西电机器人夏令营

SUMMER CAMP 2018

六组技术报告

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			
机械：确定机械方案、验证可行性、方案设计以及加工	2天	7.30	7.31
嵌入式：确定与挑选购买所需的各种元器件，封装代码	5天	7.30	8.03
算法：学习OpenCV及进行初步的算法编写	5天	7.30	8.03
3、方案的讨论、迭代			
机械：制造原型机，分析并改进机构、进行结构优化	6天	8.01	8.06
嵌入式：调试各种元器件，写出小车的走迷宫路径	3天	8.04	8.06
·算法：搭建树莓派环境，完成通信调试与视觉算法编写	3天	8.04	8.06
4、测试与调试			
各项目组间尝试对接联调	1天	8.07	8.07
5、总装完成、优化			
实地测试与稳定性优化	2天	8.08	8.09

目录

1 机械部分

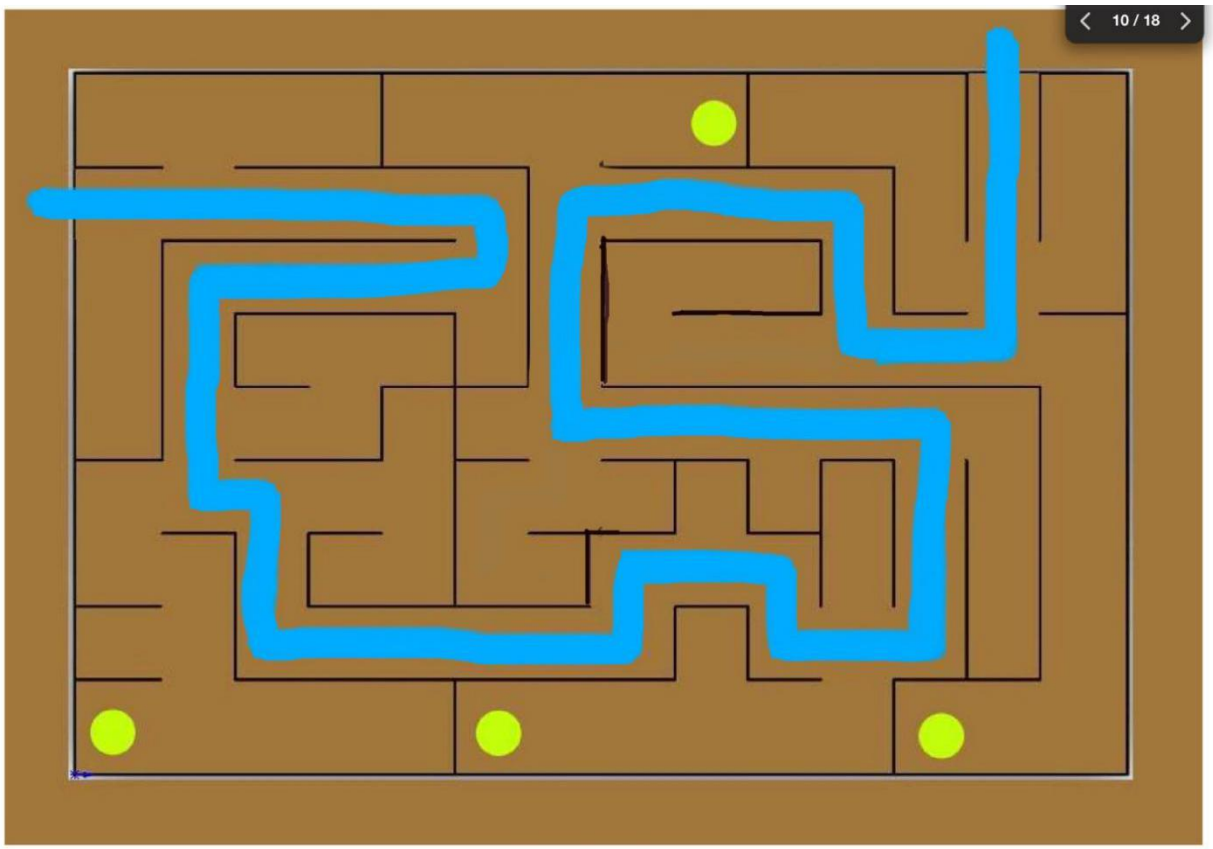
1.1 设计动机

设计动机：

在机器人夏令营中学习到了关于机械、嵌入式与算法的相关知识，希望能运用自己的所学，将知识转化为实际能力，并且在实践中熟悉操作，最终能在结营比赛中获得较好的名次，为三周的机器人夏令营做出一个完美的结束。

夏令营结营比赛规则：

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；识别距离；抓取魔方；识别颜色；走出迷宫（优先级从高到低）

目标设计时间：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橡胶防滑垫，增大摩擦力，使机械臂可以稳定的夹取魔方。爪子的开合采用滑槽设计，快速夹取魔方。

机械臂：采用二自由度机械臂，一个耐烧舵机控制机械爪的开合，另一个大扭矩舵机控制机械臂的抬升。当小车靠近目标魔方时，激光测距模块与单目相机配合，使小车车前对准魔方，此时收于小车上方的机械臂在大扭矩舵机的控制下落下，耐烧舵机控制机械爪合拢，抓住魔方，再在大扭矩舵机的作用下机械臂旋转收回车上，以减小车长，避免车刮蹭到迷宫墙壁。

测距模块：采用五个激光测距模块。一个放在车声前方，用于检测车前障碍情况；车身两侧各安放四个，用于检测小车两侧与墙壁的间距，探测路径。

机器人底板：对底板重新进行设计，借鉴了实际生活中跑车底盘的设计思路，使主控板、电池等部件可以更加合理与布局科学地安装在底板上，同时能更好的适应机械手的活动。通过加宽底板宽度、降低底板高度来提高机器人的行驶稳定性。在底板上尽可能多的留出空余位置，为后期对机器人进行改装优化和元件安装打孔提供了更大的可变空间。

稳定性分析：

1、经过多次设计迭代，我们的机械爪能保证较为稳定的抓取成功率。同时较大的爪间距，也为抓取留下了非常可观的容错空间。因此能有较为好的稳定性。

2、超大扭矩的高品质品牌舵机，能为抓取及回收的过程提供极好的动力保障。22kg扭矩的机械

爪耐烧防堵转品牌舵机，既能保证抓取魔方时较大的抓力，又能防止舵机因为长时间工作或者堵转而烧坏。17kg超大扭矩高精度双轴数字舵机，能为机械爪的回收提供强有力的动力保证，有较好的稳定性。

1.4 方案的优点与不足

优点：

1、 机械爪的两爪子在开合过程中是平移的关系，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、 前置激光测距模块，性能良好且测距精确，有助于较好的实现判断小车与魔方间距与行进过程中的测距等目标，为成功抓取魔方与走出迷宫打下良好的基础。

3、 侧向激光测距模块，精确测量小车与墙壁之间的距离，配合陀螺仪模块，能有效保证小车在直行过程中较小的偏移量。同时在陀螺仪的辅助下，能进行90度的直角转弯等操作。

缺点：

1、 机械爪的开合过程中有一定几率会出现爪子与滑槽摩擦力过大无法收紧的问题。

2、 机械爪的耐烧舵机扭矩较大，可能出现夹持力量太大而3D打印的爪子和固定件较脆弱出现损坏的问题。

3、 机械臂在翻上小车的过程中，有可能会出现魔方甩飞的问题。

4、 3D打印的材料精度与硬度不足，有部分连接处会有连接松动的情况。

2 嵌入式部分

2.1 整体方案

我们将路径分为了主路径与四条分路径。主路径为小车不夹取魔方直接从入口走到出口的路径，其中主路径上通向各个魔方的分岔口我们设置了四个中断点，分路径为各个中断点到魔方及返回中断点的路径。

2.2 运动学解算方法

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+ω r
分别计算 X、Y 轴的分量为：
Vx=Vtx-ω ry
Vy=Vty+ω rx
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/Vx+1/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+ω b
Vy=Vty- ω a
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、【陀螺仪】对于左转和右转，使用陀螺仪校正旋转角度。我们组使用MPU6500传感器，内置陀螺仪和加速度计功能，使用SPI通信协议读取MPU寄存器数据，得到陀螺仪旋转角速度和加速度计的三个加速度。再使用姿态解算算法将数据转化为姿态。

2.4 功能模块说明

超声波测距模块：车前及车身左右各安置一个超声波测距模块，用于检测小车的位置。
单目相机模块：车前支架上放置一个单目相机，用于图像识别等功能。

2.5 难点与不足

1、激光测距模块在stm32上的测距的实现。我们在夏令营期间只完成了激光测距在Arduino上的测距反馈，但由于通信问题较难解决，所以无法使用，而使用stm32芯片的激光测距算法虽然经过了三天的努力，但是最后依然没有成功，最后只能被迫使用了超声波测距模块。
2、陀螺仪存在零点漂移，校准有一定难度。

第3章 算法部分

3.1 开发环境介绍

串口

树莓派有个叫做 wiringPi 的库，是专门控制树莓派的 GPIO 的，那里面有串口的 API 安装过程

```
sudo apt-get update

sudo apt-get upgrade

sudo apt-get install git-core

git clone git://git.drogon.net/wiringPi

cd wiringPi

./build

```

Apriltags

编译安装

```
git clone https://github.com/swatbotics/apriltags-cpp.git

cd apriltags-cpp

mkdir build

cd build

cmake .. -DCMAKE_BUILD_TYPE=Release

make

```

安装好后，你编译的时候链接上它的静态库就行，或者写个 makefile，但是这个库有老算法和新算法，老算法亲测鲁棒性比较差，新算法没试过，如果你要用新算法的话，还得安 装 CGAL 并在编译的时候链接上它的库才行

树莓派下编译安装OpenCV3.4.1

1. 先把软件源换为清华的源
    1. 使用管理员权限，编辑/etc/apt/sources.list文件

```
sudo nano /etc/apt/sources.list
```

用#注释掉原文件的内容，用以下内容取代

```
deb http://mirrors.tuna.tsinghua.edu.cn/raspbian/raspbian/ stretch main contrib non-free rpi
deb-src http://mirrors.tuna.tsinghua.edu.cn/raspbian/raspbian/ stretch main contrib non-free rpi
```

    2. 使用管理员权限，编辑/etc/apt/sources.list.d/raspi.list

```
sudo nano /etc/apt/sources.list.d/raspi.list
```

用#注释掉原文件的内容，用以下内容取代

```
deb http://mirror.tuna.tsinghua.edu.cn/raspberrypi/ stretch main ui
deb-src http://mirror.tuna.tsinghua.edu.cn/raspberrypi/ stretch main ui
```

    3. 更新软件源列表

```
sudo apt-get update
```

2. 为了充分使用整个存储空间，输入

```
sudo raspi-config
```

进入到软件配置工具后，选择第七项 "Advanced Options" ，再选择 A1 项 "Expand Filesystem" 即可

3. 先更新一下
    1. 软件源更新

```
sudo apt-get update
```

    2. 升级本地所有安装包

```
sudo apt-get upgrade
```

    3. 升级树莓派固件

```
sudo rpi-update
```

4. 安装构建OpenCV的相关工具

```
sudo apt-get install build-essential cmake git pkg-config
```

5. 安装常用图像工具包

```
//安装jpeg格式图像工具包
sudo apt-get install libjpeg8-dev
//安装tif格式图像工具包
sudo apt-get install libtiff5-dev
//安装JPEG-2000格式图像工具包
sudo apt-get install libjasper-dev
//安装png格式图像工具包
sudo apt-get install libpng12-dev
```

6. 安装视频I/O包（注意最后一个包的数字“4”后面是“L”）

```
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
```

7. 安装gtk2.0

```
sudo apt-get install libgtk2.0-dev
```

8. 安装优化函数包

```
sudo apt-get install libatlas-base-dev gfortran
```

9. 下载OpenCV源码（放在家目录下）

```
wget -O opencv-3.4.1.zip https://github.com/Itseez/opencv/archive/3.4.1.zip
unzip opencv-3.4.1.zip
wget -O opencv_contrib-3.4.1.zip
https://github.com/Itseez/opencv_contrib/archive/3.4.1.zip
unzip opencv_contrib-3.4.1.zip
```

10. 进入源码文件夹，新建一个名为 release 的文件夹用来存放 cmake 编译时产生的临时文件

```
cd opencv-3.4.1
mkdir release
cd release
```

11. 设置 cmake 编译参数，安装目录默认为 /usr/local , 注意参数名，等号和参数值之间不能有空格，但每行末尾“\” 之前有空格，参数值最后是两个英文的。

```
sudo cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.4.1/modules \
-D INSTALL_PYTHON_EXAMPLES=ON \
-D BUILD_EXAMPLES=ON ..
```

12. 开始编译

```
//编译
sudo make
//安装
sudo make install
//更新动态链接库
sudo ldconfig
```

3.2 整体技术方案概述

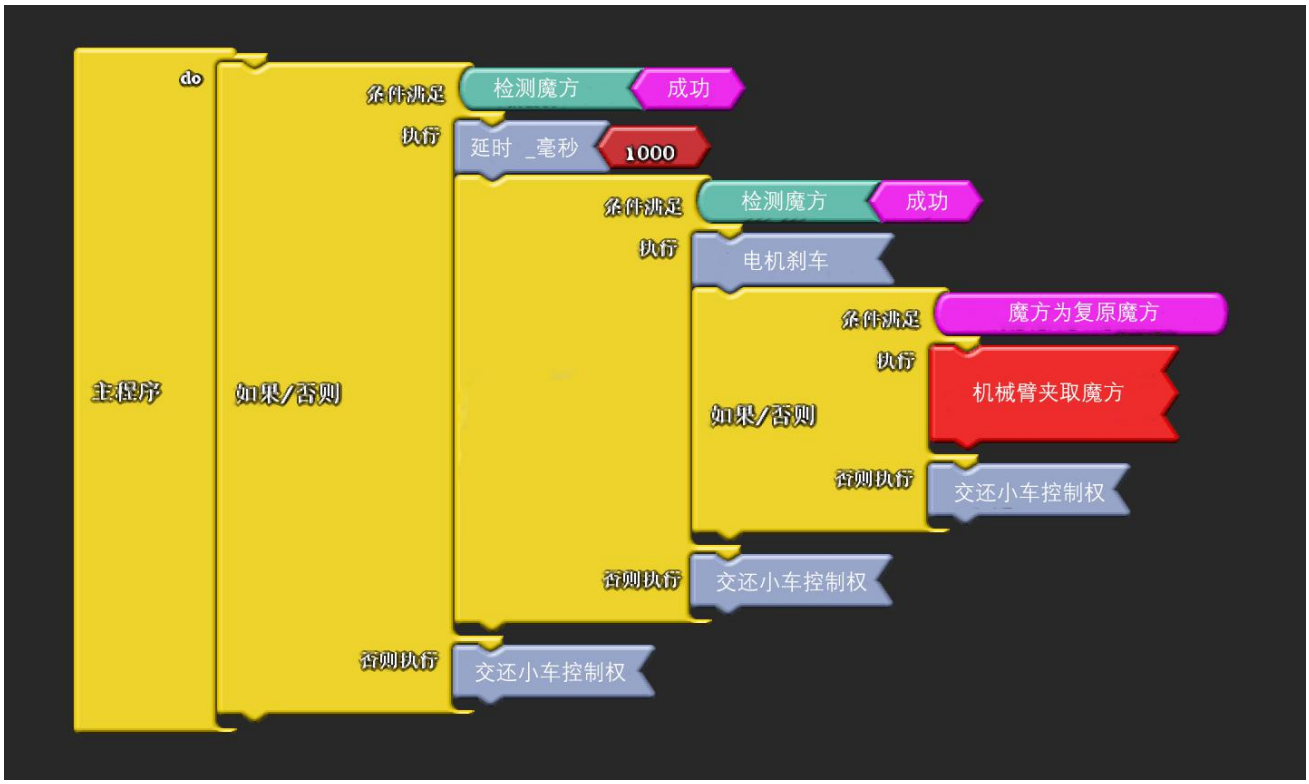
魔方识别：

先对捕获的图像帧进行降噪处理，再从图像中提取轮廓，并对提取到的轮廓进行初步筛选和对初步筛选的结果进行优化，在图像帧上画出识别到的魔方的九个块。然后将获得的图像帧转换为HSV通道，再在识别到的各个魔方小矩形中取一个适当大小的ROI，计算各个ROI区域的颜色均值，以此均值代表对应魔方小块的颜色，对得到的 9 个魔方小块颜色数据求方差，设立一个阈值，当求得的方差小于该阈值时，就可以认为魔方已拼好。确定好已经复原的魔方后，使用相机对魔方获得的数据进行 PnP 解算，求得旋转向量和平移向量。

小车控制：

先使用摄像头对魔方进行捕获，如果捕获到魔方，先延时一秒再进行捕获，如果依然能捕获到魔方，就获取小车控制权，让小车刹车。然后对魔方进行颜色识别，若魔方拼好，则根据获得的数据对小车进行位置调整，然后夹魔方，若魔方没拼好，则不夹魔方。最后交还小车控制权

3.3 算法整体框架设计



3.4 算法功能模块说明

魔方识别：

1、先对捕获的图像帧进行降噪处理

2、然后再从图像中提取轮廓

3、对提取到的轮廓进行初步筛选

a. 对提取到的轮廓画出最小矩形

b. 根据最小矩形的宽高比，以及最小矩形的边长进行筛选

4、对初步筛选的结果进行优化，进一步筛选

a. 对初步筛选的结果，如果提取的最小矩形数目过多，或过少，则丢弃该结果

b. 创建二维数组，存储筛选到的矩形之间的距离

e.g

$$\begin{matrix} 0 & 31 & 42 & \dots & 0 \\ 31 & 0 & 26 & \dots & 0 \\ 42 & 26 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{matrix}$$

【其中元素（0, 1）就表示矩形 0 到矩形 1 的距离， 元素（2, 1）就表示矩形 2 到矩形 1 的距离，以此类推，很明显这是一个对称矩阵】

c. 将第 i 行的数据加在一起，即为第 i 个矩形到其它各个矩形的距离之和，然后比较每一个矩形到其它各个矩形的距离之和，最小的那个大概率为魔方的中心矩形

d. 找到魔方的中心矩形 id （比如说是第 m 行）后，将第 m 行的各个数据进行排序，取前 9 个，这九个矩形大概率为识别到的魔方的 9 个块

5、在图像帧上画出识别到的魔方的九个块

颜色识别：

1、先将获得的图像帧转换为HSV通道

2、再在识别到的各个魔方小矩形中取一个适当大小的ROI，计算各个ROI区域的颜色均值，以此均值代表对应魔方小块的颜色

3、对得到的 9 个魔方小块颜色数据求方差，设立一个阈值，当求得的方差小于该阈值时，就可以认为魔方已拼好

单目测距：

1、先对相机进行标定，获得相机的内参矩阵以及径向和切向畸变系数

2、测量魔方尺寸

3、对获得的数据进行 PnP 解算，求得旋转向量和平移向量

小车控制：

1、先对魔方进行捕获，如果捕获到魔方，先延时一秒再进行捕获，如果依然能捕获到魔方，就获取小车控制权，让小车刹车

2、对魔方进行颜色识别

3、 若魔方拼好，则根据获得的数据对小车进行位置调整，然后夹魔方

4、若魔方没拼好，则不夹魔方

5、 交还小车控制权

3.5 测试结果

第一步我们测试的是识别。在电脑上测试基本没什么问题之后，将摄像头装在小车上，效果就没那么如意了。首先，可能由于小车在行驶的过程中速度太快导致每一帧的画面来不及处理就变换到下一帧，此时我们采取摄像头正对魔方，小车慢速平稳的策略去改进；其次，可能由于光线的原因，会出现小车在左方识别的很好，却在右方一点都识别不出来的情况，这个问题我们采取调节摄像头的亮度值去尽可能的逼近，此值通过不断实践得到，不过这种方案解决力度不太明显。

第二步，就是识别魔方后的夹取问题，原理是通过对相机标定进行测距，测的是魔方离相机的距离，然后通过串口通信发出一系列指令，使得小车能正确夹取。不过实际情况中测试很不稳定，夹取的总是会偏一点或者直接越过魔方，目前在改一些代码细节和一些参数的设置。

在实践中有的优化很可观，有的优化却不那么尽人意，需要时时变通，该舍弃一些就要舍弃，去寻找真正的优解。

3.6 可优化方案

1、测距模块原本想使用激光测距模块，但是由于激光测距的代码较难编写，夏令营期间时间较短，最后被迫使用了超声波测距模块。

2、走迷宫的路径优化不足，对碰撞等偶然因素较难把握，一旦发生碰撞，就可能会较难判断自身所在的位置，从而可能无法继续走出迷宫。希望能在较高处增加一个摄像头，实时传输迷宫中小车所在位置，以增强小车走迷宫时的稳定性。

3、走迷宫的适配性较差，只能走出比赛所设计的迷宫，遇到其他的迷宫将无法正常运行。可以通过在计算机视觉生成路径平面图的基础上，建立环境模型，并且使用蚁群算法，动态规划路径。

4 夏令营感想、总结

一、 感想：

作为一个什么都不会的项管，一开始很茫然不知道去做什么，每天跟着一起去学习，虽然什么都不懂，但是身边有一群可爱的组员一直在帮助我，我感觉很幸运能够认识他们，与他们一起度过这个暑假，让这个燥热的暑假不再枯燥。在这个过程中，我学到了很多，在我们做机器战车中，也有过分歧，也有的人曾沮丧过，但我们都没有退缩，坚持一起做机器车。在最后的调试中，终于，我们的小车成功的从迷宫中跑出，每一个人脸上都洋溢着笑容，不管最后结果如何，相信此次的经历将使我一生难忘。

——崔鑫语

这短短的三周时间，我很开心能在这里遇见这么多志趣相投的伙伴。虽然晚回家了三周，但是我觉得这非常的值得，在其他大部分同学都还宅在家里打游戏或者各种出去玩的时候，我留在学校与志同道合的小伙伴们相识相知，一起度过了这段美好的学习与奋斗时光。我很高兴，我的暑假没有白白浪费。

我在夏令营中不仅仅学会了基本的嵌入式与机械等等相关的知识与能力，还明白了如何与一个团队协作，共同努力为一个目标而战。从最初的生疏，到后面的默契与相互信任，我体会到了一种无法言说的快乐。感谢学校与各位 Staff 们的努力，让我们能在这个暑假相聚，也希望我们第六组能在最后的结营比赛中交出一份完美的答卷。

——王立恒

很幸运入选这个夏令营，很幸运组员都很好相处。从一个连开发板都没碰过的小白，到现在对单片机有一点点了解的小白，我觉得没有在学校浪费这三周时间，认识了有趣的人，又学到了知识。

第一次团队合作，跟一群有着共同目标的人一起奋斗一起学习的感觉很爽很开心。夏令营之前我想，在这三周我能学到什么，可渐渐的，我的想法变成了我能为这个团队做出什么。这三周才是我上大学之前想象中大学生活的样子。

——张梦

美好的暑假时光已过去了大半，你或许会躺在沙发上追着剧，或许会去五湖四海游玩，或许约着好友三三两两去谈天说地，可这些通通都是别人的假期，而我们的假期，注定与众不同。

在这个假期，很多朋友参加了历时 20 天的西电机器人夏令营，不论你是怀着怎样不同的初衷，我们都相聚在这里，结识了来自不同专业的小伙伴，踏上了独属我们机器人六组的旅程。

由机械，嵌入，算法构成一个小团队，很高兴通过了面试进入到了算法组。那么，就先来说说我们算法组的学习进程吧。

前两天基本上被笼罩在安装软件和配环境的噩梦下，照着教程来总会出一些小问题，然后就去 CSDN 和浏览器上各种查资料，总算配置成功了。毫不夸张的说，我的桌面瞬间就多出一列的图标。有安装 VM，在 VM 里装 Ubuntu，opencv 配环境（永久配置就很折磨人），装 git（其实到现在都不会），还有之前装过的 VS 和 Matlab，还有 AMCap 和 Typora……，还下了一个现在都没用过的 VScode，还因为好玩也装了其他组的软件，虽然有的软件的用法还没有精通，不过也是积累了很多下载破解软件的经验，这些软件在以后的学习中可能也会慢慢用到。

安装和配置成功了之后，就进入了opencv的学习中了，因为我算是初学者，之前多opencv没有接触过，所有就跟着那本教程去学，从简单的学起，从基本的显示图像到可以对图像进行一些基本操作，再到后来可以进行一些稍微复杂的工程，进步真的是慢慢积累的。正如 C 语言老师所说“没有敲过一万行代码，就不是一个程序猿”，虽然有点逗趣的成分在，不过也反应出一些经验比如找快速找 bug 和优化代码是非敲代码不能形成的，也是每一位程序猿的必经之路。

到了硬技术识别魔方核心算法的时候，经历了各种查资料和看博客，最后大佬决定用色差范围去决定，具体技术就不赘述了，在之前的核心技术报告里有写。经过了一番修改和调试，识别基本没有问题。后面开始给树莓派配置环境和装系统，还有写串口通信，每一次新任务就是一次学习的过程。

后来最后两天就是去场地真实实践，来设置一些参数的具体值，中间也对代码做了一些小改动，之后测试就很顺利。

总的来说，每一次对未知领域的挑战无疑也是一次学习的过程，看到自己一点小小的进步就觉得很满足，很幸运有这次宝贵的经历，很幸运结识一群小伙伴，谢谢你们的陪伴与努力。

——算法组（沈京龙、侯雪静）

二、 总结

在刚经历完期末考试，我们来到了机器人夏令营，组成十人小队一起去做个小车，并让小车从迷宫中找到复原的魔方并抓取从迷宫中走出。在这短短的三周期间，我们学到了不少东西。

在夏令营的第一周主要是学习制作机器人的一些基础知识。机械组学会如何运用 solidwork 去制图和使用 3D 打印机，并尝试画出宿舍的床。电控组掌握单片机的原理和绘制

电路图。算法组也在去熟悉相机识别魔方。大部分时间都是大家自己自学并在网上查阅资料。第一周过的很快也很匆忙。

在还没消化完第一周所学的知识就开始紧张的边学习边做小车。起初，机械组每天从早忙到晚，一直在设计机械臂和机械爪，想了一个又一个方案，所用的时间也比预期要长，在打印的过程中也耗费了不少时间，好在最后做出了一个满意的机械臂和机械爪。这一周，由于主控板的问题，嵌入式没那么紧张，同时也给了充裕的时间去学习 pid、陀螺仪、麦轮解算、舵机。

算法也没有闲着，也完成了魔方的基本识别，可能有的颜色识别不出来。这一周过得很忙碌。

在最后一周，暴露出很多问题，机械组一直在修改方案和制作一些配件的支架，因为没有仿真遇到了许多小麻烦。而嵌入式同样由于时间的紧迫要做出方案，但是在激光测距模块这方面浪费了不少时间，最后迫不得已放弃激光测距而采用超声波测距。同时算法组由于摄像头烧坏也耽误了不少时间。直到临近比赛的前两天才到场地去试走迷宫，这也是我们第一次熬夜，嵌入式一行一行的去修 bug，还要考虑场地打滑问题，到凌晨三点终于小车以缓慢而平稳的速度用时 2 分多走出了迷宫。所有人都很兴奋，但我们还没实施抓取魔方。等摄像头到了，算法组也加紧时间去调试。不管最后结果如何，我们每个人付出的努力都是值得的。