上海交通大学， 2017-2018 学年第 2 学期

《工程实践与科技创新 IV-D（智能车控制算法设计与实践）》实验报告

小组编号：9

组长姓名学号：严威豪 516021910806

组员姓名学号：赵寅杰 516021910626

刘启明 516021910585

任课教师：杨明 教授

日期：2018 年 5 月 16 日

目录

1. 实验简介

本实验是在 CyberTORCS 平台上进行的智能车控制仿真实验。仿真平台由杨明教授实验室成员黄炫圭、杨辰兮开发。用户可通过设计、编写自己的算法，利用车辆控制的输入输出变量，根据道路弯度、车辆偏航等因素改变车体状态，实现平台中车辆的自动行驶。本实验分为三部分：巡线实验、跟车实验、泊车实验。编程平台为 Visual Studio 2017。

巡线实验给出 6 条难度不同的赛道，要求编写一套算法，使车辆能沿着这 6 条赛道跑完全程，考评赛道为 6 条已知赛道和 1 条未知赛道，以（完成时间+碰撞量/10）/10+Error/Car_L 作为评判标准，巡中线精度越高，速度越快，碰撞量越小，则算法越完善。

跟车实验要求设计领航车行为，同时编写后车算法，使它能跟随领航车行驶，跟车过程中不能与领航车发生碰撞，也不能超过领航车，以两车的平均间距保持在 5m 作为算法的评判标准，车间距偏差越小，碰撞量越小，则算法越完善。

泊车实验给出 5 个泊车位的坐标，要求编写算法，使车辆能够以车尾或车头入库方式泊入车位，考评车位为 5 个已知车位和 1 个未知车位，以泊车速度和精度（车辆四个顶点距离标准位置的偏差）作为评判标准，速度越快，精度越高，则算法越完善。

希望同学们在完成实验的同时，对智能车控制有更多了解，也欢迎大家加入智能车团队！

上海交通大学 实验报告

2. 巡线实验

一、实验目的和要求
二、实验内容
三、实验工具
四、操作方法和实验步骤
五、算法框架与程序代码
六、实验结果与分析（必填）
七、讨论、心得

一、实验目的和要求

1. 学习、掌握 CyberTORCS 仿真平台的使用
2. 理解用于车辆控制的输入、输出变量的确切含义
3. 学会用 Microsoft visual studio 2017 编程实现车辆控制
4. 通过实验对 PID 等控制算法有更深入理解
5. 设计自己的巡线控制算法

二、实验内容

1. 参看软件说明书，熟悉 CyberTORCS 的各菜单
2. 选择 CyberOnHand 菜单，任意选择赛道，利用上下左右键手动控制赛车，对车辆、赛道有直观感受。
3. 用 Microsoft visual studio 2017 打开 driver_onhand.sln，通过改写 driver_onhand.cpp 中的代码输出自己感兴趣的参数，同时了解档位与发动机转速之间的关系。
4. 用 Microsoft visual studio 2017 打开 driver_cruise.sln，将说明书中的简单例程写入 driver_cruise.cpp 中，选择 CyberCruise 菜单，观看车辆运动状态
5. 通过改写 driver_cruise.cpp 中得代码，对 6 条不同难易不同的赛道进行巡线高精度控制编程。

要求：最终能用一套程序跑完所有赛道，不能在程序中对赛道进行识别以使用不同代码。

三、实验工具

CyberTORCS 仿真平台、Microsoft visual studio 2017

四、操作方法和实验步骤

参见软件说明书

五、算法框架与程序代码

1、赛道识别

因为赛道有公路和土路之分，而这两种赛道的特征有很大的不同，对车辆的行进有较大的影响。所以用某种方法将这两种道路区分开来，分别进行算法调整就很有必要了。我们实现公路与土路识别的基本方法是用一个 count 变量进行计数，计算前 count 次的平均速率，由于土路上车辆在相同油门下的加速度偏小一些，所以刚开始一小段的时间内平均速度也较公路而言偏小一些，找到土路地图 dirt6 与普通公路的数值差异，以此作为判断依据。实现代码如下：

```
if (count <= 250)
{
    speedsum += _speed;
    count += 1;
    speedaverage = speedsum / count;
}
if (speedaverage <= 35 && count > 250)  …  //判断为土路
else  …  //判断为公路
```

2、速度控制

基本思路是通过与速度相关的动态预瞄点，计算前方道路半径，并根据道路类型（是否为 dirt）以及前方道路的不同半径落在不同的区间中，使用不同的公式计算得出期望速度，并且给出不同的期望上下限。因为不同的道路类型在转弯的时候对速度的要求是不同的，而且对于同一条赛道，在半径小的地方速度也应相应减小，避免由于速度过快转弯过猛而导致车辆脱离中心线造成误差过大使成绩下降。通过半径计算完成后，又引入了角速度的绝对值来影响期望速度，引入角速度的思想与物理当中的圆周运动吻合，在角速度大的时候需要一个很大的线速度来保证转弯时的曲率半径基本保持不变，以更加贴合中心线。最后借助 PID 来控制油门和刹车，下面展示通过速度计算预瞄点距离，再通过预瞄点处的半径得到期望速度的部分代码：

```
startPoint = _speed * 0.2;
delta = constrain(10, 25, _speed / (fabs(_yaw) + 1) / 7);
//用来计算预瞄点的位置，以计算 c.r
if(dirt)
{
    if (c.r <= 50)
        expectedSpeed = constrain(30, 45, c.r + 10);
    else if (c.r <= 70)
        expectedSpeed = constrain(45, 80, 0.45*1.6*c.r);
    else if (c.r <= 90)
        expectedSpeed = constrain(60, 90, 0.45* 1.8* c.r);
    else if (c.r <= 100)
        expectedSpeed = constrain(80, 95, 0.45* 1.8* c.r + 15);
```

```
            else
                expectedSpeed = constrain(85, 100, 0.4 * c.r * 1.35);
        }
        else
        {
            if (c.r <= 80)
                expectedSpeed = c.r + 16;
            else if (c.r <= 90)
                expectedSpeed = constrain(60, 95, 0.45* 1.8* c.r + 15);
            else if (c.r <= 100)
                expectedSpeed = constrain(80, 100, 0.45* 1.8* c.r + 20);
            else
                expectedSpeed = constrain(80, 100, 0.4 * c.r * 1.4);
         }
            expectedSpeed += fabs(_yawrate) * 13;
```

3、方向控制

预瞄前方 1m、3m、4m、5m、6m，计算出反正切值，再加一项调整因子来计算 D_err，其中的参数根据不同的道路半径以及道路类型进行不同的优化。因为巡线要控制精度，所以每时每刻都得保证与中线基本重合，所以我们的预瞄点选得比较近，就是为了防止由于前方的中线位置变化而提前做出反应致使当前的误差较大的情况。后面那一项调整因子是经验公式，单纯地把觉得与 D_err 相关的变量按正负相关凑成了一个公式，然后调了调参数，使总体成绩略微提升。最后是常规的 PID 来控制*cmdsteer 实现代码部分展示如下（实际代码中我们根据不同路况进行了不同的参数调整，由于篇幅所限我们仅展示框架和部分代码，但是整体结构不变）：

```
    float a1 = atan2(_midline[1][0], _midline[1][1]);
    float a3 = atan2(_midline[3][0], _midline[3][1]);
    float a4 = atan2(_midline[4][0], _midline[4][1]);
    float a5 = atan2(_midline[5][0], _midline[5][1]);
    float a6 = atan2(_midline[6][0], _midline[6][1]);
    switch (static_cast<int>(c.r / 100))
    {
       case 5:
       case 4:
            kp_d = 0.5;
            D_err = 1 * (-2 * a6 - 2 * a4 - 4 * a3 - 2 * a1) - 2.5 * _midline[delta
- 5][0] / cos(_yaw) / _width;
            break;
       case 3:
            kp_d = 0.85;
            D_err = 1 * (-2 * a6 - 2 * a4 - 4 * a3 - 2 * a1) - 1.5 * _midline[delta
```

```
- 5][0] / cos(_yaw) / _width;
            break;
        case 2:
            kp_d = 0.6;
            D_err = -1.5 * a6 - 2 * a4 - 4 * a3 - 3 * a1;
            break;
        case 1:
            kp_d = 0.4;
            D_err = -1.5 * a6 - 2 * a4 - 4 * a3 - 3 * a1;
            break;
        case 0:
            kp_d = 0.3;
            D_err = -1.5 * a6 - 2 * a4 - 4 * a3 - 3 * a1;
            default: break;
    }
    //the differential and integral operation
    D_errDiff = D_err - Tmp;
    D_errSum = D_errSum + D_err;
    Tmp = D_err;
    //set the error and get the cmdSteer
    *cmdSteer = constrain(-1.0, 1.0, kp_d * D_err*1.2 + ki_d * D_errSum + kd_d *
D_errDiff);
```

六、实验结果与分析

我们巡线的实验结果如下：

赛道	Dirt6	CG-speed	E-road	E-track4	Brondehach	Street1	Total
成绩	123.09	79.77	105.07	104.72	94.29	96.38	603.32

第一次自测成绩不太理想，于是又在之前的基础上加班加点地修改代码。终于在资格赛上取得了第四名的成绩，最后在巡线比赛上出人意料地反超成为第三（可以说是很惊喜了）。这一次算是 TORCS 入门，也算是摸着石头过河。之前没有接触过类似的东西，完全从零开始，反而取得了比较好的结果，令人很是欣慰。

比较我们组与第一名的区别，主要还是在 dirt 这条赛道。虽然我们也进行了识别，可是由于算法还是用的与公路相差无几，仅仅是参数进行了特殊优化，相当于存在一个天花板无法逾越，所以在成绩上并没有实质性的进展，而另外几条公路赛道我认为还是可以，从比赛时的 X 赛道的成绩来看，通用性也不差，这也归功于我们在撰写算法已经增减参数时有意地思考了通用性问题，毕竟有一条 999 就完了。所以即便后来通过赛道宽度的三次回归拟合出了一个很好的补偿变量，在已知赛道上表现相当不错，但考虑到这个拟合普遍性较差所以最后还是弃用了。

不过这次凸显的问题也很多。首先，因为是初学者，所以我们在开始的第一周本来给三个人每人分配了几篇文献，打算大家分开阅读之后再进行汇总找出我们认为最好的一种或者几种算法并加以实现，最后再由成绩盖棺定论。不过遗憾的是，就我个人而言，文献当中讲到的算法高深繁杂，理解不易，实现更难。加上个人的英文水平不是很好，阅读能力奇差，所以读一篇文献需要耗费大量的时间和精力，即便如此也不能透彻地理解算法的内涵，所以做了很多无用功。最后我们的算法基本是按照助教给的模板的预瞄点的思路，没有用到那些高层次的譬如 Stanley 之类的模型，也没有从文献中获得丝毫助益。我觉得在这一点上吃了很大的亏。我们如同盲人摸象般摸索着算法的路子，消耗了大量的时间。

其次，我个人认为整个参数的结构相互之间的依赖性很强，某个参数的很小的变动就可能引起小车行为的大幅度改变。这与我预期的大相径庭。我本来以为各个参数之间是垒砖头一样的比较粗放结构，互相影响，而且我想当然地认为这里面是有一种负反馈机制的，所以一个参数的很小的改动就像拆掉一块砖，并不会有多大的影响。事实证明这种想法是完全错误的。整个体系要比我想象的精细地多，各个参数之间看似风平浪静实则暗潮涌动，如同一座纸牌搭建的塔楼，又像一串精心制作的多米诺骨牌，一触即倒，而且其间的复杂性和关联性不是我人脑可以比拟的。

最后是调参当中的盲目性。感觉 TORCS 真是一个很神奇的东西，就拿我个人经历来看，在我以为可以很好地理解巡线算法的逻辑后，我会按照自己预期的目标去调整参数，比如在看完小车的整体表现之后，我会凭主观臆断找出哪一点做的还不够好，哪一点是限制成绩的主要因素等等，然后按照找出的问题去定向地改变参数。比如我觉得速度太慢了，我会在期望速度上再加一些因子，可是往往事与愿违，在经过我以为的可以调高小车速度的一波操作之后，小车的整体速度反而会下降，成绩会下降的非常明显，这着实让人倍感受挫，也让调参的过程变得冗长枯燥。总之巡线结果还是令人满意的，只是过程曲折艰辛，令人记忆犹新。

七、讨论、心得

总体来讲，这次巡线对我们来说还算是一次愉快的经历，虽说是第一次接触，可能是因为巡线本身在三个比赛中是最基础的一个，所以这次并没有让人感到太多的焦虑。而且与后面两次相比，也没有出现过无法掌控的局面，再加上初上手的新鲜感等等，着实有趣得紧。

感觉这门课真的对创新有很高的要求（尤其是像我们一样从来不看文献只有自己拍脑袋想算法的组来说），要一步一步地构成整个框架，再一点点地填充关键的公式，当中不乏有各种脑洞和我们也不甚理解的偶然间想出的公式。在想公式的过程中确实相当有趣，像是一个盲人漫无目的地搜寻中无意间获得了一根手杖一般，令人欢欣鼓舞。其间自然也有很多的失败和不甘，也不乏失望和灰心，但总归下场是好的。

其中我觉得最重要的就是保持心态以及高效的执行力。感觉另外两名队友在这方面做得很棒，即便感觉要崩但依旧能谈笑风生，并且笑着调下去。在后期大家普遍感觉疲劳无力的时候，也是他们两个一直在坚持，而我则大脑空白，神志不清，精神涣散。

而在调参的过程中，也随处可见马克思主义基本原理的运用。比如说物质存在形式原理，小车是以程序的方式存在于这个空间，并且占据了我们~~大量~~的时间，运动就是它的存在方式，而时空就是运动的存在方式。

其次是实践本质原理。我们通过实践去完成我们所要达到的目的，我们花费了时间在 visual studio 上进行实践，这就是实践的直接现实性。而我们通过自己的脑回路的思维以及双手的创造使得一个本来没办法跑完所有赛道的程序最后变成总成绩较好的程序，这就体现了我们的主体能动性。

还有比较明显的就是质量互变定律。刚开始大幅度修改框架，使得总成绩有了整体的飞跃，这是质变。其后就进入漫长的量变过程，包括优化参数，添加一些无关痛痒的变量和公式，这时总成绩是缓步提高的，甚至有可能会有退步。但这绝不是历史的倒退，而是螺旋上升的结果。之后又会在某一个不经意的瞬间，可能参数突然调到了最优的地步，或者突然添加了一个相当的公式，此时总成绩又会发生一次突变，这就是之前的量变积累到一定程度后导致质变的结果。

至于赛道识别，就是将 dirt 区别对待这一部分，则是矛盾的普遍性的特殊性原理的最典型的应用。两种不同类型的赛道的基本实现方法是一样的，公式大同小异，在公路上跑出不错成绩的代码放到 dirt 上也基本不差，这说明抛开路面因素，同一套代码对不同的道路类型的适用程度是相同的，也就是解决了矛盾中普遍性的那一部分。但是从参数的角度考虑，对同一个算法来讲，不同的赛道有其独有的最优参数组合，而且这个最优的组合的成绩可能要比普适的参数要好很多，这就是矛盾之中蕴涵的特殊性。每条赛道都有其独到之处，天底下没有两条完全相同的赛道，一辆小车也不可能踏入同一条赛道，所以既然可以，就有必要对识别出来的 dirt 赛道调一套更加针对的参数。

最后，实践是检验真理的唯一标准。

上海交通大學 实验报告

3. 跟车实验

一、实验目的和要求
二、实验内容
三、实验工具
四、操作方法和实验步骤
五、算法框架与程序代码
六、实验结果与分析（必填）
七、讨论、心得

一、实验目的和要求

1. 掌握 CyberTORCS 仿真平台的使用
2. 理解新增的输入变量的确切含义
3. 设计自己的头车程序和跟车控制算法

二、实验内容

1. 打开 wtorcs_new.exe，选择 CyberFollow 菜单，单击“New Race”，切换视角，观察领航车路径。
2. 设计头车算法，用 Microsoft visual studio 2017 打开 driver_lead.sln，在 driver_lead.cpp 中的适当位置编写自己的头车程序。
3. 设计跟车算法，用 Microsoft visual studio 2017 打开 driver_follow.sln，在 driver_follow.cpp 中的适当位置编写自己的跟车程序。

   跟车规则：后车与前车的平均距离保持在 5m，偏差越小越好，后车不得有任何部位超过前车末端，也不得有 damage 大于 1 的碰撞，否则跟车失败。

三、实验工具

CyberTORCS 仿真平台、Microsoft visual studio 2017

四、操作方法和实验步骤

参见软件说明书

五、算法框架与程序代码

控制跟车主要分为两个方面：第一是速度加速度控制，第二是转向控制，接下来我们将分别介绍两方面的代码。

首先介绍速度及加速度控制部分的代码，这部分又分为前车速度与加速度的获取、速度控制、加速度控制三个小部分：

1、获取前车的速度和加速度：

我们设置两个数组 lead_y 和 lead_v，其中 lead_y 用于保存前车在后车的坐标系中的 y 坐标值，通过 Δy/Δt 计算出前车相对于后车的速度，由相对速度与后车速度相加即可得到前车的速度；lead_v 用于保存计算出来的前车速度值，并通过 Δv/Δt 计算出前车的加速度

```
for (i = 0; i<2; i++)
   { lead_y[i] = lead_y[i + 1]; lead_y[2] = _Leader_Y; }
Leaderspeed = _speed + (lead_y[1] - lead_y[0]) * 180;
for (i = 0; i<4; i++)
   { lead_v[i] = lead_v[i + 1]; lead_v[4] = Leaderspeed; }
Leaderacc = (lead_v[3] - lead_v[2] + lead_v[1] - lead_v[0])/(0.020 * 3.6 * 2);
```

2、 速度控制：

由于理想目标是和前车保持 y 方向上 10m 的距离，所以我们根据_Leader_Y 的大小分情况设置预期速度。考虑到高速时的急刹问题，安全跟车距离应随着速度的增大而增大，所以我们以 12 + _speed / 80 为分界线来区分目前是跟得较近还是跟得较远。

当跟车跟得较近时，预期速度在前车速度的基础上，由前车加速度和与前车的距离综合进行调节，其中加入前车加速度这个变量是出于后车调整的滞后性考虑的：当前车加速时，预期速度应该比前车速度更高，当前车减速时，预期速度应该比前车速度更低，这样才能跟得上前车。另外，实际距离与理想距离相差越大，预期速度就应该越大。代码中的_speed / 10 依然是出于高速急刹的安全距离考虑。

当跟车跟的较远时，由于距离较远，急刹的安全距离不再纳入考虑，此时两车的距离成为影响预期速度的主要因素。

此外，在以上的基础上，我们还设置了一些条件进行调整，当距离非常近时，我们将预期速度设为 0，当距离较远时，我们将预期速度乘以 1.1，速度控制部分的主要代码如下：

```
 if (_Leader_Y < 12 + _speed / 80)
     expectedspeed = Leaderspeed + 0.02 * Leaderacc + (_Leader_Y - _speed / 100
- 10) / 5;
 else
     expectedspeed = Leaderspeed + 0.02 * Leaderacc + _Leader_Y / 5;
 if (_Leader_Y < 10.5)
     expectedspeed = 0;
 else if (thistimedistance > (_speed / 100 + 10) && Leaderacc > 10)
     expectedspeed *= 1.1;
 else if (thistimedistance > (_speed / 100 + 11) && Leaderacc > 0)
```

```
        expectedspeed *= 1.1;
```

3、 加速度控制

整体思路是通过前车速度、两车距离、两车速度差以及 PID 进行调节。PID 调整部分我们使用了 P 和 I 两个参数，具体的实现代码遵循助教提供的原始代码，除参数外未作改动。

我们将加速度控制分为加速控制和减速控制两部分，并且根据不同头车的不同特点进行了特别的参数调整和优化。比如 B1 号头车存在着超高速行驶加急刹的状况，15 号头车是高速平稳行驶，23 号和 31 号头车是低速平稳行驶，我们进行了针对性的参数调整。

加速控制基本思路是两车距离或前车加速度越大，跟车的加速度相应越大；前车速度越大，跟车加速度相应越小，以防止急刹撞车，主要代码实现举例如下（不同赛道参数有略微不同）：

```
if (curSpeedErr > 0 && (_Leader_Y - _speed / 90 - 10) > 0 && Leaderacc > -10)
    {
        *cmdAcc = constrain(0, 1.0, kp_s * curSpeedErr * (100 * (_Leader_Y - 10)
/ (_speed + 10) + 1) + ki_s * speedErrSum  + Leaderacc / 95 * _Leader_Y);
        *cmdBrake = 0;
    }
```

减速控制的基本思路是跟车速度越大、两车距离越小、前车加速度越大，跟车刹车时的 brake 也相应越大，主要代码实现举例如下（不同赛道参数有略微不同）：

```
    *cmdBrake = constrain(0, 1.0, -kp_s * curSpeedErr * (_speed / (_Leader_Y) / 12
+ 1) - ki_s * speedErrSum - kd_s * speedErrDiff - Leaderacc / 8 / (_Leader_Y - 10));
    *cmdAcc = 0;
```

另外在赛道拐弯处，我们应该使刹车稍缓，以防甩尾漂移：

```
if (c.r <= 150)
    {
        *cmdBrake = constrain(0, 1.0, -kp_s * curSpeedErr * (_speed / (_Leader_Y)
/ 3 + 1) - ki_s * speedErrSum - Leaderacc / (Leaderspeed) / 10 / (_Leader_Y - 10));
        *cmdAcc = 0;
    }
```

在实际操作过程中我们发现B1头车成绩不甚理想，我们特别针对B1头车调整了参数。判断B1头车比较简单：我们发现该头车速度极快且初始加速度极大，只需要计算前100个时钟周期内头车的平均速度即可把B1与其它头车区别开来。

此外，我们发现初始两车距离为 15，大于 10 的最小跟车距离，于是我们把刚开始一段时间内的跟车加速度强制设为 1：

```
                    if (++count < 250 && Leaderacc > 10)
                        *cmdAcc = 1.0;
```

当前车的刹车加速度很大时且两车距离非常近时，我们编写了一段保护函数，将刹车设置为 1：

```
            if (Leaderacc < -20 && (_Leader_Y - _speed / 100 - 10) < 0)
                {
```

```
            *cmdBrake = 1.0;  *cmdAcc = 0;
        }
```

接下来介绍转向控制部分的算法及代码。我们的转向控制放弃了 PID 的调节，在非弯道行驶时只需将前车位置作为预瞄点，而在弯道时适当增加道路中心线预瞄点的权重以防止撞墙及驶入草地。预瞄点设置如下，其中一个预瞄点为前车，另一个备用预瞄点为前方 5m 沿道路中线处。

```
float leaderror = atan2(_Leader_X, _Leader_Y);
float a5 = atan2(_midline[5][0], _midline[5][1])
```

通过上述两个公式的计算间接控制车辆的转向，代码实现如下：

```
D_err = -15 * leaderror + x * a5;  //x为参数，由不同头车进行调整
*cmdSteer = constrain(-1.0, 1.0, D_err);
```

此外我们还加入了弯道外侧转向保护机制，防止跟车在弯道外侧行驶时向右撞墙，具体实现代码如下（其中函数 djudge 用来判断小车是行驶在弯道的外侧）：

```
if (fabs(_midline[2][0]) >= _width / 3 && djudge(x1, lastx1))
    *cmdSteer = constrain(-1.0, 1.0, kp_d * D_err * 0.6 + ki_d * D_errSum + kd_d
* D_errDiff);
else if (fabs(_midline[2][0]) >= _width / 2.4 && djudge(x1, lastx1))
    *cmdSteer = constrain(-1.0, 1.0, kp_d * D_err * 0.3 + ki_d * D_errSum + kd_d
* D_errDiff);
if ((_midline[2][0] < (1 - _width) && *cmdSteer > 0) || (_midline[2][0] > (_width
- 1) && *cmdSteer < 0))
    *cmdSteer /= 10;
```

通过上述算法和代码我们完成了跟车实验，但是在调整代码的过程中我们也遇到了很多问题，也想过很多办法予以解决，具体分析请参见下面的分析。

六、实验结果与分析

在第一次跟车作业时，一开始感到无从下手，先把任务分成了三块：前车速度及加速度的获取，速度加速度控制及转向控制，每个人负责一块之后再进行合并。周五晚上小组讨论三个人一起编了一晚上，当晚的初步结果是前车的加速度还不是很稳定，速度和加速度控制以及转向控制基本框架成型。但我们还是无法跑完所有的头车。

周六的小组讨论之前，各组在排除头车上花了很多功夫，我们小组也在测试的过程中发现了一些耦合情况，在大家最终的排除头车的努力下以及周六小组集体调小车的情况下，我们组当天的成果是能够完整跑完所有的头车，但每个头车的跟车成绩还在 2 位数的量级，说明我们参数调整还有很大的进步空间。

周日的小组讨论中，我们集中精力调参数，最终把每个头车的跟车成绩都降到了个位数。令我们意外的是，在交作业的周一，突然增加了 B1 和 B2 头车，我们的程序一直跑不完 B1 头车，直到交作业的 40 分钟左右，我们才得出一个能够跑完所有赛道的程序，可是成绩并不好，为了跑完 B1 而不出现 999 的情况，每个头车的跟车成绩都上升至 2 位数，这直接导致我们在第一次作业的成绩排在了倒数几名。

以下是我们第一次作业各个头车的成绩

头车号	10	15	23	25	31	B1	B2	Total
成绩	24.142	20.586	15.162	20.738	19.961	38.013	45.526	184.128

当看到其他组的第一次作业成绩时，发现我们小组和其他小组的成绩根本不是一个等级的，于是我们意识到我们组的程序肯定在某个方面有很大的问题，在输出数据观察 B1 赛道的过程中，我们发现前车的加速度数值比较不稳定，甚至会出现几百的情况，于是在资格赛作业的一开始，我们就致力于改变头车速度及加速度的计算方法，最终发现是我们第一次计算时考虑的太多，用到了坐标变换等方法反而不如直接简单计算来的准确。

在得到较为准确的前车速度及加速度之后，我们再根据此时的情况进行参数的调整以及一些特殊情况（如急刹保护，过弯控制等）的纳入考虑，总成绩已经提升了许多，但和排在前面的小组还是有差距，尤其是 B1 头车，我们跟的还不是很好。在调整 B1 头车的过程中，我们遇到了很大的困难，B1 头车的特点是高于 200km/h 超高速行驶，加速度极大以及超高速急刹，这一方面使我们在头车加速的时候由于跟车加速的滞后性跟不上，另一方面超高速急刹使得我们不敢跟的太近以至于 damage too large。所以有很长一段时间我们处在要么就是把 B1 的成绩降了下来，其他头车的成绩又纷纷提高甚至跑不完，要么就是其他头车的成绩有普遍提升，而 B1 头车跑不完。对于 B1 头车和其他头车唱反调的现象，我们决定将 B1 头车和其他头车分开，对 B1 进行专门的优化及参数调整，在一次偶然的尝试中，我们把巡线的 best cruise 的档位复制到跟车的 cpp 中，发现 B1 的成绩提高了一些，于是我们又针对性的调整了档位变换的数据，使得跟不上 B1 的问题改善了一些，这样我们的成绩又进步了一些。

在周一晚，我们组还存在一个问题，那就是跟车跟得不够紧，导致 y 方向的误差会比较大，对于这个情况，一开始只是想当然地增大(_Leader_Y - 10)在 acc 中的权重，但是常出现 distance is too small 或着 damage is too large 的情况，后来，我们发现了某些头车具有一些

共同点，比如 23/31 号头车的速度始终没有超过 80 且行驶较稳定，15 号头车虽然高速行驶但几乎没有急刹情况，在提交作业的前 1 个小时我们根据不同赛道的特点进行了(_Leader_Y - 10)权重的相应调整，最终使成绩又进步了一些。

以下是我们资格赛的成绩：

头车号	10	15	23	31	B1	B2	B3	Total
成绩	1.777	2.605	1.665	1.627	6.017	3.611	4.029	21.331

可以看出，虽然我们对 B1 赛道进行了针对性的调整，但成绩还是不理想。

在比赛过程中，我们组的小车表现较好，在未知 X 头车的成绩比排在我们前一名的小组要好一些，无奈成绩计算方式对我们不利，最终小组排名没有改变，排在第七。

七、讨论、心得

在刚接触到跟车作业时，面对空白的模板代码我们感到无从下手，可是当我们真正开始着手于问题的解决时，发现其实并没有想象中的那么困难，只要我们有认真去思考就行。

第一，将任务分块很重要。面对一个整体的大问题时，个人的能力不太足够，我们把整个跟车程序分成前车速度加速度的获取模块，速度加速度控制模块和转向控制模块使得这个大问题细节化，并将每一块交给合适的同学负责，这样不仅有利于问题的简单化，也促进了整体的进展。

第二，抓问题的主要矛盾很重要。其实早在一开始，助教就提醒我们速度和加速度的控制是最主要的，可是在相当长的一段时间里，我们小组主主攻的方向是转向控制，当时转向确实还不够理想，最后在三个人的共同修改下转向部分已经做得相对较好了。可是由于在转向部分花费了太多的时间，导致我们小组的速度加速度控制部分做得很粗糙，没有经过完整的一轮三人共同调整。导致在交资格赛成绩的当晚，在加速度方面做了稍微调整，成绩都有进步，最后一波调整操作，我们组的成绩又进步了一些。但是，如果我们早意识到这个问题，有充足的时间花在速度加速度调整上，我相信我们组的成绩还会更进一步。

第三，具体问题具体分析，分门别类的思想也是很重要的。就拿 B1 来说，B1 一直是我们小组的成绩杀手，后期我们处在要么 B1 降不下来，要么 B1 降下来了，其他的道路又上升了许多，面对这种情况，一开始我们感到很绝望。后来想到既然 B1 一直和其他头车唱反调，说明 B1 的特性和其他头车很不相同，那么我们就不应该将他们看成一个整体进行调整，所以我们将 B1 特别针对性的识别了出来，并为其专门设置了一套适合它的代码，这样才使我们整体进程得以继续下去。同样，在其他的几个头车中，我们也发现了部分头车是具有类似的特性的，我们于是又将其归为一类进行具体化的调整，这同样使我们的成绩提高了。

第四，心态同样也是重要的。在第一周的作业成绩出来后，发现我们的成绩和其他小组的成绩根本不是一个数量级的，第二周时我们几乎是把原有的代码全部推翻重新开始，这样给了我们不少的时间压力。同时在 B1 这个主要矛盾没有解决时，整体成绩难以取得较大进步，整体进程几乎没有什么推进，当时三个人都相继地感到很无力，好在我们互相鼓励，基本上没有泄气的状态。感谢队友的全力投入~

上海交通大學 实验报告

4. 泊车实验

一、实验目的和要求
二、实验内容
三、实验工具
四、操作方法和实验步骤
五、算法框架与程序代码
六、实验结果与分析（必填）
七、讨论、心得

一、实验目的和要求

1. 掌握 CyberTORCS 仿真平台的使用
2. 理解新增的输入变量的确切含义
3. 设计自己的泊车控制算法

二、实验内容

1. 打开 wtorcs_new.exe，选择 CyberParking 菜单，泊车赛道为 Road Tracks 中的 CG Track3，单击“New Race”，观察最小样例程序的泊车过程。
2. 根据泊车规则设计算法。泊车规则：将车泊入绝对坐标为(lotX, lotY)的车位，成绩计算：泊车时间*（1+10*偏差距离/车长），以下情况泊车失败：
   - 泊车时有 damage 值大于 1 的碰撞，
   - 没有到达规定里程（即开始计时的里程）就置位 bFinished，
   - 置位 bFinished 时车体速度大于 0.2 km/h
   - 最后车体的航向角和提供的 lotAngle 之差超过 10 度
3. 用 Microsoft visual studio 2017 打开 driver_parking.sln，在 driver_parking.cpp 中的适当位置编写自己的跟车程序。

三、实验工具

CyberTORCS 仿真平台、Microsoft visual studio 2017

四、操作方法和实验步骤

参见软件说明书

五、算法框架与程序代码

泊车实验算法与巡线和跟车算法有本质的区别，主要体现在以下几点：

1、巡线与跟车的目的都是瞄准给定的目标前进，在输入变量中会给定行进方向的坐标值，两者有共通之处。而泊车不仅要做到车与车位的坐标重合，还要求车中心轴线与车位中心轴线的重合；

2、巡线与跟车的算法代码是一体式的，而泊车实验的代码是分阶段、分步骤的，不同阶段要完成的功能不同（比如向右大转弯阶段、倒车阶段、出库阶段等等），代码的结构也有很大的差异；

3、尽管在精度巡线实验中，我们强调既要“快”也要“准”，但是仔细研究巡线成绩计算公式可以发现，“准”才是影响成绩的主要因素，而巡线时间由于所占权重极低几乎不需要予以考虑。跟车更为简单：只需要考虑如何使两车距离变得更小即可。所以巡线和跟车几乎只考虑单影响因素。但是在泊车实验中，“快”和“准”是两个同样重要的方面，要想取得好成绩，两方面必须同时着手调整，而且难点是这两个方面似乎存在着些许对立，难以兼顾。

综上所述，泊车实验的算法有着特殊性，接下来将介绍我们小组的基本算法框架和部分实现代码。

算法框架方面，我们小组采取倒车入库的方式，整个泊车过程分为五个阶段：

第一阶段：巡线，对应程序中 flag=0 的状态；

第二阶段：距车位一定距离时靠左行驶，对应程序中 flag=1 的状态；

第三阶段：接近车位时向右大转弯，对应程序中 flag=2 的状态；

第四阶段：倒车入库，对应程序中 flag=3 的状态；

第五阶段：漂移出库完成比赛，对应程序中 flag=4 的状态。

每一阶段执行时都不断判断跳转条件，当跳转条件满足时程序跳转至下一状态。例如在第三阶段向右大转弯时，车必然有一个短暂的停车过程，于是我们规定当速度小于 1 即可认定为转弯完成已经停车，程序即跳转至第四阶段倒车入库。实现代码举例如下：

```
if (_speed < 1 && flag == 2)   flag = 3;
```

具体实现代码方面，我将按不同阶段分别介绍。

首先是一些基本函数功能的介绍，在实验中只给定了车和车位在地图坐标系下的绝对坐标，这对于车辆的控制上带来了诸多不便，因此我们还要根据绝对坐标以及车位的朝向，计算出以下两个坐标：

1、车在车位坐标系下的坐标（车位中心点为坐标原点，车位中心线指向车道的方向为 y 轴正方向，x 轴方向指向车开来的方向）；

2、绝对坐标系下任意一点在车坐标系下的坐标（以车为坐标原点的坐标系，即输入绝对坐标，输出该绝对坐标在车坐标系下的坐标）

为实现上述第一个功能，我们编写代码如下：

```
rel_X = _carX * cos(_lotAngle) + _carY * sin(_lotAngle);
rel_Y = _carY * cos(_lotAngle) - _carX * sin(_lotAngle);
```

```
transx = _lotX * cos(_lotAngle) + _lotY * sin(_lotAngle);
transy = _lotY * cos(_lotAngle) - _lotX * sin(_lotAngle);
rel_X -= transx;   rel_Y -= transy;
temp = rel_X;
rel_X = -rel_Y;   rel_Y = temp;
```

其中rel_X和rel_Y即为车在车位坐标系下的x坐标和y坐标。上述第二个功能实现代码与上面相似，在此不再赘述。

在第一阶段（即flag=0）时，由于助教给我们的初始程序已经可以完成此功能，故我们未作改动，具体代码也与初始巡线代码完全相同。

在第二阶段（即flag=1）时，车靠左行驶，只需要将行车的预瞄点调整至车道靠左侧即可，其余方向控制代码与阶段一的代码也完全相同。在此阶段中，我们使车辆进行了适度的加速，使其以比较快的速度进入计时区，以尽可能缩短泊车时间，速度与到车位的x轴距离有关，即：

```
if (_speed > rel_X * rel_X / 3)
   { *cmdBrake = 0.6; *cmdGear = 1; *cmdAcc = 0; }
else
   { *cmdBrake = 0; *cmdGear = 1; *cmdAcc = 0.15; }
```

在第三阶段（即flag=2）时，车辆向右转弯，为倒车作准备，向右转弯的目标是尽可能与车位中心线靠近且车头朝向尽可能与车位中心线平行。为实现此功能，我们使车瞄准车位前方40个单位长度的点进行转向，由于这个点离车位较远，故当车转向完成时，车就能基本做完成上述的转弯目的。具体实现代码如下（absolute_X与absolute_Y是预瞄点的绝对坐标，通过前面所说的坐标变换可以输出这个点在车坐标系下的坐标，分别是relcar_X和relcar_Y）：

```
absolute_X = _lotX + 40 * cos(_lotAngle);
absolute_Y = _lotY + 40 * sin(_lotAngle);
D_err = -0.5 * atan2(relcar_X, relcar_Y);
```

我们在向右转弯时还有相应的保护机制，以防止转弯过度和转弯不足，一是在转弯不足（即快靠近车位中心线时车头与中心线夹角仍较大）时加大D_err的值，具体代码如下：

```
if (rel_X < 4 && fabs(_caryaw - _lotAngle) > 0.17)
   D_err *= 2 * (3 - rel_X / 2);
```

二是转弯过度（即车头与中心线夹角已经几乎为零但是离车位中心线仍有较大距离）时减小D_err的绝对数值，具体代码如下：

```
if (fabs(_caryaw - _lotAngle) < 0.5 && fabs(rel_X) > 0.4)
   D_err += rel_X / 2;
```

在速度控制上我们规定当车头与车位中心线夹角基本消除且车此时距离中心线距离较小时满足停车条件，车辆刹车，否则车辆速度与到车位的距离成负相关。

```
if (fabs(_caryaw - _lotAngle) > 0.27 || rel_X > 0.4)
   {
      if (_speed > 36 - rel_Y)
         { *cmdBrake = 0.25; *cmdGear = 1; *cmdAcc = 0; }
```

```
            else
                { *cmdBrake = 0; *cmdGear = 1; *cmdAcc = 0.11; }
        }
    else
        { *cmdBrake = 0.85; *cmdGear = 1; *cmdAcc = 0; *cmdSteer = 0; }
```

在第四阶段（即flag=3）时，车辆开始倒车，预瞄点是车位后方3个单位的位置，速度与距离车位的距离成正相关。由于方向控制和速度控制与阶段三有很大的相似性，故不再赘述。另外我们为了防止在接近停车位时车辆抖动的问题，特别添加了如下代码，效果明显：

```
if ((_caryaw - _lotAngle) < 0.085)  D_err /= 3;
```

在第五阶段（即flag=4）时，车辆出库完成比赛，由于error的计算已经完成，此阶段用时越少越好。我们采用了漂移的方法，首先（flagout=0）向左打死方向并将油门踩到底，车辆会迅速出库并伴有车尾逆时针漂移；随后在车辆与道路中心线夹角小于一定值的时候（flagout=1）向右打死方向盘以修正漂移甩尾，同时使油门适度调低；最后当车辆角速度降低到一定值之后（flagout=2）即可认为漂移完成，此时再将油门踩到底并使车辆沿道路中心线行驶，比赛完成。具体代码如下：

```
if (flagout == 0)
  {
      *cmdAcc = 1; *cmdSteer = 1; *cmdBrake = 0;
      if (abs(_yaw) < 0.5)  flagout = 1;
  }
else if (flagout == 1)
  {
      *cmdAcc = 0.5; *cmdSteer = -1; *cmdBrake = 0;
      if (fabs(_yawrate) < 0.1)  flagout = 2;
  }
else if (flagout == 2)
  {
      *cmdAcc = 1;
       *cmdSteer = -4 * atan2(_midline[30][0], _midline[30][1])) / 3.14;
      *cmdBrake = 0;
  }
```

上面详细介绍了我们组的泊车算法与相应的主要代码，其中有很多不完善的地方，有些算法甚至比较低级，在接下来的分析中将介绍算法的一些不足和我们调试过程中遇到的困难。

六、实验结果与分析

依据我们的代码测得 1~5 号赛道的总用时、误差、总成绩如下表所示：

赛道号	1#	2#	3#	4#	5#
Time	8:35	8:38	8:60	9:39	8:64
Error	0.0228	0.0310	0.0234	0.0209	0.0109
Score	8.749	8.910	9.011	9.794	8.831

分析上面的结果可以发现，五条赛道的泊车精度都相对较高，成绩之所以还不甚理想，与泊车用时过长有较大的关系，因此要想显著提高我们的成绩，减少停车用时是避不开的话题。我们在实验过程中与其它小组也有交流，发现他们的 Error 与我们差不多，但是他们的泊车时间可以控制在 7 左右，因此他们的总成绩比我们优秀很多。意识到主要问题之后，我们开始在减少停车用时上下功夫。

减少泊车用时有两种基本的思路，一是通过提升速度减少用时，这是比较常规且“看似”可行的方法；二是改变车辆的行车轨迹，比较直接地讲就是漂移。

一开始我们考虑到漂移比较难以控制，遂首先尝试第一种方法。但是我们发现，在粗暴提升速度的时候，Error 会有爆炸式的增长，停车时间好不容易下降 0.1，误差就会增大 0.1，反映到具体成绩上不降反升。

随后我们开始尝试通过漂移的方法减少泊车用时，但是遇到了之前设想到的困难，我们难以控制车辆的漂移精度，车辆常常是漂移过度或漂移不足，导致车辆在倒车时调整不及时以至于 Error 过大。我们在询问其它小组时得知漂移入库是可以实现的，但是我们发现自己并不能很好地驾驭，于是便很快放弃这种做法。放弃漂移后我们又转向了通过调整速度减少用时的老道路上，直到提交成绩时仍然收效甚微。

赛后我们小组进行了一些交流与分析，总结了泊车阶段的几个不足之处：

1、在刚开始编写整个代码时忽视了时间对于总成绩的重要性，整个代码成型之后才发现时间过长，而此时再来调整相关参数已经变得非常困难；

2、我们一致认为在倒车阶段我们对车辆的转向控制是非常薄弱的，主要体现在：车辆右转结束时如果有较大偏差，倒车阶段修正偏差的能力就会非常弱，我们不得不通过尽量提高右转阶段的精度来保证倒车时比较精确地进入停车位。而且倒车阶段较弱的误差修正能力也会直接影响我们的倒车精度，这也是我们稍微提高车辆速度时 Error 会猛增的原因；

3、在倒车阶段车辆即将进入停车位时，车身会有剧烈的抖动，主要原因是我们瞄准了车位后方的一个固定点进行倒车，在调整方向的过程中会出现超调的现象，这也是我们误差的主要来源，我们一开始想过用 PID 来减少车身抖动，但是发现收效不太明显（主要是我们在巡线和跟车时对 PID 留下了不好的印象，每次加上 PID 之后成绩都会变差），所以我们最后不得已使用了一行低级的代码，直接将 D_err 的值除以 3，以减小抖动的剧烈程度；

4、没有掌握漂移技巧是我们整个实验中非常遗憾的一点，我们其实非常期待在泊车时实现炫酷的漂移。在调整代码的过程中我们没有真正掌握到漂移所需要满足的条件，比赛后听其它小组介绍算法时才明白些许。当然想漂移仅限 CyberTORCS，在以后的实际驾驶中我们一定遵纪守法，做一个好公民，老师和助教大可放心。

七、讨论、心得

泊车实验可谓一波三折。第一周作业提交时我们小组排名第六，这个排名使我们用“血汗”好不容易换来的。第一周我们在图书馆小组学习室总共约了四次，每次接近四个小时，前两次无论我们怎么改算法、调参数，总有一两个车位完成不了，心情崩溃、心态爆炸。在第三次约图书馆时我们仔细静下心来，几乎把代码重新写了一遍，也多亏数学强无敌的赵寅杰写了一段坐标转换代码，我们突然发现小车能够把所有赛道都跑完了，虽然总成绩仍然在400以上，但是我们总算能够跑完了！这是一个标志性事件！这个事件必将名垂史册！而且我们以敏锐的直觉和丰富的经验立即察觉到，调参数的阶段终于到来了！激动的心情伴随着不断颤抖的双手，我们齐心协力调整参数，总成绩也飞速下降，直到第一周提交作业时的50左右。我们用仅仅八个小时的时间实现了华丽的逆转，从4995的置底成绩跃升至第六，终于舒了一口气。

可能是第一周把所有的运气消耗殆尽，也可能是较好的成绩造成了心理上的一丝丝懈怠，我们在第二周调整参数时阻碍重重，成绩停滞不前。上面的成绩分析提到，我们意识到了泊车时间过长是我们成绩不好的主要因素，我们也曾想过右转阶段的代码通过漂移实现，但是在初步尝试之后我们发现难以控制，以为漂移是一种可行性不高的方法，遂放弃。但是在比赛时我们才发现，很多组都采用了漂移右转的方法且实现非常完美，这也是他们成绩优于我们的主要方面。我们发现好的创意必须还要有持之以恒的毅力，这也给我们上了非常深刻的一课。

我们在成绩始终无法进步的时候，想到了针对每一个特定赛道调参数的方法。因为在泊车实验中，由于车位的位置不同，车辆从开始行进到开始泊车所用的时间也一定不同，因此对每个泊车位的参数调整不会影响到其他泊车位的成绩，也不会造成比赛时X赛道“翻车”的情况。对每条赛道设置特定的参数过后，总成绩数值下降了5左右，效果比较显著。

在提交泊车资格赛成绩时，我们从上个星期的第6名降到了第13名，在比赛中又降到了第15名，泊车是我们三次比赛中成绩最差的一个，这也给了我们很多教训：

1、这门课的名字中含有“创新”二字，说明我们需要创新，而且创新是我们取得好成绩的必要途径。有时候我们宁愿花大量的时间调整参数也不愿意修改泊车的方式与算法，这种惰性是病，得治。泊车比赛的时候看了其他小组的泊车方式，有些小组的泊车非常惊艳，也让我们体会到了创新的魅力所在；

2、与其他小组的合理沟通也是必要的，虽然不同小组之间存在着竞争，但是合理的沟通有助于打开思路，吸取对方的长处，最后能达成共赢；

3、思想上的懈怠非常可怕，我们小组巡线和跟车的成绩不错，还有幸取得了坐无人小车的资格，于是做泊车时就明显感觉开课之初的激情不再，思想上有些放松，泊车实验让我们看清楚了思想懈怠的后果；

4、在代码苦苦求之不得的时候不如暂时清空记忆，躺在椅背上闭上眼睛放松一分钟。有时候五行代码的功能可能强于五十行代码，清醒的头脑和注意力的集中很重要。

总的来说，泊车实验是这门课给我们最深痛的教训，也让我们意识到工程实践中创新、坚持、逻辑思维的重要性，我们以后会在实践中吸取这些教训，争取做得更好。

5. 小组分工、个人心得体会与课程建议

5.1 小组分工

严威豪：跟车 PPT、上台展示、日常调参、订小组自习室、撰写跟车报告
刘启明：泊车 PPT、日常调参、订小组自习室、撰写泊车报告及汇总修改
赵寅杰：巡线 PPT、日常调参、撰写巡线报告及小组分工部分

5.2 个人心得体会与课程建议

组员一（严威豪）：

在选课的时候就听学长说过工科创4D是一门很花时间的课程，在每一个有作业的周末，我真实地感受到了什么叫做“很花时间”（可还是要保持微笑）。在组队的时候，有幸能和两位非常 nice 的同学一起，明明是本组最菜却莫名其妙地成了组长，随着一次次小组讨论的进行，小组内的默契度越来越好。

在巡线实验中，我主要负责 dirt 道路的识别及 dirt 优化，参与了其他道路参数的调整以及方向控制部分。巡线实验助教的代码已经给出了较为完整的实现方式，整体难度不大，在加入了一些小组自己的想法以及优化后，资格赛成绩还算不错，排在第四，与第三的差距主要在 dirt 部分，其他道路我们表现还是更好的，所以在面对 X 赛道时我们组是有优势的，不得不说第一次比赛现场是最紧张刺激的，当看到我们小组的小车登场时心里那叫一个跌宕起伏，看上去感觉并没有表现的很好可是跑完的成绩却出乎我们意料的好，最终我们小车的惊艳表现让我们反超成为第三，内心狂喜的一晚。

在跟车实验中，我主要负责转向控制，参与了速度加速度控制及参数优化。跟车实验中的代码模板比较空，也导致我们一开始无从下手。在分配了每个人的任务之后，我们整体上终于有一些进展了，一切都有条不紊地发展着直到邪恶的 B1 头车出现了（我觉得一定是排头车到石乐志的助教的意图），第一次作业 ddl 当晚我们一直处在很崩的状态，当晚的小组自习室充满了我们三人间歇性叹息哀嚎，一直没有出现一个跑完所有头车的版本，后来我弄出了一个成绩非常差的版本作为交作业备用，最终我们还是交了这个成绩非常差的版本，看着其他小组都是 20+的成绩而我们组却是 100+的成绩，心里很不是滋味（有些游戏还没开始就已经输了 orz）。第二周我们重新开始，在我们的共同努力之下最终成绩排在第七还过得去，在几乎重新开始的前提下进步很大了。

泊车实验时，我主要负责倒车部分，参与了参数调整。泊车实验我们进行的并不顺利，感觉主要问题还是出现在倒车上，所以可以说主要责任在我身上，很是愧疚。在泊车实验中我们有很多问题没有解决，首先时间没有降下来，在看其他组的比赛测试时，我发现我们自始至终都没有掌握好漂移的技巧；倒车做得精度不好，容易出现随机较大误差。而最终我们小组采取的是每条赛道分别识别来提高成绩的方法，没想到会被逼迫到这种不是办法的地步，最终成绩也不太好，总的来说泊车实验我们小组没有尽力更没有做好。

三次实验中的泊车实验成绩最差，我觉得自己有较大的责任，第一，作为组长没有很好的调动大家的积极性。发现大家到了后期对调小车都是抵触情绪，也包括我。这时，虽然我们的成绩还是很不好看，还有很多地方需要改进，但是我们最后一个周末并没有花更多的时间在小车上，反而宁愿看其他科目的书也不愿调小车。第二，自己周五有 PRP 的讨论会而资料还有很多没看，少花时间在小车上也不好意思叫另外两位调小车，所以放任两位干自己的事。现在想起来最后一个周末没有充分利用导致成绩不理想真的是很遗憾。

总的来说，我觉得我们小组有一个很大的问题就是不愿意花时间去阅读文献，几乎都是凭借自己的想法去编写代码。而后期也出现了懒惰的现象：宁愿埋头调参数也不愿修改算法，

说到底还是不勤于思考。

在课程建议方面，有以下几点个人看法。

第一，部分比赛成绩比例不合适。我觉得这门课应该将重点放置于算法，代码的通用性，即未知赛道的比例要相对高一些，这一点跟车比赛做得特别不好，平时成绩占 0.8，比赛成绩占 0.2，只要平时有针对性的对各赛道进行优化，总成绩降下比其他小组少 1，而其他小组若想翻盘比赛时必须要少 4，这几乎是不可能的，当时在调跟车时我想到了这一点但没有说出来，我还是觉得通用性更重要，让大家往通用性的方向走更正确。

第二，前几名讲解时代码透明度不高。在被小车折磨了一番之后，大家肯定很希望知道前几名尤其是前三的同学是如何做到这么好的，可是当我集中精气神准备接受大佬们的教诲时，发先前几名的同学的 pre 中大部分是用于讲解心路历程的，而不是讲解算法程序，课堂时间有限展示心路历程是可以的，但也希望在课后能够把前三的 cpp 公开，让我们可以学习学习，如果担心后届的学生从我们这里得到代码，可以在最后要求组长核查队员在课程结束后是否删除了 cpp（其实也觉得没有什么必要，大家都挺自觉的）。

第三，课堂时间安排。大家都有这样的体会，在比赛时，大家相对更集中精神；在小组 pre 时，大家除了前几名几乎都在做自己的事，我觉得可以减少上台 pre 的小组数，增加前几名同学 pre 的内容，尤其是前三的同学，既然得到了奖励，pre 也可以要求他们多做一些内容，也可同时解决第二个问题。

第四，奖励名额设置。三次比赛，只有前三名能得到奖励，此处我建议再设置一个奖励项目，每次排名后根据排名小组得到对应的积分，比如第一名 20 分，第二名 19.7 分等，最终三次比赛结束再对三次小组总积分进行一次排名，如果想扩大奖励名额，则规定已获得的小组不再在总积分排名中获得奖励名额进行顺差，这样可以鼓励同学们持续发力，而不是在某一次比较好之后不愿再认真完成任务（就像我们小组前两次比赛比较好，第三次就没有了热情），以及让平均成绩都很好但就是没有进入前三的同学也是怀有希望的。

第五，平时作业建议分配小比例成绩。觉得每一份努力都值得肯定，即使是平时作业做得好的也应该奖励，并且这样可以起到促进同学们重视平时作业，多花时间的目的，也可以让下一次的起点高一些，减少从头开始调小车以及同学们隐藏真实成绩的心机大戏现象。

第六，新比赛模式的建议。1.撞墙（鬼畜娱乐）模式：成绩指标有 total time 和 damage 两项，给定几条赛道，在每条赛道的规定区间内，使 damage 尽可能大的同时（被 damage is too large 支配的恐惧，我只想潇潇洒洒的撞墙~撞墙~撞墙~），使 total time 尽可能小，当然这里面的成绩比例调配还是需要思考的。2.事故安全模式：成绩指标为 damage 和 total time 两项，每条赛道都有一个容易出事故的道路区间，在开始的一段时间内先规定所有小车必须按相同的容易出问题的行车方式行驶（比如高速撞墙，驶入草地沙地，360 度回旋，弯道外侧向外拐），在小车即将出事故时开始让同学们进行控制，damage 越小，时间越少越好，可以类似于头车，让各小组自设计开始的出事故控制。旨在让我们如何在危险情况保护自己以及尽早离开危险地段。

通过这门课，我也收获了很多。第一时和两位同学之间的交流更多了，懂得了团队合作及交流的重要性。第二，体验了一次小组长，懂得了在一个团队中组长的重要性，要能够调

动大家的积极性，要主动接活，要调整团体的心情状态，承担更多一份责任，我的表达能力不太好，在被 push 上台进行小组 pre 的时候心里还是蛮紧张的，但是感觉只要硬着头皮去做，也没有那么难。第三，认识到一些解决问题的意识的重要性：比如把大问题细分化，具体问题具体分析，分门别类的思想等在解决问题时有很大的用处。第四，面对难题时的心态要好，虽然在这门课三次实验中每一次都会碰到很多问题，也被压迫的失去了周末玩耍的时间，对着成绩无法提升的焦虑苦恼，内心一度惴惴不安，担心完不成任务，会拖小组的后退，但发现组员们都很好，会及时安慰自己不给彼此太大的压力，心态不常出现大崩的情况。

最后，感谢杨明老师开设这门课使我对控制有了初步的了解~感谢两位助教很热心的帮助以及解答我们的问题~感谢二位组员整个过程都非常认真，为提高成绩花了很多时间~

组员二（赵寅杰）：

这门课带给了我很多第一次——第一次上工科创，第一次接触无人车，第一次玩 TORCS，第一次调参到深夜，第一次小车比赛，第一次下载 vs，第一次使用加密狗，本来高中戒了但上大学以来又第一次吸维他柠檬茶。简言之，我不能说这门课是使我快乐的，但它的的确确是给予过我成就感的，除开知识方面的学习，它也让我愚笨的大脑有了一丝参禅似的、隐秘的不可言说的改变。

一开始是有一点点焦虑的，因为之前从来没有接触过类似形式的课，也没有学习过类似的东西，唯一学过的 C++也久远得像是上个世纪的事情。摸着石头过河总会让人有些恐慌，不过还好是三个人同时摸，好歹可以减轻一些痛苦，也让人更加充满了底气。团队分工从一开始时就不明确这不假，但大家都是朝着一个目标去的，即便走的路可能不同，但大家最后还是会汇聚到一点，如同平行的太阳光经过了凸透镜。

自己一直是急性子，有些毛毛躁躁，总希望结果按着我所设想的方向去，下场就是被小车一次又一次地打脸。我追求鲜花着锦，烈火烹油，可小车的表现，或者说是我自己总是让所有人失望。会感到灰心，可是看着另外两位依旧坚持，自己也不免强打起精神，继续在那一行行我的大脑已然拒绝阅读的代码中，寻找那隐藏在字里行间的不易察觉的契机。原本急躁的我，在小车面前只能煞住性子，真真是道高一尺，魔高一丈。然而在经历了几次这样令人难以忍受的时刻之后，我发现自己好像也可以沉得住气了，再不会像以往那样急躁了，像一只剥了壳的榴莲，拔了刺的豪猪。

第一次比赛的时候，早早地就把心脏提到了嗓子眼，可还要拿着一本书佯装镇定，双眼盯着方形的汉字，耳朵却是竖起来像一只警觉的兔子听着周围的风吹草动。看着一组组的成绩水落石出，内心更加不安。看到了别人的反超，但更害怕自己这组被反超。终于轮到我们组的时候，看着助教把我们的 dll 拖到相应位置时，突然想起了初中参加接力跑的时候父母去旁观的场景，一种为人父母的沧桑感涌上了心头，毕竟这辆车有三分之一流淌着我的血脉。还好最后结果很好，甚至可以说是出乎意料地进入了前三，按捺下内心的欢呼，眼神却像扑棱的麻雀一样出卖了我。

就这样一次，又一次，直到结束。到最后，我也不清楚，是我给小车更多，还是小车带给我更多。无数次，我看着小车在一条条赛道上不辞辛苦地奔波，一遍又一遍，撞了一次又

一次，可每次重新打开的时候，都仿佛是第一次被打开一样，保持着充足燃料，承载着经我染指的代码，义无反顾地撞了出去。以致到最后，我都不忍再折磨这辆朴实无华的小车，并不是我不想调参数。我用上帝视角观看着小车，心生一种全知神般的悲悯。

唯一美中不足的是，每次比赛之后的展示环节，前几名好像都没有怎么涉及代码的问题。每当我瞪大眼睛想要欣赏前三组同学的代码时，最后听到的总是心路历程。如果可以在比赛完成之后，把前三名的代码公布出来，让大家一睹为快，可能不失为一桩趣事。另外，我觉得可以在三次正式比赛之外穿插一些趣味赛，比如巡线速度+泊车，或者障碍赛、瘟疫赛什么的（脑洞来自《狂野飙车》），我觉得这样才能激发大家的兴趣和创造力，也能给最后的集锦增加无数素材。

总之，感谢你们付出的思想。以上。

组员三（刘启明）：

工科创 4D 结束了，让我说感想，只能说是百感交集。我还是逻辑一点，分点来答。

首先从这门课的课程体验来说。我有两位非常给力的队友，让我体会到了团队的力量，这也是我通过这门课获得的非知识性的最宝贵财富。课程期间我们每周都有大量的时间泡在图书馆，一起调参数，一起掉头发。我们有过鏖战十几小时仍然 999 的尴尬、无奈与焦急，也有过成绩飞升时的欣喜与希望。仍然记得跟车第一周我们排名倒数时的相互自嘲，也记得巡线比赛取得前三时的那种掩饰不住的狂喜，我们同时见证了小车的“成长”，同时经历了调代码的艰辛，因此同时分享着结果的喜或悲，这种感觉非常美妙。调完小车，我们时不时约着去吃夜宵，能聊小车，也能聊其他，我在此过程中也收获了友情。

除此之外，我也体会到了良性竞争给所有团队带来的进步。跟车时我们的思维陷入了瓶颈，车总是撞，直到快要提交跟车资格赛成绩时我们仍然完成不了所有头车。焦急万分，当我向黄嘉岚小组请教他们的思路的时候，他不仅给我详细讲了他们的实现方式，还给我展示了一些重要的代码，给了我们极大的帮助，在此也再次向他们表示感谢！在三次比赛课堂上，我也发现很多小组之间都有经常的交流，这种交流不仅使得大家开拓了视野，涌现了非常多的优秀程序，也使得工科创 4D 这门课变得更加有趣味性。希望 4D 能够把良性竞争的氛围继续维持下去。

其次我要说说这门课给我的知识。杨明老师的课堂永远不乏趣味，好的内容遇到好的老师，使得我对每一次的授课充满期待。课程条件所限，我们只能通过软件仿真来模拟整个无人车系统一小部分的实现，但是通过老师授课的方式，我仍然比较全面地了解了无人车的发展与相关的先进技术，获益颇多。通过三次课程实验，我了解了一些基本的控制方法，懂得了如何通过代码将输入到系统的信息处理为输出系统的控制信息，这些都是这门课给我的将理论知识用于实战的机会。

最后，我还想谈一谈我们在课程实践中还存在的一些问题。首先，我们小组三人虽然非常投入，花了大量的时间，但是有的时候工作效率并不太高。比如三个人一起调代码，可能一晚上只有一个人的代码有比较好的使用价值，那么另外两个人花的时间就会被浪费。所以在以后的团队中如何高效协作是一个非常重要的问题。第二，我觉得我们的思维还比较固化，

创新性还不够强。这一点特别体现在泊车比赛上，由于我们没有能够想到比较好的泊车路径，导致泊车时间过长，成绩不理想，仅仅流于调整参数。最后，我在这门课开始时特别想学习漂移算法，但是到课程结束时还没有学会，这是这门课里我非常遗憾的一点。

我还有一些与这门课有关的建议，可能比较尖锐，但是这是我的真心所想~~，希望助教小哥哥手下留情，不要打我~~。

第一，我认为每次比赛后的小组展示应该多一些干货，少谈一些感悟，上台演讲的小组数目可以压缩到十个甚至更少。每次我想学习排名靠前小组的算法介绍，但是他们似乎很多都以“很皮”的心路历程作为演讲的主要内容，令我有些许失望。减少上台演讲小组的数目也可以节约大量时间，这些时间可以被用来测试两条 X 赛道（我建议多测几条 X 赛道，下面将会讲到），也可以让老师多讲一些相关领域的前沿知识；

第二就是上面提到的测多条 X 赛道的问题，我认为鲁棒性是判断一个算法好与坏的重要方面。包括我们小组在内的很多小组，在某一条赛道成绩不理想时，首先想到的是针对它调参数，而不是修改更好的算法，我觉得这个导向是不对的。测试多条 X 赛道，不仅可以测试小组算法的普适性，还可以增大课堂比赛的刺激性和趣味性（大家比较喜欢看这个）。既然是比赛，我们也可以设计不同的比赛机制，比如前二十名测试 X-1 赛道，之后前十名测试 X-2 赛道，之后前三名测试 X-3 赛道，最终形成排名，这样才有“比赛”的味道。至于时间，我相信把演讲小组数量压缩至十个以内，是会有多余的时间的；

第三，我觉得巡线比赛中的成绩计算公式过度强调了精度，使得很多小组放慢了速度。成绩计算公式体现出速度与精度的矛盾性，适当增大难度。

第四点让我来展望一下：我觉得 CyberTORCS 还可以有很多玩法。比如双车竞速，漂移比赛等等，听起来有点荒诞，这仅仅是我的设想。

课程结束了，课程报告也以我感想的结束画上句号。感谢杨明老师、袁伟和郝继伟助教、两位队友和其他所有对我们提供过帮助的同学，也感谢努力奋斗的自己，谢谢你们！