室内行人移动方位推算技术实验报告

1. 小组成员信息

成员姓名	成员学号	成员分工
史浩宇	201300052	算法选择、姿态矫正、数据预处理、绘图、收集数据
史浩男	201300086	算法选择、步幅预测、参数寻优、绘图、收集数据、数据共享
鲁权锋	201830168	算法选择、步幅预测、收集数据、测试评估

2. 数据收集

为了便于后续更好地提高模型的泛化性能，充分训练我们的模型，我们收集了各种状态下、不同区域下的数据，尽可能保证了数据集的多样性。为了保证数据的真实、准确和有效，我们的每一份数据都是在户外独立收集，且时长均大于 3 分钟，频率设为 50Hz。此外，考虑到压力传感器的数据仅用于计算高度，并且有些手机是无法收集压力传感器的，我们在数据收集时对压力传感器没有要求。

2.1 数据量

我们创建了一个 Github 仓库，发起了一个数据共享计划：2022 秋高级机器学习第一次作业 PDR-小组数据共享计划，现在已经有 17 名同学加入了我们发起的仓库。我们小组三人共收集了 50 条数据，此外我们与其他小组合作共收集了 84 条数据，所有数据均上传至 github 上共享。

README.md

Dataset-of-Pedestrian-Dead-Reckoning

2022秋高级机器学习第一次作业

> PDR-小组数据共享计划

- 加入共享计划的小组可以使用其他小组收集的数据
- 请及时维护自己上传的数据集文件

成员id

为防止数据集名称重复，为每个上传数据集的成员分配唯一id

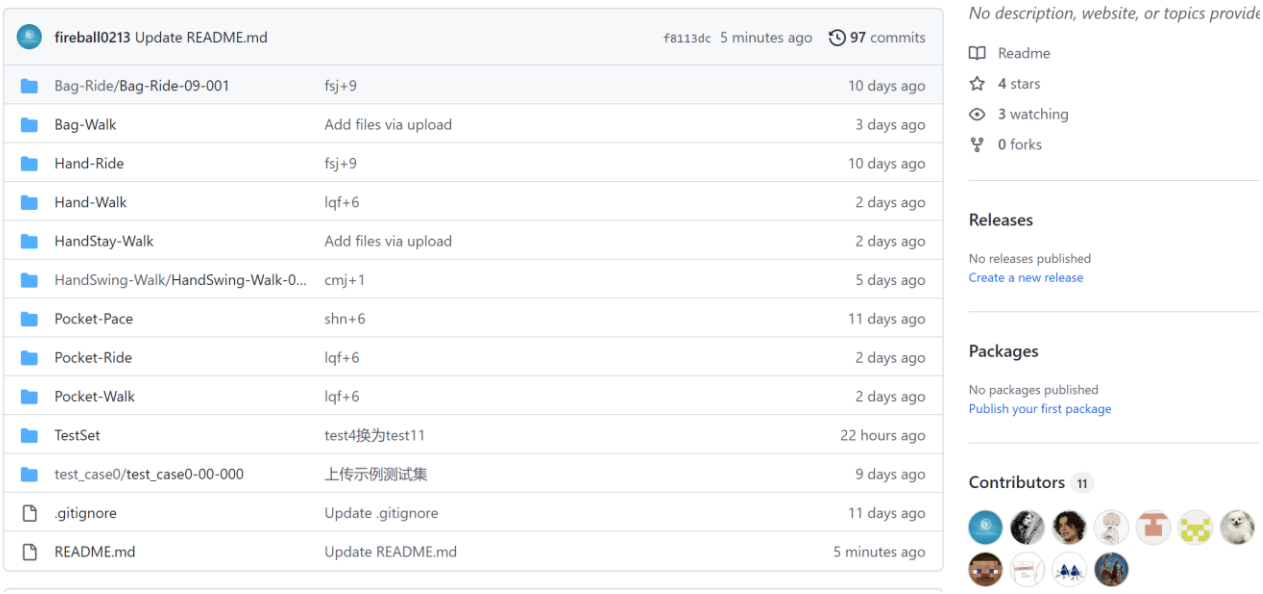
成员姓名	id	成员姓名	id	成员姓名	id
xxx	00	lth	06	cyz	12
shn	01	cmj	07	qk	13
shy	02	zyl	08	wzc	14
lqf	03	fsj	09	mcz	15
rwq	04	myw	10	wyd	16
xyh	05	yr	11	hyh	17

2.2 数据状态及来源

以上收集到的数据包括了如下 9 种状态的数据：

- 背包里-走路
- 背包里-骑车
- 手持-骑车
- 手持-走路
- 手持平稳-走路
- 手持摆臂-走路
- 口袋里-走走停停
- 口袋里-骑车
- 口袋里-走路

以上数据包含仙林和鼓楼两个区域，其中仙林 54 条，鼓楼 30 条。



2.3 数据集划分

我们先按照我们实验的需求，抽取 16 条数据用于我们的消融实验。对于剩下的数据，我们对各种状态、不同区域、不同设备收集的数据按 3:1 的比例分层随机抽样，其中 75%的数据作为训练集，另外 25%作为测试集。此外，我们还将助教提供的 test_case0 加入我们的测试集当中，因此最后我们共有 52 条数据作为训练集，17 条数据作为总体测试集，16 条数据作为消融实验的测试集。

3. 尝试方法与具体实现

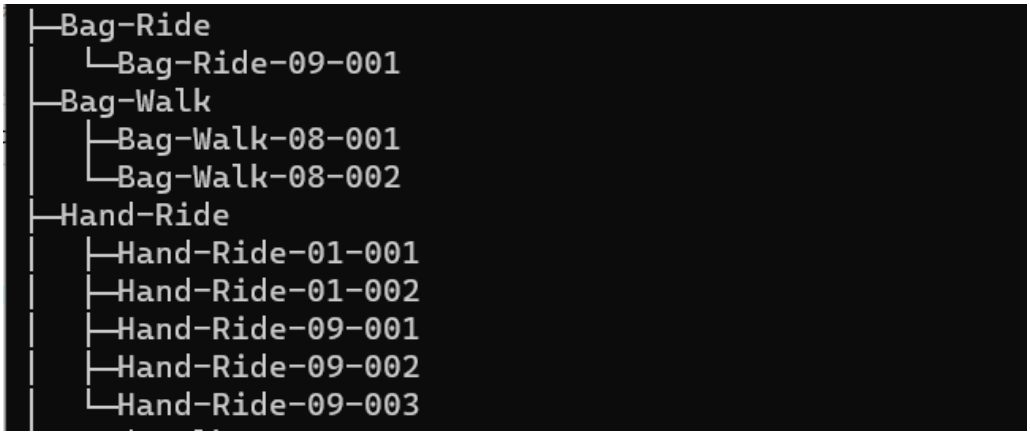
3.1 算法分析与选择

我们在分析数据集后发现，手机 IMU 传感器具有两个主要的缺点：噪声较大、会产生漂移问题；手机传感器的精度并不高；数据之间有较为明确的关系：例如，对于方位角我们可以通过磁场和角速度来计算得到，对于经纬度我们也可以通过物理公式计算求解。考虑到数据集的数量（大约 80 条）以及数据的特性，我们选择使用物理建模的方式，难点和重点在于物理公式的推演，数据的去噪和平滑以及误差的矫正方面。

3.2 数据预处理

3.2.1 数据集类使用方法

我们将所有的步行数据按照二级结构存储，第一级用于存储他所属的类型（例如：背包里-骑车），第二级用于存储某一条路径记录，二级文件夹下就是各种传感器的数据了：



当代码编写者需要调用数据集中的数据时，只需要在 config.json 中配置好数据集文件的路径，就可以按照如下示例进行调用：

```
dataset = PedestrianDataset(["Bag-Ride", "Bag-Walk"], window_size=200,
                            acceleration_filter=default_low_pass_filter)

for name, locus in dataset:
    print("正在遍历移动轨迹{}... \n".format(name))

    for sample in locus:
        for k, v in sample.items():
            print(k + ":" + str(v.shape))
        break

    print(locus.columns_info())
    break
```

3.2.2 数据集类内部具体处理方式

在开始处理数据时，我们就发现了数据的时间对齐其实是存在问题的，每一个处理器都是按照自己的周期来进行采样，因此每个处理器 csv 文件的记录对应的时间戳是无法严格对齐的。但是，在对数据的性质进行了仔细地分析以后，我们认为数据其实是符合 L-

Lipschitz 条件的。因此，某一数据点与它相距不到 0.01s 的状态应该是十分相近的，我们直接按照最近对齐的方式将所有传感器数据按照加速计的时间进行了合并对齐。其中比较特殊的是 Location 对应的 GPS 数据，它并没有直接和传感器数据进行对齐，而是进行了一些特殊的处理。

对于 Location 数据，其中的位置数据是以经纬度的形式进行存储的，但是，我们的预测系统最好是在以 m 为单位的笛卡尔坐标系上运行，毕竟，这样许多公式才有具体物理含义。我们利用地理支持库计算出了两点之间的距离与方位角，分别用$dist(P_1, P_2)$与$bear(P_1, P2)$来进行表示，用 O 表示原点：

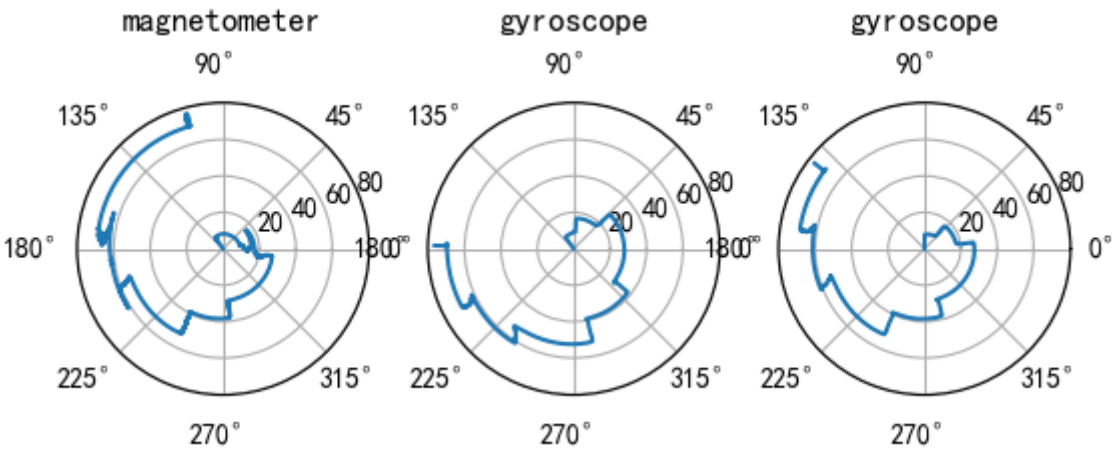
$$P_x = O_x + dist(O, P)\sin(bear(O, P))$$
$$P_y = O_y + dist(O, P)\cos(bear(O, P))$$

当最后程序要输出到 Location_output.csv 时，再通过逆变换将笛卡尔坐标系转换回经纬度表示的坐标体系。

3.3 方位角预测

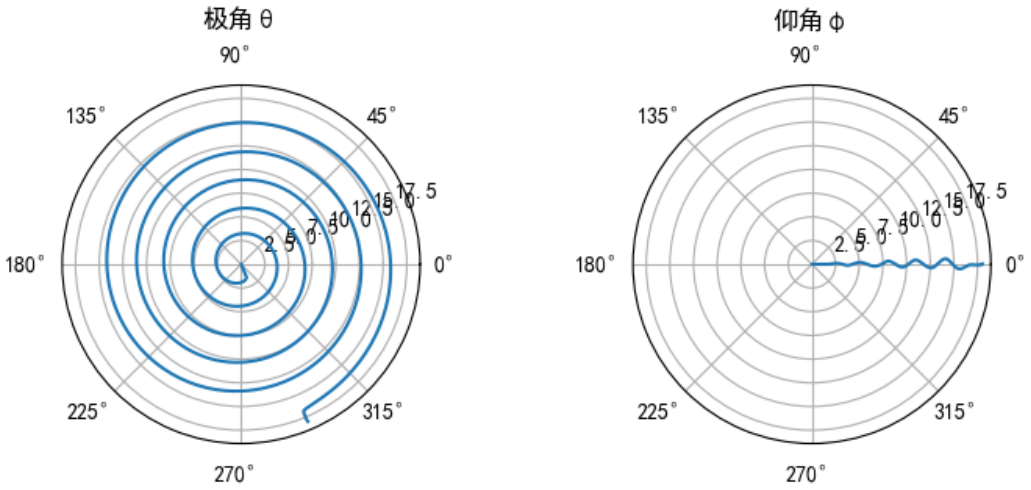
3.3.1 初步尝试：直接使地磁与重力加速度

最开始，我们希望在每一时刻，直接利用地磁场数据（磁力计直接得到）与重力加速度数据（加速计与先加速计相减得到）来确定人和手机移动的方向角。不过，在尝试的过程中，我们发现磁力计的数据变化不是非常稳定，而测力计的数据噪声也很高。此外，经过研究也发现，由于手机的姿态有着 3 个自由度（绕 xzy 轴旋转），而之前提到的“地磁场数据”与“重力加速度数据”仅仅能帮助我们确定两个方向的信息，按照我们有限的了解，根据这两个数据是不太能完全矫正手机的姿态的。因此，我们决定更改思路，通过追踪陀螺仪的数据，来确定手机的姿态。



对比磁力计与陀螺仪效果

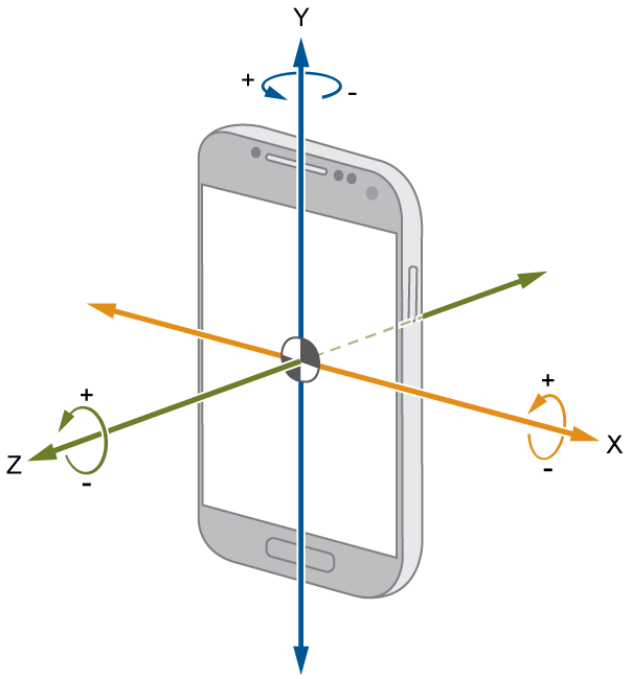
我们可以发现，左侧的磁力计效果是十分不稳定的，尤其是当运动速度变快的情况下。而即使是我们将手机放在电脑椅上高速旋转，陀螺仪仍可以精确捕捉旋转轨迹。



手机在电脑椅上旋转

3.3.2 最终方案：使用陀螺仪追踪手机姿态变化

这是一个典型的牛顿力学问题，如果我们能够确定确定手机的初始姿态，再追踪确定手机在每一个轴上的旋转量，我们就可以获得手机在每一个时刻的姿态，进而通过计算，获得人移动的方向角。为统一讨论的方便，我们将 z 轴上的旋转角记为α，将 y 轴上的旋转角记为β，将 x 轴上的旋转角记为γ。



手机 IMU 坐标系

首先我们介绍一下，我们是如何测量初始姿态的。为了解决前文提出的困难，并使问题变得可处理，我们对手机的初始状态做出了一个假设：数据手机开始的时候，手机保持着被手持的姿势，也即，手机在 IMU 坐标系的 y 轴上旋转角为 0。

我们所假设的手持手机姿态

我们记重力加速度数据为 g，记地磁场数据为 B，其中：

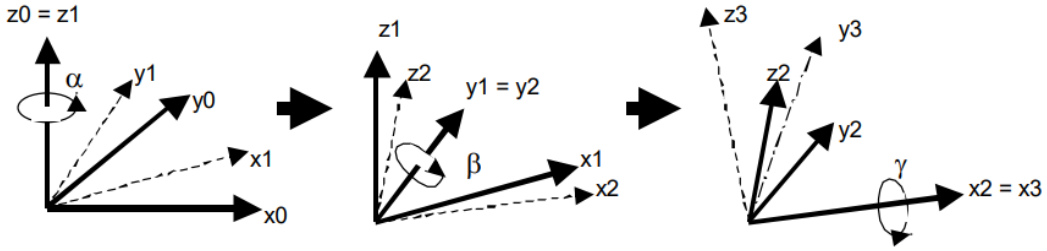
$$g = a_{\text{加速器}} - a_{\text{线加速器}} \in \mathbb{R}^3$$

因此，我们可以得到：

$$\beta = \frac{\pi}{2} - \arctan 2(g_z, \sqrt{g_x^2 + g_y^2})$$

$$\alpha = \arctan 2(g'_x, g'_y), \quad where\ g' = \mathbf{R_z}(\beta)g$$

这样，我们就得到了手机的初始姿态信息。之后我们使用旋转矩阵对姿态旋转进行建模，这里的建模使用的是 ZYX 内旋方式：



Rotation about z0 of angle α + Rotation about y1 of angle β + Rotation about x2 of angle γ

$$T_{0,3} = T_{0,1}T_{1,2}T_{2,3} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma) & -\sin(\gamma) \\ 0 & \sin(\gamma) & \cos(\gamma) \end{bmatrix} =$$

ZYX 的内旋矩阵构建

之后，我们通过持续追踪陀螺仪在各个轴上的旋转量，经由中矩形积分，可以得到

$$\Delta\theta_i = \frac{d\theta}{dt}_i \times (t_{i+1} - t_{i-1})/2$$

之后，我们将之前得到的当恰尼姿态矩阵，按照 xyz 轴的旋转角外旋，可以得到新的姿态矩阵，我们将姿态矩阵记为R^{IMU}_{earth}：

$$R^{IMU}_{earth} \leftarrow \mathbf{R_z}(\alpha)\mathbf{R_y}(\beta)\mathbf{R_x}(\gamma)R^{IMU}_{earth}$$

那么，这个姿态矩阵如何使用呢？我们可以用它来做两件事情：

1. 我们可以通过姿态矩阵，将手机加速计测得的加速度从 IMU 坐标系直接转换到 earth（世界）坐标系下，用于后续计算：
$$a^{earth} = R^{IMU}_{earth}a^{IMU}$$
2. 我们可以通过姿态矩阵，计算得到从 IMU 坐标系绕 z 轴旋转多少度可以到达世界坐标系，用于后续测算行人的移动方向：

$$\begin{bmatrix} \cos(\alpha)\cos(\beta) & \cos(\alpha)\sin(\beta)\sin(\gamma) - \sin(\alpha)\cos(\gamma) & \cos(\alpha)\sin(\beta)\cos(\gamma) + \sin(\alpha)\sin(\gamma) \\ \sin(\alpha)\cos(\beta) & \sin(\alpha)\sin(\beta)\sin(\gamma) + \cos(\alpha)\cos(\gamma) & \sin(\alpha)\sin(\beta)\cos(\gamma) - \cos(\alpha)\sin(\gamma) \\ -\sin(\beta) & \cos(\beta)\sin(\gamma) & \cos(\beta)\cos(\gamma) \end{bmatrix}$$

ZYX 的内旋矩阵

在通常情况下，我们只需要计算$\arctan 2(R_{21}, R_{11})$就可以得到$\alpha$对应的取值了，其中 R 对

应为姿态矩阵。然而，在特殊情况下，例如仰角$\beta \to \frac{\pi}{2}$，这种计算方法将出现数值上的不稳定。为此我们对姿态矩阵的奇异情况进行了特殊的处理，通过更换为 ZXY 的内旋矩阵建模的方式，同样求得了航向角α，并规避了原本的仰角β为 90 °的情况。

3.4 经纬度预测

3.4.1 初步尝试：直接卡尔曼滤波建模、牛顿运动力学直接计算轨迹

1. 我们曾尝试直接使用卡尔曼滤波建模，通过前 10%的方位信息去矫正观测值的误差，并且使用神经网络去学习偏差项。

$$\omega_n^{IMU} = \omega_n + b_n^{\omega} + w_n^{\omega}$$
$$a_n^{IMU} = a_n + b_n^{a} + w_n^{a}$$

w_n^{ω}, w_n^{a}即为我们尝试学得的误差项

但是由于 location.csv 的数据测量的频率远低于加速度计等数据的测量频率，EKF 会不断参与计算，在梯度回传的过程中很容易带来像 RNN 一样梯度消失或爆炸的问题，因此我们放弃使用神经网络。

2. 我们在做经纬度预测的时候，尝试过直接使用牛顿运动力学公式来计算移动的距离：

$$\vec{p}_{t+1} = \vec{p}_t + \vec{v}_t \Delta t + \frac{1}{2}\vec{a_t}\Delta t^2$$
$$\vec{v}_{t+1} = \vec{v}_t + \vec{a_t}\Delta t$$

牛顿运动力学公式计算移动距离

但是由于手机加速度计传感器有偏，且噪声干扰明显，最终位置的偏差较大。

我们直接使用运动公式的结果图示（预期图形为正方形）

我们也曾尝试过使用卡尔曼滤波来矫正，但是仍然难以校准控制向量的误差，所以我们最后放弃这个思路。

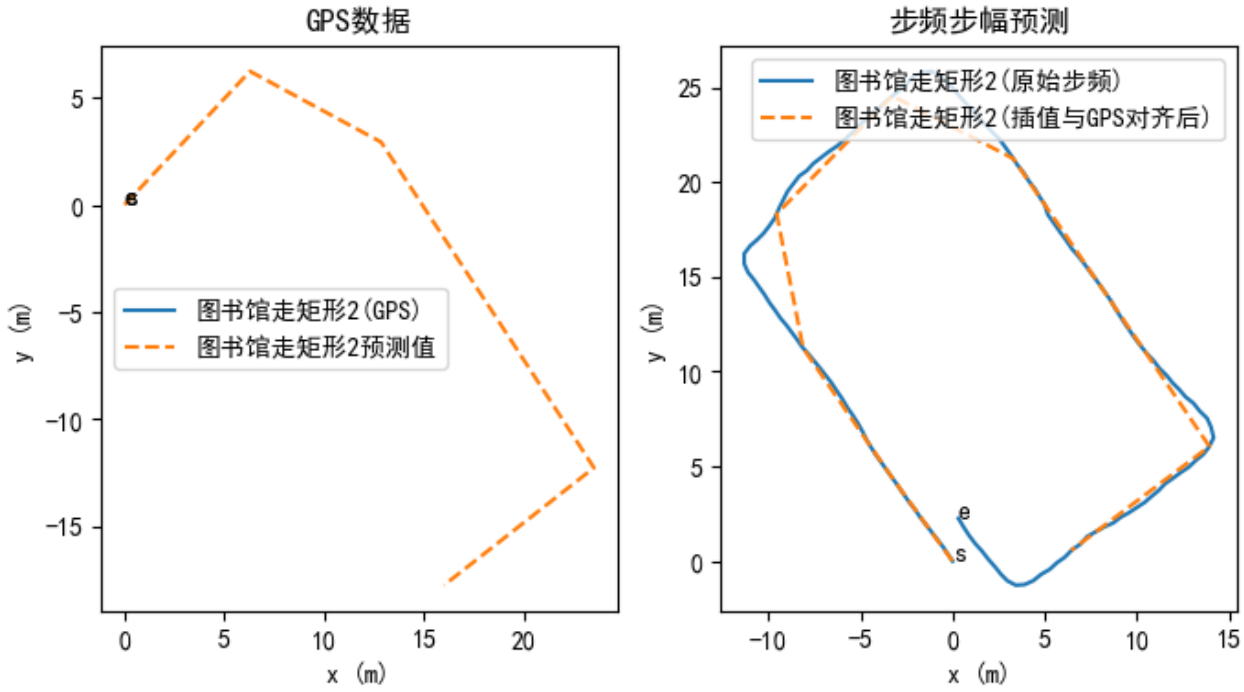
3.4.2 最终方案：使用步频+步幅预测轨迹

1. 由于直接使用牛顿运动力学公式的偏差过大，我们后来选择使用步频+步幅的方式来预测运动的距离：

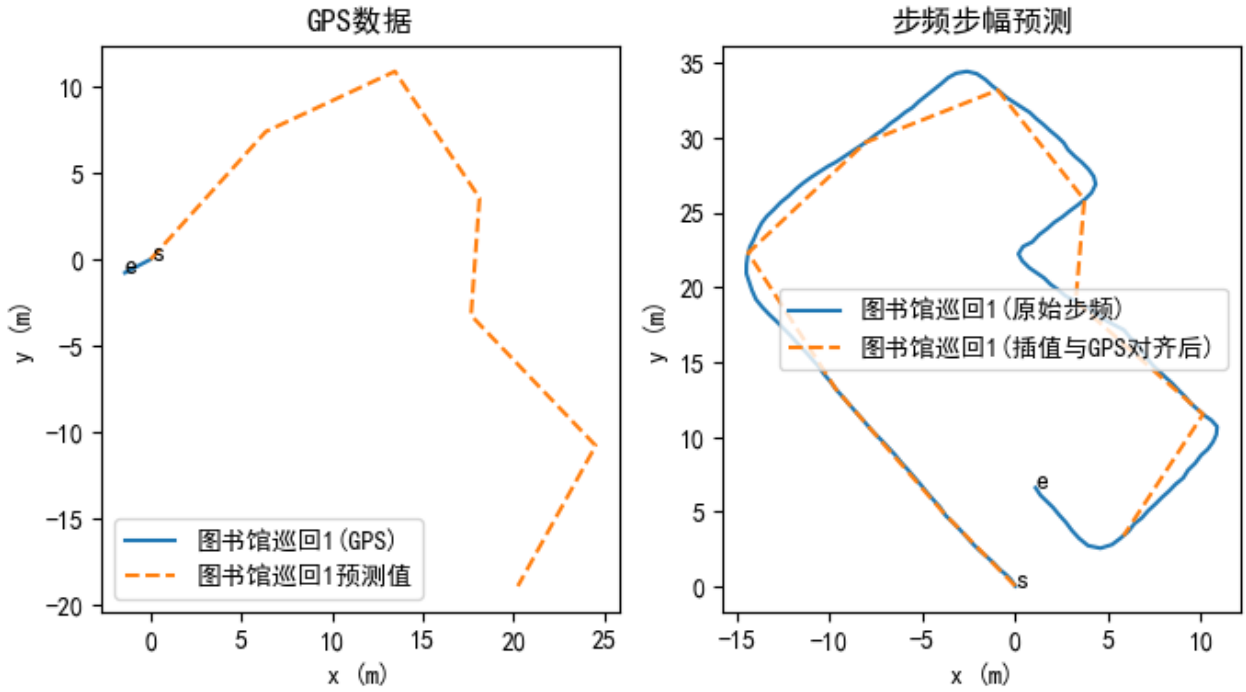
$$\overrightarrow{p}_{t+1} = \overrightarrow{p}_t + \overrightarrow{\Delta p}$$

对于步频，我们认为世界坐标系的 z 方向速度的变化和步频强相关。我们首先对线加速度计得到的的加速度做低通滤波，利用姿态矩阵将其变换到世界坐标系后，对时间积分，得到 z 方向的速度。然后根据 z 方向的速度的波形图来计算峰值的个数，得到步频。

在最开始，我们粗略假定步幅不变，设定步幅：$|\overrightarrow{\Delta p}| = 0.8$，发现得到了和预期结果接近的轨迹图，相比直接使用牛顿运动力学预测有极大的效果提升。



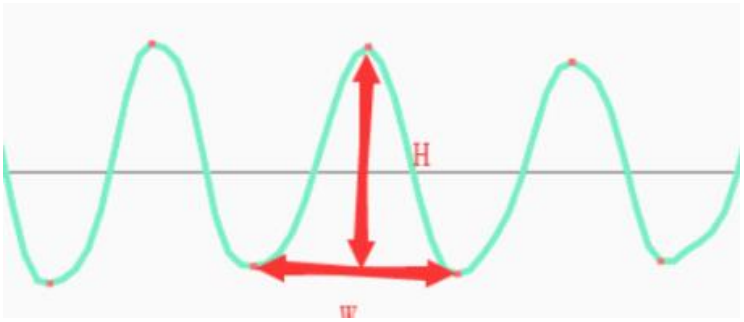
步频+步幅预测轨迹（预期图形为矩形，可以看到比直接使用运动公式的效果有了极大提升）



步频+步幅预测轨迹（左图为 GPS 定位图，右图为步频+步幅预测图）

注：由于该仅用于测试效果的数据采样时位于室内，GPS 数据有所残缺。

2. 接下来我们对步幅的估计进行改进。我们首先对加速度做 8 阶的低通滤波，然后我们尝试根据世界坐标系的加速度的波幅（下图中的 H）和间距（下图中的 W）来估计步幅。



波幅（H）和间距（W）示意图

我们结合物理经验，利用如下公式来计算步幅：

$$steplength = A - B \cdot W + C \cdot \sqrt{H}$$

其中 A、B、C 为三个待学参数，W 和 H 分别为波幅和间距

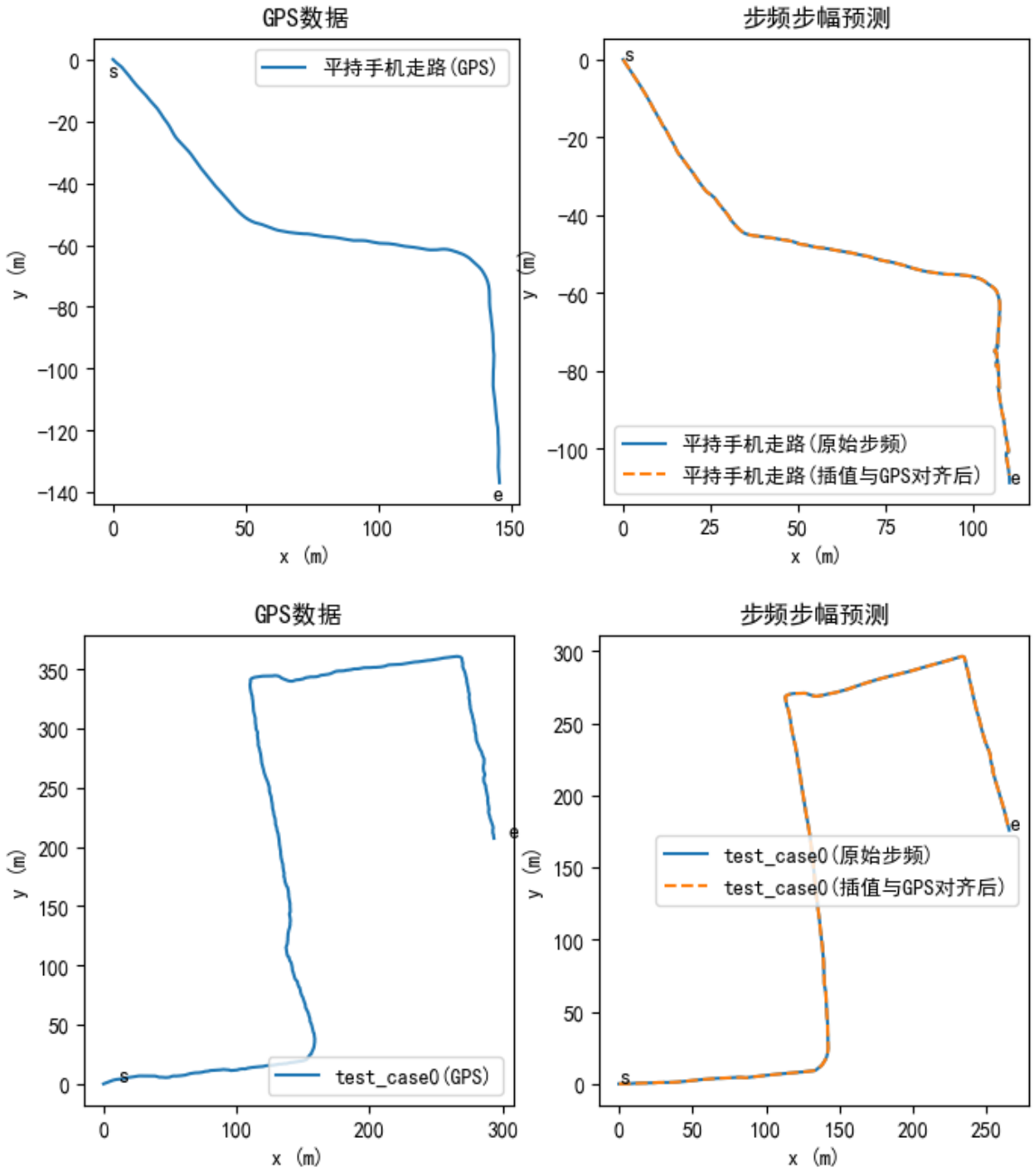
我们在训练集上对 100%的数据，利用最小二乘法解出 A、B、C 三个参数的最优值：

$$A = 0.407$$
$$B = 0.00013175$$
$$C = 0.18018$$

由于步幅是使用相邻的两个波峰（波谷）来计算的，因此步幅预测个数会比步频数少一个。对于最后一次步幅，我们使用指数平滑方法估计：

$$L_n = c\sum_{i=0}^{n-1}\lambda^i L_{n-i} \ , \ where \ \ c = \frac{1-\lambda}{1-\lambda^n}$$

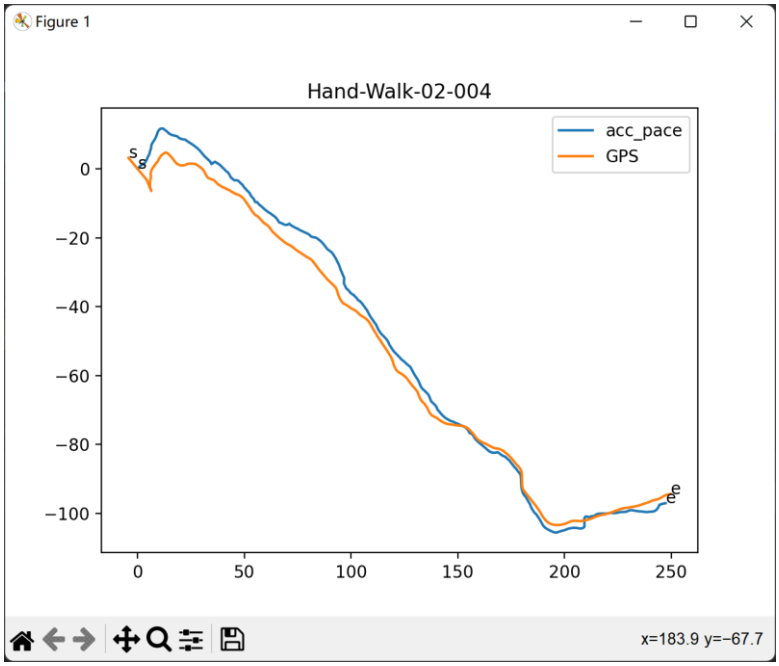
我们在测试集上部分试验结果如下：



其中左图为 GPS 测量数据，右图为估计步幅预测数据

可以看到，效果有了进一步的提升，但是仍然略有不准。我们猜测问题来源于每个人的走路姿态、手机传感器灵敏度都有不同，不能仅根据训练集来确定 A、B、C 三个参数。因此我们考虑针对每一条具体的测试数据，对 A、B、C 三个参数做进一步微调。

3. 我们对步幅估计进一步改进。我们考虑利用每一条测试数据的前 10%，使用最小二乘法对上述求出来的三个参数 A、B、C 做微调寻优，为了使得参数更好的学习，考虑到人每一步走路的变化趋势并不大，我们对稀疏的 GPS 做插值。针对前 10%方位数据微调后的结果图如下，可以看到我们可以做到非常好的拟合结果。



对前 10%的方位数据参数寻优后的结果，橙色曲线为 GPS 数据，蓝色曲线为预测数据

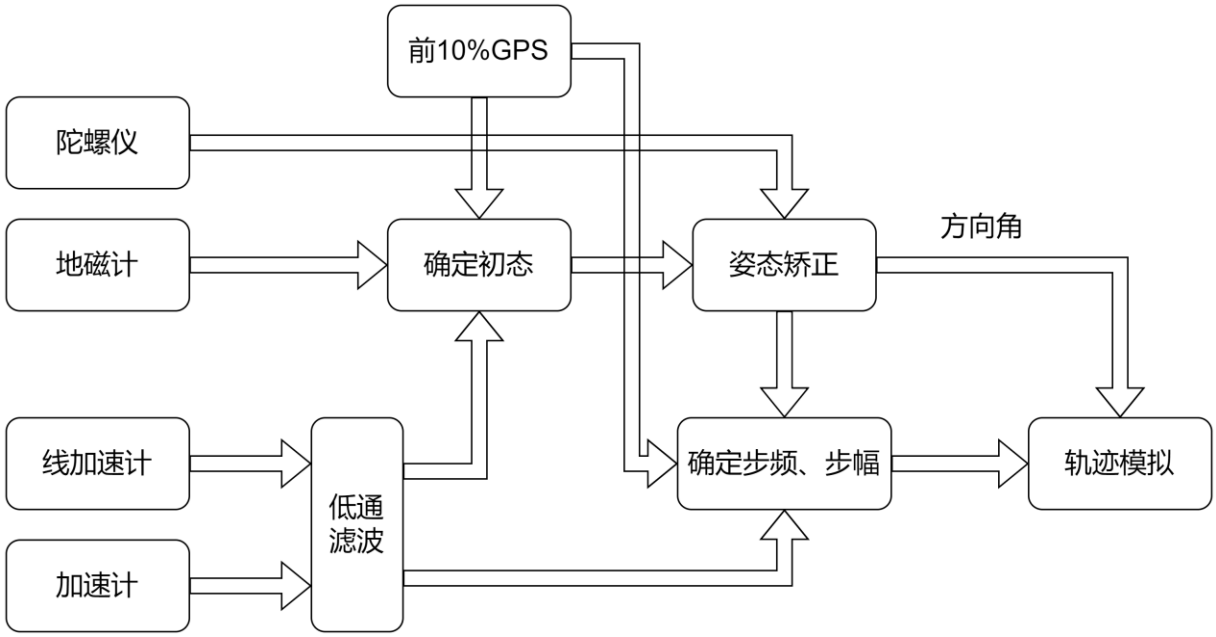
3.5 最终算法实现

我们最终的算法过程如下：

1. 对数据做预处理，对不同传感器测量的数据做最近邻时间对齐，去除经纬度前若干个偏离明显的异常值后转换到笛卡尔坐标系上。
2. 使用地磁计、做完低通滤波后的线加速计和前 10%的 GPS 数据确定轨迹初态。
3. 使用陀螺仪追踪手机姿态的变化，对姿态建模，将加速度映射到世界坐标系上，并计算方向角。
4. 使用做完低通滤波后的加速计数据来预测步幅和步频，并使用前 10%的 GPS 数据进行矫正。
5. 利用前面计算出的方向角和步幅步频，模拟并计算出轨迹。

6. 将笛卡尔坐标系的轨迹映射回经纬度。

我们最终的算法流程图如下：



算法流程

4. 算法性能评估与代码运行

4.1 测试集评估结果

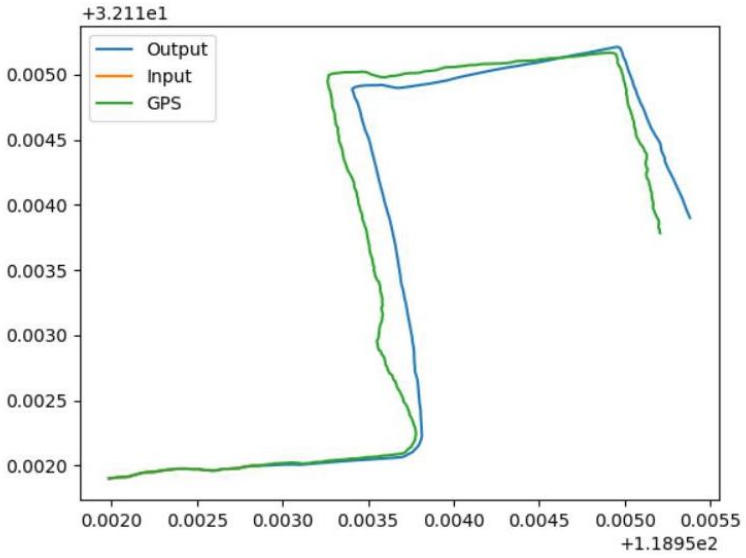
4.1.1 总体测试集评估

我们的测试集包含了来自不同状态、不同区域、不同设备上的 17 条数据。
我们在我们收集的数据测试集的运行情况如下图所示：

```
----------data: test03 ----------
magic:[ 0 ,  0.40700000000000003 ,  0.00013175 ,  0.18018 ]
magic:[ 0.3405514192591226 ,  0.40700000000000003 ,  0.00013175 ,  0.18018 ]
local_error
>15° percent: 23.893805309734514%
Distances error:  33.90900254585608
Direction error:  15.577968428338425
----------data: test04 ----------
magic:[ 0 ,  0.40700000000000003 ,  0.00013175 ,  0.18018 ]
magic:[ -4.349621743462083 ,  0.40700000000000003 ,  0.00013175 ,  0.18018 ]
local_error
>15° percent: 10.714285714285714%
Distances error:  27.614093461678237
Direction error:  7.908031478248625
----------data: test15 ----------
magic:[ 0 ,  0.40700000000000003 ,  0.00013175 ,  0.18018 ]
magic:[ 0.38529880772335506 ,  0.40700000000000003 ,  0.00013175 ,  0.18018 ]
local_error
>15° percent: 51.38121546961326%
Distances error:  16.36783127226145
Direction error:  19.071122234781797
测试集总距离平均误差：  79.33829331207392
测试集总方位角平均误差：10.888553933623669
```

在测试集的运行情况

在助教提供的 test_case0 数据上的预测结果如下：



我们模型在 test_case0 上的预测结果，绿色为 GPS 数据，蓝色为预测数据

通过分析模型在数据集运行后的结果可以得到如下初步结论：

1. 当手机姿态较为平稳时，我们模型表现的很好，方位角平均误差在 7 °左右，约 10%的方位角数据大于 15 °。当手机姿态波动较大（表现在走路时大幅度摆臂，或者走路时非正常抖动手机）时，模型性能会有所下降。

```
----------data: test11 ----------
magic:[ 0 ,  0.481 ,  0.00014725 ,  0.167895 ]
magic:[ 0.12535633352588527 ,  0.481 ,  0.00014725 ,  0.167895 ]
local_error
>15° percent: 0.5405405405405406%
Distances error:  42.70287699588609
Direction error:  6.216642312036951
```

```
----------data: test12 ----------
magic:[ 0 ,  0.481 ,  0.00014725 ,  0.167895 ]
magic:[ 0.16167674599842938 ,  0.481 ,  0.00014725 ,  0.167895 ]
local_error
>15° percent: 17.15686274509804%
Distances error:  10.267051113613274
Direction error:  9.058024175370747
```

当手机姿态平稳时我们模型的性能

```
----------data: test7 ----------
magic:[ 0 ,  0.37 ,  0.000155 ,  0.1638 ]
magic:[ -1.1217562657415279 ,  0.37 ,  0.000155 ,  0.1638 ]
local_error
>15° percent: 90.33232628398792%
Distances error:  131.47623487144557
Direction error:  61.495937417960015
```

```
----------data: test8 ----------
magic:[ 0 ,  0.37 ,  0.000155 ,  0.1638 ]
magic:[ 2.3901295589841993 ,  0.37 ,  0.000155 ,  0.1638 ]
local_error
>15° percent: 85.56338028169014%
Distances error:  59.282253352484496
Direction error:  38.475477122635574
```

当手机姿态波动较大时我们模型的性能

2. 我们的测试数据集共包含了 12 个不同的设备。我们的模型在不同设备上所表现的性能几乎没有明显变化，这也说明了我们模型具有较强的设备迁移能力。

4.1.2 消融实验

为了验证上述得到的两种初步结论，我们采用控制变量法，额外做了两组不同的实验。

1. 接下来为了进一步评估人走路姿态对我们模型的影响，我们将我们的模型放在从设备**“03”**采集出来的数据上评估（该数据集共有 8 条数据，**均来自不同的姿态**），得到结果如下：

```
测试集总距离平均误差：  107.7326621312805
测试集总方位角平均误差：28.42449114573326

测试集总距离方差：  77.81712664563064
测试集总方位角方差：24.85577282733517
```

在设备是“03”的数据集上的评估结果

可以看到，我们的模型在同一设备、不同的姿态的数据集上运行时，会产生较大的方差。这也说明我们模型对于不同姿态的波动有性能上的变化，但总体上性能仍然可以保持较高的水平。

2. 为了进一步评估我们模型的设备迁移能力，我们将模型放在手机姿态是**“手持平稳-走路”**的数据集上评估（该数据集共有 8 条数据，**均来自不同的设备**），得到结果如下：

```
测试集总距离平均误差：  41.90900813471081
测试集总方位角平均误差：14.409757516230762

测试集总距离方差：  20.50477101559648
测试集总方位角方差：5.687651351251497
```

在手机姿态是“手持-走路”数据集上的评估结果

可以看到，我们的模型在姿态较为平稳的数据集上，在不同设备上的运行结果的距离和方位角的方差均较小，这也说明我们的模型在不同的设备上的性能波动并不大，因此我们的模型具有较强的设备迁移能力。

结合上述两组实验得到的结果，可以得出结论：当人的走路姿态波动较大时，我们的模型预测效果和稳定性会有下降，但总体上仍能保持在较好的水平；我们的模型在不同设备上所展现的性能变化不大，即我们的模型具有较强的设备迁移能力。

4.2 如何运行

1. 首先在***/config*** 目录下新建一个 ***config.json*** 文件，具体配置方式请参见***/config/Help.md*** 文件，用于指定数据集根文件夹的绝对路径，或者数据集相对于代码文件夹的相对路径。

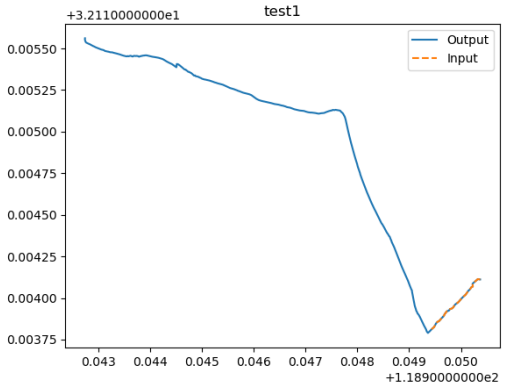
2. 在将我们收集的数据集路径正确配置好之后，运行***/testset_eval.py*** 文件，即可得到 ***4.1*** 所展示的包括测试集总体评估和两个消融实验在内的结果。

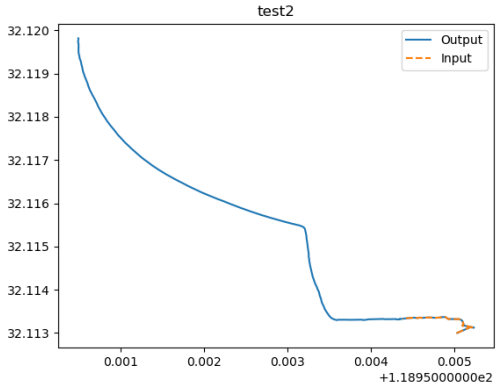
```
69  if __name__ == '__main__':
70      test()  # 测试集：模型总体性能评估
71      test_extra_1()  # 消融实验1：不同姿态对模型的影响
72      test_extra_2()  # 消融实验2：不同设备对模型的影响
```

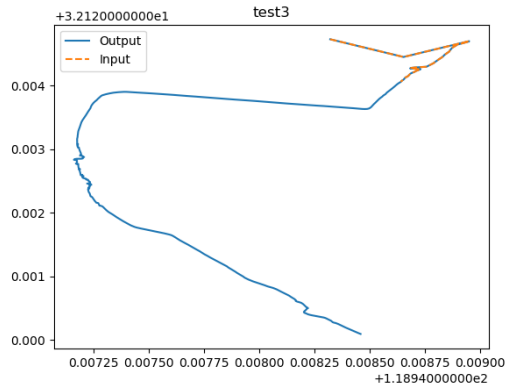
其中三个测试函数的内容如注释所示

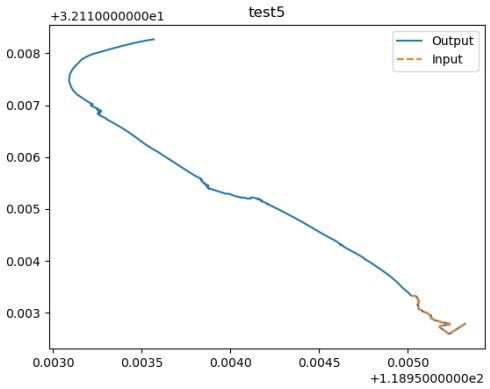
4.3 迁移测试集路径绘制

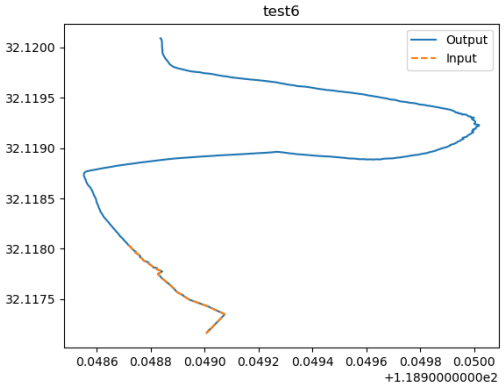
我们在助教提供的 10 个数据集上，根据我们模型的预测，绘制得到的路径结果如下：

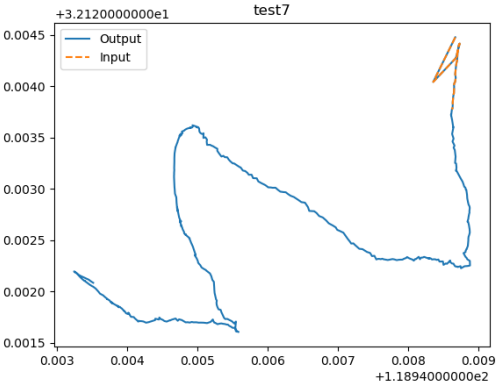


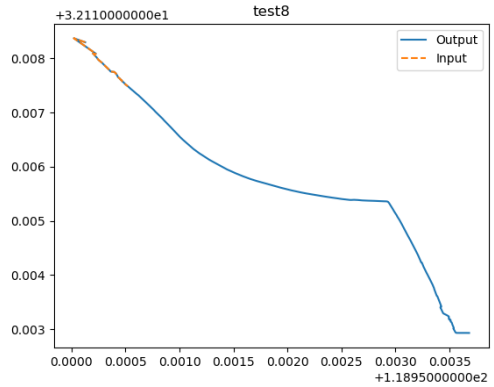


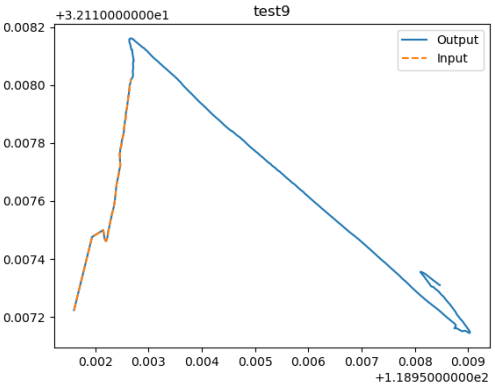

test3
+3.2120000000e1
Output
Input
0.004
0.003
0.002
0.001
0.000
0.00725 0.00750 0.00775 0.00800 0.00825 0.00850 0.00875 0.00900
+1.1894000000e2

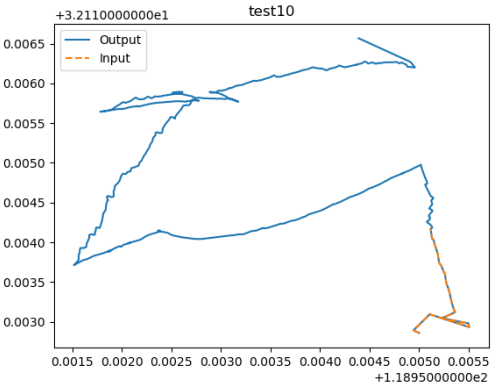

test5
+3.2110000000e1
Output
Input
0.008
0.007
0.006
0.005
0.004
0.003
0.0030 0.0035 0.0040 0.0045 0.0050
+1.1895000000e2

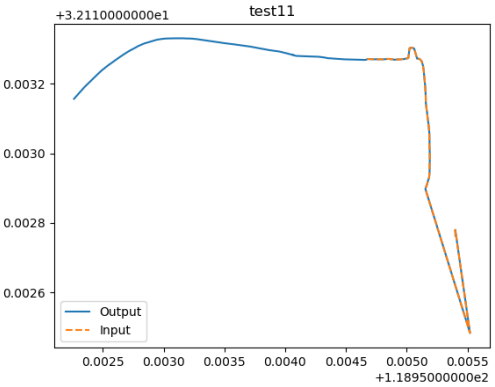

test6
Output
Input
32.1200
32.1195
32.1190
32.1185
32.1180
32.1175
0.0486 0.0488 0.0490 0.0492 0.0494 0.0496 0.0498 0.0500
+1.1890000000e2


test7
+3.2120000000e1
Output
Input
0.0045
0.0040
0.0035
0.0030
0.0025
0.0020
0.0015
0.003 0.004 0.005 0.006 0.007 0.008 0.009
+1.1894000000e2


test8
+3.2110000000e1
Output
Input
0.008
0.007
0.006
0.005
0.004
0.003
0.0000 0.0005 0.0010 0.0015 0.0020 0.0025 0.0030 0.0035
+1.1895000000e2


test9
+3.2110000000e1
Output
Input
0.0082
0.0080
0.0078
0.0076
0.0074
0.0072
0.002 0.003 0.004 0.005 0.006 0.007 0.008 0.009
+1.1895000000e2


test10
+3.2110000000e1
Output
Input
0.0065
0.0060
0.0055
0.0050
0.0045
0.0040
0.0035
0.0030
0.0015 0.0020 0.0025 0.0030 0.0035 0.0040 0.0045 0.0050 0.0055
+1.1895000000e2


test11
+3.2110000000e1
Output
Input
0.0032
0.0030
0.0028
0.0026
0.0025 0.0030 0.0035 0.0040 0.0045 0.0050 0.0055
+1.1895000000e2