班级 021161

学号 02116026

西安电子科技大学

本科毕业设计论文

题　　目　基于三轴加速度传感器的手指运动参数检测系统硬件设计

学　　院　电子工程学院

专　　业　电磁场与无线技术

学生姓名　沈云彬

导师姓名　余柏生 张伟涛

摘 要

手指运动参数的检测可以应用于很多领域，例如手指的康复治疗、机器的操控、游戏控制等。目前，手指运动检测的方法大概有测角仪、加速度传感器、陀螺仪、磁传感器、摄像技术等。虽然方法很多，但是也存在很多问题需要解决，其中问题之一就是算法都过于复杂，一般都无法在单片机或者低成本设备上完成运算。为了解决这个问题，本文提出了一种比较可行的方案，可在单片机上实时完成对手指运动参数检测和数据处理。

本文通过利用 ADXL345 三轴加速度传感器进行手指运动参数检测，创新地采用简化的算法来实现单片机对手指运动参数的检测，并传输给上位机，通过 VB.NET 程序开发平台显示并验证算法的精确性。该方法包含了两种算法，分别适用于简单不连续运动和连续运动的情况下的位移计算。算法大致主要包括以下几部分：加速度的滤波、加速度去趋势项、加速度积分得到速度位移、速度位移去趋势项、速度位移实时校正。最后，为了验证算法的可行性，我们在不同的情况下对算法做了验证工作。

本文提出的算法基本上可以除去重力加速度影响，并能检测手指运动的位移和倾角，具有实时修正的功能以保证数据的精确性。系统采用数据无线传输功能，可以方便地进行手指运动参数检测，具有良好的应用前景。

关键词：　手指运动参数　位移　倾角　检测　单片机

ABSTRACT

Detecting a finger motion parameters can be applied in many fields, such as rehabilitation of the finger, machine control, game control. Currently, the method of detecting finger movement about goniometer, acceleration sensors, gyroscopes, magnetic sensors, camera technology. Although many ways, but there are many problems to be solved, one of which is the arithmetic problems are too complex, are generally unable to complete the operation on a low-cost single-chip or device. To solve this problem, we propose a more viable option, so that the finger motion parameter detection can be done on a single chip for processing data.

By utilizing ADXL345 three-axis acceleration sensor finger motion parameter detection using innovative methods to achieve simplification complete detection of finger motion parameters of the microcontroller, it is displayed on the PC VB.NET program and verify the accuracy of the algorithm. The method includes two algorithms were applied to a simple displacement discontinuity motion and continuous motion case calculation. The algorithm basically includes the following sections: acceleration filtering, acceleration to trend item, obtained by integrating velocity displacement acceleration, velocity displacement to trend item, velocity displacement time correction. Finally, in order to verify the feasibility of the algorithm, we have different circumstances to do verification algorithm.

The proposed algorithm can remove the basic influence of gravity and detect finger movement and displacement angle, real-time correction features to ensure accuracy of the data, and a wireless transmission function, you can easily detect finger movement parameters, it has a good application prospects.

Keywords: finger movement detection angle displacement parameters SCM

目 录

第一章　绪 论

1.1　手指运动参数检测的意义及发展现状

手指是人类最灵活也是用的最多的器官之一，我们可以通过手指运动方便地完成很多事。因此，手指运动参数检测可以应用于很多领域例如手指的康复治疗、机器的操控、游戏控制等。

目前，用于人体运动检测的方法大致有利用测角仪、加速度传感器、磁传感器、摄像技术等。测角仪的原理是在人体上绑上测角仪或者穿上带有传感器的衣服，通过测量关节的运动角度来获取运动参数，比如角速度和角加速度等。不过，这种方法对于不同的人需要不同大小的装置和参数，通用性会有一定的限制，而且体积比较庞大，并不是很方便。磁传感器可以测量倾角和方位，精度也比较高是一个不错的方法，然而容易受到外界的干扰。摄像技术亦可以检测手指运动轨迹，采用摄像机记录人体运动，形成一系列图像，通过图像识别就可以分析人体的运动。但是，这种方法也有一个缺点，当人体的一部分被遮挡的时候就不能识别那部分的运动参数，而且算法也相对很复杂，并不适用于低端的设备。加速度传感器的原理就是通过测量重力加速度与三个坐标轴的夹角来测量手指的倾角，并且利用加速度积分后来计算速度和位移。

随着计算机技术、穿戴设备和微电子技术的发展，目前，手势交互也是人机交互中研究中的热点话题，与传统的交互模式相对比，前者更加自然和便捷。手势交互可以实现一些列的交互游戏，并且在一些比较复杂的设备例如机械手、飞行器等的控制上也占有很大的优势。手指运动参数检测作为手势检测的一个基础，还是很有发展和研究的意义。

1.2　基于三轴加速度传感器运动参数检测研究的进展

目前国内对于利用三轴加速度传感器研究运动轨迹的论文并不是很多，对于三轴加速度传感器的应用大多还停留在物体倾角的检测上，即通过测量三轴加速度

传感器三轴对重力的夹角来获得运动姿态。其次，也有利用三轴加速度传感器检测简单的直线式运动轨迹，通常是在一个水平面上，只需要做一些简单的处理即可，因为位置相对比较固定，因此数据处理也相对比较简单。

除此之外就是利用三轴加速度传感器检测空间运动轨迹了，对于利用三轴加速度传感器检测空间运动轨迹就相对比较困难了，因为有传感器倾角和重力加速度的影响，导致数据处理很复杂，一般来说主要由以下三个流程。首先，对加速度传感器输出的数据进行平滑滤波处理进行去毛刺，常用的方法有 5 点 3 次平滑法，利用快速傅里叶变换低通滤波等。第二，对于处理后的加速度进行去趋势项处理，以消除重力加速度及模块倾角的影响。第三，将处理过的加速度数据在一定频域范围内进行频域积分，再转化为速度和位移。这种方法相对来说可以比较精确地计算出运动轨迹，并对于连续的运动的检测还是比较精确的。

但是这样处理数据依旧还存在很多问题，其中之一就是快速傅里叶变化的计算量很庞大，然后需要取比较长的一段时间的数据才能达到预期的效果，因而导致利用计算机实时处理数据也显得非常困难，更不能在运行内存和频率都很受限制的单片机上运用了。其中之二的话就是数据处理往往是在电脑上进行的，而且是一种先采集数据后再慢慢进行事后的数据处理，并没有很好地与实际应用中的实时要求相结合，也不能做到很便携。

1.3 本文研究的内容及意义

本文通过利用 ADXL345 三轴加速度传感器进行手指运动参数检测，创新地采用简化的算法来实现单片机对手指运动参数的检测，并传输给上位机，通过 VB.NET 程序开发平台显示并验证算法的精确性。该装置成本低廉、运行高效，可以除去重力加速度影响并检测手指运动的位移和倾角，有实时修正的功能以保证数据的精确性，并具有无线传输功能，可以方便地进行手指运动参数检测。具体的研究内容分为以下几个部分：

1. 完成基于 ADXL345 三轴加速度传感器的手指运动参数检测模块的硬件设计
2. 改进基于单片机的运动参数处理算法，使运动参数可以在单片上运算并传输到 PC 计算机

3. 完成对上位机 VB.NET 程序的设计，使运动轨迹可以直观的显示出来

本文研究的意义在于，本文提出了一种简单化可以在单片机上运行的算法来精确地计算手指运动参数，该算法不仅占用较少单片机资源，并且在不连续动作的检测上相对于其他方法更精确，可以很方便地应用于一些小型化低成本的设备上，具有很好地应用前景。

第二章　手指运动参数检测系统的设计

2.1　手指运动参数检测系统硬件组成及工作原理

手指运动参数检测系统由上位机和下位机组成。下位机手指运动参数检测模块由 ADXL345 三轴加速度传感器、Arduino Nano 单片机、CC1101 无线串口透传模块和干电池组成的电源模块组成。具体下位机框架图如下图 2.1 所示。

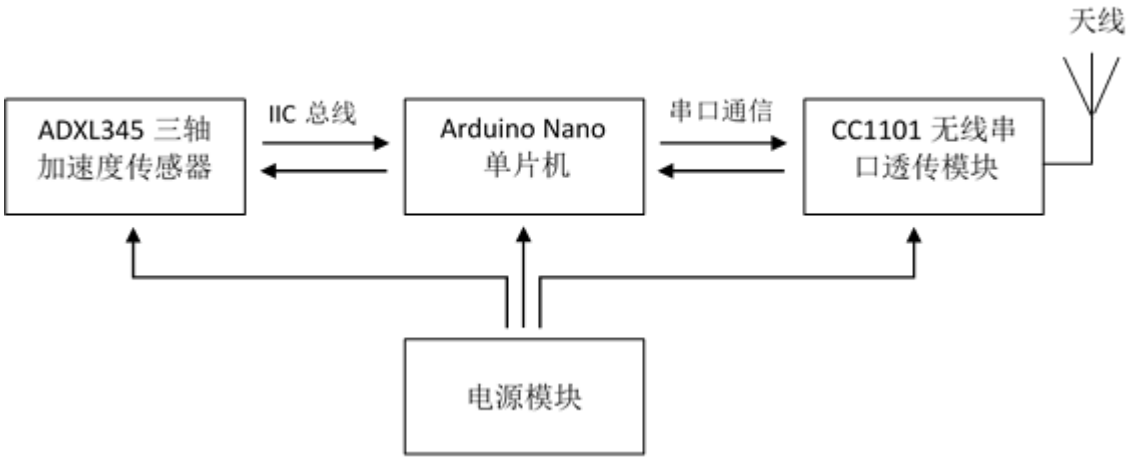

图 2.1 手指运动参数检测系统下位机总体框架图

其中 Arduino 单片机是系统的主要组成部分，它负责以 50Hz 的速度使能 ADXL345 加速度传感器模块采集数据，然后对加速度传感器的数据进行分析处理和计算，接着通过串口输出数据。ADXL345 加速度传感器模块主要负责把 X、Y、Z 轴三个方向变化的加速度信号转换成数字信号并通过 IIC 总线传输给单片机处理。CC1101 无线串口模块是将串口数据转换成 433MHz 的无线信号通过向空间发射出去，实现模块的无线通信功能。

另外，上位机系统主要由 CC1101 无线串口透传模块、PL2303 串口通信模块、计算机 VB.NET 程序组成，主要负责数据的接收、记录和显示功能，将手指运动参数检测模块检测的数据储存下来，并输出运动轨迹。硬件组成如下图 2.2 所示。

图 2.2 手指运动参数检测系统上位机总体框架图

其中 CC1101 无线串口透传模块负责接收无线信号并转换成串口通信的方式将数据传输到 PL2303HX 模块中，PL2303HX 再将串口数据传输到 USB 口，上位机 VB.NET 程序主要负责采集数据并显示轨迹。

2.2 手指运动参数检测系统主要模块介绍

2.2.1 Arduino Nano 单片机模块

MCU 是整个手指运动参数检测系统的核心部件，其单片机的性能在很大程度上决定了系统的工作能力，因此选择合适的单片机是很关键的，单片机具体选型原则是：

（1）单片机功耗要低：由于手指运动参数检测模块是由电池供电的，因此芯片要适合于电池供电，并且续航能力强。

（2）单片机芯片封装小：简单说就是要求单片机的外观和体积要尽可能的要小，这样就可以减小单片机所占用的体积，整个系统就可以做的很小，既方便携带，同时也能方便应用和测量。

（3）处理速度快，运行内存容量大：因为首先数据处理是在单片机上运行的，而且三轴加速度传感器输出的数据需要滤波积分等处理，需要占用大量的运行内存，接着数据是实时采集实时计算的，于是对于单片机的运算性能也有很高的要求。

基于上述条件，单片机选用 Arduino Nano v3.0 单片机来构建系统。Arduino Nano 是 Arduino 单片机 USB 接口的微型版本，但是没有电源插座。Arduino Nano 的尺寸非常小的，它的最大尺寸为 0.73 x 1.7 英寸，而且可以直接插在电路板上使用。其处理器核心是 ATmega328，它的 IO 脚直流电流为 40 mA，同时具有 14 路数字输入/输出口，并且其中 6 路可作为 PWM 输出。它具有 8 个 10 位精度的模拟信号

输入口，并且可以大致以 100Hz 的速度采集模拟信号。内置一个 16MHz 速率的晶体振荡器，一个 mini-B 型号的 USB 口，一个 ICSP header 并带有一个复位按钮。它具有 32KB 的闪存用于存储程序和 2KB 的内存用于存储变量。其他方面，它还具有一个 IIC 接口、一个硬件串口和两个外部中断，并且可设置成上升沿、下降沿或同时触发。单片机采用的是精简指令集，因此与 51 单片机相比具有更加良好的运算性能。

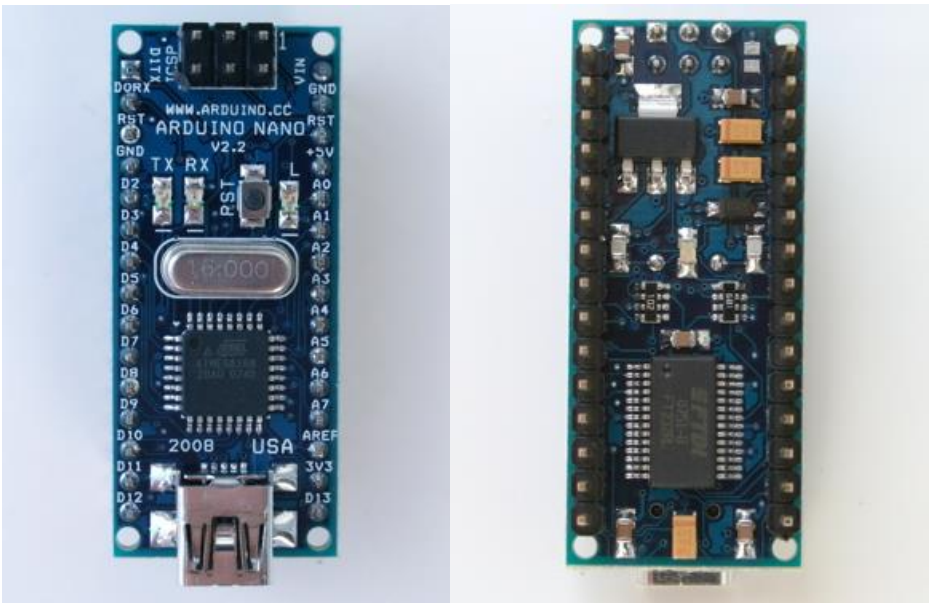

图 2.3 Arduino Nano v3.0 单片机

2.2.2　ADXL345 三轴加速度传感器

ADXL345 三轴加速度传感器是一款小体积的超低功耗 3 轴加速度传感器计，可以测量的范围达±16g，分辨率高达 13 位。数字输出的数据为二进制补码的格式，可通过 SPI 或 I2C 数字接口进行访问。因此 ADXL345 非常适合在移动设备中应用。

首先，它可以测量静止状态的重力加速度，其次，它还可以测量运动或冲击导致的动态加速度。它具有极高的分辨率，能够测量出不到 1.0° 的倾斜角度变化。该器件有两个中断引脚，可以提供多种特殊动作检测功能。具体如下：

（1）活动与非活动检测：首先用户设置一个阈值来检测有无运动的发生。接着，传感器通过对比任意坐标轴上的加速度测量值与用户设置的阈值来判断是否产生

中断信号。

（2）单击双击检测：单击检测功能是通过将用户设置的阈值与加速度传感器的测量值比较，检测出来任意方向的单振动作。双击检测功能是检测一定时间内是否产生两次单击，并输出中断信号。

（3）自由落体检测：此功能是用于检测自由落体运动，可以用于判断三轴加速度传感器的状态。

这些功特殊检测的能可以独立设置到两个中断输出引脚中的一个。它还有一个 32 级 FIFO 缓冲存储器，可用于存储数据，从而可以提高采样频率的稳定性，并且降低处理器的负荷提高系统的稳定性。在本次应用中，ADXL345 数字三轴加速度传感器模块使用 IIC 总线传输数据，具体接线方式如下图所示。

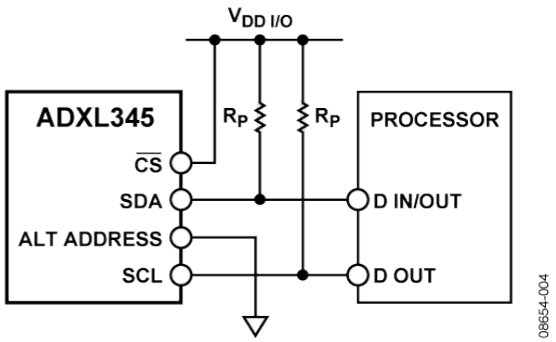

图 2.4 ADXL345 数字三轴加速度模块连线方式

2.2.3 CC1101 无线串口通信模块

CC1101 是 Chipcon 公司（现已被 TI 收购）推出的一款单片射频的 UHF 收发器。该芯片具有较低的成本，并且有较低的功耗。CC1101 可以设置工作在 315、433、868 和 915MHz 的频段，芯片中集成了一个软件可编程的调制解调器，数据传输速率最高可达 500kbps。主要应用于需要低功耗但对速率要求不高的系统中。

汇承 CC1101 串口透传模块基于 CC1101 模块构成。CC1101 串口透传模块工作频段为 433MHz，有效传输距离大于 100 米。该芯片具有低电流消耗，空闲电流可以设置为 80uA 或 3.5mA 或 22mA。模块可以设置 127 个信道，串口波特率可以设置为 1200、2400、4800、9600、38400、57600、115200。当波特率为 115200

时，一次最多只能发 245 个。模块的优点是用户无须对模块进行编程，只需要管收发串口数据就可以， 使用比较方便。

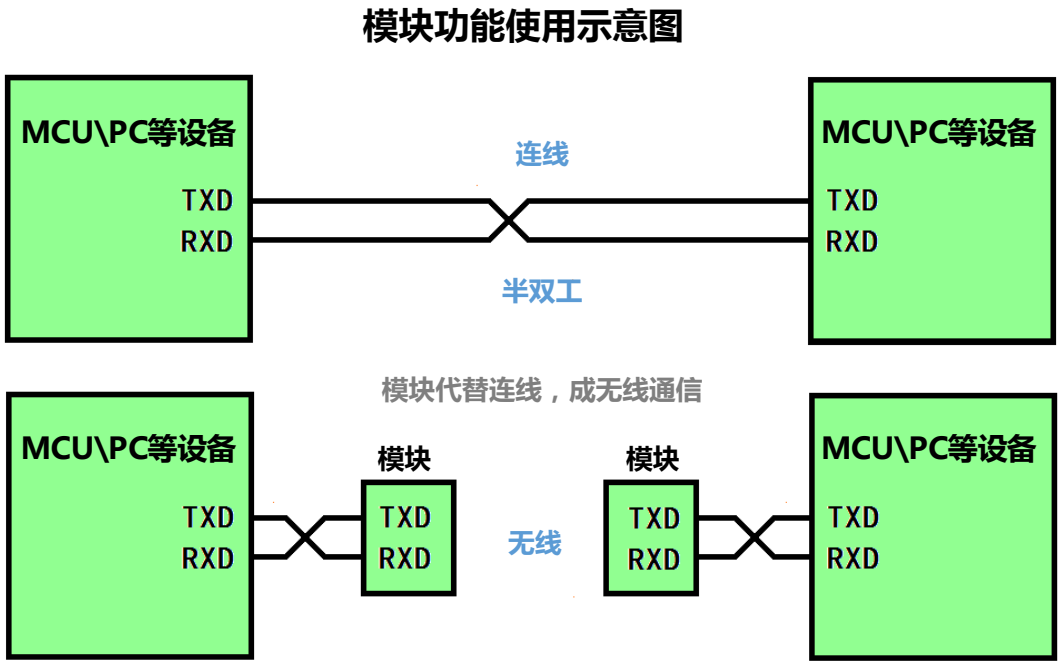

图 2.5 CC1101 无线串口透传模块连线图

模块的连线如图 2.5 所示，首先左边 MCU、PC 等设备向模块发送串口数据，左边模块从 RXD 端口收到数据后， 将收到的数据通过电磁波发射出去。 右边的模块可以自动接收无线信号，并自动处理并从 TXD 口输出原始的串口数据，反之亦然。在手指运动参数检测系统中我们使用了 115200 的波特率来通信，以数据保证高传输效率。

2.2.4　PL2303HX 串口通信模块

PL2303HX 串口通信模块是 Prolific 公司旗下的一种 RS232-USB 接口转换器，可以在计算机中建立一个虚拟串口，让 RS232 串行通信的数据通过计算机的 USB 口传输，从而实现计算机与单片机的串口通信。PL2303HX 的结构图如图 2.6 所示，因为已经内置 USB 控制器、收发器、振荡器和可以适应各种波特率的 UART，所以可以轻松的实现 USB 信号与串口信号的转换。作为一个 USB/RS232 的双向传输转换器，双向信号转换的工作可以全部由 PL2303HX 自动完成，在实际应用中无需再考虑其他软件设计，因此该器件也经常被一些组织推荐使用。

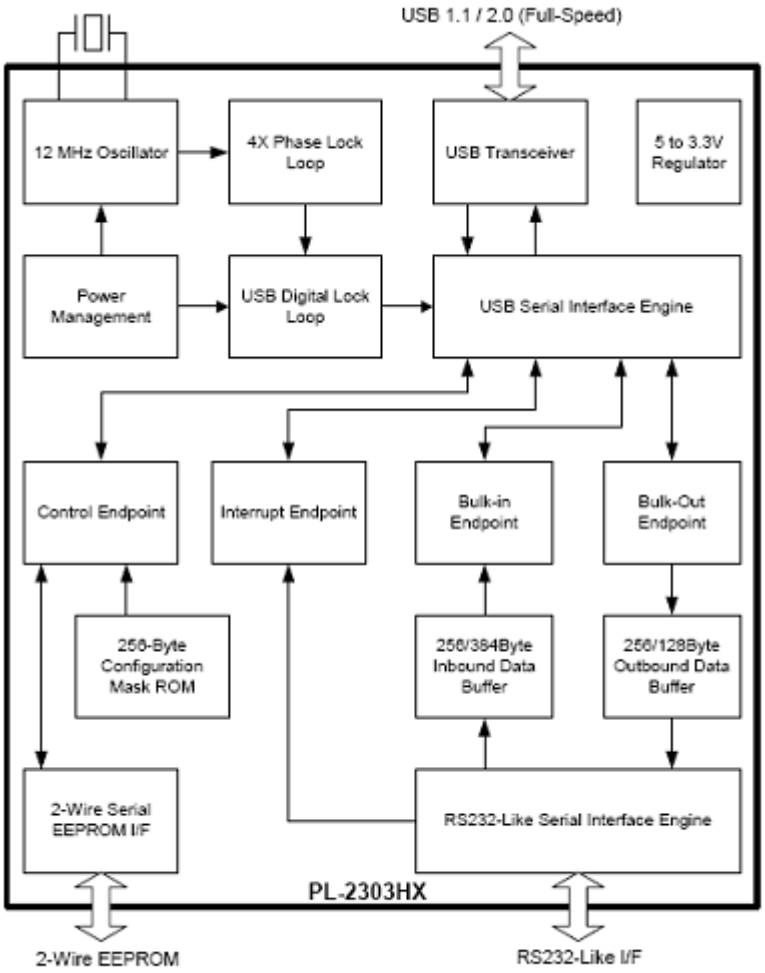

Figure 2-1 Block Diagram of PL-2303HX

图 2.6 PL2303HX 内部机构

PL2303HX 具有多个驱动，并可以可在操作系统上形成模拟 COM 端口进行串口通信，通讯波特率可以高达 6 Mb/s。该器件功耗低，是微型或者小型化设备的的理想选择。PL2303HX 具有以下几个特性：

(1) 可以兼容 USB1.1\USB2.0 协议

(2) 同时具有 5V 和 3.3V 电压输出

(3) 可在软件中自行设置的波特率：75b/s～6 Mb/s

(4) 支持 Windows98 以上操作系统

2.3　上位机 VB.NET 程序任务与检测模块无线通信设计

2.3.1　上位机 VB.NET 程序的工作任务

上位机 VB.NET 程序的主要任务是采集下位机所输出的数据，并进行记录和直观的显示出传感器的状态。上位机 VB.NET 程序的框图如图 2.7 所示。

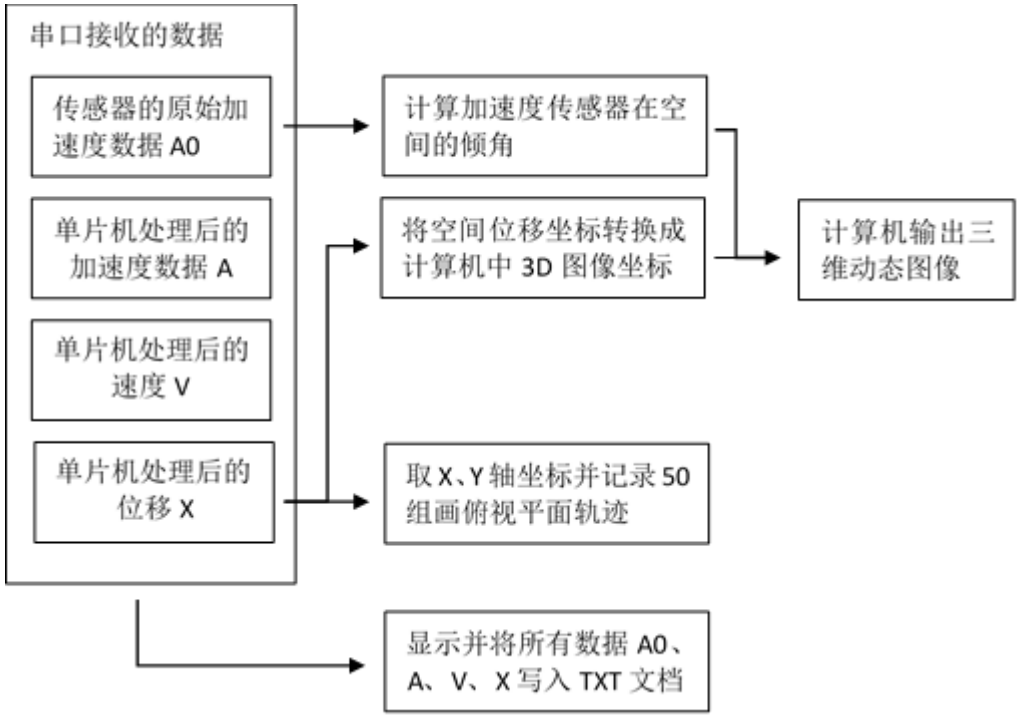

图 2.7 上位机 VB.NET 程序结构功能框图

上位机 VB.NET 程序的功能包括以下几点：

(1) 在三维图像中显示三轴加速度传感器的位置

(2) 在三维图像中显示三轴加速度传感器的倾角

(3) 显示三轴加速传感器在俯视平面的运动轨迹

(4) 显示三轴加速度的现在的空间坐标

(5) 显示三轴加速度传感器的速度变化

(6) 记录单片机所输出的全部数据

上位机 VB.NET 程序的具体工作流程如下，首先从串口接收数据，数据包括传感器的原始加速度数据 A0、单片机处理过后的加速度数据 A、速度数据 V，位移数据 X。然后，将三维坐标数据即位移数据进行三维坐标到二维图像的转换，并输出计算机上 3D 图像中的坐标。接着，从原始加速度数据可以得到传感器 X、Y、

Z 轴对重力加速度的夹角，进行计算即可得到三轴加速度传感器的倾角。之后，将三轴加速度传感器位移和倾角数据整合在一起画出三轴加速度传感器的实时图像。最后，再将单片机传来的数据写入 txt 文档并在软件上显示三个坐标轴的速度变化、传感器所在位置坐标和运动轨迹。

上位机 VB.NET 程序运行界面如下面图 2.8 所示。上位机的操作界面中，左上角用于设置串口参数，左边中间用于设置记录数据的 txt 文件储存的位置，左下方则用于显示串口所接收的数据。界面中间部分显示三轴加速度传感器的状态，包括位置和倾角。最右边显示三轴加速度传感器 X、Y、Z 轴的速度变化曲线和 XY 平面的移动轨迹。最后，为了提高程序的运行速度，中间的画图部分和右边的画图部分被设置成分别在两个线程上运行。

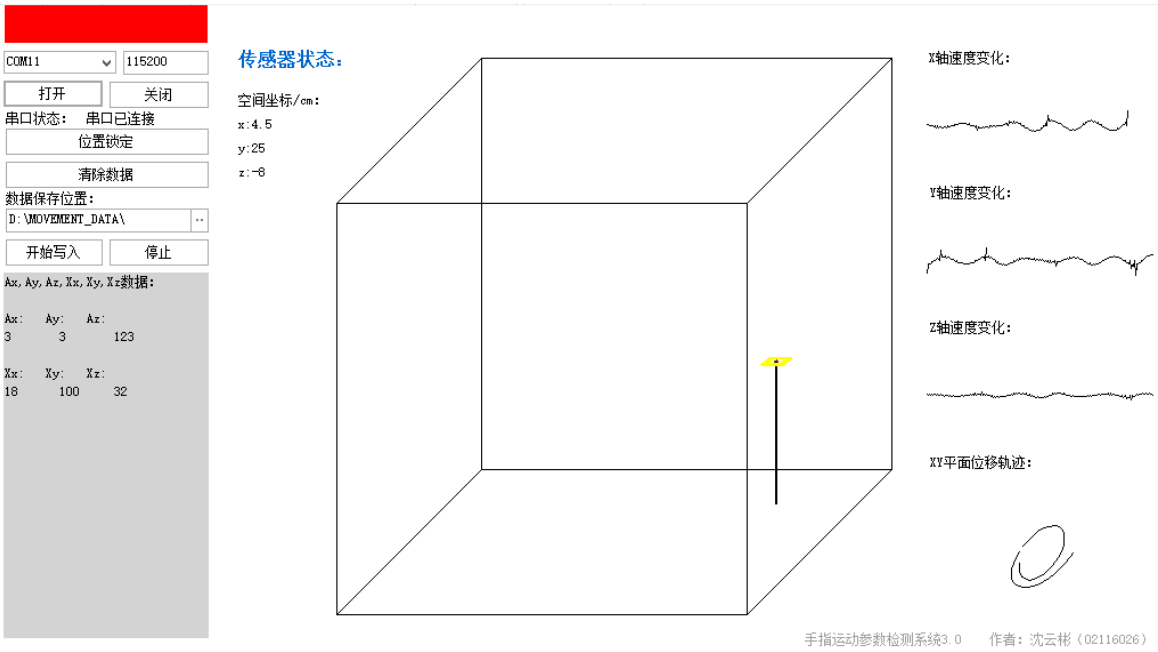

图 2.8 上位机 VB.NET 程序界面图

2.3.2 上位机 VB.NET 程序串口通信设计

计算机通信有并行通信和串行通信两种通信方式。一般情况下，在多微机系统中，信息的交换大多数采用串行通信的方式。串行通信是指将原始数据的字节分成一位一位的形式通过一条传输线逐个逐个地发送。串行通信的好处是传输线少，成本低并且连接简单。异步通信是指通信的发送设备与接收设备使用各自的时钟来控制处理数据的发送和接收。同时，为了使双方的收发信号协调，发送和接收设备的时钟需要尽可能一致，不然就可能产生乱码。

在手指运动参数检测系统中的串口通信是一种串行异步通信机制。它的数据格式如下面图 2.9 所示。一个字符帧由起始位、数据位、校验位、停止位组成。其中校验位分为奇校验和偶校验，一般在数据的最后一位，即用一个值用来确保传输的数据当中有偶个或者奇个逻辑高位，这样就可以判断是否有噪声干扰了通信或者是否传输和接收数据是否不同步。

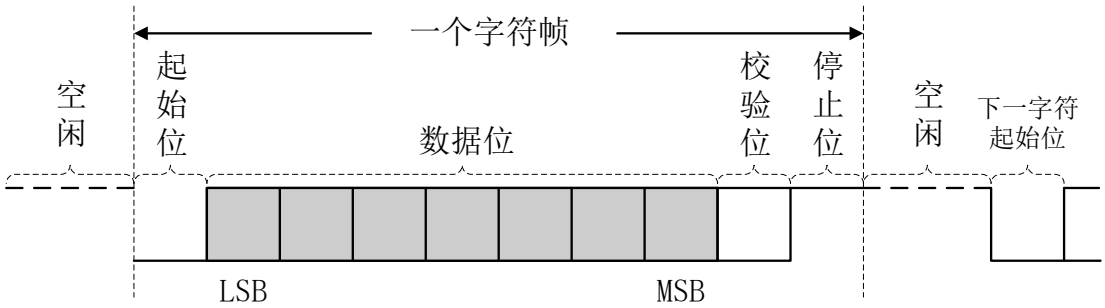

图 2.9 串口通信的数据格式

在手指运动参数检测系统中，需要每 20ms 传输 12 组数据包括 X、Y、Z 轴的原始加速度和处理后的加速度、速度、位移。由于传输的数据量较大，所以使用的是波特率 115200，以降低在串口通信中所消耗的时间，以提高软件的运行效率。同时，为了保证数据传输的准确性，同时还对串口数据进行偶校验，并设置其停止位为 2 位。这样，在一定程度上减少了乱码的概率，增加了串口通信的质量。

单片机所输出的数据格式是由 26 个字符组成，包括一个数据起始标志字符、24 个用于储存数据的字符和一个结束标志字符。数据具体排列格式如下图 2.10 所示。起始标志字符和结束标志字符用于检测数据的准确性，如果数据检测结果为准确，才将数据进行处理并读入程序。

图 2.10 单片机输出的数据格式

VB.NET 中的串口通信是相对比较容易实现的。首先，VB.NET 中串口通信常用的有两种方法，一种是用 API 函数自己来编写串口通信程序，这种方法相对比较复杂，但是可以拥有更强大的功能。另一种方法是直接使用官方的控件，这样就

比较简单了。在 VS2005 以前的版本中，串行通信所使用的控件是 Mscomm，而在 VS2005 以后的版本中该控件被升级为 SerialPort。SerialPort 功能十分强大，完全可以满足一般的开发需要。SerialPort 控件所在的类是 Microsoft 最新在 Microsoft .Net Framework 2.0 中引进的。控件的主要属性和驱动事件有下面几个部分：

BaudRate：设置串口的波特率。

DataBits：设置串口数据位，即一个字符由几个位组成。

Parity：设置校验方式，有 None、Odd、Even、Mark、Space 这么几种。

PortName：设置串口号。

Read：从串口接收缓冲区读数据。

ReadBufferSize：设置接收缓冲区的大小，本系统中为默认值。

ReceivedBytesThreshold：当接收到若干个字符，就触发 DataReceived 事件。

StopBits：设置停止位大小。

Write：向发送缓冲区写入数据。

WriteBufferSize：设置发送缓冲区的大小。

DataReceived 事件：在此事件中可以对数据进行处理。

在实际应用中，需要先获取虚拟通信端口，再设置波特率校验性、停止位、缓冲区大小。串口初始化的代码如下面所示：

```
SerialPort1.BaudRate = Val(baudratebox.Text) '设置波特率
SerialPort1.PortName = portnamebox.Text '获取串口名称
SerialPort1.DataBits = 8 '设置数据位为 8 位
SerialPort1.StopBits = IO.Ports.StopBits.Two '设置停止位为 2 位
SerialPort1.Parity = IO.Ports.Parity.Even '设置校验位为偶校验
```

由于串口通信是异步通信，如果每一次接收都要判断接收数据是不是起始位或者结束位，这样就会大大降低接收的效率以至于无法跟上单片机的发送速度。但是，如果不是每一个字符地检测，我们不知道数据的开头和结束是在哪里，所以为了提高接收的数据的准确性，可以将 DataReceived 事件中的触发方式设置为每接收 26 个字符触发，接着再判断接收的数据的准确性，并且在处理后清除接收缓冲区的数据。程序代码如下：

```
SerialPort1.ReceivedBytesThreshold = 26 '触发方式设置为每接收 26 个字符触发
```

```
Public Sub Sp_DataReceived()  Handles SerialPort1.DataReceived
    Me.Invoke(New EventHandler(AddressOf Sp_Receiving)) '调用接收数据函数
    SerialPort1.DiscardInBuffer() '清除接收缓冲区数据
End Sub
```

这样以后，就大大减少了对接收缓冲区的读写操作，极大地提高了程序的效率，即使产生了一次错误的数据，也不会影响下一组数据的接收。

第三章　基于单片机的手指运动加速度数据处理原理及要求

3.1　加速度传感器测量手指运动参数的基本原理

3.1.1　加速度传感器测量倾角

ADXL345 三轴数字加速度传感器可以测量 X、Y、Z 轴上的加速度分量，当模块处于静止状态时，不同倾角的情况下，由于重力加速度方向是不变的，因此三个坐标轴的加速度分量是不相同的。所以，利用重力加速度与其在三轴加速度传感器的 X、Y、Z 三轴的分量关系，就可以计算出各轴与重力加速度的夹角，从而得出系统倾角。如图 3.1 所示，描述了几种不同的状态下，三轴加速度传感器坐标轴与空间原先确定的标准坐标轴之间夹角的情况。

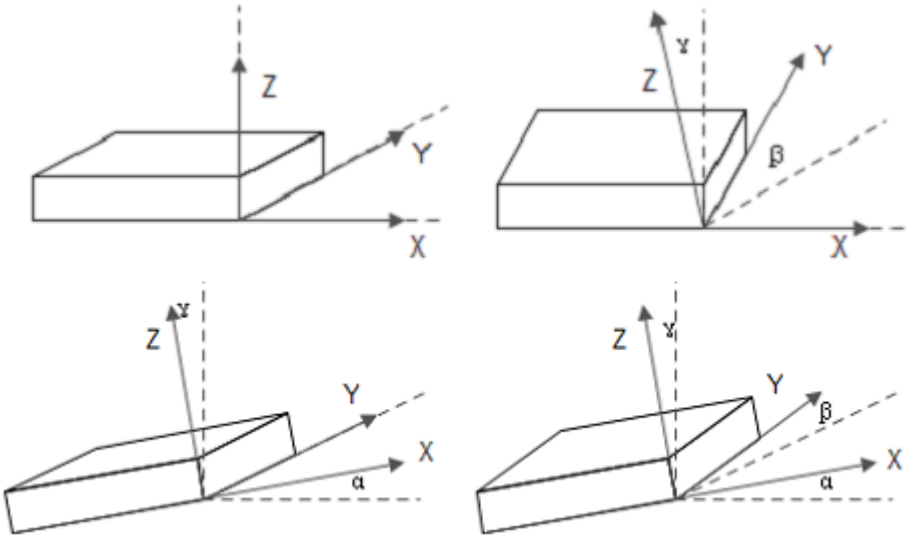

图 3.1 ADXL345 三轴加速度分量与倾角关系

根据物理的力学分析和三角函数，我们可以得到，X、Y、Z 轴与原先的标准坐标轴的夹角α、β、γ的计算公式为：

$$\alpha = \tan^{-1}\left\{\frac{A_x}{\sqrt{A_y^2+A_z^2}}\right\} \qquad (3\text{-}1)$$

$$\beta = \tan^{-1}\left\{\frac{A_y}{\sqrt{A_X^2+A_Z^2}}\right\} \quad (3\text{-}2)$$

$$\gamma = \tan^{-1}\left\{\frac{A_z}{\sqrt{A_X^2+A_y^2}}\right\} \quad (3\text{-}3)$$

根据上面计算得到的三个坐标轴与原先的标准坐标轴夹角的值，就可以得到三轴加速度传感器在空间倾斜角度和方向。从而就可以得到手指的姿态。

3.1.2 加速度传感器测量位移

从三轴加速度传感器可以直接得到三个轴的加速度数据，但是由于有重力加速度的影响，用三轴加速度传感器测量位移并不是一件容易的事情。往往需要将加速度数据经过多步处理后才可能得到想要的结果。

一般来说加速度传感器测量位移的误差主要来自于以下几个方面：

(1) 三轴加速度传感器本身就有一定的误差

(2) 三轴加速度传感器检测出的结果是离散的，因此无法很精确的计算

(3) 离散的加速度参数经过双重积分运算才能算出位移，计算过程难免会有误差

(4) 实际应用中三轴加速度传感器难免会有倾角，从而造成测量值产生一定的偏移量

(5) 人的手是在不停抖动的，再加上三轴加速度传感器输出的参数本来就有抖动，从而会导致在计算的时候会产生较大误差

因此，利用三轴加速度传感器测量位移需要经过多个步骤处理数据。首先，需要去除加速度数据的抖动，即消除高频分量。第二，去除三个轴加速度的趋势项，以消除传感器本身的影响和其空间倾角对加速度结果的影响。第三，利用加速度计算速度和位移。第四，同时需要根据加速度数据的特点实时校准速度和位移数据。在手指运动参数检测系统采用的数据处理步骤如图 3.2 所示。

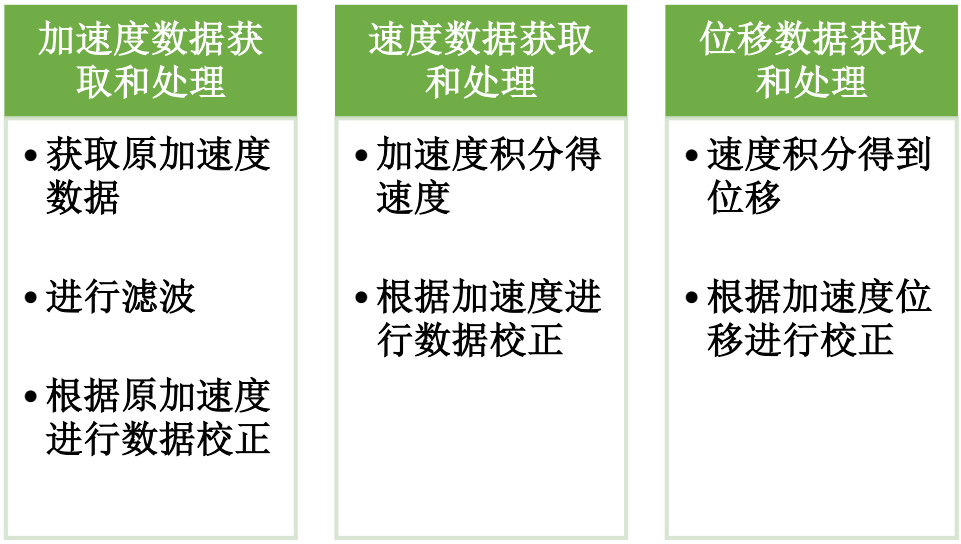

图 3.2 手指运动参数检测系统采用的数据处理步骤

除此之外，手指运动参数检测时还会有一定的误差偏移，特别是在做连续运动的时候，由于手指对三轴加速度传感器倾角不能很好的控制，偏移会变得很大，因此在测量连续动作时需要进一步去除偏移量来达到数据的精确性。

3.2　手指运动检测数据处理的要求

3.2.1　数据处理对单片机的要求

由于三轴加速度传感器的加速度数据在进行处理时需要滤波和积分运算，因此对于单片机的要求还是很苛刻的。总体来说，数据处理对单片机的要求主要有以下两方面。

第一，数据处理中对单片机运行内存的要求。由于加速度原始数据的处理中需要去除低频趋势项，一般而言需要去除 0.5Hz 以下频率的由于加速度传感器角度倾斜或者自身误差产生的趋势项，然而根据 50Hz 的采样频率下至少需要 100 个数据才能实现单个坐标轴数据的滤波。因此，光是三个坐标轴的加速度数据将占用 300 个数据，按整型数据占两个字节来算，至少需要 600 个字节的运行内存。因此，对于单片机的运行内存有着很高的要求，一般而言程序至少需要占用 1200 字节的运行内存。

第二，数据处理中需要用到储存数据和滤波处理，如果采用一般的快速傅里叶

变换，根据傅里叶变换的计算次数公式 N+2*（N/2)^2 就可以得到，对于 50Hz 的采样率，要去除三个数据的 0.5Hz 以下频率，需要至少对 200 个点进行傅里叶变换，此时它的运算次数就达到 3*20200*50=3.03MHz，由于有如此之大的运算次数，在单片机上是不大可能进行傅里叶变换运算的。但是我们可以通过简单的方法来代替傅里叶变换，可以用取多个点的平均相减的方法来实现。在手指运动参数检测系统中采用了取 15 点平均减去 100 个点平均的算法，这样数据的处理量就降到了 3*50*（100+15）=17250Hz 左右，但是这只是单个步骤的处理量，对于这个运算的处理量还是很可观的。对于采用精简指令的单片机来说，能正常运行至少需要 4MHz 的运算速率，对于 8051 之类的单片机要求会更高。

由于以上运算的特性对单片机的要求，我们最终选用 Arduino Nano 来作为平台，它的运行内存是 2k，时钟频率为 16M，可以满足需求。

3.2.2 加速度传感器原始数据的特性

加速度传感器所输出的原始数据存在很多误差和缺陷会对之后的计算造成很大的影响。首先加速度传感器输出的数据是有抖动的，将加速度传感器沿 Y 轴作往返运动以 50Hz 频率所采集的数据，结果如图 3.3 所示，可以看到加速度数据有十分明显的波动。并且 Y 轴加速度并不在原点来回而是有很大的趋势项。

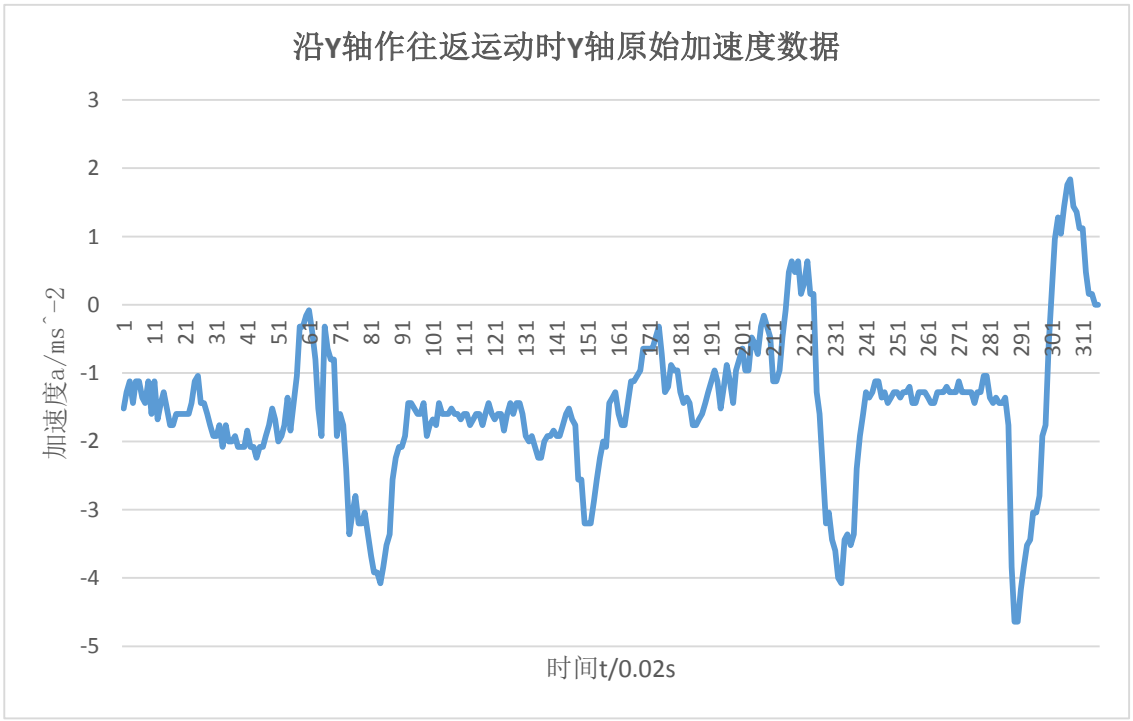

图 3.3 沿 Y 轴作往返运动时 Y 轴原始加速度数据

根据实际情况而言，产生这种情况的主要原因可能包括两个方面，第一是加速度传感器角度倾斜或者自身误差产生的趋势项引起的，第二，可能是由于加速度传感器自身抖动产生的高频分量引起的。为了进一步研究数据的特性，实验中，沿着 Y 轴作往返运动，以 50Hz 频率采样，并对对 2000 多组数据在 Matlab 中进行了频谱分析，最后得到一张 1-300 个点的频谱分析图，如下面图 3.4 所示。通过图像可以直观的看到，前面部分 50/10=5Hz 以上的高频分量对结果产生了较大的影响，同时有用的频率范围应该在第 10-60 个点左右，然后，为了除去低频趋势项，应该去除 0.5Hz 以下的低频分量。

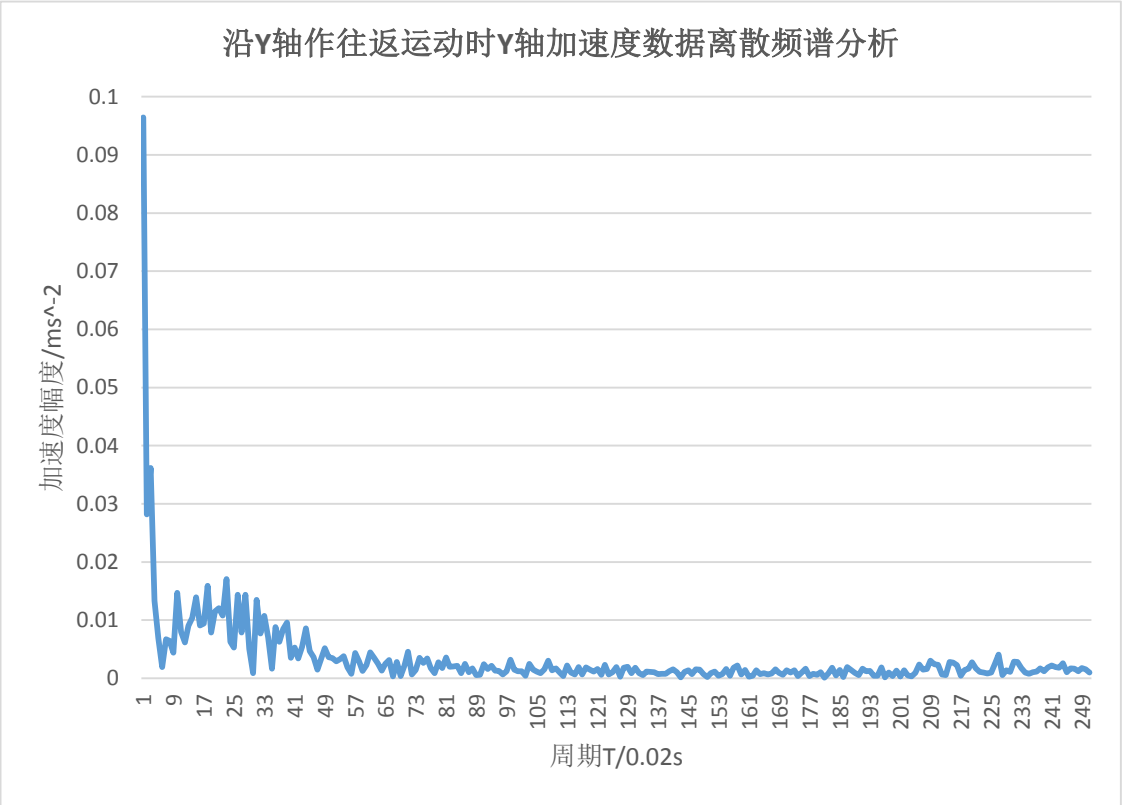

图 3.4 沿 Y 轴作往返运动时 Y 轴加速度数据离散频谱分析

3.2.3　加速度和速度积分的特点以及对数据的要求

由于从加速度数据得到速度和位移需要进行积分和双重积分，因此对积分的精确性要求比较高。加速度积分运算对数据的要求主要有两点，第一是数据需要是平滑的，第二数据的采集频率一定要稳定，且不能产生遗漏。

手指运动参数检测系统中测量的加速度值是离散的，再加上三轴加速度传感器

本身输出数据具有波动性，因此在进行梯形积分运算时，需要去除抖动才能得到相对准确的值，否则，抖动的加速度就会被误认为是平均的加速度，如果采样频率不是完全稳定的话，有些波动就会被扩大影响，从而影响结果的准确性。然后，在数据的开始如果发生抖动抖动的影响会在积分运算的作用下一直持续到一个波形的结束。如图 3.5 中，两组加速度数据分别是抖动的和不抖动的，两者的曲线基本重合。到了图 3.6 两者的速度差别是 0.3，但也并不是很明显。然而，到了图 3.7，两者的最终位移为 0.45 和 0.5，足足有了 10%的误差。从结果可以看出，引起主要误差的还是在开始和结束的部分，试验中仅仅取了 11 组数据，如果是上千组数据，引起的误差是可想而知的。

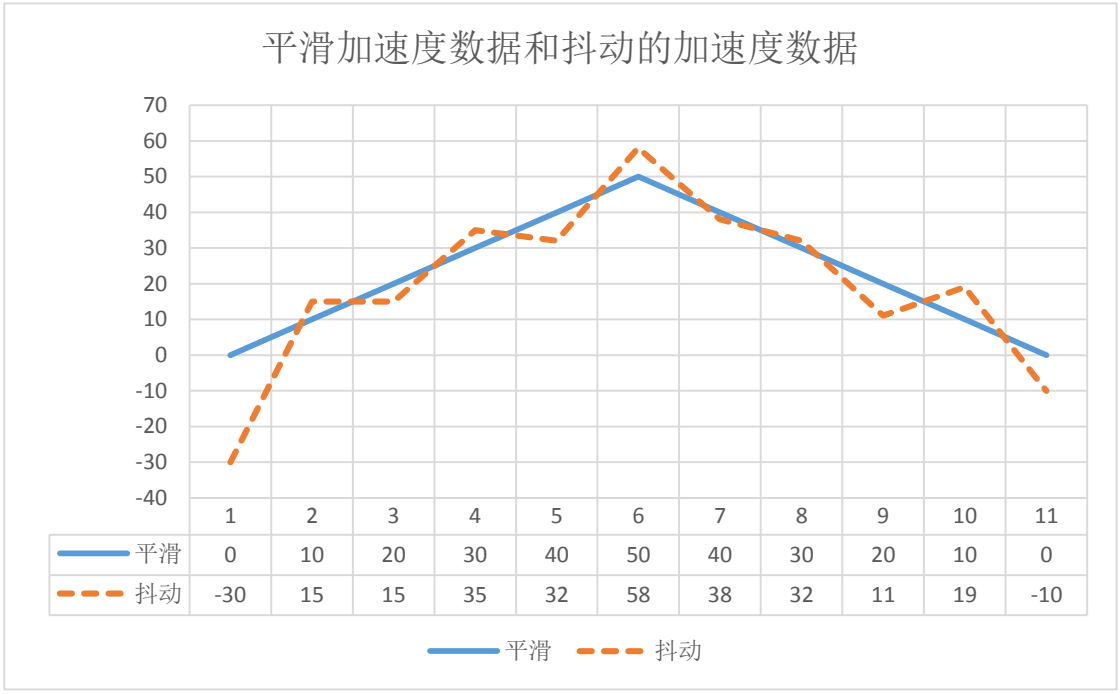

图 3.5 平滑加速度数据和抖动的加速度数据

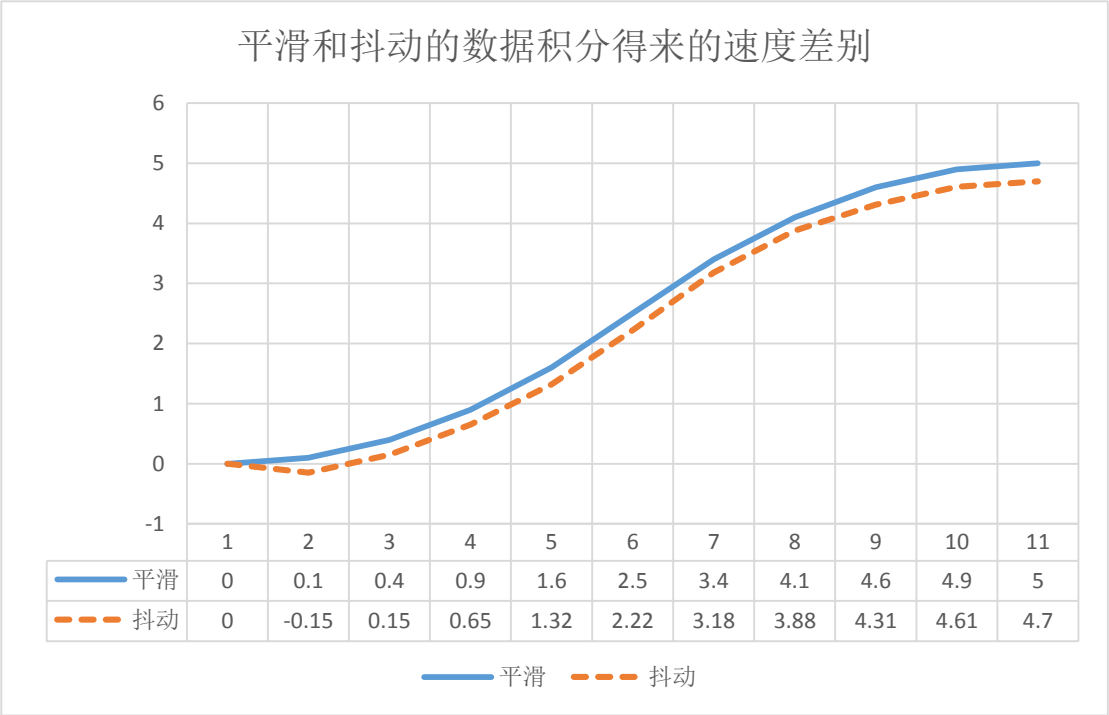

图 3.6 平滑和抖动的数据积分得来的速度差别

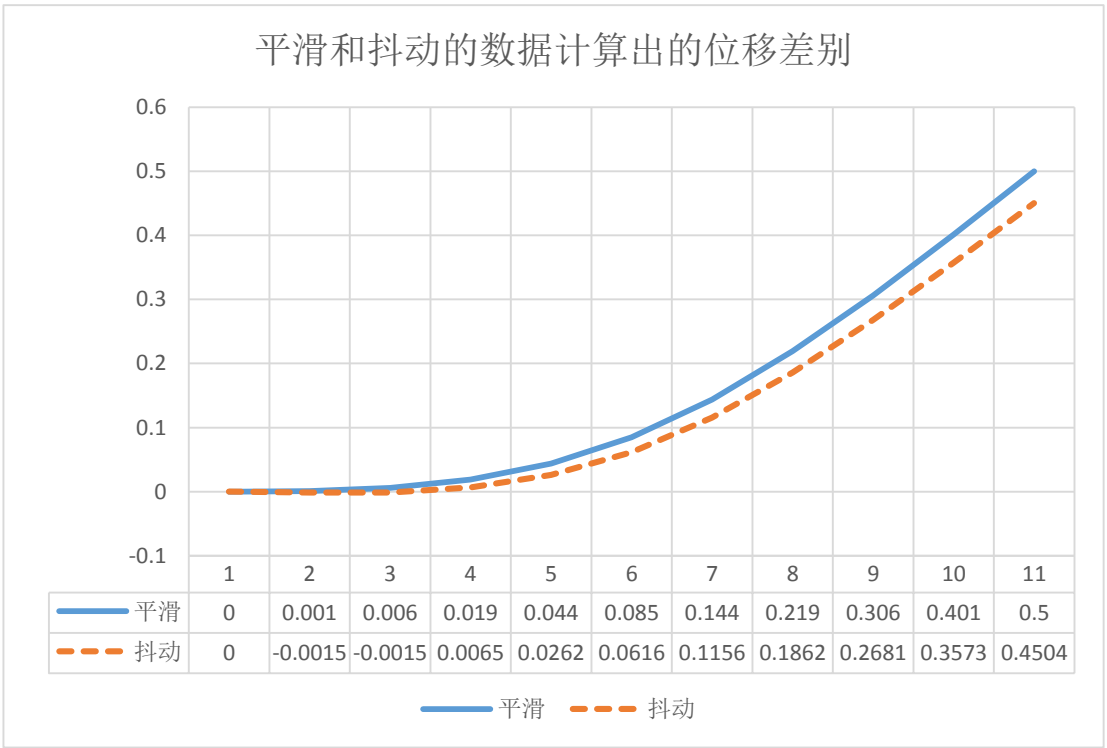

图 3.7 平滑和抖动的数据计算出的位移差别

其次，数据采集频率不稳定也会影响结果的准确性，如果一组数据采集时间间隔比其他的都要长一些或者更短一些，但是在计算中还是被认为是正常的间隔，于是计算的结果就会有偏差，如果采集的时候有遗漏的话亦是如此。因为在计算中用

到了二次积分，任何小的误差都会对速度和位移造成很大的影响，所以对加速度数据的要求还是很苛刻的。只有有了精确的加速度，才能得到精确地速度和位移。所以，在手指运动参数检测系统中对加速度的处理进行了多重校正机制以保证加速度的准确性。

3.2.4 单片机中常用积分算法及实现方法探讨

日常应用中我们常用的离散数据积分算法主要有矩形近似解法、切线近似法、梯形法，三种方法各自的实现方法具体如下：

(1) 矩形近似法计算积分

矩形近似法是求离散数据积分最简单的一种方法。如下面图 3.8 所示，即利用前一项的大小来计算一小段时间内的积分，只要两个离散数据的时间间隔即$t_{n+1}-t_n$的值足够小，该方法就可以得到相对较高的精确度。

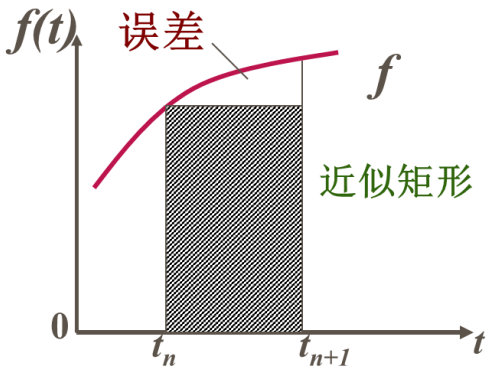

图 3.8 矩形近似法计算积分

对于计算所给出的方程：

$$\begin{cases} y' = f(t,y) \\ y(t_0) = y_0 \end{cases} \quad (3\text{-}4)$$

在区间[tn,tn+1]上积分，得到公式 3-5：

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t,y)dt \quad (3\text{-}5)$$

于是就可以近似的得到公式 3-6:

$$y(t_{n+1}) \approx y(t_n) + hf(t_n,y_n) = y_{n+1} \quad (3\text{-}6)$$

这种方法计算量较少，在单片机运算中效率高，但是从图中可以看出，当采样率并不是很高的时候，计算的误差还是很大的。所以在实际应用中除了对精确度要

求不高的计算，一般都不会采用这种方法。

(2) 切线近似法计算积分

切线近似法的计算步骤是首先求得曲线在该点的切线，通过该点的切线来计算积分。如下面图 3.9 可以看到，只要所求曲线的弯曲程度不大，该方法也可以获得比较好的结果。

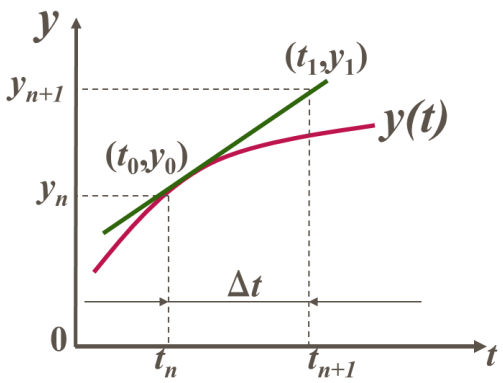

图 3.9 切线近似法计算积分

对于在所示图中计算$y(t_{n+1})$我们可以得到 y(t)在t_n处得切线方程 3-7：

$$y = y_n + f(t_n, y_n)(t - t_n) \quad (3\text{-}7)$$

则得到，

$$y(t_{n+1}) \approx y(t_n) + \Delta t \cdot f(t_n, y_n) = y_{n+1} \quad (3\text{-}8)$$

所以得到的面积即积分公式 3-9：

$$S_0 = \frac{y(t_{n+1}) + y(t_n)}{2} \cdot \Delta t \quad (3\text{-}9)$$

这种方法相对于矩形近似法而言，计算量会增大几倍，但是计算的精确度会高一些。不过对于离散的数据，我们很难准确地获得在一个点对时间的偏导数，因而在实际应用中，得到相对精确地值还是比较困难的。

(3) 梯形法计算积分

梯形积分算法是通过一个时间段前后两点数据来计算积分。如下面图 3.10 所示，当计算的步长很小时，梯形算法也可以获得比较精确地值。

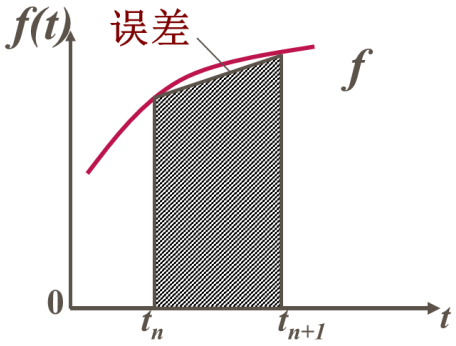

图 3.10 梯形法计算积分

对于上面图 3-9，我们可以知道它曲边梯形的面积计算公式 3-10：

$$S_1 = \int_{t_n}^{t_{n+1}} f(t,y)dt = y(t_{n+1}) - y(t_n) \quad (3\text{-}10)$$

同时可以得到下面的直边梯形的计算公式 3-11：

$$S_2 = \frac{1}{2}h[f(t_n,y_n) + f(t_{n+1},y_{n+1})] \quad (3\text{-}11)$$

当 h 的值比较小时，直边梯形的面积就可代替曲边梯形的面积。相对于上面两种方法，梯形积分法误差是最小的，计算量也只是矩形法的 2 倍左右，完全可以在单片机上很好的运行。因此，在单片机上进行对于简单的离散积分运算，梯形法是一种比较可行的方法。

第四章　基于单片机的手指运动加速度数据处理方法

4.1　加速度传感器位移数据处理的方法介绍

手指运动参数检测系统中提出了一种创新的方法，使得在单片机上利用三轴加速度传感器得到相对精确地位移轨迹成为可能。该方法基于目前广泛应用的利用加速度传感器数据进行频域积分计算位移的方法改进而来，化繁为简并且利用一些列的措施校正数据，从而得到相对精确的位移值。

另外，由于人的手指在做简单运动和连续运动的情况是有很大区别的，在作不连续简单运动时，人可以很好地控制传感器的倾角，从而可以有较小的误差，但是，当人的手指进行连续运动时，倾角的变化就会加剧，于是就会产生较大的偏移量。所以在实际应用中，针对两种不同的情况，需要采用两种不同的模式来计算。同时，对于不连续的运动，由于动作比较简单，因此可以通过一些方法得到比较准确的测量值，但是对于连续运动而言，由于倾角的不稳定性得到相对准确的位移是一件十分困难的事。因此，对于简单不连续运动，可检测的很精确，对于连续运动，则需要进一步去除趋势项，因此只能做到显示出圆周运动和直线运动大致轨迹。

在手指运动参数检测系统中采用了两种方法来分别检测简单运动和连续运动的轨迹，并通过特殊的方法实现两种计算方式的转换以达到良好的处理性能。下面将会详细讲述手指运动参数检测系统中两种数据处理方法和两者之间的转换转换方式。

4.2　基于传感器不连续运动情况下的位移数据处理

4.2.1　基于传感器不连续运动情况下的位移数据处理流程

在手指运动参数检测系统基于传感器不连续运动时所采用的具体的数据处理流程如图 4.1 所示。

首先，是对于原始加速度的处理。原始加速度数据 A0 首先经过两次低通滤波，即取 15 个点和 100 个点加速度平均，分别去除高频分量（波形抖动）。之后将两组

数据相减得到初步处理后的加速度值。由于三轴加速度传感器产生的数据有一定的波动，因此初步输出的结果会和阈值比较后再输出，以消除抖动，增加系统稳定性。最后，输出的加速度 A 还会通过原始加速度值的变化状况来校正，当传感器静止时，原始加速度值并没有很大变化，此时加速度的输出值被校正为 0。

其次，是对处理后的加速度的处理。一方面，利用处理后的加速度数据进行积分计算得到速度和位移。另一方面，为了数据的准确性，继续利用处理后的加速度数据的特性，即当传感器静止时所输出的数据为 0，来校正当前速度的值。

然后，是对于位移的处理，由于三轴加速度传感器输出的数据的偏移量会很大，有时候难免会遇到位移不断向一个方向偏移的情况，因此，利用边界处理，将位移限制在一定的范围内，从而避免了偏移量过大的情况。

最后，对于连续运动的处理，由于具有很大的偏移量和偏移变化，因此，我们还需要进一步去除偏移量才能获得较好的运动轨迹。

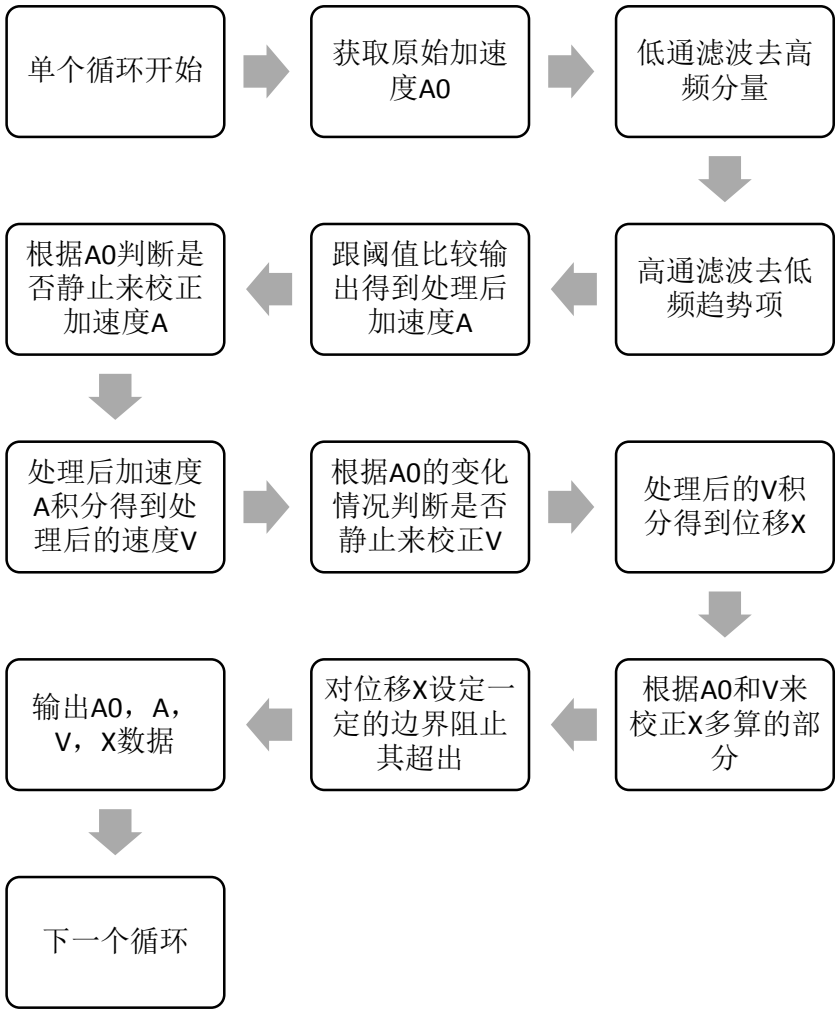

图 4.1 手指运动参数检测系统具体数据处理流程

4.2.2　加速度平滑处理

对于原始加速度数据，我们前面已经知道具有很大的抖动，说明高频分量很大，因此第一步我们就需要平滑处理。常用的平滑算法低通滤波法、有 5 点 3 次平滑法等。前面已经讲到，使用低通滤波法时会产生很大的数据计算量，单片机并不能满足计算需求。因此我们只能用多点平滑的方法来尝试。

以 5 点 3 次平滑算法为例子，取相邻的 5 个数据点，可以拟合出一条 3 次曲线来，然后用 3 次曲线上相应的位置的数据值作为滤波后结果。简单的说就是 Savitzky-Golay 滤波器 。只不过 Savitzky-Golay 滤波器并不特殊考虑边界的几个数据点，而这个算法还特意把边上的几个点的数据拟合结果给推导了出来。其实平滑算法并不止是 5 点 3 次，还有 5 点 2 次，7 点 2 次，7 点 1 次平滑算法等。取点越多，次数越少，曲线自然就越平滑。对于三轴加速度传感器的加速度数据，由于抖动较大，采用多点一次的平滑算法是比较合适的。多点一次平滑算法的实现并不困难，列如三点一次的算法如下：

```
out[0] = ( 5.0 * in[0] + 2.0 * in[1] - in[2] ) / 6.0;
for ( i = 1; i <= N - 2; i++ )
{
    out[i] = ( in[i - 1] + in[i] + in[i + 1] ) / 3.0;
}
out[N - 1] = ( 5.0 * in[N - 1] + 2.0 * in[N - 2] - in[N - 3] ) / 6.0;
```

由于加速度传感器所输出的数据是连续的因此我们只要计算中间的数据对其进行平滑计算即可，而不需要考虑边界的数据。即以上面 3 点 1 次平滑算法为例，只取中间 out[i] = (in[i - 1] + in[i] + in[i + 1]) / 3.0 的公式来进行曲线平滑即可。这样处理第一可以加快运算速度，第二可以使结果更加精确。为了寻找最合适的多点一次算法，我们对原始加速度数据进行了 5 点、10 点、15 点 1 次平滑计算，得到结果如下面图 4.2 所示。

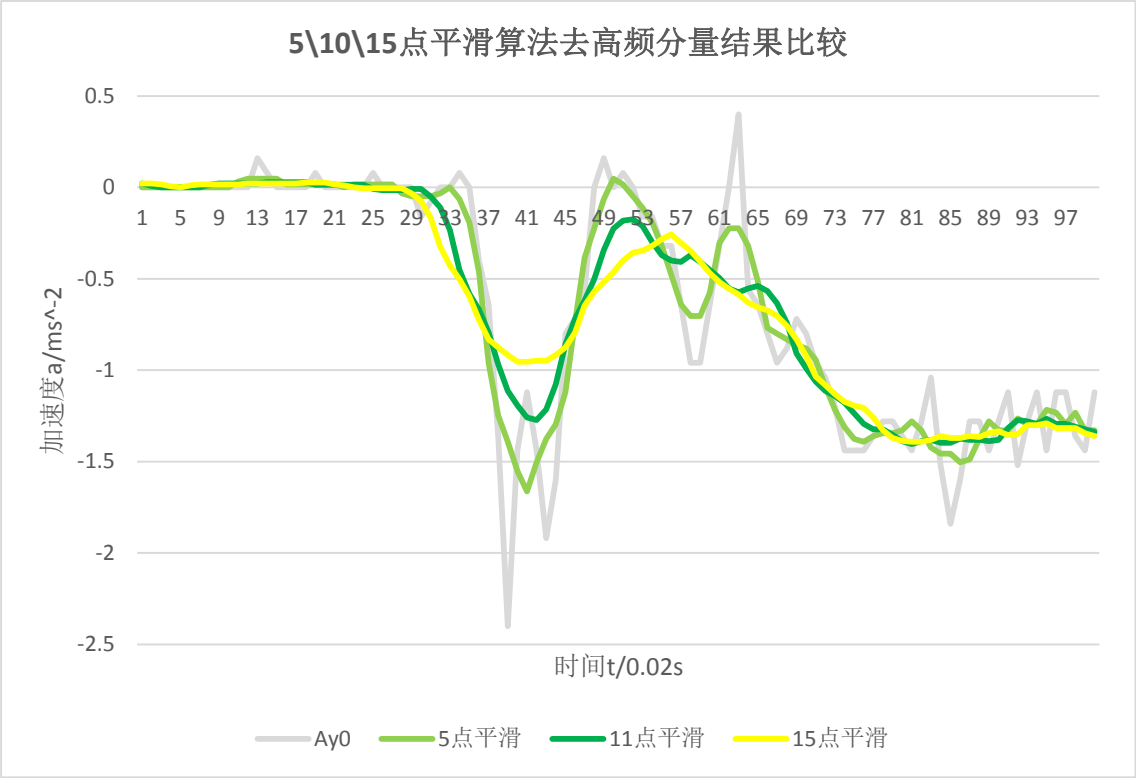

图 4.2 多种多点一次平滑算法比较

由上面图像可以看出，5 点平滑算法和 11 点平滑算法都是相对较好的一种算法，但是在实际应用中，发现使用 11 点平滑算法时，手指运动参数检测系统会显得过于灵敏，反而不好控制。然而 15 点平滑法虽然不是特别灵敏同时稳定性会提高很多，因此在一般对精度要求不高，只是检测手势运动的情况下，应该采用取的点数较多的平滑算法以保证系统的性能。在手指运动参数检测系统中，为了增加稳定性，还是采用了 15 点 1 次平滑算法来处理，具体算法如下：

```
int   F15(int in[])   {
   int averange = 0;
   for (int i = 43; i < 58; i++) averange += in[i];
   return averange /= 15.0;
}
```

4.2.3 消除加速度趋势项

由上面的说明可以知道加速度传感器由于本身的特性，输出的加速度结果具有一定的偏移量。关于偏移量的程度，可以通过图 4.3 直观的看到，图 4.3 是将三轴加速度传感器沿 Y 轴作往返运动所生成的图像，其中绿色的直线反映了它的

趋势项变化。

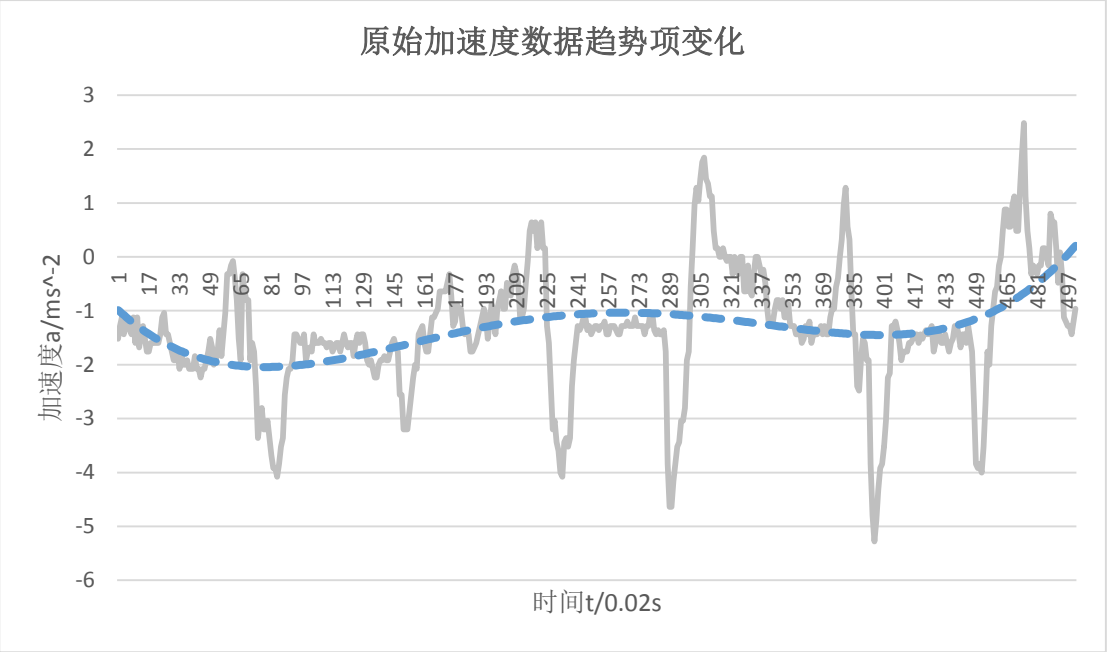

图 4.3 加速度原始数据趋势项变化

从图中可以明显的看到，加速度原始数据具有很大的偏移并且在不停的改变着，如果不经处理，偏移量就会被成百上千倍的放大从而导致巨大的误差。对于去除趋势项，手指运动参数检测系统中采用了多点一次平滑的方法。具体实现方法是取很多个点进行平滑计算，得出的值即为加速度原始数据的偏移量。再将原始数据与偏移量相减，就得到去除偏移量的加速度数据。同时，为了使手指运动参数检测系统具有良好的实时性，取的点的个数不能过多，不然采集数据就会耗费很长的时间，从而会影响系统的实时性。因此，为了寻找最优的多点一次平滑算法，实验中试验了 50 点、100 点、150 点的平滑算法来去除趋势项，即将原来加速度的值减去用 50 点、100 点、150 点的平滑算法算出的值得到除去趋势项后的曲线，最终得到的结果如下面图 4.4 所示。

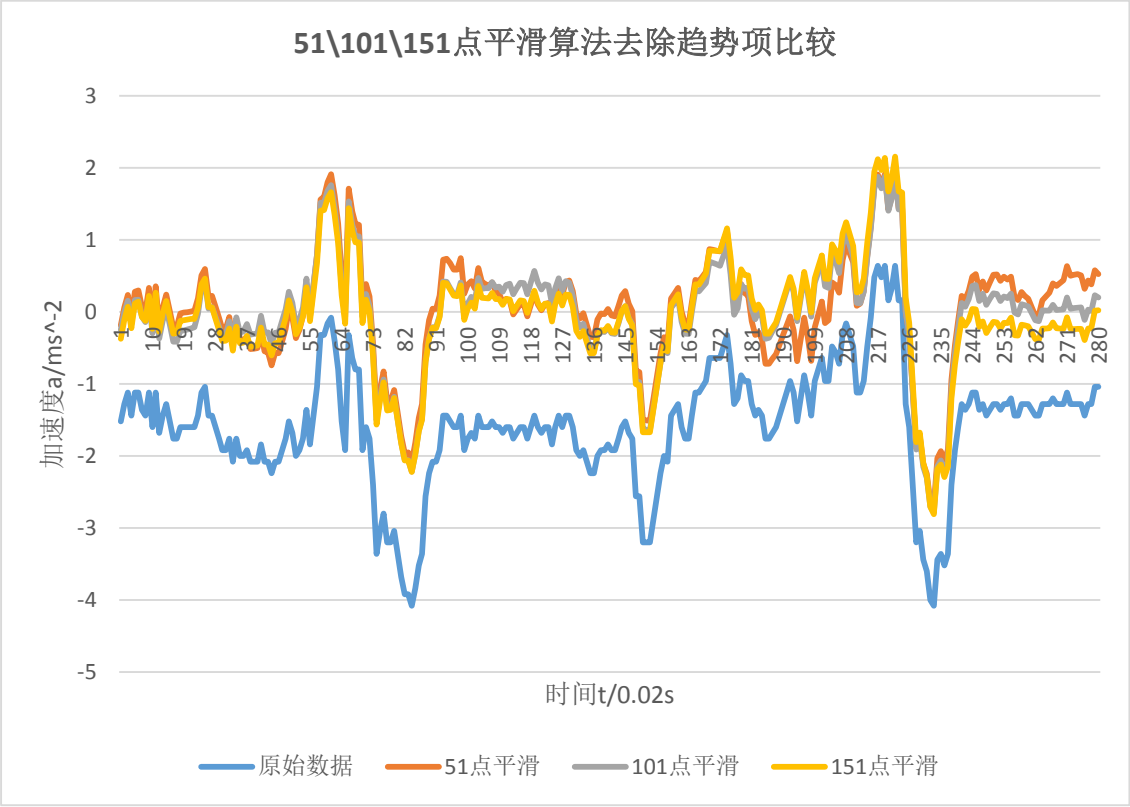

图 4.4 51 点\101 点\151 点平滑算法去除趋势项比较

由上图可以看到当采用 101 点 1 次平滑算法的时候，加速度数据看起来是最好的。因此，在手指运动参数检测系统中也采用了 101 点 1 次平滑的算法来除去趋势项。但是在实际应用中发现，采用此种去除趋势项的方法还会产生另一个现象，就是用多点平滑算法算出的趋势项当加速度突然变化时，趋势项也会产生比较明显的影响。即当加速度突然变大时，算出来的趋势项也会跟着变大，反之当加速度快速减小时趋势项也会随之减小，但是实际的趋势项变化并没有那么大。因此，加速度测量就会产生一些误差，特别是在波形方面。为了更好的说明这个现象，从试验中特地的找了一个常见的波形，与其进行 101 点平滑算法去处去实项后的图形进行比较，结果如图 4.5 所示。

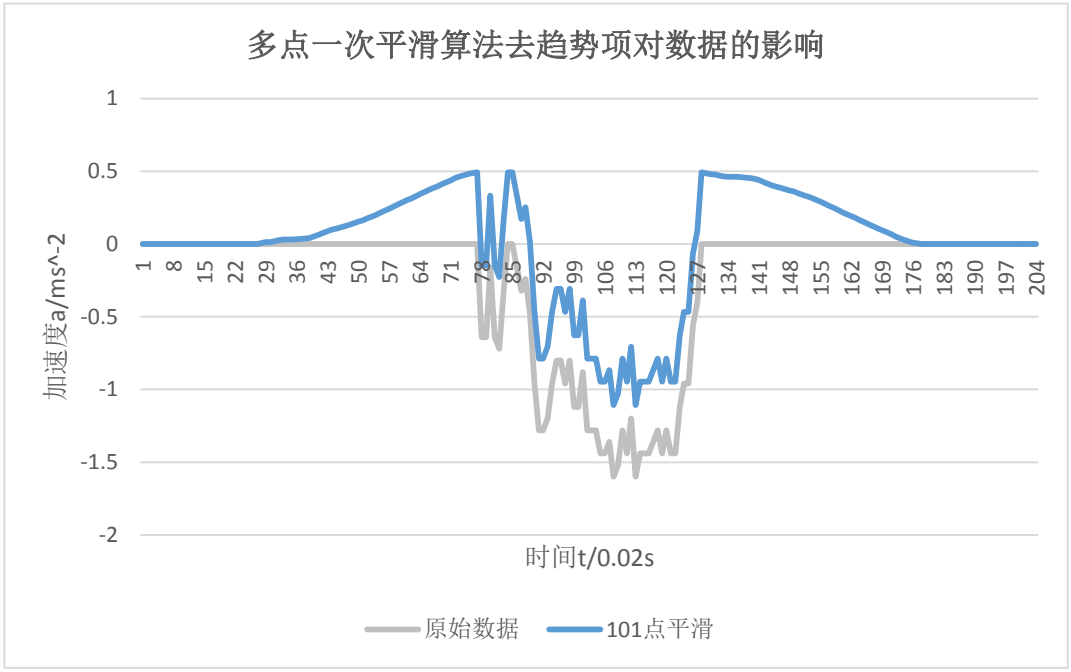

图 4.5 多点一次平滑算法去趋势项对数据的影响

由上图可以看到，对于没有偏移的加速度数据，采用去趋势项算法后，在一个波形的两边出现了反方向的正的加速度数据，这与原始数据并不相符，为了进一步减少这种误差，在手指运动参数检测系统中采用了两种方法。

第一种方法是通过对 101 点 1 次平滑算法进行一阶滤波，来降低 101 点 1 次平滑算法所算出的趋势项的变化程度。具体的代码如下：

```
int F101A(int in[], int *Tin)    {
    int averange = 0;
    for (int i = 0; i < 101; i++) averange += in[i];
    averange /= 101.0;
    *Tin = 0.5 * averange + 0.5 * (*Tin);    //加速度一阶滤波
    return *Tin;
}
```

即最终算出的加速度数据趋势项的值是前一次算出的趋势项的值和这一次采用 101 点 1 次平滑算法所算出的趋势项的值的平均，这样就可以很大的减少误差。

第二种方法是通过原始加速度数据的特点来校正处理后的加速度数据，当原始加速度数据不变时，三轴加速度传感器一般处没有加速度的状态，此时处理后的加速度也会置 0。这样得以保证数据的准确性。具体实现方法如图 4.6 所示。

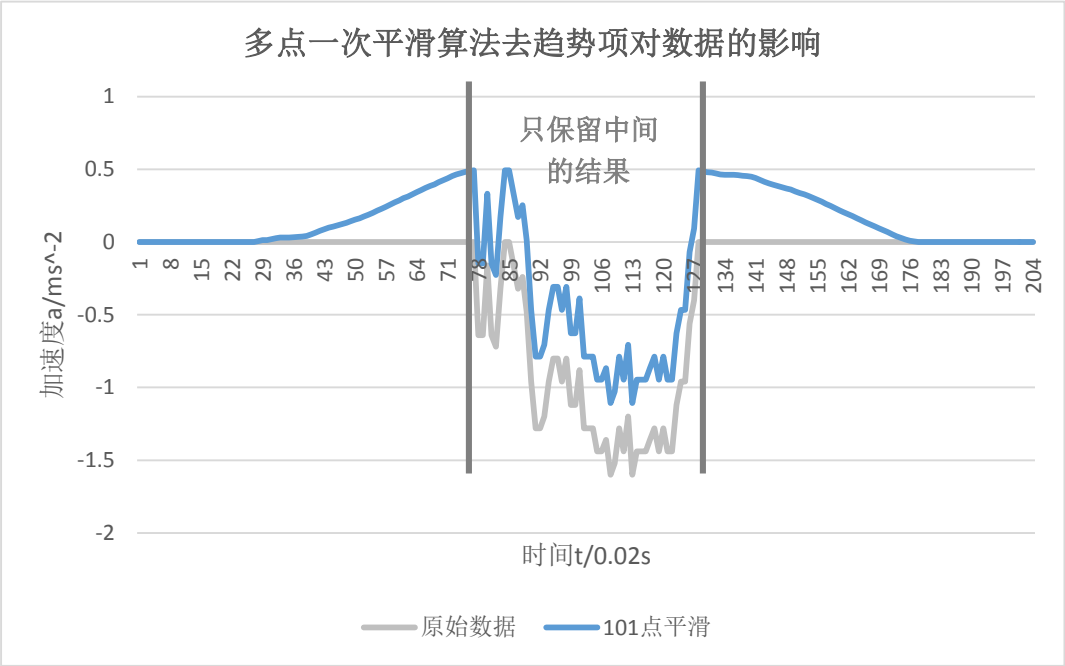

图 4.6 利用原始加速度数据校正处理后的加速度数据

4.2.4 速度和位移的梯形积分运算

由上面可以知道，三种单片机上常用的离散积分方法中，梯形积分是一种最好的方法，因此在手指运动参数检测系统中采用了梯形积分的方法来计算速度和位移。

首先取前一个时间点的加速度值A_n，到了第二个时间点测量加速度A_{n+1}，此时，计算两个时间点之间的面积即可算出速度。计算公式 4-1 如下：

$$V_{n+1} = V_n + \frac{1}{2}(A_n + A_{n+1}) \cdot \Delta t \qquad (4\text{-}1)$$

对于位移的计算，也是如此，计算公式 4-2,如下：

$$X_{n+1} = X_n + \frac{1}{2}(V_n + V_{n+1}) \cdot \Delta t \qquad (4\text{-}2)$$

梯形算法相对来说要精确一些，相对于其他方法其可以在一定程度上降低单片机的运算负担。对于短时间内位移的计算再加上实时校正，这种计算方法已经足够达到手指运动参数检测系统进行手势检测所需要的精确度。

4.2.5 传感器静止状态检测和速度位移校正

由于手指运动的特性，即当手指在运动时就有加速度的产生，而且在很短的范围内，人的手指是不可能长时间作匀速运动的，从图 4.3 中的加速度原始数据就可

以看到这种规律。因此，我们可以利用这个特点来判断加速度传感器是否处于静止状态，并同时对速度和位移进行校正，以增加结果的准确性。

静止状态检测的过程是根据处理过后的加速度数据进行判断的，如图 4.7 所示，可以从图中看出，当三个加速度绝对值之和连续多次（在 50Hz 采样频率下，次数可以根据实际情况设为 3-10 次）小于阈值时会自动判定加速度传感器为静止状态。此时输出加速度为 0，并校正速度为零，同时会减去之前多算的位移的量。

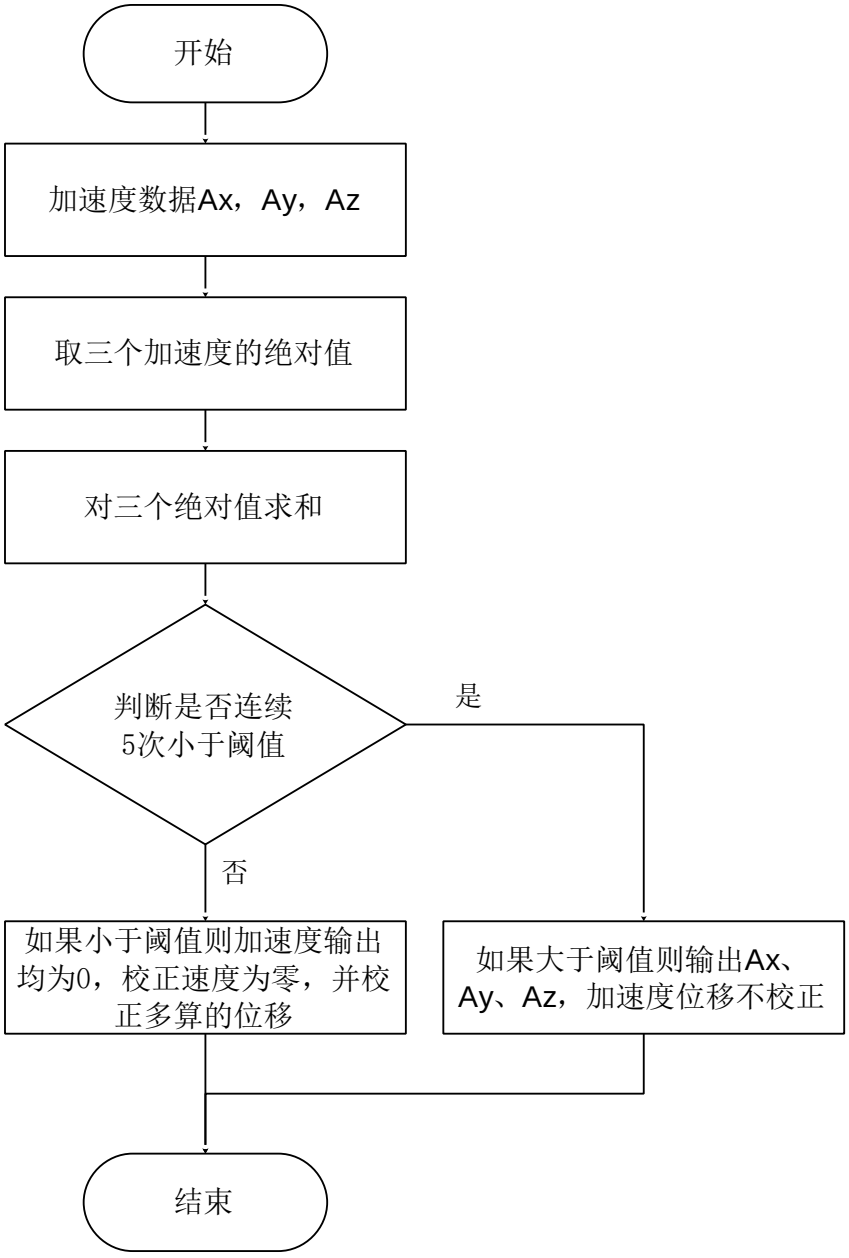

图 4.7 加速度传感器静止状态的检测和速度位移的校正

4.2.6 位移边界的设置

前面说到加速度数据具有偏移量，虽然在前面的加速度数据处理的过程中已经去除了很大一部分的偏移，但是由于位移由加速度数据双重积分的到，如果加速度数据仅仅只有一点的偏移误差，到了位移之后，时间一长就会变得很大。因此，对位移设定一个边界是一件十分有必要的事。

手指运动参数检测系统中对位移设定边界的方法是，在由速度计算加速度的时候，比较计算的结果和边界的大小，如果超出边界范围，则输出边界的值，这样，就可以将位移数据保持在一定的范围内，图 4.8 显示出加边界条件和不加边界条件处理结果差异还是很明显的。加边界条件后结果明显和实际更贴近了许多。

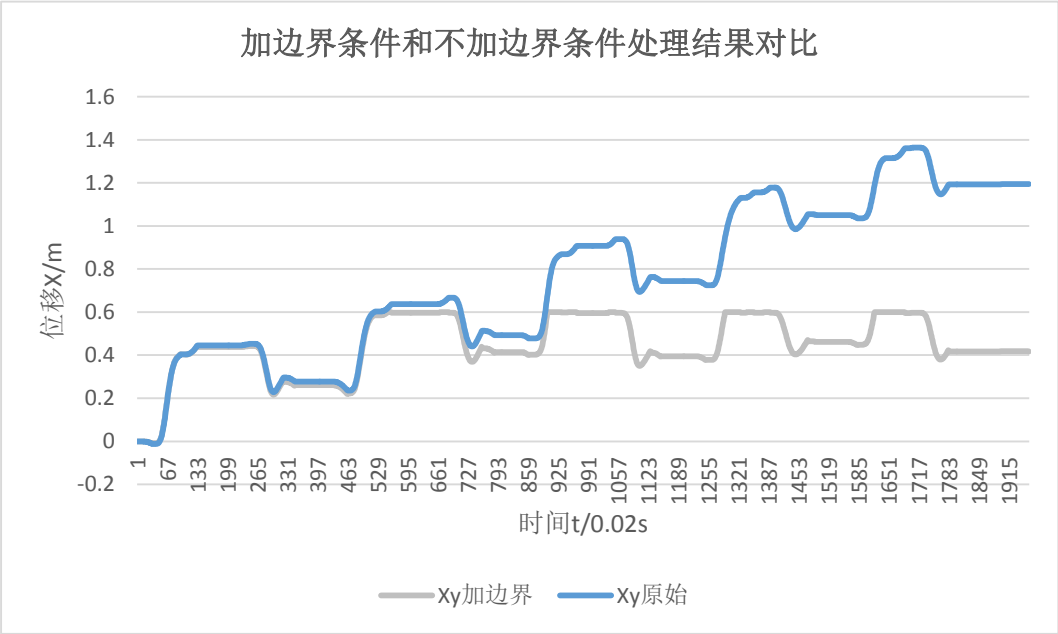

图 4.8 加边界条件和不加边界条件处理结果对比

4.3 基于传感器连续运动情况下的数据处理

4.3.1 基于传感器连续运动情况下的位移数据处理流程

前面已经讲到在加速度传感器连续运动下，加速度传感器的偏移量会进一步加大，因此，之前对于不连续简单运动的算法并不适用于连续重复运动的情况。另外，在前面一种方法中采用了太多特殊处理方法。但是，上述的特殊处理方法例如通过原始加速度校正加速度数据，通过加速度数据校正速度位移数据等都具有很大的

局限性，并不适用于连续运动情况下的数据处理。因此，对于连续运动的情况下，必须采用另外一套数据处理方法来保证运动轨迹的准确性。

对于传感器连续运动的情况，手指运动参数检测系统所采用的数据处理方法如下面图 4.9 所示。从下面图中可以看到，原始加速度首先经过两次低通滤波相减即简单处理抖动和偏移量后就直接得到处理后的加速度数据；之后，通过处理后的加速数据进行积分和一阶滤波就得到了速度；最后，通过处理过的速度也经过一样的积分和滤波得到了位移。一阶滤波的方法会在下面详细提到。这样处理的好处是对各种动作的适应性更强，更适用于连续重复的运动的检测。但是也存在很大的缺点，就是对距离测量的精确度大不如第一种方法。

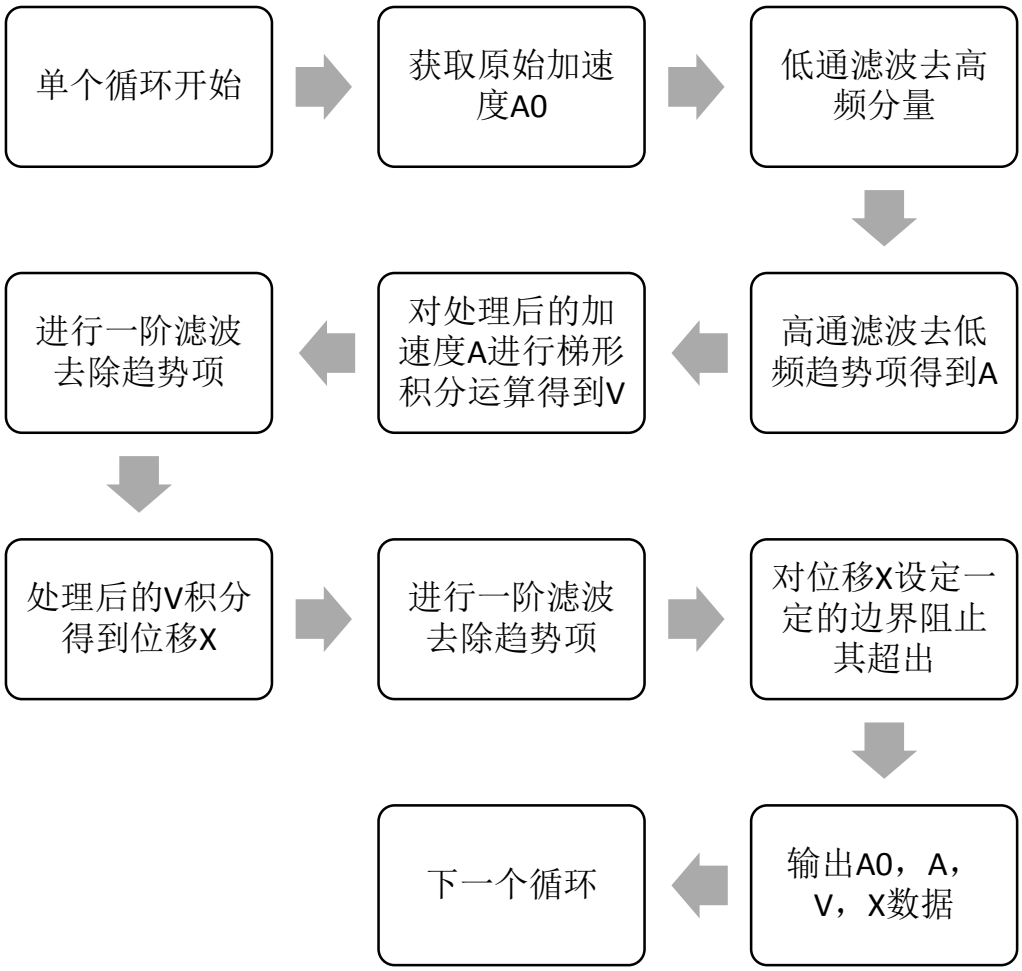

图 4.9 传感器连续运动情况下的位移数据处理流程

4.3.2 基于传感器连续运动下速度位移趋势项的去除

前面已经说到，在三轴加速度传感器连续运动的情况下，测得的数据具有很大的偏移量。对此做了实验来验证。试验中，将加速度传感器沿着 Y 轴作连续的往返运动，记录加速度数据并对数据进行积分得到速度的变化情况，之后得到的结果如下面图 4.10 和图 4.11 所示。对于做直线往返运动，理论上来说加速度和速度都是在坐标轴上下徘徊的，但是从图中可以明显看到，速度已经有了很明显的偏移。

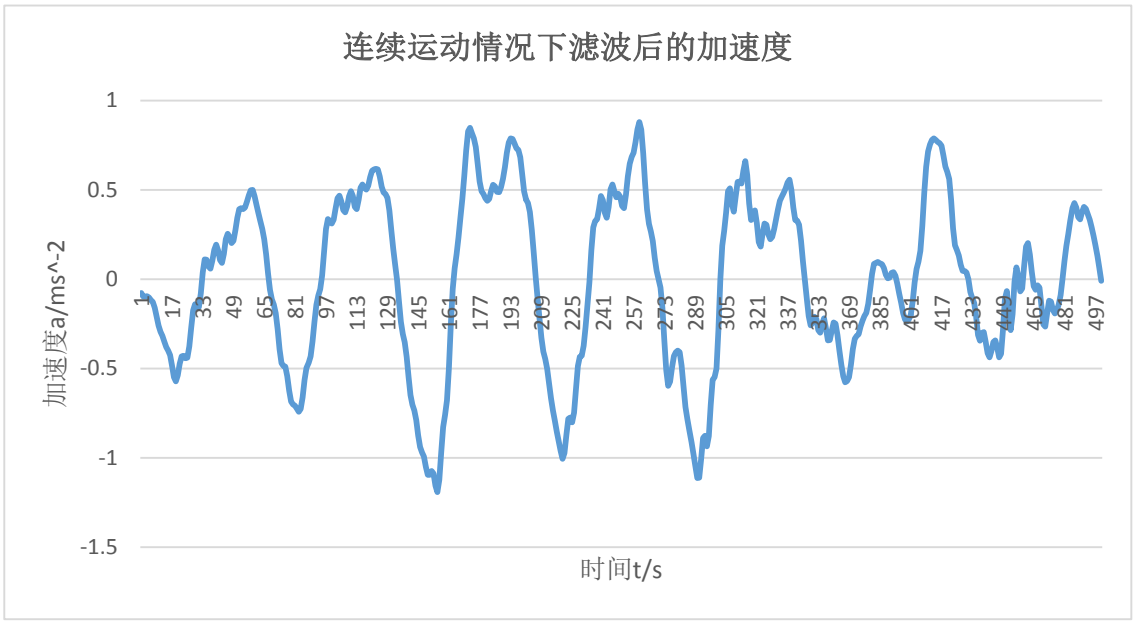

图 4.10 连续运动情况下滤波后的加速度

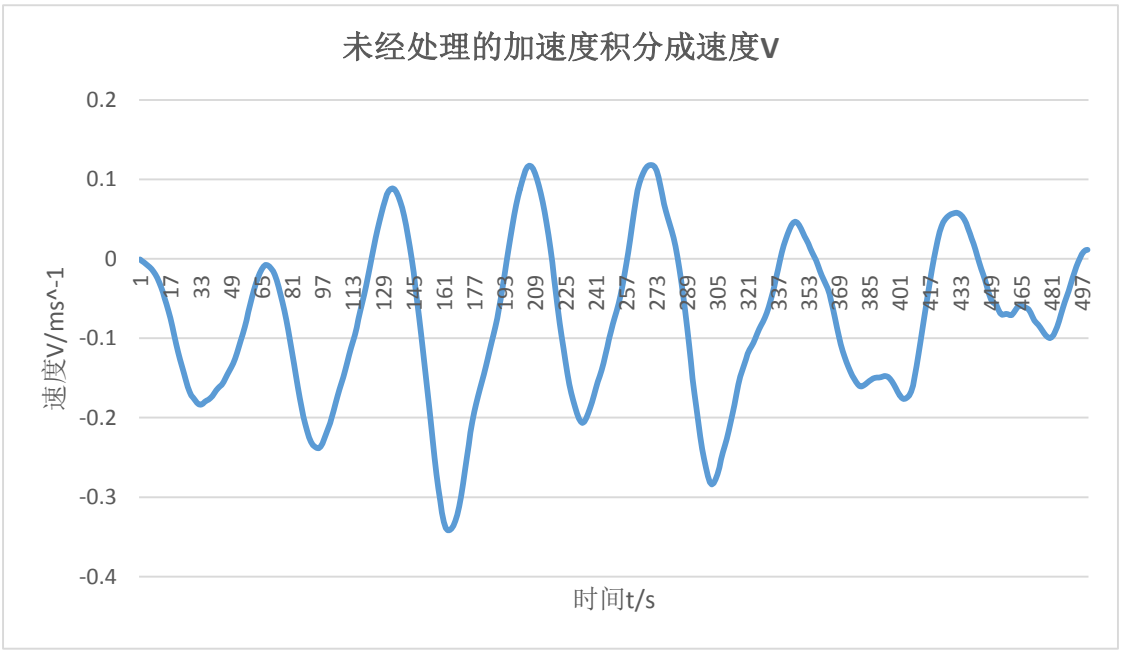

图 4.11 未经处理的速度 V 图像（滤波后加速度 A 原始积分）

对于上面速度产生的偏移，可以采用特殊的一阶滤波的方法来消除，具体的实现公式如下面所示：

$$V_{n+1} = \alpha \cdot V_n + \frac{1}{2}(A_n + A_{n+1}) \cdot \Delta t \qquad (4\text{-}3)$$

即在原来梯形积分的基础上，对上一项加速度乘以一个小于 1 的系数 α，这样偏移量就可以大大减少。为了进一步讨论 α 的取值，α 从 1 开始往下取并生成图像对比，结果如图 4.12 所示。从图中可以看到，当 α=0.9 的时候就可以达到比较好的结果，如果取值偏大可能会导致在后来的积分中再次产生较大的偏移，如果太小，就会使误差进一步加大。

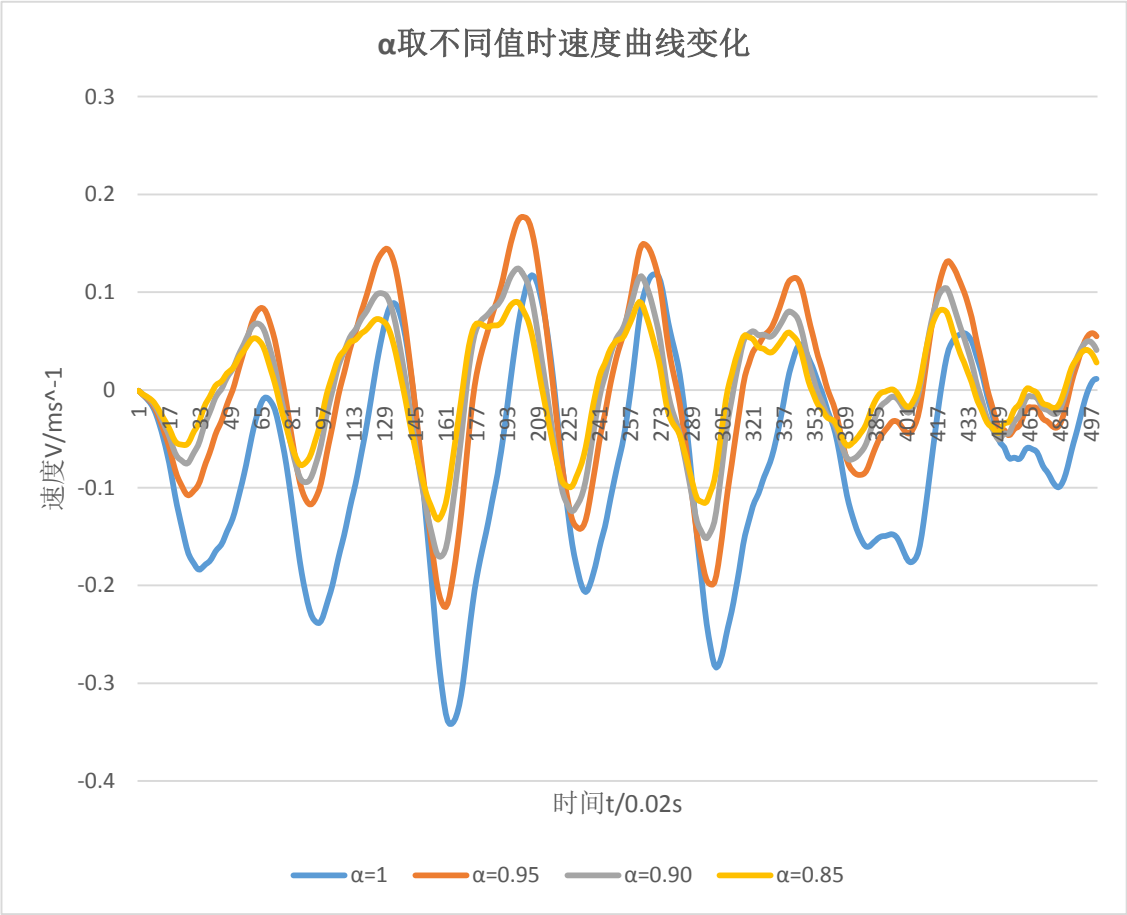

图 4.12 α 取不同值时速度曲线变化

同时，位移也必然会有偏移，对于位移的偏移。采用了同样的方法来去除，具体的实现公式如下面所示：

$$X_{n+1} = \alpha \cdot X_n + \frac{1}{2}(V_n + V_{n+1}) \cdot \Delta t \qquad (4\text{-}3)$$

和上面速度的算法一样，取上面 α=0.9 时所计算出的速度结果，再利用位移求解公式来算出位移，α 分别取 1、0.9、0.85，从图像中可以得到，当 α=0.9 时的曲线是

比较理想的。

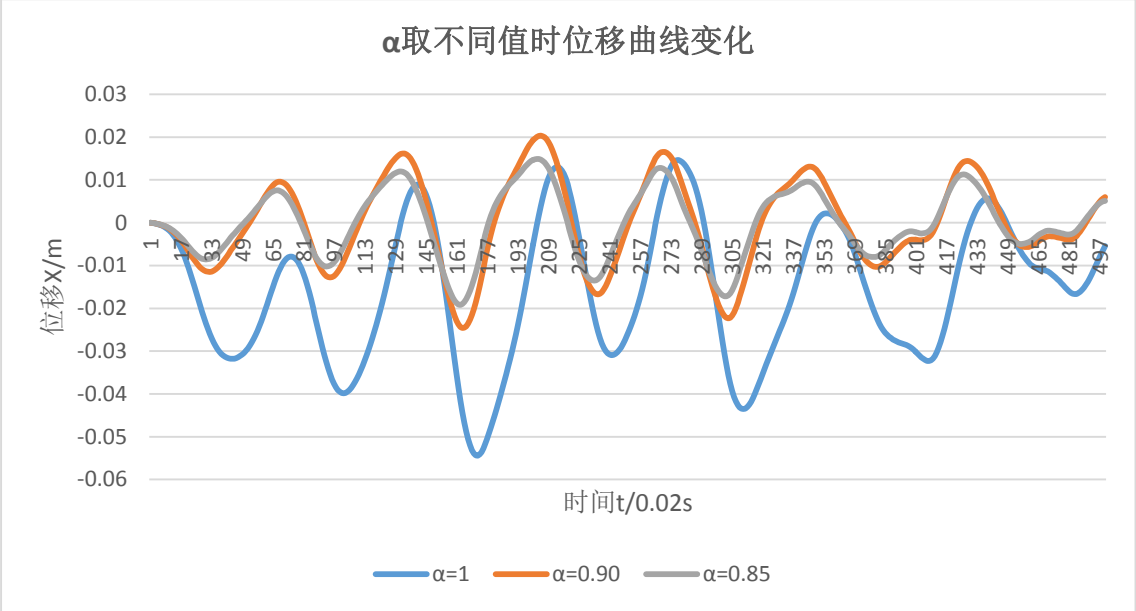

图 4.13 α 取不同值时位移曲线变化

从上面可以得到，通过取适当的 α 的值可以大大地减少偏移量。但是这也并不一定是最终的取值，在实际应用中可以根据运动的情况来改变两个 α 的取值来得到最终的取值。再后续的实践中，发现之前的取法偏移量还是比较大，最终将求位移的 α 改为了 0.8。

4.4 基于两种位移计算模式转换方式的实现

4.4.1 简单运动和连续运动区别

要让程序自动识别三轴加速度传感器做的是简单运动还是连续运动，首先需要了解简单运动和连续运动的区别所在才能让程序在两种算法之间转换。

为了寻找简单不连续运动和连续运动的区别，我们取了简单运动和连续运动时 Y 轴处理过后的加速数据并绘制了处理后的加速度变化图像。通过对数据的观察，发现简单不连续运动和连续运动的本质别在于运动的时间占总时间的比重是不同的。

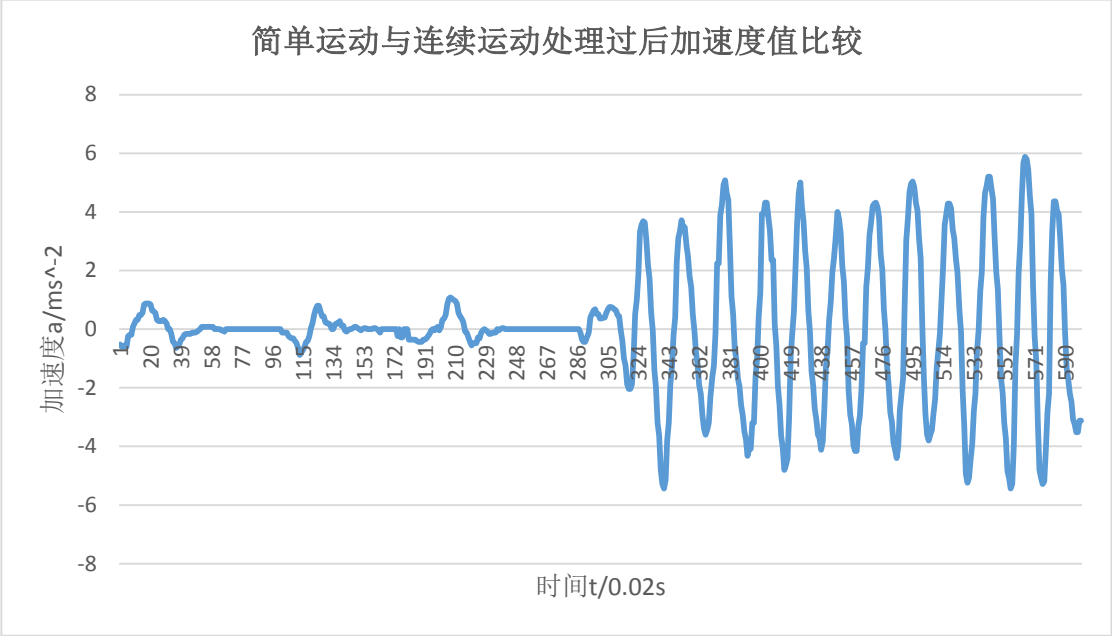

图 4.14 简单运动与连续运动处理过后加速度值比较

接着对上面图 4.14 的数据进行处理，当数据绝对值大于 2 时一律输出 1，否则输出 0，最后得到图 4.15。从图中可清晰的看出不连续运动和连续运动在绝对值大于 2 的加速度的分布上有着很明显的区别。第一，对于不连续运动的情况下，处理后的加速度每隔一段时间就会有空闲的状态，然而在后面的连续运动中，很少有检测到加速度值不大于 2 的情况。有了这种区别，让程序去自主转换计算方式就不难了。

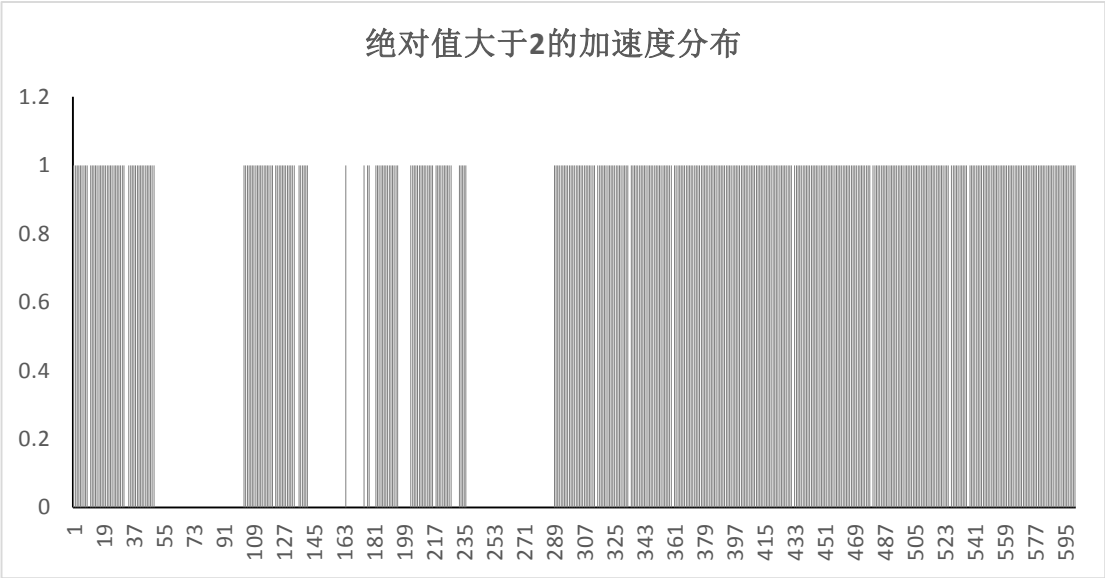

图 4.15 绝对值大于 2 的加速度分布

4.4.2 两种模式转换方式的设计

在手指运动参数检测系统中，采用了检测绝对值大于 2 的加速度的连续性的方法来判断是简单不连续运动还是连续的运动。具体的实现方法如图 4.16 所示。图中，参数 k 为描述运动频率的一个特征值，并且特征值 k 在 0-260 范围内变化，超过边界值时会自动校正到边界值大小。当传感器在运动时，即输出的处理过后的加速度绝对值大于 2 时，特征值 k=k+2。反之，当传感器在“静止”状态时，即输出的处理过后的加速度绝对值小于等于 2 时，特征值 k=k-20。所以，当传感器连续运动了 150 个点以上的时候，特征值 k 就会大于 300，因而程序就会自动识别为连续运动状态，对加速度数据就采用第二种算法。若在中间又连续静止了 3 个以上的点，则 k 的值就会降到 300 以下，此时程序就会自动识别到当前是简单不连续的运动，从而对加速度数据采用第一种算法。

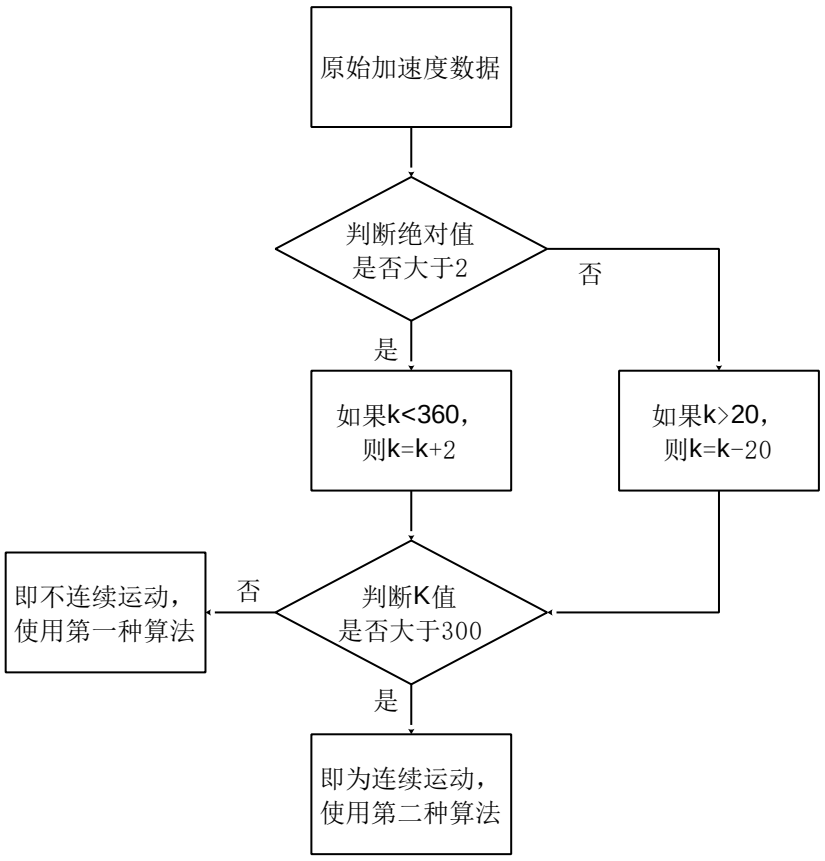

图 4.16 两种计算转换原理

通过这种方法，就可以简单又比较准确的进行对两种运动模式的检测，从而实现了两种算法的自动转换。同时，这种方法也有一定的缺陷，即当从简单不连续运动模式转换到连续运动模式时可能大部分时间至少需要检测 150 点以上，大概需

要 3 秒钟的时间，转换延迟还是比较严重的。因此，在转换的过程中可能会出现不是特别理想的波形，如果需要精确处理，则只要在刚刚检测到连续运动之后，再去修改前面的结果即可。

4.5　基于静止状态加速度传感器倾角测量

在手指运动参数检测系统中倾角的测量是由上位机完成的，这样做的主要原因是为了减少数据的传输量，从而提高软件运行效率。由上面 3.1.1 节的叙述可以得到 X、Y、Z 轴与原先的标准坐标轴的夹角α、β、γ的计算公式为 3-1、3-2、3-3。在实际应用当中，只要测出 X、Y 轴对重力加速度的夹角就可以获得三轴加速度传感器的倾角。　如图 4.17 所示，算出 α 、β 的夹角就可以或者传感器的倾斜状态。

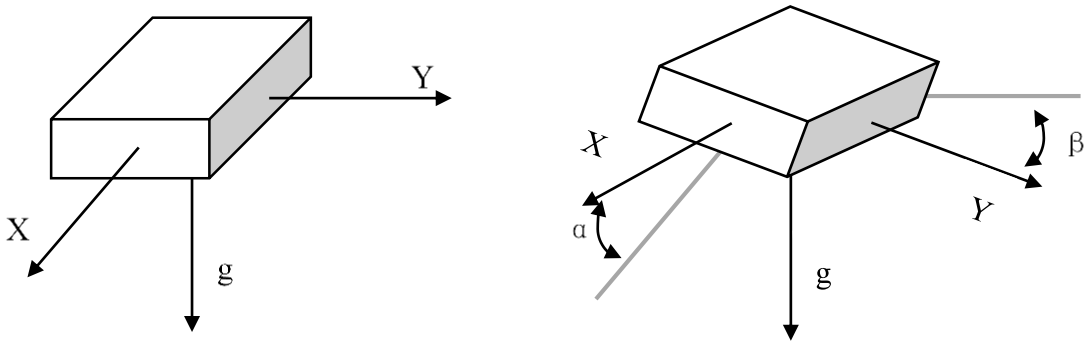

图 4.17 利用 X、Y 轴加速度测量倾角

在手指运动参数检测系统中，计算倾角的 VB.NET 算法如下所示：

```
anglex = Math.Atan(ay / Math.Sqrt(az ^ 2 + ax ^ 2)) * 180 / Math.PI
angley = Math.Atan(ax / Math.Sqrt(az ^ 2 + ay ^ 2)) * 180 / Math.PI
```

其中 ax、ay、az 是三轴加速度传感器三个轴的加速度值，anglex、angley 是加速度传感器 X 轴 Y 轴与重力加速度的倾角。由于 Math.Atan 函数计算出的是弧度值，因此变换成角度还需要乘以 180/PI。

第五章 手指运动参数检测系统检测性能的评估

检测中使用的手指运动参数检测系统的下位机如下图 5.1 所示，主要由单片机、加速度传感器、无线模块、电池组成。验证工作通过采集上位机和下位机的数据，并在 Matlab 和 Excel 中进行画图和计算，最终得出一个客观的对手指运动参数检测系统的评价。

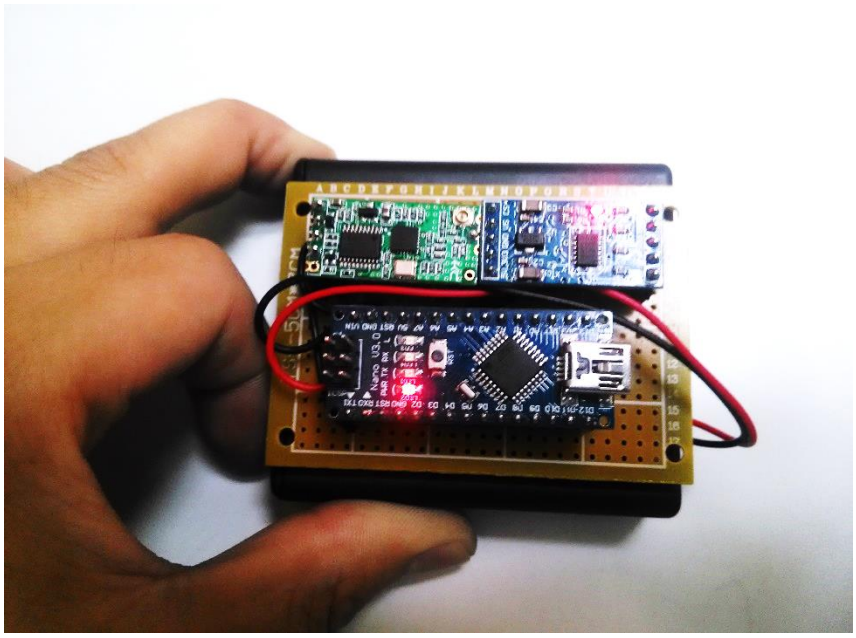

图 5.1 手指运动参数检测系统下位机模块

为了进一步验证手指运动参数检测系统的准确性，分别用了几个常用的手指运动来检测手指运动参数检测系统的精确性。对于简单运动的检测，主要检测简单的直线往返运动；对于连续运动的检测，主要检测沿直线往返、沿正圆、沿椭圆的连续重复运动。检测的结果主要从检测测量的平面轨迹、空间轨迹、测量误差几个方面来定性的评价这种新的手指运动参数算法。总体来说，对于手指运动趋势的检测相对来说比较精确，但是对于距离的检测还是有一定的误差，特别是对于连续运动的检测，误差还是相当大的。具体在后面会详细讲到。

5.1 对于手指做不连续直线往反式运动参数检测

对于手指做不连续直线往返式运动的检测过程如图 5.2 所示，模拟手指分别沿 20cm 和 30cm 长度有间歇的来回运动，并从串口接收数据进行分析和处理。由于，前面的章节中说到了加速度的阈值，在一定程度上可能会影响测量结果的稳定性，为了探究阈值对检测结果的影响，测试中取了阈值为 4 和 10 并对手指运动参数检测系统进行实验。

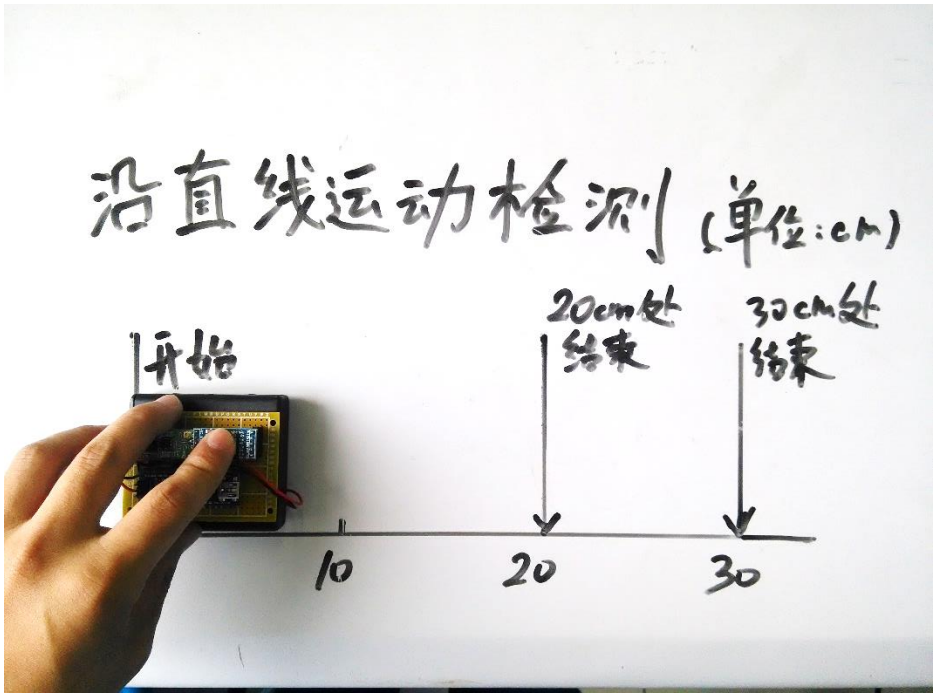

图 5.2 不连续直线往返式运动检测

经过若干次来回运动后，测得阈值分别为 10 和 4 时，在比较快速移动的情况下和在 20cm、30cm 长度下来回运动的多组数据。并且进行比较和分析。首先，取 30cm 长度下阈值为 10 时，沿 X 轴方向快速来回运动所测得的数据，得到三维轨迹图如图 5.3 所示。

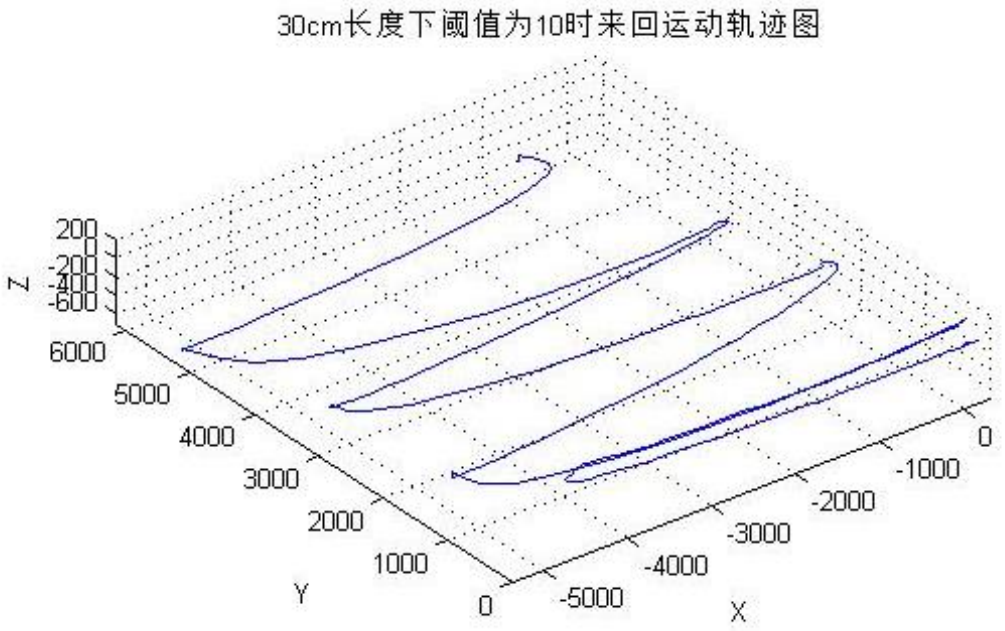

图 5.3 30cm 长度下阈值为 10 时来回运动的三维轨迹图

从图中可以看出，沿 X 轴方向运动的轨迹还是比较明显的，但同时由于手指运动并不稳定，在 Y 轴方向还是会有一定的偏移。为了探究在 X 轴方向测量的精确度，从图中取 X 轴的参数变化，获得在 X 轴方向的位移轨迹，得到图 5.4。

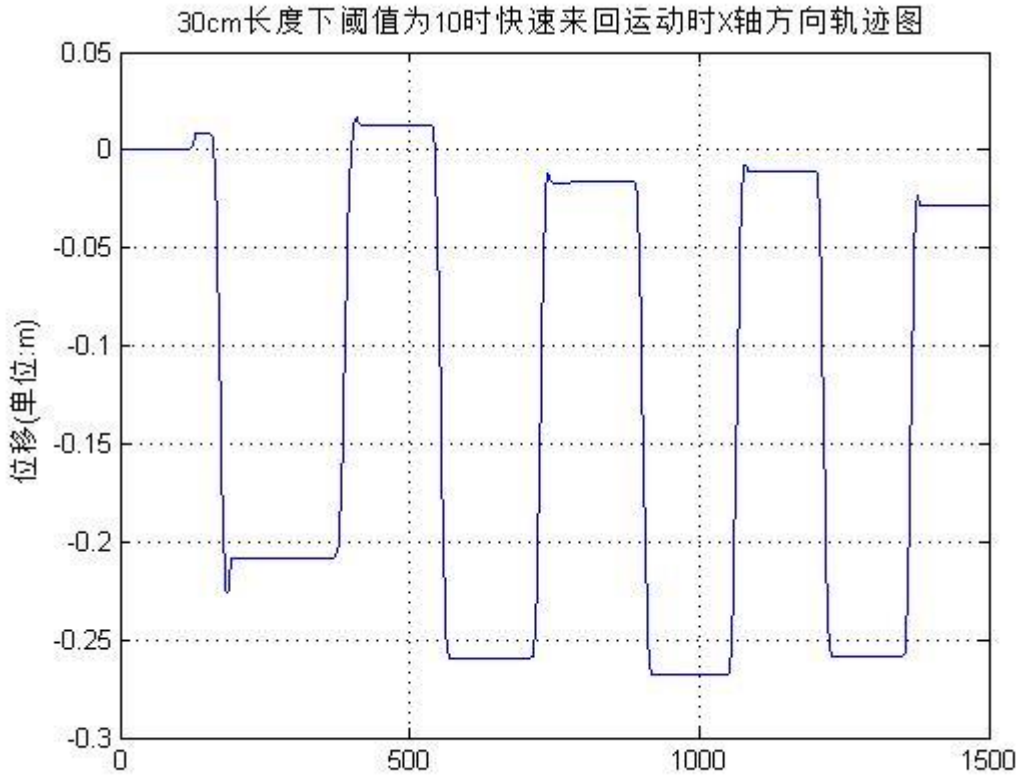

图 5.4 30cm 长度下阈值为 10 时来回运动的 X 轴方向轨迹图

从图中得到每一次位移停止后的坐标位置和误差如表 5.1 所示。

表 5.1 30cm 长度下阈值为 10 时不连续来回运动测量误差

序号	1	2	3	4	5	6	7	8	平均
位移大小/m	0.2082	0.2211	0.2723	0.2433	0.2518	0.2569	0.2477	0.2304	0.2414
误差/%	30.6%	26.3%	9.2%	18.8%	16.1%	14.3%	17.4%	23.2%	19.5%

从表中可以看出误差还比较大的但是还是在可以接受的范围内，主要原因之一是本来手指运动并不是特别稳定会导致一定的误差，其二可能是由于趋势项的消除，导致了测量结果偏小。为了验证加速度阈值对检测结果的影响，取加速度检测阈值为 4 的时候，对以上进行相同的处理得到以下图 5.5 和表 5.2。

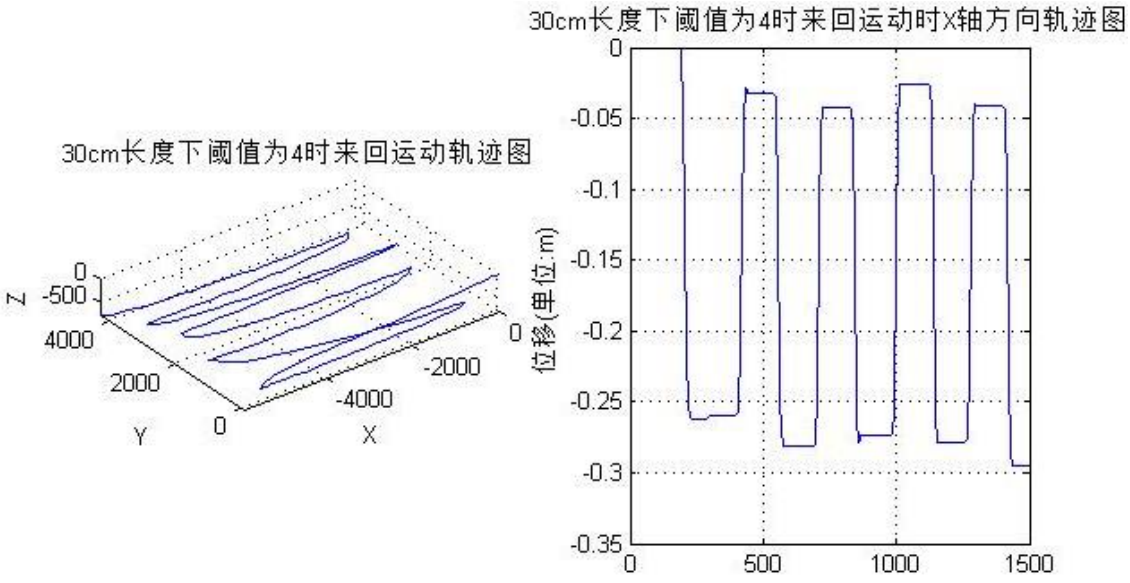

图 5.5 30cm 长度下阈值为 4 时来回运动的轨迹图

表 5.2 30cm 长度下阈值为 4 时不连续来回运动测量误差

序号	1	2	3	4	5	6	7	8	平均
位移大小/m	0.2592	0.2274	0.2490	0.2390	0.2318	0.2476	0.2525	0.2372	0.2432
误差/%	13.5%	24.2%	16.9%	20.3%	22.7%	17.4%	15.8%	20.9%	19.0%

由上面结果可见，当加速度设置的阈值减少时，手指运动参数的测量值趋于稳定一些，阈值设置为 4 时位移测量值的起伏变化较阈值为 10 时小一些。因此，在

测量一些简单但精度要求较高的运动时可以设置较小的阈值。但是，较小的阈值在实际应用中会导致手指的稍微一抖动就有位移出现，从而也会增加实际应用的不稳定性。因此，具体参数大小还需根据实际调整。

最后，为了探究 20cm 长度和 30cm 长度下不连续来回运动下测量结果的差异，同时取阈值为 4，测量在 20cm 长度下不连续来回运动下手指运动参数检测系统输出的结果，得到图 5.6 和表 5.3 如下面所示。

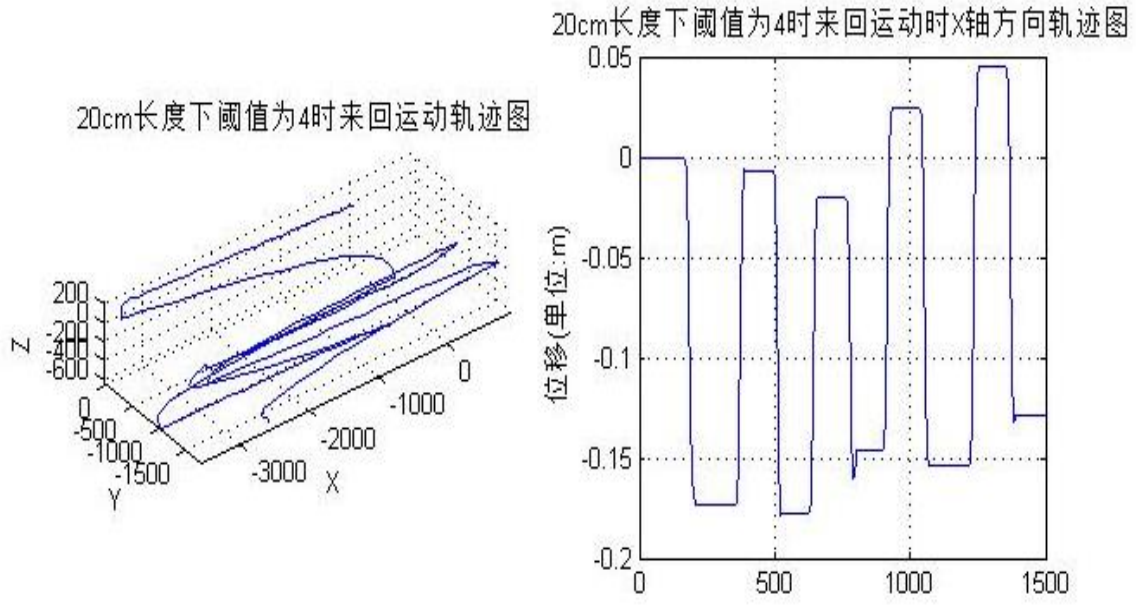

图 5.6 20cm 长度下阈值为 4 时来回运动的轨迹图

表 5.3 20cm 长度下阈值为 4 时不连续来回运动测量误差

序号	1	2	3	4	5	6	7	8	平均
位移大小/m	0.1735	0.1669	0.1707	0.157	0.1261	0.1714	0.1789	0.1991	0.1679
误差/%	13.2%	16.5%	14.6%	21.5%	36.9%	14.3%	10.5%	0.4%	16.0%

由上图可见，当在 20cm 长度下来回不连续来回运动的情况下误差相对于 30cm 长度时要小一些，位移长度为 30cm 时与 20cm 时的平均位移之比为 0.243/0.168≈1.45，同时 30cm/20cm=1.5，说明两者的测量结果和位移距离还是有一定的比例关系的。但是由于采用的计算方法的特殊性，两者的比例系数很可能也会与手指运动的快慢等其他因素影响。

总的来说，手指运动参数检测系统对于不连续往返式运动的检测平均误差在 20%左右并且测量的值一般小于实际值，更小的阈值可以提高测量的精确度但也会增

加实际应用中的不稳定性。同时，对于运动轨迹的检测，手指运动参数检测系统还是做得比较好的。

5.2 对于手指做连续直线往返式运动参数检测

对于连续运动的检测，由于处理加速度、速度和位移的趋势项中数据有太多的变化，因此，对于连续运动的检测主要检测其运动轨迹的正确性。对于手指做连续直线往返式运动的检测过程如前面小节中图 5.2 所示，模拟手指分别沿 20cm 和 30cm 长度连续不间断的来回运动，并从串口接收数据进行分析和处理。得到手指运动轨迹和 X 轴位移的图像如下面图 5.7 和图 5.8 所示。

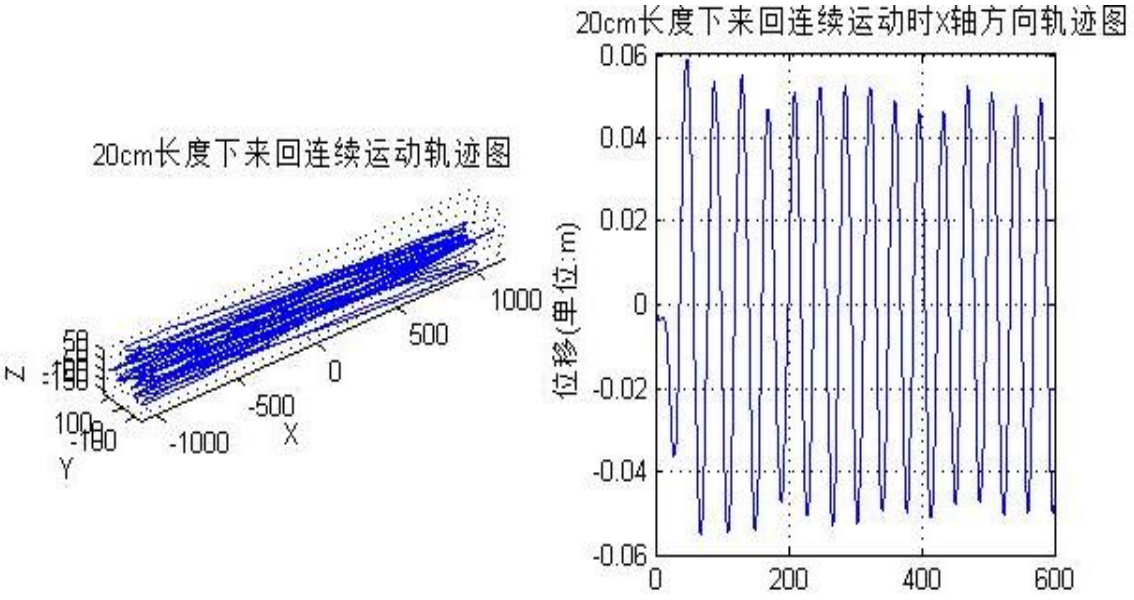

图 5.7 20cm 长度下来回连续运动的轨迹图

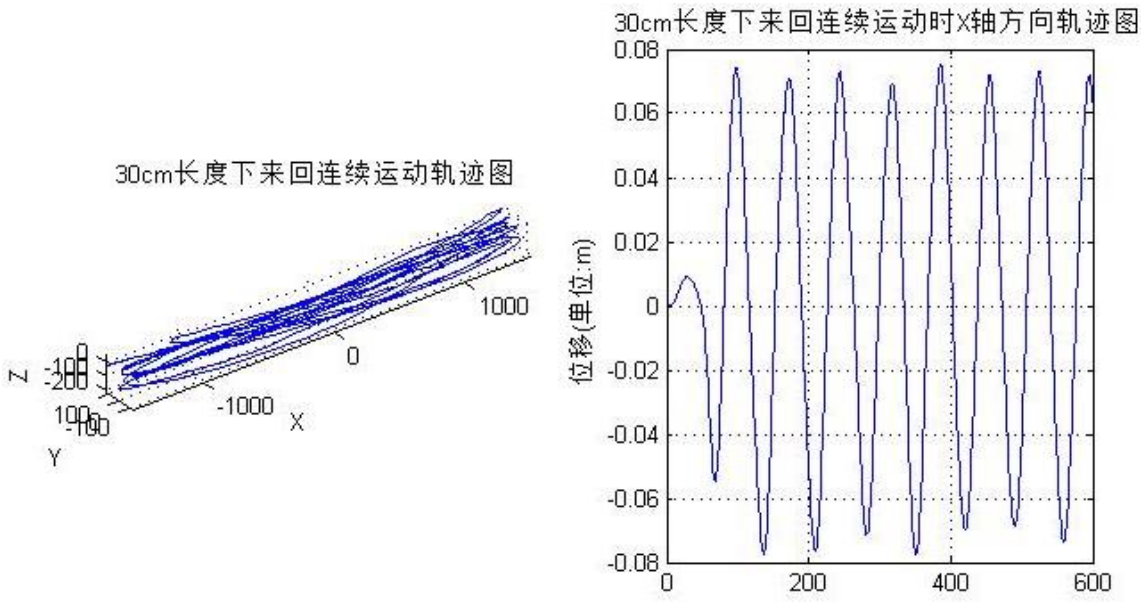

图 5.8 30cm 长度下来回连续运动的轨迹图

从上述结果可以看出，对运动轨迹的检测明显要优于第一种对不连续运动的算法，对于在 30cm 和 20cm 长度不断来回连续运动的情况下还是有很大的区别来识别的。但是测量出的位移均只有原来正确值的 1/2 左右，因此在实际应用中更适合做运动轨迹的检测，而不太适合做距离的测量。

5.3　对于手指做连续圆周运动参数检测

对于手指做连续圆周运动的情况下，试验中对沿正圆连续运动和沿椭圆连续运动分别做了检测。检测的过程照片如图 5.9 和图 5.10 所示。

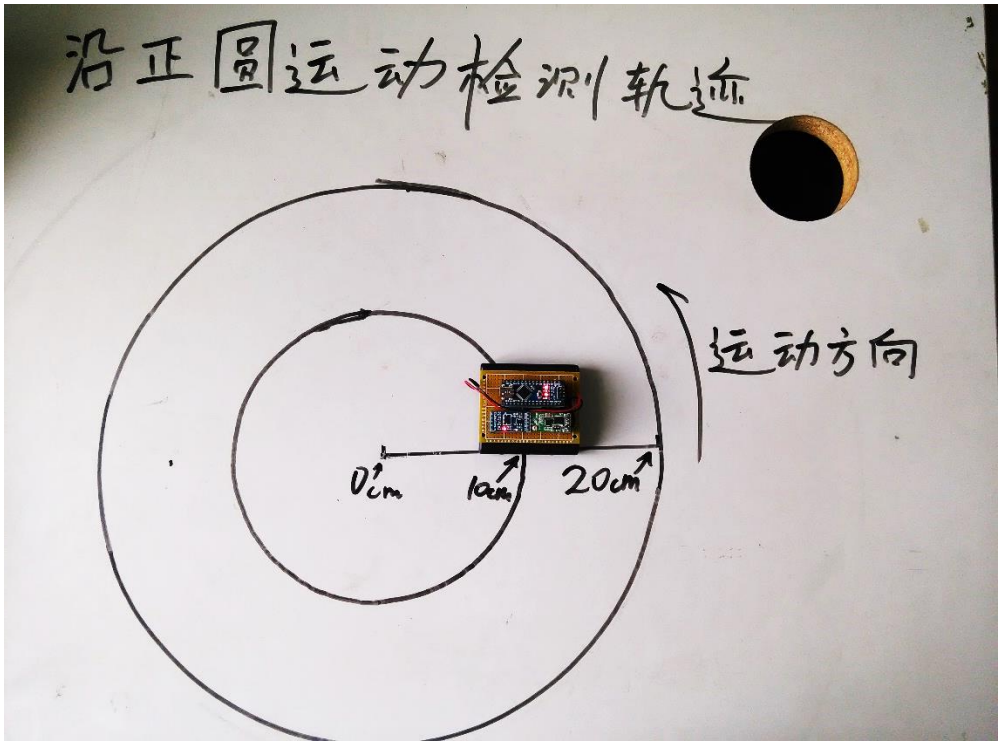

图 5.9 沿正圆做连续圆周运动检测

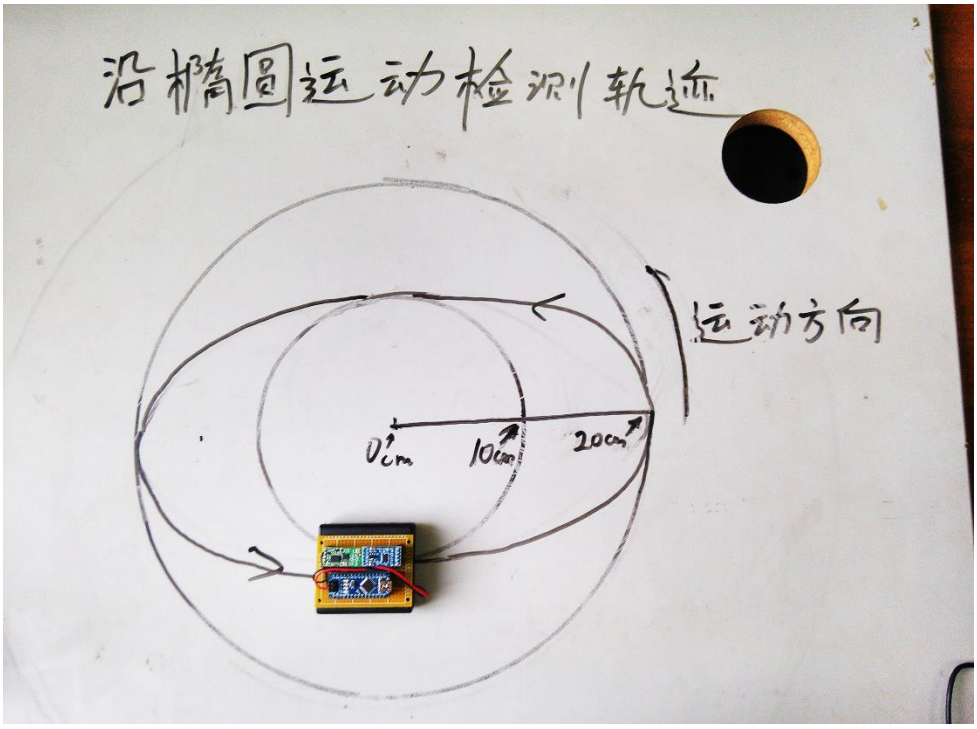

图 5.10 沿正圆做连续圆周运动检测

如上面图 5.9 和图 5.10 所示，首先分别沿半径 10cm 和 20cm 的正圆做连续圆周运动并检测结果，然后再沿长轴为 40cm 短轴为 20cm 的椭圆做连续圆周运动并检测结果。之后得到三者的轨迹图如图 5.11、5.12,、5.13 所示。

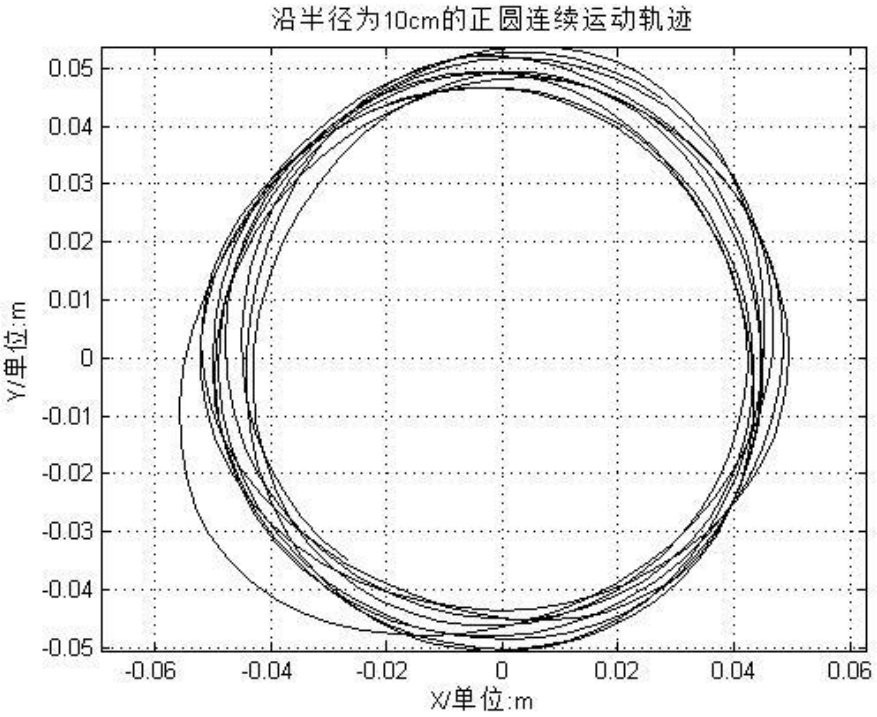

图 5.11 沿半径为 10cm 的正圆连续运动轨迹

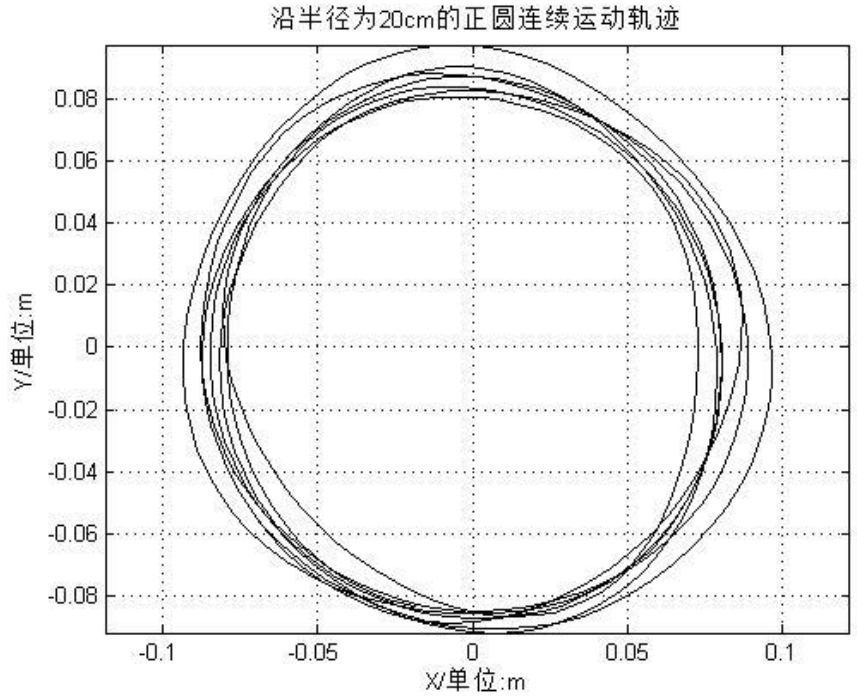

图 5.12 沿半径为 20cm 的正圆连续运动轨迹

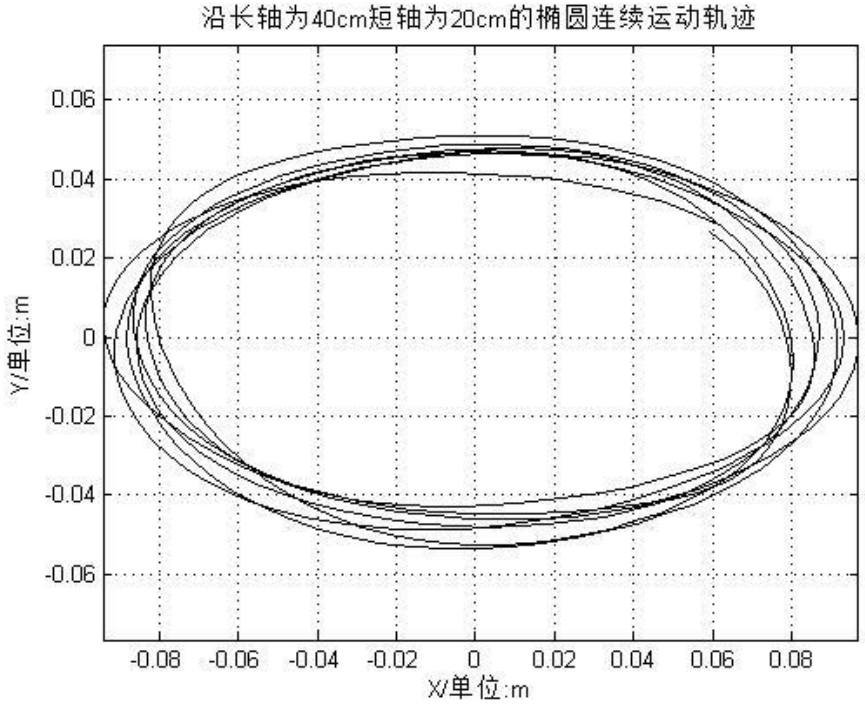

图 5.13 沿长轴为 40cm 短轴为 20cm 的椭圆连续运动轨迹

测得它们平均半径或者平均长轴和短轴的长度，并算出测量值与平均值的误差。最终得到结果如表 5.4 所示。

表 5.4 三种运动轨迹测量值与检测精确度比较

测量形状	大小	测量平均值	测量值与平均值比的平均误差	平均值与实际值比的误差
正圆	半径：10cm	半径：4.78cm	5.86%	52.20%
正圆	半径：20cm	半径：8.53cm	5.35%	57.35%
椭圆	长轴：40cm	长轴：18.22cm	5.39%	54.45%
	短轴：20cm	短轴：9.74cm	7.28%	51.30%

从上面数据可以看出轨迹检测的测量值和平均值比较的误差在 5%到 7%左右，同时，对于实际位移的测量则是原来值的一半左右，不过，对于轨迹的识别还是比较精确的。在实际应用中，第二种算法更偏向于在位移轨迹的检测中使用。

5.4 对于手指的倾角检测

手指运动参数检测系统对于手指倾角的检测如下面图 5.14 所示。通过依次改变三轴加速度传感器的倾角，并测量加速度传感器的角度值来验证结果的准确性。

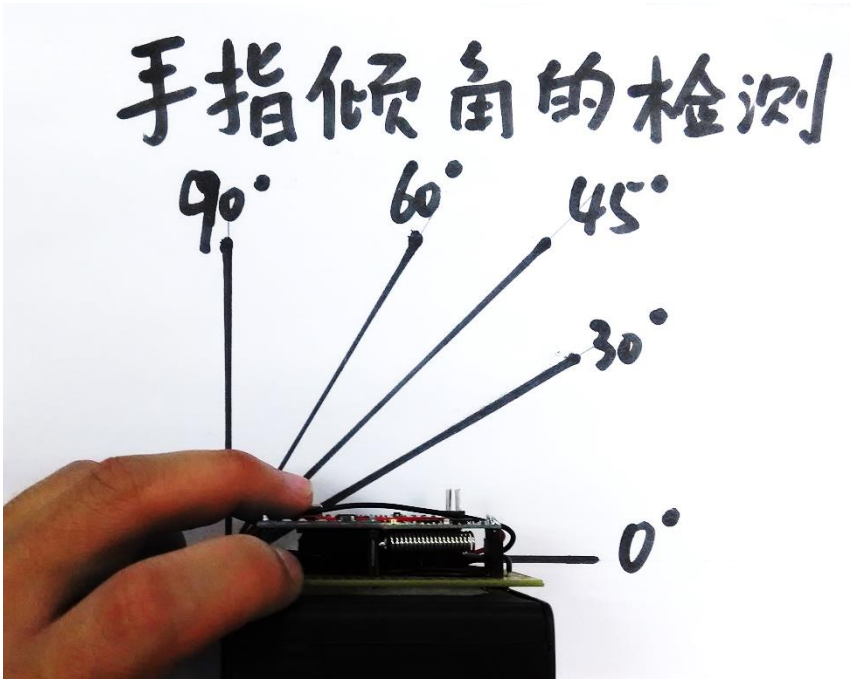

图 5.14 手指倾角的检测图

按照上面的图传感器角度分别取 0°、30°、45°、60°、90°然后测得传感器输出的角度值，得到表格如下。

表 5.5 对手指倾角的检测结果（单位：°）

角度值	0	30	45	60	90
测量值	-1.3	29.1	44.0	58.8	89.1
误差	1.3	0.9	1	1.2	0.9

从上面表中对比五组数据实际值与测量值的比较，可以看出对于倾角的检测是相对精确的，总体误差在 1.2 度以内。对于手指运动参数检测而言，还是能很好的满足一般的需求。

第六章　总结与展望

6.1　总结

总的来说，本文设计了一个基于三轴加速度传感器的手指运动参数检测系统，该系统由上位机和下位机组成，上位机主要基于计算机上的 VB.NET 程序，主要负责接收储存数据并画出轨迹和倾角，下位机主要基于 Arduino Nano 单片机，主要功能是采集三轴加速度传感器的数据并进行处理和计算出位移。两者通过无线模块传输数据，进一步提高了系统测量的便携性。

关于单片机算法的处理，本文提出了一种创新的利用三轴加速度数据计算位移的方法。该方法简单高效，相对于传统方法相比，运算量大大减少了，但是对于轨迹的测量仍具有很好性能。算法分为对不连续运动的处理和对连续运动的处理，两者都主要分为三个步骤，首先是通过对加速度原始数据的处理去除抖动和趋势项，再是加速度数据的积分处理，最后是对速度位移的校正和去除趋势项处理。

本文最后是对算法性能的评估。首先，在简单不连续运动算法下，对直线位移距离测量的误差在 20%左右，在实际应用中误差还是比较大。同时，对手指运动轨迹或者说运动手势的检测还是比较精确的。特别是在连续运动下，对于圆周和直线运动的检测结果都比较精确。总的来说，该算法更适用与对手势的检测。

6.2　展望

随着智能设备的发展，手指运动参数的检测也会越来越重要，手势控制也是近来一直很热门的一个研究方向。由于手指的灵活性，手指运动可以轻松的完成一些比较复杂的控制，从而可以完成更好的人机交互性能。就像上面提到的，手势交互可以很好地运用在游戏控制、大型设备控制上。同时，现在的手指运动参数检测都是基于一些高性能的设备来进行运算和检测的，如果这种手势检测能广泛地应用到微型低成本低性能的设备上，将会是一个很大的发展前景。

致谢

历时将近两个月的时间终于将这篇论文写完，在论文的写作过程中遇到了无数的困难和障碍，都在同学和老师的帮助下度过了。尤其要衷心感谢我的论文指导老师—余柏生老师和张伟涛老师，他们对我进行了无私的指导和帮助，不厌其烦的帮助进行论文的修改和改进。另外，在校图书馆查找资料的时候，图书馆的老师也给我提供了很多方面的支持与帮助。在此向帮助和指导过我的各位老师表示最中心的感谢！

感谢这篇论文所涉及到的各位学者。本文引用了数位学者的研究文献，如果没有各位学者的研究成果的帮助和启发，我将很难完成本篇论文的写作。

感谢我的同学和朋友，在我写论文的过程中给予我了很多你问素材，还在论文的撰写和排版灯过程中提供热情的帮助。

由于我的学术水平有限，所写论文难免有不足之处，恳请各位老师和学友批评和指正！

参考文献

[1] 徐涛，罗武胜，吕海宝等.地下定向钻进姿态测量系统的设计[J].中国惯性技术学报,2004,12(2):5-8.

[2] 袁西，陈栋，田湘等.三轴数字加速度计 ADXL345 及其在捷联惯导中的应用[J].电子设计工程,2010,3:138-139.

[3] 唐立军，滕召胜，陈良柱等.电子秤倾角自动检测与称量误差补偿方法研究[J].电子测量与仪器学报,2011,1:62.

[4] 段晓敏，李杰，刘文怡等.基于 MEMS 加速度计的数字倾角测量仪的设计[J].电子设计工程,2009,8:72.

[5] 潘春洪，马颂德．基于区域的手指三维运动跟踪[J]．中国图形图象学，2003，10：1205．1210

[6] 任留成. 空间投影理论及其在遥感技术中的应用[M]. 北京：科学出版社 2003.

[7] 马明建．数据采集与处理技术[M]．西安：西安交通大学出版社 2005.

[8] 沈兰荪．数据采集技术[M]．北京：中国科学技术大学出版社 1990．8.

[9] 李全利．单片机原理及接口技术[M]．高等教育出版社, 2009：158-165..

[10] Michael McRoberts .Arduino 从基础到实践., 电子工业出版社 2013:57-60.

[11]Ed Burnette. Hello, Android: Introducing Google's Mobile Development Platform . Pragmatic Bookshelf; 3 edition (July 20, 2010).

[12]卢聪勇. Arduino 一试就上手. 科技出版社. 2013:50-65.

[13]传感器应用技巧 141 例[美], 科学出版社. 2006:209-214.

[14]Joe Smiley Micros . An Arduino Workshop Pardue. 2010.

[15]董铮．基于 Arduino 控制板的温室大棚测温系统设计[J]．安徽农业科学报.2012, 40(8).

[16]张玉华. 基于 Arduino 控制板的光引导小车设计.自动化仪表.2011.

[17]程晨. Arduino 开发实战指南[美], 机械工业出版社.2012:78-85.

[18]Dale. Arduino 技术内幕 . 北京邮电出版社.2013.

[19]C 程序设计教程[美] 清华大学出版社. 2010:135-138.

[20]于欣龙,郭浩斌, 爱上 Arduino Massimo Banzi.人民邮电出版社. 2011:256-260.

[21]Simon Monk. .基于 Arduino 的趣味电子制作. .科学出版社. 2011.

[22]蔡睿妍. Arduino 的原理及应用[J]. 电子设计工程.2012:98-101.

[23]Massimo O'Reilly Media, Getting Started with Arduino Banzi, Inc, USA 2008.

[24]Simon TAB Books Inc. Arduino Projects for the Evil Genius Monk, 2010.

[25]徐文鹏. 计算机图形学[M]. 北京：机械工业出版社 2009.

[26]侯文生，戴加满，郑小林，杨琴，吴小鹰，许蓉. 基于加速度传感器的前臂运动姿态检测[D]. 重庆：重庆大学生物工程学院，2009.

[27]傅其凤，崔彦平，葛杏卫. 基于 USB 的虚拟振动测试系统的研制[D]. 石家庄：河北科技大学机械电子工程学院，2005.

[28]科峰. 基于 USB 接口的数据采集系统相关技术研究[D]. 绵阳：西南科技大学计算机学院，2007.

[29]陈财政，邢动秋. 基于 Matlab 的加速度传感器振动信号处理方法研究[D]. 西安：西安石油大学，2006.

[30]张书勇，马平，田沛，李伟，基于 Matlab 的列车加速度信号处理[D]. 保定：华北电力大学控制科学与控制工程学院，2006.

[31]孙苗钟，赵鹏. 基于 Matlab 的振动测试信号处理与分析系统设计与实现[D]. 天津：天津科技大学机械工程学院，2008.

[32]苗钟. 基于 Matlab 的振动信号平滑处理方法[D]. 天津：天津科技大学机械工程学院，2007.

[33]章毓晋. 图像处理和分析 EM]. 北京：清华大学出版社，2000：82—83

[34]吕凤军. 数字图像处理编程入门 EM]. 北京：清华大学出版社，1999

附录

手指运动参数检测系统 Arduino 程序：

```
#include "Wire.h"
#include "I2Cdev.h"
#include "ADXL345.h"
ADXL345 adxl;
int ax, ay, az, vx, vy, vz, xx, xy, xz;//用于储存处理过的 x、y、z 轴的加速度、速度、位移
int ax0, ay0, az0;//用于储存加速度传感器输出的原始加速度
int Tx, Ty, Tz;//x、y、z 轴的加速度趋势项
int Xt[101], Yt[101], Zt[101];//原始加速度
int AXt[15], AYt[15], AZt[15];//处理后加速度
int VXt[5],VYt[5],VZt[5];//处理后速度
int boundary = 20; //设置加速度阈值
int FS;//表示手指运动时间占总时间的程度，用于连续运动和不连续运动的检测
//整型数据发送
void INT16_SEND(int int16)   {
   unsigned int uint16;
   uint16 = int16 + 32767;
   Serial.write(uint16 >> 8);
   Serial.write(uint16);
}
//数据发送
void DATA_SEND()   {
   Serial.write(253);
   INT16_SEND(ax0);
   INT16_SEND(ay0);
```

```
    INT16_SEND(az0);
    INT16_SEND(ax);
    INT16_SEND(ay);
    INT16_SEND(az);
    INT16_SEND(vx);
    INT16_SEND(vy);
    INT16_SEND(vz);
    INT16_SEND(xx);
    INT16_SEND(xy);
    INT16_SEND(xz);
    Serial.write(254);
}
//发送数据到串口 2
void TEST1() {
    Serial.print(ax0);
    Serial.print("\t");
    Serial.print(ay0);
    Serial.print("\t");
    Serial.print(az0);
    Serial.print("\t");
    Serial.print(ax);
    Serial.print("\t");
    Serial.print(ay);
    Serial.print("\t");
    Serial.print(az);
    Serial.print("\t");
    Serial.print(vx);
    Serial.print("\t");
    Serial.print(vy);
    Serial.print("\t");
```

```
    Serial.print(vz);
    Serial.print("\t");
    Serial.print(xx);
    Serial.print("\t");
    Serial.print(xy);
    Serial.print("\t");
    Serial.println(xz);
}
//发送数据到串口 2
void TEST2() {
    Serial.print("\t");
    Serial.print(ay);
    Serial.print("\t");
    Serial.print(vy);
    Serial.print("\t");
    Serial.println(xy);
}
//取 101 个点中 40-54 个点,15 个点的平均值
int    F15S(int in[])    {
    int averange = 0;
    for (int i = 40; i < 55; i++) {
        averange += in[i];
    }
    averange /= 15.0;
    return averange;
}
//取 101 个点中 43-57 个点,15 个点的平均值
int    F15(int in[])    {
    int averange = 0;
    for (int i = 43; i < 58; i++) {    //43 58
```

```
        averange += in[i];
    }
    averange /= 15.0;
    return averange;
}
//101 点取平均值
int F101B(int in[])    {
    int averange;
    for(int i=0; i<101;i++){
        averange+=in[i];
    }
    averange/=101.0;
return averange;
}
//101 点取平均值（带一阶滤波）
int F101A(int in[], int *Tin)    {
    int averange = 0;

    for (int i = 0; i < 101; i++) {
        averange += in[i];
    }
    averange /= 101.0;
    if (*Tin == 0) *Tin=averange;
    else *Tin = 0.5 * averange + 0.5 * (*Tin);    //加速度一阶滤波
    return *Tin;
}
//加速度滤波+平滑处理（超过临界值才输出加速度）
void AProcess1( int *ai1, int *ai2, int *ai3,
                        int in1[], int in2[], int in3[],
                        int *Tin1, int *Tin2, int *Tin3 ) {
```

```
    int Apro1, Apro2, Apro3,F1,F2,F3;
    F1=F15(in1);
    F2=F15(in2);
    F3=F15(in3);
    Apro1 = F1 - F101A(in1, Tin1);
    Apro2 = F2 - F101A(in2, Tin2);
    Apro3 = F3 - F101A(in3, Tin3);
    if ((abs(Apro1) + abs(Apro2) + abs(Apro3)) > boundary) {    //加速度超过阀值是输
出
        *ai1 = Apro1-2;
        *ai2 = Apro2-2;
        *ai3 = Apro3-2;
    }
    else {
        *ai1 = 0;    //加速度置 0
        *ai2 = 0;
        *ai3 = 0;
    }
    if((abs(F15S(in1)-F1)<2)&&(abs(F1)<20))*ai1=0;
    if((abs(F15S(in2)-F2)<2)&&(abs(F2)<20))*ai2=0;
    if((abs(F15S(in3)-F3)<2)&&(abs(F3)<20))*ai3=0;
}
//简单的加速度滤波+平滑处理
int AProcess2(int in[]){
    return F15(in)-F101B(in);
}
//大致计算手指活动时间占总的时间的程度
void iABS(int *Trend,int in1,int in2,int in3){
    if((abs(*Trend)>362)&&(abs(in1)>2||abs(in2)>2||abs(in3)>3))
*Trend=360;//in1!=0||in2!=0||in3!=0
```

```
    else if(abs(in1)>2||abs(in2)>2||abs(in3)>3) *Trend+=2;
    else if(abs(*Trend)>20) *Trend-=20;
}
//加速度大于一定值时才计入速度变化
void VProcess1( int *vi, int in[]) {
    *vi += (in[1]+in[0])*0.02*100;
}
//当加速度均连续为 0 时的速度置 0 处理
void VProcess2( int *vi1, int *vi2, int *vi3,
                        int *xi1, int *xi2, int *xi3,
                        int in1[], int in2[], int in3[] ) {
    if ( (in1[8] == 0) && (in1[5] == 0) && (in1[3] == 0) && (in1[0] == 0) &&
            (in2[8] == 0) && (in2[5] == 0) && (in2[3] == 0) && (in2[0] == 0) &&
            (in3[8] == 0) && (in3[5] == 0) && (in3[3] == 0) && (in3[0] == 0) ) {
        //*xi1 -= 0.5 * (*vi1);//消除速度置 0 前所多算的位移长度
        //*xi2 -= 0.5 * (*vi2);
        //*xi3 -= 0.5 * (*vi3);
        *vi1 = 0;    //速度置 0
        *vi2 = 0;
        *vi3 = 0;
    }
    else if( (abs(*vi1) < 450) &&
                    (abs(*vi2) < 450) &&
                    (abs(*vi3) < 450) &&
                    (in1[0] == 0) &&
                    (in2[0] == 0) &&
                    (in3[0] == 0) ) {
        *vi1 = 0;    //速度置 0
        *vi2 = 0;
        *vi3 = 0;
```

```
    }
}
//加速度大于一定值时才计入速度变化
void XProcess1( int *xi, int in[]) {
    *xi += (in[4]+in[3])*0.02*2*2.5;//standard is 5
}
//位移边界设定
int XProcess2(int xi) {
    int bond = 10000;
    if (xi > bond)return bond;
    else if (xi < -bond)return -bond;
    else return xi;
}
//数组数据更新
void cir(int a[], int num, int temp) {
    int i;
    for (i = 0; i < num - 1; i++) {
        a[i] = a[i + 1];
    }
    a[i] = temp;
}
//获取加速度数据
int GetA1() {
    adxl.readAccel(&ax0, &ay0, &az0);//读取原始 3 个数据
    cir(Xt, 101, ax0);    //将原始加速度值存入数组，用于滤波
    cir(Yt, 101, ay0);
    cir(Zt, 101, az0);
    AProcess1( &ax, &ay, &az,
                    Xt,    Yt, Zt,
                    &Tx, &Ty, &Tz );
```

```
    cir(AXt, 15, ax);   //将处理过的加速度值存入数组
    cir(AYt, 15, ay);
    cir(AZt, 15, az);
    iABS(&FS,ax,ay,az);
}
//获取加速度数据
int GetA2() {
    adxl.readAccel(&ax0, &ay0, &az0);//读取原始 3 个数据
    cir(Xt, 101, ax0);   //将原始加速度值存入数组，用于滤波
    cir(Yt, 101, ay0);
    cir(Zt, 101, az0);
    ax = AProcess2(Xt);
    ay = AProcess2(Yt);
    az = AProcess2(Zt);
    cir(AXt, 15, ax);   //将处理过的加速度值存入数组
    cir(AYt, 15, ay);
    cir(AZt, 15, az);
}
//获取速度数据
void GetV1() {
    VProcess1(&vx, AXt);
    VProcess1(&vy, AYt);
    VProcess1(&vz, AZt);
    VProcess2( &vx, &vy, &vz,
                    &xx, &xy, &xz,
                    AXt, AYt, AZt );
    cir(VXt, 5, vx);   //将处理过的加速度值存入数组
    cir(VYt, 5, vy);
    cir(VZt, 5, vz);
}
```

```
//获取速度数据
void GetV2() {
   vx=vx*0.90+(AXt[1]+AXt[0])*0.02*100; //去除速度的偏移
   vy=vy*0.90+(AYt[1]+AYt[0])*0.02*100;
   vz=vz*0.90+(AZt[1]+AZt[0])*0.02*100;
   cir(VXt, 5, vx);   //将处理过的加速度值存入数组
   cir(VYt, 5, vy);
   cir(VZt, 5, vz);
}
//获取位移数据
void GetX1() {
   XProcess1(&xx, VXt);
   XProcess1(&xy, VYt);
   XProcess1(&xz, VZt);
   xx = XProcess2(xx); //设定位移的边界
   xy = XProcess2(xy);
   xz = XProcess2(xz);
}
//获取位移数据
void GetX2() {
   xx=xx*0.90+(VXt[4]+VXt[3])*0.02*2*3*2.5; //去除位移的偏移
   xy=xy*0.90+(VYt[4]+VYt[3])*0.02*2*3*2.5;
   xz=xz*0.90+(VZt[4]+VZt[3])*0.02*2*3*2.5;
}

void setup() {
   Serial.begin(115200,SERIAL_8E2);//串口初始化
   adxl.powerOn();//ADXL345 初始化
   delay(20);
}
```

```
void loop() {
  for (int i = 0; i < 100; i++)
  {
    adxl.readAccel(&ax0, &ay0, &az0);//读取原始 3 个数据
    cir(Xt, 101, ax0);
    cir(Yt, 101, ay0);
    cir(Zt, 101, az0);
    delay(18);
  }
  while (1)
  {
    delay(18);
    if(FS<300){ //检测到不是连续运动时，运行第 1 种算法
      GetA1();
      iABS(&FS,ax,ay,az);
      GetV1();
      GetX1();
    }
    else{   //检测到是连续运动时，运行第 2 种算法
      GetA2();
      iABS(&FS,ax,ay,az);
      GetV2();
      GetX2();
    }
    DATA_SEND();//发送数据给上位机 VB.NET 程序
    //TEST2();//发送数据到串口调试助手，仅用于程序检测
  }
}
```