关于RM夏令营机器人的技术报告大纲(13组)

组员及分工：覃董学（机械、组长）、林文韬（机械）、李波萱（机械、Vlog）、黄毓莹（机械）、杨洲（嵌入式、视觉）、彭伟浩（嵌入式）、方彦哲（算法）

一、需求分析

视觉方向

分析比赛规则得知第二阶段有十五秒的滞停时间，己方操作手无法看到图像，为了更好的去利用赛场时间，可以利用视觉识别技术在十五秒中自动控制机器人完成相应的动作来提高得分效率。

机械方向

考虑到比赛规则，块的摆放会决定了效率问题，所以我选择了气动夹子，因为气动夹子不会被块的长短所影响，这样无论是什么方向夹取，都可以进行夹取。

二、所需技术点，关键词

- 气动夹子
- 镂空设计
- 行程可改变
- 结构重心问题
- 加固抬升
- 三角设计
- 曲柄滑块

三、总体方案

针对与比赛第一阶段和第二阶段的任务性，选择了双夹的结构可以进行累积块，同时可以多块运输，这样可以提高效率，气动只需要吸气放气的两个自由度。通过曲柄滑块的方案，双电机抬升导轨，在气动夹住上面第二层的方块后，下面限位夹夹住第一层方块后进行抬升可将方块分开。

四、各模块方案

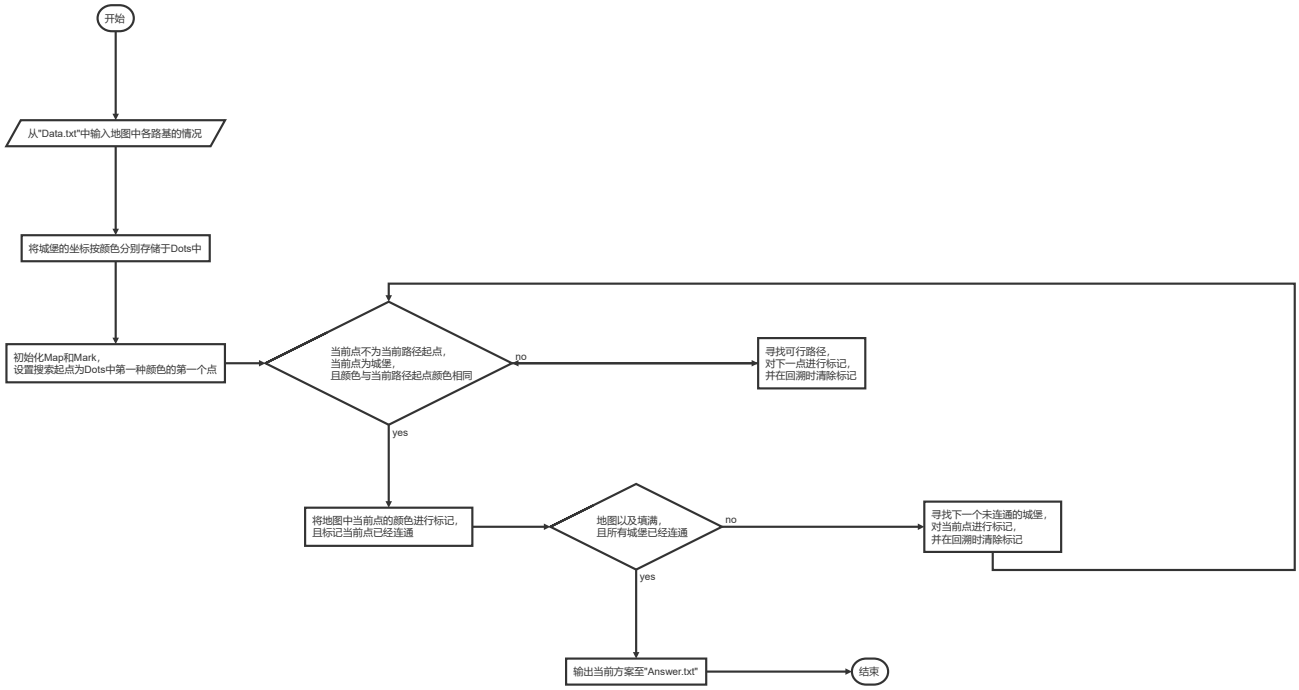
算法方向

方案决策部分

介绍

比赛第一阶段的方案决策部分使用深度优先搜索，并在编译时采用`O2`优化，以减少此算法在运算时间上的不足。

程序逻辑

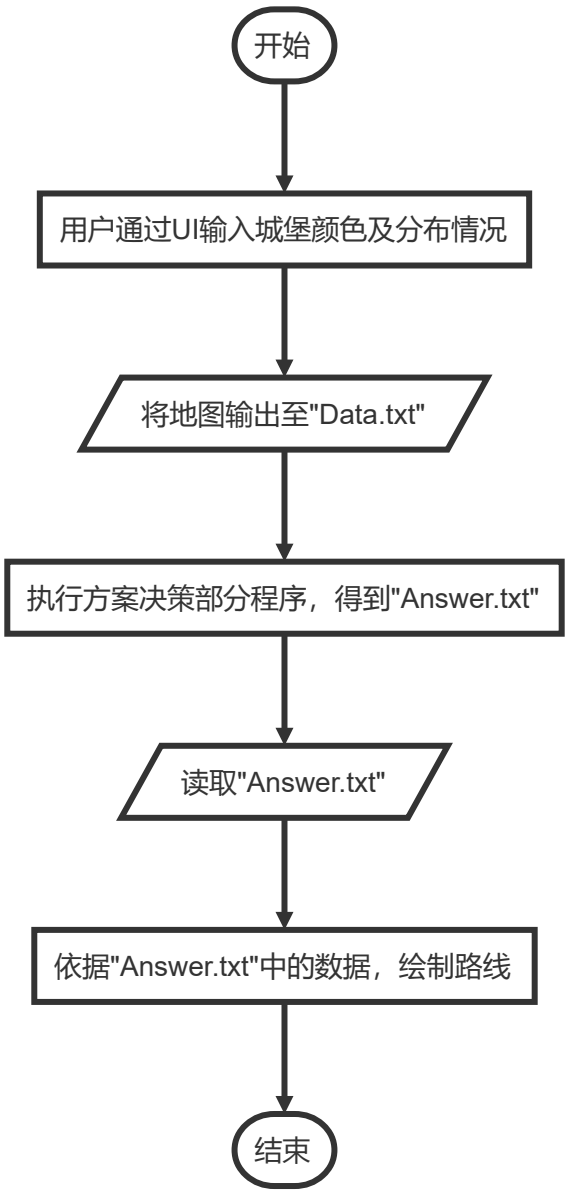


UI部分

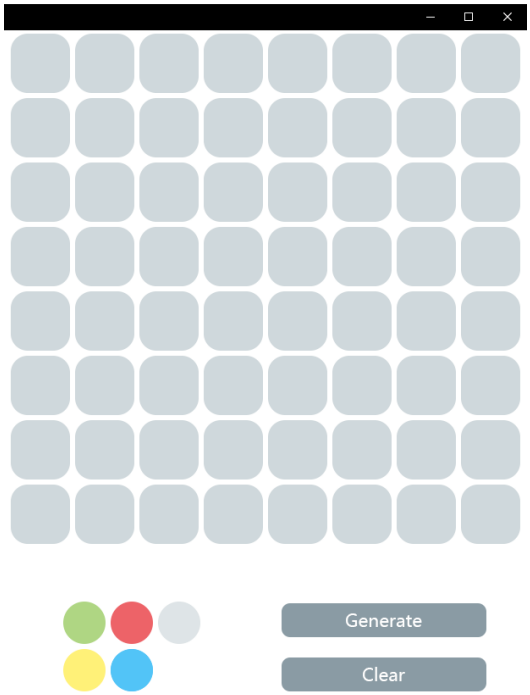
介绍

UI部分意为方便操作手在比赛过程中能够更加简单明了的对路径进行决策，以减少在准备时间的花费。其使用`C#`实现，并通过贴图来使其在给人的直观感受上更好。

程序逻辑


开始
用户通过UI输入城堡颜色及分布情况
将地图输出至"Data.txt"
执行方案决策部分程序，得到"Answer.txt"
读取"Answer.txt"
依据"Answer.txt"中的数据，绘制路线
结束

界面截图



嵌入式方向

can通信

```
void set_motor2006_current(int16_t mymoto2006_current[])
{
  static uint8_t data1[8];

  data1[0] = mymoto2006_current[0] >> 8;
  data1[1] = mymoto2006_current[0];
  data1[2] = mymoto2006_current[1] >> 8;
  data1[3] = mymoto2006_current[1];
  data1[4] = mymoto2006_current[2] >> 8;
  data1[5] = mymoto2006_current[2];
  data1[6] = mymoto2006_current[3] >> 8;
  data1[7] = mymoto2006_current[3];
  write_can(USER_CAN1, CAN_GIMBAL_ID, data1);
}
```

`can通信`分为两部分，发送`ID`和`信息`，`信息`是四个电机的`转速`，`ID`就是选择四个电机

uart通信

`uart通信`是一个一个字节传输的，但是程序似乎已经封装好了`DMA`，所以直接用就可以了，但一次要发定长的数据，不然可能会出现错位的情况

机械方向

覃董学，李波萱负责气动夹子，两人设计了双层夹子，第一层夹子起到限位作用，可以矫正方块，第二层夹子可以夹住方块后进行抬升，可以储块，这样提高了夹箱子的效率。夹子的摆放采用了错位夹，改变了夹子的形状，使上下层夹子不会碰撞，而且尽可能加大夹子与箱子的接触面积，增大表面摩擦力，能更好地夹起箱子。

黄毓莹，林文韬负责抬升，两人设计了曲柄滑块做为抬升装置，由于前面的夹子较重，所以夹子下垂较严重，两人设计了一个三脚架，并在合适的地方加大力臂，有效地防止了夹子下垂以及抬升下降过程中的不通畅。

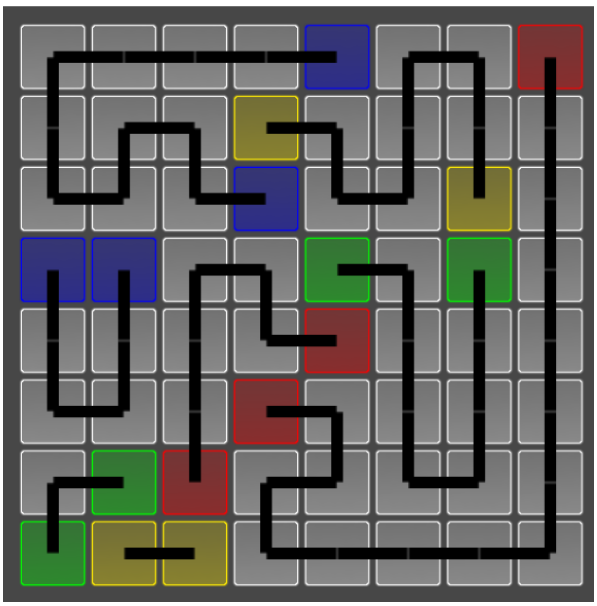
五、理论分析

算法方向

可行性分析

第一阶段

在读过本次夏令营的规则之后，其实比赛第一阶段所用的算法已经十分的显而易见了。而大佬们所实现的算法也不外乎两种：深度优先搜索和基于连通性状态压缩的动态规划。下面我会对于这两种算法的可行性进行简要的分析。

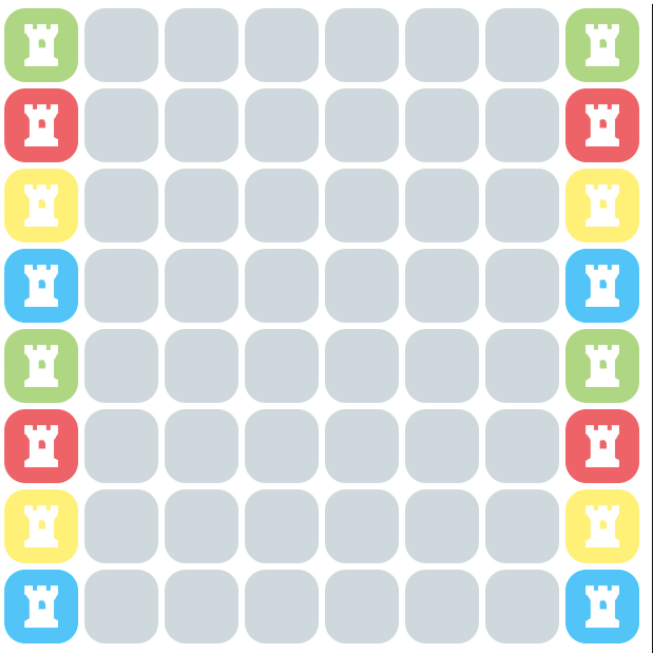


基于连通性状态压缩的动态规划

从比赛规则中第一阶段所附的图（如上图）中可以知道，本题是符合动态规划的最优性原理和无后效性的两个基本性质。那么我们就可以使用陈丹琦同学在她的论文《基于连通性状态压缩的动态规划问题》中所说的算法。这样就可以在期望的1秒钟的时间内完成答案的求解。

但是因个人实力以及之前从未接触过这一算法，在研究《基于连通性状态压缩的动态规划问题》这一论文将近一天后，本人对这一算法的掌握程度还达不到解决这一问题的水平。因此，这一算法对于本人来说并不可行。

深度优先搜索



第二阶段

根据比赛第二阶段的规则，在比赛过程中，在不同情况下理论可行的决策方案数是不能接受的。且比赛为两联盟之间的攻守交替对战，那么我们可以采用博弈的思路。

基于本人在热身赛时的观察，可以进行城堡搭建的小组几乎没有，那么我上述的思路在条件不改变的情况下是可行的。

运算速度分析

第一阶段

在这里，我在保证测试数据充分随机的前提下，对于本人所写的基于深度优先搜索算法的程序进行了测试。为了保证测试的严谨性，测试环境及测试数据如下：

```
➜ ~ g++ --version
g++ (Ubuntu 7.3.0-16ubuntu3) 7.3.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```
➜ ~ screenfetch
                          ./+o+-       ben@ben-OptiPlex-7050
                  yyyyy- -yyyyyy+      OS: Ubuntu 18.04 bionic
               ://+//////-yyyyyyo      Kernel: x86_64 Linux 4.15.0-29-generic
           .++ .:/++++++/-.+sss/`       Uptime: 0m
         .:+++o:  /++++++++/:--:/-      Packages: 2069
        o:+o+:++.`..```.-/oo+++++/      Shell: zsh 5.4.2
       .:+o:+o/.          `+sssoo+/    Resolution: 1600x900
  .++/+:+oo+o:`             /sssooo.   DE: GNOME
 /+++//+:`oo+o               /::--:.   WM: GNOME Shell
 \+/+o+++`o++o               ++////.   WM Theme: Adwaita
  .++.o+++oo+:`             /dddhhh.   GTK Theme: Vimix-Beryl
       .+.o+oo:.            `oddhhhh+  Icon Theme: Papirus-Light
        \+.++o+o``-````.:ohdhhhhh+     Font: Ubuntu 11
         `:o+++ `ohhhhhhhhyo++os:      CPU: Intel Core i5-7500T @ 4x 3.3GHz
           .o:`.syhhhhhhh/.oo++o`      GPU: intel
               /osyyyyyyo++ooo+++/     RAM: 881MiB / 7841MiB
                   ````` +oo+++o\:
                          `oo++.
```
```

过本人测试，得出在测试数据相同时，开启O2优化和不开启O2优化存在一定差距，但都可以在期望时间内得出答案。

```
➜  Summer_Camp_Alogorithm git:(master) ✗ time ./Road_Finder_Full
./Road_Finder_Full  0.84s user 0.00s system 99% cpu 0.838 total
```

```
➜  Summer_Camp_Alogorithm git:(master) ✗ time ./Road_Finder_Full
./Road_Finder_Full  2.68s user 0.00s system 99% cpu 2.677 total
```

第二阶段

因第二阶段不使用算法程序来进行方案决策，故不存在此阶段的运算速度分析部分。

嵌入式方向

底盘部分

陀螺仪辅助控制

直接在`chassis_costom.c`里面的`chassis_custom_control`加入的代码

目的

在操作车的过程中，因为某些物理原因，会在停止和启动过程中因为四个轮子的加速或减速时间不一样，而造成车辆偏转，因为陀螺仪可以检测角度，所以用角度来闭位置环。

```
double x,y,rand;
    if(chassis.vw==0&&rc.sw1!=1){
        times++;
        get_imu_data(&imu_data);
        if(angle_status==NONE){
            angle_z=imu_data.angle_z;
            angle_status=CACU;
        }
        else {
            chassis.vw=-pid_calc(&angle_pid,imu_data.angle_z,angle_z);
```

```
            rand=(imu_data.angle_z-angle_z)*PI/180;
            x=cos(rand)*(double)chassis.vx+sin(rand)*(double)chassis.vy;
            y=-sin(rand)*(double)chassis.vx+cos(rand)*(double)chassis.vy;
            chassis.vx=(float)x;
            chassis.vy=(float)y;
        }
    }
    else {
        angle_status=NONE;
        times =0;
    }
    if (times >=500){
        angle_z=imu_data.angle_z;
        times=0;
    }
```

一步一步看

第一步

```
if(chassis.vw==0&&rc.sw1!=1)
{
```

只有当没有旋转的时候才使用陀螺仪校准

```
get_imu_data(&imu_data);
if(angle_status==NONE)
{
    angle_z=imu_data.angle_z;
    angle_status=CACU;
}
```

在第一次进入循环的时候获取当前角度为标准角度

`angle_status`在第一次进入循环的时候值为0，之后为一，这样可以避免重复获取

```
else {
            chassis.vw=-pid_calc(&angle_pid,imu_data.angle_z,angle_z);
```

利用`pid`计算旋转出旋转速度防止偏转

```
rand=(imu_data.angle_z-angle_z)*PI/180;
x=cos(rand)*(double)chassis.vx+sin(rand)*(double)chassis.vy;
y=-sin(rand)*(double)chassis.vx+cos(rand)*(double)chassis.vy;
chassis.vx=(float)x;
chassis.vy=(float)y;
```

因为车身偏转，所以现在的`vx`和`vy`也会是偏的 因此将`vx`、`vy`看作是一个二维向量的分量，车身偏转就相当于坐标系旋转，因此我们只需将其乘上一个旋转矩阵，变换到新的坐标系中就可以了

```
else
{
    angle_status=NONE;
    times =0;
```

如果有旋转角度那么就什么都不做

```
times++;
```

```
if (times >=500){
        angle_z=imu_data.angle_z;
        times=0;
    }
```

这两段是因为陀螺仪会持续有误差，因为一个周期是`10ms`，所以每循环`500`次也就是5秒就重新获取一次数据。

其他部分

主要包括抬升电机和电磁阀

抬升电机

```
int16_t moto2006_speed[4];
int16_t moto2006_current[4];
int16_t moto2006_angle[4] = {0};
```

这些变量用于储存抬升电机的的`电流`，`速度`，`角度`。

```
void moto2006_calc_speed(void)
```

速度闭位置环函数，用`PID`计算`速度`

```
void moto2006_calc_current(void)
```

电流闭速度环函数，用`PID`计算`电流`

电磁阀

```
void clamp_set(CLAMP_ID id,GPIO_PinState set)
{
    switch(id)
    {
        case UP_CLAMP:
        {
            HAL_GPIO_WritePin(GPIOC,GPIO_PIN_2,set);
        }
        break;
        case DOWN_CLAMP:
        {
            HAL_GPIO_WritePin(GPIOB,GPIO_PIN_0,set);
        }
        break;
    }
}
```

我们用了两个电磁阀，直接设置`GPIO口`高低电平

抬升电机

```
void base_complex_control(void)
```

主要逻辑是当获取的抬升电机速度为零的时候，停在某个位置不动，当速度不为零是，按照速度移动，这个函数不包含数据获取，是主要控制函数。为了防止下降速度过快撞到底部导致机器损坏，当抬升装置接近底部的时候会限制其速度。

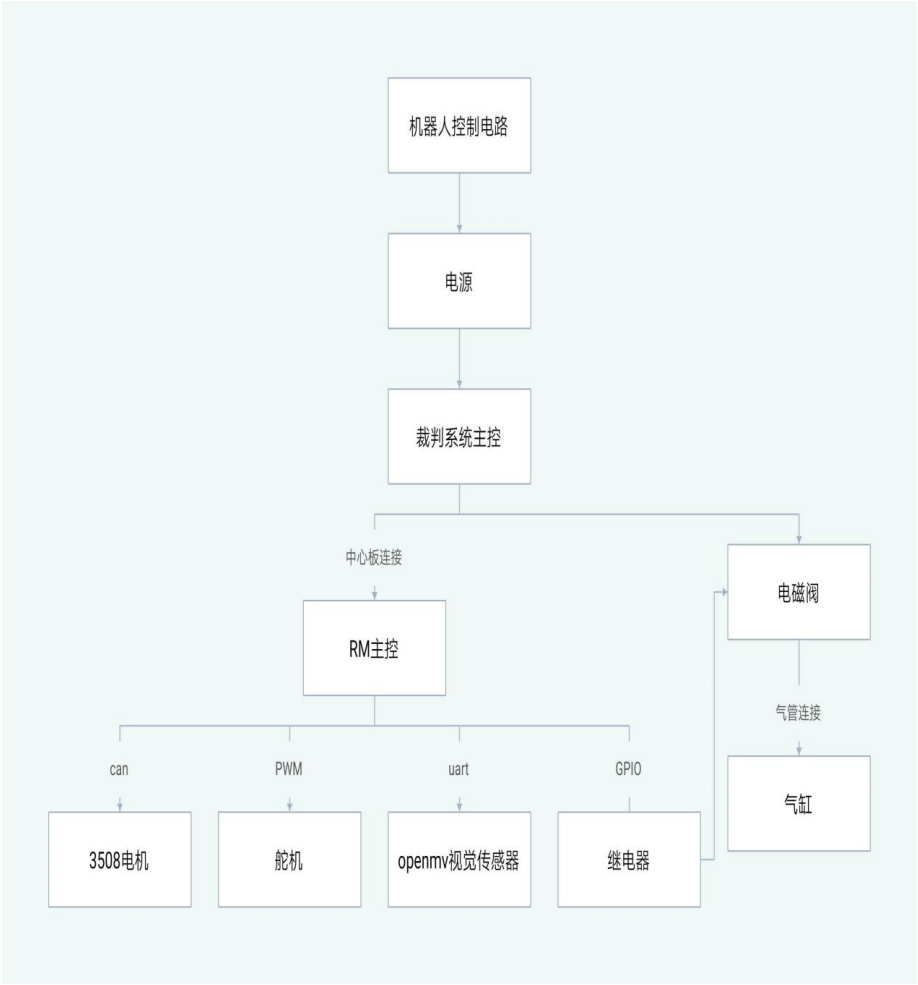
与遥控器通信

```
void simple_get_clamp(void)
void moto2006_key_get_speed(void)
```

用拨杆控制气动夹，之后直接将遥控器一个通道的值乘上一个最大速度再除以遥控器最大值，就可以作为期望遥控器速度了。

硬件线路

从电源出发，分析RM主控、裁判系统、各个执行器以及各种传感器的供电电压大小和供电方式，了解主控的板载接口功能，将相应的电压输入、输出端口与电源串联，将其能够正常的供电使用，然后将电机和电调相互连接并接到RM主控相应的can端口，舵机连接在主控相应的PWM的IO口，视觉传感器根据通信方式对应连接UART端口，继电器通过分析使能方式连接相应的GPIO口来控制气缸的开闭，分析气路传导将气瓶、气缸电磁阀连接，组成一个能控制的气动执行机构，最后反复检查各个线路，直到确认正确无误，避免线路模块烧毁。



视觉方向

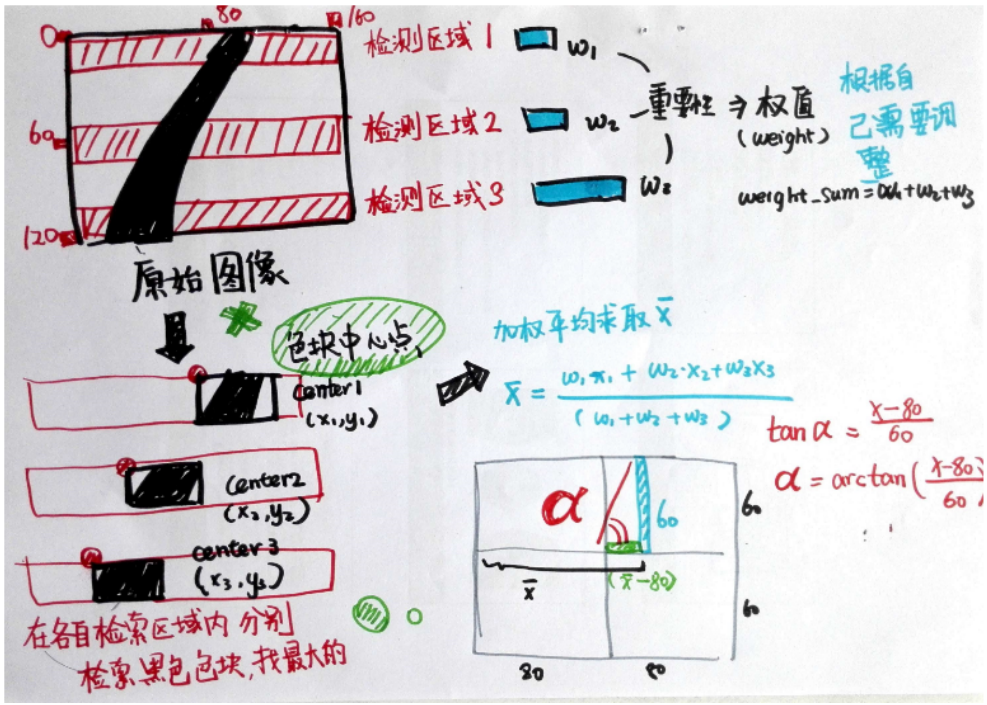
思路

利用`openmv3R2`视觉传感器的视觉识别去巡比赛场地中的黑线，每一个十字相交的黑线作为一个标识符，当巡到十字后，前进一段固定的距离（通过测量获得）使机器人能够达到县的中段，因为城堡或者路基在黑线围成的正方形正中间，所以走到边长的一半之后转90度能够更好的面朝积木块，接着识别最短路径中的色块，如果没有色块继续寻找，如果色块不是城堡则继续寻找，当找到城堡之后就打开夹子停在城堡处，为下一步的遥控节省时间。

代码原理

巡单轨黑线

将传导的图像划分多个检测区域，根据自己的需要和重要性（远近）调节每个区域的权值，在各自的检测区域检索最大的黑色色块，根据反馈的色块的中心点坐标加权平均取结果值，然后通过图像坐标轴和加权值进行勾股定理，计算出相应的偏角。根据这个偏转的量来调节底盘电机的运动达到巡线的作用。



识别十字

利用巡线检测黑线的矩形区域的大小来区别是否识别到十字，当路过十字的时候该检测的区域中矩形的最大色块会大于其他的检测区域，通过openmv开发环境中的代码实时反馈获取矩形的长宽，并与其他检测区域的矩形比较面积大小，当达到某一个设定的面积区间，说明检测到了十字。

找到视野中最短路径的色块

通过设置各种积木色块的LAB阈值，设定相应的ROI区域，通过函数取寻找相对应的颜色并且特征识别矩形，然后反馈色块的中心点坐标，与图像的中心点坐标计算距离，通过比较距离找到距离视觉传感器最近的色块。

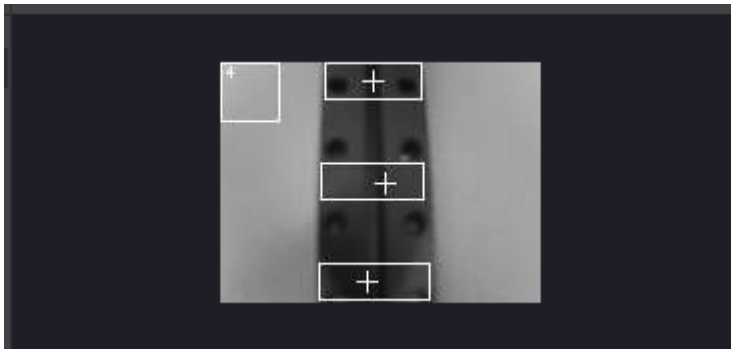
识别城堡

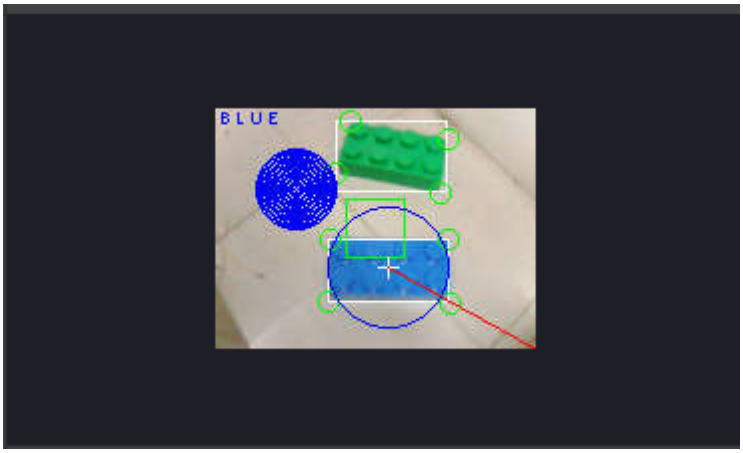
利用函数反馈检测矩形区域的长宽进行大小的判断，当检测的特征识别矩形区域达到两块积木块长宽时，证明找到了城堡。

可行性分析

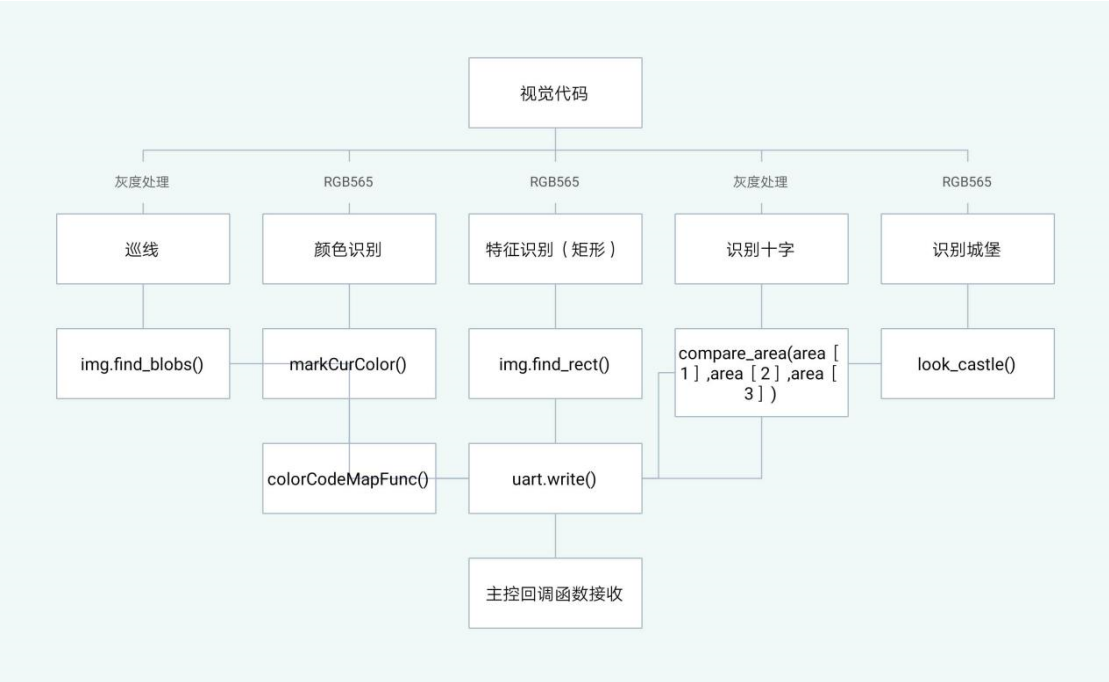
`RM主控`与之间能够通过`UART协议`来通信相对稳定传输相应的数据，主控处理数据来控制机器人。视觉传感器受光值影响很大，赛场环境很复杂，误差会很大，先利用灰度处理去巡线会降低影响，达到机器人平稳自动运行的效果。

效果图





流程图



机械方向

此次设计机器过程中，我们对受力处和易损处进行了受力分析，发现气动夹位置的玻纤收到气动的力较大，会使玻纤形变，玻纤形变之后会导致箱子夹不稳，会影响夹箱效率。所以我们打了几块pom板，在受力处进行加固，有效地防止玻纤形变，有效地保证了夹箱效率。

六、制作与测试流程

机械方向

一开始比赛规则出来之后，出了第一套方案，确定了小底盘的底盘结构和轮吸加气动夹子限位的结构，但是比赛场地上资源区的限位位置，推翻了第一套方案进行了修改，修改成了双夹的结构，运用了四个气缸的驱动，进行夹子的开合，在第一轮热身赛的时候，底层夹子与地面限位会进行冲突，过后进行了调试和调整，将底层夹子抬高和修改，并且将夹子进行镂空设计，减少了前面夹子的重量。

七、结果与评价

机械方向

项目在设计的时候，并没有考虑到机器的储蓄功能，拼箱子时的校准的问题比较严重，且拼箱效率不高，所以在赛场上放弃拼箱。此次机械设计我们考虑了较全的功能，是场上功能较全的机器之一。

八、附录

算法方向

方案决策

Alogrithm.zip

UI

UI.zip

机械方向

机械图纸.zip

嵌入式方向

嵌入式代码.zip

视觉方向

视觉代码.zip

九、感想与感悟

方彦哲

本次夏令营我的收获很多，在许多方面提高了自己的水平。

彭伟浩

这次夏令营，我收获了很多，学到了很多知识，比如调pid，交到了一些朋友，这次活动，让我懂得了团队合作的重要性，只有多沟通，才能使整个项目向前推进，不然就会发生很多事故，这次夏令营，让我感到很开心。

杨洲

夏令营到现在已经接近尾声，虽然只有短短的二十天，但却受益无穷。这一路坎坎坷坷，有欢笑、有犯愁、有坚持、有拼搏。不断的修改，持续的争辩才构建了今天在赛场驰骋的机器人，讲到这儿我觉得这个过程已经很知足了。不管最终的成绩如何，这次经历永远都是我人生轨迹浓墨的一笔！当然也收获了很多的友谊，懂得如何去更好的合作，与队友相处。感谢我的队友们，谢谢你们，这些天和你们一起努力的日子很快乐，以后可能天各一方，情终不散！

林文韬

对于管理方面从组长那学到了很多，也通过导师助教那学会了很多新的一下结构设计。

覃董学

第一次当组长，当组长可以学会很多，包括技术和管理。

黄毓莹

在夏令营中，和队伍讨论激发了许多的想法，也感受到了团队分工的重要性。也有了许多的机会练习solidworks的应用。

李波萱

在这次的夏令营中我学到了很多知识，认识了很多志同道合的小伙伴，见到了很多与之前所接触的不一样的东西，这样的经历是以前不曾有过的。