

**课 程 实 验 报 告**

**课程名称： 传感器原理及工程应用**

**专业班级： 物联网1701**

**学 号： U201714830**

**姓 名： 莫昆桦**

**指导教师： 宋恩民**

**报告日期： 2020.5.11**

**计算机科学与技术学院**

目录

[1、 实验目的 1](#_Toc371)

[2、 实验内容 2](#_Toc9659)

[3、 实验方案 3](#_Toc7819)

[4、 核心代码注释 5](#_Toc29072)

[5、 实验过程、结果及分析 7](#_Toc13521)

[6、 实验中遇到的问题及分析、总结 9](#_Toc6159)

1. 实验目的

通过实验，学习将手机上的传感器用于做复合数据采集和应用，包括：多传感器协同测量，多维数据的记录、整理和分析。

# 实验内容

实验内容如下：

一、将一部或两部手机，固定在自行车上的合适地方。推着或骑着自行车行走一段距离，同时采集和自动记录手机上的多个传感器的值。试着用这些值来分析描述路途的情况。

二、自己编写程序，根据记录的值绘出自行车的运动轨迹图。最好能在网上下载一个当地的局部地貌图，把轨迹绘制这个图上。

三、在轨迹上若干处（至少5处）有特点的地方（如拐弯、上坡、树荫）标出当时的各种传感器的值。

1. 根据各点处的传感器的值，计算各种环境和运动参数，如时间、速度、加速度、坡度、拐弯角度、温度、光照强度、噪声强度、颠簸程度等。
2. 解释各处状况对传感器值的影响（如有汽车从旁呼啸而过，所以噪声强度瞬时增大；到了树荫处，所以温度和亮度都暂时变低；下坡到了坡底转上坡，所以此处有较大的负加速度。
3. 试着用这些传感器的值计算一些间接的参数，如在某转弯处自行车转弯圆弧的半径，转弯时自行车向弯道内侧倾斜的角度；某下坡路段的长度；在某上坡路段人登脚踏的节奏（频率）。
4. 加分选择项：有条件的同学，可以增加蓝牙或wifi传输功能，把传感器数据实时传输到笔记本电脑上，由笔记本电脑实时记录并实时绘制轨迹地图。
5. 实验方案

首先是确定手机绑定的位置为自行车把上，手机屏幕与地面保持平行。因为自行车运动的时候，车头是运动最能彰显自行车整体的运动，（比如拐弯的时候车头的左右运动也会带着手机进行左右运动），所以将其固定在自行车的车把上。获取的传感器是加速度计、陀螺仪、光强传感器、罗盘的值。

路线的测量主要是依靠加速度计和罗盘的值来进行测定。自行车的行进距离由自行车的车轮的转动圈数进行间接测定，然后通过圈数乘以圆的周长得出行走距离。自行车的圈数的计算与之前实验设计的计步器原理一样，首先在自行车的前轮系上一根绳子，这样每转过一圈自行车的前轮就会上下震动，每一次震动即为一圈。自行车的行进方向通过罗盘测出。

绘图的时候设置定时器，每隔一秒绘一个点。绘图的时候以坐标的形式将上一个记录的点和当前的点连线，坐标的计算方法如下图3.1所示。因为绘图的时候的点是二维坐标，而自行车处于三维空间，在行进过程中会有上坡下坡等过程，所以我们需要将将自行车的位置压缩为一个平面上的二维坐标。其中上坡的角度的计算与之间斜度计的设计一样。速度由行进距离与时间的商所求。

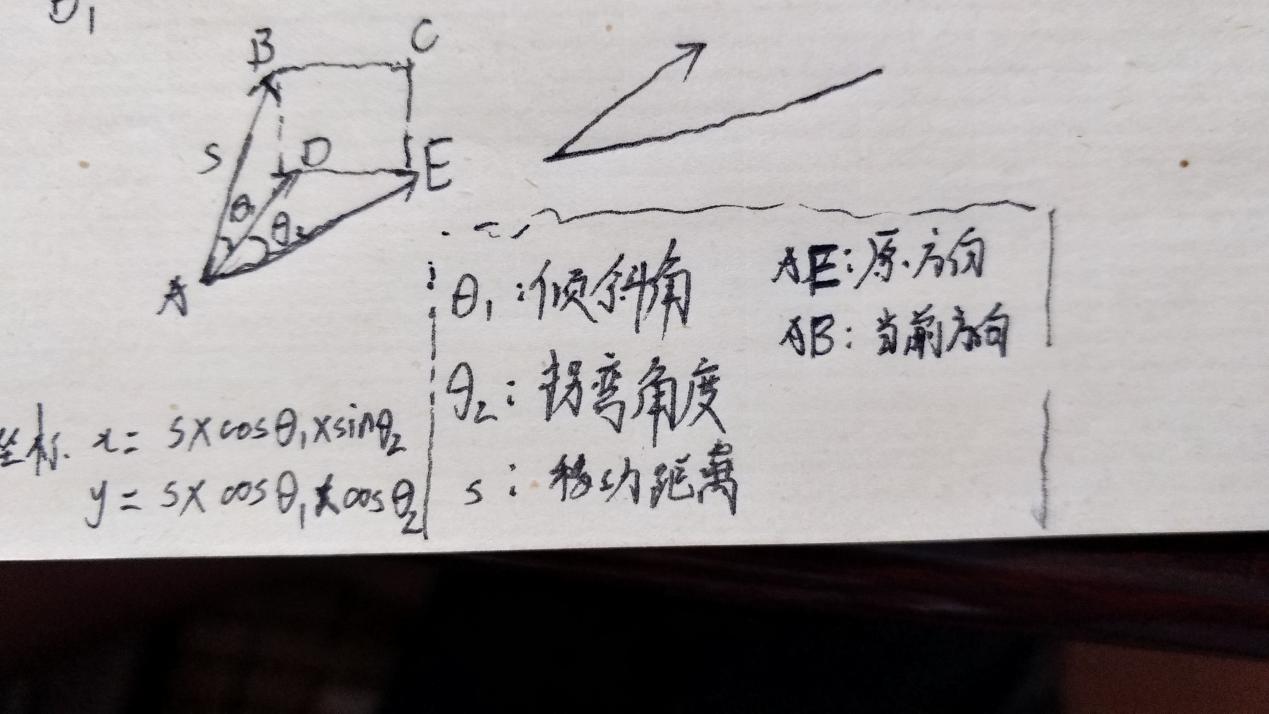


图3.1 坐标计算

记录的传感器的值还可以计算拐弯的时候圆弧半径，计算方法如图3.2所示。利用陀螺仪测出的角速度，还有加速度计间接获取的弧长（也就是自行车的行进距离）进行计算。

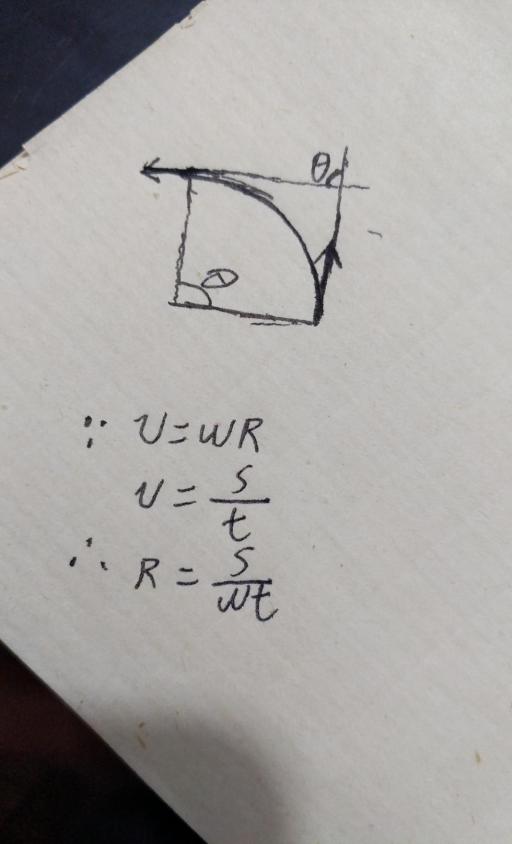


图3.2 拐弯半径计算

4、核心代码及注释

onLoad: function() {

    this.ctx = wx.createCanvasContext('myCanvas', this); //创建画布绘图环境

    this.ctx.setStrokeStyle(this.data.color);//设置画布颜色为背景颜色

    this.ctx.setLineWidth(this.data.pen);//画笔粗细

    this.ctx.setLineCap('round');

    console.log(h+" : "+m+" : "+s);

    console.log(h\*3600+m\*60+s);

  },

touchStart: function() { //设置绘制起点

    //this.x1 = points[0]; //将开始点x坐标赋值给x1

    //this.y1 = points[1]; //将开始点x坐标赋值给y1

    //console.log("x1: "+this.x1+" y1: "+this.y1); //坐标

      this.ctx.beginPath();//开始一个路径

    //}

  },

  touchMove: function() { //在起点与终点之间连线

    //var x2 = points[0]; //将当前点的x坐标赋值给x2

    //var y2 = points[1]; //将当前点的y坐标赋值给y2

     //画线模式

      this.ctx.moveTo(last\_x, last\_y); //将画笔移动到起点

      this.ctx.lineTo(now\_x, now\_y);//在起点与当前点之间画线

      this.ctx.stroke();

      last\_x = now\_x;//将当前点x坐标赋值给起点x坐标

      last\_y = now\_y;//将当前点y坐标赋值给起点y坐标

    this.ctx.draw(true);//绘制

  },

startBrightness:function(){//获取光强

    var that=this;

    wx.getScreenBrightness({

      success:function(res){

        console.log(res.value)

        that.setData({

          value: res.value

        })

      }

    })

  },

   startCompass: function() {

    var that = this

    wx.startCompass({ //启动罗盘传感器监听功能

      success: function() {

        wx.onCompassChange(function(res) { //监听罗盘传感器

          Cpass = res.direction;

          that.setData({

            result: res.direction  //res为回调函数的参数

          })

        })

      }

    })

  },

startAcc: function() {

    var that = this;

    wx.startAccelerometer({ //启动加速度感器监听功能

      success: function() {

        wx.onAccelerometerChange(function(res) { //监听加速度传感器

          var temp = res.x \* res.x + res.y \* res.y + res.z \* res.z;//三轴加速度矢量和

          temp = Math.sqrt(temp);

          var ay = Math.abs(res.y);

          var az = Math.abs(res.z);

          var its = 0.0;

          if(az !== 0) its = Math.atan(ay/az)\*180/Math.PI;//倾斜角度不为90度时使用公式计算

          else its = 90.0;//倾斜角度为90度的时候

          du = Math.abs(its);//记录倾斜度

          var ni = Date.parse(new Date());

          date = new Date(ni);

          h = date.getHours();//时

          m = date.getMinutes();//分

          s = date.getSeconds();//秒

          var now = h+" : "+m+" : "+s;//时间

          var time = h\*3600+m\*60+s;

          if(flag == 0 && temp > 2) {//flag表示状态，当a状态到达上限时转为b状态

            flag = 1;

            console.log("flag = 1");

          }

          else if(flag == 1 && temp < 1) {//当b状态到达下限时计数，转为b状态

            flag = 0;

            console.log("flag = 0")

            var kaku = Cpass - last\_dre;//自行车转过的角度

            now\_x = last\_x + Math.cos(du) \* round \* Math.sin(kaku);

            now\_y = last\_y + Math.cos(du) \* round \* Math.cos(kaku);//计算坐标

            if(time-last\_time > 0)speed = round/((time - last\_time)\*10);//计算速度

            var r = 0;

            if(time-last\_time > 0)r=Math.abs(round/(Gy.z\*(time-last\_time)));//计算半径

            last\_time = time;//更新时间

            count++;//计数

            console.log("count = " + count);

          }

          that.setData({

            resAcc:res,//res为回调函数的参数

            speed:speed,

            du:its,

            rounds:r

          })

        })

      }

    })

  },

startGyroscope: function() {

    var that = this;

    wx.startGyroscope({ //启动陀螺仪传感器监听功能

      success: function(res) {

        wx.onGyroscopeChange(function(res) { //监听陀螺仪传感器

          Gy = res;

          var ni = Date.parse(new Date());

          date = new Date(ni);

          h = date.getHours();//时

          m = date.getMinutes();//分

          s = date.getSeconds();//秒

          var now = h+" : "+m+" : "+s;//时间

          //this.startBrightness();

          that.setData({

            resG: res,  //res为回调函数的参数

            now:now

          })

        })

      }

    })

  },

start : function () {

        var that = this;

        this.timer = setInterval(repeat, 1000); //设置定时器，每1000ms调用一次repeat函数

        this.touchStart(); //调用绘图函数

        this.startAcc(); //调用各个函数监听传感器

        this.startCompass(); //调用各个函数监听传感器

        this.startGyroscope(); //调用各个函数监听传感器

        function repeat() {

            console.log('repeat');

            that.touchMove();//连线并更新起点终点坐标

            that.startBrightness();//调用获取光强

        }

    },

    end : function () {

        console.log('end');

        clearInterval(this.timer);//删除定时器定时器

        this.ctx.draw();

        this.stoptAcc();//停止监听加速度计

        this.stopCompass();//停止监听罗盘

        this.stopGyroscope();//停止监听陀螺仪

    }

# 5、实验过程、结果及分析

将手机绑定在自行车的车把上，屏幕与地面平行。进入小程序，选择轨迹绘制，点击启动后，骑自行车进行运动。轨迹绘制如图5.1所示，在每次进行实验的时候，程序会设置运动的原点为画布坐标系的原点（0，0），即画布的左上角，所以每次运动的轨迹都是从左上角出发进行绘制的。运动的过程会实时显示当前时间其中小程序的界面设计是上方为一个矩形的画布用于绘制轨迹；下方是各个传感器的值和由这些值计算得到的结果，如时间、速度、坡度等等。多次进行实验查看绘制轨迹，大多数结果与实际运动相符，由传感器计算的值如速度、坡度都在小误差内显示正确。



图5.1 绘制轨迹

本次实验在拐弯、阴影处、上坡、下坡四处地方采取了传感器的值，并进行相应计算。如图5.2所示为拐弯时的各个测量值。分别在左拐弯和右拐弯的两种情况下进行了多次实验，实验得到的拐弯半径、速度也在小误差内计算正确；因为是在亮处进行的实验，所以亮度显示很大（亮度范围0~1，1为最亮状态）。在阴影处的测量如图5.3所示。可见亮度变小了。上坡时的测量值如图5.4所示，我们可见程序计算出来的坡度；还有速度变小了。下坡时的测量如图5.5所示，同样显示了下坡的坡度，而且速度比较大。



图5.2 拐弯测量结果



图5.3 阴影测量结果



图5.4 上坡测量结果



图5.5 下坡测量结果

# 6、实验中遇到的问题及分析、总结

首先是坐标计算的问题。一开始计算的坐标相对于画布的坐标系来说变化太小，在画布上画出的线缩在一团，不好观察；后来在计算时将坐标都放大数倍（具体的实现是修改自行车的轮子半径，增大为十倍，这样计算出来的行进距离也为十倍），这样画出来的线就清晰多了。相对的，在计算速度的时候，会将其除以10再显示相应的结果

圈数的计算也有许多问题，因为是靠震动来计数，所以实验场所被限定在了平坦路面；计算震动计数如图6.1所示，当加速度从A状态转为B状态再次转为A状态的时候计一次，这样高线和低线的设置就需要先通过其他软件绘出波形图，观察后进行设置，借助了外部工具，反复设置高低线进行实验了很多次才得到比较准确结果。但是实际进行实验的时候，将程序的计数结果在控制台输出显示，然后与自己计算的震动次数进行对比发现结果还是有偏差，程序计数很多情况下会少1到3次计数。可能因为这个原因坐标的取点有时候会断断续续，这样会影响实际画图效果，所以进行多次实验后发现有时候轨迹绘制不是很符合实际运动。

拐弯的半径的测量，即使在没有拐弯的直线运动的时候，因为角速度还是有比较小的值，所以计算出来的半径值会很大。同样在水平面上运动的时候，坡度显示还是会有一个小角度。

这次实验的轨迹绘图，采取的是在画布上绘制曲线的简单形式，无论在哪出地方进行实验，轨迹的坐标点地设置为画布坐标系的原点（即左上角位置），然后进行绘图。绘制过程中使用到了多传感器的结合（轨迹坐标、拐弯半径等），也用到了之前实验的一些成果（如坡度计算用于坐标计算），是一次非常综合的实践。

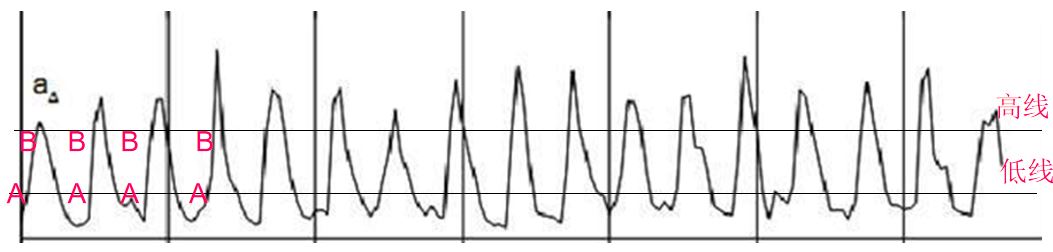


图6.1 计数高低线