视觉巡线小车

18329015 郝裕玮

一、实验目标

为小车添加相机，GPS，使得小车沿着规定的轨道（黑线）按照一定的速度进行移动，并计算小车的绕行一圈的时间，平均速度和巡线误差。

二、实验内容与步骤

（1）在 Robot 的 children 中添加 camera 节点，并在 camera 的 children 中添加 transform 节点，再在 transform 的 children 中添加 Shape 节点，同时需要设置 camera 的句柄为 camera，最后设置其位置，大小等参数即可；

（2）在 Robot 的 children 中添加 GPS 节点，并在 camera 的 children 中添加 Solid 节点，再在 Solid 的 children 中添加 Shape 节点，同时需要设置 GPS 的句柄为 gps，最后设置其位置，大小等参数即可；

（3）在 RectangleArena->floorAppearance->baseColorMap->url 选择 Circle 图，替换之前的轨迹图。

控制器代码如下所示（**代码注释中已包含所有的思路分析和算法实现**）：

```
#include <webots/Robot.hpp>
#include <webots/Motor.hpp>
#include <webots/Keyboard.hpp>
#include <webots/Camera.hpp>
#include <webots/GPS.hpp>
#include <iostream>

#include <algorithm>
```

```cpp
#include <iostream>
#include <limits>
#include <string>
#include<string.h>

using namespace std;
using namespace webots;

int main() {
    Motor *motors[4];//电机和键盘都要用 webots 给的类型
    webots::Keyboard keyboard;
    char wheels_names[4][8]={"motor1","motor2","motor3","motor4"};//对应
RotationMotor 里的句柄

    Robot *robot=new Robot();//使用 webots 的机器人主体
    Camera *camera = robot->getCamera("camera");//获取相机，句柄名为
camera
    camera->enable(1);//设置相机每 1ms 更新 1 次

    keyboard.enable(1);//运行键盘输入设置频率是 1ms 读取一次

    GPS* gps = robot->getGPS("gps");//获取 GPS，句柄名为 gps
    gps->enable(1);//设置 GPS 每 1ms 更新 1 次

    double speed[4];//此数组会在后面赋值给电机以速度
    double velocity=10;//初速度
    int i;
    double cur_speed=0;//瞬时速度
    double avg_speed=0;//平均速度
    double speed_sum=0;

    //初始化
    for(i=0;i<=3;i++){
        motors[i]=robot->getMotor(wheels_names[i]);//按照你在仿真器里面设
置的名字获取句柄
        motors[i]->setPosition(std::numeric_limits<double>::infinity());
        motors[i]->setVelocity(0.0);//设置电机一开始处于停止状态
        speed[i]=3;//给予小车一个初速度
    }

    double speed_forward[4]={velocity,velocity,velocity,velocity};//前进
方向
    double speed_leftCircle[4]={velocity,-velocity,-
velocity,velocity};//左自旋(即左转弯)
```

```
    double speed_rightCircle[4]={-velocity,velocity,velocity,-
velocity};//右自旋(即右转弯)

    int timeStep=(int)robot->getBasicTimeStep();//获取你在 webots 设置一帧
的时间
    int cnt=0;//统计帧数
    double error_sum=0;//记录瞬时的半径累加和
    double result=0;//记录误差
    double x0=0,y0=1.22;//手动大致测量圆心坐标
    double cur_x,cur_y;//记录 GPS 的实时坐标
    double pi;//当前的瞬时半径
    double cur_error;//当前的瞬时误差
    double r=1.185;//手动测量的真实半径

    while(robot->step(timeStep)!=-1){//仿真运行一帧
        cnt++;//统计帧数
        const unsigned char *a=camera->getImage();//读取相机抓取的最后一张
图像。图像被编码为三个字节的序列，分别代表像素的红、绿、蓝

        int length=camera->getWidth();//图像长度
        int width=camera->getHeight();//图像宽度
        cout<<"图像尺寸为："<<length<<'*'<<width<<endl;//输出图像尺寸

        int b1,b2,b3,b4;
        b1=length*3*width/2;//图像中间一行的最左边的像素点
        b2=length*3*width/2+(width/2+3)*3;//图像中间一行的中间靠左的某像素
点
        b3=length*3*width/2+(width/2+5)*3;//图像中间一行的中间靠右的某像素
点
        b4=length*3*width/2+(length-1)*3;//图像中间一行的最右边的某像素点
        //其中 b1，b2 代表图像中间的黑色轨迹的两侧(大致估计)

        cur_x=gps->getValues()[0];//获取实时 x 坐标
        cur_y=gps->getValues()[1];//获取实时 y 坐标

        pi=sqrt((cur_x-x0)*(cur_x-x0)+(cur_y-y0)*(cur_y-y0));//计算当前的
瞬时半径
        cur_error=abs(pi-r);//实时误差=|pi-r|
        error_sum+=cur_error;//误差累加
        result=error_sum/cnt;//计算当前的平均误差

        cur_speed=gps->getSpeed();
        speed_sum+=cur_speed;
        avg_speed=speed_sum/cnt;
```

```
        cout<<"当前经历了"<<cnt<<"帧"<<endl;
        cout<<"当前实时半径为"<<pi<<endl;
        cout<<"当前瞬时速度为"<<cur_speed<<","<<"当前平均速度为
"<<avg_speed<<endl;
        cout<<"当前实时误差为"<<cur_error<<","<<"当前平均误差为
"<<result<<endl;

        //以 rgb 的 r 为标准，当颜色为黑时，r 的值必定小于 80(10-30 左右)
        if(a[b1]<80&&a[b2]>80&&a[b3]>80&&a[b4]>80){//若只有最左边像素点为
黑色，则小车需要右转弯使得轨道黑线向中间靠拢
            for(i=0;i<=3;i++){
                speed[i]=speed_rightCircle[i];//速度方向为右转
            }
        }
        else if(a[b1]>80&&a[b2]<80&&a[b3]<80&&a[b4]>80){//若中间两个像素
判断点为黑色，则小车可选择继续直行
            for (i=0;i<=3;i++){
                speed[i]=speed_forward[i];
            }
        }
        else if(a[b1]>80&&a[b2]>80&&a[b3]>80&&a[b4]<80){//若只有最右边像
素点为黑色，则小车需要左转弯使得轨道黑线向中间靠拢
            for (i=0;i<=3; i++){
                speed[i]=speed_leftCircle[i];
            }
        }
        else if(a[b1]>80&&a[b2]>80&&a[b3]>80&&a[b4]>80){//若四个判断像素
点全为白色，则小车可选择继续直行
            for(i=0;i<=3;i++){
                speed[i]=speed_forward[i];
            }
        }

        //将速度赋值给电机
        for(i=0;i<=3;i++){
            motors[i]->setVelocity(speed[i]);
        }

    }
    return 0;
}
```

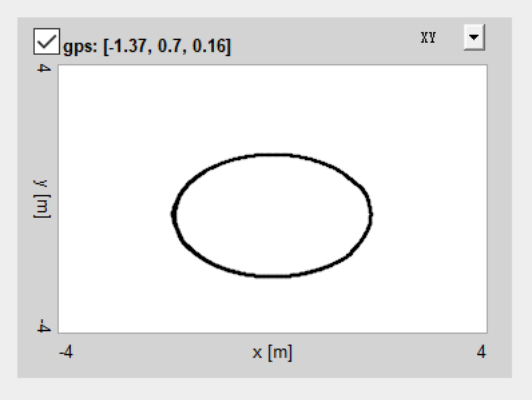
三、实验结果与分析

Robot 的 children 节点结构如下图所示：

最终运行结果如下表所示：

	第 1 次	第 2 次	第 3 次	平均值
运动时间	19.456	19.552	19.424	19.477
运行帧数	607	610	606	607.677
平均速度	0.670808	0.670202	0.670917	0.670642
巡线误差	1.3173	1.31133	1.31936	1.31597

运行轨迹如下图所示：



四、实验中的问题和解决方法

（1）一开始没有在网上找到相关的参考资料，导致安装相机后无从下手，不知道该如何将相机拍摄到的图像转换为判断小车前进方向的依据。后来通过查询 Webots controller API（C++）明白了 Camera 库中的可用函数，最终选择采用 getImage 函数获取图像每个像素点的 RGB 值，并根据粗略的特定位置的像素点是否为黑色来判断小车的前进方向；

（2）本想再添加 2 个 Robot 并添加 GPS，使得可在圆上任意一点和圆心处再放置 2 个 GPS，但是项目报错，说只能添加 1 个机器人实例，只好作罢。后来发现群里助教说可大致估计圆的半径并作为已知量使用，所以我自行移动了 Robot 的 GPS 将其定位在圆心，记录其坐标，并只改变 x 值使得 GPS 再停留在圆的边缘上，计算 x 的差值即可得到圆的半径的估计值，约为 1.185m。

（3）一开始我发现 GPS 的速度只有 0.62m/s，觉得不符合常理（毕竟电机速度为 10rad/s）。以为是我调用函数有问题，Debug 了很久，最后发现该结果实际上是正确的，证明过程如下：

理论最优平均速度=$\frac{2\pi r}{t} \approx \frac{2*3.14*1.185}{19.477}$= 0.382m/s

实际估算平均速度=$\frac{2\pi r}{t} \approx \frac{2*3.14*(1.185+1.31597)}{19.477}$ = 0.806m/s

又因为实际轨迹不可能是真正的圆，只是近似于圆，所以最终实际速度必定在[0.382,0.806]这个区间内，所以结果无误。