机器人导论作业

视觉巡线小车

中山大学计算机学院　计算机科学与技术

19335174　施天予

目录

一、实验目标

巡线就是让小车沿着规定的轨道（通常是黑线）按照一定的速度进行移动。在本次实验中，需要为小车添加相机，根据图像信息，设计算法使小车能沿着地面的黑线行驶。在补充要求中，需要计算小车巡线一周的时间、平均速度、小车倒圆心的距离。

二、实验内容与步骤

1. 巡线原理

巡线小车能够巡线，基本原理就是区分黑线和白色地面，判断黑线处于什么位置，进而进行直行、左转、右转等操作。以红外传感器为例，巡线小车会搭载两对红外传感器，红外光线遇到物体时，就会形成反射的光线，而红外光线对于不同的物体反射特性是不一样的，对白色反光的物体，红外光线的反射量将会多一点。而对黑色不反光的物体，红外反射量将会大量的减少。那么我们就可利用这个特性来完成黑与白的判断。

视觉巡线也可以基于类似的原理。对于 camera 来说，需要通过求得获取 imagy 的灰度值判断小车所在的位置。如果灰度值过高说明在白色部分，如果灰度值过低说明在黑线上。这样，小车就能根据两个 camera 的灰度值不断调整方向，巡线移动。

2. 添加巡线地图

在 objects 的 floors 中新增一个 RectangleArena 的节点，将 rotation 的四个参数设为 1 0 0 1.57，再在 url 中设置图片`circle.png`的绝对路径，即可添加完成。

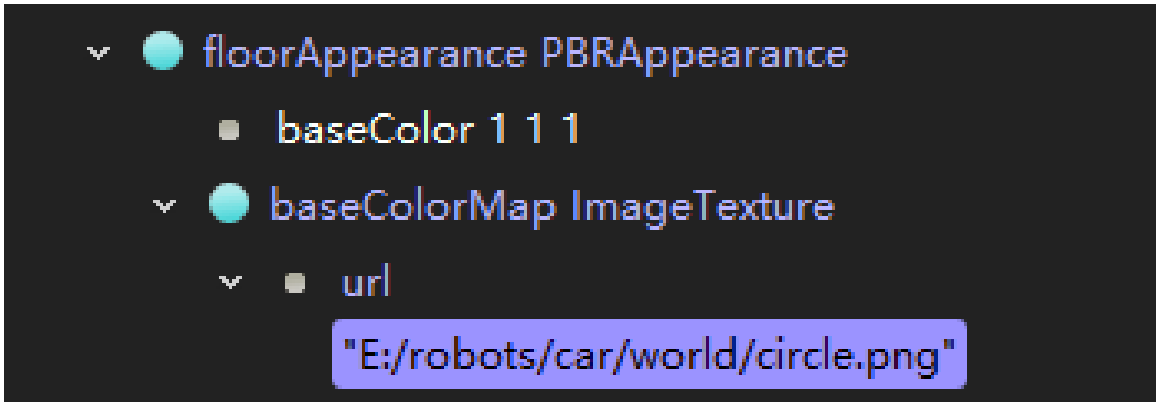

图 1: 添加巡线地图

3. 添加相机

这次的巡线小车可以在上次小车的基础上加上 camera 和 gps，即可完成模型的搭建。

1. 在 robot 的 children 下新建 camera 节点，修改其 name 为 $left\_camera$。
2. 调整 $left\_camera$ 的大小，然后将其移动到贴近小车的位置。

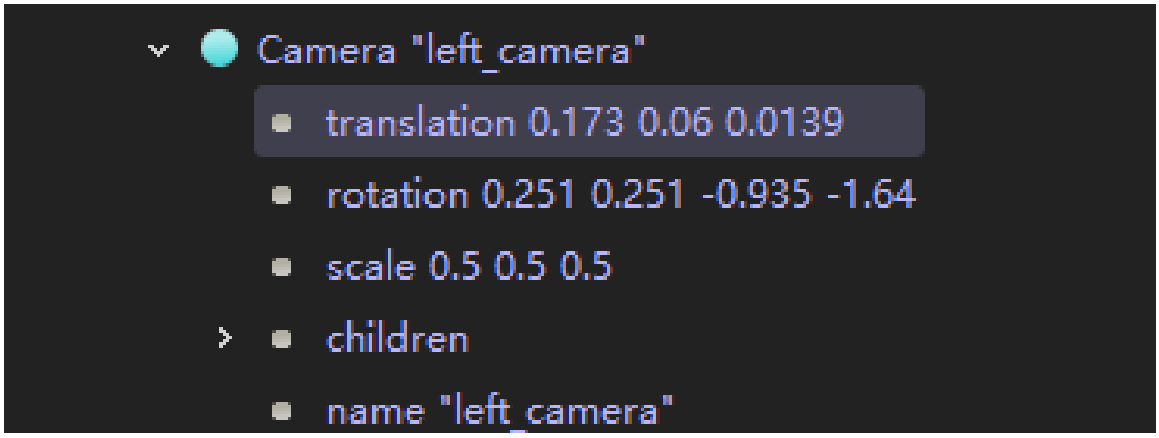

图 2: *left_camera* 的参数

3. 复制 *left_camera*，将新 camera 的 name 改为 *right_camera*，将其贴近小车。
4. 调整两个 camera 的间距，需要略大于黑线才能完成巡线。
5. 打开 View - Optional Rendering - Show Camera Frustums，显示相机的视角范围。旋转 camera，使其视角方向朝向地面。

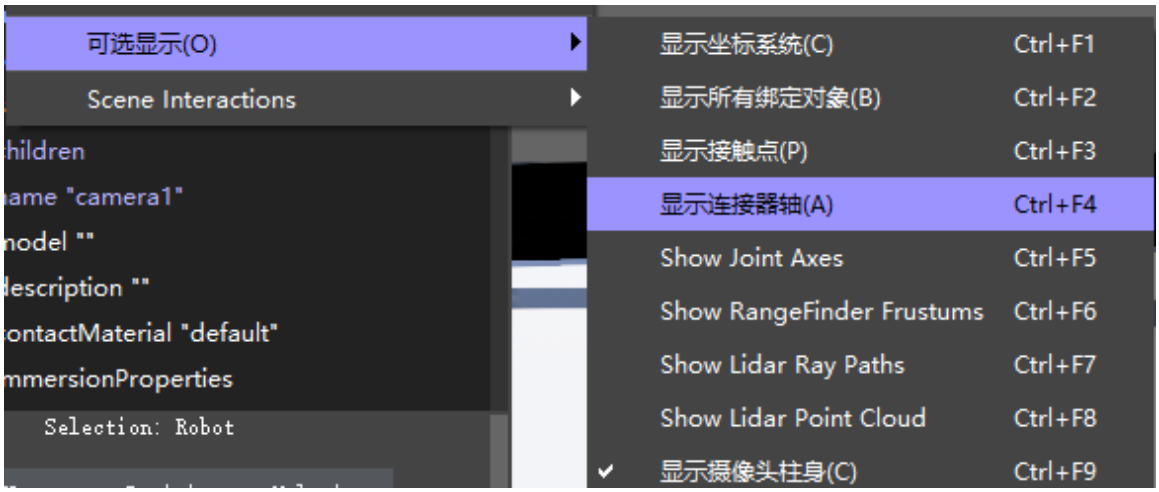

图 3: 显示 camera 视角范围

添加完相机的小车如图4所示

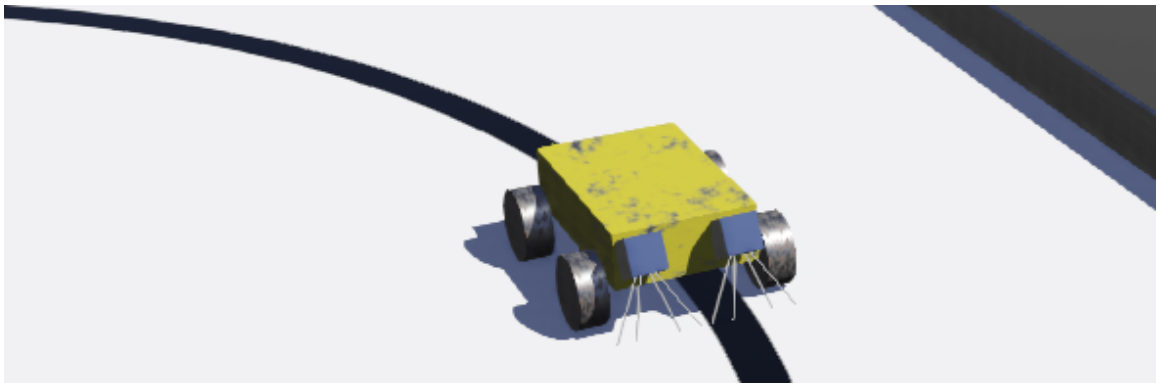

图 4: 添加完相机的小车

4. 添加 GPS

1. 在 robot 的 children 下新建 GPS 节点，修改其 name 为 *gps*。
2. 然后在 GPS 节点->children 中添加一个 solid 固件
3. 设置这个 solid 固件的 children 中添加 shape 节点，并设置外观和形状。
4. 将这个 GPS 移到小车正上方，具体参数如图5所示。

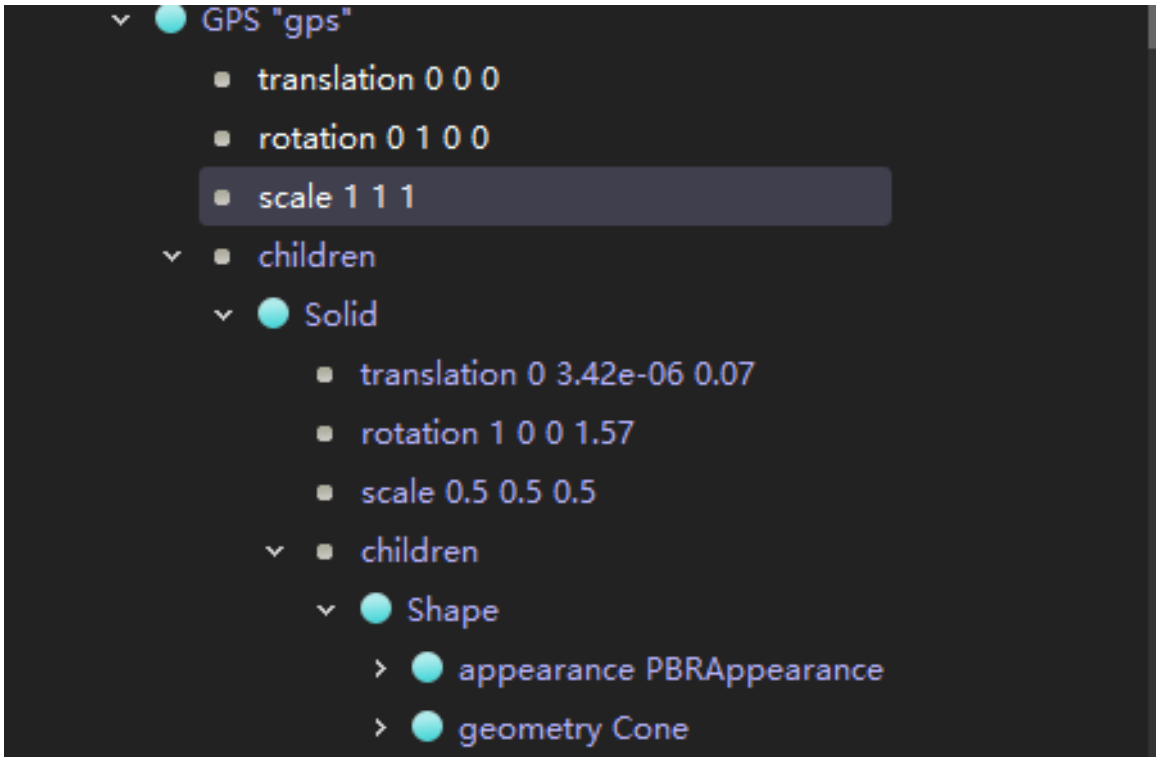

图 5: GPS 的参数

最终完成的小车如图6所示

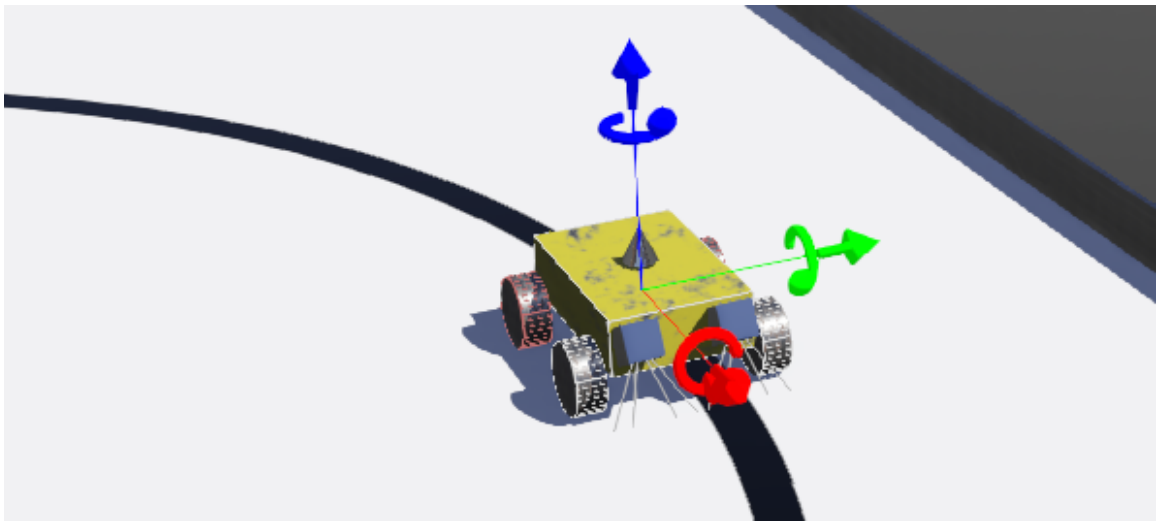

图 6: 最终的小车

5. 添加代码

首先用 *define* 定义速度值和灰度阈值。SPEED1 代表直行每个轮子的速度，SPEED2 代表旋转时内侧轮子的速度（比如往右偏那么右轮速度要小一点）。GRAY 代表灰度阈值，在后面会有对这个参数更详细的说明。

```
#define SPEED1 30
#define SPEED2 8
#define GRAY 120
```

初始化 robot、camera、gps、motor。将开始时左轮和右轮的速度初设为 SPEED1。step 代表后面 while 循环的次数，*average_speed* 代表小车平均速度，*average_position* 代表小车距离圆心的距离，posion 是小车的实时位置。

```
Robot *robot = new Robot();
int timeStep = (int)robot->getBasicTimeStep();
Camera *camera[2];
char camera_names[2][15] = {"left_camera", "right_camera"};
Motor *motors[4];
char wheels_names[4][10] = {"FR_motor", "FL_motor", "BL_motor", "BR_motor"};
GPS *gps = robot->getGPS("gps");
gps->enable(timeStep);

for (int i = 0; i < 2; i++) {
    camera[i] = robot->getCamera(camera_names[i]);
    camera[i]->enable(timeStep);
}
for (int i = 0; i < 4; i++) {
    motors[i] = robot->getMotor(wheels_names[i]);
```

```
        motors[i]->setPosition(std::numeric_limits<double>::infinity());
        motors[i]->setVelocity(0.0);
    }
    double left_speed = SPEED1;
    double right_speed = SPEED1;

    long step = 0;
    double average_speed = 0;
    double average_position = 0;
    const double *position = nullptr;
```

首先通过左右两个 camera 获取图像，并获取图像的宽和高。对于一个图像的每个像素点，需要通过 imageGetGray 函数获取灰度值，再记起来取平均求整个图像的灰度值。如果左右灰度值都高于或低于灰度阈值 GRAY，小车往前走；如果灰度值左高右低，小车右轮速度改为 SPEED2 往右偏；如果灰度值左低右高，小车左轮速度改为 SPEED2 往左偏。这样，小车巡线即可完成。

小车距离圆心的距离可以用欧氏距离计算，平均速度需要加上每一轮迭代的速度再除以 step 即可求得。

```
    // 根据左右相机的灰度值控制轮子的速度
    while (robot->step(timeStep) != -1) {
        step++;
        position = gps->getValues();
        // 计算小车距离圆心的距离
        average_position += sqrt(position[0] * position[0] + position[1] * position
            ↪ [1]);

        const unsigned char* left_image = camera[0]->getImage();
        const unsigned char* right_image = camera[1]->getImage();
        int height = camera[0]->getHeight();
        int width = camera[0]->getWidth();
        double count_lb = 0, count_rb = 0;
        for (int i = 0; i < width; ++i) {
            for (int j = 0; j < height; ++j) {
                // 获取左右相机每个像素点的灰度值
                int l_g = camera[0]->imageGetGray(left_image, width, i, j);
                int r_g = camera[1]->imageGetGray(right_image, width, i, j);
                count_lb += l_g;
                count_rb += r_g;
            }
        }
        // 加起来后要取平均
        count_lb /= width * height;
        count_rb /= width * height;
```

```
25
26      if ((count_lb <= GRAY && count_rb <= GRAY) || (count_lb > GRAY && count_rb
            ↪ > GRAY)) { // 左右都低(高)往前走
27          left_speed = SPEED1;
28          right_speed = SPEED1;
29      }
30      else if (count_lb <= GRAY && count_rb > GRAY) { // 左低右高往左偏
31          left_speed = SPEED2;
32          right_speed = SPEED1;
33      }
34      else if (count_lb > GRAY && count_rb <= GRAY) { // 左高右低往右偏
35          left_speed = SPEED1;
36          right_speed = SPEED2;
37      }
38      // 计算平均速度
39      average_speed = (average_speed * (step-1) + (left_speed + right_speed)/2) /
            ↪  step;
40      // 输出距离圆心距离、平均速度、左右相机的灰度值
41      printf("position: %lf, ", average_position / step);
42      printf("speed: %lf,", average_speed);
43      printf("left: %lf, right: %lf\n", count_lb, count_rb);
44
45      motors[0]->setVelocity(right_speed);
46      motors[1]->setVelocity(left_speed);
47      motors[2]->setVelocity(left_speed);
48      motors[3]->setVelocity(right_speed);
49  }
```

三、实验结果与分析

因为在补充要求前已经提交了实验报告，之前的两个地图的测试也做了就不删去了，仅供“欣赏”。最终的圆形地图一系列测试在第三部分。

1. 跑道地图测试

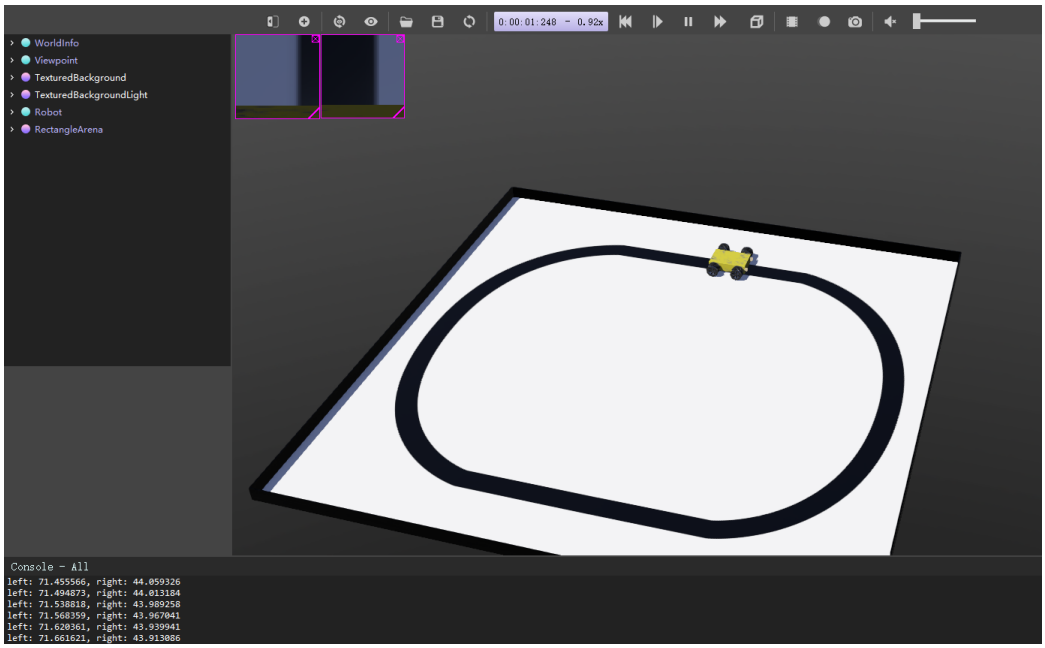

图 7: 跑道地图的直线

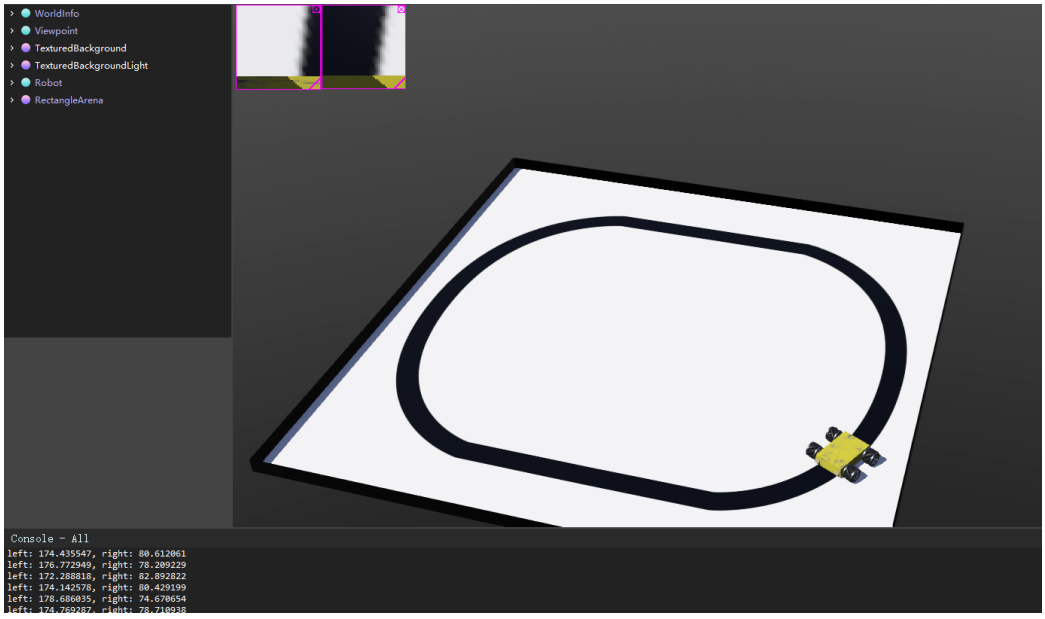

图 8: 跑道地图的弯道

在跑道地图上，可能是这个地图的黑线比较粗，小车在直线移动两个 camera 的灰度值都低于 120，所以保持前进。在弯道上，小车一边 camera 的灰度值高达 170 多，另一边低于 120，需要不断转向。

2. C 型地图测试

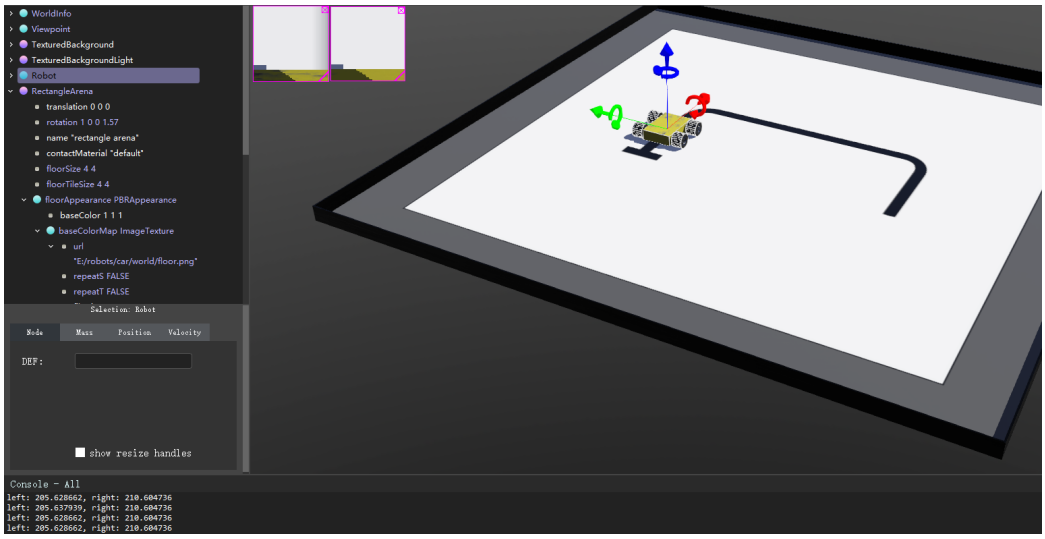

图 9: C 型地图的直线

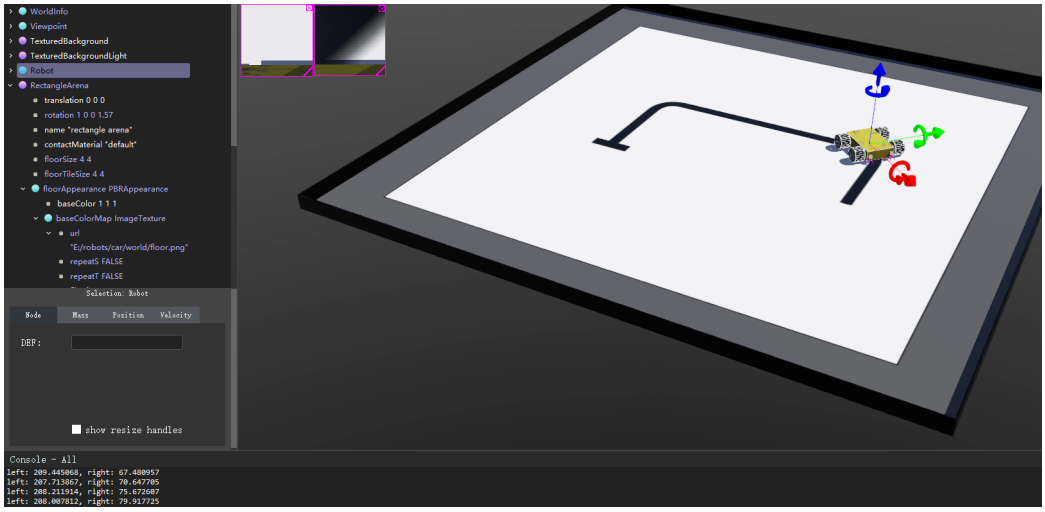

图 10: C 型地图的弯道

在 C 型地图上，因为黑线比较细，小车在直线移动两个 camera 的灰度值都高于 200，所以保持前进。在弯道上，小车一边 camera 的灰度值超过 200，另一边低于 120，向灰度值低的方向转向完成巡线。

3. 圆形地图测试

新加的 circle 地图，添加了计算小车距离圆心和平均速度的代码，可以发现小车运行一圈需要 9.72 秒，感觉已经是我的小车的最快速度，跟其他同学比还有一定差距。

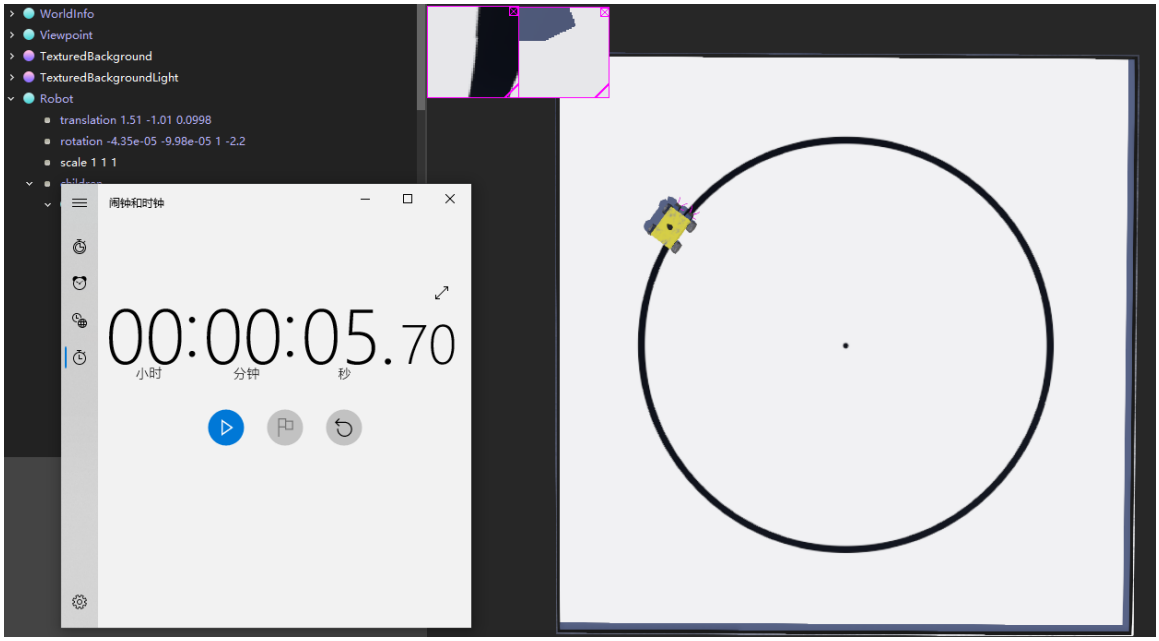

图 11: 圆形地图巡线中

图 12: 圆形地图巡线一圈

如图13所示，小车距离圆心的距离大概在 1.8 左右，平均速度大约 24 左右。根据灰度值与阈值 120 的比较不断改变内轮的速度。

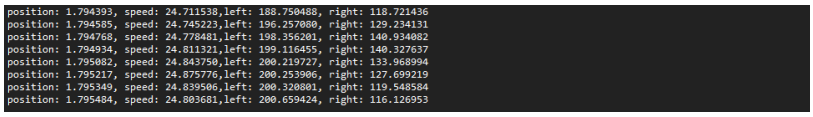

图 13: 小车运行时距圆心距离、平均速度、左右相机灰度值

四、实验中的问题和解决方法

这次实验没有了详细的 PPT 指导，花费了我大量时间去自己探索，但还是十分有趣的。实验过程中也遇到了许多问题，好在我查阅官方文档并请教同学，最后都一一解决了。不过因为上交的项目文件加载 `circle.png`路径会有不同，所以如果打开项目文件可能需要重新加载图片。

1. floor 呈现蓝色

原来我的巡线地图白色呈现蓝色，且 RGB 都已为 1，十分奇怪。后来通过将 floor 的 metalness 参数从 1 改成 0，就可以正常显示白色。

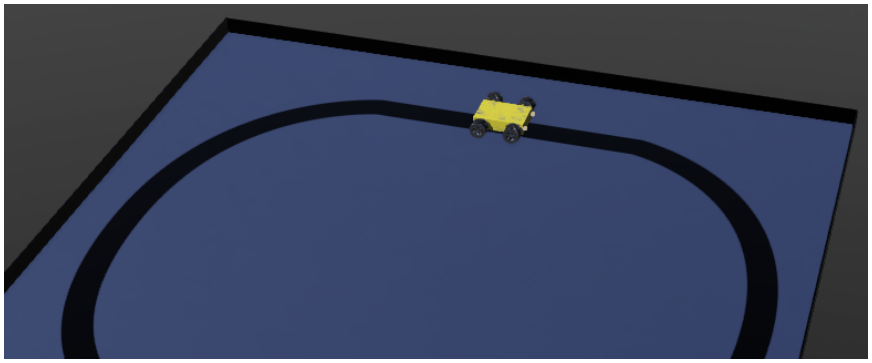

图 14: 蓝色的 floor

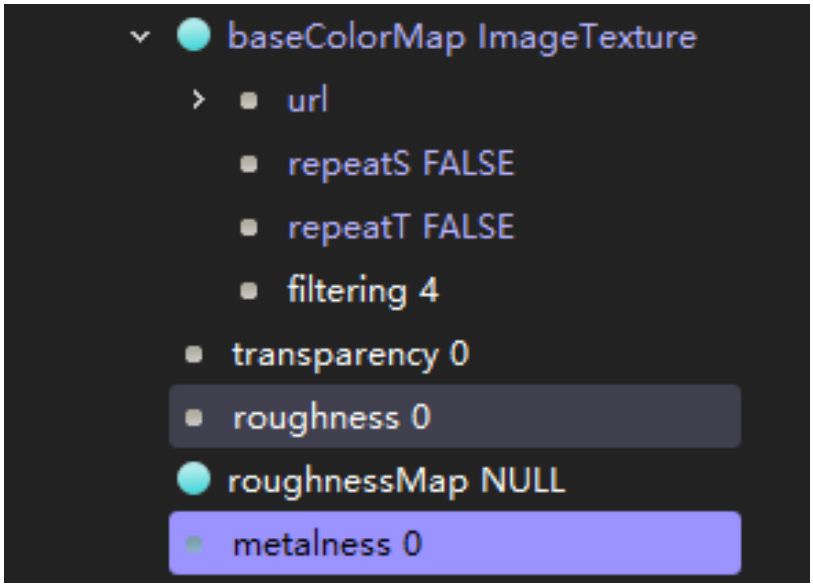

图 15: floor 的参数

2. 只显示单个 camera imagy 且方向不正确

发现是因为两个方框叠了起来，拖动一个方框即可使两个 camera 都能正常显示画面。对于画面显示方向不对的问题，需要将两个 camera 实体沿 z 轴旋转，找到正确的方向。

3. 小车在弯道颤抖不往前

刚开始我的代码写的是两个 camera 灰度值小于等于 60 则往前，有一方高于 60 就朝另一方向转，但这会使小车在弯道一直颤抖，移动十分缓慢。我发现有时候两个 camera 灰度值都很高超过阈值，所以添加了都超过阈值也往前走的代码。通过不断调参，我发现将阈值设为 120 比较合理，小车能平稳完成巡线。

4. 速度超过限制

一开始我的小车速度超过 14.81 就会报错，如图16所示。

```
WARNING: Robot > DEF WHEEL1 HingeJoint > RotationalMotor: The requested velocity 20 exceeds 'maxVelocity' = 14.81.
WARNING: Robot > DEF WHEEL3 HingeJoint > RotationalMotor: The requested velocity 20 exceeds 'maxVelocity' = 14.81.
```

图 16: 速度超过限制

后来发现将 4 个 motor 的 device 的 maxVelocity 设置为 100 就不会报错了

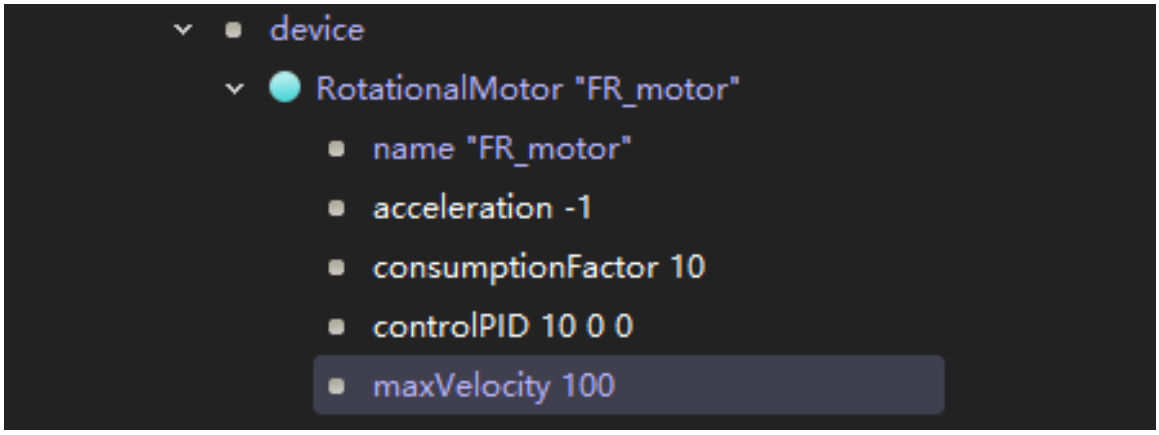

图 17: maxVelocity 设置为 100

5. 小车巡线漂移或飞出去，甚至散架

新加了补充要求希望尽可能提高小车的速度后，这次实验最麻烦的地方就是调参了。不光是灰度阈值 GRAY 的调参，SPEED1 和 SPEED2 的参数也很难调。当速度太大时，我的小车出现了漂移、飞出去，甚至还有散架的“惨象”。

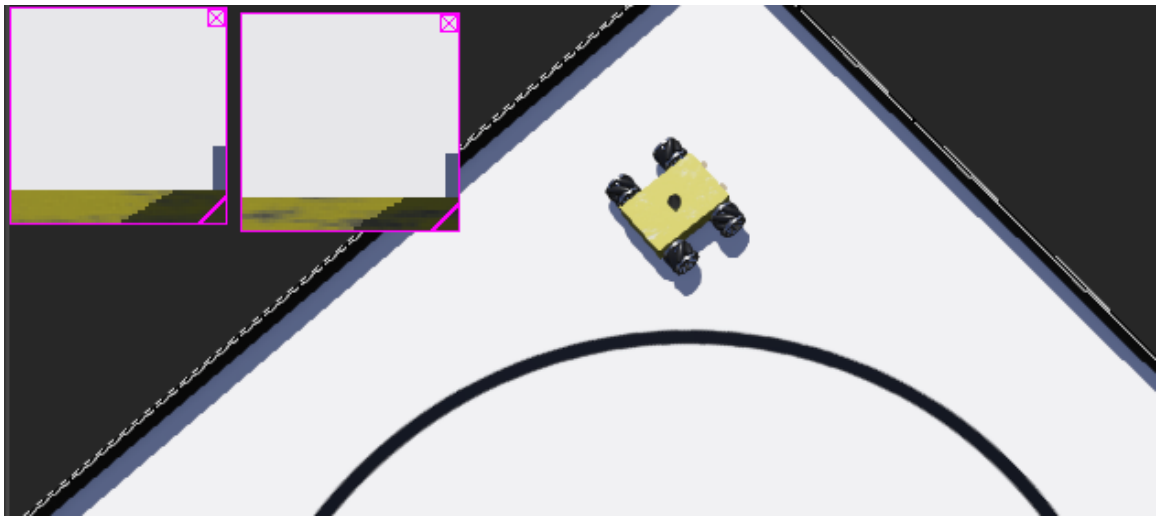

图 18: 小车飞出去了

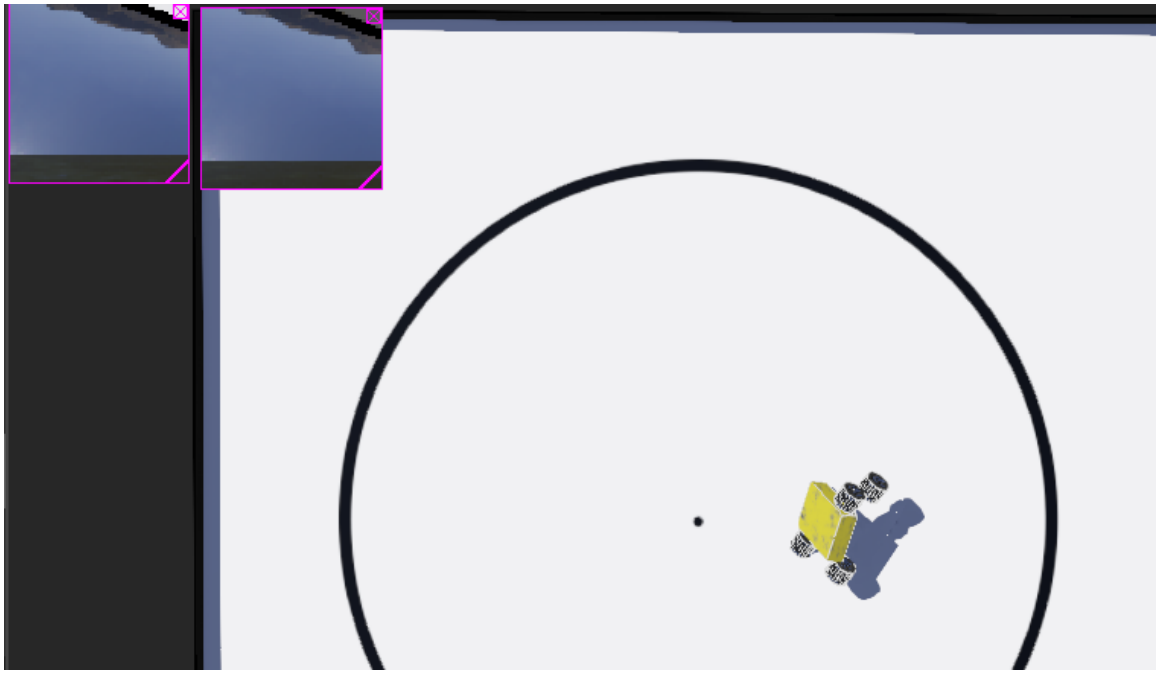

图 19: 小车散架

最后苦苦调参，发现将 SPEED1 设为 30，SPEED2 设为 8 是我的小车最快能完美巡线的速度。