西电机器人夏令营

SUMMER CAMP 2018

二组技术报告

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	7.28	7.29
2、方案初步设计	2天	7.29	7.30
机械：确定机械方案、验证可行性、方案设计以及加工	2天	7.29	7.30
嵌入式：确定实现小车各种功能所需要的功能模块	2天	7.29	7.30
算法：确定识别和抓取所需要的各种函数	2天	7.29	7.30
3、方案的讨论、迭代	6天	7．31	8.5
机械：改进机构、结构优化	6天	7.31	8.5
嵌入式：模块整合，总程序推进	6天	7.31	8.5
算法：放弃自己思路，采用学长开源代码	6天	7.31	8.5
4、测试与调试	3天	8.6	8.8
5、总装完成、优化	2天	8.9	8.10

目录

1 机械部分.....3
1.1 设计动机.....3
1.2 设计需求.....4
1.3 设计方案.....4
1.4 方案的优点与不足.....5
2 嵌入式部分.....6
2.1 整体方案.....6
2.2 运动学解算方法.....6
2.3 机械臂与控制方案.....7
2.4 功能模块说明.....7
2.5 难点与不足.....7
第 3 章 算法部分.....8
3.1 开发环境介绍.....8
3.2 整体技术方案概述.....10
3.3 算法整体框架设计.....10
3.4 算法功能模块说明.....10
3.5 测试结果.....11
3.6 可优化方案.....12
4 夏令营感想、总结.....13

1 机械部分

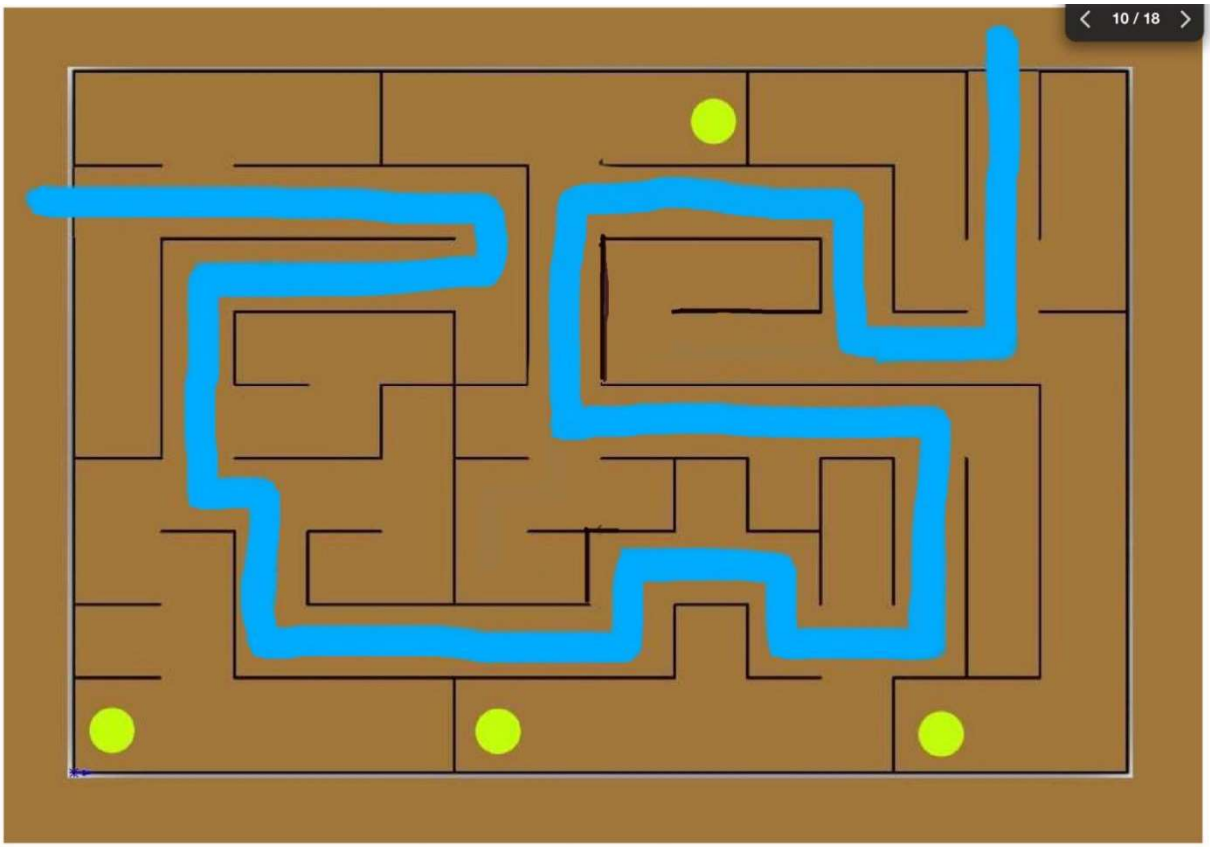
1.1 设计动机

比赛的具体内容是小车走出迷宫并找到拼好的魔方。即以最短时间走出迷宫，在备赛第一阶段中的主要目标为找到一条最优路径，其中路径中包含四个魔方的位置，目标为在更短
时间完成比赛。为了取得最短时间，我们需要能抓取出完全解好的魔方（减分魔方）。

能运用自己的所学，将知识转化为实际能力，在结营比赛中获得较好的名次。

夏令营比赛规则：

每支夏令营队伍需要自主研发一辆全自动的机器人战车，寻找魔方并穿越迷宫。 比赛场地全部使用木板搭建：



绿色标记是魔方是放置地点。

迷宫和迷宫当中的魔方放置地点不会有任何变化。4个魔方当中有一个魔方是复原的。其余魔方没有复原。实际场地中没有引导线，但是会有一些标志物来辅助视觉识别。

机器人任选一个入口出发，开始计时；必须从另一个口出去才算参完成比赛，计时结束。根据队伍的完成时间进行排名。

机器人在迷宫中必须夹取一个魔方，若没有夹取魔方或夹取魔方失败会在总时间上+5分钟，若夹取到复原的魔方会在总时间上减去5分钟。

比赛前，队伍需要抽签来确定复原魔方在4个位置的哪一个位置。抽签后，不允许再修改机器人的结构与程序。

1.2 设计需求

比赛的功能需求：小车走直线并能转弯；
小车可以测出距离前方障碍物的距离；
小车可以识别魔方的颜色；
小车可以抓取魔方；
小车可以走出迷宫；

各需求的优先级：走直线；拐弯；识别距离；抓取魔方；识别颜色；走出迷宫（优先级从高到低）

大概的设计时间：10天（7.28~8.6）

1.3 设计方案

详细的机械方案：

机械爪：采用舵机带动两个弧形连杆随着舵机的转动夹爪分开和靠近夹取魔方的核心结构，除此之外不乏新颖设计，比如在爪子内侧增加一些强有力的防滑材料，可以稳定的夹取魔方。爪子移动采用新颖的滑槽设计，快速夹取魔方，为了减少机械臂的压力提高稳定性机械爪的材料全部采用超轻设计。预计总重量不超过150g。

测距模块：采用四个超声测距模块。两个放在车声前方，用于检测车前障碍情况；两个放在车身两侧，用于检测两侧与墙壁的间距。

稳定性分析：

在车前放置了两个超声波模块，为车前测距的准确性提供了有效保障，有了较好的容错率。机械爪的设计有着较好的可靠性，平行滑轨保证了夹取时的较好的稳定性，魔方不容易滑落。车身经过改造，能有比较优良的防撞性能，可以避免因行驶时的撞击而受到损坏。

1.4 方案的优点与不足

优点：

1、 机械爪的两爪子之间与车身是一个平移关系，在夹合过程中不会改变夹子与魔方之间的相对位置，因此能更加方便的控制舵机夹取魔方，并且能有效的简化对魔方的距离的测算。

2、 前置双超声模块，性能良好且测距精确，有助于较好的实现识别魔方与测距等目标，为成功抓取魔方打下良好的基础。

3、 侧向超声模块，精确测量小车与墙壁之间的距离，配合定时器使用，能避免小车在直行过程中出现无法走出直线等情况。同时在其辅助下，能进行90度的直角转弯等操作。

缺点：

1.抓盘为方形，在夹子滑动的时候会有些卡，魔方的夹取不太稳定

2.小车打滑严重

2 嵌入式部分

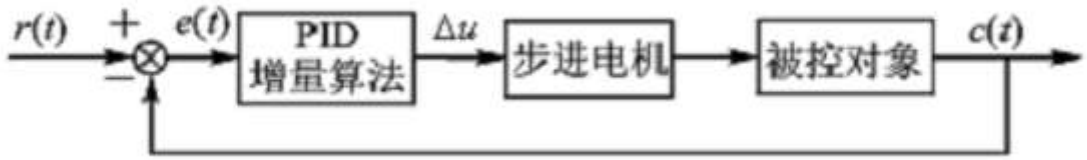
2.1 整体方案

将小车所需要走的路径一步一步的记忆到程序中，再通过电机，超声波等模块的不断配合，从而走出迷宫。程序所记录的路径主要由两个函数构成。首先进行的是最长路径的函数，通过这条路径可以找到迷宫中所有的魔方，而第二条路径是最短路径。在程序运行时，如果抓到魔方，最长路径函数会在合适的地点return，并且记录下当前的状态，知道是哪一个地方开始发生变化。然后运行最小路径函数，最小路径函数运用switch和状态使得最终可以走出迷宫。

1.运用增量型PID算法使电机运行稳定。

$$\Delta u(n) = u(n) - u(n-1)$$

$$= K_P[e(n) - e(n-1)] + K_P \frac{T}{T_I} e(n) + K_P \frac{T_D}{T}[e(n) - 2e(n-1) + e(n-2)]$$



在经过多次测试对比之后发现，由于技术和时间问题，PID算法在电机上的运用和不用PID差距不大甚至比不用更不稳定一点。所以最终去掉了PID.

2.陀螺仪设置和滤波

在开始设计的前三天对陀螺仪和滤波进行了学习和配置，为了配合小组进度以及我们对陀螺仪滤波的学习进度，最终我们否决了陀螺仪的方案。

3.路径规划

小车在迷宫中按照提前规划好的路线行进。

4.超声波测距模块

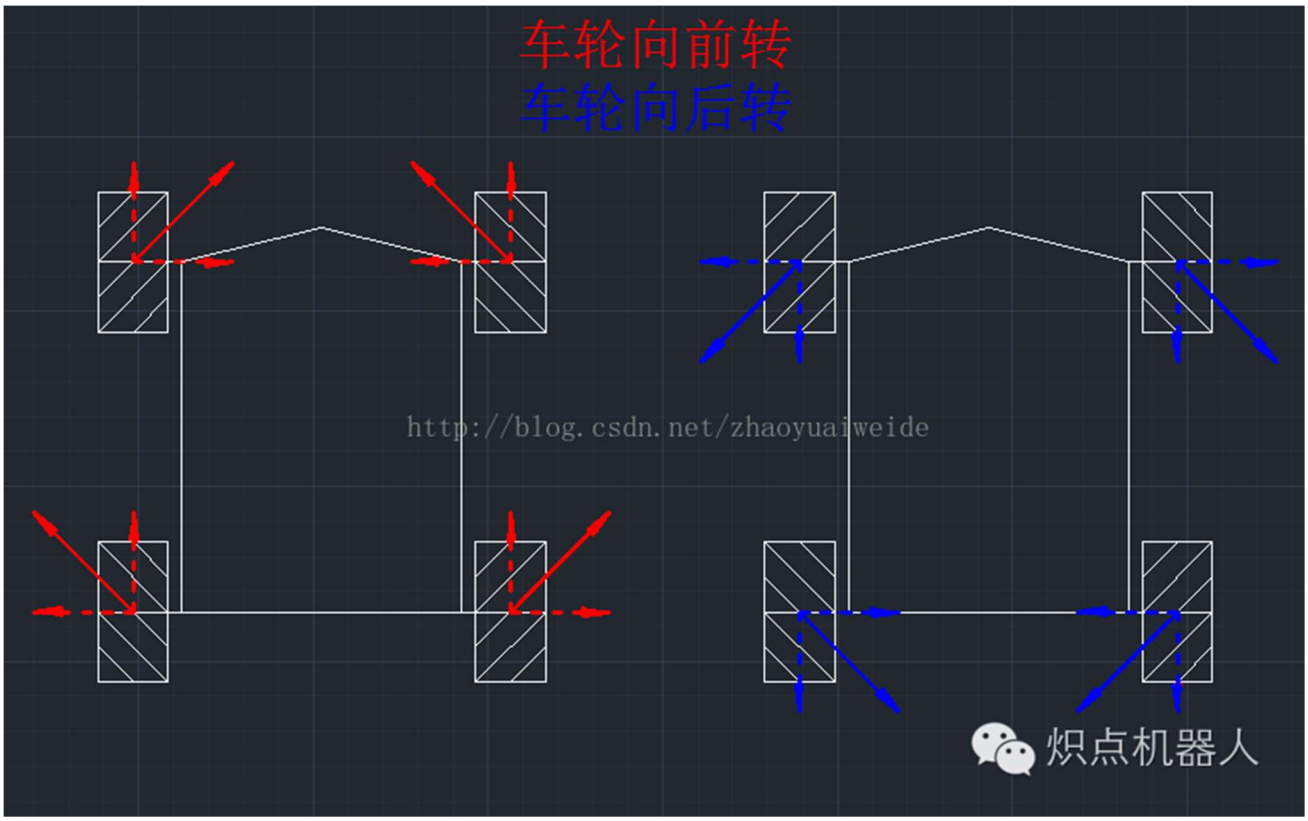
（1）小车两侧的的超声波实时检测两侧墙壁距离，超声波每60ms返回一次数据，修正小车左右电机的转速。小车两侧个设一个超声波传感器，由于点与直线的性质，所以我们读取了本次单侧超声波传感器的返回值以及上一次的返回值进行对比，如果本次返回值一侧小于上次返回值，一侧大于上一次返回值，则说明小车向返回值变小的一侧倾斜，此时判断成立，则适当增大倾斜那侧的电机占空比以达到修正小车方向的作用，这样及避免了使用两个超声波传感器的繁琐也完成了纠正小车的功能。

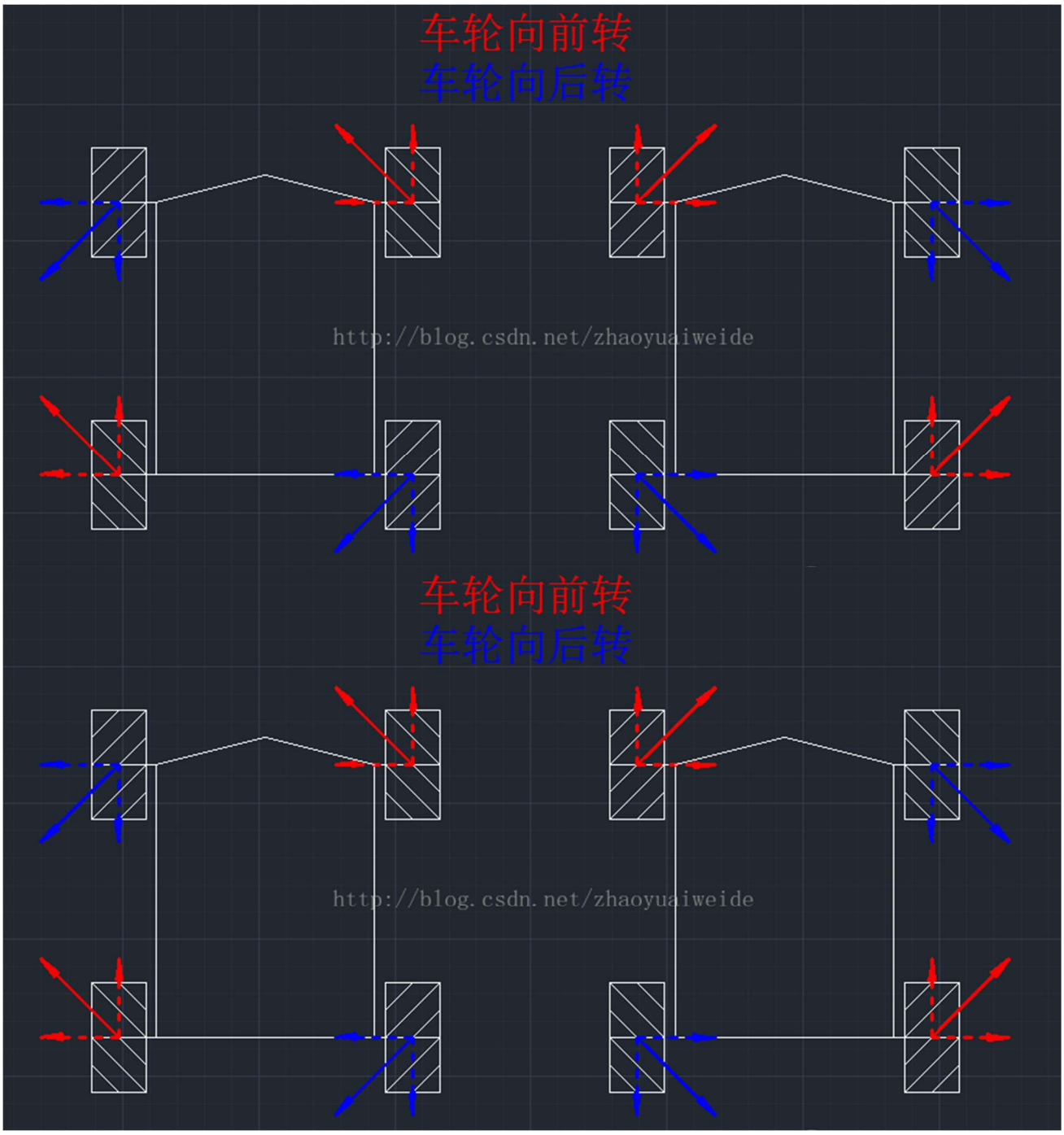
（2）在相机识别到魔方后，通过超声波测距检测魔方距离。将返回值和相机标定测距的值进行对比，误差在10厘米内，则前方测距值为模仿距离

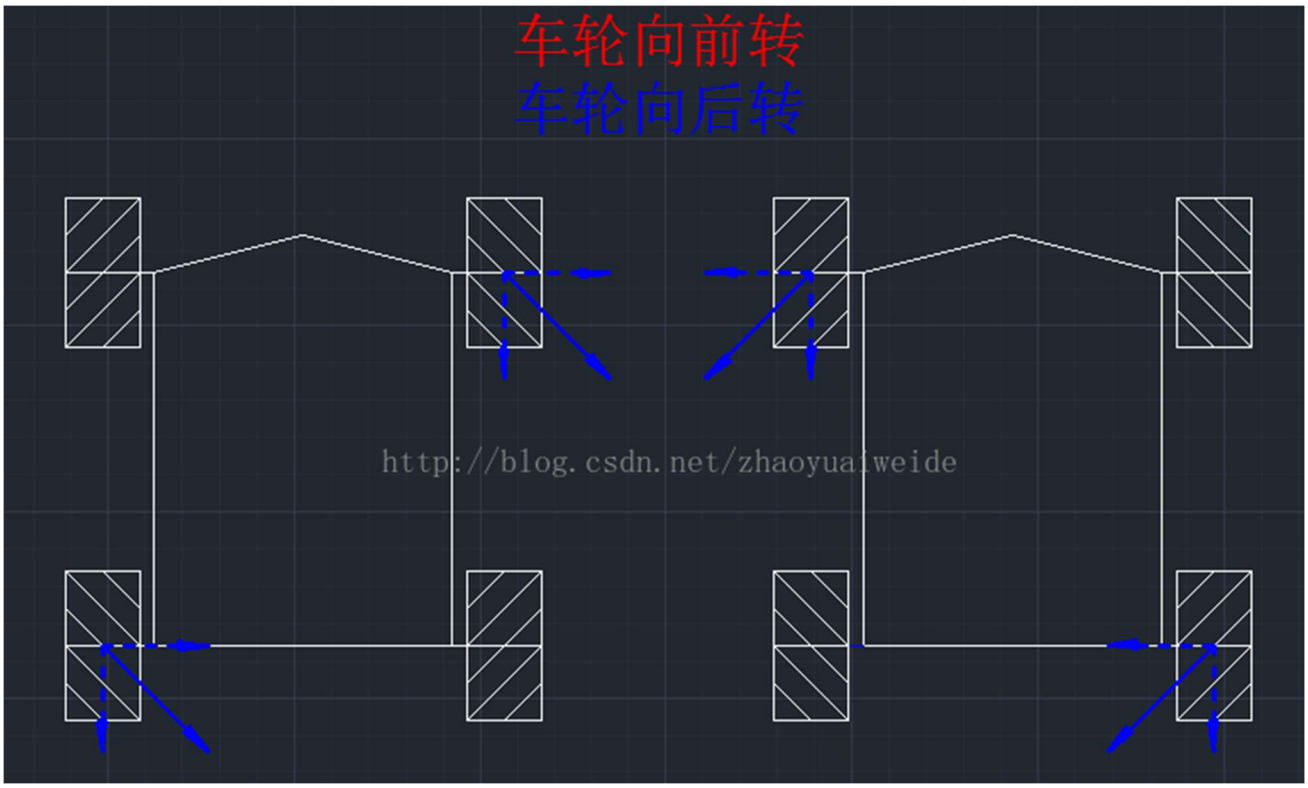
5.串口通信

树莓派驱动挂在小车正前方的单目相机识别到魔方后，发送两个状态量（0和1），单片机接收到状态量之后驱动电机向前行驶靠近魔方，并通过小车前方的超声波测距模块检测前方魔方的距离，并在魔方前10厘米左右停下来，驱动机械臂控制转动的舵机下降到和魔方水平，再驱动控制抓取的舵机抓取魔方，最后升起机械臂到小车上方。在抓取过程中失能串口通信，避免抓取过程中相机检测识别传回数据影响抓取。

2.2 运动学解算方法




车轮向前转
车轮向后转
http://blog.csdn.net/zhaoyuaiweide
车轮向前转
车轮向后转
http://blog.csdn.net/zhaoyuaiweide



1、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。底盘的运动也可以分解为三个量：

Vtx表示X轴运动的速度，即左右方向，定义向右为正；

Vty表示Y轴运动的速度，即前后方向，定义向前为正；

Ω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小车的几何中心相对于地面的速度。

2、r为从几何中心指向轮子轴心的矢量；

V为轮子轴心的运动速度矢量；

Vr为轮子轴心沿垂直于 r 的方向（即切线方向）的速度分量；

那么可以计算出：V=Vt+ωr

分别计算 X、Y 轴的分量为：

Vx=Vtx-ωry

Vy=Vty+ωrx

同理可以算出其他三个轮子轴心的速度。

3、根据轮子轴心的速度，可以分解出沿辊子方向的速度Vp和垂直于辊子方向的速度Vv。其中Vv是可以无视的，而Vp=V*U= $1/\sqrt{2}$Vx+$1/\sqrt{2}$Vy

其中U是沿辊子方向的单位矢量。

4、Vω=-Vx+Vy

根据图所示的 a 和 b 的定义有:

Vx=Vtz+ωb

Vy=Vty- ωa

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:

Vw1=Vty-Vtx+ω(a+b)

Vw2=Vty+Vtx-ω(a+b)

Vw3=Vty-Vtx-ω(a+b)

Vw4=Vty+Vtx+ω(a+b)

2.3 机械臂与控制方案

机械臂与底盘所使用的控制方法

如何解决PID的超调、震荡等问题

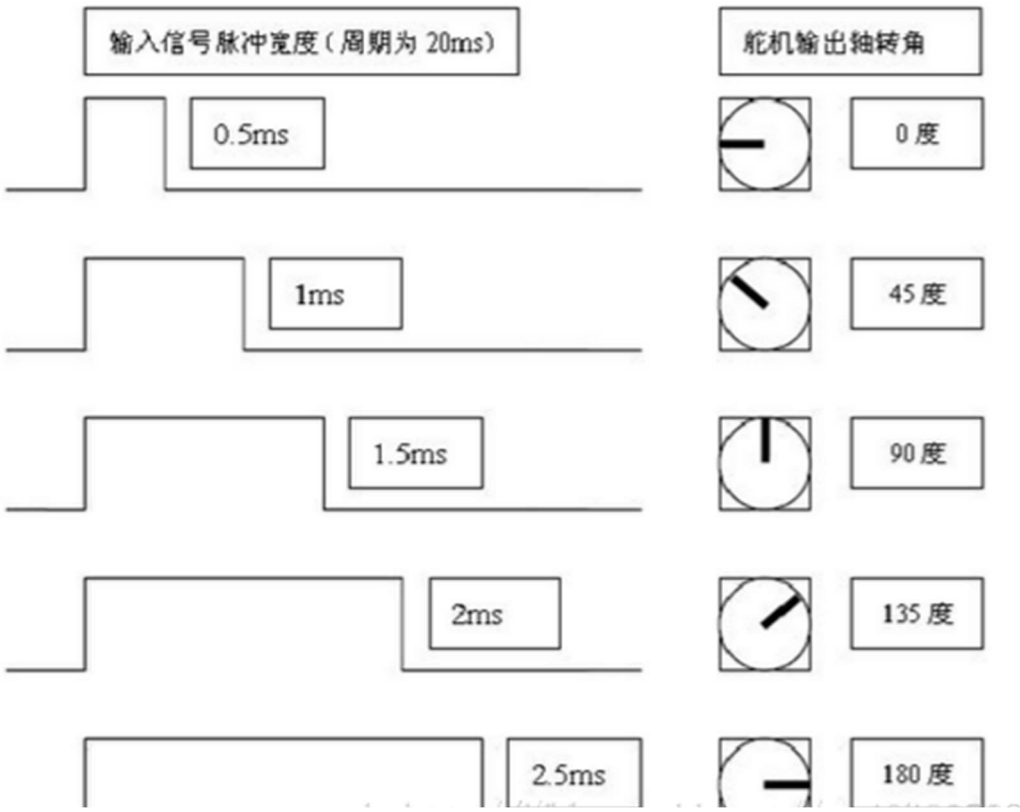
（1）舵机

舵机的主要组成部分为伺服电机，所谓伺服就是服从信号的要求而动作。在信号来之前，转子停止不动；信号来到之后，转子立即运动。因此我们就可以给舵机输入不同的信号,来控制其旋转到不同的角度。

舵机接收的是PWM信号，当信号进入内部电路产生一个偏置电压，触发电机通过减速齿轮带动电位器移动，使电压差为零时，电机停转，从而达到伺服的效果。简单来说就是给舵机一个特定的PWM信号，舵机就可以旋转到指定的位置。

舵机上有三根线，分别是GND、VCC和SIG，也就是地线、电源线和信号线，其中的PWM波就是从信号线输入给舵机的。

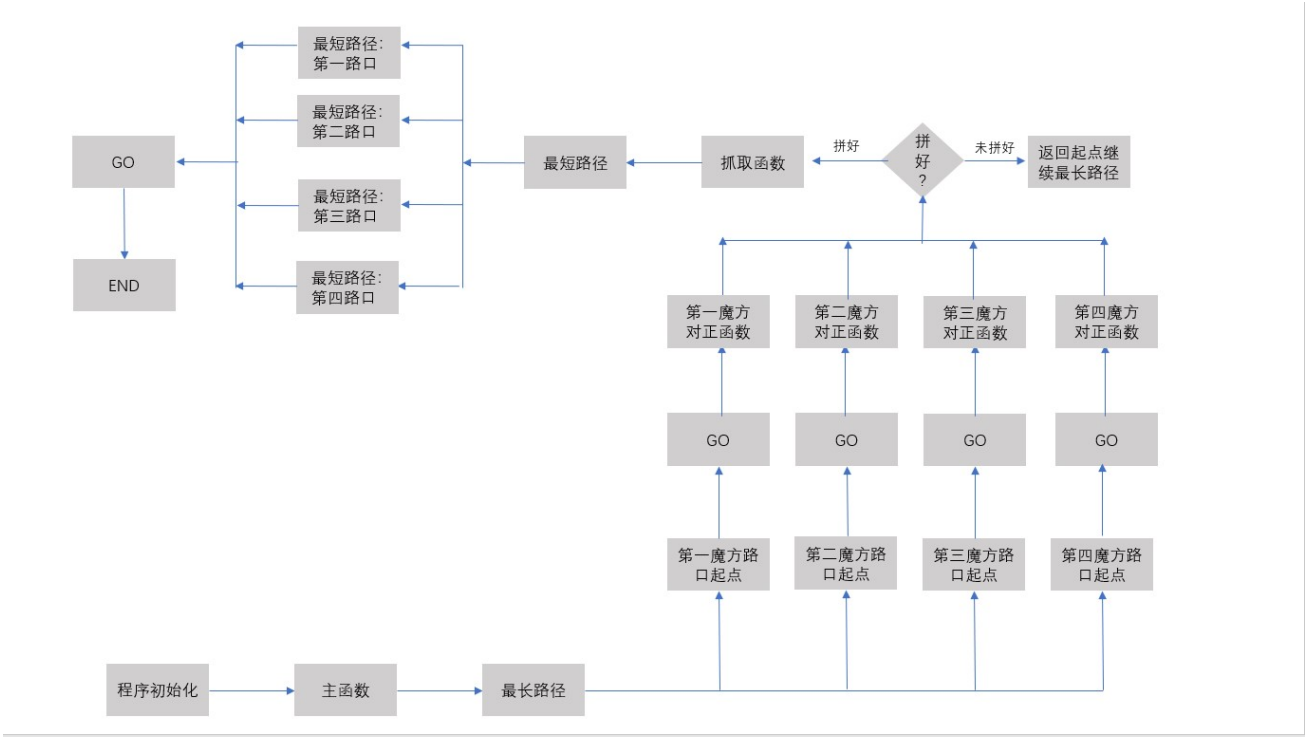
一般来说，舵机接收的PWM信号频率为50HZ，即周期为20ms。当高电平的脉宽在0.5ms-2.5ms之间时舵机就可以对应旋转到不同的角度。如下图。



此次夏令营所用主控板为stm32f429IGT6的芯片，控制舵机的定时器TIM10和TIM11，所在的总线APB1为调制为84Hz，预分频系数为839，ARR值为2000将PWM输出频率为50Hz，即周期为20ms，则所得到的角度脉宽换算公式为：

PULSE=（角度/9）*10+50

2.4 功能模块说明



主函数只跑路径函数。其余内容均在路径中完成。

Pwm.c：定义了所有PWM输出的函数，例如：四个电机的转动包括直走转弯函数，舵机运转函数等

Road.c：包含了整个路径规划和超声波测距。

Usart.c：串口通信函数。

直走转弯：固定不同状态下四个电机的占空比。

```
Go_ahead();                    //直走
Turn_right();                  //右转
Turn_left();                   //左转
Parallel_left();               //左平移
Parallel_right();              //右平移
```

舵机运转：steering_engine(); //舵机平稳运行任意角度

steering_engine_back(); // 运行任意角度后返回

测距：getdistence(); //获取测距值

超声测距模块：车前装两个超声测距模块，车身两侧各装一个超声测距模块。

单目相机模块：车前支架上放置一个单目相机，用于图像捕获和识别等功能。

2.5 难点与不足

设计的过程中遇到的问题：

(1)PID算法。

(2)陀螺仪。

(3)经常出现写出的代码功能得不到实现。

(4)分工合作的不同函数在后期整合过程中出现未知的BUG。

(5)单个程序运行正常，组合程序容易出现漏洞，代码可移植性低。

第3章 算法部分

3.1 开发环境介绍

3.1.1硬件环境

本次算法构建框架为摄像头与树莓派，摄像头选用800万像素，自动调焦，树莓派型号为3B+，系统为Raspbian。其中树莓派使用亚克力外壳，运用散热片及散热风扇进行散热，内存使用32G的TF卡，远程控制使用vnc可视化界面进行远程控制

3.1.2算法环境

本次算法环境为Ubuntu16.04+OpenCV3.4.0+Raspbian，其中树莓派环境也为OpenCV3.4.0。在虚拟机配置OpenCV3.4.0环境时， 关键在于在编译前下载依赖包，编译时进入源码目录，具体安装步骤如下：

1.安装依赖包

```
sudo apt-get install cmake
    sudo apt-get install build-essential libgtk2.0-dev libavcodec-dev
    libavformat-dev libjpeg.dev libtiff4.dev libswscale-dev libjasper-dev
```

2.编译 opencv

解压之前下载好的源码包 `tar zcvf Opencv 压缩包名称.tar.gz 想要解压的路径`

进入 OpenCV 源码目录（总有 zz 在根目录进行下面的操作，在这里强调一下，动动脑子！）

```
    mkdir build
    cd build
    cmake -D CMAKE_BUILD_TYPE=Release -D CMAKE_INSTALL_PREFIX=/usr/local ..
    sudo make
    sudo make install
```

所涉及的命令具体含义自行学习 **CMake** 的基本操作

3.添加路径

配置一些 OpenCV 的编译环境首先将 OpenCV 的库添加到路径，从而可以让系统找到

```
sudo gedit /etc/ld.so.conf.d/opencv.conf
```

执行此命令后打开的可能是一个空白的文件，不用管，只需要在文件末尾添加

```
/usr/local/lib
```

执行如下命令使得刚才的配置路径生效

```
sudo ldconfig
```

配置 bash

```
sudo gedit /etc/bash.bashrc  1
```

在最末尾添加

```
PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig
export PKG_CONFIG_PATH  12
```

保存，执行如下命令使得配置生效

```
source /etc/bash.bashrc  1
```

更新

```
sudo updatedb
```

4.测试

至此所有的配置都已经完成 下面用一个小程序测试一下

cd 到 opencv-3.4.0/smaples/cpp/example_cmake 目录下

我们可以看到这个目录里官方已经给出了一个 cmake 的 example 我们可以拿来测试下 按顺序执行

```
cmake .
make
./opencv_example123
```

即可看到打开了摄像头，在左上角有一个 hello opencv 即表示配置成功

3.2 整体技术方案概述

识别魔方阶段，考虑到魔方一个面有九个小矩形的特征，我们以此为切入点。首先识别是否存在九个小矩形，若存在即为魔方，比较九个小矩形颜色均值是否相同，若相同即为已拼好，并将信息传给主控板。识别到魔方后运用相机标定对魔方进行定位，控制小车抓取已拼好的魔方。

3.3 算法整体框架设计

我们的小车要求能够自主寻路，并且寻找一个已经完全拼好的魔方，将其一同带出迷宫，小车要想完成这些任务，寻路，识别以及位置确定是必不可少的；

对于这次比赛，我们组的整体策略如下：小车自动寻路，寻路的同时不停的判断小车前方是否存在魔方，找到魔方之后对魔方进行进一步分析，辨别魔方是否已经完全拼好，检测到魔方已经完全拼好之后对魔方的位置进行确定并且抓取魔方，最后走出迷宫，完成比赛。

3.4 算法功能模块说明

3.3.1寻找魔方算法

魔方识别算法的核心是通过对摄像机拍摄到的每一帧图片进行处理分析，判断小车前方是否存在魔方，处理过程如下：第一步是进行对图片的预处理，主要是对图像进行滤波操作，这一步可以在保留图像细节特征的前提下对图像的噪声进行抑制，其处理效果的好坏将直接影响到后续图像操作处理和分析的有效性和可靠性；第二步则是图像轮廓的查找，先要对图像进行阈值化处理以得到图像的灰度图，之后查找图像轮廓并将查找结果保存；第三步则是对魔方的寻找，主要是利用魔方一个面有九个小块的特点，如果在保存的图像轮廓中找到九个符合要求的矩形轮廓，即为寻找到魔方。

3.3.2 颜色识别算法

在找到魔方后，必须要对魔方进行颜色上的分析以判断魔方是否完全拼好，过程如下：第一步先确定魔方的九个ROI区域并对其进行预处理，排除噪声；第二步是对确定的九个ROI区域分别进行颜色识别，由于是在RGB空间进行处理，只要求出每个ROI区域的色素均值，再用均值的三个分量Red，Green，Blue与六种魔方颜色的范围进行对比，求出九个ROI区域的颜色，如果九个ROI区域的颜色范围一致，则说明 魔方已经完全拼好。

3.3.3 单目相机姿态解算

为了求出魔方和小车之间的距离，必须进行单目相机三维姿态解算。首先通过相机标定得出相机的内参数，标定方法是棋盘标定法；其次是利用已知物体在三维空间的坐标，在图像上对应的图像像素坐标以及相机内参数进而求解出此时相机相对于空间已知物体的外参数，即旋转向量以及平移向量；最后对旋转向量进行数据分析处理，求解出此时相机相对于已知物体空间坐标的三维姿态欧式角，即俯仰角，偏航角，滚轮角。

3.3.4 小车逻辑

在整个比赛中，小车共有三个状态：寻路模式，识别模式，抓取模式，比赛开始之后，小车进入寻路模式，在寻路的同时，小车也对摄像机传回的每一帧图像进行处理，判断是否存在魔方，找到魔方后，小车进入识别模式，对魔方的颜色进行识别，判断魔方是否完全拼好，如果魔方已经完全拼好，则进入抓取模式，对魔方进行抓取，抓取成功后重新回到寻路模式，随后小车模式不再改变，直至走出迷宫，完成比赛。

3.5 测试结果

小车对魔方颜色的辨别还存在一些问题，魔方的红色小块相比其他颜色要花费更长的时间去识别，我们将红色在RGB颜色空间下的分量范围扩大，以此来增加小车对红色的识别度，再次进

行测试，小车对红色的识别速度加快了许多。

3.6 可优化方案

1、机器人的适应性不足，只能适用于这一种迷宫结构。可以引入俯拍的计算机视觉，使得小车能在不同的迷宫地形下都能顺利找着出口。
2、小车的打滑问题比较严重，可以考虑使用更低的底盘，并加重小车重量。
3、在识别魔方颜色时，可以将摄像头拍摄到的图片由RGB颜色空间转换到HSV颜色空间进行分析。不同于RGB，HSV颜色空间由色相，饱和度和亮度三个分量组成，通过HSV颜色空间进行分析时，可以极大的排除光照不足等因素对颜色识别效果的影响，不过本次比赛中并不需要担心这些，所以我们还是决定在RGB空间进行分析。

4 夏令营感想、总结

很开心能参加这一次的机器人夏令营，在其他大部分同学都在各地旅行或是宅在家里追电视剧的时候，我留在学校与一群志同道合的小伙伴们相识相知，一起度过了这三周的学习与奋斗的美好经历。我很开心，我的暑假很有意义。在这里，我收获了友谊，收获了知识，也增长了才干，为更好的成为三有西电人做好准备。

在夏令营的这三个星期中，我学会了很多。我明白了团队协作的重要性。从第一周的学习与熟悉各种软件操作与基础，到后面两周的正式开展智能小车的设计实践，这个过程中我渐渐与团队成员互相信任与建立起默契。我在这个过程中体会到了一个团队为了一个共同的目标一起前进、一起奋斗的快乐，这个过程中虽然会有摩擦与争吵，但是更多的还是欢声笑语，是一致向前，是全组成员一起为了一个共同的目标而不懈努力。

这三周中，我看到了团队中每一个成员的努力与付出，有的组员在每天晚上九点结束回到宿舍后还会在群里继续讨论所遇到的问题与共同寻找解决方法。更有人为了一行代码、一个小小的零件而彻夜研究，只为了能赶上预定的工期，以保证整组能在夏令营结束前交出一个更好的智能小车，并在最后的结营比赛中取得更加优异的成绩，为自己的夏令营交出一个满意的答卷。