ABSTRACT

Title of thesis: INDOOR ROUTES AND LOCATIONS INFERENCE USING SMARTPHONE IMU SENSORS

Xinyu Zhou, Master of Science, 2020

Thesis directed by: Professor Manoj Franklin
Department of Electrical and Computer Engineering

In this paper, we devise a framework to infer a smartphone user's location and walking routes in an indoor environment, using only the information from inertial measurement unit (IMU) sensors like gyroscope and accelerometer. To overcome the shortcoming of estimation drift over time in common PDR (pedestrian dead reckoning)-IMU systems, we propose a map-aided system which uses the map elements to help correct the user's position. We generate a map based on the environment parameters and a map matching algorithm is applied to find the most likely location of the user. The reading from the IMU sensors contains amounts of noise when user is walking, therefore we propose an edge detection algorithm based on the PELT model to smooth the piece-wise signals and identify the time frame when the user is making a turn. We evaluate our system when the smartphone is held either in the user's hand or in the backpack, and the system is able to give the correct walking path in both cases.

INDOOR ROUTES AND LOCATIONS INFERENCE USING SMARTPHONE IMU SENSORS

by

Xinyu Zhou

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2020

Advisory Committee:
Professor Manoj Franklin, Chair/Advisor
Professor Adrian Papamarcou
Professor Gang Qu

Acknowledgments

The experience in the University of Maryland is a long journey for me, with many ups and downs. I own my gratitude for all the people who have offered help to me during this journey, without which this work would be impossible.

First and foremost, I would like to thank my advisor Prof. Manoj Franklin for advising this thesis. Prof. Franklin is a wonderful mentor and has always made himself available for any advice and insights I need. I would not have found my path through the graduate school without the help and guidance from him.

I would also thank my committee members: Prof. Gang Qu and Prof. Adrian Papamarcou for agreeing on serve on the committee and offering their invaluable support, time and guidance to help me finish this thesis.

I would also like to acknowledge the help and support from my colleagues in the Department of Electrical and Computer Engineering. They are both excellent researchers and great friends, the discussions and interactions with them have enriched my graduate life in many ways.

My deepest gratitude goes to my parents who always stand with me and share my feelings. Their unconditioned support and love has helped me go through the most difficult times.

Lastly, thank you all.

Table of Contents

Acknowledgements iii

1 Introduction 1

2 Problem statement 4
- 2.1 Motivating scenario . . . 4
- 2.2 System model . . . 4
- 2.3 Challenge . . . 5

3 Background Research 7
- 3.1 Quaternion Algebra . . . 7
- 3.2 Quaternion and rotation . . . 9
- 3.3 Quaternion and Euler angles . . . 10

4 System design 13
- 4.1 Graph construction . . . 13
- 4.2 Orientation estimation using complementary filter . . . 15
  - 4.2.1 Overview . . . 15
  - 4.2.2 Implementation of complementary filter . . . 16
- 4.3 Turn detection . . . 20
- 4.4 Map matching algorithm . . . 26

5 Evaluation 29
- 5.1 Orientation Estimation . . . 29
- 5.2 Route inference . . . 30

6 Conclusion 33
- 6.1 Future Work . . . 33

Bibliography 35

Chapter 1: Introduction

The demand to precisely detect one's position has received increasing attention due to the development of location based service like advertising and social networking [1]. The GPS-augmented position detection system has been successfully implemented in outdoor environments, where the GPS signal can be reliably received. Using a mobile phone equipped with GPS receiver, the error for the positioning can be around 10 meters in well-conditioned outdoor environment [2]. However, the error may increase significantly in indoor environments, due to the lack of line of sight or the attenuation of the GPS signal when traversing through the building walls. Furthermore, most indoor environments contain complicated magnetic fields which are constantly distorted by electrical circuits or moving ferrous material. The polluted magnetometer readings caused by the distorted magnetic fields will introduce additional errors in the positioning result.

Different localization techniques have been proposed in order to overcome the limit of GPS aided system in indoor environment. In general, these techniques can be classified into two categories, infrastructure-based system and infrastructure-free system. For infrastructure-based systems, we need to set up one or more anchor nodes beforehand. The position of the user can be computed based on the information obtained from the communication to these anchor nodes. The information can be Time of Flight (ToF) [3][4][5], the angle of arrival (AoA) [6][7][8] or the signal strength [9][10]. Furthermore, the position obtained can be the relative position to certain anchor node, the global position will be computed using triangulation over the relative positions over multiple anchor nodes. The other

category is self-contained system which doesn't require any other devices installed. The most common technologies are pedestrian dead reckoning (PDR) systems using inertial measurement unit (IMU) sensors. PDR systems have gained growing popularity for smart-device localization [11]. Apart from the fact that IMU-PDR systems are completely self contained, do not require any other infrastructures installed, the other benefit of IMU-PDR is the flexibility in the sensor replacement and low requirement for the sensor accuracy [12]. An IMU-PDR system consists of two functions: the estimation of the distance travelled and the estimation of heading. The travelled distance can be measured with the integration of accelemeter data, while the heading direction usually replies on the integration of the gyroscope data and ,if possible, correction from magnetometer. Both components is subject to systematic drifts, which will grow over time. As a result, some kind of corrections or calibrations are required here to reset the drift. A more detailed discussion on the PDR systems and the drift calibrations can be found in [13] and [14].

The map-aided routes inference has been discussed in paper [15] and [16]. In [15] , the authors devise a framework to compute the most frequent driving routes only with a phone's gyroscope and accelerometer, they rely on the sequence of angular speeds to differentiate various routes. However, the walking route is harder to identify compared with driving scenario because the reading of the IMU sensors contains more noise, thus the angular speeds are more difficulty to extract. The paper [16] combines various map properties like the turn angles, curve similarity and travel time for better inference of driving routes. In this paper, the authors are able to produce a list of routes which contains the true route with probability of 60% in real driving experiment. In paper [17], the authors rely on the power consumption to infer the user's route and current location. The inference is based on the previously collected power consumption data and apply a machine learning model to distinguish the user's route among a fixed number of possible routes, the same model can also be used to predict the user's location at the end of a new route based on the power

profile.

In this paper, we propose a method to infer a user's route and location based on the knowledge of indoor environment. To be more specific, we are using the elements in the map to correct the drift in the positioning. For example, when we detect the user is making a left turn, given the indoor map, there are only a few places where making a left turn is possible, usually at a intersection. Therefore, we only need to search all such places and find out the most likely one to be the current position of the user, and any further position estimation will be based on the corrected current position. One challenge during the implementation is to precisely detect the turns the user has made. To achieve this, we adapt pruned exact linear time (PELT) algorithm from [18] which was originally proposed to find the changepoints in signals.

Chapter 2: Problem statement

2.1 Motivating scenario

The user is walking under indoor environment with a smartphone held in the hand, pocket, or backpack. The smartphone is consistently collecting the inertial motion unit (IMU) data and the data can be used to infer the user's walking route and current location. Because both Android and IOS have not yet limited the access to the IMU sensors, the application can easily access the IMU data without asking for any special permission to the operating system.

2.2 System model

We first represent the indoor environment with a set of roads and connections. Without loss of generality, we assume that all of these roads are straight. When a road intersects another road or makes a turn to a different direction, a connection is formed. These connections divide the road into different segments, which can be viewed as atomic parts. Now, any indoor map can be uniquely represented as $\Omega = (B, I, \theta, \lambda)$, where B is the set of road segments and I is the set of all connections. θ represents the turning angles corresponding to the connections; $\theta(c) < 0$ indicates a left turn and $\theta(c) > 0$ indicates a right turn. Lastly, λ is a function of road segments, with $\lambda(r)$ representing the length of road segment r.

Under this setting, the user's route can be denoted as a sequence of road segments

$R = (r_1, r_2, \ldots r_N)$ such that there is always a connection c corresponding to consecutive road segments (r_i, r_{i+1}). In other words, if $F : B * B \rightarrow I$ denotes a mapping from two road segments to the connections between them, we can write:

$$F(r_i, r_i + 1) \in I \quad \forall r_i \in R \tag{2.1}$$

The IMU data collected by the IMU sensor is denoted as $S = \{(a_t, g_t)\}$, where t is the sample index, a_t is the accelerometer data, and g_t is the gyroscope data. We will derive the walking route R, based on the sensor data sequence S. More formally, we define the following:

Definition 1 (Indoor Route Inference): *Given the sensor data S and the indoor map* Ω*, the output of the inference is the road segment sequence R representing the user's walking route. Furthermore, we also assume knowledge of the user's initial location and walking direction.*

2.3 Challenge

Here we list the challenges we are faced with to derive the accurate walking routes.

1. Noisy sensor reading: most off-shelf phones are not equipped with high quality IMU sensors due to cost consideration. The data produced by the sensors may contain initial bias, which could result in angle drift. Furthermore, the sensor readings are also influenced by the environment temperature; the error caused by the temperature dependence cannot be fully eliminated by factory calibration [19].

2. Walking behavior: the walking behavior of the user further affects our direction estimation. The accelerometer readings fluctuate periodically because of the up-and-down bounce during walking, which requires special caution while trying to identify

and extract the user's walking direction. Moreover, if the user holds the phone in her hand, the swing of the hands can also introduce periodical trends into the collected data. All these outside influences need to be carefully addressed so as to derive the accurate walking direction.

3. Absence of magnetometer: we will not use the data from the magnetometer because its reading can be easily influenced by nearby magnetic fields caused by electronic devices. These errors are even more pronounced in an indoor environment. Without magnetometer, we are unable to calibrate the tilt error, which is the component of all drift errors except rotation about the vertical axis in the physical world.

Chapter 3: Background Research

We already know that any rotation can be represented by a 3×3 rotation matrix with determinant 1. However, the matrix, which has 9 elements in it, only have 4 independent parts. The composition of two rotations expressed matrix form require the multiplication of two matrices, which takes 27 multiplications and 18 additions. Compared with rotation matrix, quaternion is more efficient in describing the rotation in $\mathbb{R}^3$. Any rotation of angle θ around the axis $u = [u_x, u_y, u_z]$ with $|u| = 1$ can be expressed as

$$q = [q_0, q_1, q_2, q_3]^T = [\cos\frac{\theta}{2}, u_x \sin\frac{\theta}{2}, u_y \sin\frac{\theta}{2}, u_z \sin\frac{\theta}{2}]^T \tag{3.1}$$

3.1 Quaternion Algebra

In this section, we will give the basic definitions of quaternion and some related operations like addition, multiplication and inverse.

We can also express quaternion with three orthogonal basis $i = (1,0,0)$, $j = (0,1,0)$ and $k = (0,0,1)$. Then

$$q = [q_0, q_1, q_2, q_3]^T = q_0 + q_1 i + q_2 j + q_3 k.$$

The addition of two quaternions is realized by adding each components. If we have two quaternions $q = q_0 + q_1 i + q_2 j + q_3 k$ and $p = p_0 + p_1 i + p_2 j + p_3 k$, the addition can be

expressed as

$$\begin{aligned} q+p &= q_0+q_1i+q_2j+q_3k+p+p_0+p_1i+p_2j+p_3k \\ &= q_0+p_0+(q_1+p_1)i+(q_2+p_2)j+(q_3+p_3)k \end{aligned} \tag{3.2}$$

The multiplication of the basis satisfies the following rules and other standard algebraic rules except the communication law,

$$\begin{aligned} &i^2=j^2=k^2=ijk=-1, \\ &ij=k, \quad ji=-k, \\ &jk=i, \quad kj=-i, \\ &ki=j, \quad ik=-j. \end{aligned}$$

The product of two quaternions can be given as,

$$\begin{aligned} qp &= (q_0+q_1i+q_2j+q_3k)(p_0+p_1i+p_2j+p_3) \\ &= q_0p_0-q_1p_1-q_2p_2-q_3p_3+(q_0p_1+q_1p_0+q_2p_3-q_3p_2)i+ \\ &(q_0p_2-q_1p_3+q_2p_0+q_3p_1)k+(q_0p_3+q_1p_2-q_2p_1+q_3p_0)k \\ &= \begin{bmatrix} q_0p_0-q_1p_1-q_2p_2-q_3p_3 \\ q_0p_1+q_1p_0+q_2p_3-q_3p_2 \\ q_0p_2-q_1p_3+q_2p_0+q_3p_1 \\ q_0p_3+q_1p_2-q_2p_1+q_3p_0 \end{bmatrix} \end{aligned} \tag{3.3}$$

Let $q=q_0+q_1i+q_2j+q_3k$, the complex conjugate of q can be written as

$$q^*=q_0-q_1i-q_2j-q_3k \tag{3.4}$$

The norm of q, $|q|$ is the square root of the product of q and q^*,

$$|q|^2 = qq^* = q^*q = q_0^2 + q_1^2 + q_2^2 + q_3^2. \tag{3.5}$$

A quaternion is called unit quaternion when its norm is equal to 1.

At last, the inverse of a quaternion q is defined as

$$q^{-1} = \frac{q^*}{|q|^2}, \tag{3.6}$$

we have

$$q^{-1}q = qq^{-1} = 1. \tag{3.7}$$

3.2 Quaternion and rotation

As we mentioned before, any rotation between two frames can be described with a unit quaternion, let us define the $q_A^B = [q_0, q_1, q_2, q_3]$ as the rotation of a frame A with respect of a frame B. Then the rotation of B respect to A, q_B^A is represented as the complex conjugate of q_B^A,

$$q_B^A = q_A^{B*} = [q_0, -q_1, -q_2, -q_3]^T$$

To express the overall rotation after a sequence of rotations in quaternion, we can compute the product of the corresponding quaternions of these rotations. To be more specific, we know the rotation of a frame A with respect to a frame B as q_A^B and the rotation of B with respect of a frame C as q_B^C, then the rotation of A with respect of B can be computed as

$$q_A^C = q_A^B \times q_B^C \tag{3.8}$$

For any vector $v = [v_1, v_2, v_3]^T$ in $\mathbb{R}^3$, we can write it as a pure quaternion whose real

part is zero

$$v^q = [0, v_1, v_2, v_3] \tag{3.9}$$

Then given a direction vector v_A with respect to the frame A and the rotation of A from B, q_A^B, we can express the same vector in the observation of the frame B as

$$v_B^q = q_A^B \times v_A^q \times q_A^{B*} \tag{3.10}$$

and we can get v_B simply using the transformation in equation (3.9).

If we want get the relation between v_A and v_B without writing them in quaternion form first, we can use the direct cosine matrix (DCM) in which case, the rotation in equation (3.10) can be rewritten as

$$v^B = R(q_A^B)v^A \tag{3.11}$$

where $R(q_A^B)$ is the DCM in terms of q_A^B, which is writtern as

$$R(q) = \begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix} \tag{3.12}$$

3.3 Quaternion and Euler angles

Given the coordinates in 3D space, Euler anglers describe the rotations of rigid body by specifying the rotation angles in each axis, called roll(ϕ), pitch(θ) and yaw(ψ). Each of them corresponds to one axis of the coordinate, showed in figure 3.1,

Given a quaternion $q = [q_0, q_1, q_2, q_3]$, we can get its corresponding Euler angles using

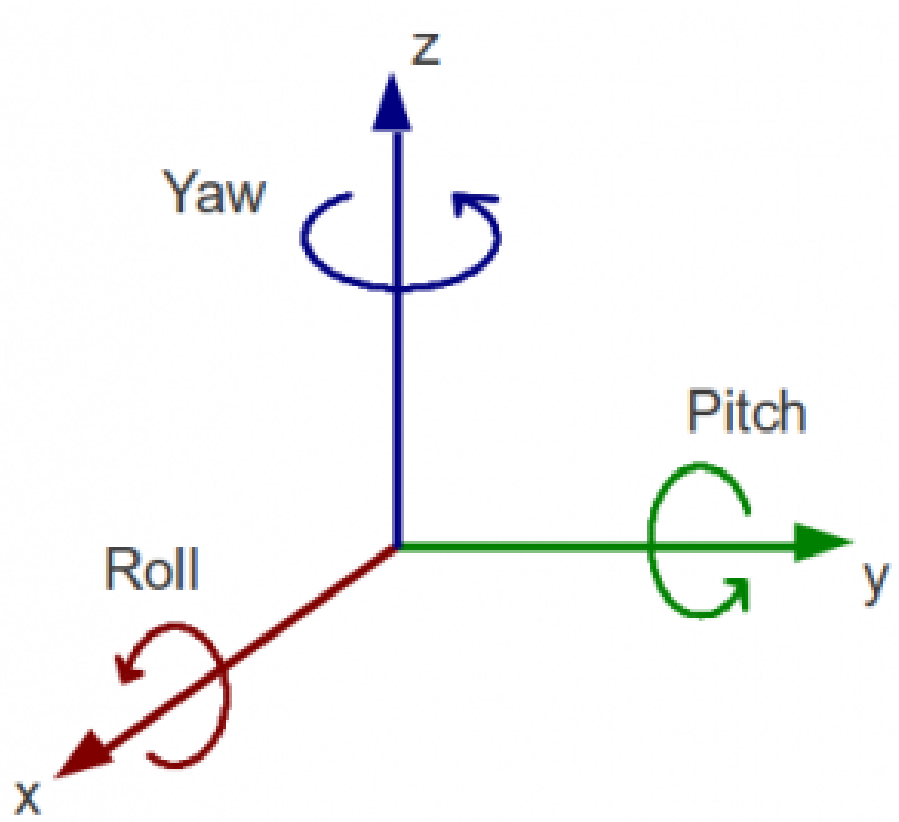


Figure 3.1: Yaw, roll and pitch rotations in 3D space

the following conversion [20]

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan \frac{2(q_0 q_1 + q_2 q_3)}{1-2(q_1^2+q_2^2)} \\ \arcsin\left(2(q_0 q_2 - q_3 q_1)\right) \\ \arctan \frac{2(q_0 q_3 + q_1 q_2)}{1-2(q_2^2+q_3^2)} \end{bmatrix} \tag{3.13}$$

However the angles returned in equation (3.14) are between $-\pi/2$ to $\pi/2$. To extend the result to the range $[-\pi, \pi]$, we need to replace the arctan function with a more general

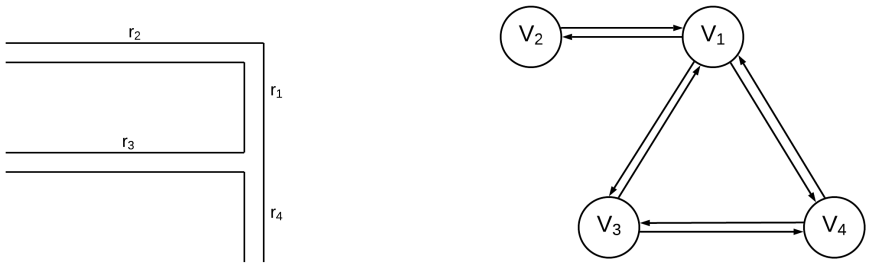
2-argument arctangent function atan2:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \text{atan2}(2(q_0 q_1 + q_2 q_3), 1 - 2(q_1^2 + q_2^2)) \\ \arcsin(2(q_0 q_2 - q_3 q_1)) \\ \text{atan2}(2(q_0 q_3 + q_1 q_2), (1 - 2(q_2^2 + q_3^2))) \end{bmatrix} \tag{3.14}$$

Chapter 4: System design

4.1 Graph construction

In order to perform the routes inference, we need to construct the directed graph $G = (V, E)$ based on the indoor environment Ω. Each road segment r is represented as a vertex in V and the connections between two road segments is represented by a directed edge in E. Each vertex will have a weight $w(v)$ denoting the length of this road segment. We also assign a weight to each edge and the weight of edge represents the change in the direction from the initial vertex to the end vertex of this edge.



(a) (b)

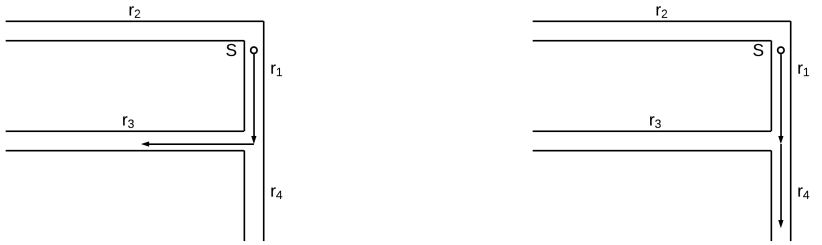
Figure 4.1: Example of an indoor environment and its corresponding graph. V_1 in (b) corresponds to r_1 in (a), the connection between r_1 and r_2 is represented as edge between V_1 and V_2.

Intuitively, the user will stay in one of the vertex v and there are two cases that will transverse her to another vertex v'.

1. he makes a turn on a connection into another segment.

2. the walking distance since he enters this segment is longer than its length, in which case v' is the extension of v over a connection and the weight of the edge $e(v,v')$ is equal to 0.

Figure (4.2) provides examples to illustrate the traverse from one vertex to another.



Case 1:the user starts at r_1 and turns right into r_3.

Case 2: the user walks from r_1 into r_4 without making a turn.

Figure 4.2: Examples of traversing from one vertex to another.

The road segments in the map are typically bidirectional. That is, on a road segment with two connections c_1 and c_2 at its both ends. The user can either walk from c_1 to c_2 or from c_2 to c_1. To keep track of the direction the user walk on a road segment, we add a binary property to each edge in G, for two outgoing edges connected to a vertex v, denoted as $e_1 = e(v,v_1)$ and $e_2 = e(v,v_2)$, we define a new function $D : E \to \{0,1\}$ such that $D(e_1) = D(e_1)$ if and only if v_1 and v_2 are connected with v on the same connection. Hence, the direction function divides all the edges starting from the same node into two groups. For example, in Figure (4.1), the vertex v_1, which corresponds to r_1 in geographic area, has 3 neighbors v_2, v_3, and v_4. Apparently v_3 and v_4 are connected with v_1 with the same connection, hence we assign $D(e(v_1,v_3))$ and $D(e(v_1,v_4))$ as 1 and let $D(e(v_1,v_2))$

be 0. In the path $R_G = (v_1, v_2, \ldots v_N)$, we will have

$$D(e(v_i, v_{i-1})) \neq D(e(v_i, v_{i+1}))$$
$$\text{where } v_i \in R_G,\ 2 \leq i \leq N-1. \tag{4.1}$$

A more intuitive explanation for equation (4.1) is as follows: when the user enter a road segment with one end (connection), the exit from the road segment must be through another end (connection), which implies the connections before and after this road segment should be different with each other.

4.2 Orientation estimation using complementary filter

4.2.1 Overview

In this section, we will discuss how to estimate the orientation of the smartphone using complementary filter. There are two ways to calculate the orientation, based on different sensors used. Firstly, the orientation can be derived by integrating over the gyroscope data. Gyroscope measures the angular velocity in three axis of the local frame. If the sample rate is high enough, the measurement can be considered as continuous, thus simply integration over the angular velocity vector will give us one way to estimate the orientation. However, the gyro data always contains different types of errors, a common gyro error model [21] can be written as

$$\tilde{\omega} = K\omega + b \tag{4.2}$$

where K consists of the scale factor error and the cross-axis coupling error, b is the zero-rate offset (bias). The values of K and b can be determined by factory calibration. However, these errors cannot be completely eliminated due to the imperfections in the calibration

procedure or temperature dependent procedure. When we are integrating over the gyro data, the bias component will accumulate over time and our estimation will gradually drift away from the real orientation.

By using the accelerometer reading, we are able to correct part of the drift errors. The correction is achieved by seeing the gravity as a constant field, and if there is no other linear motions, after we bring the accelerometer readings back to global frame, it should point to the same direction of the gravity, which implies that

$$R^T(q_{acc})g = a \tag{4.3}$$

where R is the direct cosine matrix (DCM) in terms of the orientation quaternion q, $a = [a_x, a_y, a_z]^T$ is the accelerometer reading in local frame, and g represents the true earth gravitational acceleration which is usually defined as unit vector

$$g = [0, 0, 1]^T. \tag{4.4}$$

Thus equation 4.3 can be rewritten as

$$R^T(q_{acc}) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{4.5}$$

4.2.2 Implementation of complementary filter

Complementary filter is one way to combine signals affected by noise with different frequencies. In our case of orientation estimation, the complementary filter applies high-pass filter to the gyro data which suffers from low frequency drift error. Meanwhile, we apply

low-pass filter on the accelerometer data which is affected by high-frequency noise. Ideally, with appropriate cut off frequency, we are able to get completely noise-free orientation estimation.

Let the gyro reading at time k be $\omega^k = [\omega_a^k, \omega_b^k, \omega_c^k]^T$, and our orientation estimation q_{k-1}, the quaternion derivative based on the gyro reading can be written as

$$q'_k = -\frac{1}{2}\omega_k \otimes q_{k-1} \tag{4.6}$$

where $\otimes$ represents quaternion multiplication which is defined as,

$$p \otimes q = \begin{bmatrix} p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3 \\ p_0q_1 + p_1q_0 + p_2q_3 - p_3q_2 \\ p_0q_2 - p_1q_3 + p_2q_0 + p_3q_1 \\ p_0q_3 + p_1q_2 - p_2q_1 + p_3q_0 \end{bmatrix} \tag{4.7}$$

Finally, after integrating the quaternion derivative numerically using the sampling period $\Delta t = 1/f$ where f is the sampling rate, we can get the prediction of the orientation at time k as

$$\hat{q}_k = q_{k-1} + q'_k * \Delta t \tag{4.8}$$

Our prediction $\hat{q}_k$ will be further corrected using the accelerometer data. We first transform the accelerometer a into global frame by rotating it with our predicted orientation $\hat{q}_k$,

$$R(\hat{q}_k)a = g_p. \tag{4.9}$$

where $g_p = [g_p^x, g_p^y, g_p^z]^T$ is called "predicted gravity". If our prediction is precise and there is no linear acceleration, g_p should be equal to the true gravity g. If there exists deviation from g_p to g, we will compute the delta quaternion $\delta q_c c$ which can rotates g in g_p by

solving the following equation,

$$R^T(\Delta q_{acc}) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} g_p^x \\ g_p^y \\ g_p^z \end{bmatrix} \tag{4.10}$$

After Solving the equation 4.10, we can get

$$\Delta q_{acc} = \left[\sqrt{\frac{g_p^z + 1}{x}}, -\frac{g_p^y}{\sqrt{2(g_p^z+1)}}, \frac{g_p^x}{\sqrt{2(g_p^z+1)}}, 0 \right]^T \tag{4.11}$$

To avoid the influence from the high frequency noise in acceleromenter reading, we first scale it down before applying Δq_{acc} on the top of the prediction. This can be achieved using linear interpolation between Δq_{acc} and the identity quaternion $q_I = [1,0,0,1]^T$ such that

$$\bar{\Delta q}_{acc} = (1-\alpha) q_I + \alpha \Delta q_{acc} \tag{4.12}$$

where $\alpha \in [0,1]$ is the gain to characterize the cutoff frequency of the filter [22][23]. The correction quaternion can be derived after we normalize $\bar{\Delta q}_{acc}$, that is

$$\hat{\Delta q}_{acc} = \frac{\bar{\Delta q}_{acc}}{|\bar{\Delta q}_{acc}|} \tag{4.13}$$

where $|\cdot|$ is the L_2 norm of a vector.

In the last step, we can calculate the orientation estimation at step k by multiplying the prediction $\hat{q}_k$ with the correction quaternion $\hat{\Delta q}_{acc}$

$$q_{k+1} = \hat{q}_k \otimes \hat{\Delta q}_{acc} \tag{4.14}$$

When we get our estimate of the smartphone orientation in the form of quaternion, we

need to convert the derived orientation to the expression of Euler angles which contains the angles of yaw, pitch and roll. The yaw angle describes the rotation around the vertical axis in the physical world, therefore, the change in the yaw angle of the smartphone orientation implies the change in the user's walking direction (assume the relative position between the smartphone and the user does not change). In Figure (4.3), we provide an example of walking route in the indoor map and the corresponding estimate of yaw angles over time, we can see noticeable gaps in the yaw angle when the user are making a turn. In next section, we devise a statistical way to find the gaps or changepoints in the yaw angles and derive the time of the turns in the walking path.

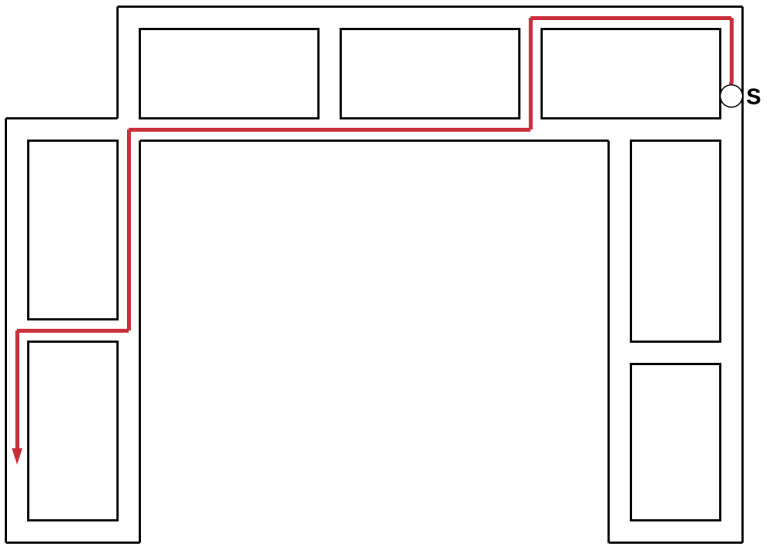
4.3 Turn detection

A changepoint is a sample or time stamp at which some statistical property of the signal change abruptly. In our indoor navigation case, if we choose the statistical property as the mean of the signal, the change point in the fused orientation signal implies the time when user is making a turn, either left or right. If we can record a sequence of changepoints in the yaw rotation, we can get all the turns the user have made and thus match the turning sequence to the map to get the exact location of the user.

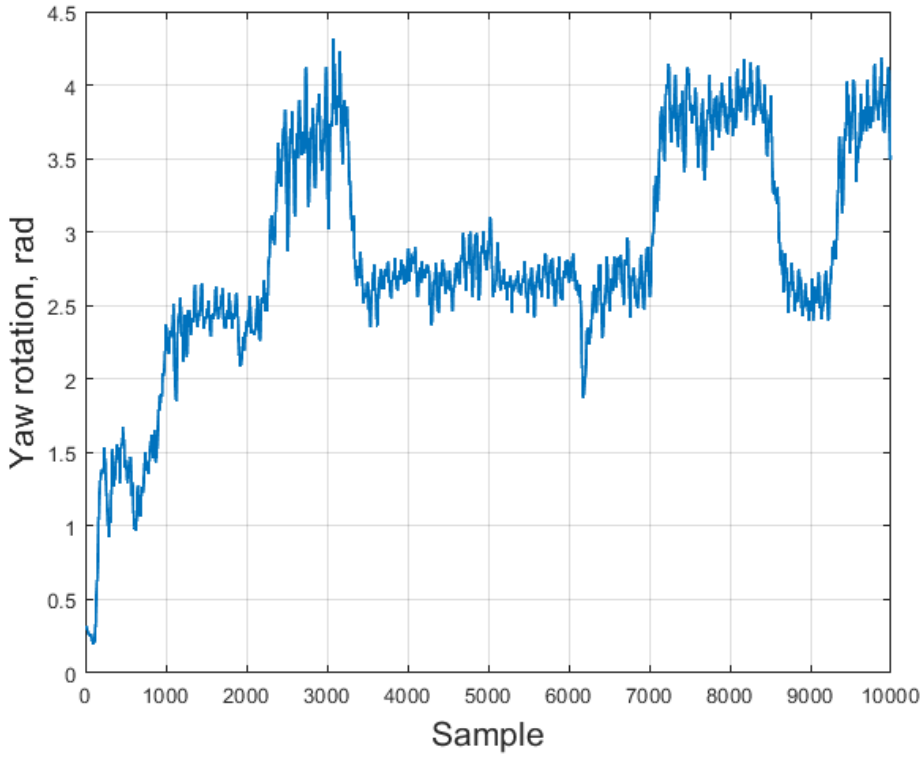
We first assume there is only one changepoint and the chose statistic is mean, we need to minimize the residual error from the best horizontal level for each section. Given signal $X = (x_1, x_2, \ldots x_N)$, we are trying to find the number of changepoints $1 \leq k \leq N$ together with a sequence $\tau = (\tau_1, \ldots, \tau_k)$ indicating the position of the changepoints. We further assume that $\tau_0 = 0$ and $\tau_{m+1} = N$. The m changepoints will divide the signal into $(m+1)$ segments and each segment can be written as $x_{(\tau_{i-1}+1:\tau_i)}$. To determine the number of changepoints and their positions, the following object function will be minimized,

$$\sum_{i=1}^{k+1} \left[C(x_{(\tau_{i-1}+1:\tau_i)}) + \beta \right] \tag{4.15}$$

where $C(\cdot)$ is the cost function for each segment and β is a penalty to prevent overfitting. The choice of the cost function depends on the statistics we are tracking, in our case, we are most interested in the mean of signal segments, and the abrupt change in the mean value indicates the fact that the user is making a turn. Therefore, the cost function for each segment is defined as the sum of the deviation between each point and the mean of this



(a)



(b)

Figure 4.3: (a) Example walking path (b) The corresponding estimated yaw rotation for each sample

segment, that is,

$$C(x_{(\tau_{i-1}+1:\tau_i)}) = \sum_{i=\tau_{i-1}+1}^{\tau_i} (x_i - < x >_{\tau_{i-1}+1}^{\tau_i})^2 \tag{4.16}$$

$$= (\tau_i - \tau_{i-1})\text{Var}(x_{(\tau_{i-1}+1:\tau_i)}) \tag{4.17}$$

We are going to use the pruned exact linear time (PELT) algorithm from paper [18] to detect the change points. PEIT is an extension of the optimal partitions method which conditions on the last point of change and relate the optimal value of the cost function to the optimal partition of the data prior to the last change point added with the cost of the segment from that point to the current point. The runtime of optimal partition is on $O(n^2)$ and PELT reduced the runtime to $O(n)$ on average by remove those values of positions that can never be a valid changepoints to minimize equation (4.15). To achieve that end, the following condition has to be satisfied,

Theorem 1 [18] We assume that when introducing a changepoint into a sequence of observations the cost, C, of the sequences reduces.More formally, we assume there exists a constant K such that for all $t < s < T$,

$$C(x_{(t+1):s}) + C(x_{(s+1):T}) + K \leq C(x_{(t+1):T}) \tag{4.18}$$

Then if

$$F(t) + C(x_{(t+1):s}) + K \geq F(s) \tag{4.19}$$

holds, at a future time $T > s$, t can never be the optimal last changepoint prior to T, where $F(s)$ denote the minimization from equation (4.15) for data $x_{1:s}$.

To use PELT, we remain to find the value of K so assumption (4.18) can be satisfied. We use the next corollary to prove that with the cost function defined in (4.17), assumption (4.18) will be satisfied when K is equal to 0.

Corollary 1: With the cost function C defined as

$$C(x_{(\tau_{i-1}+1:\tau_i)}) = (\tau_i - \tau_{i-1})\text{Var}(x_{(\tau_{i-1}+1:\tau_i)}) \tag{4.20}$$

For any $t < s < T$, we will have

$$C(x_{(t+1):s}) + C(x_{(s+1):T}) \leq C(x_{(t+1):T}) \tag{4.21}$$

Proof: Without loss of generality, we assume $t = 0$. We also denote the mean of data segment $x_{1:s}$ as m_1, the mean of $x_{s+1:T}$ as m_2 and mean of $x_{1:T}$ as m_3. It is straightforward to get

$$\begin{aligned} m_1 * s + m_2 * (T - s) &= m_3 * T \\ m_3 &= \frac{m_1 * s + m_2 * (T - s)}{T}. \end{aligned} \tag{4.22}$$

The left side of equation (4.21) can be written as

$$C(x_{1:s}) + C(x_{(s+1):T}) = \sum_{i=1}^{s} (x_i - m_1)^2 + \sum_{i=s+1}^{T} (x_i - m_2)^2. \tag{4.23}$$

Meanwhile, the right side of equation (4.21) can be written as

$$C(x_{1:T}) = \sum_{i=1}^{T} (x_i - m_3)^2 \tag{4.24}$$

Hence, by plugging in the value m_3 using equation (4.22) we have

$$\begin{aligned} & C(x_{1:T}) - \left[C(x_{(1:s)}) + C(x_{(s+1):T})\right] \\ &= \sum_{i=1}^{T}(x_i - m_3)^2 - \sum_{i=1}^{s}(x_i - m_1)^2 + \sum_{i=s+1}^{T}(x_i - m_2)^2 \\ &= \frac{s(T-s)}{T}(m_1 - m_2)^2 \geq 0 \end{aligned} \tag{4.25}$$

Therefore, for any $t < s < T$ we have

$$C(x_{(t+1):s}) + C(x_{(s+1):T}) \leq C(x_{(t+1):T}).$$

After assumption (4.21) is satisfied, we can write down the PELT method with $K = 0$,

Algorithm 1: PELT Method

input : A set of data of the form $(x_1, x_2, \ldots, x_n)$ where $y_i \in R$.
The cost function $C(\cdot)$ defined in equation (4.17).
A penalty constant β

initialization: Let n be the length of the data and set $F(0) = -\beta$, $cp(0) = NULL$, $R_1 = \{0\}$

for $\tau^* = 1, \ldots, n$ **do**

- Calculate $F(\tau^*) = \min_{\tau \in R_{\tau^*}}[F(\tau) + C(y_{(\tau+1):\tau^*}) + \beta]$.
- Let $\tau^1 = \arg\{\min_{\tau \in R_{\tau^*}}[F(\tau) + C(y_{(\tau+1):\tau^*}) + \beta]\}$.
- Set $cp(\tau^*) = [cp(\tau^1), \tau^1]$.
- Set $R_{\tau^*+1} = \{\tau \in R_{\tau^*}\{\tau^*\} : F(\tau) + C(y_{(\tau+1):\tau^*}) \leq F(\tau^*)\}$

output: the changepoints recorded in $cp(n)$

The last step in the turn detection is to find out the step number in each segment. The human body bounces up and down during walking and bounce manifests into a periodic sinusoidal signal in the IMU's accelerometer. Step detection is essentially to count the number of peak in the accelerometer signal [24]. Hence, we will apply a peak detection algorithm to get the count of the steps. Although the step counts may be subject to the error caused by the fluctuations in the signal, we can mitigate the influence by setting a threshold

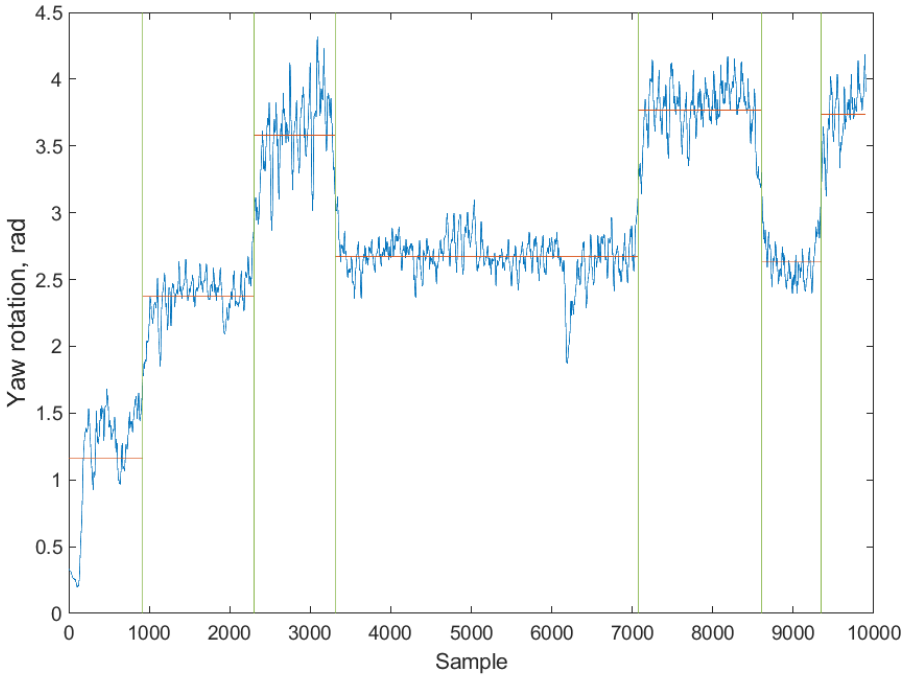


Figure 4.4: Using PELT to partition the yaw rotation signals into 6 parts which are separated by detected changepoints. The mean of each part is significantly different from that of neighboring parts.

for the minimum peak distance, hence we will focus on the tallest peak in the interval with the length set by the minimum peak distance and ignores the other peaks within this interval. Since the frequency of people's step is around 2 HZ, several peaks which are close to each other must contain false alarms that should not be counted towards the number of steps.

4.4 Map matching algorithm

After we successfully determine the direction and time of the turns the user has made using our segmentation algorithm, we will receive a sequence of turning pairs $T = \{(\alpha_1, s_1), (\alpha_2, s_2), \ldots (\alpha_N, s_N)\}$, where α_i is the angle of the i^{th} turn and s_i is the number of steps from the $i-1^{\text{th}}$ turn to the i^{th} turn. Our map matching algorithm takes the turning pairs as the input and maintains a list of routes in the map that are most likely to match the derived mobile trace. To be more specific, the algorithm tries to find $R = (v_1, v_2, \ldots v_M)$ such that the conditioned probability $P(R|T)$ is maximized.

Given the estimate of possible route $R_{i-1} = (v_1, v_2, \ldots v_k)$ at the $i-1^{\text{th}}$ turn, we are trying to find the updated route R_i based on the turning pair at the i^{th} turn, that is, (α_i, s_i). To achieve this, we need to figure out at which road segment the user makes the turn (pre-turn vertex, denoted as v_{pre}) and which road segment the user enters after the i^{th} turn (post-turn vertex, denoted as v_{post}). For example, in Figure (4.5), we already know that the user enters r_1 after the $i-1^{\text{th}}$ turn and then makes a left turn of 90° after s_i steps. In this case, r_3 is the road segment where the user is making the i^{th} turn (v_{pre}) and r_4 is the user's location after this turn (v_{post}).

One observation is that the pre-turn vertex v_{pre} is always on the extended line of v_k, the last road segment of R_{i-1}. To get v_{pre}, we first list all possible vertices $L = [v^0, v^1, \ldots v^t]$

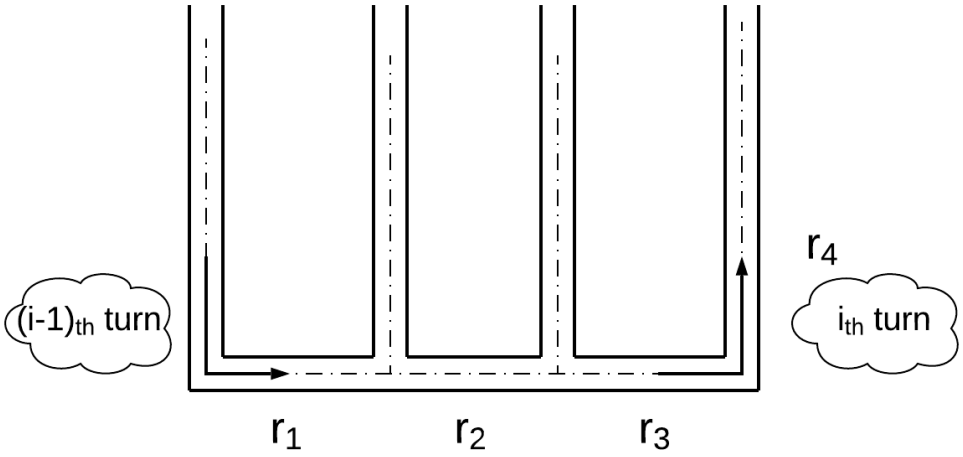


Figure 4.5: Examples of the route parts containing two consecutive turns. The user first enter r_1 at $(i-1)$th turn and go straight until r_3, then he enters r_4 by making a 90 degree left turn.

that are located on the extended line of v_k, where v_k is equal to v_0 and we also have

$$w(e(v^i, v^{i+1})) = 0 \quad \text{for } 0 \leq i < t. \tag{4.26}$$

Note that the weight of an edge represents the change in the direction from the initial vertex to the end vertex of this edge; hence when the weight of the edge is equal to zero, the road segments besides the corresponding connection are in the same line. Given the list of possible vertices L and the number of steps s_i, the pre-turn vertex v_{pre} can be derived by solving the following optimization problem

$$\begin{aligned} & v_{pre} = v^k \in L \\ & \text{with } k = \arg\min_j \left| \sum_{m=0}^{j} w(v^m) - s_i * sl \right| \end{aligned} \tag{4.27}$$

where sl is the step size of the user, which can be either measured beforehand or estimated

based on the IMU information [2].

After we figure out v_{pre}, the next step is to derive v_{post} which corresponds to the road segment the user enters after the turn. Given the pre-turn vertex v_{pre} and the angle of this turn α_i, let N be the set of neighbors of v_{pre} which satisfy the direction constraint defined in equation (4.1), that is,

$$N = \{v | e(v, v_{pre}) \in E \text{ and } D(v^{k-1}, v_{pre}) \neq D(v_{pre}, v)\}, \tag{4.28}$$

then we can get v_{post} by solving the following equation

$$v_{post} = \arg\min_{v \in N} |w(e(v_{pre}, v)) - s_i|. \tag{4.29}$$

Intuitively, v_{post} is one of v_{pre}'s neighbors whose direction change from v_{pre} is closest to s_i.

In summary, the updated route after we obtain at ith turn (α_i, s_i) can be expressed as

$$R_i = R_k \parallel (v^1, \ldots v^k, v_{post}) \tag{4.30}$$

where v^i and k are derived from equation (4.27), $|\cdot|$ represents concatenation of two sequences.

Chapter 5: Evaluation

In this section, we present the details of our experimental results. We firstly evaluate the performance of the orientation estimation. Then we investigate the accuracy of our turn detection algorithm and the routes inference result. Since the results show different performance when the smartphone is placed in different areas of our body, all results will include the cases when the smartphone is held in the user's hand (hand mode) and when the smartphone is in the user's backpack (backpack mode). The device we are using is a Google pixel 3 and the IMU data is sampled at 100 HZ.

5.1 Orientation Estimation

We evaluate the accuracy of the walking direction estimation in real walking scenarios. To assess the estimation errors, we get the ground truth of the true angles from the map. In our cases, all turns in the map, either left or right, are 90° ($\frac{\pi}{2}$ rad), and our estimation of the turn angles is computed as the difference between the mean of yaw angles in the road segments after and before the intersection (y-axis of the red line in Figure 4.4). The error is the absolute value of the difference between our estimation and true turn angles.

Position	No. Turns	Mean	Std dev
Backpack	14	4.01°	3.87°
Hand	14	39.10°	27.05°

Table 5.1: The position of the phone when tested, and the mean and standard deviation of the estimation error.

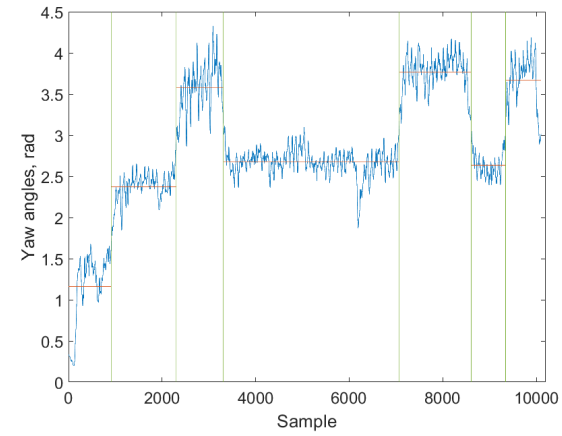
One observation is that the estimation gives a much better performance when the smartphone is in user's backpack than in user's hand. The performance difference can be explained by the swing of user's hand when the user is walking. When the user hold the phone, the motion of swing is captured by the IMU sensors and introduce additional noise to the estimation result. On the other hand, when the smartphone is in user's backpack, even some up-and-down bounce from the walking behaviors may affect the estimation results, but most of them can be corrected by the complementary filter we implemented which will lead to more stable estimation result.

5.2 Route inference

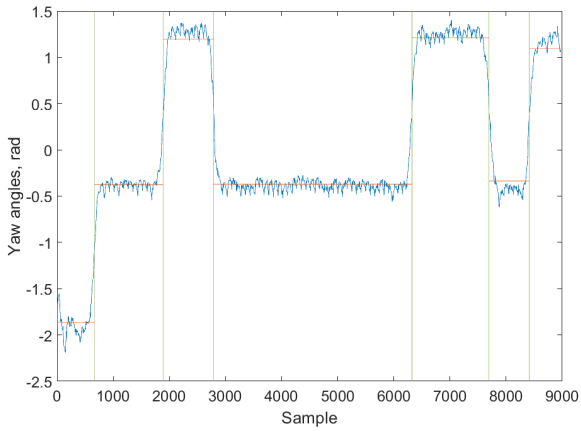
We first present the result of the turn detection algorithm. We collect the IMU readings from three different routes and apply the turn detection algorithm, we get 100 percent accuracy in the number of the turns in the walking routes. Figure 5.1 include the results of the turn detection for one example route in both hand mode and backpack mode.

The turn detection algorithm is robust to the noise in the orientation estimation especially in hand mode. In Figure 5.1(a), the yaw angle signal is kind of blurred, but we can still detect the turns correctly even with large volumes of the noise.

The last part is the route inference, we will try to distinguish two similar paths which are almost same except one road segment, the walking paths and the corresponding yaw rotations are presented in figure 5.2. Since the hand mode is considered more challenging for route inference compared with the backpack mode, we will only give the result of the hand mode. From figure 5.2, these two routes both have four turns and the only difference between them is the distance from the third turn and fourth turn. The difference is reflected in the corresponding yaw angles segmentation. We can see both yaw angles signals are partitioned into five parts, which implies 4 turns in the walking routes. However, the fourth



(a)



(b)

Figure 5.1: Results of the turn detection algorithm on the same route, (a) hand mode (b) backpack mode

part in figure 5.2(c) is clearly shorter than that in figure 5.2(d), when we convert the length of this segment into number of steps, we have the user has walked 10 steps for the fourth part in the first route and 28 steps for the same part of the second route. The difference in the number of steps will detected by our route inference algorithm to distinguish different routes.

However, the performance of the route inference highly depends on the setting of the indoor map. If the indoor map only contains left and right turn of 90 degrees (the one we

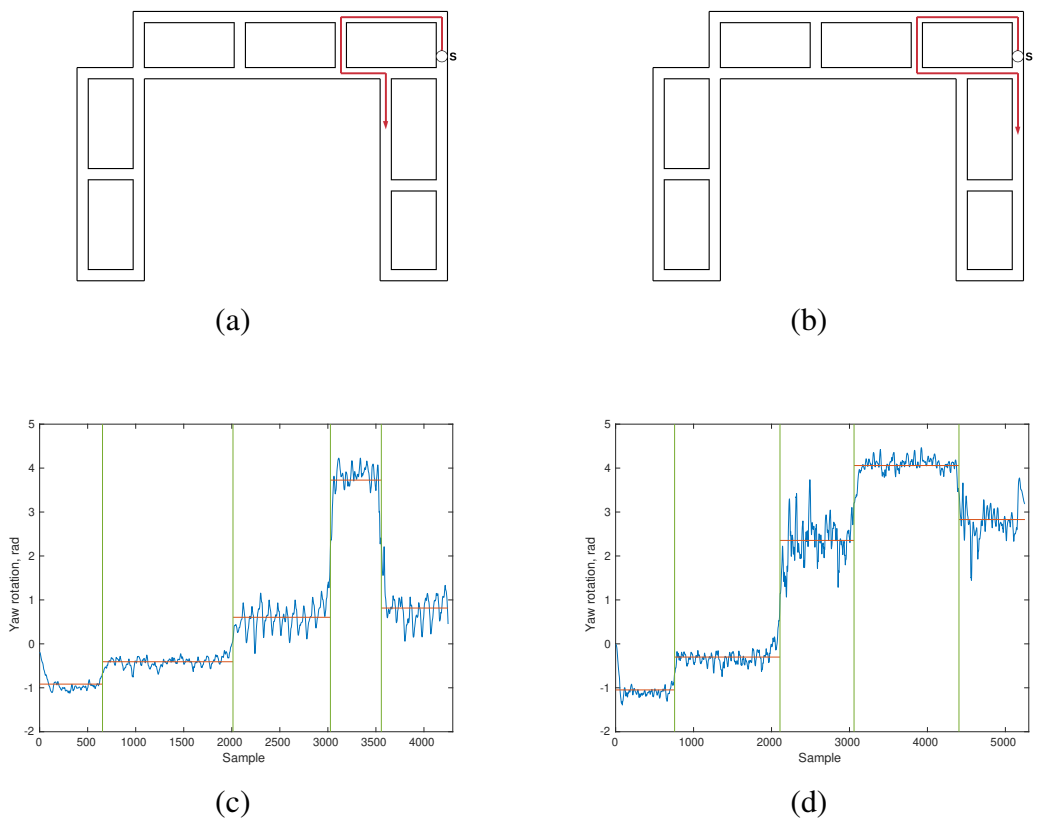


Figure 5.2: Comparison of the yaw rotations for two similar routes. (a) and (b) are the experimental walking routes, and (c) and (d) are the estimated yaw rotations of (a) and (b) respectively. The only difference is that the distance between the fourth turn and third turn in (b) is longer than that in (a). This difference is reflected in the segmentation of the yaw angles, where the fourth segment in (c) is much longer than that in (d)

are using), then the route inference algorithm can still correctly distinguish different road segments connected to the intersection, even with high volumes noise in the orientation estimation, but the performance will degrade if the indoor environment gets more complicated, for example, multiple road segments connect to the same intersection and they are close to each other. In this case, the route inference algorithm may not be able to form the right path and more accurate orientation estimation methods are needed.

Chapter 6: Conclusion

IMU PDR systems are commonly used for indoor navigation or positioning. However, the drift in the output grows over time if the estimation is not corrected frequently. To overcome the shortcoming of estimation drift, we propose a map-aided system to infer the user's walking routes, which is robust to the errors in estimating the heading directions. We have shown that our route inference algorithm can still give the correct walking path with the errors around 30° in the heading direction. This will allow more flexibility for the application of the algorithm when the smartphone is located on different areas of the user's body. We also discuss the trade-off between the complexity of the indoor environment and requirement for accurate head direction estimation. As the indoor map gets more complicated, the algorithm will ask for more accurate head direction estimation to give the correct walking path.

6.1 Future Work

There are two directions to extend this work. The first one is to relax the condition of the knowledge of the initial position and direction. Provided with the turn sequence and the number of steps in each segment, we can devise a search algorithm such that the initial position can derived as long as we have enough IMU data. This is possible because the sequence of turn and step pairs is unique for each route, if one or two turns can be identified in the walking routes, then we can obtain the initial position by reversing the process of map

algorithm from the known turns all the way back to the beginning of the signals. Another limitation of the route inference algorithm is we have to guarantee that the search of the next road segment have to give the correct result at every step, otherwise it is impossible to recover the correct routes. This guarantee can be satisfied when the indoor map is not complicated and the walking behaviour is ”predictable”, but as the indoor environment gets more complex, we would expect some errors during our map matching phase. To add some tolerance of error to our algorithm, especially in map matching phase, we will maintain a list of all possible routes instead of only one route, when deciding next road segment, we first compute the probability for the next segments for all candidates, in our approach, we only select the one with highest probability, one possible extension is to set a threshold and add all candidates with probability higher than the threshold as the possible next segments. Some of them may be removed during further analysis, the algorithm outputs the routes which are still in the list till the end of the experiment.

Bibliography

[1] Anind K Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7, 2001.

[2] Chuanhua Lu, Hideaki Uchiyama, Diego Thomas, Atsushi Shimada, and Rin-ichiro Taniguchi. Indoor positioning system based on chest-mounted imu. *Sensors*, 19(2):420, 2019.

[3] Robert K Harle and Andy Hopper. Deploying and evaluating a location-aware system. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 219–232, 2005.

[4] Deepak Vasisht, Swarun Kumar, and Dina Katabi. Decimeter-level localization with a single wifi access point. In *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, pages 165–178, 2016.

[5] Kaikai Liu, Xinxin Liu, and Xiaolin Li. Guoguo: Enabling fine-grained indoor localization via smartphone. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 235–248, 2013.

[6] Swarun Kumar, Stephanie Gil, Dina Katabi, and Daniela Rus. Accurate indoor localization with zero start-up cost. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 483–494, 2014.

[7] Jie Xiong and Kyle Jamieson. Arraytrack: A fine-grained indoor location system. In *Presented as part of the 10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*, pages 71–84, 2013.

[8] Jon Gjengset, Jie Xiong, Graeme McPhillips, and Kyle Jamieson. Phaser: Enabling phased array signal processing on commodity wifi access points. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 153–164, 2014.

[9] İsmail Güvenc. *Enhancements to RSS based indoor tracking systems using Kalman filters*. PhD thesis, University of New Mexico, 2003.

[10] Eladio Martin, Oriol Vinyals, Gerald Friedland, and Ruzena Bajcsy. Precise indoor localization using smart phones. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 787–790, 2010.

[11] Fuqiang Gu, Kourosh Khoshelham, Chunyang Yu, and Jianga Shang. Accurate step length estimation for pedestrian dead reckoning localization using stacked autoencoders. *IEEE Transactions on Instrumentation and Measurement*, 68(8):2705–2713, 2018.

[12] Hongyu Zhao, Luyao Zhang, Sen Qiu, Zhelong Wang, Ning Yang, and Jian Xu. Pedestrian dead reckoning using pocket-worn smartphone. *IEEE Access*, 7:91063–91073, 2019.

[13] Surat Kwanmuang. *Filtering and Tracking for Pedestrian Dead-Reckoning System.* PhD thesis, 2015.

[14] Johann Borenstein, Lauro Ojeda, and Surat Kwanmuang. Heuristic reduction of gyro drift for personnel tracking systems. *The Journal of navigation*, 62(1):41–58, 2009.

[15] Sarfraz Nawaz and Cecilia Mascolo. Mining users' significant driving routes with low-power sensors. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, pages 236–250, 2014.

[16] Sashank Narain, Triet D Vo-Huu, Kenneth Block, and Guevara Noubir. Inferring user routes and locations using zero-permission mobile sensors. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 397–413. IEEE, 2016.

[17] Yan Michalevsky, Aaron Schulman, Gunaa Arumugam Veerapandian, Dan Boneh, and Gabi Nakibly. Powerspy: Location tracking using mobile device power analysis. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 785–800, 2015.

[18] Rebecca Killick, Paul Fearnhead, and Idris A Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012.

[19] Steven M LaValle, Anna Yershova, Max Katsev, and Michael Antonov. Head tracking for the oculus rift. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 187–194. IEEE, 2014.

[20] Jose-Luis Blanco. A tutorial on se (3) transformation parameterizations and on-manifold optimization. *University of Malaga, Tech. Rep*, 3, 2010.

[21] David Titterton, John L Weston, and John Weston. *Strapdown inertial navigation technology*, volume 17. IET, 2004.

[22] Roberto G Valenti, Ivan Dryanovski, and Jizhong Xiao. Keeping a good attitude: A quaternion-based orientation filter for imus and margs. *Sensors*, 15(8):19302–19330, 2015.

[23] M De Franceschi and D Zardi. Evaluation of cut-off frequency and correction of filter-induced phase lag and attenuation in eddy covariance analysis of turbulence data. *Boundary-layer meteorology*, 108(2):289–303, 2003.

[24] Jay Prakash, Zhijian Yang, Yu-Lin Wei, and Romit Roy Choudhury. Stear: Robust step counting from earables. In *Proceedings of the 1st International Workshop on Earable Computing*, pages 36–41, 2019.