

國立交通大學

資訊科學與工程研究所

碩士論文

應用於 2.5D 環境以粒子過濾演算法與感測
元件輔助的無線定位系統

Wireless Location Tracking by a Sensor-Enhanced
Particle Filter in 2.5D Buildings

研究生：朱瑞浩

指導教授：曾煜棋 教授、易志偉 教授

中華民國 九十八年六月

應用於 2.5D 環境以粒子過濾演算法與感測元件輔助的無線定位系統
Wireless Location Tracking by a Sensor-Enhanced Particle Filter in 2.5D Buildings

研究生：朱瑞浩

Student：Jui-Hao Chu

指導教授：曾煜棋、易志偉

Advisor：Yu-Chee Tseng、

Chih-Wei Yi

國立交通大學
資訊科學與工程研究所
碩士論文



Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年六月

應用於 2.5D 環境以粒子過濾演算法與感測元件 輔助的無線定位系統

學生：朱瑞浩

指導教授：曾煜棋 教授
易志偉 教授

國立交通大學資訊科學與工程學系研究所

摘 要

在室外空曠環境，GPS 具有不錯的定位結果，目前，如何得到使用者在室內環境的位置，受到廣泛的討論和研究，其中尤其以基於無線電訊號強度的樣本比對演算法具有最好的定位結果，但是其最大的缺點在於，無線電訊號先天上容易受到環境影響，導致訊號強度飄移的問題，造成定位結果產生誤差，同時，過去的研究，鮮少討論多層樓等室內立體定位環境 (2.5D 環境)，因此，在此篇，我們為無線電訊號強度樣本比對演算法提出了應用於 2.5D 定位環境的 SEPF (sensor-enhanced particle filter) 模型，透過慣性感測元件 (IMU sensors) 感測使用者的移動軌跡，藉以調整粒子 (particles) 的位置分佈和權重值 (weight)。在此篇中，我們會介紹如何建構室內定位環境的 2.5D 模型，並且在此模型中，粒子的實際取樣 (sampling) 與重新取樣 (re-sampling) 的實作方式，以及藉由分析慣性感測元件的感測值來估測使用者目前的移動行為，例如行走於路面上或樓梯、搭乘電梯等，透過慣性感測元件來克服無線電訊號強度飄移的問題。最後利用我們開發的定位系統雛形，驗證我們的系統效能。

關鍵字：慣性感測元件，定位系統，粒子演算法，遍佈計算，感測網路。

Wireless Location Tracking by a Sensor-Enhanced Particle Filter in 2.5D Building

Student : Jui-Hao Chu

Advisors : Dr. Yu-Chee Tseng
Dr. Chih-Wei Yi

Institute of Computer Science and Engineering
National Chiao Tung University

ABSTRACT

For outdoor localization, GPS already provides a satisfactory solution. For indoor localization, however, a globally usable solution is still missing. One promising direction is the pattern-matching solution that relies on RF signals from existing network infrastructures. One major drawback of such systems is the signal-drifting problem, which is an inherent physical constraint. Also, most works only consider single-floor buildings. However, buildings normally have multiple floors (i.e., of 2.5 dimensions). This paper proposes a SEPF (sensor-enhanced particle filter) model for RF-based pattern-matching localization in a 2.5-D building. IMU sensors are adopted to capture human mobility, while particles reflect the belief on where the user is located. Our framework addresses the following important issues. First, our 2.5-D building model considers multiple floors connected by stairs and elevators. Second, we show how particles should be sampled/re-sampled in a 2.5-D building to reflect change of belief. Third, IMU sensor inputs are exploited to conquer the signal-drifting problem and to predict user's behaviors (walking on grounds/stairs and taking elevators). A prototype has been developed and intensively tested to verify the model.

Keywords: IMU (inertial measurement unit), location tracking, particle filter, pervasive computing, sensor network.

誌 謝

首先，誠摯的感謝我的兩位指導教授，曾煜棋以及易志偉教授，在這兩年持續的指導和鼓勵，在這段時間中碰到許多困難，但是只要多與老師們請教，都可以迎刃而解，所以非常感謝兩位老師，幫助我得以完成這篇論文。

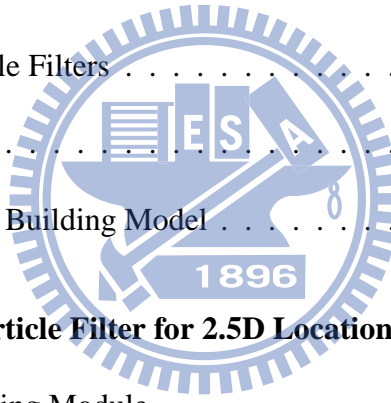
接著，我要感謝我們小組的學長，羅榮鐘、郭聖博兩位學長，在研究上，給我許多建議和指導。並且我也要感謝實驗室所有的同學，在這段時間的相互關心和幫助。

最後，我要感謝我的家人，不僅栽培我，並且在這段時間給我許多鼓勵和支持。

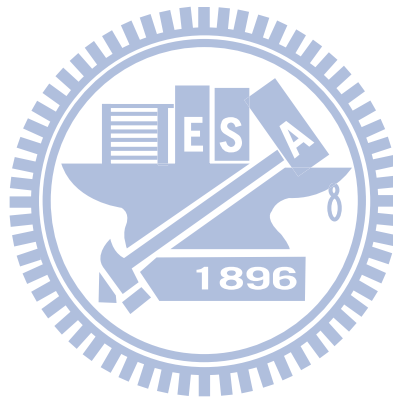


Contents

1	Introduction	1
2	Preliminaries	5
2.1	Bayes and Particle Filters	5
2.2	IMU Sensors	6
2.3	2.5-Dimensional Building Model	10
3	Sensor-Enhanced Particle Filter for 2.5D Location Tracking	14
3.1	Behavior-Predicting Module	16
3.1.1	Horizontal Motion Detection	16
3.1.2	Stair Motion Detection	20
3.1.3	Elevator Motion Detection	20
3.2	Particle-Sampling Module	21
3.3	Particle Filtering Module	25
3.3.1	The Speed Filtering	25



3.3.2	The Passing Wall Filtering	26
3.4	Particle Weighting Module	27
3.5	Particle ReSampling Module	28
4	Simulation Environment and Results	30
4.1	Singal Path Loss Models	31
4.2	The Impacts of System Parameters	34
5	System Implementation	38
6	Conclusions	42

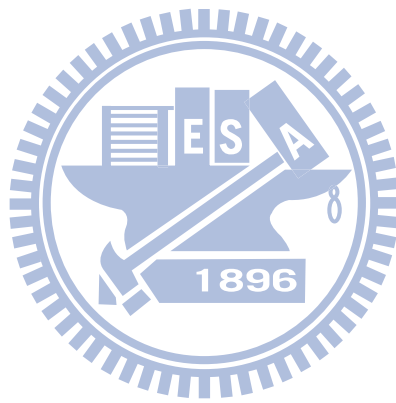


List of Figures

2.1	mounting sensor device on waist or hanging on chest.	7
2.2	from moving state to halting state.	8
2.3	walking on grounds.	8
2.4	walking on a circle.	8
2.5	going up/down stairs.	9
2.6	going up/down an elevator.	9
2.7	2.5-D Building Model.	11
2.8	The Subgraphs of 2.5-D Building Model.	12
3.1	The SEPF architecture.	15
3.2	An illustration of the horizontal and vertical motion prediction. . .	17
4.1	An illustration of the simulation environment.	31
4.2	An impact of the APs	35
4.3	An impact of noise deviation with training grain size.	35
4.4	An impact of DOI	36

4.5 An impact of *WAF*. 36

4.6 An impact of estimating dynamic of *IMU* Sensors. 37



Chapter 1

Introduction

Location-based services LBSs have been regarded as a killer application in mobile networks. A key factor to its success is the location estimation accuracy. For outdoor localization, GPS already provides a satisfactory solution. For indoor localization, however, a globally acceptable solution is still missing. Many indoor localization technologies have been proposed, such as infrared-based [4], ultrasonic-based [10], and RF-based systems [2]. Generally, localization models can be classified as AoA-based [8], ToA-based [1], TDoA-based [11], and fingerprint-based [2][7][9].

In this work, we are interested in the *pattern-matching* localization method, such as RADAR [2]. This method does not rely on calculating signal fading in an environment. Instead, it relies on a *training phase* to collect the *radio signal strength (RSS)* patterns at a set of training locations from pre-deployed beacons

into a database (called *radio map*). These beacons can be existing infrastructures, such as IEEE 802.11 access points, GSM base stations, or a sensor network. Then, during the *positioning phase*, an object to be localized can collect its current RSS pattern and compare it against the radio map to predict its location.

The pattern-matching solutions have three main drawbacks. First, its training process is very labor-intensive, especially in a large-scale field. Second, it suffers from the signal-drifting problem; the radio maps measured earlier may significantly deviate from the current ones. Third, most works are limited to 2-D sensing fields (e.g., a single-floor building). However, buildings normally have multiple floors. The first issue has been studied in [13], the second in [17][16], but the last one has been rarely addressed.

This paper proposes a *SEPF* (*sensor-enhanced particle filter*) model for RF-based pattern-matching localization in a 2.5-D building. IMU sensors are adopted to capture human mobility patterns, while particles help reflect our belief on where the user might be located. Our framework addresses the following important issues. First, our 2.5-D building model considers multiple floors connected by stairs and elevators. Second, we show how particles should be sampled/re-sampled in a 2.5-D building based on sensor inputs to reflect change of belief. Third, IMU sensor inputs are exploited to conquer the signal-drifting problem and to predict human's main indoor behaviors (such as halting, walking on grounds, going up/down stairs, and taking elevators). A prototype has been developed and intensively

tested to verify our model.

Particle filters are sample-based implementation of Bayesian filters using a set of samples, i.e., particles, to reflect the probability densities of our belief [6]. In the scope of indoor localization, it estimates locations by recursively calculating current probability distributions based on measurements of current observation. The key components of Bayesian filter are observation, prediction, and history models. The observations could be from sensors or mathematical models. Theoretically, it can be applied to most positioning algorithms as long as we have some sort of prediction on user mobility. For example, a tracking system can exploit mobility history to conjecture user's trajectory [3]. For pattern-matching solutions [14], RSSI is the only observation used to represent its belief about a dynamic system at time t as a probability distribution over the state space. The parameters and noise components of particle filters can be estimated from training data or tuned manually.

Hence, we propose a SEPF architecture with particle filter to obtain the actual mobility model. And then, using mobility model to predict the propagation of particles. Traditionally, the real time mobility model can not directly get from users, but we presently utilize the sensor to achieve it. According to our study, the observation model also conforms to the model of the fingerprint base positioning algorithm [5]. The philosophy of particle filter conforms to the tracking positioning algorithm by consulting history. And, the real time mobility model is

appropriate for adding the prediction model of it. Based on the mobility model, we can get better in positioning, because the particles are distributed around the true user's positioning location [15]. Therefore, we propose a positioning algorithm based on particle filter involving changeable mobility model.

The rest of this paper is organized as follows. Chapter 2 gives some preliminaries. Chapter 3 describes our SEPF architecture. Performance figures are presented in Chapter 4. Chapter 5 introduces our prototyping details and experiment results. Finally, Chapter 6 concludes this paper.



Chapter 2

Preliminaries

2.1 Bayes and Particle Filters

Bayes filters probabilistically estimate a dynamic system's state from observations that could be disturbed by noise. In pattern-matching localization, the state could be a person's location and the observations are RSS patterns. Bayes filters represent the state at time t by a random variables x_t . It establishes a probability distribution over x_t , called belief $Bel(x_t)$. The goal is to sequentially estimate such beliefs over the state and time spaces. Specifically, let the sequence of time-indexed observations be z_1, z_2, \dots, z_t . $Bel(x_t)$ is defined by the posterior density of state x_t conditioned on all previous observations:

$$Bel(x_t) = p(x_t | z_1, z_2, \dots, z_t). \quad (2.1)$$

The belief answers the question: “What is the probability that the person is at location x_t if the sequence of observations is z_1, z_2, \dots, z_t ?” In general, the complexity to compute $Bel(x_t)$ grows exponentially over time. To make the computation tractable, Bayes filters assume that the dynamic system is Markovian in that the current state x_t contains all relevant information. So states before x_{t-1} provide no additional information. In our localization example, this means that we only need to work on the relation between x_{t-1} and x_t .

To realize different density functions of x_t , particle filters represent beliefs by a set of samples, or *particles*:

$$Bel(x_t) = S_t = \{ \langle x_t^{(i)}, w_t^{(i)} \rangle \mid i = 1, \dots, n \}, \quad (2.2)$$

where, each $x_t^{(i)}$ is a state and $w_t^{(i)}$ is its weight. These $w_t^{(i)}$ s sum up to one. Particle filters realize Bayes filters by a sequence of sampling, weighting, and resampling procedures. Its key advantage is the capability to represent arbitrary density functions, even in non-Gaussian, non-linear dynamic systems. It allows us to focus on resources (particles) in state spaces with high probabilities.

2.2 IMU Sensors

We are interested in using IMU (inertial measurement unit) sensors to capture typical human mobility patterns inside a building. We consider four main mobility patterns: halting, walking on grounds, going up/down stairs, and taking elevators.



Figure 2.1: mounting sensor device on waist or hanging on chest.

Mobility patterns will be tracked by a triaxial accelerometer (g-sensor), which can report 3D accelerations, and an electronic compass, which can report the angle relative to north.

The g-sensor combines three angular rate gyros with three orthogonal DC accelerometers and three orthogonal magnetometers to output its orientation in dynamic and static environments. The outputs of them are rate of rotation, quantity of gravity and quantity of gauss. The size of it is 64 millimeter by 90 millimeter and 25 millimeter. And, the gravity is 75 grams with enclosure. Based on the official specifications, the accuracy of the orientation is about 0.5 degree for static test conditions and 2.0 degree for dynamic test conditions.

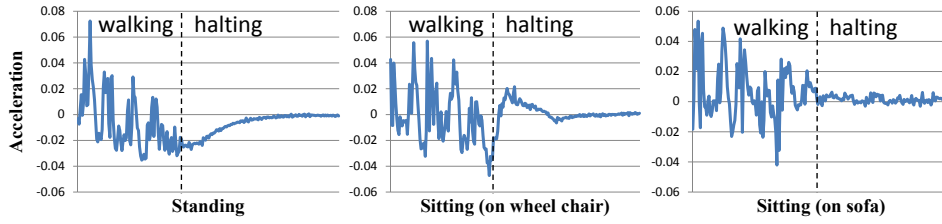


Figure 2.2: from moving state to halting state.

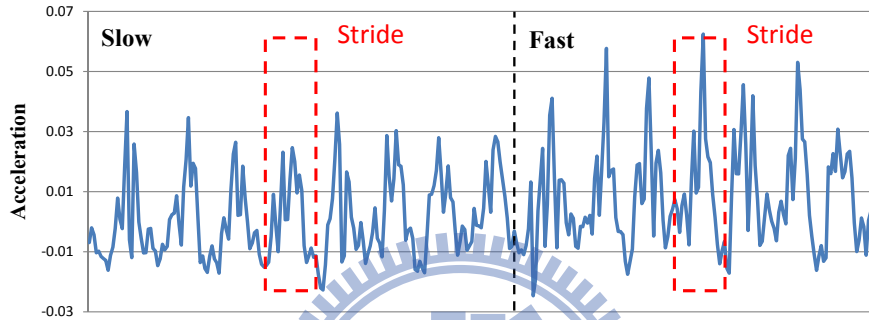


Figure 2.3: walking on grounds.

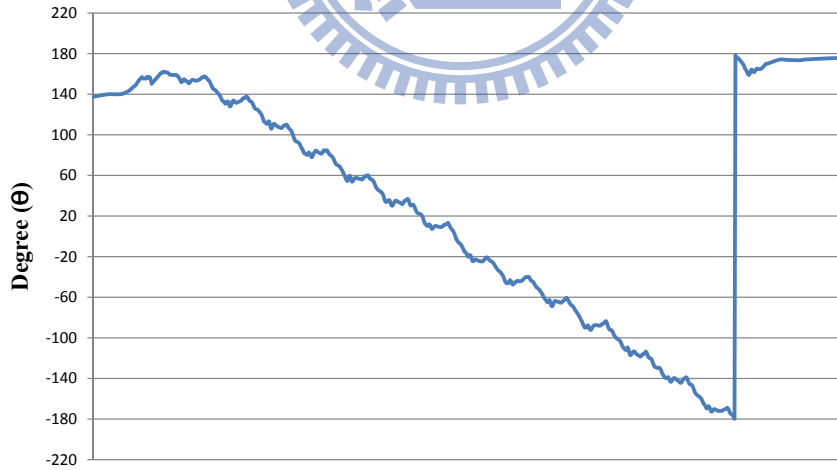


Figure 2.4: walking on a circle.

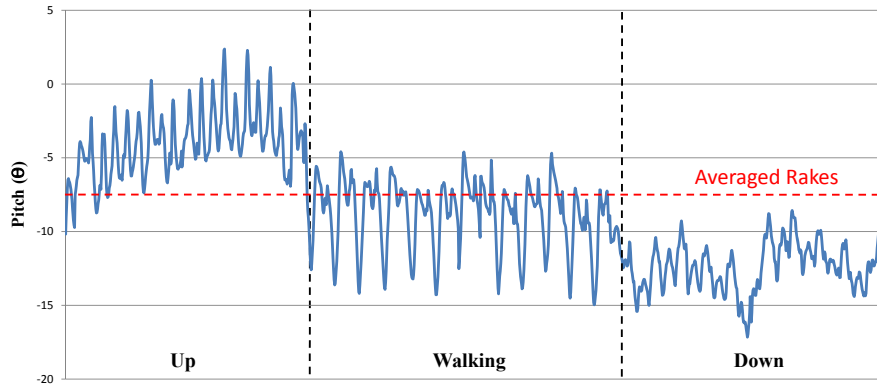


Figure 2.5: going up/down stairs.

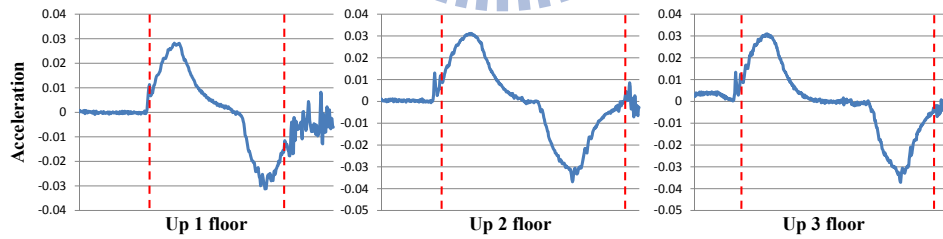
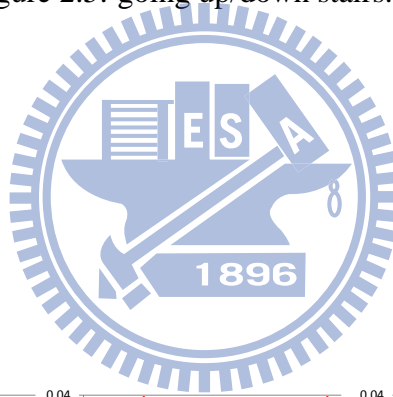


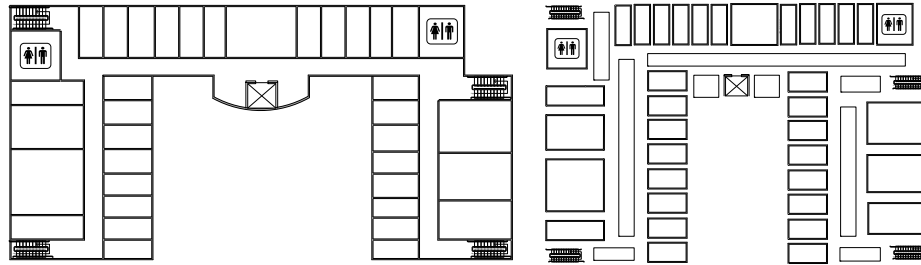
Figure 2.6: going up/down an elevator.

Figure 2.1 shows two ways to mount sensors devices on a human body in our experiment. Figure 2.2 shows the outputs of a g-sensor when a person transits from a moving state to a halting state. Figure 2.3 shows the outputs of a g-sensor when a person walks on grounds. Figure 2.4 shows the outputs from an electronic compass when a person walks in a circle of radius 2m. Figure 2.5 shows the outputs of a g-sensor when a person goes up/down stairs. Figure 2.6 shows the outputs of a g-sensor when a person goes up/down an elevator.

As can be seen, these patterns all have their special features that can be easily recognized.

2.3 2.5-Dimensional Building Model

We now develop a graph-like model to represent a 2.5-D building. The representation facilitates us to describe how particles flow around a building in the yet-to-be-presented indoor localization scheme. We are given the floor plans of a 2.5-D building. Each floor may have rooms, partitions, hallways, etc. Floor is connected by stairs and elevators. The graph is denoted by $G = (V, E)$. A vertex in V could be one partition unit on a floor, a stairway, or an elevator. An edge in E connects two vertices in V by specifying the passable part between them. Figure 2.7(a) shows a floor plan of the Engineering Building III at NCTU, where our experiments are conducted.



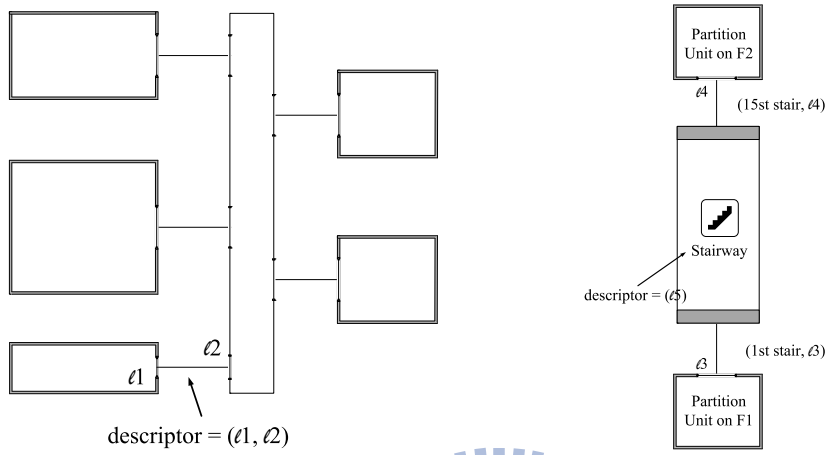
(a) 3f, Engineering Building III, NCTU.

(b) Partitioning of a floor plan.

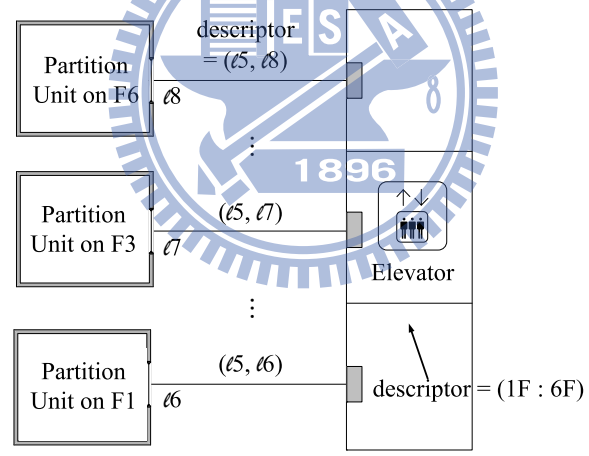
Figure 2.7: 2.5-D Building Model.

Given a building's floor plan, we first deal with each floor by dividing it into multiple *partition units*. Each partition unit should be a convex polygon. Note that a convex polygon has the property that the line segment connecting any two points inside must fall inside the polygon itself; from particle filters' prospect, this means that it allows a particle to move in a straight line between these two points. Figure 2.7(b) shows an example. Between two partition units, there are *passable* and *impassable* parts. Each passable part is represented by an edge in E together with a descriptor to specify the line segment of the two corresponding polygons constituting the passable part. From particle filters' prospect, these passable parts are pathways for particles to flow around. Excluding passable parts, the rests are impassable parts, which particles are not allowed to cross over. Figure 2.8(a) shows how we construct partition units. Note that the labels on edges are their descriptors (for passable parts).

Each stairway is represented by one vertex in V and it has a descriptor to



(a) Floor subgraph. (b) Stair subgraph.

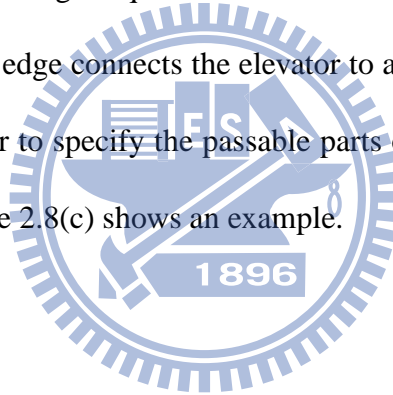


(c) Elevator subgraph.

Figure 2.8: The Subgraphs of 2.5-D Building Model.

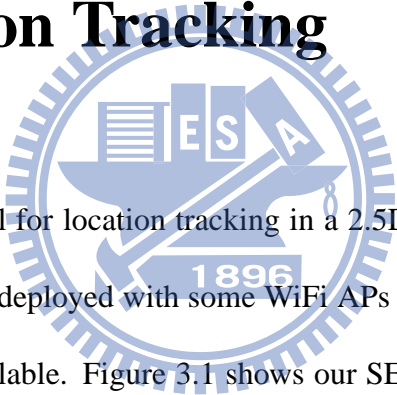
specify the number of stairs inside. A stairway connects two floors and we use two edges in E to represent such relationships. Each edge connects the stairway to a partition unit of a floor and it two descriptors: (i) the stair that the partition unit is connected and (ii) the passable part in the partition unit. Figure 2.8(b) shows how we abstract such concepts.

Each elevator is also represented by a vertex and it has two descriptors: (i) the range of floors that it moves to and (ii) the polygon of its ground part. Note that for (i), it is not necessary for the elevator to be stop at each floor within this range. we use the same number of edges equal to the number of floors that the elevator stops to denote this. Each edge connects the elevator to a partition unit of a floor and it also has a descriptor to specify the passable parts of the partition unit and its own ground part. Figure 2.8(c) shows an example.



Chapter 3

Sensor-Enhanced Particle Filter for 2.5D Location Tracking



We propose a SEPF model for location tracking in a 2.5D building. We assume that the building has been deployed with some WiFi APs and a pattern-matching localization system is available. Figure 3.1 shows our SEPF work flow. It takes IMU sensor measurements, RSS patterns, and floor planes of a 2.5D environment as inputs to predict users' locations. Each user has to carry a WiFi interface, tri-axial accelerometer, and an electronic compass. The wireless interface can collect the RSSIs of its surrounding wireless beacons and the IMU sensors measurements and report to the PF to compute the user's location. Users can only move on floor surfaces or go up/down stairs/elevators. The 2.5D building model defined in Sec. 2.3 is used.

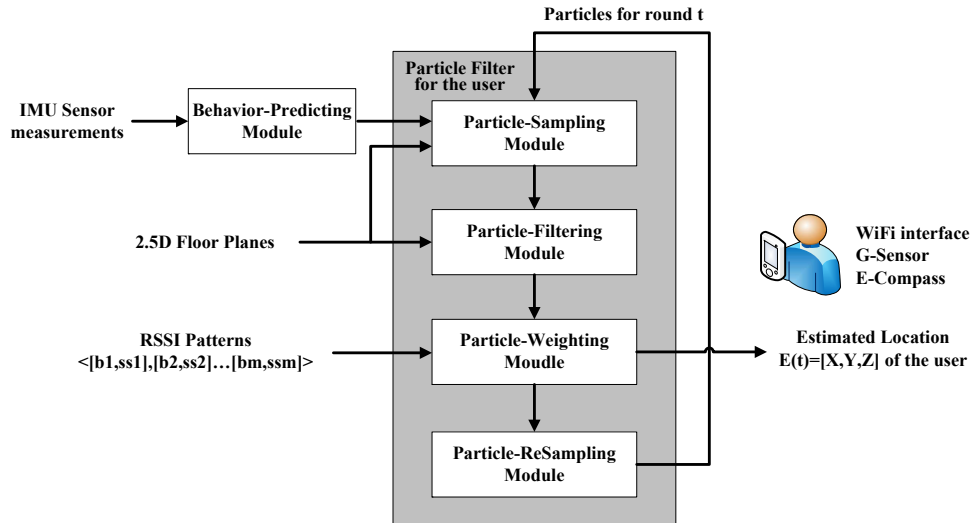


Figure 3.1: The SEPF architecture.

Like typical PFs, our architecture also has *particle-sampling*, *particle-weighting*, and *particle-resampling* modules. In addition, we add two new modules: behavior-predicting and particle-filtering modules. The behavior-predicting module collects the IMU sensor outputs to predict the user's current mobility pattern. The prediction, together with the floor planes, helps the particle-sampling module to propagate particles. Then those particles with low credibility will be filtered out by the particle-filtering module according to the floor planes. Using the RSSIs, the particle-weighting module will assign weights to particles. It also produces the estimated location of the user. Finally, the particle-resampling module re-generates particles for the next round.

3.1 Behavior-Predicting Module

Based on floor plans and IMU sensor measurements, the behavior-predicting module estimates the user’s current motions. This improves over traditional PFs, which usually adopt a random model in sampling particles. This module generates three outputs: $H(t)$, $S(t)$, and $E(t)$. $H(t) = \{(t_i, d_i), i = 1, \dots, \infty\}$ is a series of horizontal motion vectors, t_i is the timestamp and d_i is the displacement vector on the xy-plane measured between time interval $(t_{i-1}, t_i]$. $S(t) = \{(t_i, s_i), i = 1, \dots, \infty\}$ is a series of stair motions, where t_i is a timestamp and s_i is the estimated number of stairs that the user has taken during time interval $(t_{i-1}, t_i]$ (we use 0 to mean “no stair motion”, and a negative/positive integer to mean how many stairs down/up). $E(t) = \{(t_i, e_i), i = 1, \dots, \infty\}$ is a series of elevator motions, where t_i is a timestamp and e_i is the estimated number of floors that the user has taken during time interval $(t_{i-1}, t_i]$ (we use 0 to mean “no elevator motion”, and a negative/positive integer to mean how much floor down/up). Below, we discuss how these series are computed in our model.

3.1.1 Horizontal Motion Detection

The horizontal motion detection module is pedometer-based. A pedestrian’s horizontal motion is composed of a series of *stepping* events. In other words, the displacement vector d_i is corresponding to a stride. The readings of the accelerom-

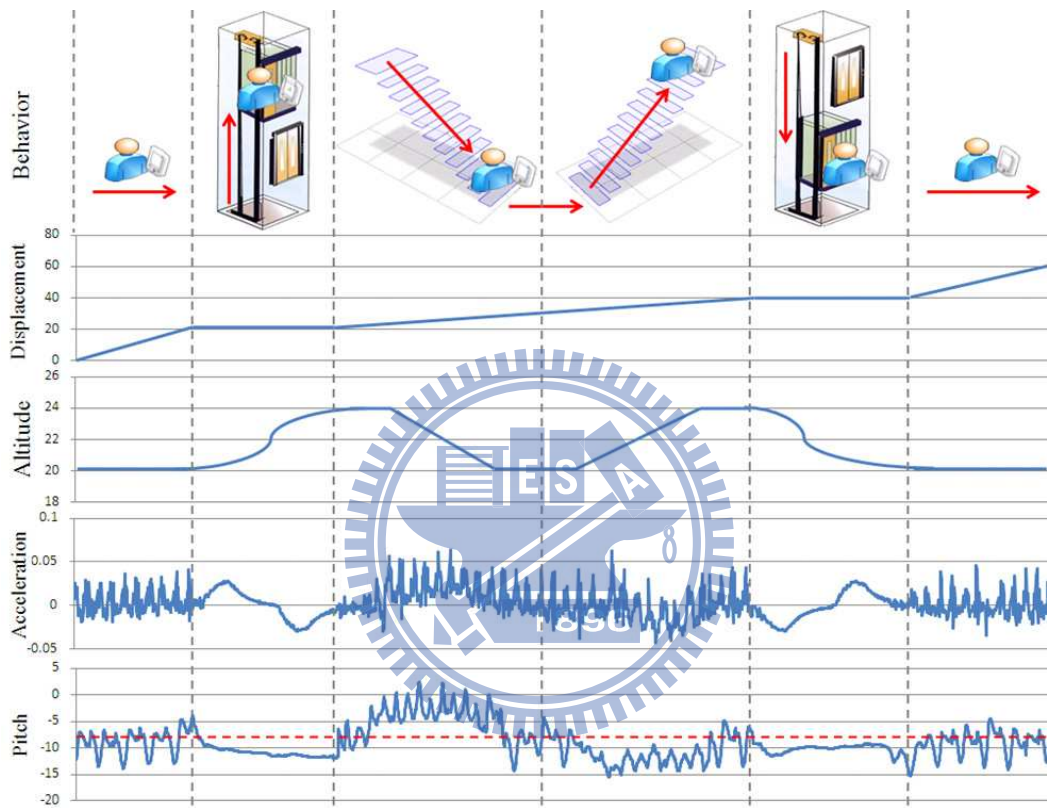


Figure 3.2: An illustration of the horizontal and vertical motion prediction.

eter is used to detect stepping, estimate each stride length, and also decide walking directions by the helping of the magnetometer.

Stepping Detection: The readings of the accelerometer are decomposed into two vertical components and a horizontal components. We observed that each step incurs a pulse waveform in the vertical components. A pattern matching technique is used to detect pulse for counting steps. To improve the accuracy, the amplitude and duration of each pulse and the magnitude of horizontal components are used to filter out false detections. In average, the pedometer has an accuracy around 96%.

Stride Length: After a stride is detected, two features A and X are extracted from the pulse where A is the sum of the magnitude of the left slope and right slope and X is the norm of the first component of the discrete Fourier transform of the pulse. We use quadratic regression to estimate the stride length. In other words, the stride length is estimated by the following formula

$$aX^2 + bXA + cA^2 + dX + eA + f. \quad (3.1)$$

The constants a, b, c, d, e, f are obtained by the least square method.

Walking Direction: The horizontal components tend to parallel with the walking direction. Therefore, we classify horizontal components into two groups, forward vectors and backward vectors. Forward vectors are roughly in the walking direction and backward vectors are roughly in the opposite directly. To determine

the walking direction, we sum forward vectors with the inverse of backward vectors. The sum vector is used to represent the walking direction. Then, to with the readings of the magnetometer, we can know the absolute walking direction (in the earth frame).

Figure 3.2 shows an example, where an user moves over around the indoor environment. The user starts at first floor and moves up to 3th floor by elevator. Then, user goes down to the first floor and returns to 3th floor by stair. In the end, user goes back to the first floor. Our goal is to generate the output $H(t)$, where each d_i is a displacement vector that user has moved from t_{i-1} to t_i . We assume t_i is a periodic event. Between t_{i-1} to t_i , the stride events may occur j times, name as $\vec{d}_{i,j}$. We detect each stride event by capturing the sine wave from acceleration and estimate stride length by integrate each sine wave as $\vec{d}_{i,j}$. The direction θ is represented by the angle measured in counterclockwise to the North Pole. If $\vec{d}_{i,j}$ and θ respectively denote the stride length and direction of a step, the displacement vector can be obtained by the formula

$$d_i = \vec{d}_{i,1} + \vec{d}_{i,2} + \cdots + \vec{d}_{i,j}. \quad (3.2)$$

3.1.2 Stair Motion Detection

The stair motion detection module is also pedometer-based. In our system, the typical value of the stair count s_i is ± 1 . $S(t) = \{(t_i, e_i)\}$ is a discrete-time sequence to represent whether the user has walked up/down stairs and, if so, how many stairs the user has gone. Since the event of one step up/down a stair can only be determined after the completion of the step, the reporting sequence $S(t)$ is always *posterior*.

Specifically, we will periodically look at the sensor measurements and attach a report to $S(t)$ with a period of p . (The recommended p is around 2~4 seconds.) At any point t' when an observation is made, we will merge the output from our step detector and the corresponding rake value. The rake value can be retrieved from the z-axis output of the g-sensor. A report of (t', k) will be attached to the sequence $S(t)$, where k is the number of stair-up/stair-down events that are detected during the interval $(t' - p, t']$ (note that k should include the stair-up/stair-down event, if any, that was not reported at time $t' - p$).

3.1.3 Elevator Motion Detection

Depending on the measurements from the g-sensor, we need generate $E(t) = \{(t_i, e_i)\}$. $E(t)$ is a discrete-time sequence to represent whether the user has taken an elevator and, if so, how many floors he/she has gone up/down. Since the num-

ber of floors that a user has moved can only be determined after the completion of signal changes has been observed and an elevator event normally takes 3 to 20 seconds, the reporting sequence $E(t)$ is sometimes periodical and sometimes aperiodical. For this reason, the report of an elevator event is always *posterior*.

Specifically, we will periodically look at the measurements of the g-sensor in the z -axis with a period of p . (The recommended p is around 1~5 seconds.) Let the output of the g-sensor at the z -axis over time be $g_z(t)$. At any point t' when we make an observation, we will conduct a curve-fitting on $g_z(t)$ for the part right before t' . If no curve in $g_z(t)$ matches with any of the curves representing an elevator event as shown in Figure 2.6, a report of $(t', 0)$ will be attached to the sequence $E(t)$. Otherwise, if a match is found, let t'' be the time point when the elevator event ends in the curve $g_z(t)$. A report of (t'', k) will be attached to the sequence $E(t)$, where k is the number of floors that the user has gone up/down (a positive value means “up” and a negative one means “down”). Also, in the later case, we will adjust the offset of our periodical reports to t'' (i.e., the upcoming reports will be adjusted to $t'' + p, t'' + 2p$, etc.).

3.2 Particle-Sampling Module

The particle-sampling module takes three inputs: (1) the particles from the previous round, (2) the sequences $H(t)$, $S(t)$, and $E(t)$, and (3) the floor plans. This

module will adjust the location of each of these particles. Let t_a be the timestamp of the previous event that was processed and $\{P_i(t_a)\}$ be the set of particles at t_a . Each particle $P_i(t_a)$ is associated with three descriptors: $P_i(t_a).par$, $P_i(t_a).loc$, and $P_i(t_a).wt$. The partition unit and the location in the partition unit $P_i(t_a)$ is located are written as $P_i(t_a).par$ and $P_i(t_a).loc$, respectively. The weight of $P_i(t_a)$ is written as $P_i(t_a).wt$. Given $H(t)$, $I(t)$, and $E(t)$, we retrieve those unprocessed events one-by-one according to their timestamps. We start from the unprocessed event ε with the earliest timestamp. Then we relocate each $P_i(t_a)$ according to the following rules:

- Case 1: If $P_i(t_a).par$ is a partition unit on some floor, there are three cases.

1. If $e \in H(t)$, let $\varepsilon = (t_b, \vec{d})$. We place $\{P_i(t_a)\}$ at the location:

$$P_i(t_b).loc(t) = P_i(t_a).loc(t) + \vec{d} + \vec{R}, \quad (3.3)$$

where $\vec{R} = r \cdot (\cos \theta, \sin \theta)$ is a 2D random vector, where r is a scalar randomly selected from the interval $[0, r_{max}]$ and θ is a angle randomly selected from $[0, 2\pi]$. Here, \vec{d} represents the displacement vector that is detected from IMU sensors. However, to accommodate the existence of noise, \vec{R} is to add some randomness to the new location of $P_i(t_b)$. Note that if $P_i(t_b)$ remains in the same partition unit as $P_i(t_a)$, we let $P_i(t_b).par = P_i(t_a).par$; otherwise, we let $P_i(t_b).par$ be the partition unit on the same floor that contains $P_i(t_b).loc$. One exception,

perhaps due to noise, is when $P_i(t_b)$ falls outside the current floor. In this case, we let $P_i(t_b)$ be located at the point when it first moves out of the boundary of the floor during the above movement.

2. if $\varepsilon \in S(t)$, let $e = (t_b, s)$. We first check if $P_i(t_a).loc$ is nearby any passable part of a stairway. If so, we move this particle into this stairway. Specifically, if $s = +1$ and the distance from $P_i(t_a).loc$ to any passable part of a stairway sw is within δ_{f2s} and the stairway connects the current floor to a higher floor, we let $P_i(t_b).par = sw$ and $P_i(t_b).loc = 1$. Similarly, if $s = -1$ and distance from $P_i(t_a).loc$ to any passable part of a stairway sw is within δ_{f2s} and the stairway connects the current floor to lower floor, we let $P_i(t_b).par = sw$ and $P_i(t_b).loc = -1$. If none of those two cases sustain, $P_i(t_b)$ will remain at the same partition unit and location as $P_i(t_a)$.

3. if $\varepsilon \in E(t)$, let $e = (t_b, e)$. We check if $P_i(t_a).loc$ is nearby any passable part of an elevator. If so, we move this particle into this elevator. Specifically, if $e = "U"$ and the distance from $P_i(t_a).loc$ to any passable part of an elevator ev is within a threshold δ_{f2e} , we let $P_i(t_b).par = ev$ and $P_i(t_b).loc = k^+$, where k is the current floor number and the superscript "+" means that the elevator is going up. Similarly, if $e = "D"$ and the distance from $P_i(t_a).loc$ to any passable

part of an elevator ev is within a threshold δ_{f2e} , we let $P_i(t_b).par = ev$ and $P_i(t_b).loc = k^-$, where k is the current floor number and the superscript “-” means that the elevator is going down. If none of these two cases sustain, $P_i(t_b)$ will remain at the same partition unit and location as $P_i(t_a)$.

- Case 2: If $P_i(t_a).par$ is a stairway, there are two cases.
 1. If the input is H(t) or E(t), we ignore that.
 2. If the input is S(t), the particle $P_i(t).loc$ goes up/down by the estimated number of stairs s_i . Also, a random movement is needed to disturb the particles. If the new $P_i(t).loc$ belongs to the first/last n stairs and its trend is up/down, the particle $P_i(t).obj$ changes to floor and $P_i(t).loc$ goes to the boundary of the floor that connected with the stairs.
- Case 3: If $P_i(t_a).par$ is an elevator, there are two cases.
 1. If the input is H(t), S(t), or , the begin event of E(t) we ignore that.
 2. If the input is the end event of E(t), the particle moves to the floor that belong to the estimated number of s_i . For example, if the input is $E(t) = (t_1, +3)$, actually we may move from 3th floor to 6th floor, $P_i(t).obj$ will change to 4th, 5th, and 6th with a probability distribution then $P_i(t).loc$ goes to the boundary of the floor that connected

with the elevator.

In the above discussion, we did not address how to assign weights to particles. Here we assume that each particle will inherit its previous particle's weight, i.e.,
$$P_i(t_b).wt = P_i(t_a).wt.$$

3.3 Particle Filtering Module

Generally, the sampled particles are weighted directly in weighting module. It only uses the pattern matching scheme to figure out the location having the highest possibility from the particle with maximum weight. Intuitively, the precision will be gained if we took additional information of user into account for decision. Therefore, the particle filtering module utilizes the speed of user and floor plans to filter out the sampled particles that they are incredible.

3.3.1 The Speed Filtering

The speed filtering is used to adjust the coverage of particles. It restricts the particles be spread on a radius of confident size c which's center is the last estimated result with mobility shift. The particles which are outer this range will be filtered. The mobility shift is the displacement vector \vec{d} obtained by the IMU module, and the confident size reflects the confidence of the shifted result. If the shifted result is trustworthy, we can concentrate the particles on it by decreasing confident size.

We utilize the averaged speed of user to evaluate the confident size. It is averaged by referring the past speeds with decreasing weight. Let \vec{v}_t denotes the speed of user and α_t denotes the weighting factor at time t . If k is the number of past speeds to be summed, then

$$c = \frac{\sum_t^{t-k} \vec{v}_t \times \alpha_t}{\sum_t^{t-k} \alpha_t}. \quad (3.4)$$

. For example, k is 5 and the speeds are 5,5,0,3 and 2 from time t to $t - 5$ with weights decreased by 0.2.

$$c = [(5 \times 1) + (5 \times 0.8) + (0 \times 0.6) + (3 \times 0.4) + (2 \times 0.2)]/3. \quad (3.5)$$

Based on the speed filtering, the drifting problems and the errors of sensor measurement can be mitigated. When user has stopped in a while, the coverage of particles is reduced to decrease the variance of estimated results. Contrary, the coverage is enlarged to tolerate the estimating error of sensors, when user is running. The revising for these errors will consign to the weighting module.

3.3.2 The Passing Wall Filtering

This behavior is used to filter out the particles passing the walls that the impossible situation of user. It is judged by the trajectories of particles and floor plane maps. Let L denotes the walls of the maps. Each wall l_i in L has two end-point v_{is} and v_{ie} . If the straight line between locations $P_i(t - 1)$ and $P_i(t)$ of particle i is

pass through $l_i(\overrightarrow{v_{is}v_{ie}})$ where $l_i \in L$, this particle will be filtered. It means that the belief of this particle is incredible. The filtering process has an exceptional case that all of the particles are determined to be incredible. Hence, we propose a trick that it restarts the particle sampling module and increases the constant r of random vector \overrightarrow{R} to enlarge the range of disturbance until at least one particles is reserved.

3.4 Particle Weighting Module

This module assigns each particle in $P(t)$ a weight to reflect the probability that the user is at its location. For each $P_i(t) \in P(t)$, the particle-sampling module already define its location as $P_i(t).loc$. We need to compute the weight $P_i(t).wt$ given that $P_i(t)$ is located at $P_i(t).loc$. This includes two steps. First, we will estimate, form the location database, the RSSI pattern S_{est} at location $P_i(t).loc$. Second, we will compare S_{est} against the currently observed pattern S_{obs} to compute $P_i(t).wt$. We use a likelihood function $Pr(o|l_i)$ to estimate the weight that the probability of receiving o at l_i of particle i as below.

$$Pr(o|l_i) = \prod_{j=1}^m K(ss_j; s\bar{s}_j), \quad (3.6)$$

$$K(ss_i; ss_j) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(ss_i - ss_j)^2}{2\sigma^2}\right). \quad (3.7)$$

The likelihood function is derived by the multiplication of m kernel function to each access point. The kernel function estimate the probability of receiving ss_i from access point i by gaussian distribution with mean ss_j . The $s\bar{s}_j$ is the signal-strength received form access point j of the characteristic at l_i . In the offline phase, the RSSI patterns are collected at each training location. These patterns is averaged by histogram to be the characteristic at this location. The $s\bar{s}_j$ is derived from the training locations which locate nearby l_i by interpolation.

After all the particles are estimated, the weights of them will be normalize as below.

$$w_i^t = \frac{w_i^t}{\sum_{i=1}^n w_i^t}. \quad (3.8)$$

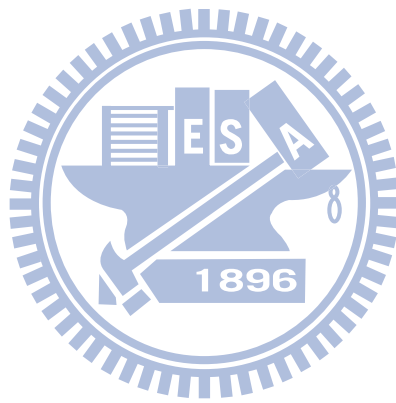
Finally, the localization result at time t is obtained by

$$E(t) = \arg \max Pr(o|l_i). \quad (3.9)$$

3.5 Particle ReSampling Module

This module will take the current set of particles $P(t)$ as input and generate a new set of particles called $P(t + 1)$ for the next round. Note that $P(t + 1)$ may be a multiset. Let \bar{W} be the summation $\sum_{P_i(t) \in P(t)} P_i(t).wt$. Let n be the expected number of particles in the beginning of each round. Then for each $P_i(t) \in P(t)$, we will generate $n \frac{P_i(t).wt}{\bar{W}}$ copies of the same particle $P_i(t)$ in $P(t+1)$. For each of such new $P_i(t+1)$, we let $P_i(t+1).obj = P_i(t).obj$ and $P_i(t+1).loc = P_i(t).loc$.

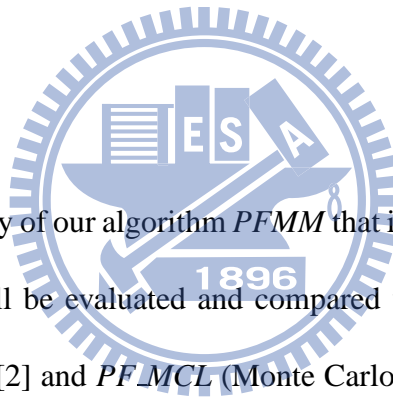
However, $P_i(t + 1).wt$ will be decided in the next round. The result is a new multiset $P(t + 1)$ with n elements.



Chapter 4

Simulation Environment and

Results



In this chapter, the accuracy of our algorithm *PFMM* that integrates mobility models into particle filters will be evaluated and compared with *NNSS* (the nearest neighbor in signal space) [2] and *PF_MCL* (Monte Carlo Particle Filters) [12] in simulation. The *PF_MCL* is the particle filter approach without any information of user mobility. We examine the impacts of various parameters, including the number of access points, the number of training locations, noise deviation, degree of irregularity, wall attenuation factor, and the accuracy of the *IMU* module.

In the simulation, we consider a 100×100 square meters sensing field. See Figure Figure 4.1.

The access points are placed at $(5, 5)$, $(40, 5)$, $(60, 5)$, $(95, 5)$, $(95, 40)$,

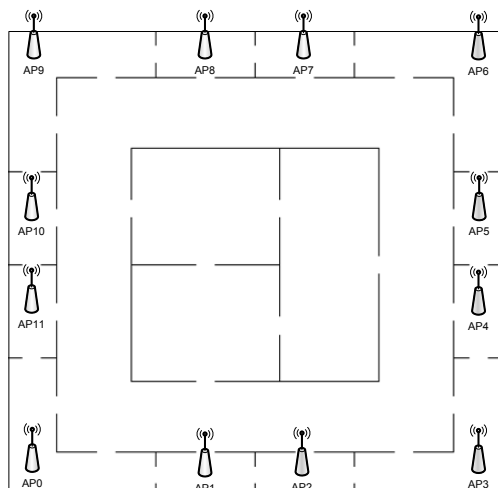


Figure 4.1: An illustration of the simulation environment.

(95, 60), (95, 95), (60, 95), (40, 95), (5, 95), (5, 60) and (5, 40). The solid lines indicate walls. The walls will not only affect the path loss of radio frequency signals but also limit the mobility of users. Particles will be filter out if their trajectories cross the walls.

4.1 Singal Path Loss Models

The signal path loss model we apply here is a variation of RIM [18]. Let $P_t^{VSP}(b_j)$ denote the transmission power of transmitter b_j and $P_r(\ell, b_j)$ be the reception

signal strength at location l . Then,

$$P_r(\ell, b_j) = P_t^{VSP}(b_j) - PL^{DOI}(\ell, b_j) - PL^{OBS}(\ell, b_j) + N(0, \sigma_f), \quad (4.1)$$

where $PL^{DOI}(\ell, b_j)$ is the path loss caused by obstacle, $PL^{OBS}(\ell, b_j)$ is the path loss caused by the obstacles in the environment, and $N(0, \sigma_f)$ representing background noises is a zero-mean normal distribution random variable with a standard deviation σ_f . Details are given below.

P_t^{VSP} is hardware-dependent and also affected by the remaining battery level that is modeled by a *variance of sending power (VSP)*, e.g.,

$$P_t^{VSP} = P_t \times (1 + N(0, VSP)), \quad (4.2)$$

where P_t denotes the initial transmit power and $N(0, VSP)$ is a zero-mean normal distribution random variable with a standard deviation VSP . Each AP randomly initials its P_t^{VSP} as the simulation starts.

In real-world environments, the irregularity of signal fading is a common phenomenon. However, most path loss models do not take this non-isotropic property into consideration. In our simulation, the *degree of irregularity (DOI)* is applied to control the amount of path loss in different directions, e.g.,

$$PL^{DOI}(\ell, b_j) = PL(\|\ell, b_j\|) \times K_i, \quad (4.3)$$

where $PL(\|\ell, b_j\|)$ is the optimal obstacle-free path loss formulation and the co-

efficient K_i is to reflect the level of irregularity at degree $i = 0, \dots, 359$ such that

$$K_i = \begin{cases} 1 & \text{if } i = 0 \\ K_{i-1} \pm W(0, \sigma_d, \gamma) \times DOI & \text{if } i = 1..359 \end{cases} \quad (4.4)$$

where $|K_0 - K_{359}| \leq DOI$ and $W(0, \sigma_d, \gamma)$ is a zero-mean Weibull random variable. The parameter DOI controls the allowable difference of two consecutive degrees¹. Specifically, a larger $|K_i - 1|$ means that the amount of path loss has greater deviation from the optimal path loss formulation at the i -th degree. The iterative definition of K_i lets the variation of irregularity be continuous.

In an indoor environment, complicate partition is one of the major factors which influence the performance of positioning algorithms. When signals penetrate through obstacles, such as walls, dramatic signal attenuation is companioned. The path loss $PL^{OBS}(\ell, b_j)$ stands for the amount of signal strengths absorbed by obstacles between the transmitter b_j and the receiver at ℓ . We adopt the concept of *wall attenuation factor (WAF)* proposed in [2]:

$$PL^{OBS}(\ell, b_j) = \min(N_{obs}, maxW) \times WAF, \quad (4.5)$$

where N_{obs} is the number of obstacles which exist in the middle of the line-of-sight path of signal transmission from b_j to ℓ , $maxW$ is the maximum number of obstacles that can influence $PL^{OBS}(\ell, b_j)$, and WAF is a parameter which de-

¹The irregularity of those non-integer degrees can be inferred by interpolating the values of two adjacent coefficients K_i and K_{i+1} with integer degrees.

notes the amount of signal attenuation caused by one obstacle. Note that different materials may have different WAF values.

The default parameters of the simulation is as below. The path lose reference power is 37.3, path lose exponent is 3.3 and path lose noise deviation $\sigma_f = 3$. The number of access points $APs = 8$. The training grain size $grain_size = 1$. 100 training samples will be collected at each training location. The mobility speed is 1 (m/s) and positioning period is 1 second. $P_t = 15$ for all APs, $VSP = 0.02$, $DOI = 0.004$, $maxW = 4$ and $WAF = 3$. Finally, the number of particles of particle filter is 500. In the simulation, we only adjust the corresponding parameters and the rest parameters are set to default value.

4.2 The Impacts of System Parameters

The number of APs, denoted as APs , somehow means the dimensions of signal patterns and affects the accuracy of pattern-matching localization. However, it also reflects deployment cost and training effort. Refer to Figure 4.2. We can see the errors decrease obviously as APs increases. The accuracy of $NNSS$ decreases dramatically if APs became fewer. However, based on the mobility model, $PFMM$ alleviates this problem and outperforms others.

Noise disturbs $RSSI$ received by users. That affects the accuracy of positioning. We wonder if the noise problem can be mitigated by increasing the number

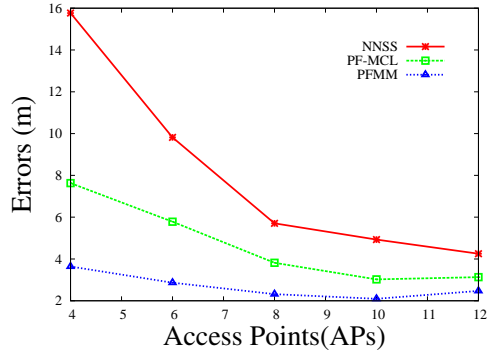


Figure 4.2: An impact of the APs .

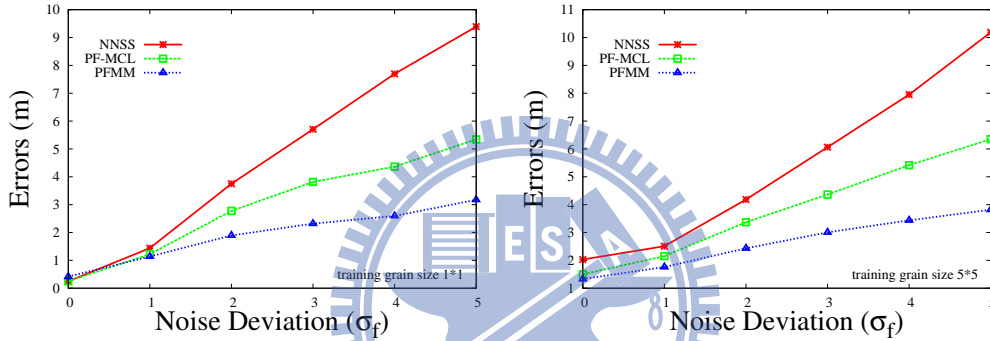


Figure 4.3: An impact of noise deviation with training grain size.

of training locations. A small training grain size is used to increase the number of training locations. Figure 4.3 illustrates the outcomes. The left figure is with $grain_size = 1$ and 10201 training locations, and the right figure is with $grain_size = 5$ and 441 training locations. The errors increase with the noise deviation σ_f . Nevertheless, the errors of $PFMM$ is within three meters no matter with dense or sparse training locations. Especially, in the case of $\sigma_f = 3$,

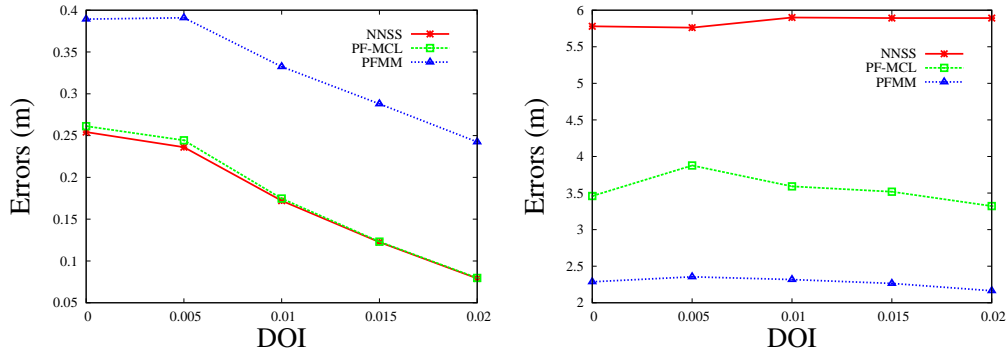


Figure 4.4: An impact of DOI .

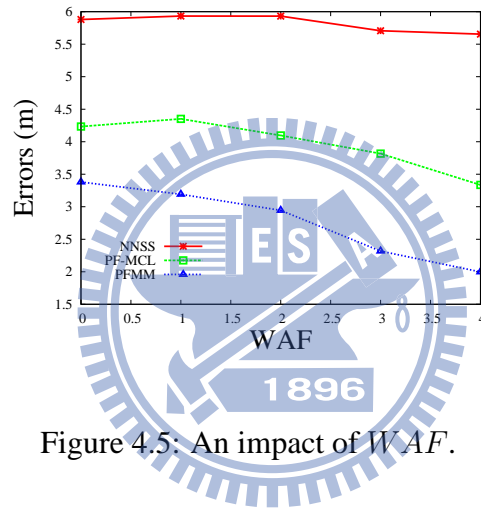


Figure 4.5: An impact of WAF .

$PFMM$ improves $NNSS$ up to one half and PF_MCL more than 25%.

The DOI is used to reflect the irregularity of radio frequency signals. The larger DOI is used, the more irregular the $RSSI$ patterns become. We measure the errors with various DOI setting in the environment with different noise deviation $\sigma_f = 0$ and $\sigma_f = 3$. In Figure 4.4, the left figure is for $\sigma_f = 0$ and the right one is for $\sigma_f = 3$. The curves in the right figure is flatter than in the left figure.

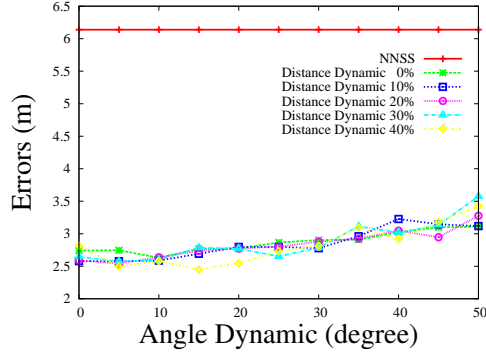


Figure 4.6: An impact of estimating dynamic of *IMU* Sensors.

The impact of signal noise is superior to *DOI*.

WAF is the wall attenuation factor representing the complexity of indoor environments. Large *WAF* implies that the RSS decreases dramatically when passing wall. In Figure 4.5, we can see an apparent decreasing trend as the *WAF* increases.

At last, we evaluate the impact of performance by varying estimating dynamic of the sensors. In our localization engine, we refer to the *RSSI* patterns and sensor measurements for positioning. Hence, the measuring errors of sensors will be corrected by the *RSSI* patterns.

Now, we increase the measuring dynamic of sensors with signal noise ($\sigma_f = 3$) as Figure 4.6 to evaluate the robust of *PFMM*. We set the mobility speed at 5 to strengthen the impact. As we can see, the system is robust. Besides, the other analysis is that the effects of angle dynamic is much bigger than distance dynamic.

Chapter 5

System Implementation

The system is divided into three parts to implement which are mobile user, positioning server, and localization environment. The mobile user carries an Android smart phone, called HTC magic and equipped with a WiFi interface, and an IMU, called 3DM-GX1 made by MicroStrain company. The dimension of HTC magic is $113 \text{ mm} \times 55.56 \text{ mm} \times 13.65 \text{ mm}$, and its weight including battery is 118 g. The version of the android operating system is 1.5. It supports 2G (GSM), 2.5G (GPRS), 3G (WCDMA) and WiFi 802.11 b/g networks. And, the GPS, G-sensor and electronic compass are embedded. 3DM-GX1 is composed of one triaxial accelerometer, one triaxial magnetometer, and one triaxial angular rate gyroscope. The dimension of 3DM-GX1 is $64 \text{ mm} \times 90 \text{ mm} \times 25 \text{ mm}$, and its weight is 75 grams. The IMU can provides 3D g-values in the range of $\pm 5 \text{ g}$, 3D magnetic field in the range of $\pm 1.2 \text{ Gauss}$, and the rate of rotation in the range of 300° per

second. The sampling rate of these readings is 350 Hz at most. In addition, it can provide its orientation in Euler angle (pitch, roll, yaw) but at most in the rate of 100 Hz. The 3DM-GX1 communicates with the handheld device via an RS-232 or RS-485 interface. The optional communication speeds are 19.2, 38.4 and 115.2 kBaud.

The 3DM-GX1 is equipped by user with the belt as Figure 2.1. We gain the prompt mobility and behaviors of user by it's measurements. For instance, we adopt a proposed walking judged algorithm using accelerations to simulate a pedometer. Therefore, it can determine that the user is walking or stopping, and cumulate the strides. And, we use the yaw of euler angle directly to simulate the electronic compass to track the orientations of user.

The g-sensor communicates with the mobile device via a UART interface or Zigbee protocol. The mobile device has the ability to sense Wi-Fi radio signals sent from surrounding access points. In the localization, the user are positioned accordance with the positioning interval. We use a program to collect the Wi-Fi radio signals and IMU sensor measurements regularly. And then, the sensor measurements will be imported into the Behavior Predicting Module. When positioning interval expired, a positioning pattern including RSS, strides, heading, and user's behaviors will be packaged and send to the positioning server via the WLAN. Secondly, once the positioning server receives a positioning pattern, the positioning pattern will be inputted to the positioning algorithm. Upon the lo-

calization result is estimated, it will be forwarded to the mobile user for demonstrating. In the last part, the environment of localization is a multi-storey building deploying WLAN infrastructure. The floor layout maps of the building are constructed in 2.5-dimensional (2.5D) description including hallways, rooms, walls, stairways and elevators. In the localization, the positioning server uses it as the positioning reference.

The performances are evaluated by the errors of localization in the building (the 4th, 5th and 6th floors) of the Computer Science Department at the National Chiao Tung University, Engineering Building III. The dimensions of the floors are 74.4 meters by 37.2 meters. The map of the positioning area is represented in the 2.5D format to assume that the feet of the user are constrained to lie on the floor during the stance phase. It is consisted of the floor plane maps including walls and rooms. And, they are connected by the stairways and elevators. In the positioning, this map will be imported into the localization system. There are 153 fingerprints deployed randomly on the hallways and other public areas of the map. Each of them is trained by 100 RSS patterns. The performances are contrasted with the nearest neighbor in signal space (*NNSS*), the mobility free particle filter (*PF_MCL*) and our sensor enhanced particle filter (*PFMM*) by moving around in the positioning area. The number of particles of these particle filter approaches is 500. The user is positioned with the period per second, and the average amount of access points which can be detected of each pattern is 11. The errors are 7.33,

4.92 and 3.01 meters individually. The *PFMM* improves *NNSS* by 59% and *PF_MCL* by 39%.



Chapter 6

Conclusions

In this paper, we developed an application level location tracking system. We proposed a sensor-enhanced particle filter scheme to assist the RF-based pattern matching localization system by user's immediate information. The user equips the IMU sensors (called G-Sensor) and electronic compass (called M-Sensor) with the belt. And, the sensor measurements will be converted to a mobility model. Based on these models, we enhanced the sampling stage of the traditional particle filter. The particles will be propagated by horizontal displacement vector and vertical displacement vector to close the user. And, we added a particle filtering module between the particle sampling module and particle weighting module. The incredible particles will be filtered out by speed filtering and passing wall filtering with maps information. We examined the performance by errors of localization in the simulation and experiment. According to the statistics in the simulation,

our performances are outstanding in spite of any environment. And, our engine is robust regardless the increasing of sensor errors. In the experiment, we improved *NNSS* by 59% and *PF_MCL* by 39%.



Bibliography

- [1] M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, and A. Hopper. Implementing a Sentient Computing System. *Computer*, 34(8):50–56, 2001.
- [2] P. Bahl and V. N. Padmanabhan. RADAR: An In-building RF-based User Location and Tracking System. In *IEEE INFOCOM*, volume 2, pages 775–784, 2000.
- [3] S. Beauregard, Widyawan, and M. Klepal. Indoor pdr performance enhancement using minimal map information and particle filters. In *Position, Location and Navigation Symposium, 2008 IEEE/ION*, pages 141–147, May 2008.
- [4] E. Brassart, C. Pegard, and M. Mouaddib. Localization using infrared beacons. *Robotica*, 18(2):153–161, 2000.

- [5] B. Ferris, D. Hhnel, and D. Fox. Gaussian processes for signal strength-based location estimation. In *In Proc. of Robotics Science and Systems*, 2006.
- [6] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello. Bayesian Filtering for Location Estimation. *IEEE Pervasive Computing*, 2(3):24–33, 2003.
- [7] S.-P. Kuo and Y.-C. Tseng. A Scrambling Method for Fingerprint Positioning Based on Temporal Diversity and Spatial Dependency. *IEEE Trans. on Knowledge and Data Engineering*, 20(5):678–684, 2008.
- [8] D. Niculescu and B. Nath. Ad Hoc Positioning System (APS) Using AOA. In *IEEE INFOCOM*, volume 3, pages 1734–1743, 2003.
- [9] J. J. Pan, J. T. Kwok, Q. Yang, and Y. Chen. Multidimensional Vector Regression for Accurate and Low-Cost Location Estimation in Pervasive Computing. *IEEE Trans. on Knowledge and Data Engineering*, 18(9):1181–1193, 2006.
- [10] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *ACM/IEEE MOBICOM*, pages 32–43, 2000.
- [11] A. Savvides, C.-C. Han, and M. B. Strivastava. Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors. In *ACM/IEEE MOBICOM*, pages 166–179, 2001.

- [12] S. Thrun, D. Fox, and W. Burgard. Monte carlo localization with mixture proposal distribution. In *Proc. of the National Conference on Artificial Intelligence*, 2000.
- [13] T.-C. Tsai, C.-L. Li, and T.-M. Lin. Reducing Calibration Effort for WLAN Location and Tracking System using Segment Technique. In *IEEE Int'l Conf. on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*, volume 2, pages 46–51, 2006.
- [14] Widyawan, M. Klepal, and S. Beauregard. A novel backtracking particle filter for pattern matching indoor localization. In *MELT '08: Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments*, pages 79–84, New York, NY, USA, 2008. ACM.
- [15] O. Woodman and R. Harle. Pedestrian localisation for indoor environments. In *UbiComp '08: Proceedings of the 10th international conference on Ubiquitous computing*, pages 114–123, New York, NY, USA, 2008. ACM.
- [16] L.-W. Yeh, M.-S. Hsu, Y.-F. Lee, and Y.-C. Tseng. Indoor localization: Automatically constructing today's radio map by irobot and rfids. In *IEEE Sensors Conference*, 2009.

- [17] J. Yin, Q. Yang, and L. M. Ni. Learning adaptive temporal radio maps for signal-strength-based location estimation. *IEEE Transactions on Mobile Computing*, 7(7):869–883, 2008.
- [18] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic. Impact of Radio Irregularity on Wireless Sensor Networks. In *ACM MobiSys*, pages 125–138. ACM Press New York, NY, USA, 2004.

