基于加权右移平均法的精准运动分割

摘要

基于MEMS惯性传感器的运动数据提取技术一直是近年来智能穿戴领域的研究热点。本文通过利用**加权右移平均法**，建立数理模型对传感器数据进行处理，求解出手势动作及过渡动作的起止点。该问题的研究能提高系统对短时运动提取的精确度，为惯性传感器的下游应用提供技术支撑。

针对利用MEMS惯性传感器测得的三轴加速度、三轴角速度数据构成的多元时间序列对运动过程中62个动作起止点进行精准分割这一问题，我们考虑到加速度与角速度之间存在很大程度上的**协整关系**，因此先对加速度建立相应的数学模型，再将加速度与角速度进行线性组合套用上述模型提高精度。本文借鉴滑动平均滤波的思想及积分原理，对时间序列分析中常用的移动平均法进行了改造，设计出了一种向右取点的移动平均法，建立**局部最值模型**获取离散序列的极值。在此求平均值的方法中，我们编程出一种能根据滑动窗口的长度自动同步调整选取局部极值所需窗口长度的算法，再通过数据的纵向比较得出能使分割效果最精确的滑动窗口长度。最后，我们将加速度与角速度赋予不同的权重，对加权后的数据应用上述移动平均法，实现分割结果最优。最终，此方法较精确地求解出了Data1-Data5每个动作的起止点，并与标准数据差值在0.3s以内。

本文的亮点在于：首先，我们利用由特殊到一般的思想，结合标准数据对时间序列片段分析，发现了动作起止点出现时间的一般规律，进而建立数学模型进行分析；其次，利用一般的滑动平均滤波（moving average filter）的思想，**创新性地设计了右移滑动平均算法**，并结合**变上下限积分的原理**进行分析；第三，将窗口长度设置为调度变量，与机器学习思想相结合设计出最优化算法；第四，运用加权平均思想，充分利用加速度和角速度这两种数据，最大化地减小误差。经分析验证，本文提出的**加权右移平均法**对Data1-Data5均有较好的运动分割结果，具有普适性，并且对智能穿戴的传感器识别技术具有现实意义。

关键词：移动平均法 多元时间序列分析 加权平均算法

# 问题重述

## 问题背景

MEMS 惯性传感器体积小、采样率高、精确度高，可与其他多种传感器集成在智能手机等电子设备上，同时采集多类运动状态数据，应用场景广泛。同时，利用此种传感器完成对运动过程提取分析，对比起其他外部观测法，具有成本低，外界制约小，低能耗等优点，因此MEMS 惯性传感器拥有广阔的市场需求。由于实际应用中传感器会持续采集数据，因此需要精确地从数据流中提取“短时运动数据”，因此对惯性传感器收集到的数据建立相应的运动速度，加速度，角速度的数学模型，实现对运动过程的精准分割，从中提取出具有完整语义信息的动作，已经成为研究的重要课题，也是本题的核心任务。

## 问题重述

用户佩戴惯性传感器，一共做了包括英文字母大小写和数字0到9，一共26\*2+10=62个对应的手势动作，以及两个手势间的过渡动作。在这约150-200秒的运动过程中，传感器记录了采样率为0.005s的时间序列中x，y，z三轴加速度及角速度，并将其存储在表格中。本文需要对表格中的数据进行建模分析，来判定每个动作的开始时间（帧）和结束时间（帧）进行语义动作提取。传感器共采集了5次运动数据(data1-data5)，本题任务即提取出5次所有动作的起止时间，将结果存在表格中。

# 问题分析

## 确定思路（如图一）

针对题目任务——分割时间序列中的动作起止点，我们结合给出的示例视频及物理学知识，从任务出发，得出以下分析：

首先，选取分析区间。**设定适当加速度阈值的方法进行活动窗口检测[1]，从而得到整个运动过程的起止点**（即对应*1*中的-2时间点）再进行单运动分割。

其次，选取参考数据。查阅相关资料可知，陀螺仪更多关注自身的旋转情况（原位运动），而加速度计更适合于空间运动的判断。因此本模型**先以MEMS加速度计测得的加速度情况进行分析，最后再用加权的方法结合角速度数据对模型进行优化。**

## 数据优化

**数据预处理**：基于传感器产生的噪声，以及数据有毛刺的现象，我们综合比较了**移动平均法、**三次样条插值、滑动平均滤波等曲线平滑算法，发现最终采用时间序列分析中常用的**移动平均法**效果最优。（下文中，分别将某个时间点左边，右边的函数值的平均数作为该点的函数值称为**左滑，右滑**）。运用对一段时间序列取平均的方法，可降低自然散乱的时间序列的折线波动，消去部分噪声。

同时由于运动时手部抖动，等间隔采样采集到的加速度信号频率差别较大，因此，我们尝试建立其他降噪模型（如傅里叶变换，小波变换，正弦升采样方法等）以减小误差，但由于效果不理想，故舍弃。

分析Lable\_of\_Data1发现在第88个动作起止时间差为负值，动作起始时间晚于终止时间，不合逻辑，推测此现象是由于数据输入错误造成的。因此，我们**手动剔除参考数据中的异常数据**，以便后续优化模型。

## 模型构建

基于动作结束的一段时间内加速度局部最小，因此**移动平均法**运用**右滑**取的是小加速度的平均，形成的函数图像极小值对应终止动作时刻；而运动起始点后短时间内加速度较大，**移动平均法**运用**右滑**取的是大加速度的平均，形成的函数图像极大值对应动作起始时刻，因此**移动均值模型**能对运动的起止点具有较好的预测效果。

将模型得到的结果与标准结果进行绘图对比，倘若得到多于62个分割点，推测其可能由传感器误差导致多重分割，尝试采用K-means聚类算法、系统层次聚类法将较近的分割点合并，但我们发现反而会造成已对齐的序列集中错位，而舍弃。因此我们考虑**优化极值点分析算法**，实现模型自动调整选取的滑动数及滑动窗口长度，智能确定分割点个数，使运动分割更精准。

## 模型升级

基于本题含加速度，角速度等双变量，为多元时间序列，考虑多元时间序列分析中的ARMA模型，但发现其无法达到运动分割的目的，故舍弃。由于加速度、角速度都能在一定程度上反应运动状态，根据多变量问题中的消元思想，我们先将数据进行归一化处理消去单位，再**乘以相应的权重得到线性组合后的新变量**，基于此新变量套用原来对加速度分析的数学模型进行处理得出权重的最优解。

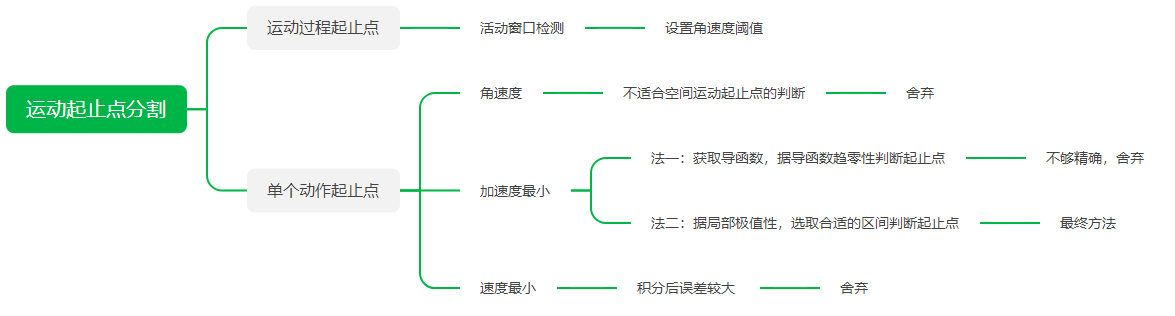
图1.思路分析



图2 求解过程分析

# 模型假设

1. 假设加速度计与陀螺仪是正交坐标系
2. 光学传感器标定的动作起止点数据基本准确
3. 用户手部在每个动作中不出现长时间匀速运动

# 符号说明

表 1：符号说明

|  |  |  |
| --- | --- | --- |
| 符号 | 说明 | 单位 |
|  | 角速度的时间序列 | deg/s |
|  | 加速度的时间序列 | m/s2 |
|  | 滑动数 | 个 |
|  | 标准数据的时间序列(目标值) | \ |
|  | 运用移动平均法后的时间序列（预测值） | \ |

# 模型的建立与求解

## 模型建立

**移动平均法[2]**是时间序列分析中的常见方法，可以用来估计变量的局部均值，使得变量的更新与一段时间内的历史取值有关。具体方法是，根据时间序列资料逐渐推移，依次计算包含一定项数的时序平均数，以反映长期趋势的方法。当前时间序列的数值由于受周期变动和不规则变动的影响，起伏较大，使用移动平均法，消除这些因素的影响。

|  |  |
| --- | --- |
|  | () |

选取当前观测即其过去的个数据还有未来个数据取平均，，代替本时间点的数据，称为**m阶移动平均**，阶数越高曲线越平滑。结合题目特点，我们独创性地设计了向右滑动平均法：

|  |  |
| --- | --- |
|  | () |

取当前观测的数据与右边个数据计算平均值来取代当前观测点的数据，其中称为滑动窗口的长度。同理我们也可以定义向左滑动平均法。

在时间序列的采样间隔足够短的情况下，在误差允许的范围内，我们可以将其看作是**对离散点的定积分**。

由于时间序列含长达将近五万条数据，数据点散乱，绘制在同一张图上不利于对每个微观动作的分析。为了研究对各语义动作分割的规律，我们随机截取了多段时间序列的一部分放大进行分析，将离散点连线绘制成可视化图像，并将参考数据提供的语义动作起止点标注在图像上绘制成函数，希望借助从特殊到一般的思想将局部结论推广到整体达到解决问题的目的。由传感器测得的三轴加速度数据，剔除重力因素的干扰，可得计算加速度的表达式：

|  |  |  |
| --- | --- | --- |
|  |  | () |

据此绘制出图像

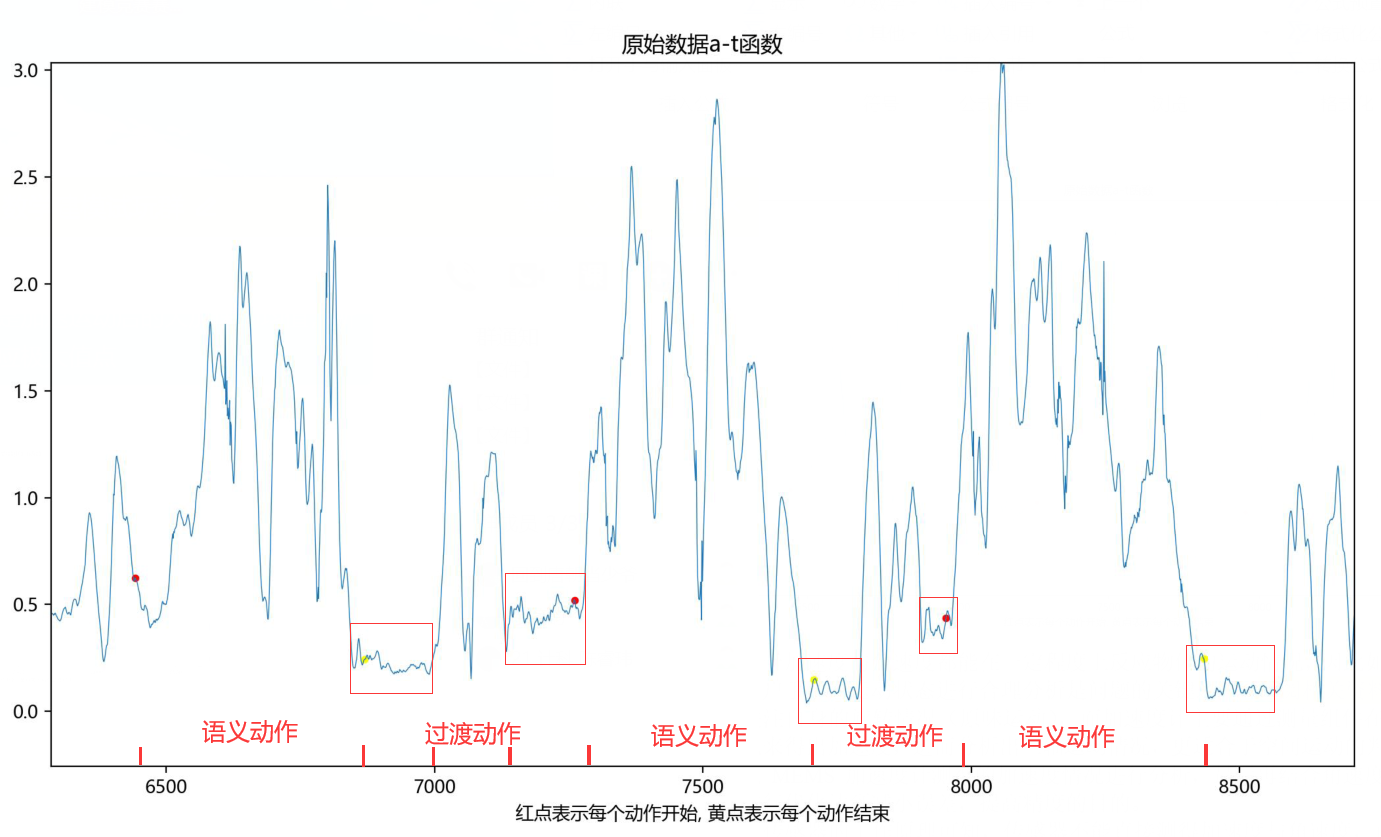


图3.原始数据a-t图

图像分析：通过分析红点（动作起始点）、黄点（动作结束点）我们发现，它们绝大部分总处于一段加速度较小的平稳区段，并且黄点位于平稳区段的左方，红点位于平稳区段的右方，根据参考点的位置，我们推断出函数图像峰值高且剧烈不规则抖动的部分为语义动作所对应的时间序列，函数趋于平稳的阶段是分割点所在的区段，起止点分别为区段右、左端点。

原因分析：由*运动示例.mp4*并结合物理知识和生活实际也不难理解，用户在前一个语义动作结束后于空中悬停了一段时间，此时加速度维持在较小的水平，然后做过渡动作导致加速度增大，过渡动作持续时间短于语义动作，接着又悬停一小段时间，准备下一个语义动作，此后加速度又剧烈上下震荡，但整体数值偏高。

另外，我们发现动作结束点所在波段高于动作起始点，而正常情况下两波段都非运动过程中，加速度理应都等高且接近0，我们查阅资料分析了误差来源。由传感器的工作原理，传感器是通过内部的可移动部件，将位移信号转化为电信号，再通过计算得到当前的运动加速度矢量，因此不可避免地产生由硬件方面造成的误差，如陀螺仪由零点漂移产生的误差，加速度在载体运动中受到环境的干扰等等，这统称为系统误差。

由以上分析我们得出了分离出运动起止点的方法：对时间序列进行向右滑动平均，得出新时间序列图像。由于动作终止点位于加速度较小的平稳波段左端点，因此我们右滑取的是小加速度的平均值，对应的是右滑所生成图像的极小值点。由于动作起始点位于较小平稳波段的右端点，因此我们右滑取的是大加速度的平均值，对应右滑生成函数的极大值点，因此采用该模型能将运动起止点较好分割。

从本质来说这种方法可以看作采用恒定积分区域长度的变上下限积分处理加速度数据，由于时间序列采样率为，采样时间间隔仅为，离散点足够密集，因此离散点的函数值乘以时间间隔所表示的极小段矩形区域的面积累加可近似看作离散序列与轴围成的面积。据此，我们可以定义如下对离散序列的积分：

|  |  |
| --- | --- |
|  | () |

将称为点的右滑积分，其中，表示滑动窗口的长度，我们构造的右滑函数也可以表示为如下形式：

|  |  |
| --- | --- |
|  | () |

将图（1）中的语义动作片段抽象出来，我们可以得到如下模型：

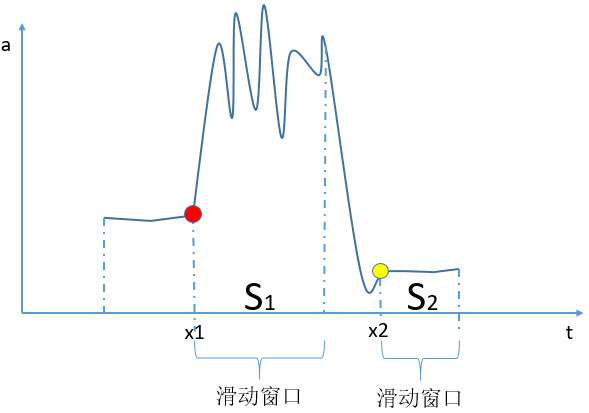


图4.滑动窗口示意图

可以看出，选取合适滑动窗口后，动作起始点的右滑积分即表示，动作结束点的右滑积分即表示，的面积非常大，则非常小，这是区别于时间序列其他点的显著特征，另外，由于起始点所在平稳区域高于结束点，左段的右滑积分小于的右滑积分，因此不会影响到极小值点的提取，据运动起止点的此特性我们可以计算出右滑积分的极大极小值，即分别对应动作起止点。由于运动起始点与终止点右侧函数具有不同特性，我们可以针对运动起止点与运动终止点分别选择不同的滑动窗口长度与使其分割效果准确。

## 模型求解

**步骤一：数据预处理**

传感器测得的原始数据无法直接用于数据挖掘与分析，因此为了提高模型准确度，防止异常数据对模型结果造成较大干扰，有必要进行数据预处理的工作。

1.剔除异常数据

我们分析了题干所给出的参考数据*Label\_of\_Data1.csv*，计算各个动作的起止时间差，希望据此判断出每个动作的持续时间，发现在第88个动作时出现了明显的异常值，动作起始时间晚于结束时间，猜想这是由人为因素导致的错误，故舍去。

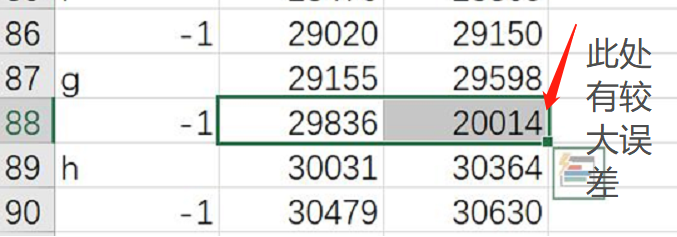
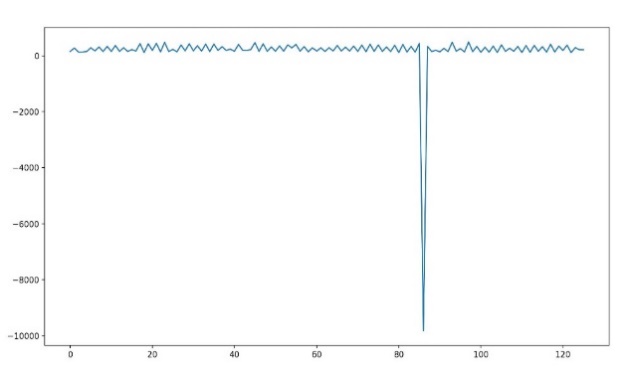


图5. 动作持续时间与动作序号的函数图像 图6.对应的表格数据位点

2.舍弃与运动过程无关的首尾时间序列。

观察Data1及Label\_of\_Data\_1，传感器在一开始（约900帧前）及数据最后（约44000帧后）各方向角速度趋于0或固定值0.0061deg/s，可以判断开始一段时间与最后一段时间，数据无有效语义信息。但此时由于惯性传感器自身产生的漂移误差，加速度不趋近于零。基于此，我们设定加速度阈值为0.005，进行对时间序列首尾进行活动窗口检测，当检测到数据触发了大于阈值的条件便将其设为数据流的起止点（即对应Label\_of\_Data\_1中的-2时间点）分别标记为与。

**步骤二：向右滑动平均**

运用模型中的方法，将*Data1.csv*导入到中，设计右滑函数对数据进行处理。我们设置滑动窗口的大小为，事实上这种类似于非对称滑动平均滤波的算法也能在一定程度上滤去传感器所产生的噪声与数据的毛刺，使得数据更为平滑，有助于分析出数据的变化趋势。

**步骤三：分析离散序列的极值点**

获取极值点有两种方法：其一，对离散序列插值拟合构造连续函数，获取导函数零点。其二，选取合适长度的开区间，将区间内最值设为局部最值点。前者鉴于运动过程中加速度的不连续性及不规则变动，易将原本完整的语义动作分割，效果不佳。而后者在适当长度的区间内最值唯一，更合题意。

**局部最值法**求**加速度的极小值**确定时间点：设定开区间长度为，遍历时间序列，若时间序列中某时间点的右滑函数值在此点范围内最小则将其设为动作结束点，最大则设为动作开始点，分别存入列表中。

**步骤四：确定极值点个数**

鉴于用户比划了及字母大小写等62个语义动作，理论上我们需要个分离出62个极大、极小值点。我们的模型在适当调整与两个变量的长度后都能完美地分割出62个，但是考虑到仍可能存在多出或少出零星点的情况，因此我们考虑优化极值点分析算法，基于机器学习的思想，我们让程序能自动调整窗口长度，精准选取出62个极大、极小值点。

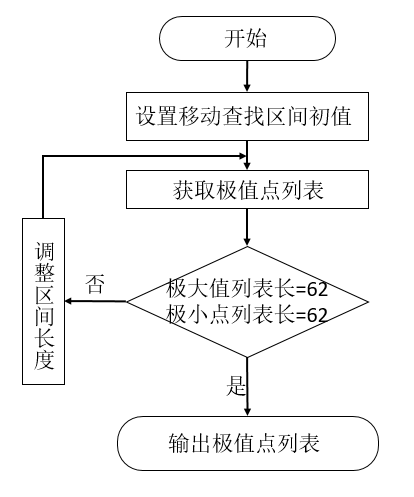


图7.模型求解示意图

# 模型的分析，检验与优化

## 模型的分析

在已分割出所有动作的基础上，我们对模型求解的结果与参考数据比对进行定量分析，比较我们的预测与实际的误差，并对模型进行优化达到减小误差、提高精度的目的。

平均绝对误差MAE是一种常用的回归损失函数，它是目标值与预测值之差绝对值的和，表示了预测值的平均误差幅度，而不需要考虑误差的方向，其公式如下所示：

|  |  |
| --- | --- |
|  | () |

我们代入用模型计算出的预测值与和参考值与计算平均绝对误差，下面是根据不同的右滑窗口长度和程序计算出的分析出的和（具体数据见附录文件……）：

**表1改变滑动窗口对预测结果的影响**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

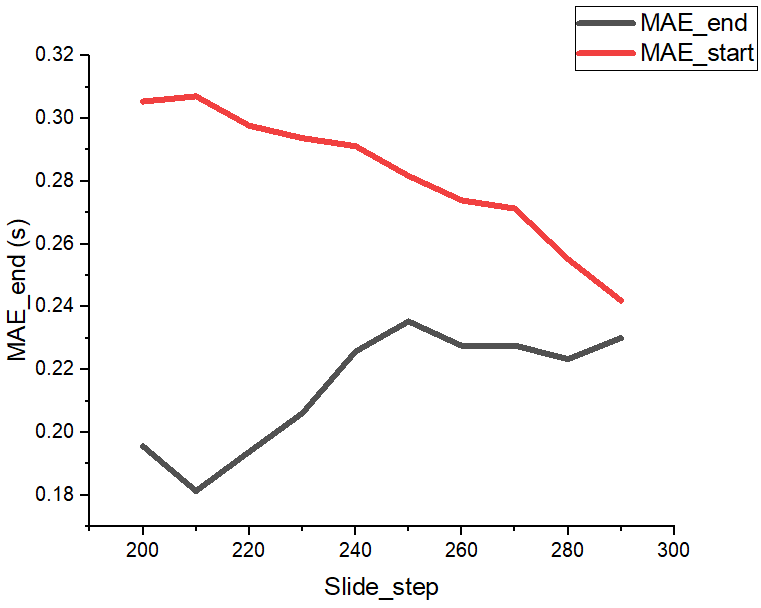


图8. 滑动窗口对预测值的影响

我们将对各个语义动作的预测误差绘制成图如下：（绘制图像的文件见文末）

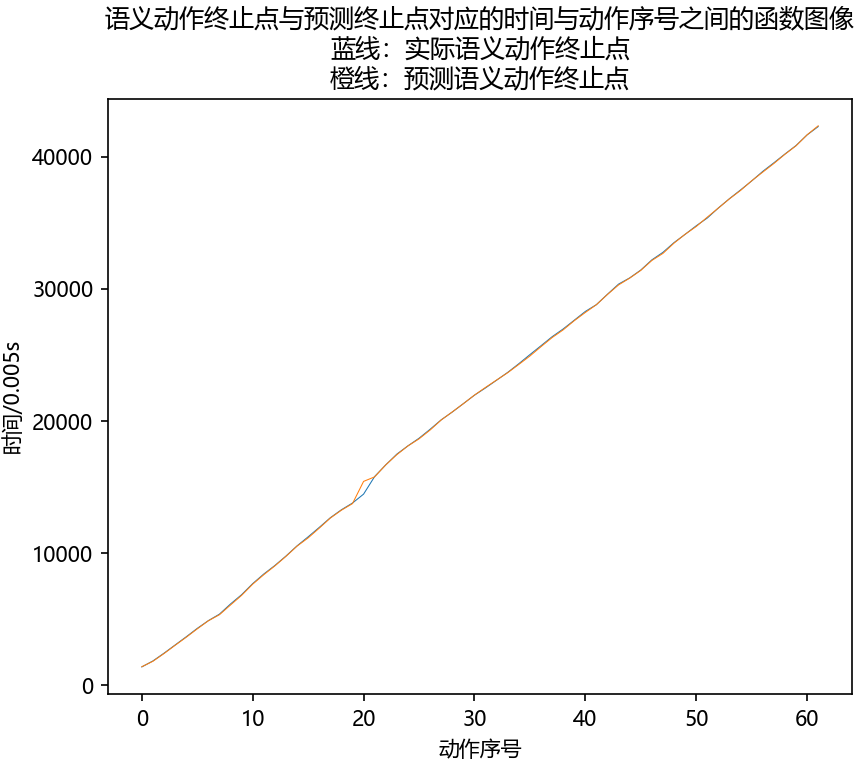
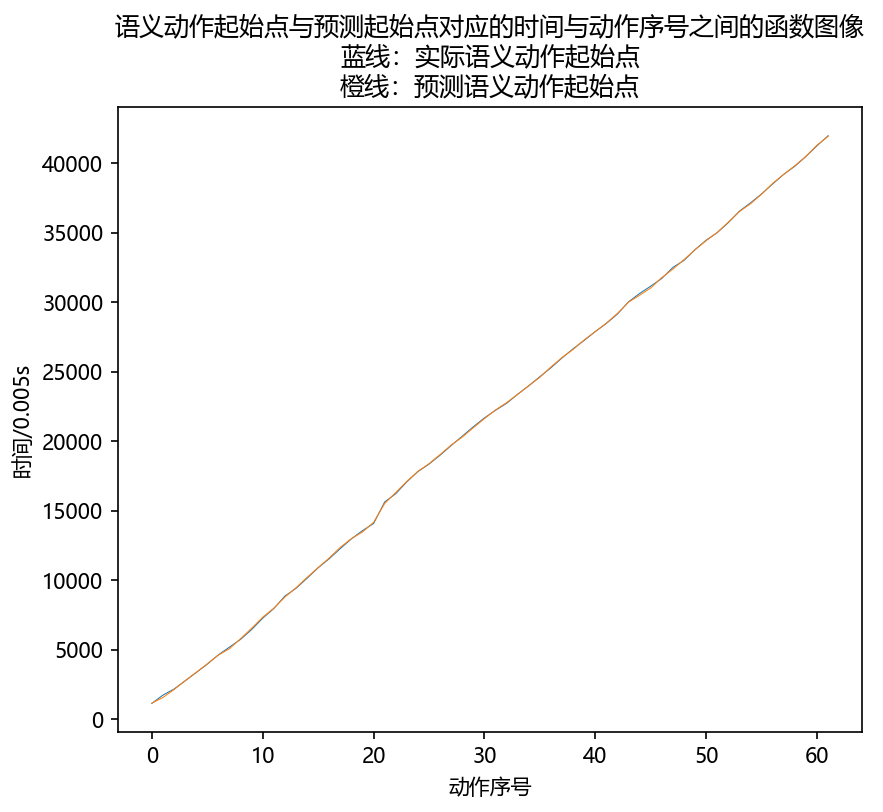


图9.误差分析图像

## 模型的检验

经比较后发现，对除字母外其余动作的起止时间的预测非常准确，两者图像几乎重合，但是对结束时间的预测竟与参考数据相差900多帧，为了探究其原因我们计算了所给参考数据每个动作的间隔时间，下面为最长的个语义动作数据：

**表2参考数据两个动作**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | 14475 | 15623 | 5.74 |
|  | 36122 | 36558 | 2.18 |
|  | 29598 | 30031 | 2.165 |
|  | 40842 | 41267 | 2.125 |
|  | 8433 | 8850 | 2.085 |

可以看出，用户在比划完后竟过了之久才开始下一个动作，推测是由期间手部晃动产生非语义信息被识别出来，倘若将这段异常情况剔除，我们得到在滑动窗口为210时对结束动作的判断误差仅0.18s，可见模型预测之准确。

纵向对比表格中的数据，我们选取的运动起始点最佳滑动窗口大小为290，运动终止点最佳滑动窗口大小为210，当滑动窗口大于300时数据发生了错位偏离，未列出，最终起始动作的预测时间偏差为，终止动作的预测时间偏差为。

## 模型的优化

鉴于全运动过程为含加速度变量与角速度变量的多元时间序列，加速度与角速度两序列之间具有非常密切的长期均衡关系，两者间是一种协整关系，并都能在一定程度上反映物体的运动状态。

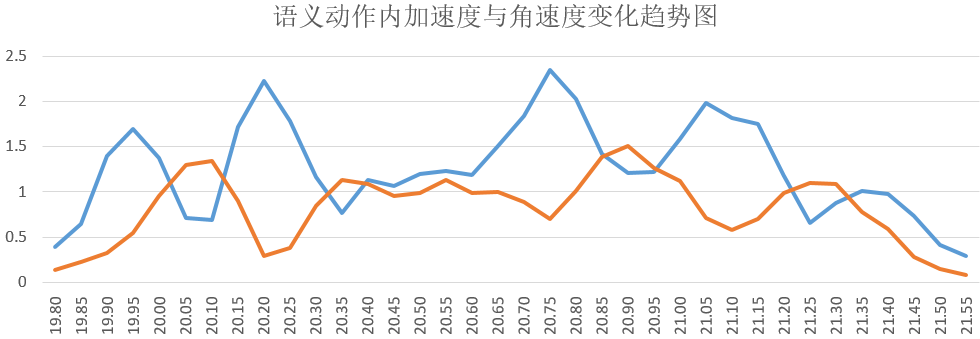


图10.语义动作内加速度与角速度变化趋势图

基于对加速度数据分析的模型我们已经得到了最优解，现在我们考虑引入角速度这一变量，分析出加速度与角速度之间的最佳加权比例，将复杂的多变量处理问题化归为单变量分析，我们得到新变量

|  |  |
| --- | --- |
|  | () |

由于模型中取的是右滑函数的极大值点与极小值点，因此右侧表达式的系数不影响最终结果，我们可以令，改变的系数即可。

归一化处理：由于不用评价指标往往具有不同的量纲，为了消除指标之间的量纲影响，需要进行数据标准化处理，以解决数据指标之间的可比性。原始数据经过数据标准化处理后，各指标处于同一数量级，适合进行综合对比评价。

查阅相关资料可知，加速度计相比于陀螺仪更适合做空间运动的判断，因此加速度所赋的权重更大，基于此我们考虑将角速度作为辅助变量进行判断，只对角速度进行归一化处理。

首先由三轴角速度可以得出时间序列中某点的角速度：

|  |  |
| --- | --- |
|  | () |

下面采取最大最小标准化的方法：

|  |  |
| --- | --- |
|  | () |

使用线性函数将原始数据线性化的方法转化到的范围，表示原始数据，计算结果为归一化后的数据。通过调试的系数我们可以得出如下结果：

表3. 双变量权重及相应误差

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 | 310 | 0.18131 |
| 0.1 | 310 | 0.17836 |
| 0.2 | 310 | 0.17163 |
| 0.3 | 310 | 0.17082 |
| 0.4 | 310 | 0.17123 |
| 0.5 | 310 | 0.17213 |
| 0.6 | 310 | 0.17556 |
| 0.7 | 310 | 0.17811 |
| 0.8 | 310 | 0.18360 |
| 0.9 | 310 | 0.19418 |
| 1 | 310 | 0.19377 |
| 2 | 310 | 0.19885 |

据上表我们可以得出的结论是当与的系数比时对动作结束点的预测最准确。另外我们以相同原理对动作起始点进行了调试，发现的加入对预测结果影响不大，调试反而多数情况下使得预测结果更偏离，因此对起始点的预测应以为准。

# 模型的评价、改进与推广

## 模型的优点

1. 原创性非常强，文章中的大部分模型都是基于我们已学的数学知识自行推导建立的。
2. 对运动起止动作的分割结果十分准确，平均误差在0.3s左右，对运动结束的误差仅。
3. 能针对不同滑动长度自动选取出62个预测起止点，具有很好的通用性和推广性。
4. 能基本过滤掉本题中的不含语义信息的过渡动作。
5. 综合运用加速度计和陀螺仪测得的数据，两者对两者进行线性组合，提高了预测的精准度，降低了仅据一方判断造成的误差。。

## 模型的缺点

1. 对未知语义动作个数的运动过程预测效果一般。
2. 仅据第一组参考数据选择的最优滑动窗口长度与权重不一定为其他四组效果的最优解。
3. 滑动长度与权重选取不当可能造成即使程序能自动调整窗口长度仍不能得到62个分割点

## 模型的改进

1. 可设计算法使模型也能根据数据自动选取滑动长度与权重
2. 可考虑运用卡尔曼滤波算法消除加速度计、陀螺仪的零偏误差，从而对三轴加速度二次积分成为可能，得到空间运动轨迹从而更加精准判断运动分割点

# 参考文献

[1]刘蓉, 刘明. 基于三轴加速度传感器的手势识别[J]. 计算机工程, 2011, 37(24):141-143.

[2]张鹏, 高放, 阚宏林,等. 一种改进的加速度传感器数据预处理方法[J]. 仪表技术, 2017(6):4.

# 附录

|  |
| --- |
| **附录1** |
| 介绍：使用python生成预测点的代码 |
| import **numpy** **as** **np**  import **csv**  import **math**  import **matplotlib.pyplot** **as** **plt**  from **numpy** **import** \*  *# 读取表格中三轴加速度*  **def** read\_data\_a(x):  a\_ls = []  **with** open('./数据/Data' + x + '.csv') **as** fp:  rows = csv.reader(fp)  next(rows) *# 跳过首行*  **for** row **in** rows:  ax = float(row[1])  ay = float(row[2])  az = float(row[3])  a = math.sqrt(ax \*\* 2 + ay \*\* 2 + (az - 1) \*\* 2)  a\_ls.append(a)  **return** a\_ls  *# 读取表格中三轴角速度*  **def** read\_data\_w(x):  w\_ls = []  **with** open('./数据/Data' + x + '.csv') **as** fp:  rows = csv.reader(fp)  next(rows) *# 跳过首行*  **for** row **in** rows:  wx = float(row[4])  wy = float(row[5])  wz = float(row[6])  w = math.sqrt(wx \*\* 2 + wy \*\* 2 + wz \*\* 2)  w\_ls.append(w)  **return** w\_ls  *# 设定加速度阈值判断起止时间*  *#如果一个点所在的range内，有90％的加速度小于阈值，则将这样的点存入列表里*  *#在序列的前五分之一和后五分之一内各建两个列表，最后返回列表的最值即可*  **def** begin\_and\_end(a\_ls, step):  begin\_ls = []  end\_ls = []  begin\_and\_end\_ls = []  max\_value = 0.05  **for** i **in** range(step, len(a\_ls)//5):  cnt = 0  **for** j **in** range(step):  **if** a\_ls[i-j] < max\_value:  cnt += 1  **if** cnt >= step\*9/10:  begin\_ls.append(i)  **for** i **in** range(len(a\_ls)-step - 1, len(a\_ls)\*4//5, -1):  cnt = 0  **for** j **in** range(step):  **if** a\_ls[i+j] < max\_value:  cnt += 1  **if** cnt >= step\*9/10:  end\_ls.append(i)  begin\_and\_end\_ls.append(max(begin\_ls))  begin\_and\_end\_ls.append(min(end\_ls))  **return** begin\_and\_end\_ls  *# 读取文件*  file\_num = '1'  a\_ls = read\_data\_a(file\_num)  w\_ls = read\_data\_w(file\_num)  x\_ls = range(len(a\_ls))  *# 运用最大最小标准化方法将角速度归一化*  w\_max = max(w\_ls)  nor\_w\_ls = []  *# 设置角速度系数加权处理，其中coefficient人工调整*  coefficient = 0  **for** i **in** range(len(w\_ls)):  nor\_w\_ls.append(w\_ls[i]/w\_max)  a\_ls[i] += nor\_w\_ls[i] \* coefficient  step = 200  begin\_and\_end\_ls = begin\_and\_end(a\_ls, step)  x\_start = begin\_and\_end\_ls[0]  print('加速度开始变化点: ', x\_start)  x\_end = begin\_and\_end\_ls[1]  print('加速度结束变化点: ', x\_end)  *# 右移平均法*  slide\_step = 210  x\_slide\_ls = range(len(a\_ls))  a\_slide\_ls = []  **for** i **in** x\_slide\_ls:  **if** x\_start <= i < len(a\_ls) - slide\_step:  sum\_a = 0  **for** j **in** range(slide\_step):  sum\_a += a\_ls[i + j] / slide\_step  a\_slide\_ls.append(sum\_a)  **else**:  a\_slide\_ls.append(0)  *#局部最小值模型*  *# 程序自动选取取+-arrange范围内的极值点*  arrange = 230  x\_ls\_local\_max\_a\_slide\_ls, x\_ls\_local\_min\_a\_slide\_ls = range(63), range(63)  **while** **not**(len(x\_ls\_local\_max\_a\_slide\_ls) <= 62 **and** len(x\_ls\_local\_min\_a\_slide\_ls) <= 62):  x\_ls\_local\_max\_a\_slide\_ls, x\_ls\_local\_min\_a\_slide\_ls = [], []  **for** i **in** range(x\_start, x\_end):  **if** arrange + x\_start <= i < x\_end - arrange:  max\_flag, min\_flag = 1, 1  **for** j **in** range(-arrange, arrange):  **if** a\_slide\_ls[i] < a\_slide\_ls[i + j]:  max\_flag = 0  **break**  **for** j **in** range(-arrange, arrange):  **if** a\_slide\_ls[i] > a\_slide\_ls[i + j]:  min\_flag = 0  **break**  **if** max\_flag == 1:  x\_ls\_local\_max\_a\_slide\_ls.append(i)  i += arrange  **if** min\_flag == 1:  x\_ls\_local\_min\_a\_slide\_ls.append(i)  i += arrange  print('当前: ')  print('arrange = ', arrange)  print('极大值点列表长度: ', len(x\_ls\_local\_max\_a\_slide\_ls), '极小值点列表长度', len(x\_ls\_local\_min\_a\_slide\_ls))  arrange += 10  local\_max\_a\_slide\_arr = np.array(a\_slide\_ls)[x\_ls\_local\_max\_a\_slide\_ls]  local\_min\_a\_slide\_arr = np.array(a\_slide\_ls)[x\_ls\_local\_min\_a\_slide\_ls]  *# a-t函数*  plt.plot(x\_ls, a\_ls, linewidth='0.5')  *# 滑动处理后a-t函数*  plt.plot(x\_slide\_ls, a\_slide\_ls, linewidth='0.5')  *# 绿点表示极大值点*  plt.scatter(x\_ls\_local\_max\_a\_slide\_ls, local\_max\_a\_slide\_arr, c='green', s=10)  *# 紫点表示极小值点*  plt.scatter(x\_ls\_local\_min\_a\_slide\_ls, local\_min\_a\_slide\_arr, c='purple', s=10)  print('预测起始点列表:')  print('长度: ', len(x\_ls\_local\_max\_a\_slide\_ls))  print(x\_ls\_local\_max\_a\_slide\_ls)  print('预测终止点列表:')  print('长度: ', len(x\_ls\_local\_min\_a\_slide\_ls))  print(x\_ls\_local\_min\_a\_slide\_ls)  *# 将数据写入Excel*  result = open('predict\_of\_data' + file\_num + '.xls', 'w', encoding='gbk')  result.write('pre\_start**%.1f**/**%d**/**%d**' % (coefficient, slide\_step, arrange))  result.write('**\n**')  **for** i **in** range(0, len(x\_ls\_local\_max\_a\_slide\_ls)):  result.write(str(x\_ls\_local\_max\_a\_slide\_ls[i]))  result.write('**\n**')  result.write('**\n**')  result.write('pre\_end**%.1f**/**%d**/**%d**' % (coefficient, slide\_step, arrange))  result.write('**\n**')  **for** i **in** range(0, len(x\_ls\_local\_min\_a\_slide\_ls)):  result.write(str(x\_ls\_local\_min\_a\_slide\_ls[i]))  result.write('**\n**')  result.close()  plt.show() |
| **附录2** |
| 介绍：使用python绘制误差分析图的代码 |
| import **matplotlib.pyplot** **as** **plt**  x\_ls\_start\_predict = [1149, 1552, 2123, 2756, 3362, 3970, 4603, 5063, 5781, 6533, 7328, 7979, 8779, 9448, 10193, 10904, 11593, 12363, 12989, 13464, 14159, 15512, 16304, 17109, 17830, 18380, 19036, 19722, 20297, 20970, 21624, 22248, 22775, 23372, 23993, 24601, 25345, 26026, 26606, 27292, 27890, 28501, 29203, 30010, 30506, 31028, 31780, 32386, 33071, 33777, 34436, 35025, 35768, 36526, 37089, 37781, 38560, 39208, 39829, 40502, 41312, 41941]  x\_ls\_start = [1128, 1723, 2159, 2749, 3344, 3954, 4619, 5177, 5733, 6442, 7261, 7953, 8850, 9407, 10134, 10888, 11541, 12272, 12968, 13565, 14072, 15623, 16217, 17076, 17812, 18350, 18991, 19689, 20359, 21039, 21675, 22224, 22727, 23378, 23993, 24646, 25286, 25992, 26653, 27255, 27892, 28476, 29155, 30031, 30633, 31161, 31714, 32509, 33012, 33793, 34470, 35014, 35732, 36558, 37152, 37800, 38518, 39210, 39790, 40498, 41267, 41985]  x\_ls\_end = [1405, 1856, 2443, 3061, 3680, 4320, 4905, 5396, 6166, 6869, 7706, 8433, 9077, 9791, 10561, 11245, 11955, 12689, 13292, 13801, 14475, 15810, 16686, 17502, 18126, 18701, 19380, 20095, 20685, 21321, 21959, 22510, 23095, 23685, 24340, 25026, 25698, 26379, 26968, 27629, 28294, 28809, 29598, 30364, 30830, 31425, 32195, 32765, 33503, 34126, 34773, 35360, 36122, 36827, 37485, 38163, 38888, 39535, 40200, 40842, 41646, 42276]  x\_ls\_end\_predict = [1419, 1832, 2414, 3039, 3646, 4284, 4895, 5352, 6091, 6819, 7665, 8381, 9046, 9767, 10542, 11165, 11901, 12663, 13259, 13758, 15438, 15781, 16690, 17460, 18121, 18655, 19324, 20082, 20693, 21315, 21965, 22549, 23106, 23661, 24284, 24922, 25628, 26315, 26899, 27593, 28217, 28806, 29581, 30301, 30814, 31397, 32150, 32685, 33463, 34132, 34716, 35427, 36115, 36813, 37447, 38174, 38832, 39482, 40181, 40821, 41652, 42335]  print(len(x\_ls\_end))  print(len(x\_ls\_end\_predict))  sm = 0  abs\_dx\_start\_ls = []  abs\_dx\_end\_ls = []  **for** i **in** range(62):  *# print('start', i)*  *# print(abs(x\_ls\_start[i]- x\_ls\_start\_predict[i]))*  *# print("end", i)*  print(abs(x\_ls\_end[i] - x\_ls\_end\_predict[i]))  abs\_dx\_start\_ls.append(abs(x\_ls\_start[i]- x\_ls\_start\_predict[i]) / 200)  abs\_dx\_end\_ls.append(abs(x\_ls\_end[i] - x\_ls\_end\_predict[i]) / 200)  sm += abs(x\_ls\_start[i] + x\_ls\_end[i] - x\_ls\_start\_predict[i] - x\_ls\_end\_predict[i])  print(abs(-1))  print('每个动作点平均时间误差: ', sm / 124 \* 0.005, 's')  plt.rcParams['font.family'] = ['Microsoft YaHei']  *# plt.title('语义动作起始点与预测起始点对应的时间与动作序号之间的函数图像\n蓝线：实际语义动作起始点\n橙线：预测语义动作起始点')*  *# plt.ylabel('时间/0.005s')*  *# plt.xlabel('动作序号')*  *# plt.plot(x\_ls\_start, linewidth='0.5')*  *# plt.plot(x\_ls\_start\_predict, linewidth='0.5')*  plt.title('语义动作终止点与预测终止点对应的时间与动作序号之间的函数图像**\n**蓝线：实际语义动作终止点**\n**橙线：预测语义动作终止点')  plt.ylabel('时间/0.005s')  plt.xlabel('动作序号')  plt.plot(x\_ls\_end, linewidth='0.5')  plt.plot(x\_ls\_end\_predict, linewidth='0.5')  plt.show() |