文章编号：2095-2163(2023)03-0039-08 中图分类号：TP311.1 文献标志码：A

基于 Android 的轨迹分析应用设计与实现

肖雷鸣[1]，卿粼波[1]，冯 田[2]
（1 四川大学 电子信息学院，成都 610065；2 四川大学 建筑与环境学院，成都 610207）

摘 要：针对目前轨迹分析研究中轨迹数据采集困难、数据分析片面、分析程序复杂、实用性弱的现状，提出并设计了一款集轨迹采集、(多语义)轨迹分析、轨迹可视化、轨迹分享四种功能为一体的轨迹分析应用程序(App)。首先，通过百度地图软件开发工具包(SDK)获取单点 GPS 定位信息并设计实时显示模块，然后利用轨迹切片方法，结合速度与距离联合判断轨迹停留点计算出分段轨迹的多种语义信息(始末时间、距离、速度、出行方式等)。特别地，在切片分析的基础上，提出利用轨迹的出行语义信息对高速轨迹片段的出行方式进行二次判别，在对比测试中，使用语义分段的分析结果更符合实际情况。最后，绘制出轨迹分析结果，并设计分享功能。测试结果表明，软件各模块能稳定运行，软件采集的 GPS 定位信息能达到传统 GPS 采集设备的精准度，轨迹分析的结果与实际行程相符合。

关键词：轨迹分析；应用程序(App)；GPS；语义轨迹；城市规划

Design and implementation of trajectory analysis application based on Android

XIAO Leiming[1], QING Linbo[1], FENG Tian[2]
(1 College of Electronics and Information Engineering, Sichuan University, Chengdu 610065, China;
2 College of Architecture and Environment, Sichuan University, Chengdu 610207, China)

【Abstract】Aiming at the current situation of trajectory data collection difficulties, one-sided data analysis, complex analysis procedures, and weak practicability in trajectory analysis research, this paper proposes and designs a trajectory analysis application (App) that integrates four functions: trajectory collection, (multi-semantic) trajectory analysis, trajectory visualization, and trajectory sharing. First of all, single-point GPS positioning information is obtained through Baidu Map Software Development Kit (SDK) and a real-time display module is designed, then the trajectory slice method is used, combined with speed and distance to jointly judge the stop point of the trajectory, a variety of semantic information (start and end time, distance, speed, travel mode, etc.) of the segmented trajectory are calculated. In particular, on the basis of slice analysis, the travel semantic information of the trajectory is used to make a secondary discrimination of the travel mode of the high-speed trajectory segment. In the comparative test, the analysis results using semantic segmentation are more in line with the actual situation. Finally, the trajectory analysis results are drawn, and the sharing function is designed. The test results show that each module of the software can run stably, the GPS positioning information collected by the software can reach the accuracy of traditional GPS acquisition equipment, and the results of trajectory analysis are consistent with the actual itinerary.

【Key words】trajectory analysis; application (App); GPS; semantic trajectory; urban planning

0 引 言

在卫星定位技术与移动互联网软硬件技术的高速发展背景下，人们出行时产生的海量轨迹数据以各种方式被获取并保存下来。这些轨迹数据都记录了移动对象长时间的位置变化，其反映出的移动对象人群的移动与活动特征、兴趣爱好和社会习惯等丰富的时空特征信息，引起了城市规划、社会学等多个领域研究学者的关注。至今，研究者们已利用采集到的轨迹数据进行了大量的研究，挖掘出了轨迹数据在许多领域的应用价值[1-3]。例如，轨迹数据已用于通勤[4]与职住空间分析[5]、交通路线的优化与设计[6-7]、城市交通状态的划分与识别[8]、城市绿道系统效用评估[9]、识别城市功能区[10]、商业选

基金项目：国家自然科学基金(61871278)。

作者简介：肖雷鸣(1999-)，男，硕士研究生，主要研究方向：软件开发、计算机视觉；卿粼波(1982-)，男，博士，教授，博士生导师，主要研究方向：图像处理、图像/视频编码通信、机器视觉与智能系统；冯 田(1995-)，女，博士，主要研究方向：出行碳排放、低碳住区。

通讯作者：卿粼波 Email：qing_lb@scu.edu.cn

收稿日期：2022-11-03

址[11]、个性化推荐路线[12]、道路推荐[13]、交通热点分析[14]等方面。

大量对轨迹数据的研究,使轨迹数据的分析方法形成了一定的研究范式。研究者们通常将轨迹处理研究分为轨迹信息采集与轨迹分析两个步骤。GPS 嵌入式设备采集的轨迹数据[15]与信令数据[16]是研究者常用的数据来源,但由于前者造价昂贵,后者数据精度较低、获取渠道存在限制等原因,且随着本世纪以来 Android 操作系统的迅猛发展[17],借由地图平台营造的位置定位信息服务,致使研究者们更多地选择自主设计相关软件来持续获取定位信息[18]。在轨迹数据分析方面,有研究者使用基于时间序列的聚类算法识别轨迹的停留点[19]、高频点与异常点[20]以及打车热点[21]等轨迹语义点;也有研究者利用判别分析[22]、支持向量机[23]等方法[24]识别轨迹中不同的出行方式,以及利用隔离机制进行轨迹异常检测[25]、利用轨迹信息测算持有者的运动能量消耗[26]等方法对轨迹进行分析。

尽管研究者们对轨迹数据的采集与分析流程已较为熟悉,但纵观现有研究,仍普遍存在以下 3 个问题:

(1)轨迹采集与轨迹分析的割裂:采集端(GPS 嵌入式设备、Android 设备)与分析处理端(PC 服务器)的分离导致无法对轨迹进行实时分析,同时也提高了实验环境的搭建成本。

(2)轨迹分析算法与实际应用的割裂:研究多针对单一语义进行分析(停留点、出行方式等),研究成果无法整合,非专业人员复现难度大,投入应用难度更大。

(3)缺乏易用的实时可视化系统。

基于上述原因,本文开发了一款集轨迹采集、轨迹分析(多语义)、轨迹可视化、轨迹分享四种功能为一体的轨迹分析应用。该应用基于 Android 平台与百度地图 SDK 开发,能持续采集使用者的经纬度、速度、室内状态、POI 等轨迹信息;同时,通过嵌入在软件内部的相关算法(时间切片、停留点识别、语义分段等)对轨迹进行分段,计算出轨迹停留点、出行方式、起始时间、总时间、总距离等重要语义信息;此外,设计分享模块将分析结果以 Excel 表格形式分享,便于研究者二次分析。本文开发的轨迹分析软件,实现了对轨迹的实时采集、多种语义分析、可视化与保存分享,具有实时、集成、易用的特点,适用于各领域有轨迹采集与分析需求的相关人员。

1 系统设计

1.1 需求分析与模块设计

从目前轨迹研究中采集困难、分析片面、程序复杂、实用性弱四个角度进行需求分析。采集轨迹数据时,用户需要实时查看定位数据以确认数据的正确采集或者进行轨迹的调整;轨迹采集结束后,用户需要即时得到分析结果,包括轨迹的时间、距离、速度、出行方式等详细信息;用户需要借助可视化界面对出行轨迹进行判断;在得到分析数据后,用户希望保存或分享原始数据与分析数据,便于后续研究。根据上述需求设计了四大功能模块,如图 1 所示。由图 1 可知,对这 4 个模块的研发功能,拟展开阐释分述如下。

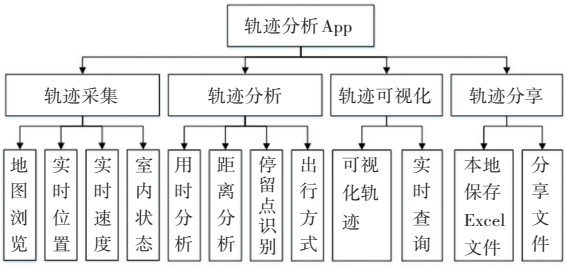


图 1 轨迹分析应用模块设计

Fig. 1 Design of trajectory analysis application module

(1)轨迹采集模块:在地图上能实时查看持有者的位置、速度、室内状态、附近 POI 等信息;同时保存采集到的单点定位信息。

(2)轨迹分析模块:对单点定位信息进行分析。通过轨迹切片、停留点识别、切片整合、语义分段等算法对长时轨迹进行分段,并获得每段轨迹的用时、距离、出行方式等详细信息。

(3)轨迹可视化模块:在地图上可视化长时轨迹。根据持有者不同的出行方式,以不同的颜色可视化轨迹片段,并添加始末点和停留点的点标记,点击标记能查看距离和用时等信息。

(4)轨迹分享模块:分享模块设计了保存与分享两大功能,能保存定位信息与分析结果至 Excel 表格,并支持一键分享至微信和 QQ。

1.2 软件服务流程

软件的服务流程如图 2 所示。用户打开 App 后,软件自动加载地图界面;点击开始定位即可进行轨迹定位(定位过程中用户能在地图上查看当前的定位信息,也可以退出 App 界面,系统则会自动在后台采集定位信息);点击结束定位,软件自动进行轨迹分析、可视化以及保存分析文件,最后,用户可

以选择是否分享文件。整个操作过程中，使用按键少，操作简便。

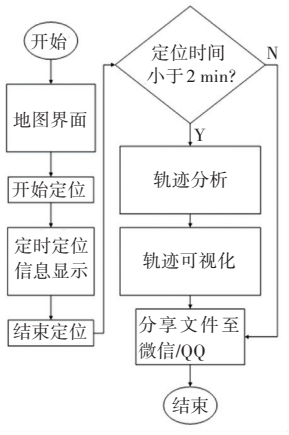


图 2　轨迹分析应用服务流程

Fig. 2　Service process of trajectory analysis application module

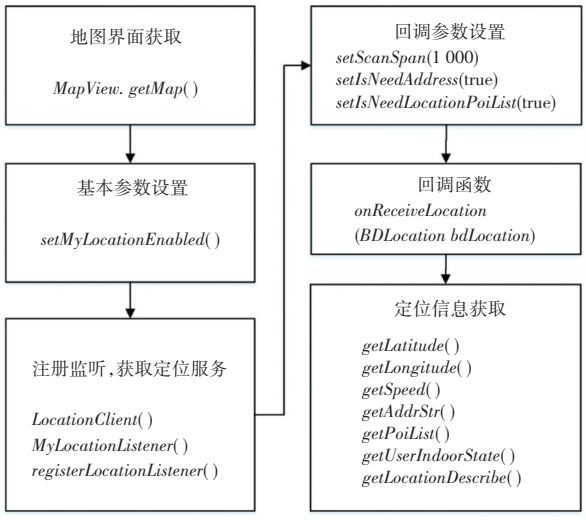


图 3　定位模块方法流程

Fig. 3　The method flow of the positioning module

2　关键模块实现

2.1　地图与定位模块

地图与定位模块是 App 的基础模块。此模块提供用户可视化与交互界面，用户能实时查看所处位置，获取地址、GPS、速度等详细信息。地图模块中添加监听按钮用于开启与结束定位，实现轨迹采集与轨迹分析的功能。

模块调用百度地图 SDK 中的 *MapView. getMap*( ) 方法获取基础的地图可视化界面，地图界面拥有基础的缩放查看功能；然后，通过基本参数的设置与监听注册，获取定位服务；获取定位服务对象后，通过 *setScanSpan*( ) 等方法设置定位时间间隔等回调参数；最后，在回调函数中使用相应的 *get* 方法获取所需的定位信息；图 3 为定位模块的方法流程图。此外，当前时刻回调的定位信息会储存至动态 *Stringbuffer* 变量中，使用 *TextView.setText*( ) 方法将储存的变量值以文本框的形式添加到地图界面，如图 4 所示。同时，自定义 *Point* 类来描述当前时刻定位点，自定义的静态 *List* < *Point* > 变量 *pointlist* 储存所有时刻的定位信息，用于后续的轨迹分析模块。

图 4　定位信息

Fig. 4　Positioning information

2.2　轨迹分析模块

轨迹分析模块是轨迹分析应用的核心模块，对定位模块储存的连续单点定位信息进行分析，得到轨迹的用时、距离、停留点、出行方式等信息。轨迹分析模块细分为轨迹切片、切片识别与整合、语义分段与输出规范化四个部分。各部分的功能划分如图 5 所示。

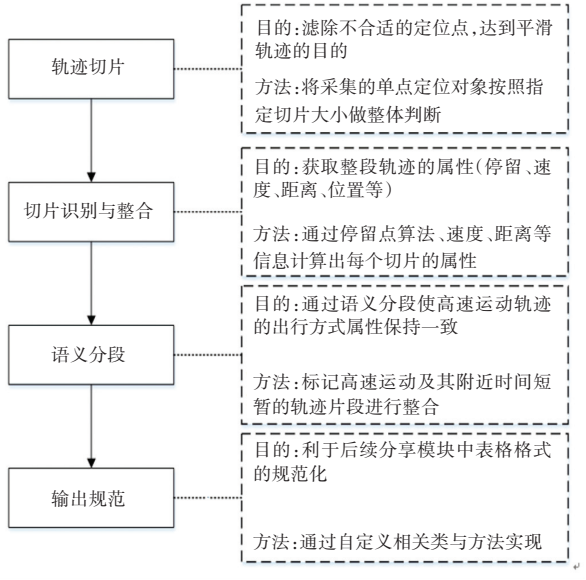


图 5　轨迹分析模块

Fig. 5　Trajectory analysis module

2.2.1　轨迹切片

即便定位模块单点定位信息的采集频率足够高，但对于时间跨度较大的轨迹，单点 GPS 定位信息很难有效地描述轨迹的状态。

问题 1 使用连续的单点定位描述长时轨迹，描述结果整体性差。例如，当一段长时间的步行中有多次短暂停留（几秒左右），这一段步行轨迹就会被描述为大量的步行轨迹片段与停留轨迹片段。

问题 2 使用采样间隔较大的单点定位来描述长时轨迹，相当于对连续的单点定位进行抽样，而使用单点定位信息来描述连续的轨迹会产生较大的误差。

针对上述存在的问题，为保证轨迹分析的整体性和低误差，本文提出使用轨迹切片的方法对长时轨迹进行分析，将整个轨迹片段按固定时间尺度进行切分，然后进行整体分析。切片大小（切片包含的单点定位个数）为 *clipsize*，由本小节问题 1 可知，*clipsize* 值不宜过小，此外 *clipsize* 值根据采集的轨迹时长灵活设置（本文系统测试时长为 30 min 内，测试中 *clipsize* 的值设置为 60，即轨迹分析的精度以 1 min为单位）。

轨迹切片如图 6 所示，长时轨迹被切片为固定长度的片段，一个轨迹切片中有多个定位点，切片和单个定位点具有独立的属性，其中单点定位的 *pointlogi*、*pointlati*、*pointspeed*、*pointindoor* 等属性由定位模块的 *getLatitude*( )、*getLongtitude*( )、*getSpeed*( )、*getUserIndoorState*( ) 等方法获取，切片的属性（*clipspeed*、*clipindoor*、*clipvehicle* 等）由单点定位属性计算得到。*keypoint* 为切片的第一个单点定位索引。

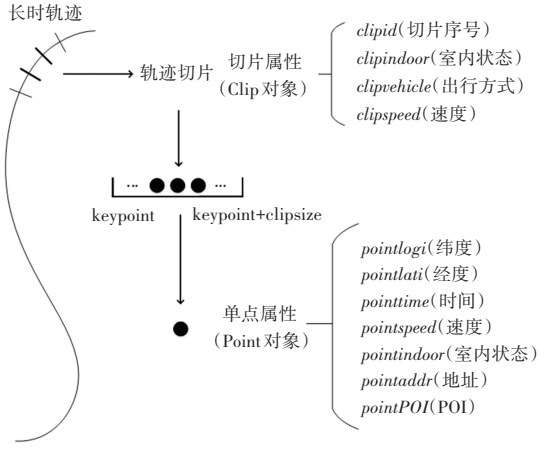


图 6 轨迹切片

Fig. 6 Trajectory slices

2.2.2 切片识别与整合

轨迹切片后，利用切片包含的单点定位的属性对切片的速度、室内状态、出行方式等属性进行识别。这里给出研究阐述如下。

（1）速度：*clipspeed* 由从 *pointlist* 中获取的 *pointspeed* 计算得到，即：

$$clipspeed = \sum_{i=0}^{clipsize-1} pointspeed_i / clipsize \quad (1)$$

（2）室内状态：当每个 clip 中 *clipindoor* 的数值大于 *clipsize*/2 时，*clipindoor* 设置为 1（1 表示室内，0 表示室外）。

（3）出行方式：停留与非停留两种状态。当切片识别为非停留状态时，其出行方式由 *clipspeed*（单位为 m/s）确定，在采集的社会实验数据[27~29]的范围内可将出行方式划分为步行（0< *clipspeed* ≤ 1.4）、自行车（1.4 < *clipspeed* ≤ 5）、汽车（5 < *clipspeed* ≤ 8.3），地铁（*clipspeed* > 8.3）四类。

停留状态的识别是出行方式识别的关键部分。停留点类型如图 7 所示。由图 7 可知，停留点分为静止型停留点和徘徊型停留点两类。在识别停留片段时，存在以下情况：

情况 1 使用速度判别停留点时，会漏判徘徊型停留点。

情况 2 使用距离阈值判别停留点，当运动速度较快时，存在切片内大量定位点速度为 0、但仍有定位点超出距离阈值，导致静止型停留点误判为运动轨迹片段的情况。

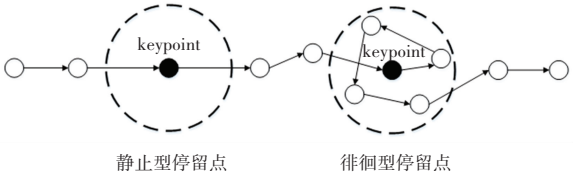


图 7 停留点类型

Fig. 7 Stop point types

为同时保证 2 类停留点的准确识别，本文提出使用速度和距离联合判断停留点，判别算法如下：

算法 1 联合速度与距离判断的停留点识别

```
for ( int point = keypoint;point - keypoint < clipsize;point ++) {
    if ( (pointlist.get(point).speed) == 0)
        i ++; //统计速度为零的定位点
    if ( GetDistance(pointlist.get(point).pointlist.get(keypoint)) < 30)
        j ++;// 统计处于距离阈值内的定位点
}
if ( i > = clipsize/2 || j == clipsize) {// 联合判断
    clipvehicle = "停留"
    clipspeed = 0.0;
}
```

程序中，*point* 为单点定位在 *pointlist* 中的索

引，i 为速度为 0 的 *point* 个数，j 为小于距离阈值的 *point* 个数，距离阈值设置为 30 m，代表停留时所允许的徘徊范围，可根据实际需求设置。

切片识别后，为进一步分析与输出，将判别属性相同的切片进行整合。切片整合示意如图 8 所示。

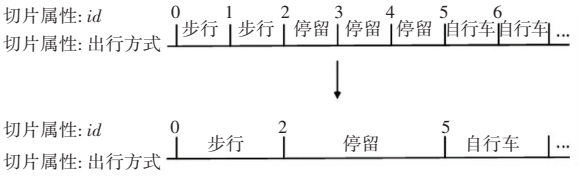


图 8　切片整合

Fig. 8　Slice integration

2.2.3　语义分段

在实际测试中，行驶速度较高的交通工具在行进过程中，速度不会稳定地保持在某段大小范围内，通常会因为不同的路况导致暂时的低速行驶，从而导致出现机动车运动轨迹被误判为步行、自行车的情况。

针对上述误差，提出使用已判别后的轨迹片段的语义进行二次分析的方法。语义分段算法的部分核心代码如下：

算法 2　语义分段算法

```
for(int i = 0;i < cliplist.size() - 1;i ++){
  if (cliplist.get(i + 1).id - cliplist.get(i).id < 2)
   //连续值小于 clipsize 的 2 倍则视为短时轨迹
   cliplist.get(i).semantic = "timetooshort";
  }
for(int i = 0;i < cliplist.size() - 1;i ++){
  if (cliplist.get(i).semantic == "timetooshort")
      if (cliplist.get(i + 1).semantic == "timetooshort")
       c ++;…;// 变量 c 为短时轨迹连续出现的次数
       else{
       //加权平均前 c 段短时轨迹的速度
       clip.speed = getavespeed(i - c,i,cliplist);
       //二次判别出行方式
       clip.vehicle = getavevehicle(clip.speed);
       //重新统计连续短时轨迹数量
       c = 0;…;
}}
```

首先，标记出高速运动片段及其邻近的短时轨迹片段的语义属性（semantic）为短时（timtooshort），然后使用 *getavespeed*( ) 方法对连续的标记轨迹做均值处理，使用 *getavevehicle*( ) 方法重新判断速度和出行方式属性，同时考虑到速度为零的片段对速度均值的影响较大，停留片段将不被标记；语义分段的示意图如图 9 所示。

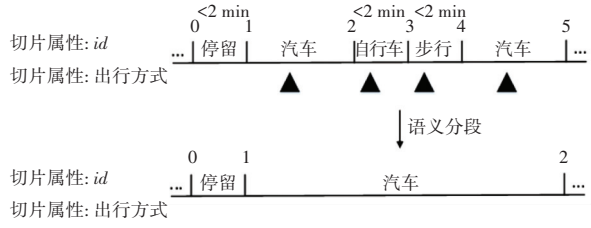


图 9　语义分段

Fig. 9　Semantic segmentation

2.2.4　输出规范化

通过自定义 *Output* 类将每段轨迹的停留、出行方式、速度、室内外状态等属性信息存入实例中，并通过自定义的 *getallDistance*( ) 函数与 *getspendTime*( ) 函数计算各段轨迹的始末时间与总距离，通过 *Output* 实例的各类属性规范描述每段轨迹的信息，为后续的轨迹可视化及轨迹文件生成与分享提供易用的输入数据。

2.3　轨迹可视化模块

轨迹可视化模块的功能为将轨迹分析模块得到的结果呈现在地图上。在绘制地图上不同交通方式的轨迹以不同颜色绘制，且添加了停留点点击窗口用于呈现相关信息。点击后的可视化结果如图 10 所示。可视化使用百度地图 SDK 组件，可视化之前将会获取轨迹段的始末索引信息，使用 *mBaiduMap.addOverlay*( ) 方法将非停留轨迹片段所包含的 *point* 绘制在地图上，实例化 *PolylineOptions*( ) 对象设置绘制的粗细与颜色；对于停留的轨迹段，将其对应的 *point* 存储至一个数组中，同时通过 *mBaiduMap.showInfoWindows*( ) 方法批量绘制点标记，创建 *InfoWindow.OnInfoWindowClickListener* 监听对象，重写 *onInfoWindowClick*( ) 方法来设置点击后弹出的信息窗口，显示的内容为 *Output* 类的对象。

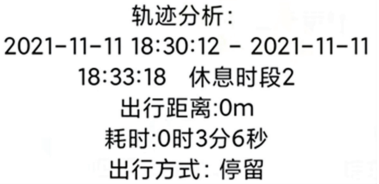


图 10　轨迹可视化

Fig. 10　Trajectory visualization

2.4　保存与分享模块

保存与分享模块的功能主要由自定义 FileUtil 工具类中的 *writeToExcel*( ) 方法与 ShareUtils 工具

类中的 *ShareWechatFriend*( ) 方法实现。在 App 文件目录下的 files 目录中创建.xls 文件。第一个 sheet 表单,命名为“GPS 信息”,用于写入 pointlist 列表中存储的单点定位信息,第一列的表头信息与 Point 类的属性对应,保存的文件格式如图 11 所示;第二个 sheet 表单命名为“分析结果”,用于写入 *outputlist* 中保存的分析结果,第一列的表头信息与 *Output* 类的属性对应,保存的文件格式如图 12 所示。

纬度	经度	时间	速度	室内状态	地址	附近POI
104.087669	30.657655	2021-12-05 20:22:25	0.0	0	在平安金融中心附近	poiName:平安金融中心, poiTag:房地产

图 11 单点定位信息保存格式

Fig. 11 Single-point positioning information save format

时间	状态	距离	耗时	出行方式
2021-12-05 20:24:29 - 2021-12-05 20:34:46	出行时段1	847.0	0时10分17秒	步行

图 12 轨迹分析结果保存格式

Fig. 12 Trajectory analysis result save format

以微信分享为例,在自定义的 *ShareWechatFriend*( ) 方法中,使用 *packageManager.getInstalledPackages*( ) 方法获取 *PackageInfo* 参数,根据 *PackageInfo* 的值判定是否安装微信客户端,然后使用 *FileProvider.getUriForFile*( ) 方法使保存在本地的.*xls* 文件能提供给外部应用,最后使用 *setPackage*(*PACKAGE_WECHAT*) 与 *setAction*(*Intent.ACTION_SEND*) 方法调用微信,加载出分享页面,并分享创建的 xls 文件。分享页面如图 13 所示。

图 13 分享页面

Fig. 13 Share page

3 系统测试

3.1 App 开发环境

本文开发的轨迹分析应用是一款在 Windows7 操作系统环境下,使用 Android Studio 3.1.23 集成开发环境进行开发的软件,其中 Android SDK 开发工具包版本为 API 29,适配 Android 10.0(Q)平台。软件的基本功能与核心算法全部使用 Java 语言编程实现,软件的可视化界面设计主要使用 XML 语言与 Java 语言完成设计。开发中,Java 语言使用的开发工具包版本为 jdk 1.8.0_301,保存与输出的数据使用 xls 文件格式存储,使用 Redmi Note 7 手机进行调试与测试,软件输出的所有结果都由测试机实测产生。Release 版本的 APK 文件大小在 15 Mb 左右。

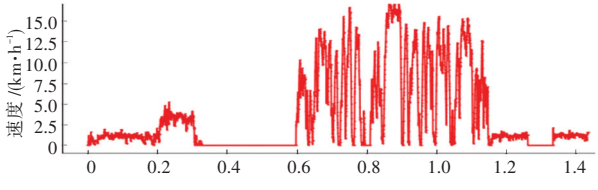
3.2 GPS 定位精准度测试

本文软件使用百度定位 SDK 采集单点定位信息,虽然定位信息已用相应的格式保存至 Excel 表格,但是无法直观地判断采集的定位信息是否合理,因此设计了对比实验,对软件 GPS 定位的精准度进行测试,测试方法如下。

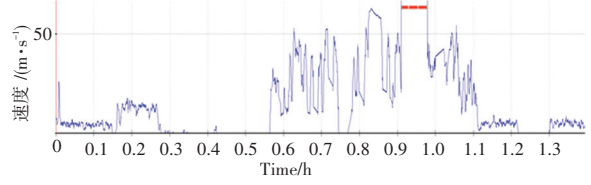
使用谷歌 GPS 定位手表采集的定位数据与轨迹分析 App 采集的定位数据进行多次比对,并可视化两者的轨迹路线图。GPS 手表的可视化使用谷歌专用的“GPX”软件完成,轨迹处理 App 定位数据的可视化由自主设计的可视化模块完成;绘制二者的速度曲线进行对比,如图 14(a)、14(b)所示。

从图 14 中的速度曲线图可看出,两者采集的单点 GPS 数据准确度基本一致,证明 App 采集的单点定位数据的可靠性。

此外,由图 14(a)可知,0.6~1.2 h 的高速时段是一段客车的运动轨迹,但由于轨迹速度曲线并不稳定,容易出现类别误判为停留点或者其他交通工具;图 14(b)中的红色虚线部分,GPS 手表甚至出现速度异常片段。上述现象表明,利用单点定位信息作为轨迹分段的依据是存在较大误差的,这也是本文对轨迹进行切片与语义分段的一个重要原因。



(a) App 记录的速度曲线图(python 绘制)



(b)GPS 手表记录的速度曲线图(GPXSee 软件绘制)

图 14 精准度对比

Fig. 14 Comparison of accuracy

3.3 轨迹分析测试

根据出行的距离,轨迹分析测试分为短程测试与长程测试。

3.3.1　短程测试

短程测试步骤如下：打开 App→点击开始定位开始采集定位信息→查看可视化轨迹→点击结束定位→保存与分享文件。

本例短程测试的轨迹路径为：教学楼停留-步行至食堂-食堂用餐-步行寻找共享单车-等待共享单车解锁-骑车返回教学楼门口-步行进入教学楼。测试流程如图 15 所示。轨迹采集过程中，软件可退至后台运行，整个操作过程只需点击 3 次按钮。

（a）分享界面

时间	状态	距离	耗时	出行方式
2021-12-08 18:12:51 - 2021-12-08 18:19:05	休息时段1	0	0时6分14秒	停留
2021-12-08 18:19:05 - 2021-12-08 18:26:29	出行时段1	510.0	0时7分24秒	步行
2021-12-08 18:26:29 - 2021-12-08 18:46:23	休息时段2	0	0时19分54秒	停留
2021-12-08 18:46:23 - 2021-12-08 18:47:29	出行时段2	44.0	0时1分6秒	步行
2021-12-08 18:47:29 - 2021-12-08 18:48:31	休息时段3	0	0时1分2秒	停留
2021-12-08 18:48:31 - 2021-12-08 18:51:33	出行时段3	446.0	0时3分2秒	自行车
2021-12-08 18:51:33 - 2021-12-08 18:52:32	出行时段3	72.0	0时0分59秒	步行

（b）轨迹分析结果文件

图 15　短程测试

Fig. 15　Short distance test

3.3.2　长程测试

由于测试人员在测试过程中不可避免会乘坐高速移动的交通工具，而此类交通工具的速度往往是不稳定的，因此增加长程测试来验证语义分段算法处理此类轨迹的有效性。测试结果是一段乘坐机动车的出行轨迹，由于存在红绿灯、堵车等情况，机动车的速度不够稳定。将未使用语义分段的分析结果（图 16（a））与使用语义分段的分析结果（图 16（b））全部保存至本地进行对比，可以看出，使用语义分段算法后，将交通工具识别为自行车的轨迹片段归化为了汽车。最终的分析结果与实际相符，验证了语义分段算法的有效性。

时间	状态	距离	耗时	出行方式
2021-12-13 18:53:27 - 2021-12-13 18:56:27	出行时段1	1705.0	0时3分0秒	汽车
2021-12-13 18:56:27 - 2021-12-13 18:57:27	出行时段1	159.0	0时1分0秒	自行车
2021-12-13 18:57:27 - 2021-12-13 18:58:27	休息时段1	0	0时1分0秒	停留
2021-12-13 18:58:27 - 2021-12-13 19:02:28	出行时段2	1412.0	0时4分1秒	汽车
2021-12-13 19:02:28 - 2021-12-13 19:03:28	出行时段2	212.0	0时1分0秒	自行车
2021-12-13 19:03:28 - 2021-12-13 19:05:27	出行时段2	527.0	0时1分59秒	汽车

（a）轨迹分析结果文件（语义分段前）

时间	状态	距离	耗时	出行方式
2021-12-13 18:53:27 - 2021-12-13 18:57:27	出行时段1	1864.0	0时4分0秒	汽车
2021-12-13 18:57:27 - 2021-12-13 18:58:27	休息时段1	0	0时1分0秒	停留
2021-12-13 18:58:27 - 2021-12-13 19:05:27	出行时段2	2151.0	0时7分0秒	汽车

（b）轨迹分析结果文件（语义分段后）

图 16　长程测试

Fig. 16　Long distance test

4　结束语

本文基于 Android 平台与百度地图 SDK，开发了一款集轨迹采集、轨迹分析、轨迹可视化、轨迹分享四种功能为一体的轨迹综合处理系统，能实现对轨迹进行实时采集并分析轨迹的多种语义信息，还可支持可视化与轨迹分享；App 可用于辅助解决通勤分析、职住协调、道路优化等问题，但由于研究时间的局限性，App 中仍有一些不足之处，其功能也有待进一步开发。今后可优化或添加如下功能：

（1）实时绘制轨迹功能。在轨迹分析 App 中，可视化模块是在用户停止定位后可视化整段轨迹的，不够直观。今后可将可视化模块升级为实时绘制模块，这样在采集定位信息的过程中就开始绘制轨迹，能使用户更直观地了解到当前的轨迹动向，带来更好的交互性。

（2）语义分段优化。语义分段还不够精准。在遇到复杂的轨迹路线或者苛刻的定位环境（地铁站内）会导致语义分段生成的结果不够精准。今后可考虑将附近 POI 类型或者车道信息纳入语义分段的判断条件中，以进一步提高轨迹分析的准确度。

参考文献

[1] 高强，张凤荔，王瑞锦，等. 轨迹大数据：数据处理关键技术研究综述[J]. 软件学报，2017，28(04)：959-992.

[2] 许佳捷，郑凯，池明旻，等. 轨迹大数据：数据、应用与技术现状[J]. 通信学报，2015，36(12)：97-105.

[3] 毛嘉莉，金澈清，章志刚，等. 轨迹大数据异常检测：研究进展及系统框架[J]. 软件学报，2017，28(01)：17-34.

[4] 王艳涛，魏海平，何源浩，等. 基于位置轨迹挖掘的城市居民职住地识别方法研究[J]. 测绘与空间地理信息，2017，40(02)：113-116.

[5] 毛峰. 基于多源轨迹数据挖掘的居民通勤行为与城市职住空间特征研究[D]. 上海：华东师范大学，2015：115-130.

（下转第 50 页）