感知技术与应用

实
验
报
告

学　　院　网安学院

年　　级　2023 级

班　　级　1065

学　　号　2310422

姓　　名　谢小珂

2025 年 5 月 1 日

目录

一、实验目的

本次实验的目的是让大家了解 Android 中加速度传感器的基本知识，掌握 Android 中加速度传感器的使用方法。

二、实验内容

1.学习加速度传感器的分类（压电式、压阻式、电容式、伺服式）及其应用领域（如汽车安全、游戏控制、图像自动翻转等）。
2.掌握 Android 系统中加速度传感器（TYPE_ACCELEROMETER）的使用，包括获取 X、Y、Z 三轴的加速度值。
步骤：
• 编写布局文件（activity_main.xml），定义显示加速度值的 TextView 控件。
• 编写程序文件（MainActivity.java），实现传感器监听功能，实时显示三轴加速度值。
3.设计一个 APP，功能包括：
• 显示手机三个方向的加速度值。
• 通过加速度值判断手机是否处于静止状态。
• 记录用户步行步数。

三、实验步骤及实验结果

• 实验步骤

1.编写文件 activity_main.xml，具体实现如下所示。

定义了应用的界面布局，包含多个 TextView 用于显示传感器数据和状态信息。

1.1 垂直线性布局：

标题和五个 TextView 分别显示：X 轴加速度、Y 轴加速度、Z 轴加速度、手机状态、当前步数。

统一设置文字颜色（白色）和大小（30px）。

1.2 设置背景图片：使用自定义图片作为背景。

2.编写文件 MainActivity.java，具体实现如下所示。

MainActivity 是 Android 应用的主活动，负责初始化界面、注册传感器监听器，并实时显示加速度数据、移动状态和步数统计。

2.1 控件初始化：

• 作用：获取布局中的 TextView，用于显示传感器数据和状态。

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main); // 加载布局文件

    // 初始化 TextView
    myTextView1 = (TextView) findViewById(R.id.myTextView1); // X 轴加速度
    myTextView2 = (TextView) findViewById(R.id.myTextView2); // Y 轴加速度
    myTextView3 = (TextView) findViewById(R.id.myTextView3); // Z 轴加速度
    myTextView4 = (TextView) findViewById(R.id.myTextView4); // 手机状态（静止/移动）
    myTextView5 = (TextView) findViewById(R.id.myTextView5); // 步数统计

    // 初始化 SensorManager
    mySensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
}
```

通过 findViewById 绑定 XML 布局中的 TextView；myTextView1-3 分别显示 X、Y、Z 轴加速度值；myTextView4 显示手机状态（静止/移动）；myTextView5 显示计步结果。

2.2 传感器监听：

• 作用：注册加速度传感器监听，实时获取数据

通过 SensorManager 注册加速度传感器监听器（SensorListener）。

onResume() 和 onPause() 管理监听器生命周期。

2.3 传感器数据处理（onSensorChanged）：

• 作用：实时处理加速度数据，更新 UI。

```
private SensorListener mySensorListener = new SensorListener() {
    @Override
    public void onAccuracyChanged(int sensor, int accuracy) {
        // 精度变化时触发（通常不需要处理）
    }

    @Override
    public void onSensorChanged(int sensor, float[] values) {
        if (sensor == SensorManager.SENSOR_ACCELEROMETER) {
```

```
            // (1) 调用 StepDetector 检测步伐
            stepDetector.processAccel(values, System.currentTimeMillis());

            // (2) 显示三轴加速度值
            myTextView1.setText("x 方向上的加速度为: " + values[0]);
            myTextView2.setText("y 方向上的加速度为: " + values[1]);
            myTextView3.setText("z 方向上的加速度为: " + values[2]);

            // (3) 计算合加速度并判断静止状态
            float x = values[0], y = values[1], z = values[2];
            float _3Da = (float) sqrt(x * x + y * y + z * z); // 欧几里得距离

            if (abs(_3Da - 9.8) < 1.0f) { // 静止条件：合加速度接近重力加速度
                myTextView4.setText("当前状态判断: 静止");
            } else {
                myTextView4.setText("当前状态判断: 移动");
            }

            // (4) 更新步数
            myTextView5.setText("当前步数为: " + stepDetector.stepCount);
        }
    }
};
```

2.3.1 步伐检测：

调用 stepDetector.processAccel() 处理加速度数据，更新步数。

2.3.2 三轴加速度显示：

直接显示 values[0]（X）、values[1]（Y）、values[2]（Z）。

2.3.3 静止状态判断：

• 计算三轴合加速度

在 onSensorChanged 获取了 X、Y、Z 三个方向的加速度值，并计算它们的合加速度：

```
float x = values[0], y = values[1], z = values[2];
float _3Da = (float) sqrt(x*x + y*y + z*z); // 欧几里得距离算加速度
```

物理意义：

当手机静止时，合加速度 _3Da 应该接近 9.8 m/s²（地球重力加速度）。

当手机移动时，合加速度会因额外加速度（如走路、晃动）而偏离 9.8。

• 判断静止状态

通过比较合加速度与重力加速度的差值来判断手机是否静止：

```
if(abs(_3Da - 9.8) < 1.0f) {  // 差值小于 1.0 时认为是静止
    myTextView4.setText("当前状态判断: 静止");
} else {
```

```
    myTextView4.setText("当前状态判断：移动");
}
```

判断逻辑：

abs(_3Da - 9.8) < 1.0f：如果合加速度与 9.8 的差值小于 1.0，则认为手机处于静止状态；否则，认为手机在移动。

阈值 1.0f：经验值，用于容忍微小波动（如手持时的自然抖动）；可根据实际需求调整。

2.3.4 步数更新：

从 stepDetector.stepCount 获取当前步数并显示。

3. 编写文件 StepDetector.java，具体实现如下所示

StepDetector 是一个用于检测用户行走步数的类，主要基于加速度传感器的数据进行分析。它的核心逻辑包括低通滤波去除重力分量、计算线性加速度、波峰检测和步数统计。

3.1 低通滤波（Low-Pass Filter）

- 作用：分离重力分量（低频信号），保留用户运动的线性加速度（高频信号）。
- 实现方式：使用一阶低通滤波算法，公式为：

$$\text{output}[i] = \text{output}[i] + \alpha \times (\text{input}[i] - \text{output}[i])$$

其中，α = 0.8 是滤波系数（经验值，越大滤波越强）。

关键代码如下：

```
private float[] lowPass(float[] input, float[] output) {
    if (input == null || output == null) {
        Log.d("StepDetector", "输入或输出数组为空");
        return input;
    }
    for (int i = 0; i < 3; i++) {
        output[i] = output[i] + ALPHA * (input[i] - output[i]);
    }
    return output;
}
```

输入：原始加速度数据 input（含重力）。

输出：滤波后的重力分量 output（低频信号）。

作用：减少高频噪声（如手部抖动）；保留重力分量，便于后续计算线性加速度。

3.2 计算线性加速度（去除重力）

• 作用：获取用户运动产生的加速度（不含重力）。

• 方法方式：用原始加速度减去低通滤波后的重力分量。

关键代码如下：

```
float x = values[0] - gravity[0];
float y = values[1] - gravity[1];
float z = values[2] - gravity[2];
```

输入：原始加速度 values 和滤波后的重力 gravity。

输出：线性加速度（x, y, z），代表用户运动的加速度（如走路时的上下波动）。

3.3 计算加速度矢量幅度（合加速度）

• 作用：综合三轴加速度，计算总加速度大小（标量），便于检测步伐。

公式：

$$\text{accel} = \sqrt{x^2 + y^2 + z^2}$$

关键代码如下：

```
float accel = (float) Math.sqrt(x*x + y*y + z*z);
```

• 物理意义：

当手机静止时，accel ≈ 9.8 m/s²（重力加速度）。

当用户走路时，accel 会周期性波动（波峰代表步伐）。

3.4 步伐检测（波峰检测 + 时间约束）

• 作用：判断是否发生有效步伐（避免误判）。

• 检测条件：加速度变化率超过阈值（delta > ACCEL_THRESHOLD）。

两次步伐间隔足够长（timestamp - lastStepTime > STEP_DELAY_MS）。

```
private void detectStep(float accel, long timestamp) {
    // 计算加速度变化率（相对于重力）
    float delta = Math.abs(accel - 9.8f);

    // 波峰检测条件
    boolean isPeak = delta > ACCEL_THRESHOLD; // 加速度变化足够大
    boolean isTimeValid = (timestamp - lastStepTime) > STEP_DELAY_MS; // 避免连续误判

    if (isPeak && isTimeValid) {
        lastStepTime = timestamp; // 记录最后一次有效步伐时间
        stepCount++; // 步数+1
        onStepDetected(stepCount); // 通知监听器（如更新 UI）
```

```
    }
}
```

• 参数说明：

ACCEL_THRESHOLD = 10.0f：经验值，需根据实际测试调整（太大会漏检，太小会误检）。

STEP_DELAY_MS = 300：最小步伐间隔（毫秒），避免短时间内重复计数。

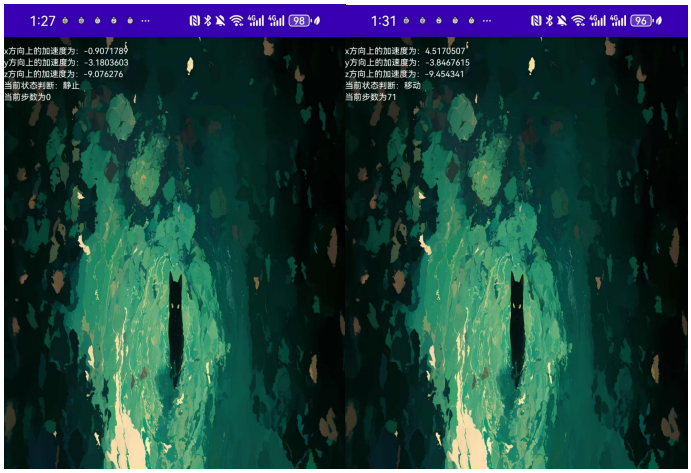
• 优化点：

可结合 滑动窗口 或 机器学习 提高检测精度。

可动态调整 ACCEL_THRESHOLD 适应不同用户步态。

· 实验结果

安卓设备检测结果如下图所示：



图左为静止状态，步数为 0，xyz 方向加速度正常。

图右为移动状态，步数为 71，xyz 方向加速度正常。

四、实验遇到的问题及其解决方法

1. 步数统计不准确

问题分析：在实验初期，步数统计存在不准确的情况，主要原因在于步伐检测机制不够完善。一方面，单纯依据加速度变化判断步伐，未充分考虑到日常使用手机时，非步行行为产生的加速度变化干扰，导致步数误判增加；另一方面，缺乏对步伐间隔时间的有效约束，使得短时间内的连续抖动等情况可能被错误计为多步。

解决方法：对 StepDetector 类进行优化。在 detectStep 添加了时间约束条件(timestamp - lastStepTime) > STEP_DELAY_MS，设定最小步伐间隔为 300

毫秒（STEP_DELAY_MS = 300）。只有当加速度变化率超过阈值 delta > ACCEL_THRESHOLD（ACCEL_THRESHOLD = 10.0f），并且两次检测到符合加速度变化条件的时间间隔大于最小步伐间隔时，才判定为一次有效步伐，进而增加步数。同时，通过 lowPass 方法对加速度数据进行低通滤波处理，去除重力分量和高频噪声干扰，使用于判断步伐的加速度数据更加准确，减少了因外界干扰导致的步数误判，有效提升了步数统计的准确性。

2. 手机状态判断不稳定

问题分析：手机状态判断不稳定，主要是因为判断静止状态的依据过于简单，仅依靠合加速度与重力加速度的差值和固定阈值比较，没有充分考虑到手机实际使用场景中的复杂情况。比如，手持手机时的微小抖动会使合加速度瞬间偏离重力加速度，从而造成静止和移动状态的频繁误判。

解决方法：在 MainActivity 类的 onSensorChanged 方法中，通过优化静止状态判断逻辑来解决该问题。利用欧几里得距离公式 float _3Da=(float) sqrt(x*x+y*y+z*z);计算三轴加速度的合加速度，然后与重力加速度进行比较 if(abs(_3Da - 9.8) < 1.0f)。其中，阈值 1.0f 是经验值，在一定程度上容忍了微小波动，减少了因微小抖动导致的误判。此外，StepDetector 类中的低通滤波处理也间接辅助了手机状态判断。低通滤波去除了高频噪声，使得计算出的合加速度更能真实反映手机的运动状态，进一步增强了手机状态判断的稳定性。

五、实验结论

通过对加速度传感器相关知识的学习与实践，我深入了解了其分类，包括压电式、压阻式、电容式、伺服式等，以及这些不同类型传感器在汽车安全、游戏控制、图像自动翻转等诸多领域的广泛应用。在实际操作中，我简要掌握了 Android 系统中加速度传感器（TYPE_ACCELEROMETER）的使用，学会编写布局文件和程序文件，实现了获取 X、Y、Z 三轴加速度值，并将其应用于手机状态监测和步数记录的 APP 开发中。

本次实验仍存在一些需要优化的地方。在步数统计功能上，尽管当前采用加速度变化率阈值和最小步伐间隔时间的方法取得了一定效果，但由于不同用户的行走习惯差异较大，且实际使用场景复杂多变，导致步数统计仍存在误差。经查阅资料得知，可以尝试引入滑动窗口算法，通过对一段时间内的加速度数据进行综合分析，平滑数据波动，更准确地捕捉步伐特征。

在手机静止状态判断方面，目前仅依据合加速度与重力加速度的差值进行判断，在手机受到微小抖动时容易出现误判。为解决这一问题，查阅得知，可以引

入陀螺仪数据。陀螺仪能够测量物体的角速度，与加速度传感器数据相互补充。通过融合加速度传感器和陀螺仪的数据，综合判断手机的运动状态，能够更准确地区分静止和移动状态。例如，当加速度传感器检测到合加速度接近重力加速度，但陀螺仪检测到有一定角速度时，可判断手机并非处于静止状态。此外，还可以探索更复杂的算法，如卡尔曼滤波算法，对多传感器数据进行融合和优化，进一步提高静止状态判断的准确性。

在数据处理环节，当前使用的一阶低通滤波算法虽然能在一定程度上去除高频噪声，但滤波效果仍有提升空间。可以进一步优化滤波系数的选择，通过大量实验和数据分析，找到更适合不同使用场景的滤波系数，以更好地平衡滤波效果和数据实时性。同时，也可以研究和尝试其他更先进的滤波算法，如巴特沃斯滤波、维纳滤波等，这些算法在不同的信号处理场景中具有独特优势，有可能更有效地去除噪声干扰，为后续的加速度数据处理提供更准确的基础。