西电机器人夏令营

SUMMERCAMP2018

四组技术报告

进度安排表

项目	工期	开始时间	结束时间
1、规划、方案讨论	1天	7.26	7.27
2、方案初步设计			
机械：确定机械方案、验证可行性、方案设计以及加工	3天	7.27	7.29
嵌入式：① 沿某一墙壁走出迷宫 ② 可识别路口走出迷宫 ③ 拿某一个路径最短魔方走出迷宫 ④ 拿三个魔方走出迷宫 ⑤ 识别拿复原魔方，走出迷宫	1天	7.28	7.29
算法：熟练操作opencv并处理图片理解开源代码并尝试优化确定定位方法	3天	7.28	7.30
3、方案的讨论、迭代			
机械：改进机构、结构优化	4天	7.30	8.2
嵌入式：① 小车行走超声波测距微调 ② 小车路口精确识别 ③ 小车行走优先级确定：左中右 ④ 小车转速过高过快的协调 ⑤ 小车旋转过程中出现的失误以及调整	9天	7,31	8.7
算法：相机标定 排除干扰 树莓派里opencv环境的配置 魔方颜色识别 测试调试	5天	7.31	8.4
4、测试与调试			
机械：装车	2天	8.3	8.4
嵌入式：代码参数测量以及微调	2天	8.8	8.9
5、总装完成、优化			
机械：总的加工优化	7天	8.3	8.9
嵌入：调试	1天	8.9	8.9
算法：路口的暴力匹配 树莓派的串口通信 优化方案	5天	8.5	8.9

目录

1 机械部分

1.1设计动机

本次比赛要求每个组自主研发一辆全自动机器人战车，寻找魔方并穿越迷宫。机械组的目标是设计合适的机械臂从而夹取魔方，尽量达到易于通过算法控制并易于夹取的目的。

1.2 设计需求

通过算法控制机械臂前伸，后缩，夹取，并转动放置魔方。各功能的优先级依次为：前伸后缩，夹取，转动放置。设计时间一天，绘制时间三天，修改时间五天，打印组装时间一天，总设计时间约为十天。

1.3设计方案

我们组所设计的机械臂大致可分为三部分，底盘控制部分、臂身部分和手爪部分。底盘部分由底座和可360度转动的舵机构成，主要负责机械臂的转动，从而将夹取的魔方放到战车的后部；壁身部分由两部分构成，肘部通过舵机可前后伸缩，从而调整距离以夹取魔方；手爪部分由两个齿轮控制机械爪开合，而两个齿轮也分主动轮和从动轮，主动轮由第三个舵机带动并控制。

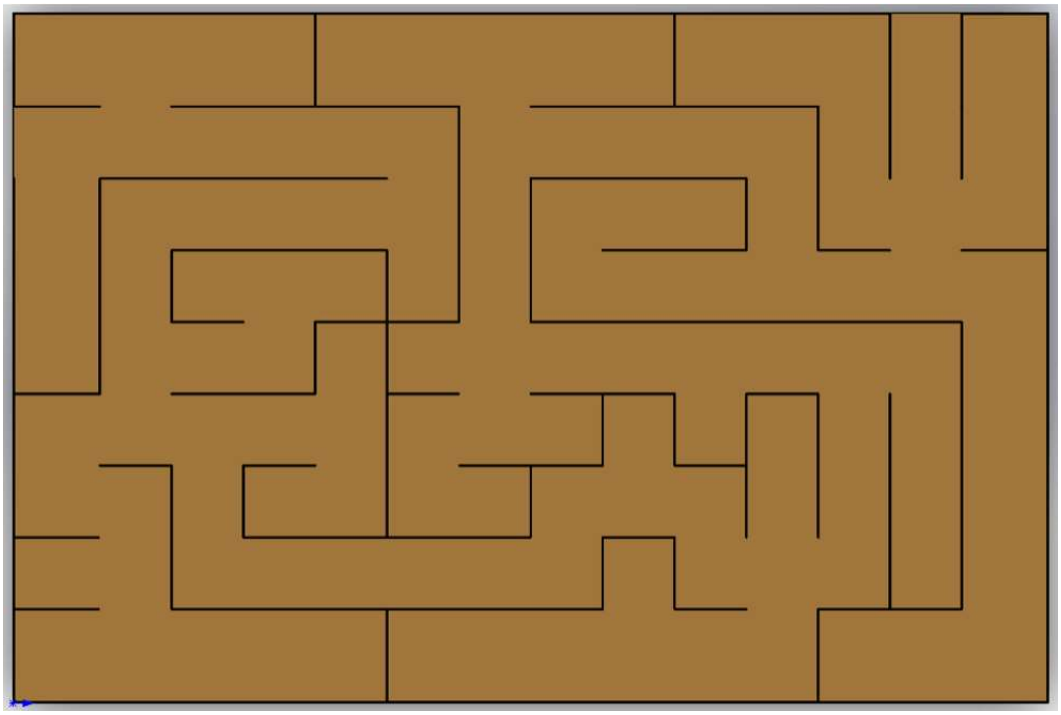
1.4方案的优点与不足

优点：舵机需求较少，结构比较简单，易于控制和配合。
缺点：机械爪部分通过角度的改变控制开合，为防止重量过大，仅为夹取条，夹取魔方时仅是两条线接触，故而摩擦力较小，夹取尤其是转动放置时容易脱落。

2 嵌入式部分

2.1整体方案

迷宫描述



小车数据

32×26cm　转弯半径20.6cm

目的：最快时间走出迷宫

沿某一墙壁走出迷宫

可识别路口走出迷宫

拿某一个路径最短魔方走出迷宫

拿三个魔方走出迷宫

识别拿复原魔方，走出迷宫

代码思路：针对多个状态下小车动作，构成闭合反馈

代码控制

传感器：超声波（4mm~4m）、摄像头

控制器：马达（速度范围）、舵机、机械臂（位置以及抓取范围）

相干条件

内部：

电池——抓取速度、形式速度、过弯速度

车体本身

摄像头——识别距离、识别速度

树莓派

超声波测距模块——识别距离、识别速度

StmF4

机械臂——舵机功率、抓取范围、抓取速度

外部：

迷宫路况——直道

```
while（1）（速度判定函数）
    {    if（正前方超声波>1m）
         { 马达占空比1000；延时1秒}
         else if （正前方超声波小于100cm||正前方测距>50cm）
         {马达70%负荷行驶；}
        else if（正前方超声波小于50&&正前方超声波>32cm）
       { 马达负荷200；}
       else if （正前方超声波<10cm）
       { break;}
    }
while（1）（位置校正函数）
    {    if （左边-右边>5cm ）
         {向右平移  延时0.3秒  break；}
         else if （右边-左边>5cm）
         {向左平移      延时 0.3秒 ；break；}
         break;
         速度判定;
    }
```

迷宫路况——路口判断

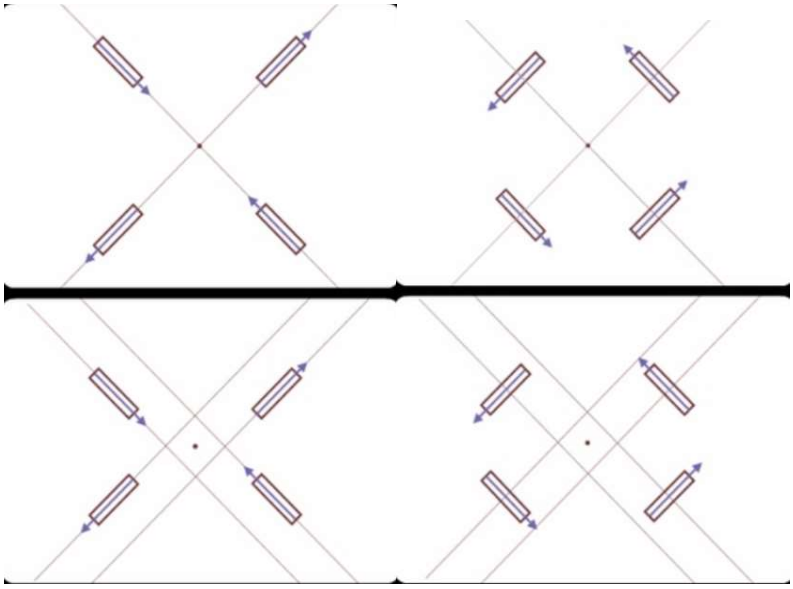
- ①if（超声波三个方向都检测>32cm）（十字路口）
  - {　　小车停止；
  - 左转90°；
  - 延时2秒；
  - 直走；
  - 延时两秒；（保证已经走出路口）
  - }
- ①if（左右>32cm||超声波前<12cm）
  -
  - {　　小车停止；
  - 左转90°l；
  - 延时两秒；
  - 直走；
  - 延时两秒；保证走出路口；
  - }
- 通用if（超声波三个方向都小于12cm）（死胡同）
  -
  - {　　小车停止；
    - 转弯180°；
    - 延时2两秒；
    - 直走；
  - }
- ②if（ 超声波三个方向都检测>32cm ）（十字路口）
  - {
  - if（相机是否检测到正前方有图案）
    - {
    - }
  - switch（）
    - 树莓派传来的值=xx；右转；
  - 几号路口怎么转，怎么走；break；
  - 转弯90°；
  - 延时2s；
  - 直走；
  - 延时2s；

2.2运动学解算方法

基于麦克纳姆轮的运动解算

麦轮的安装方法

麦轮一般是四个一组使用，两个左旋轮，两个右旋轮。左旋轮和右旋轮呈手性对称。安装方式有多种，主要分为：X-正方形（X-square）、X-长方形（X-rectangle）、O-正方形（O-square）、O-长方形（O-rectangle）。其中 X 和 O 表示的是与四个轮子地面接触的辊子所形成的图形；正方形与长方形指的是四个轮子与地面接触点所围成的形状。



X-正方形：轮子转动产生的力矩会经过同一个点，所以 yaw 轴无法主动旋转，也无法主动保持 yaw 轴的角度。一般几乎不会使用这种安装方式。

X-长方形：轮子转动可以产生 yaw 轴转动力矩，但转动力矩的力臂一般会比较短。这种安装方式也不多见。

O-正方形：四个轮子位于正方形的四个顶点，平移和旋转都没有任何问题。受限于机器人底盘的形状、尺寸等因素，这种安装方式虽然理想，但可遇而不可求。

O-长方形：轮子转动可以产生 yaw 轴转动力矩，而且转动力矩的力臂也比较长。是最常见的安装方式。

麦轮底盘的正逆运动学模型

以O-长方形的安装方式为例，四个轮子的着地点形成一个矩形。正运动学模型将得到一系列公式，让我们可以通过四个轮子的速度，计算出底盘的运动状态；而逆运动学模型得到的公式则是可以根据底盘的运动状态解算出四个轮子的速度。需要注意的是，底盘的运动可以用三个独立变量来描述：X轴平动、Y轴平动、yaw 轴自转；而四个麦轮的速度也是由四个独立的电机提供的。所以四个麦轮的合理速度是存在某种约束关系的，逆运动学可以得到唯一解，而正运动学中不符合这个约束关系的方程将无解。

先试图构建逆运动学模型，由于麦轮底盘的数学模型比较复杂，我们在此分四步进行：

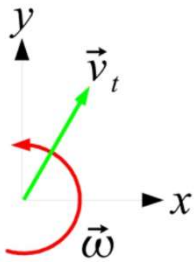
①将底盘的运动分解为三个独立变量来描述；
②根据第一步的结果，计算出每个轮子轴心位置的速度；
③根据第二步的结果，计算出每个轮子与地面接触的辊子的速度；
④根据第三部的结果，计算出轮子的真实转速。

一、底盘运动的分解

我们知道，刚体在平面内的运动可以分解为三个独立分量：X轴平动、Y轴平动、yaw 轴自转。如下图所示，底盘的运动也可以分解为三个量：

表示 X 轴运动的速度，即左右方向，定义向右为正；
表示 Y 轴运动的速度，即前后方向，定义向前为正；
表示 yaw 轴自转的角速度，定义逆时针为正。

以上三个量一般都视为四个轮子的几何中心（矩形的对角线交点）的速度。



二、计算出轮子轴心位置的速度

定义：

为从几何中心指向轮子轴心的矢量；
为轮子轴心的运动速度矢量；
为轮子轴心沿垂直于　的方向（即切线方向）的速度分量；

那么可以计算出：

$$\vec{v} = \overrightarrow{v_t} + \vec{\omega} \times \vec{r}$$

分别计算 X、Y 轴的分量为：

$$\begin{cases} v_x = v_{t_x} - \omega \cdot r_y \\ v_y = v_{t_y} + \omega \cdot r_x \end{cases}$$

同理可以算出其他三个轮子轴心的速度。

三、计算辊子的速度

根据轮子轴心的速度，可以分解出沿辊子方向的速度和垂直于辊子方向的速度。其中垂直于辊子方向的速度是可以无视的。

四、计算轮子的速度

从辊子速度到轮子转速的计算：

$$v_w = \frac{v_{\parallel}}{cos45^\circ} = \sqrt{2}(-\frac{1}{\sqrt{2}}v_x + \frac{1}{\sqrt{2}}v_y) = -v_x + v_y$$

$$\begin{cases} v_x = v_{t_x} + \omega b \\ v_y = v_{t_y} - \omega a \end{cases}$$

结合以上四个步骤，可以根据底盘运动状态解算出四个轮子的转速：

$$\begin{cases} v_{w_1} = v_{t_y} - v_{t_x} + \omega(a+b) \\ v_{w_2} = v_{t_y} + v_{t_x} - \omega(a+b) \\ v_{w_3} = v_{t_y} - v_{t_x} - \omega(a+b) \\ v_{w_4} = v_{t_y} + v_{t_x} + \omega(a+b) \end{cases}$$

以上方程组就是O-长方形麦轮底盘的逆运动学模型

另一种简单的逆运动学计算方式。

我们知道，全向移动底盘是一个纯线性系统，而刚体运动又可以线性分解为三个分量。那么只需要计算出麦轮底盘在「沿X轴平移」、「沿Y轴平移」、「绕几何中心自转」时，四个轮子的速度，就可以通过简单的加法，计算出这三种简单运动所合成的「平动+旋转」运动时所需要的四个轮子的转速。而这三种简单运动时，四个轮子的速度可以通过简单的测试，或是推动底盘观察现象得出。

当底盘沿着 X 轴平移时：

$$\begin{cases} v_{w_1} = -v_{t_x} \\ v_{w_2} = +v_{t_x} \\ v_{w_3} = -v_{t_x} \\ v_{w_4} = +v_{t_x} \end{cases}$$

当底盘沿着 Y 轴平移时：

$$\begin{cases} v_{w_1} = v_{t_y} \\ v_{w_2} = v_{t_y} \\ v_{w_3} = v_{t_y} \\ v_{w_4} = v_{t_y} \end{cases}$$

当底盘绕几何中心自转时：

$$\begin{cases} v_{w_1} = +\omega(a+b) \\ v_{w_2} = -\omega(a+b) \\ v_{w_3} = -\omega(a+b) \\ v_{w_4} = +\omega(a+b) \end{cases}$$

将以上三个方程组相加，得到的恰好是根据「传统」方法计算出的结果。

2.3机械臂与控制方案

机械臂与底盘

机械臂包括3个自由度关节、1个抓力控制的抓手，以及竖直旋转自由度和水平旋转自由度。机械臂选取的考虑因素：最大负载：因为机械臂一般都是要根据不同要求装上不同的抓手的，很多抓手的重量对机械臂的影响会超出预期，故而最大负载是一个很重要的指标。自由度个数（4轴，6轴）。运动半径：机械臂运动的半径。速度：一般是指的机械臂里电机的最大转速。重复精度：这个是非常重要的，编好程序之后反复执行的多次之间误差有多少。

机械臂基本算法思想如下

1、标定机械臂各关节，获取图像，经过处理得到关节点在空间中的坐标，通过几何运动学解算出此时各关节角的角度；

2、获取图像，经过处理得到目标物体位置，通过几何运动学解算出机械臂要抓到物体所要的角度（目标角度）；

3、计算目标角度与当前角度的误差，使用反馈控制减少误差；

4、循环1~4步，当误差足够小时即可命令机械臂抓起物体。

关于PID的超调、震荡

1．负反馈

自动控制理论也被称为负反馈控制理论。首先检查系统接线，确定系统的反馈为负反馈。例如电机调速系统，输入信号为正，要求电机正转时，反馈信号也为正（PID算法时，误差=输入-反馈），同时电机转速越高，反馈信号越大。其余系统同此方法。

2．PID调试一般原则

a.在输出不振荡时，增大比例增益P。

b.在输出不振荡时，减小积分时间常数Ti。

c.在输出不振荡时，增大微分时间常数Td。

3．一般步骤

a.确定比例增益P

确定比例增益P 时，首先去掉PID的积分项和微分项，一般是令Ti=0、Td=0（具体见PID的参数设定说明），使PID为纯比例调节。输入设定为系统允许的最大值的60%~70%，由0逐渐加大比例增益P，直至系统出现振荡；再反过来，从此时的比例增益P逐渐减小，直至系统振荡消失，记录此时的比例增益P，设定PID的比例增益P为当前值的60%~70%。比例增益P调试完成。

b.确定积分时间常数Ti

比例增益P确定后，设定一个较大的积分时间常数Ti的初值，然后逐渐减小Ti，直至系统

出现振荡，之后在反过来，逐渐加大Ti，直至系统振荡消失。记录此时的Ti，设定PID的积分时间常数Ti为当前值的150%~180%。积分时间常数Ti调试完成。

c.确定微分时间常数Td

积分时间常数Td一般不用设定，为0即可。若要设定，与确定 P和Ti的方法相同，取不振荡时的30%。

d.系统空载、带载联调，再对PID参数进行微调，直至满足要求。

2.4功能模块说明

超声波测距

超声波发射器向某一方向发射超声波，在发射时刻的同时计数器开始计时，超声波在空气中传播，途中碰到障碍物面阻挡就立即反射回来，超声波接收器收到反射回的超声波就立即停止计时。超声波在空气中的传播速度为340m/s，根据计时器记录的时间t，就可以计算出发射点距障碍物面的距离s，即：s=340t/2。

超声波发射电路：由定时器产生脉冲信号，加到超声波探头的引脚上，使内部的压电晶片产生共振，向外发射超声波。

HC-SR04超声波测距模块可提供2cm-400cm的非接触式距离感测功能， 测距精度可达高到3mm；模块包括超声波发射器、接收器与控制电路。基本工作原理：

(1)采用IO口TRIG触发测距，给至少10us的高电平信号;

(2)模块自动发送8个40khz的方波，自动检测是否有信号返回;

(3)有信号返回，通过IO口ECHO输出一个高电平，高电平持续的时间就是超声波从发射到返回的时间。

测试距离=(高电平时间*声速(340M/S))/2;

串口调试

串口调试程序也是在内启动方式下先下载2kb的串口调试程序代码并执行，它通过对任意总线地址读写操作来实现硬件接口的调试。在调试arm前，必须先对核心板的串口工作情况进行调试。为此调试编写的测试程序功能是通过串口输出一个字符串“abcdefg”，经过串口初始化和程序运行，将pc与arm板串口互连接线连接，通过此软件进行串口监听，接收到了正常的字符串“abcdefg”，即表明串口工作正常。然后将pc与arm用专用连接线连接，设置好pc的串口号和波特率，根据通信协议在串口发送查询模块状态命令，并得到返回的数据包，表明与arm

通信正常。整个程序结构比较简单，只是在串口调试的主循环里加入了实现总线读写的调试命令。利用这个串口调试程序，就可以利用串口发送命令，实现对任意总线地址的读/写操作。

PWM电机控制

脉冲：频率，方波

脉冲宽度：占空比，高电平宽度：在同一个时间内，高电平所占比例。

频率与周期：f=1/T

主要用于控制输出的电压和电流，直流电机速度的控制。

2.5难点与不足

超声波微调测距略有偏差，小车识别路口出现问题，陀螺仪参数调节实现具有难度，小车识别图像以及魔法抓取机械臂调整有问题。

第3章 算法部分

3.1开发环境介绍

3.1.1 算法环境

VMware Workstation Pro(Ubuntu系统)，OpenCV 3.3.0

最开始我们在虚拟机上的ubuntu系统上装OpenCV，但是由于对Linux系统不熟悉，一些命令也看不懂，还有之前装的ubuntu版本过低，一些命令无法执行，所以最后改在Windows系统里安装vs2017.

Windows10，VisualStudio 2017，OpenCV 3.3.0

我们队员中有人以前下载过VisualStudio，但是没有安装C++，所以无法输入C语言指令，解决办法就是打开安装软件VisualStudioInstaller修改后重新安装。OpenCV的版本我们一个是3.4.2，一个是3.3.0，具体差别不大，在配置环境的过程中也没有太大问题。但是尝试程序时发现图片打不开，找了很久发现是目标图片没有放入工程文件夹中所致。

Linux，OpenCV 3.3.0（最后在树莓派上实现）

3.1.2硬件环境

树莓派3B+：网线、显示器、鼠标、键盘

树莓派到的那天我们就开始在树莓派上配环境，但是由于我们没有显示器，所以只能通过Windows系统上的远程来调控树莓派。但是我的电脑上找不到树莓派的IP地址，这样就无法远程控制。我们分析是网线的问题，但换了以后还是不行。后来学长说可能是电脑本身配置的问题，所以我们换了别人的电脑成功找到了IP地址连上了远程。

关于树莓派上OpenCV的配置也比较头疼。我们当时用网上的教程配置完成后发现调试不了程序，找了其他很多配置的办法都无动于衷。好在最后我们借到了树莓派和显示器，解决了配置的问题，用显示器也方便了很多。

摄像头：140度广角无畸变1080p 焦距3.5mm

3.2整体技术方案概述

3.2.1 魔方识别:getcube函数

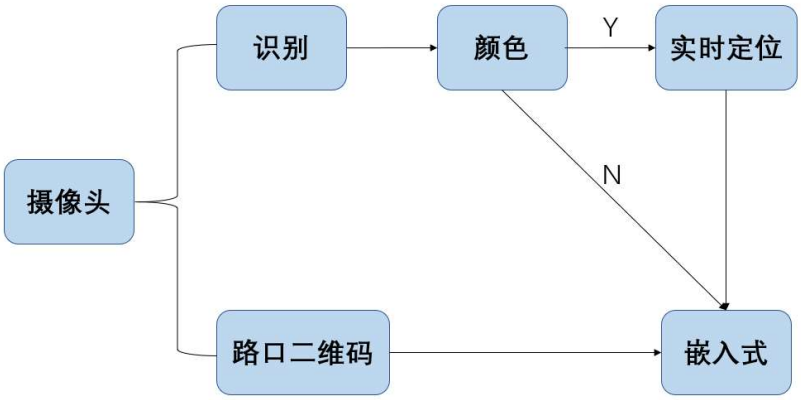
图片灰度处理和滤波处理后，通过findContours+找正方形框，以中心之间距离最小为标准确定的九个方框即为魔方，为bool变量

3.2.2 魔方定位：contours函数

同样的处理，取布尔过程中的方框定点为魔方实际位置，以摄像头图片中心为参照点，进行转化计算，得到魔方相对位置XYZ，回到main的串口输出。

3.3算法整体框架设计

我们打算实现的功能是：识别魔方、魔方定位、魔方颜色识别、路口识别。
已经实现的目标：识别魔方、魔方定位。



树莓派通过对摄像头拍摄的图像进行获取魔方以及路口二维码的信息，把路口的信息由串口发送给电控组。如果识别出颜色单一（即还原的魔方），就把魔方的实时定位发给电控组；如果颜色不是单一的（即未还原的魔方），就告诉电控组离开。

3.4算法功能模块说明

3.4.1识别

我们最开始还是使用魔方开源的代码，但是发现不稳定。所以我们是改开源里的各种滤波，原本是高斯滤波和中值滤波，我们把他们改成方框滤波、均值滤波以及互换他们的顺序。发现还是高斯滤波和中值滤波好用。接着我们把源码的参数改了，发现高斯滤波的Size(1,1)相对比较稳定。所以我们先把摄像头获取的图片转为灰度图，然后高斯滤波、阈值化、中值滤波来处理，找出最外轮廓，用minAreaRect处理得到最小包围矩形。
接着是对获取的矩形进行数据分析，基于魔方的特性，先由长宽比选出方形。接着计算各个方

形中心距离之和，距离之和最小的即设为魔方中心。这个方法计算量较大，而且易受外界干扰，好处是得到魔方定点位置有助于魔方定位模块的进行。

魔方的判断是由一个bool类型checkcube()进行的，识别出9个框，且距离在一定范围内，即确定为魔方。在函数中直接调用bool的返回值，就有了魔方识别的数据。

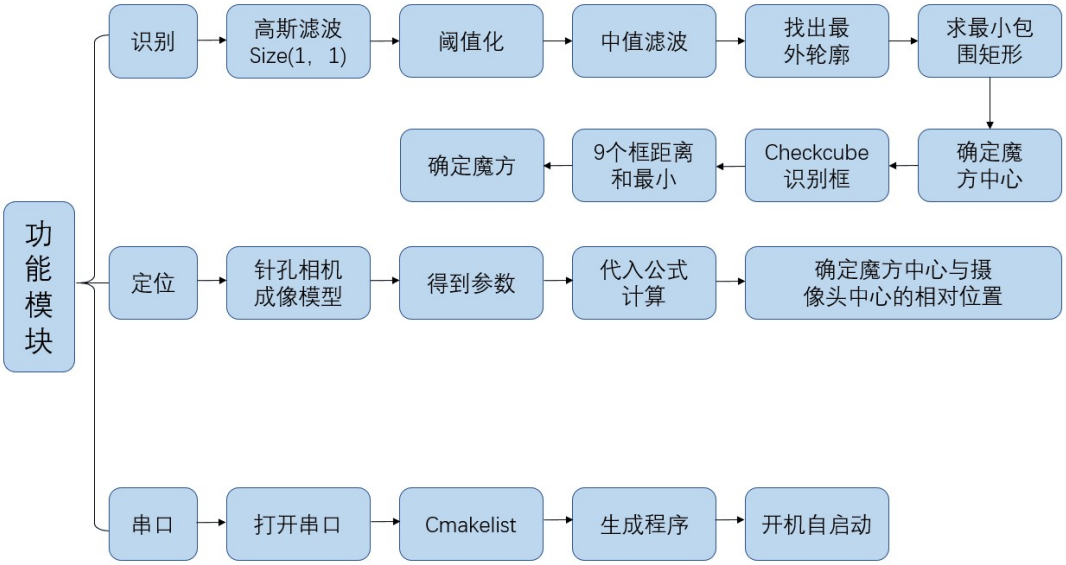
3.4.2 定位

单目摄像头的定位有很多种方法。为了先简单实现功能，我们用针孔摄像机的成像模型，计算到模仿的实际距离Z。

利用布尔中存下的魔方左上角方形的参数，通过公式

像素尺寸 = 目标物的实际长度（cm） / 目标物在图像上的像素长度（pixels）

得到图片距离与实际距离的对应关系。设置摄像头视野中心为参照点，由魔方点点与参照点的相对位置确定魔方的X、Y坐标。



3.4.3 串口

我们对串口通信都比较茫然，于是面向搜索引擎编程一波后发现树莓派3B的串口的奇妙问题：有一个硬件串口还有一个迷你串口，默认的还是硬件串口，不是40pin那边引出来的串口，得手动修改文件配置一波才能正常使用。改的时候问题很多，按着CSDN上的经验帖子尝试，很不幸，刚刚开始就失败了（这时候用的是借到的树莓派和显示器），因为根本没有找到要改动的那些文件。我们想着不能通信怕是要白干，就有点心急，各种尝试，结果第二天起来就是一个大彩蛋——树莓派开不了机了，果然凉凉。

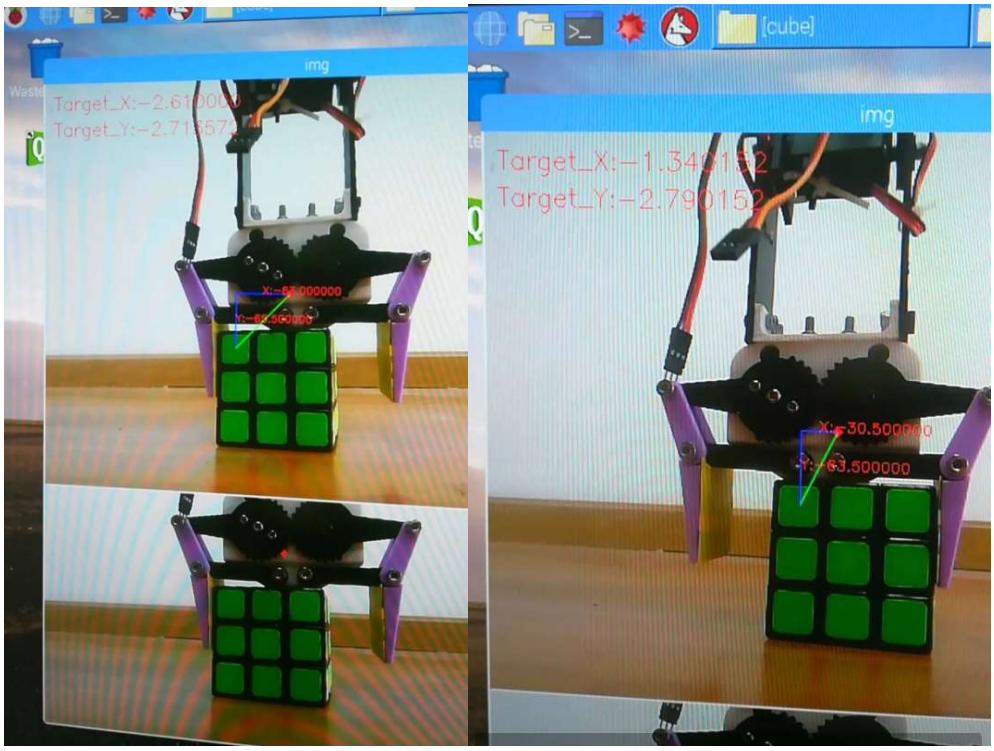
后来我们仔细的分析了一下我们为什么失败：

首先，树莓派对STM32的通信是整个算法发挥作用的基础，没有通信，程序写成什么样都用不上。而我们树莓派到手以后仍然在考虑魔方识别的优化，比较晚才考虑树莓派的使用，而我们两个对使用树莓派都没有经验。任务安排顺序不合理，很大程度上限制了自己的功能实现。

其次，我们缺少Linux系统使用的经验，在很多简单的操作上画了很多时间。在树莓派开机出问题后，在网上找解决方法，才反应过来，原来非正常断电的话Linux系统会在开机自检过程中出问题。才想起来前一天晚上确实是非正常断电的，而事实上，回宿舍用windows远程桌面连接就已经连不上了。当时还想是网线的问题（实在是深受其害），后知后觉。
再有，我们尝试了把借来的树莓派上的TF卡插到我们原有的树莓派上，准备在我们的树莓派上重装系统（毕竟是人家的卡，不敢格啊），结果，不仅我们的树莓派系统装不上，而且，TF卡装回去以后，原本卡在初始界面动不了，但还能输入东西（虽然不是控制界面），变成连输入都不行了。为什么要强调这个，还是因为后来找解决方法的时候才知道：如果是因为非正常断电而开不了机的话，输入root密码，再进行巴拉巴拉的操作，是可以解决的，再次后知后觉，感觉凉得透透的。
最后，其实是最有可能的原因，我们是改配置的时候断电的，当时是更新了系统，重新写Cmakelist，很有可能是改到了哪个地方造成树莓派开不了机的。

3.5测试结果

识别魔方的结果还是不太稳定，受到的干扰比较大。定位方面不够精确，目标点位魔方中心点，但是它检测不明确，一直在周围方格的中心跳动，所以我们和电控组的经过协商以后，决定由我们来测机械臂在魔方上的左右两侧位置，可抓取魔方的XYZ数值范围，我们传送过去的数不用改，由他们控制小车的位置，在这个范围内就可以抓取魔方。但是在这里，我们对机械和电控的组员们表示很抱歉，真的是“死于树莓派”了。
这是测试时的图片：



3.6可优化方案

路口可以始用SIFT配合暴力匹配进行关键的描述和提取，但是这种方法的头文件里包含"nonfree",非免费的文件。别的方法效果不好，可能还是环境没配置好的原因。

关于颜色识别方面，我们一直使用RGB颜色空间，但是就识别方面来说用HSV颜色空间转换更好。所以可以考虑用HSV和其他色彩空间结合的方法，识别效果可能更好

另外关于魔方识别不稳定的问题，不一定要按照识别9个邻近方块的方法，可以通过识别一样的颜色从而判断是复原好的魔方，或者用直接框魔方大框而不是小框的方法，识别度会更高

4夏令营感想、总结

机械组夏令营感悟与总结

时光飞逝，暑期夏令营转眼就结束了。这二十多天收获颇多，除了技术方面能力的长进和团队协作能力的提升，我想实际操作能力也是一项宝贵的收获。实践往往会给纸上兵法致命一击。为什么这么说呢，起初，我们设计的机械臂已经经过了数次改进，表面足够夹取物块，但事实却并非如此，打印出来的机械臂虽能完成一些基本动作，但却显得十分笨拙，魔方不仅不能被夹起，反而会被撞偏，对算法组提出了更高的要求。后来，我们与算法组进行了商讨，改进了爪子部分，加宽了厚度并增加了层数。看似如此简单的功能，真正实现起来却并不容易。 非常怀念这过去的二十天，希望与大家有缘再相逢。

电控组夏令营感悟和总结

感悟：

在这次夏令营的过程中，我们学到了很多的知识，在了解关于stm32F4的诸多知识的基础上，我们开始了学习和控制机器人的道路，电控方面我们做了很多的方案选择以及知识技术上的突破，面对小车行走过程中的诸多问题，我们提出许多解决方案，从零到小车初现雏形，我们付出了很多的汗水。首先是PWM电机的控制，在了解stm32的所有基础知识之后，我们努力探索如何基于单片机控制PWM电机调速，其中不乏很多失误，一次次调整代码和一次次更改以及学习，最终成功实现。关于超声波测距的知识，我们在了解原理后，努力学习如何做出产品实物！在关于PID的探索过程中，我们出现了很多问题，小车的平衡以及行走一开始无法确保，我们不得已手动PID，一次次调整参数，一次次摸索。为了确保小车转弯等动作稳定实现，我们多次尝试陀螺仪的协调以及代码更改，做了许多无用功，但是依然不气馁，努力学习努力奋斗，只为了我们的小车能够更好的实现目标！机器人的夏令营让我们在泪水和汗水中成长，我们学习了很多书本上没有的知识，我们也第一次把知识应用到实践上，在看到成品出来的时候，我们也终于明白一切的努力和奋斗都是值得的！电控的道路上我们成长了许多！

总结：

团队合作与团队内部协调十分重要。在学习和探索小车的行进过程中，离不开团队每一个人的努力和奋斗，小车的实现里面包含每一个人的汗水和努力。

单片机的学习和应用不仅仅只应该停留在理论和表面，实践才会出真知。

面对失败与挫折，我们可以有一时的气馁和伤心，但是不能放弃，及时重鼓勇气解决问题十分重要

算法组夏令营感悟和总结

虽然我们之前敲了许多代码，但是最后由于串口通信的失败导致最后失败。我们最后分析了一下失败的原因。

最主要的原因是前期的计划不够到位。一个好的纲领能指引一个团队的胜利，由于前期对OpenCV和树莓派的理解不够深刻，花了大量的时间去学习理解一些简单的概念。在树莓派到位的时候，我们仍然投入大量的精力在魔方识别的优化上。我们轻视了树莓派在整个项目中的作用，最终导致了通信失败，之前写出的程序完全没派上用场。

此外，对Linux系统使用经验的匮乏也是一个很大的问题，这导致我们浪费了大量时间在重复的工作上。在夏令营的20天，我们从初次接触、到尝试了解、最后真的应用OpenCV，一步步踩着坑走过来，一路上不仅仅在填坑，更是在挖坑。我们缺乏Linux系统的操作经验，本应该在Windows系统上安装系统的但是浪费了太多时间。电脑的问题使得树莓派不能连上远程，而我们在自己的电脑上吊死，没有想到换别人的电脑试一试。当显示器上显示只能初始界面时，我们输入错了提示符，使得电脑死机等等。

最后，我们觉得这个夏令营收获了许多经验，交了许多朋友，这是最可贵的。希望夏令营里这些失败的经验能帮助我们以后走得更远。

项管夏令营感悟和总结

通过这个夏令营，我感觉到管理一个团队真的很不容易。高强度的任务、团队里关于不同方案的争论等每天都让我头疼。幸亏大家都坚持下来了。还有时间进度的安排，只要有一个环节出事故，就会导致后面一系列的任务出现差错。在这次夏令营中我们收获了许多

学会了技术，社交等等。相信这个夏令营能让大家更团结、更坚强、更努力。