机器人导论作业（二）

搭建麦克纳姆轮小车

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

使用 webots 仿真软件，搭建仿真的场景（地面、光照），搭建一辆麦克纳姆轮小车，然后写一个控制器，控制小车能够八向移动（前后左右斜）和自旋。

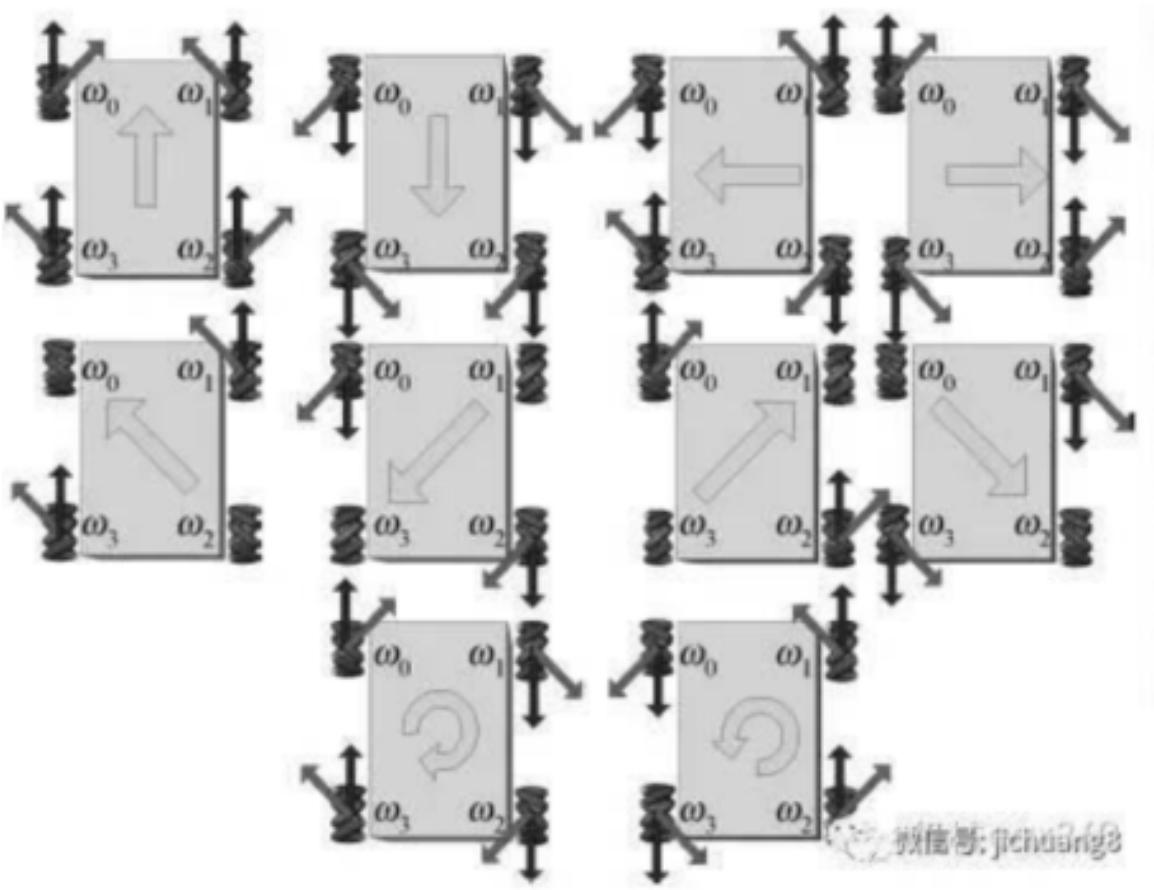


图 1: 麦克纳姆轮原理

二、实验内容与步骤

1. 搭建小车

因为小车的搭建 PPT 给出了详细的步骤，因此这里就简单概括一下，不再一一赘述

1. 修改世界坐标系为 ENU
2. 添加直射和漫反射光源，将视角调为 Back view
3. 添加地面，修改半径为 2
4. 添加机器人节点，在 children 下创建 shape 节点，改名为 body，设置车体大小 (x=0.3 y=0.2 z=0.08)m
5. 修改碰撞边界形状，同时添加物理属性

6. 按照助教程序中的参数将麦轮添加到自己的小车上

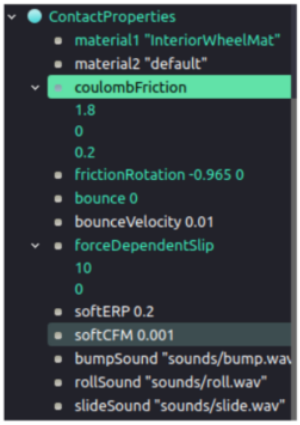
2. 设置摩擦材质

1. 添加两个`ContactProperties`节点，并通过此节点可以定义`Solid`节点碰撞时的行为，如图2所示

图 2: ContactProperties

2. 设置两个`ContactProperties`节点的参数如图3所示

第一个`ContactProperties`节点如下图所示：



第二个`ContactProperties`节点如下图所示：

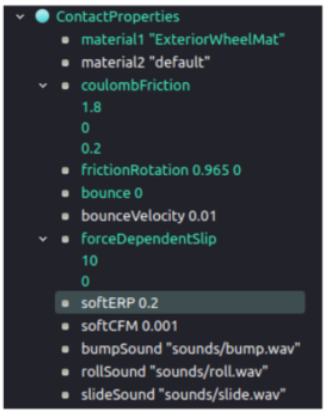


图 3: ContactProperties 参数设置

3. 添加机器控制器

在向导中添加新机器人控制器，使用`C++`，控制器名字叫`main`，参考老师的代码自己写了一个程序，生成项目即可运行

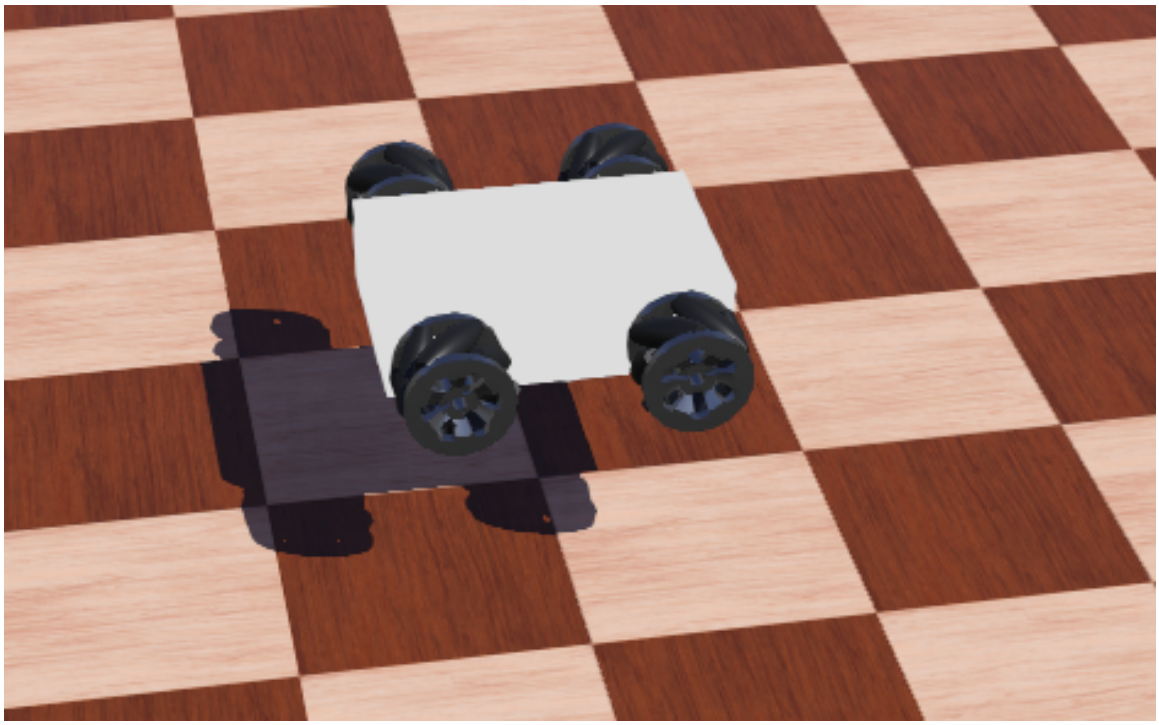


图 4: 制作完成的麦轮小车

三、实验结果与分析

1. 参数分析

在 ContactProperties 节点中，coulombFriction 字段可以设置 4 个参数。只有一个参数时摩擦是对称的；两个参数时摩擦是不对称的；三个参数时对 material1 使用不对称系数，对于 material2 使用对称系数；四个参数时对 material1 和 material2 都使用不对称系数。我们设置了 3 个参数，说明对 material1 是不对称系数，对 material2 是对称系数。

frictionRotation 字段中，第一节点设置成-0.965，第二节点设置成 0.965。bounce 字段设置成 0，对象会有效的停止在和他碰撞的表面，不会弹跳。forceDependentSlip 字段可以设置 4 个参数。使用一个参数，则该系数适用于两个方向；两个参数时力相关滑移是不对称的，对两个 material 使用相同的系数；三个参数时对 material1 使用不对称系数，对于 material2 使用对称系数；四个参数时对 material1 和 material2 都使用不对称系数。

通过使用上面的节点配置，可以实现麦克纳姆轮的运动方式

2. 代码分析

获取句柄，初始化各项参数值，注意速度不能设太大，否则会超过限制。

```
1   Motor *motors[4];//电机和键盘都要用webots给的类型
2   webots::Keyboard keyboard;
3   char wheels_names[4][8] = { "motor1","motor2","motor3","motor4" };
```

```
4
5     Robot *robot = new Robot();//使用webots的机器人主体
6     keyboard.enable(1);//运行键盘输入设置频率是1ms读取一次
7
8     double speed1[4];
9     double speed2[4];
10    double velocity = 5;// 速度为10的话会超过限制!
11
12    //初始化
13    for (int i = 0; i < 4; i++) {
14        motors[i] = robot->getMotor(wheels_names[i]);//获取句柄
15        motors[i]->setPosition(std::numeric_limits<double>::infinity());
16        motors[i]->setVelocity(0.0);
17        speed1[i] = 0;
18        speed2[i] = 0;
19    }
20
21    double speed_w[4] = { velocity ,velocity ,velocity ,velocity };
22    double speed_s[4] = { -velocity ,-velocity ,-velocity ,-velocity };
23    double speed_a[4] = { velocity ,-velocity ,velocity ,-velocity };
24    double speed_d[4] = { -velocity ,velocity ,-velocity ,velocity };
25    double speed_q[4] = { velocity ,-velocity ,-velocity ,velocity };
26    double speed_e[4] = { -velocity ,velocity ,velocity ,-velocity };
27
28    int timeStep = (int)robot->getBasicTimeStep();//获取你在webots设置一帧的时间
29    cout << timeStep << endl;
```

获取键盘输入，根据按键决定电机如何转动

```
1     while (robot->step(timeStep) != -1) {
2         //获取键盘输入
3         int k1 = keyboard.getKey();
4         int k2 = keyboard.getKey();
5         cout << k1 << ":" << k2 << endl;
6
7         switch (k1) {
8             case 'W' :
9                 for (int i = 0; i < 4; i++)
10                    speed1[i] = speed_w[i];
11                break;
12            case 'S' :
13                for (int i = 0; i < 4; i++)
14                    speed1[i] = speed_s[i];
15                break;
16            case 'A' :
17                for (int i = 0; i < 4; i++)
```

```
                speed1[i] = speed_a[i];
            break;
        case 'D' :
            for (int i = 0; i < 4; i++)
                speed1[i] = speed_d[i];
            break;
        case 'Q' :
            for (int i = 0; i < 4; i++)
                speed1[i] = speed_q[i];
            break;
        case 'E' :
            for (int i = 0; i < 4; i++)
                speed1[i] = speed_e[i];
            break;
        default:
            for (int i = 0; i < 4; i++)
                speed1[i] = 0;
    }

    switch (k2) {
        case 'W' :
            for (int i = 0; i < 4; i++)
                speed2[i] = speed_w[i];
            break;
        case 'S' :
            for (int i = 0; i < 4; i++)
                speed2[i] = speed_s[i];
            break;
        case 'A' :
            for (int i = 0; i < 4; i++)
                speed2[i] = speed_a[i];
            break;
        case 'D' :
            for (int i = 0; i < 4; i++)
                speed2[i] = speed_d[i];
            break;
        default:
            for (int i = 0; i < 4; i++)
                speed2[i] = 0;
    }

    //让电机执行
    for (int i = 0; i < 4; i++)
        motors[i]->setVelocity(speed1[i] + speed2[i]);
}
```

3. 结果展示

最终完成的麦轮小车可以八向移动并且自旋!

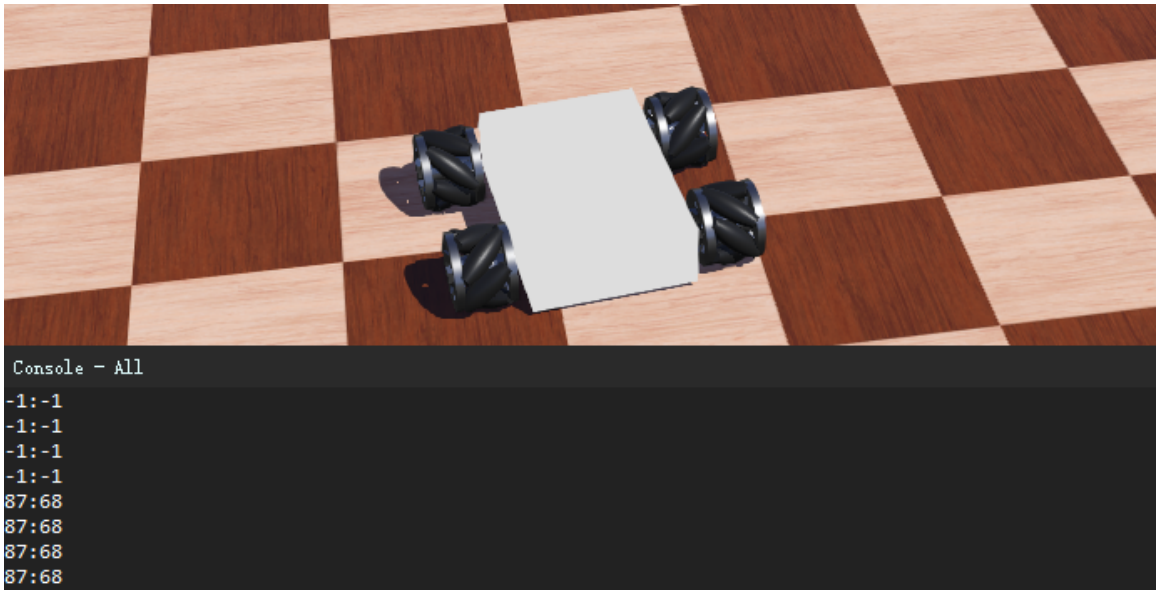


图 5: 运动中的麦轮小车

四、实验中的问题和解决方法

这次实验虽然不是很难，只要按照老师的 PPT 做就可以了，但因为是第一次使用 Webots 这个软件，所以还是花费了我大量的时间，并且一开始因为不熟悉 Webots 保存和重载的方式，导致我开始做的小车卡 bug 就直接丢失了，重做了好几次。

1. 按下 A 和 D 小车没有左右动，反而前后动

最后发现是粗心问题，不小心将第二个`ContactProperties`节点的`material1`参数也设成了`InteriorWheelMat`，修改为`ExteriorWheelMat`就解决了

2. 设置速度太大报错

刚开始按照老师的程序设置 $velocity = 10$，发现按下两个键时速度为 $velocity = 20$，超过了 $velocity = 14.81$ 的限制

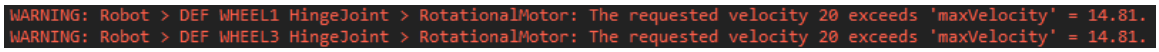


图 6: 速度设置太大

最终在程序里设置 $velocity = 5$ 即可解决问题

3. 小车和地面打滑发生“漂移”

按照要求调好参数后，我发现我的小车在移动完一段距离后会打滑“漂移”，而样例却没有这个问题，于是我通过略微加大`coulombFriction`和`Robot`的`density`两个参数的方式使打滑的现象得到缓解，但有时还是会“漂移”一点点距离。