基于 WiFi 信号的室内定位

——智能控制科学创新实践 II 期末报告

12212020 万力行

摘要

本项目旨在开发基于 WiFi 信号接收强度（RSSI）的室内定位系统，采用机器学习算法实现定位，并通过 Web 界面展示实时和历史轨迹。系统包括数据采集、算法设计及 Web 程序开发三个主要模块。数据采集通过 Android 应用实现，用户可设置采样参数并导出 WiFi 数据以构建指纹库。定位功能包含 KNN 与 MLP 算法，利用 Flask 服务器处理实时数据并返回定位结果。前端基于 Leaflet.js 构建，动态显示位置及历史轨迹，增强用户交互体验。在实验中，尽管 WiFi 信号波动和相邻点数据重叠导致准确性受限，Web 端仍可显示接近真实运动轨迹的定位轨迹。本项目验证了 WiFi 定位的可行性，并指出算法优化及用户界面智能化的改进方向，为智能空间构建提供了实用参考。

目录

1 引言

随着移动设备和无线网络的普及，室内定位技术逐渐成为信息技术领域的研究热点。相比于 GPS 信号在室内环境中的失效，WiFi 定位因其广泛部署、成本低廉和易于接入的特性受到广泛关注。通过测量无线接入点与用户设备之间的接收信号强度（RSSI），可以推算设备的位置，从而实现室内定位。随着人工智能与机器学习等算法的进步，WiFi 定位技术逐渐朝着精度优化、稳定性

提升等方向发展。未来，随着硬件成本的降低和算法优化，这一技术将更加深入人们的生活和生产，成为智慧空间的重要组成部分。

本项目以南方科技大学第三教学楼为主要实验场地，开发一个完整的基于 WiFi 接收信号强度的室内定位系统，其能根据用户需求自动采集 RSSI 数据，依托机器学习算法实现室内定位并满足精度要求，通过用户友好的 Web 程序实现实时定位和历史轨迹展示。

2 问题描述

本项目需实现一个基于 wifi 信号强度的室内定位系统，主要任务包括数据采集与预处理、定位算法设计以及 web 程序实现。

1. 数据采集与处理：实现自动采集 RSSI 数据的系统，开发数据处理模块，实现用户友好界面以输入采样点信息并作标签。
2. 算法设计与实践：选择合适的室内定位算法处理 RSSI 数据，并计算出设备的实时位置。
3. Web 程序开发：实现一个能在地图上实时显示设备位置、查看目标历史定位记录并显示目标运行轨迹的 web 程序。

3 方法和思路

3.1 数据采集

本项目开发了一个 Android 程序，旨在借助移动智能手机自身的传感器等功能完成对 WiFi 信号的采集与存储。

用户进入数据采集界面，录入当前位置坐标信息，使得每次采集到的 RSSI 数据能与具体的位置标签相关联。地图上预先标记了 72 个定位点，对应 72 个整数位置标签。程序还提供设置采集参数的功能，允许用户设定最大采集次数和扫描时间间隔，以满足不同的采样需求。用户点击“开始采集”按钮后，应用程序会根据设定的参数开始执行数据采集操作。

实现采集功能的第一步是初始化 WiFi 管理器。在代码中，通过获取 `WifiManager` 实例，使程序能够访问并管理设备的 WiFi 服务。为了确保数据采集的顺利进行，程序会先检查设备是否具有权限，如果未开启，程序会提示用户开启 WiFi 相关权限。WiFi 扫描完成后，`ScanResult` 列表会包含当前环境下校园网所有可用 WiFi 的相关信息，包括 BSSID、SSID 和信号强度（RSSI）。这些信息与用户输入的坐标绑定并存储，以便于后续定位算法使用。

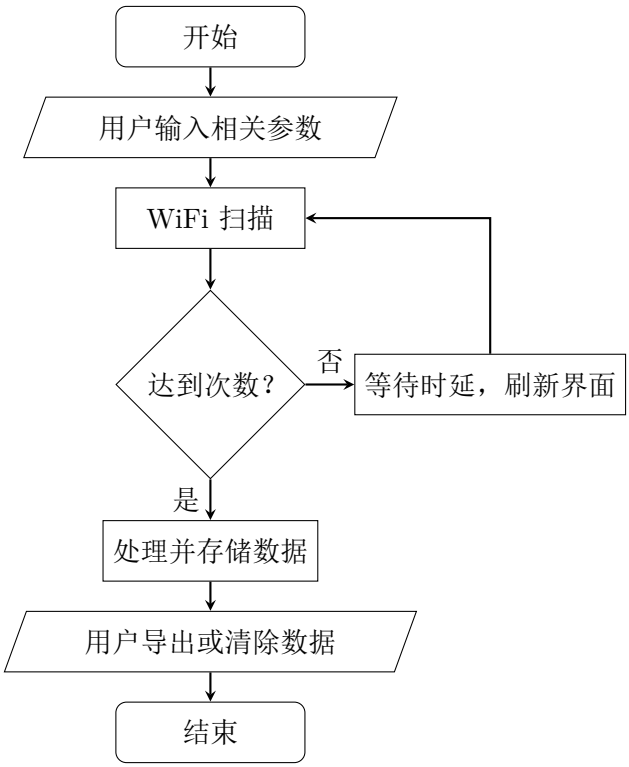
在数据采集的控制上，程序实现了循环扫描逻辑。根据用户设定的最大扫描次数，程序会在每次扫描结束后间隔一段时间再次启动扫描，直到完成所有预定的采集次数。为了让用户实时了解采集进展，界面上的状态文本会在每次扫描后更新，显示当前扫描次数和数据收集状态。

除了数据采集功能外，程序还提供数据导出和清理功能。清除数据按钮用于重置程序内存中的数据，用户可以在新采集之前清理掉之前的数据记录。通过界面上的导出按钮，用户可以将采集到的数据保存到设备的内部存储中，此阶段导出 `wifi_data.txt` 文件，用于搭建指纹库。导出文件的数据格式示例：

```
当前位置坐标: 1
BSSID: 30:c5:0f:a2:6d:80, SSID:SUSTech-wifi-5G
-64
```

```
-64
-63

当前位置坐标: 2
BSSID: 30:c5:0f:a2:78:e0, SSID:SUSTech-wifi-5G
-84
-84
-84
```



`wifi_data.txt` 文件中包含了位置坐标及相应的 WiFi 信息，为实现后续定位功能，需要利用该文件搭建指纹库。本项目开发了一个 Python 脚本，用于将 txt 文件转换为 csv 文件，最终呈现形式如下：

- 表头：包含 Location 和所有检测到的 BSSID。
- 每行数据：对于每个位置计算与之对应的每个 BSSID 的平均 RSSI 值。如果位置没有该 BSSID 的数据，则填充空白。每行数据按照 `location` 值的大小进行排序；对于相同位置在不同时间段采集的数据，处理后的数据会放在不同行。

3.2 定位功能实现

3.2.1 实时定位

为实现定位功能，需确立一个机器学习模型，系统将实时采集的 WiFi 信号传入模型得到定位结果。此处有两种解决方案，其一，在电脑端实现机器学习算法，训练模型并集成到 Android 程序中去；其二，在电脑端搭建服务器，Android 端将采集到的 WiFi 信息实时上传到服务器，经由服

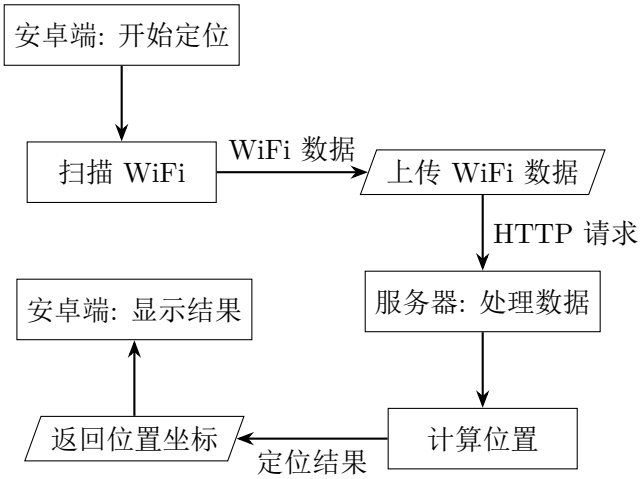
location	30:c5:0f:a2:7b:61	30:c5:0f:a2:6d:80	34:58:40:a4:44:c1
1	-81	-62	-70
2	-72	-66	
2	-71	-66	
3	-66		-64

表 1: CSV 文件数据格式示例

务器的机器学习算法得到定位结果，并返还给 Android 端。为降低 Android 端的计算压力，本项目采用第二种方案。

在安卓端，进入定位模式后，系统将允许用户输入服务器 IP 地址。开启定位后，系统将周期性扫描 WiFi 信号，提取每个信号的 SSID、BSSID 和 RSSI 值，并将这些数据整理为字符串后通过 HTTP POST 请求发送到服务器。服务器返回的位置结果会通过 UI 更新展示给用户。

在服务器端，使用 Flask 框架搭建了一个 RESTful 接口，用于接收安卓端上传的 WiFi 数据。接收到的数据被存储到本地 csv 文件，并通过相关机器学习算法计算用户当前位置编号。



3.2.2 算法设计

本项目采用 KNN 算法与 MLP 算法。`wifi_data.csv` 作为训练集（MLP 中另划分 20% 的数据作为验证集），安卓端实时上传的 WiFi 信息作为测试集。训练和测试数据中的列名（BSSID）可能不完全一致，为了保证模型输入的一致性，需要将所有 BSSID 进行对齐。其次，WiFi 数据中通常包含缺失值，例如某些位置无法接收到某些 WiFi 信号，此时将缺失值替换为默认值-100。

KNN 对于每个测试样本，计算其与训练样本之间的距离，并找到最近的 k 个邻居。随后，模型通过这 k 个邻居的标签进行投票，将得票最多的类别作为最终预测结果。

MLP 使用 pytorch 框架，使用 ReLU 作为激活函数，并采用 Adam 优化器。在训练阶段，使用 MBGD，每个隐藏层均使用了 Dropout，提高模型泛化能力。

3.2.3 Web 程序开发

服务器端接收安卓端上传的 WiFi 信息，通过机器学习算法获得一个整数类型的位置编号，然后在预定义的 CSV 映射表中找到对应的二维坐标 (x, y)，将整数值转化为平面地图上的实际坐标（像素坐标），并以 JSON 格式发送到前端。

前端 HTML 文件通过 Leaflet.js 加载地图并动态显示定位结果。前端每隔 5s 获取并解析服务器返还的数据，提取 x 和 y 坐标。如果当前已有坐标标记，则更新标记位置，否则创建一个新的红色标记。为了记录历史坐标，前端会将每次获取的坐标保存为一个灰色标记，并存储在历史记录数组中。HTML 页面将提供两个按钮，**查看历史记录**和**清除历史记录**。点击**查看历史记录**后，页面会显示所有历史点，用灰色标记表示，此时按钮文字改变为**不查看历史记录**，再次点击后，历史点从地图中移除，回到初始显示状态。点击**清除历史记录**按钮将会移除所有历史点的标记并清空记录。

4 实验结果与分析

4.1 Android 程序实现

Android 程序承担了数据采集、发送定位指令等功能。如图 4.1，本项目为 Android 程序设计了三个页面。在主界面中可选择**数据采集**与**开始定位**，以便用户进入不同的功能模块。

点击**数据采集**按钮后，会进入数据采集功能界面，用户输入当前位置坐标和其它相关参数可进行 WiFi 信号的采集。当采集到的样本数达到用户需求后，可点击**导出文件**将数据导出为 txt 文件。**清除数据**按钮允许用户清除已采集的数据，以便展开新一轮的数据采集。依托此功能可以建立 WiFi 指纹库。

点击**开始定位**按钮后，程序进入定位功能界面，用户输入服务器 IP 地址并点击**开始定位**，系统将每隔 2 秒扫描一次 WiFi 信息并上传至服务器，服务器通过机器学习算法返还定位结果，并在 Android 程序中显示。

(a) 主界面　　(b) 数据采集功能界面　　(c) 定位功能界面

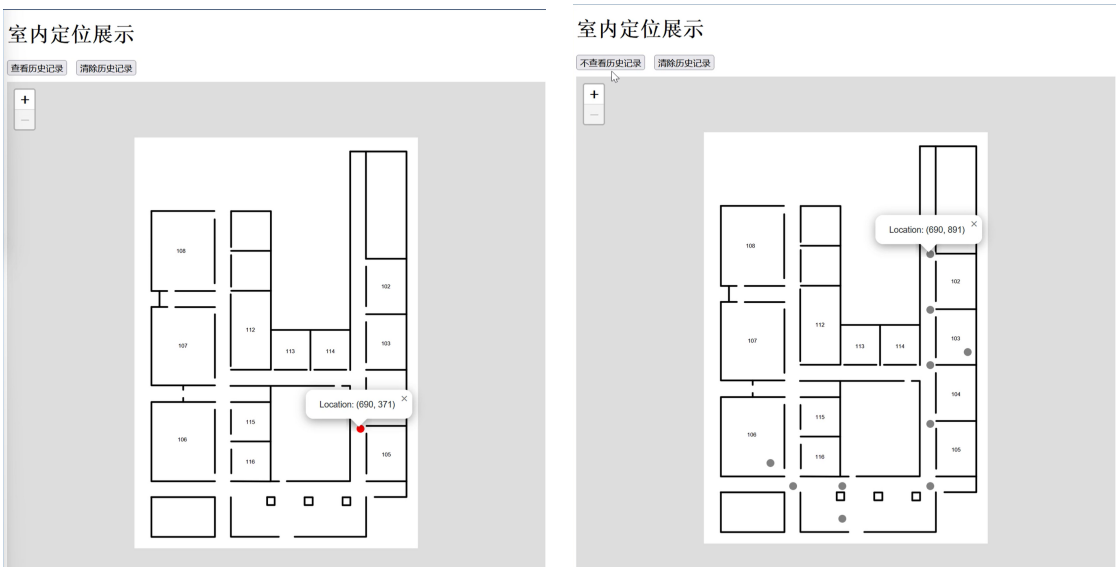
图 1: Android 程序界面

4.2 定位算法与前端实现

本项目采用 KNN 算法与 MLP 算法。经尝试，KNN 算法中 K 取 3，MLP 算法中采用三层全连接层，网络架构为 [Dimension of Input Features, 128, 64, 72]，学习率为 0.00001，epoch=2000, batch size=32 时训练效果较好。

实验发现，若严格以 Accuracy 评价模型的预测效果，则 KNN 与 MLP 的表现均较差，其中 KNN 的 Accuracy 为 0.6+，MLP 的 Accuracy 为 0.7+。究其原因，在采集的 WiFi 数据中，地图中部分相邻标记点的 WiFi 数据相似度极高甚至完全相同，又由于 WiFi 信号强度存在波动，导致部分相邻标记点难以区分。若预测值为距离真实值最近的两到三个点时，均视为预测正确，则 KNN 的 Accuracy 为 0.8+，MLP 的 Accuracy 为 0.9+。

如图4.2，前端每隔 5 秒从服务器接收当前位置坐标并刷新页面，坐标位置由红色圆点表示。当点击**查看历史记录**按钮后，页面将显示历史坐标，由灰色圆点表示。再点击该按钮会切换回原显示模式。实验发现，机器学习算法出现的误差在查看历史记录时被淡化，页面中仍可以显示一条历史轨迹，接近设备实际运动轨迹。



(a) 实时定位结果　　(b) 历史记录

图 2: Web 端定位效果展示

5 遗留问题与解决方向

1. 部分标记点 RSSI 特征重合度高，使得本项目使用的算法难以区分。可考虑引用到达时间（ToA, Time of Arrival）等更多类型的特征确定位置。此外，对于信号波动的情况，可记录信号随时间变化的特征，例如平均值、标准差或信号变化率，增加特征的维度。对信号强度进行滤波处理（如移动平均、卡尔曼滤波），减少波动影响。

2. Android 端和 Web 端的用户交互仍不够智能。在 Android 端，应允许用户选用需要执行的机器学习算法，以获取定位结果。在 Web 端，查看历史记录时，所有历史标记点均为灰色，没有时间特征。应记录每隔点出现的先后时间，并将其呈现在页面上，使历史轨迹的展示效果更为全面。

6 结语

本项目涉及 Android 程序开发，服务器与前端开发和机器学习算法设计，通过本项目的试炼，本人的编程经验得到丰富，并将课程中的机器学习算法应用到了实际工程项目中去，独立且较完整的实现了一个基于 WiFi 信号的室内定位系统。同时，本项目设计的室内定位系统仍存在许多瑕疵，如未能尝试更多定位算法、定位结果存在误差、用户交互不够智能等，仍需在后续的学习和研究中得以完善。