使云台获得更优秀的瞄准的效果, 我们采用了分段 pid, 即在目标附近和远离目标两种情况下采用不同的 pid 参数。

为了扩大视野, 我们给云台加入索敌模式, 即当视野内无目标时使云台左右旋转, 为了使旋转时的视野尽可能的稳定, 我们采用了匀速递增期望的方式。

对于弹道计算, 我们试验发现, 在五六米的有效射击范围内, pitch 轴只要加上一个常数就能有非常好的效果, 并不需要高阶函数曲线进行拟合, 于是我们去掉了曲线拟合而只用常数进行高度补偿。

在识别到目标的时候, 为了使云台在底盘扭屁股时候有更加稳定的指向, 我们利用底盘陀螺仪的角速度值作为前馈, 叠加到云台速度环的输入上; 另外底盘在扭屁股时候旋转中心并不是云台的 yaw 轴旋转中心, 这导致枪管左右平移, 又因为误差小时间短而调整不过来, 以至于射击不精准, 因此我们调节扭屁股的参数使得二者旋转中心基本一致。这两个措施使得云台即使在扭屁股时候指向也非常稳定, 对于四五米外的静止目标, 在没有误识别的情况下基本上达到百分之八



九十的命中率。

## 2、底盘控制简述

底盘分为跟随、平移、自动导航三个模式；其中跟随模式，底盘的角度以云台与之夹角为基准进行调节，前后左右以遥控器为基准运动；平移模式，只能以遥控器为基准平移而不能旋转；自动导航模式，若没有识别到目标，则以上位机发送导航数据为基准，识别到目标，则以云台角度为基准，原地扭屁股，为了减小扭屁股对云台的带动作用，我们对  $vw$  值的系数作了处理，使之在相对于云台  $\pm 20$  度内保持不变， $\pm 20$  度开外系数逐渐减小的方法。

云台枪管是主角，底盘一般跟随云台。为降低底盘的旋转对云台的运动的影响，我们将旋转的  $pid$  调的很软，而平移的  $pid$  保持比较硬。

在赛场上，云台和底盘都在自动模式，此时唯一的状态切换便是有没有识别到目标。云台和底盘都是独立的任务，二者的逻辑结构保持一致，并且将状态标志统一由一个函数进行改变，确保不会出现切换错误。

## 2.4 串口高效读取

保留了官方代码，如果需要优化，则不妨定立完整的协议栈，分析如下：

串口波特率高达 115200，如果用单字节中断来接受数据的话，每一路串口中断源的频率都有十几 k，几路串口以及 CAN 通信足以把 CPU 占尽，因此必须使用 DMA。而在 stm32 上用 DMA 实现不定长数据接受的方式只有用串口空闲中断，在这里重置 DMA 并处理数据。这一点官方代码写的很好。

我们试过用一个很大的缓冲区接受数据，然后用一个几百赫兹到几千赫兹频率的任务来循环处理数据，以帧头为基准识别数据块，效果也很好，也能实现很好的实时性，但有时候会收到莫名的 0 字符，并且并不比官方代码巧妙。

这一套串口通讯方案的特点在于，只实现了数据链路层的通信，毕竟一对一的串口是一个稳定的链路，而且无论是遥控器、裁判系统还是上位机，每个节点都是持续不断的通信，这些数据相当于从对方“映射”到主控上的，这次错了丢掉等下次，根本不需要刻意的防护。

缺点或许在于，方案不够形式化。如果用形式化的“接口”来写，则需要 `getter` 的概念：在 `get` 方法下，发送 `get` 请求到从机（从而“新建一条链路”），从机（在这条链路上）返回带有校验的数据，若检验失败，主机可以持续请求（抓住链路），

也可以直接丢掉（释放链路），由更上层的部分进行检测重传，由此引入诸如滑动窗口协议之类的东西。但是扪心而论，区区几十厘米的串口线通信还比较可靠，不必多此一举了。

## 2.5 难点与不足

**设计的过程中遇到的问题：**

云台超调，传感器数据没有正确初始化、正反馈，这些会造成严重的超调。

整体逻辑在整改之前晦涩不清，难找 bug。就像 FPGA 中状态机应该分段写一样，状态标志以及相应状态下的动作应该分开写，特别是需要协同工作的状态机，要刻意保持逻辑的统一。

运动学解算的单位跟上层不对应，导致自动导航经常失败，需要仔细计算参数并且实地调节。

**不足之处：系统各部分耦合度太高，模块化不足。应该有以下措施：**

将所有 PID 计算都集中到一个任务进行，这个模块向外提供的接口是：更改各个控制的期望值，上层的模块是诸如底盘云台的任务，只负责设定期望值。这样实现层次分明的逻辑结构。

云台和底盘作为同一层次并且高度耦合的模块，应该放到一个任务中进行，改良状态不一致的情况，节省调 bug 的时间。

## 3 算法部分

### 3.1 开发环境介绍

#### 3.1.1 硬件环境

本组步兵车使用 Nvidia TX1 作为计算设备，使用罗技 C270 摄像头识别敌方装甲板，搭载了步兵车地盘、激光雷达、单轴陀螺仪、USB hub、路由器、接弹机构这些硬件。

本组 M100 飞行器使用 Manifold 妙算作为计算设备，使用 Guidance 作为视觉传感器，使用罗技 C70 摄像头识别接弹机构中的 6cm\*6cm 绿色正方形方块。机身下部是机械组设计的机械爪取弹机构，使用 STM32F103 单片机控制机械爪的舵机，妙算使用 Uart1 与 STM32 通信。

M100 飞行器在飞行中有两套坐标系：一是以停机坪起飞点为原点的坐标系，二是以步兵车取弹机构中的识别方块为原点的坐标系。在指令给出步兵车在第一套坐标系下的位置后，M100 先在第一套坐标系下出发到达步兵车处，随后切换到第二套坐标系，对准步兵车并下降投弹。

#### 3.1.2 算法环境

本组的 M100 飞行器使用了妙算，基于 Ubuntu 14.04，ROS indigo 开发，安装了 Onboard SDK，Guidance SDK，ROS serial 串口通信库，搭环境较容易。

本组在第一阶段使用了 Nvidia TX2 作为步兵车的主控，搭环境过程中遇到了一下坑：

(1) Nvidia TX2 缺少 CP210x 驱动，识别不了 USB 转串口设备，需要重新编译 Linux 内核烧写该驱动；

(2) TX2 上 Open CV3 与部分 Open CV2 函数出现了版本兼容问题。

(3) 经在场地上实践检验，TX2 的 wifi 通信质量比 TX1 差很多，最终在第二阶段，本组切换 TX1 作为步兵车主控。

## 3.2 整体技术方案概述

### 3.2.1 技术原理介绍

M100 飞行器有在无人控制环境下给步兵车补弹的任务。在 M100 飞行器控制算法部分，主要使用位置-速度双环 PID 算法准确控制飞机给步兵车补弹。

如上文所述，M100 使用了两套坐标系。第一套坐标系以停机坪上起飞位置为坐标原点，指令输入步兵车的 X、Y、Z 三轴位置作为外环位置 PID 的位置期望输入，从飞控中读取 50HZ 的速度数据积分得到位移作为位置 PID 的观测值，位置 PID 的输出作为内环速度 PID 的输入，从飞控中读取速度信息作为速度 PID 的观测值，速度 PID 的输出传给 OnBoard SDK 的 `drone->attitude_control()` 接口控制 M100 的速度。此外，yaw 角的期望 PID 一直是 0。通过第一套坐标系以及对应的 PID 控制，能使 M100 到达步兵车附近。

当 M100 到达步兵车附近后，要切换以取弹机构识别标志为中心的第二套 PID，同样是外环位置内环速度。位置 PID 的 X 轴 Y 轴期望是 0，视觉 PnP 算法给出了飞机里标志的 X，Y 轴距离，以此作为位置 PID 的观测值。其余部分的原理与第一套坐标系下的 PID 相同。

通过两套坐标系+内外环 PID，以及视觉算法的闭环控制，M100 能准确对准取弹机构。

需要说明的是，视觉 PnP 提供的识别标志的 XY 轴位置坐标，还经过了乘以姿态矩阵转置矩阵的运算，切换到大地球坐标系，从而避免了飞机摆动的干扰。无人机导航如图 3.2.1.1 所示，PID 控制如图 3.2.1.2 所示：

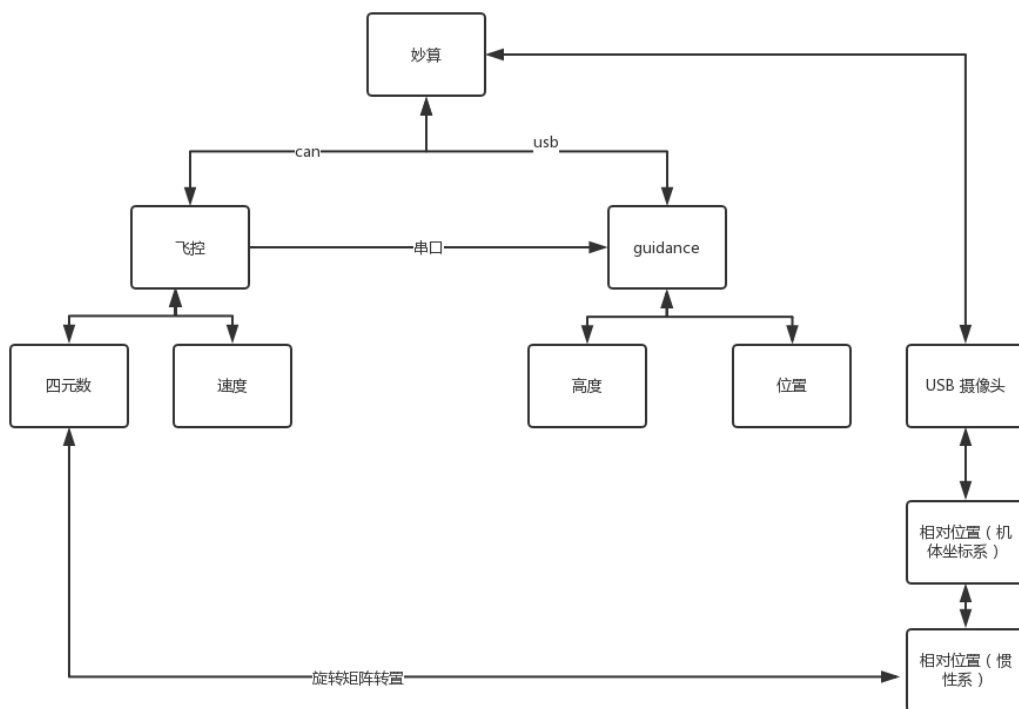


图 3.2.1.1 无人机导航

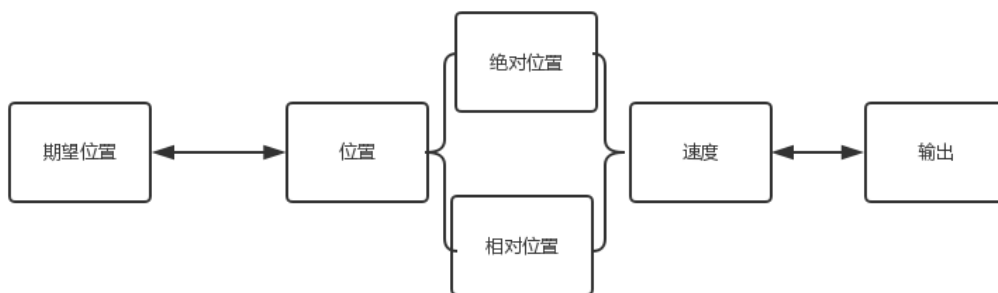


图 3.2.1.2 无人机 PID 控制

### 3.3 算法整体框架设计

本组步兵车的定位算法是结合里程计+amcl自适应蒙特卡洛定位，调用move\_base包+TEB 算法路径规划；跟踪射击方案是通过 PnP 解算，获取装甲板与摄像头之间的旋转矩阵与平移矩阵，进而得到位姿信息反馈控制。步兵车的传感器及数据来源如图 3.3.1 所示：

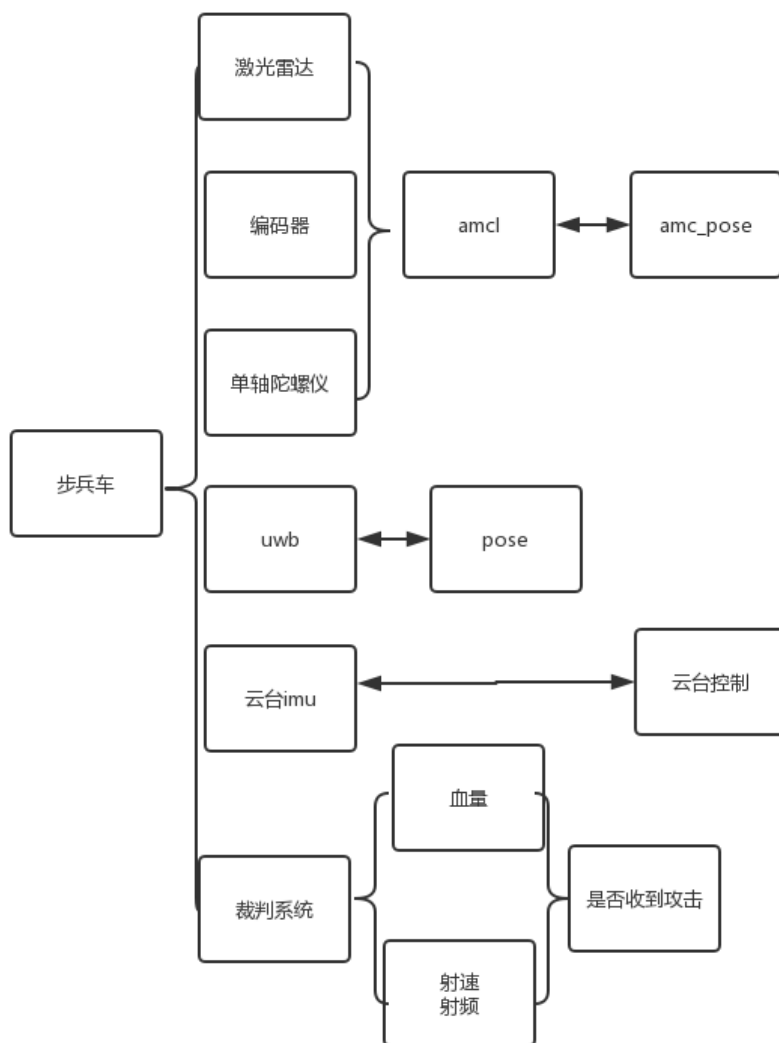


图 3.3.1 传感器及数据来源

本组的步兵车的主逻辑设计使用了状态机切换的思想。步兵车具有开局加弹、巡逻、射击、三个模式，每种模式对应有敌人/无敌人两种情况。M100 飞行器控制整体框架的设计也使用了状态机的思想，具有起飞、飞至定点、识别取弹机构、

投弹四个状态。

## 3.4 算法功能模块说明

### 3.4.1 定位算法

本组在底盘里程计的数据尽可能准确的前提下，尝试过激光雷达+撒粒子定位、UWB 定位，以及激光雷达与 UWB 的融合三种方案。

最终在第一阶段采用了激光雷达+撒粒子的方案。将里程计的数据作为预测值，通过雷达数据来匹配地图得到观测定位值，利用粒子滤波将观测值对预测值修正，最终得到位置的概率分布。

在定位时，由于场地是对称的，很容易定位到场地另一边，只能手动校正；此外刚开始定位时位置经常不准，此时让步兵车多跑几步就校正回来了。

### 3.4.2 导航算法

本组的路径规划算法尝试过 TEB、DWA，最终采用的是 TEB 算法。

调试时发现，车经常在导航中失去方向原地旋转。在导航的参数设置上，步兵车对于障碍物不太敏感，敢于加速冲撞。

### 3.4.3 跟踪射击

跟踪射击方案是通过 PnP 解算，获取装甲板与摄像头之间的旋转矩阵与平移矩阵，进而得到位姿信息反馈控制。检测到装甲板后，PnP 解算将装甲板的相关信息丢给底层，直接让底层控制，云台追踪目标装甲并适时射击。

同时将激光新建一个节点去订阅激光雷达原始数据 `/scan`；将订阅到的原始数据划分四个区（激光雷达按照本次的装配情况车车头是 180 方向，车尾为 0 度，雷达扫描一周出来 360 个数据）。四个区分别是车的前后左右，夹角 30 度，依次统计四个区的障碍物低于 0.7m 的个数，数量越多。避障等级越高，相应的避开。这样就能保证车在设计的过程中左右前后移动同时也不会和周围障碍物撞击。

射击模式如图 3.4.3.1 所示：

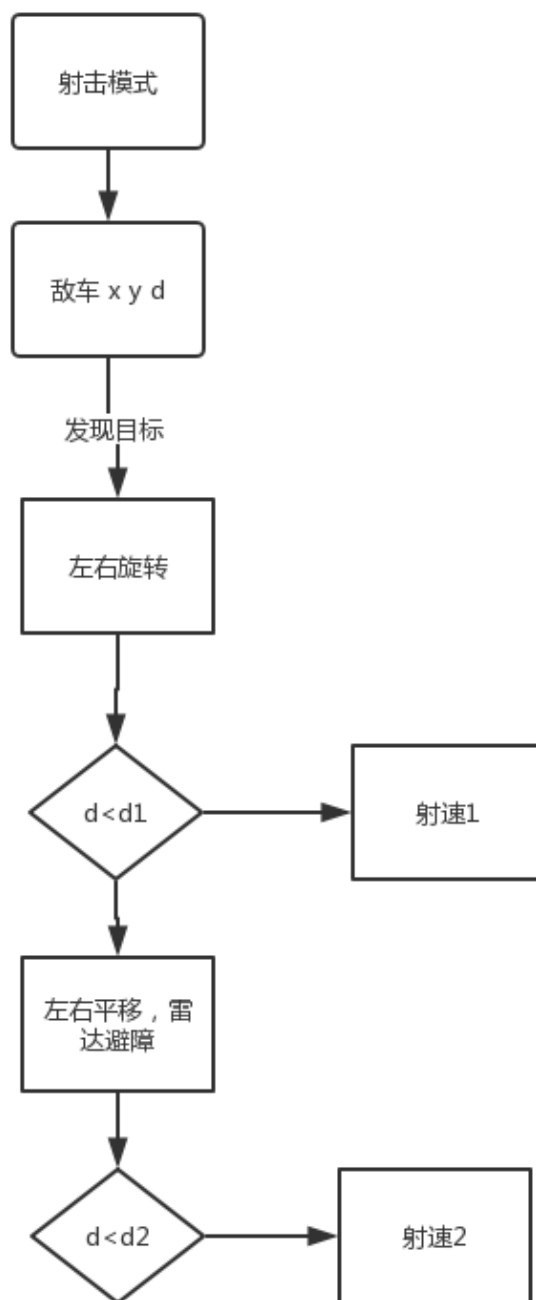


图 3.4.3.1 射击模式

#### 3.4.4 单兵逻辑

本组的步兵车使用了状态机切换的思想。步兵车具有开局加弹、巡逻、射击、三个模式，每种模式对应有敌人/无敌人两种情况。



在开局加弹模式下，步兵车开局开到指定位置，车头对外等待飞机完成接受指令。中途发现受到攻击转入攻击模式，向飞机发送指令停止加弹，飞机继续悬停。步兵车与飞机连在同一子网下，ROS mater 在步兵车上，通过 ROS publish/receive message 机制通信。加完弹后，在巡逻模式下，步兵车会跑定点。当摄像头捕捉到地方装甲板时，步兵车自动切换到射击模式，时刻和敌车保持一个 3.5-4.5 米的距离。同时将控制交给底层，嵌入式。边打边转，左右移动（别人打不中自己，也不好打中别人）。当摄像头又丢失目标后，步兵车又切换到巡逻模式。但在第二阶段正式比赛时，由于导航系统突然不稳定，步兵车加弹后放弃了导航的控制，始终停留在启动区。单兵逻辑如图 3.4.4.1 所示：

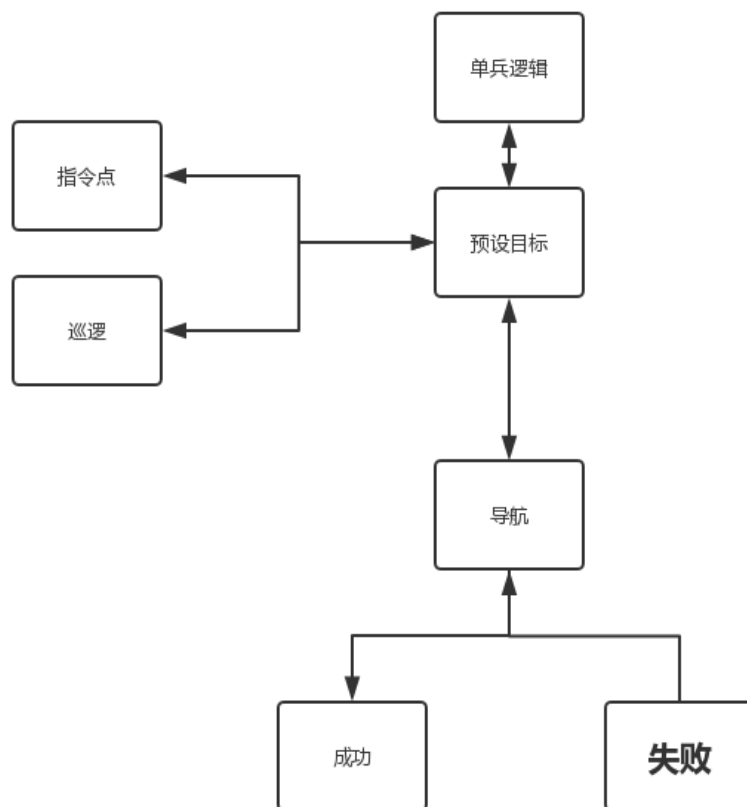


图 3.4.4.1 单兵逻辑

一旦导航失败车就在场地中变成一条咸鱼！此时的控制逻辑如图 3.4.4.2 所示：

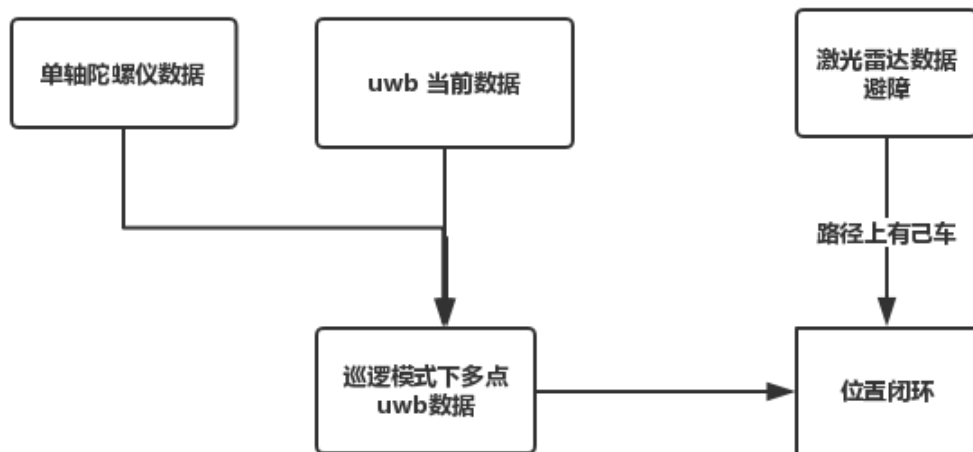


图 3.4.4.2 导航失败后控制逻辑

M100 飞行器控制整体框架的设计也使用了状态机的思想，具有起飞、飞至定点、识别取弹机构、投弹四个状态。由于在第二阶段中加一次弹已经足够了，本组在第二阶段最终采用了手飞降落方案，没有加入识别停机坪/自动返航的状态。

M100 根据时间/事件触发在四个状态之间的切换：初始位于起飞状态，键盘按下’t’键后起飞，切换到飞至定点状态，根据位置期望飞至步兵车处；当罗技摄像头连续计数 10 次看见取弹机构后转化坐标系，切换到识别取弹机构状态。当 X 轴、Y 轴的期望与实际值相差小于阈值且持续超过 8s 后，切换到投弹状态，下落投弹。

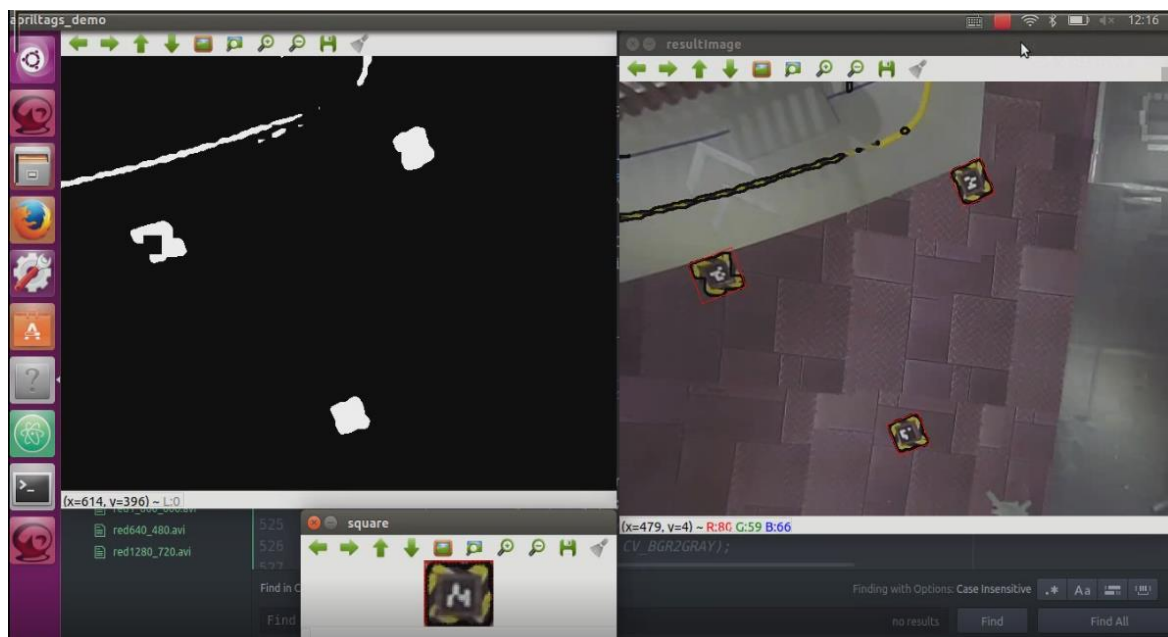
### 3.4.5 多兵作战

多兵作战时，三个组的步兵车不直接通信，通过三个操作手在操作间查看步兵车状态信息，降低了难度。

### 3.4.6 控制

通过串口写告诉底层期望的三轴速度以及 yaw 角度。

### 3.4.7 二维码识别



在第一阶段，赛场两侧启动区分别有 1 个二维码，场地中 8 个黄黑边框中有随机对称放置的两个二维码，车需要视觉提供二维码的位置信息自动遍历随机放置的四个二维码，用时短者胜。

本组发现用现有的 `apriltags` 和 `acruo` 库识别二维码均基于角点识别，在复杂场景下很容易产生误识别，因此决定识别二维码所在的黄黑边框和规则打算识别单侧的三个黄黑边框，加入相对位置以及场地中单侧四个黄黑边框中只有一个二维码的条件，给出中间两个二维码实际编号，并索引之前用 `uwb` 得到的 8 个黄黑框位置坐标，得到中央区域的两个二维码位置。

算法实现步骤如下：

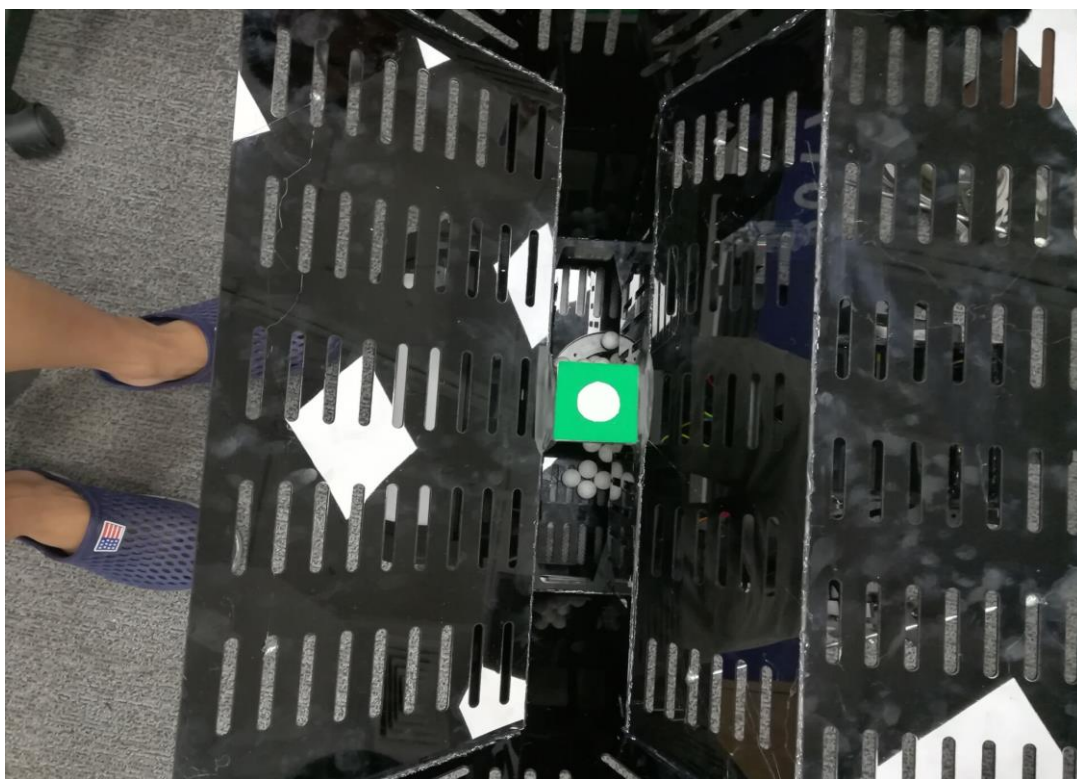
- 1 在原图基础上直方图均衡化增大边框中黑色与背景灰色的对比度；
- 2 自适应阈值二值化分割图像中黑色部分得到二值图一；
- 3 在原图基础上屏幕周边均匀设置种子点采用漫水填充滤掉暗色区域；
- 4 在 HSV 颜色空间设置阈值分割黄色区域得到二值图二；
- 5 图一图二采取与运算，并中值滤波过滤小面积噪声点，闭运算得到黄黑边框的实心矩形；
- 6 利用 `findcontours` 找到图像中的轮廓，进一步利用最小外接矩形得到矩形，通过矩形长宽比以及三个框间距离比设置阈值得到最终三个黄黑边框；

7 基于原图利用 `werpPerspective` 仿射变换分割黄黑边框所在矩形；

8 设置掩模过滤黄黑边框，并二值化，`sumS` 求和得到中心区域面积（有二维码方框中心灰度值面积很高），利用面积站整个方框面积比设置阈值判断该黄黑边框内是否有二维码；

9 如果三个边框内都没有二维码，则判断二维码在四号边框内；

实际完成度达到 80%，飞行高度在 2 到 4 米区间内能得到稳定的编号输出，但是由于时间关系，对面启动区的二维码坐标并没有识别；



### 3.4.8 弹舱识别

任务需求剖析：飞行器在停机坪取弹后，需要自动起飞识别己方战车对其进行加弹；

方案设计：

较小的被识别面积有利于飞行器在持续识别并动态调整位置的前提降低投弹高度，减小子弹入射速度和散落区域面积，从而使投弹命中率大大提高，并且能留出更大的空间安放其他机构和设备。结合颜色特征考虑，赛场中并没有三原

色中的绿色，因而决定采用 5cmx5cm 的绿色方块，内部加白色圆进行识别。

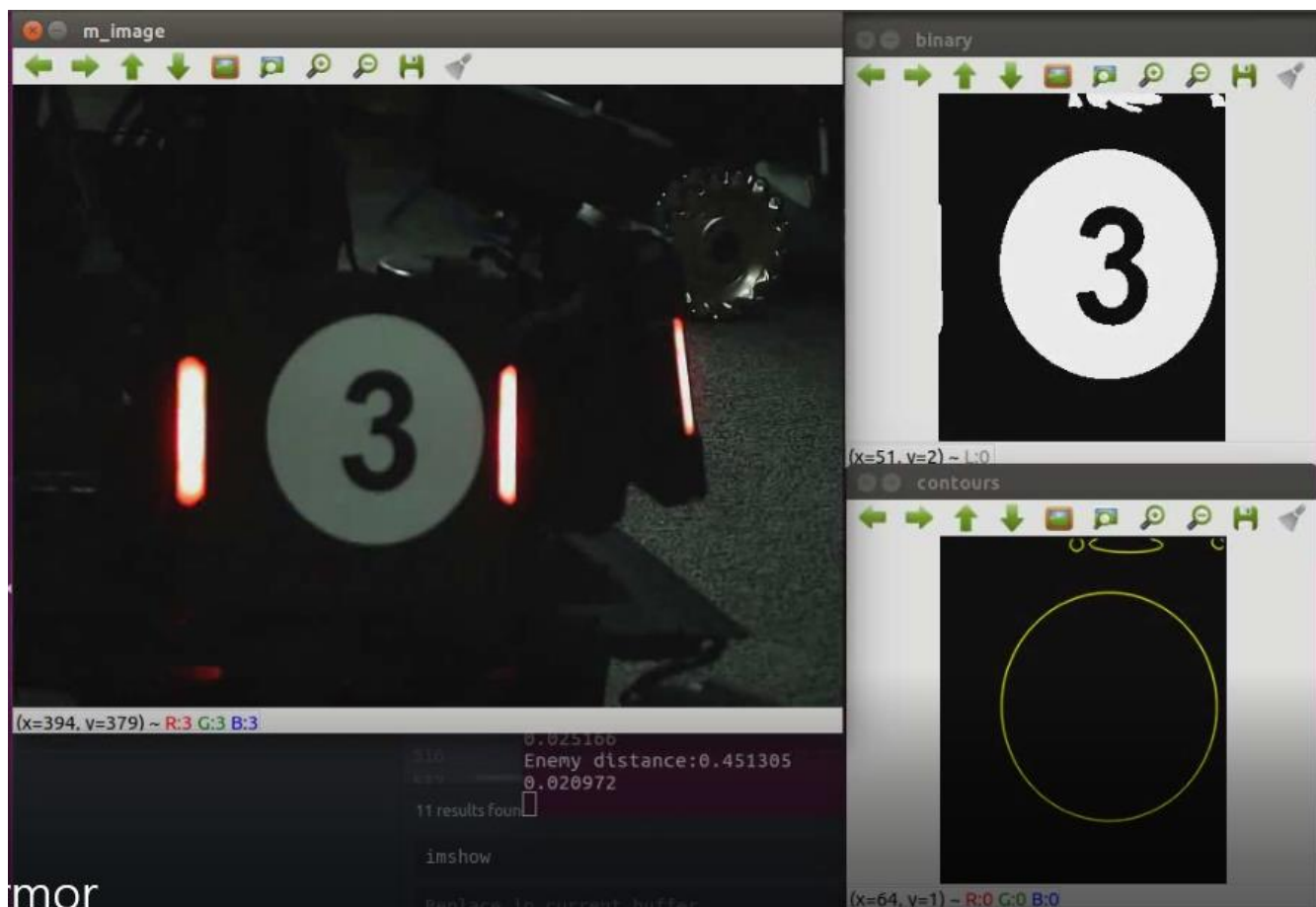
算法实现：

- 1 基于原图转换为 HSV 空间，设置阈值，初步提取绿色；
- 2 中值滤波去除少量的椒盐噪声；
- 3 `findContours` 寻找轮廓并 `meanRectArea` 找到图像中的最小外界矩形；
- 4 根据飞行高度设置最小面积阈值和矩形长宽比阈值筛选最终矩形；
- 5 考虑到飞行器偏航角误差较小，根据矩形四个角点的相对位置对其进行排序，左上角开始，顺时针为 0，1，2，3。
- 6 利用 `solvePnp` 得到绿方块相对于相机的三维位移向量和三维旋转向量；
- 7 结合飞机四元数对位移向量做旋转修正，得到飞行器相对于绿方块的三个相对距离作为反馈量。

完成度总结——100%

在指定飞行高度下，在战车装甲板外围 0.8 米距离的允许误差方框内，能较好地识别绿块，并下降至距离绿块 0.6 米高度投弹，效果较好。

### 3.4.9 装甲板识别



子就磕住了圆环无法继续取弹。因此把机械爪做小一点或支持上下伸展能方便多次取弹。

任务需求剖析：

识别红色或蓝色装甲板框，提供摄像头相对于装甲板的三维距离。

方案设计：

基于官方提供程序得到的 `possible armors`(识别灯条以及通过两两灯条间的平行度和距离长度比值，得到可疑灯条)

算法实现：

1 根据 `possible armors` 四个角点，高度扩大一倍得到新的四个角点，分割出包含完整圆形标签的图像一；

2 将原图固定阈值二值化处理，拟合椭圆；

3 遍历椭圆得到图像中面积最大椭圆；

4 根据椭圆中心点  $x, y$  相对于图像一宽，高的比例设置阈值以及该椭圆外接矩形与分割得到的图片一的面积比设置阈值判断是否为装甲板；

5 继续用 `solvePnp` 得到装甲板到摄像头的三维距离。

四 完成度总结——60%

通过设置曝光值可以让程序对光照条件的敏感度降低，但是有时会将尾灯柱中央识别成椭圆，造成误识别，而且程序有时候会崩（传参可能有问题），故最终只是采用 `possible armors` 作为输出。

## 3.5 测试结果

实际测试时路由器 wifi 信号延时成了最困扰我们的问题，各组之间路由器干扰很大。我们尝试把路由器放车上、车和飞机用两个路由器、调试时用 5G 信号、选择空闲信道等方法，但最终该问题也没有得到很好地解决。

## 3.6 可优化方案

1.本组的机械爪方案具有取弹快、一次取弹量大的优点。但在第三阶段的比赛中，飞机需要自动落在停机坪上以实现多次加弹的功能，机械爪容易磕住圆柱形边框，给降落时的精度带来了挑战。飞机降落到离停机坪 1m 高度时圆环已经



超出了摄像头视野范围，无法通过视觉校正，此时 landing 会受气流扰动影响不是竖直下落，因此给算法控制带来了挑战。因此机械爪可以在体积上进行优化。

2.战车的鲁棒性有待提高，比如战车在导航失败时，如何快速实现快速启动。

3.通过视觉 pnp 算敌车距离当敌车距离较远的时候距离算的不准，这时候雷达的作用没有很好的发挥起来，本来视野中有敌车的时候，雷达是可以扫描到敌车的位置，同时它也不属于地图，可以融合图像搜索确定地方位置。

4.导航算法的运动参数调节的不是很优化，运动的时候有时有点突兀。

## 4 夏令营感想、总结

感谢队友的信任和支持。

感谢大疆组委会工作人员的辛苦付出。

感谢大疆创新给我们提供了这样一次学习和实践的机会。





RoboMaster 大赛组委会

邮箱：[robomaster@dji.com](mailto: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

ROBOMASTER™ 是大疆创新的商标。

Copyright © 2017 大疆创新 版权所有