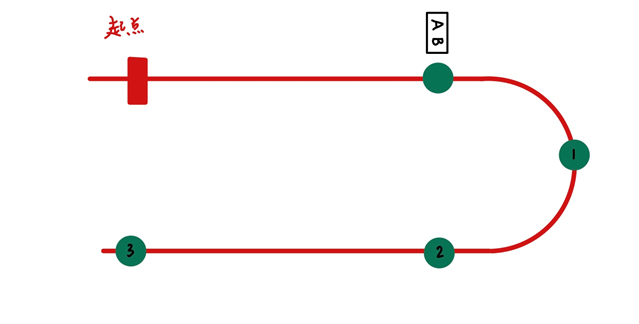
## 1.竞赛内容概述

1）比赛规则和场地的改变：



比赛场地示意图

比赛规则：

主要任务：比赛队伍从下水点出发，依次经过三个打卡点，完成闪灯打卡，并拍摄存储照片；其中，当经过字母牌目标物处，需拍摄目标物照片，并作出字母识别，即刻将字母识别结果发送至电脑，当全部打卡完成出水后，即刻将打卡照片和字母照片回传至电脑。全程时间给定3分钟，根据任务完成度和实际花费时间来评分。

评分标准：

1. 经过一个打卡点，闪灯则初步判定打卡成功，闪灯次数与打卡点号码需一致；待打卡照片回传检查无误后，最终判定打卡有效；对于每一个打卡任务，初步判定打卡成功则得10分，照片回传无误则加得10分，三个打卡任务总计满分60分。（说明：LED灯带仅作为导航辅助，不要求必须沿灯带运动；打卡必须依次按顺序进行；打卡照片有效的标准是打卡点绿色圆饼上的数字能拍到。）
2. 字母牌两个字母均能拍摄到则得10分；每个字母识别正确则各得10分，两个字母识别均正确则再加5分；识别结果的先后顺序需与字母牌一致（以靠近起点/池边的字母为首字母），顺序颠倒则认定识别错误。
3. 裁判根据机器人运行稳定性等因素进行主观打分，满分5分。
4. 以得分高低进行名次排序，得分相同的队伍以实际耗时排序。

2）对比赛规则和场地变化的分析：

* 1. 场地由原来的舟山校区游泳池变成了实验室的水池，相比于游泳池，实验室水池尺寸大大减小，从原来的标准泳池尺寸变成了如今的底面积为4m\*2m矩形，水深大大降低，由原来的114cm水深变成了如今的59cm水深，泳池的好处在于比赛场地的尺寸更大，更长的巡线路径给了水下机器人更大的容错率，即使控制算法有着瑕疵导致巡线出现偏差也能利用长巡线路径的优势及时纠偏，以及水深的增加为字母的识别提供了方便，更深的水深意味着水下机器人可以下潜更深，得到更大的视角，字母板上字母的大小也能比小水池的更大，这些都有利于字母识别功能模块准确率的提升；实验室水池的好处在于，游泳池由于存在水循环系统，导致其一直有一定流速的水流干扰，因为水下机器人有几个自由度是无法加以控制的，一定流速的水流不断地冲击极容易使树下机器人的姿态发生意想不到且难以修正的改变，但是实验室的水池基本上就是静水环境，这对水下机器人姿态的控制提供了极大的方便。
  2. 原来等腰直角三角形的路径改变成了更简单的U型路径，二者的难度差别在于：之前因为是路径是直角三角形，所以我们是沿着直角三角形的三条边来行进，因此走的都是直线，最困难的地方在于直角三角形三个顶点中的两个分别要执行90°和135°的原地旋转，旋转的好坏对于巡线以及字母识别都有很大的影响，众所周知在水下大角度的精确转向是十分困难的；如今因为路径变化成U型，虽然避免了大角度的旋转，但是却增加了曲线的巡线，难度在于控制航向角的左右电机旋转产生力偶的不断调整，在水下直升机在向前运动过程中叠加一个控制航向角的力偶使其能够精准地沿着圆弧轨迹进行巡线，这是本次水下机器人控制的关键。
  3. 打卡点由黄色浮球改为绿色铁饼，好处在于布置于水底黄色的浮球体积大而且漂浮时状态不稳定容易摆动摇晃，这对水深提出了很高的要求，但是绿色铁饼，因为其饼状结构和自身质量大的优点，可以紧贴巡线采用的红色灯带布置，相比于黄色浮球体积减小，因此对水深要求大大降低，因此满足了实验室水池进行比赛的要求。
  4. 打卡点打卡不仅仅要求水下机器人透明球罩内的LED灯亮灯，还要求拍摄水下绿色铁饼的图片，水下绿色铁饼上用黑色马克笔写有数字，拍摄到的绿色铁饼图片必须拍摄到相应的数字，这其实就要求了必须紧贴着灯带进行巡线而不能开环巡线，稍微地偏离了巡线路径或者开环巡线就有可能导致拍摄不到水底布置的绿色铁饼图片；最让我们感到困难的是，一开始的场地为三个打卡点，且在圆弧弧顶处没有打卡点，由于我们的小组程序做了识别不到巡线路径时，水下机器人按照最后能观察到巡线路径所确定的巡线参数继续巡线的程序，因此我们小组能够在场地允许的范围内实现所谓“漂移巡线”的功能，因此如果圆弧顶点不存在打卡点，我们小组也能在进行圆弧巡线时，即使水下机器人较大程度地偏离巡线路径导致无法观察到红色灯带时，也能有较大几率成功回到正确的路径上，但是在圆弧顶点增加打卡点后，其实就变相要求了水下机器人必须能够完美巡线，不允许出现较大程度的偏离巡线路径的情况发生，否则将无法对圆弧顶点的打卡点进行打卡拍照。
  5. 字母板的尺寸变得更小了以及识别的字母个数由原来的4个变为了2个，这是由于水深的限制导致水下机器人视野的范围有限，因此字母板的尺寸和字母板上字母的个数和尺寸必须相应地变小；字母的字体由Times New Roman变成了黑体加粗，因为Times New Roman字体的大写英文字母部分的字母边会变得很细很难识别，加之摄像头的成像质量一般，水下拍摄环境复杂，往往拍摄到的字体图像就是糊的，原来的字体就导致了字母识别的难度变得很大；如今使用的黑体加粗，就摒弃了Times New Roman这种字体的缺陷，降低了字母识别的难度；我们小组使用的是caffe神经网络框架并训练了几乎所有常规字体甚至包括部分手写字体，网络层数为三层并迭代了2070次，最后显示的识别准确率高达99%以上，这就保证了我们水下机器人在运动过程中进行字母识别的准确率。

## 2.就某个环节或事情进行描述、创新点

2.1 辅助博雅工道工程师对返修机器人进行质检：

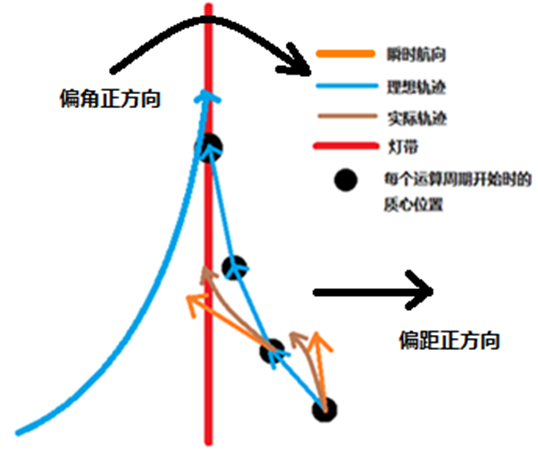
由于机器曾出现大量进水的问题，博雅工道的工程师在暑假实习初期一直陪伴着我们，一起对机器渗水、漏水、进水问题进行攻关，博雅工道的工程师们负责进水的水下机器人维修，我们负责经过维修后机器的检测，最终决定用密封胶将水下机器人所有可能导致水进入密封舱的缝隙封死，不再仅仅依赖于o型圈的径向挤压密封和硅脂的阻水作用，这固然会导致再次拆装和维修的困难，但是经过这次密封胶的涂抹后机器的进水故障问题基本得以解决，我们小组的检测结果为六台机器中有一台机器有渗水起雾情况但勉强可以使用，还有两台机器完全可以使用，没有任何进水的迹象，有三台机器进水仍然严重，但总体上良品率几乎达到了百分之五十，相比于原来未涂抹防水胶的机器的良品率有了极大的提升，最后在26号时博雅工道的返厂维修机器人又到了20台，我们小组再次对其中三台进行质检，结果全部满足比赛要求。

2.2 STM32算法创新点：

我们根据本次比赛的规则调整了STM32的巡线算法，其中最关键的调整在于，**我们的直线巡线和弯道巡线采用了两套不同的参数**，当水下直升机遇到第一个打卡点准备进入弯道时，以第一个打卡点作为参数改变的标志，此时的巡线参数由直道巡线参数变为弯道巡线参数，遇到第三个打卡点时将弯道巡线参数改变为直线巡线参数，直线巡线时速度更快航向角更稳定，但对灯带角度变化不敏感，弯道巡线时的速度较慢，与此同时对灯带角度的变化敏感，这保证了水下机器人巡线的快速性、准确性和稳定性；此外在进入弯道的过程中我们使用了螺旋桨高速反转一定时间，使水下机器人在极短时间内实现较大程度的减速，避免了水下机器人因直线时运动速度过快导致其冲出赛道的情况；其次，因为水下布置用来巡线的红色灯带因其扁平的外观出现侧翻情况，导致红色灯带的光晕分布不均（光晕大量出现在巡线路径外侧），因此水下机器人巡线时常常出现向巡线路径外侧漂移的情况，导致水下机器人时常丢失在圆弧顶点的打卡点（即出现漏打卡的情况），为了解决这种情况，**我在进入圆弧弯道巡线时，在树莓派回传的关于红色灯带偏移量中减去了一个常数，使水下机器人在运动过程中尽量向路径内侧运动，尽可能地避免了弯道巡线时的漂移现象**。

2.2.1 切线导航算法

PID导航算法难以消除直升机在灯带周围的振荡，因此我们自己构思出一种切线导航算法，使得直升机在行进过程中能“一边贴合到灯带，一边修整航向角”。下图的蓝色连续轨迹就是我们的理想轨迹，我们的目标是设计一种新的算法，能够让直升机在周期循环中离散地逼近它。

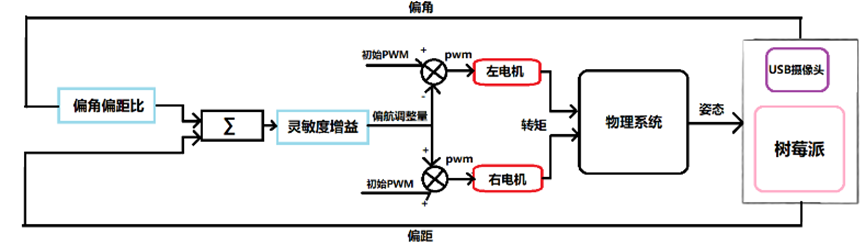


思路：蓝色轨迹中，每一个点处，航向角都指向轨迹的切线方向。也就是说，偏距和切线角度是一一对应的。因此假设在平面上有无数个点，我们根据每个点的偏距和该对应关系，即可画出无数条切线，这些切线的包络线就是一条理想的导航轨迹。所以只凭偏距和切线的对应关系就可以唯一地确定一条平滑的导航轨迹。为了便于控制算法的实现，我们期望这个这个对应关系是线性比例的，并记该比例系数为Navi\_pro。

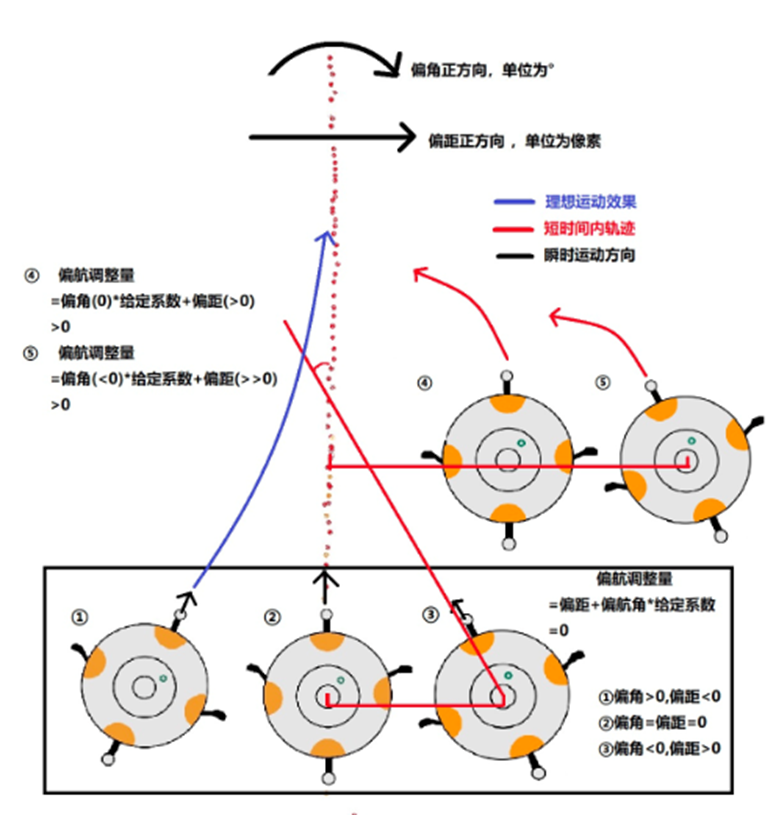
程序实现：每一个运算周期时，STM32读取树莓派传送过来的偏距和偏角，当偏距和偏角满足下述关系式时，直升机延切线方向直行(理想导航轨迹的每一点都满足该式)

Navi\_angle/Navi\_pro+Navi\_dis=0

因此只要以Navi\_angle/Navi\_pro+Navi\_dis为调整量(Navi\_pro就是最关键的那个线性比例系数，根据上图定义的正方向，该系数为一个负数)，那么当瞬时偏距和偏角恰好和理想轨迹相符合时，调整量为0，直升机直行；当瞬时偏距和偏角不符合理想轨迹时，那么在该运算周期内，将非零的调整量以力偶的形式叠加到左右电机上，直升机就会逼近理想导航轨迹。下一个运算周期开始时，再次根据新的位置和新的偏角代入上式计算调整量，直升机会开始逼近下一段理想导航轨迹。周而复始，实际轨迹就会离散地逼近理想轨迹。导航程序的框图如下：



下图为导航的效果示意图：



* 1. 树莓派算法创新点：

我们根据本次比赛的规则优化了树莓派的红色灯带识别算法和字母板定位、分割、提取算法：红色灯带识别算法我们将红色的提取范围进行调整，并调整了关于灯带偏移距离的计算方法，使其识别结果（即根据灯带图像回传的距离和角度参数更加准确）；字母板的定位、分割、提取算法，在原先35步图像预处理算法的基础上，**进行了关于拍摄到字母板发生旋转时纠偏算法的调整**，使得水下机器人的上摄像头即使因为水下机器人姿态不稳（航向角、俯仰角摆动幅度过大）导致拍摄到的字母板图片发生旋转偏移时，仍能获得水平的字母板图像。

2.3.1 灯带：

灯带最为显著的特征是其颜色，因而想到基于颜色的特征区域ROI提取；提取出区域后采用的预处理目的是要将红色不连续的部分连接起来便于检测直线特征；随后检测直线，要考虑速度和需要，多根直线的检测是否必要是主要的讨论之一；最后将检测出的直线规范成偏距和偏航角两个信息输出给主程序。

树莓派识别红色灯带大致思路：

1． 使用Python-Opencv读取摄像头帧后直接进行后续运算会导致图像采集的帧率下降，因而将上下摄像头的图像获取操作单独做到一个进程当中，并且采用模拟堆栈的方式进行数据的共享，这样可以保证主程序始终读到最新的一帧图像进行处理，图像采集的延迟几乎为零。此外，将摄像头采集图像做到单独的进程中可以明显增加识别灯带程序的运算速度，灯带识别速度可以做到5-10帧以上；

2． 我们采用对颜色的特征对灯带图像进行处理，为尽可能减小环境光和亮度对色彩识别的影响，首先将图像转到HSV色彩空间，通过对色调、饱和度、明度的分别截值，提取算法对环境光敏感度降至最低，利用取色软件对亮（晴天，中午），暗（雨天，晚上）等实际光照条件下的游泳池比赛场地的水下灯带采集的图片进行颜色提取操作，最终确定最佳的颜色范围，将颜色的识别精度做到最大；

3． 提取出的ROI后，进行简单的形态学运算保证区域的连通性；

4． 程序调试阶段，采用多种方式拟合直线，各有优缺点：霍夫法首先将图像变换到霍夫空间，按照容许角度和距离误差判断直线，随后变换到原图像空间；霍夫法的优点是可以同时检测出多根直线（灯带），缺点是霍夫直线检测耗时长，且需要对检测出的多条直线进行取舍，算法复杂度高；由于本算法单元对快速性要求极高，因而最终采用最小二乘法：首先对所有联通域进行轮廓发现，而后对每个轮廓的所有点集合进行最小二乘直线拟合（实质为一步矩阵运算）；最小二乘拟合法时间开销极小，且对水体散射，池底反射，底罩水雾影响的鲁棒性高，但在处理两根接触在一起的直线时体现出一定的弊端，转弯时需要其他信息（例如黄球）的辅助；

5． 检测出直线后，简单运算换算出水下直升机行进偏航角和偏距信息，最后通过串口发送到stm32

2.3.2黄球

识别球有很多种方法，可以采用轮廓发现然后对发现的轮廓进行霍夫圆检测，也可以通过球颜色（黄色）来获取球的大小和位置，我们采用的是颜色的方式。这种方式利用了球颜色特殊性将球与背景、杂质分离开来，只要颜色的阈值范围选择正确即可得到球的大小和位置信息，为运动控制提供必要信息。

球的检测流程如下：

1. 获取水下拍摄的包含黄球的图片frame，并得到图片的高H和宽W；

2. 将图片颜色空间从RGB转成HSV；

3. 用取色器获取黄球黄色的上下限，并根据上下限来得到黄球的二值图像th1，；

4. 将二值图像进行稍微膨胀以获得完整轮廓；

5. 从二值图像中得到图像的矩M；

6. 从图像矩中得到黄球的质心位置cX和cY

7. 计算黄球质心位置与图像质心位置在x和y两个方向上的距离cX和cY；

将cX和cY回传至stm32。

2.3.3字母

字母识别的难度主要是如何将白色背景字母板上的黑色字母完整清晰地提取出来，水下拍摄环境中，有大量的环境光干扰导致目标图像周围存在着大量难以摒除的杂质，使得目标图像的获取变得十分困难，所以我们图像提取分割算法的主要目标就是将环境光的干扰尽可能地减小以及获得清晰完整的字母图像。我们先后编写了三套完整的识别代码，但前两套虽然对空气中拍摄的字母板图像处理效果极佳，但是在水下环境中的表现一般，因此我们依据水下环境的一些特点来编写了第三套识别代码，获得了符合预期要求的字母图像。

卷积神经网络的训练是我这个学期一开学的工作，当时我分别学习并成功尝试了caffe和tensorflow两种知名的神经网络，最终因为caffe在机器人的树莓派上已经配置成功以及机器人的初始网络配置无法成功连接互联网（机器人供应商三令五申不能改变机器人的网络配置以免出现异常），我最终决定使用caffe来做识别程序。因为考虑到水下环境对字母图像的影响（造成字母图像的模糊、畸变、小部分残缺等），虽然比赛规则中已经规定字母均为大写且使用new romance这种字体，但我仍然选择使用多种字体再加上部分手写字体对caffe进行训练以提高识别的鲁棒性，事实证明我的选择是正确的（很多小组在空气中拍摄图片识别效果好但是一旦识别水下拍摄的图片识别效果就急剧下降，我觉得大部分原因都是他们只采用了规定字母字体进行训练，使得识别的鲁棒性不高的原因）。

字母识别程序流程如下（其中的英文单词代表经过此方法处理后得到的处理结果，也对应着程序中的变量名）：

1. 获取主函数传递的摄像头图片frame，并得到它的高H和宽W；

2. 将图片转成灰度图gray；

3. 对灰度图进行二值化thesh1处理（二值化的阈值范围由预演时获取的图片来决定）；

4. 对灰度图再进行局部二值化thresh2；

5. 对两张二值化的图片进行与操作，得到新的二值化图thresh；

6. 发现thresh的轮廓contours0；

7. 将thresh图像contours0中不满足轮廓点数的噪声杂质涂黑，以达到去除杂质的目的；

8. 对thresh图像进行闭操作closed，使字母连接在一起；

9. 对closed进行发现轮廓contours1操作，并找出closed所有轮廓中的最大轮廓cnt；

10. 创建一张与原图像大小相同的黑色图像black0，并扩展成800x800的图像black；

11. 对cnt进行获取最小外接矩形rect；

12. 得到rect的四个角点box；

13. 因为black是800x800的图像，原图像时640x480的图像，所以box的位置应调整；

14. 在black上根据调整后box的位置绘制最小外接矩形；

15. 对black进行闭操作dilate，将外接矩形范围稍微扩张；

16. 从rect中获取最小外接矩形的角度angle；

17. 将原图像frame也扩展成800x800的图像；

18. 确定旋转中心center为扩展后图像的中心（400，400）；

19. 将dilate和frame以旋转中心center和角度angle进行旋转修正图片，之后图像分别为rotated和rotated2；

20. 对rotated进行发现轮廓，将第一个轮廓赋予cnt；

21. 对cnt进行外接矩形框选，得到左上角的坐标x，y和轮廓的宽w和高h；

22. 根据x，y，w，h在rotated2上对目标区域进行割取，割取后的图像为crop\_img；

23. 将crop\_img转成灰度图gray；

24. 将gray进行局部二值化thresh；

25. 对thresh进行闭运算填充图像的孔洞和连接断开的区域；

26. 对thresh进行发现轮廓contours3并以像素数作为判据来去除杂质点；

27. 再次对thresh进行闭运算填充图像孔洞和连接图像

28. 对thresh发现轮廓contours4；

29. 对每个轮廓进行矩形框选并获取所有矩形框的x值；

30. 按照矩形框的x值对矩形框进行从左到右排序cnt\_sorted；

31. 将排序好的cnt\_sorted获取其对应矩形框的x，y，w，h并在thresh上割取crop0；

32. crop0按照外接矩形的外接圆半径的1.2倍扩展其大小后的图像为crop；

33. 对crop进行高斯模糊的图像为pi0并调整其大小为64x64的图像pi；

34. 将pi图像保存n.png，1<=n<=num(排序长度)；

35. 加载已经训练好的caffe模型（用手写字母和大量不同字体的字母进行训练，提高识别程序的鲁棒性，且一共迭代训练了2070次，最后训练结果的准确率达99%以上）并将n.png送入卷积神经网络中进行判别；

36. 卷积神经网络所需文件主要包括：神经网络结构（.prototxt），类别标签（.txt），训练完成后的参数（caffemodel）。只要在上位机（树莓派）中成功配置caffe所需环境，将这三个训练完成后得到的文件复制到上位机中，即可成功运行识别程序；

37. 每个字母图片送入卷积神经网络中进行识别后会得到13个（比赛规定字母识别范围是大写字母的A~M）置信度，在类别标签文件中寻找置信度最高的对应的标签作为结果返回；

38. 将判别的结果按与字母板排列相同的顺序回传至stm32。