國立交通大學

網路工程研究所

碩 士 論 文

行動裝置平台上之樣本比對定位演算法的快取記憶體管理

A Cache Management Scheme for Pattern-Matching Localization at the Client Side

研 究 生：張哲賓

指導教授：曾煜棋　教授

中 華 民 國 九 十 八 年 八 月

行動裝置平台上之樣本比對定位演算法的快取記憶體管理

A Cache Management Scheme for Pattern-Matching Localization at the Client Side

研 究 生：張哲賓　　Student：Che-Pin Chang

指導教授：曾煜棋　　Advisor：Yu-Chee Tseng

國 立 交 通 大 學

網路工程研究所

碩 士 論 文

A Thesis

Submitted to Institute of Network Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

In

Computer Science

June 2008

Hsinchu, Taiwan, Republic of China

中華民國九十八年八月

行動裝置平台上之樣本比對定位演算法的快取記憶體管理

學生: 張哲賓　　指導教授: 曾煜棋 教授

國立交通大學網路工程研究所碩士班

摘　　要

隨著行動裝置與區域無線網路的普及,位置相關系統與服務獲得越來越多的注意。現今熱門的 GPS 定位系統由於在室內無法定位的缺點，導致其不適用於位置相關服務。相反地，類似 RADAR 的定位系統在室內外都可以定位，此外，不需額外硬體成本使的它更適用於位置相關服務。然而目前以 RADAR 為基礎的定位演算法都需要大量的資料與龐大的計算量來達到可接受的精準度,因此並不適合在硬體資源有限的行動裝置上運行。為了解決此問題，本論文提出了一個新穎的記憶體管理機制與預測下載機制。透過這些方法，我們希望能實現一具有一定精準度且較不須依賴後端伺服器的行動裝置端樣本比對定位系統。

關鍵字:指紋定位演算法、行動運算、樣本比對定位演算法、無線網路。

A Cache Management Scheme for Pattern-Matching Localization at the Client Side

Student: Che-Pin Chang Advisor: Prof. Yu-Chee Tseng

Department of Computer Science
National Chiao-Tung University

ABSTRACT

The proliferation of mobile computing devices and local-area wireless networks has fostered a growing interest in location-aware systems and services. Current positioning technique, GPS, are not suitable for more and more LBS(Location Based Service) because of its incapability of indoor positioning. In contrast, RADAR-like system can work both indoor and outdoor. Furthermore, they only need to use existing hardware and are, thus, more suitable for LBS. However, existing RADAR-like algorithms all need extensive data and large amount of computation to achieve acceptable accuracy. Therefore, not suitable for positioning on resource limited portable device. In order to solve this problem, we propose a novel cache management scheme and prediction method in this paper. By using these techniques, we hope to achieve a client-based pattern-matching positioning system with fair enough accuracy and less dependent to server.

Keywords: Fingerprinting Localization, Mobile Computing, Pattern-Matching Localization, Wireless Network.

誌謝

由衷地感謝曾煜棋教授這兩年來不辭辛勞的指導，讓我學到了很多寶貴的知識跟經驗還有為人處事的方法，感謝實驗室的學長姐跟學弟妹，謝謝你們這兩年多來的幫忙跟提攜，讓我得以順利完成學業，感謝爸媽的辛苦，讓我在求學的過程中沒有後顧之憂，感謝交大資工系計中的同仁們，讓我學習了系統管理的技巧與實際操作的經驗，感謝一路陪伴我的交大田徑隊，讓我除了學業之外能有個豐富的生活，最後，特別感謝郭聖博學長跟羅榮鐘學長，謝謝你們總是適時的提醒我，告訴我正確的人生態度跟處理事情的方法，真的讓我獲益良多。

在此向大家獻上我誠摯地謝意跟祝福

張 哲 賓 謹識於

國立交通大學網路工程研究所碩士班

中華民國九十八年八月

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recently, portable devices have been more and more popular. From the statistic, global mobile penetration is about 61% now. With the stronger computation power, much more interesting application has been developed for the portable device. With the advent of Global Positioning System (GPS) [1] ,cellular-based and multi-lateration approaches [2] [3], Location-Based Services(LBSs) has gathering more interest than other application.

Unfortunately, the level of localization accuracy needed in such applications cannot be achieved by the existing cellular-based methods. Furthermore, coverage of the GPS system is limited in indoor environments and dense urban areas. Therefore, much effort has been focused on the development of Radio Frequency (RF)-based location-estimation techniques using Received Signal Strength (RSS) measurements, such as RADAR [4]. Such systems consist of two phases: an offline training phase and an online positioning phase. In the offline training phase, a mapping between received signal strength (RSS) patterns and location labels at a set of training locations will be learned. During the online positioning phase, we can compare the newly received RSS pattern against those of the training locations [4, 5, 6, 7, 8, 9, 10].

For RADAR-like positioning technique, the larger scale of the sensing field is, the more training location will be needed. To reduce the location grain size,

a numerical method is proposed to generate some virtual training data inferred from nearby real training data [8]. Hence, the training overhead can be limited to a relatively small number of real training locations. However, when it comes to a scale of a city, a database is still needed to store the huge amount of training locations. Limited by the storage size, if we want to make positioning available on the resource limited portable device whether indoor or outdoor, the only way possible is via a centralized positioning server. The portable device simply report its RSS to the central sever. After calculation, the server will response a possible location to the client. This is the widely used server-client architecture. Under this situation, the client must stay connected to the Internet to be able to talk to the server. Since the calculation is done by the server, current location of the client is not a secret to the server and even worse, might be eavesdropped during transmission. Therefore, it might incur network connectivity and privacy concern for portable device.

To solve the insufficient resource, network connectivity and privacy concern problem, we propose a novel cache management scheme aimed at positioning on portable device. For different pattern-matching positioning algorithms, we try to extract feature data from its overall training data first. When portable device starts, it just need to download partial feature data from the central server relative to its current location. The current location here can be a rough one to keep privacy. With these small amounts of data, a portable device can have its self positioning ability.

To evaluate the proposed scheme in large-scale environments, we adopt a close-to-reality *radio irregularity model (RIM)* [11] to simulate the decay of signal strengths. This model has been shown to be able to reflect the physical reality of radio signals, such as the influence of hardware difference, remaining battery, non-isotropic propagation, and dynamic signal fading effect. These features cause the difficulty of indoor positioning. In our simulation study, we tune the param-

eters of RIM to evaluate our cache management scheme under different conditions of such close-to-reality environments. The results support that our cache management scheme do provide less network connection need and availability of positioning on resource limited portable device without losing privacy.

The rest of this paper is organized as follows. Section 2 reviews previous works on pattern-matching localization. The proposed cache management scheme are presented in Section 3. Performance studies are in Section 4. Finally, Section 6 draws a brief conclusion.

Chapter 2

Problem Statement

A pattern-matching localization system [4] generally works as follows. We are given a set of *beacons* $\mathcal{B} = \{b_1, b_2, \ldots, b_n\}$, each capable of transmitting radio signals periodically in a sensing field $\mathcal{F} \subseteq \mathbb{R}^2$, and a set of known training locations $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_m\}$ in $\mathcal{F}$. The system works in two phases. In the *training phase*, at each training location ℓ_i, $i = 1..m$, we measure the signal strengths from beacons for a period of time and create a *feature vector* $\boldsymbol{v}_i = [v_{i,1}, v_{i,2}, \ldots, v_{i,n}]$ for ℓ_i, where $v_{i,j} \in \mathbb{R}$ is the averaged RSS from b_j, $j = 1..n$. The feature space of $\boldsymbol{v}_i$ is written as $\mathcal{S} \subseteq \mathbb{R}^n$. The set of feature vectors $\mathcal{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m\}$ is stored at a location server. In the *positioning phase*, an object can measure its RSS vector $\boldsymbol{s} = [s_1, s_2, \ldots, s_n]$ and compare $\boldsymbol{s}$ against $\mathcal{V}$ to estimate its location in $\mathcal{F}$.

Traditionally, the positioning task is conducted at the server side. This is due to the facts that the feature vector set $\mathcal{V}$ is typically very large and portable devices have limited resources. Not only the comparison task is costly, but also transporting $\mathcal{V}$ to portable devices requires large bandwidths.

In this work, we are interested in moving the positioning task to the client side. This implies that $\mathcal{V}$ or part of $\mathcal{V}$ needs to be stored at a portable device. The positive effect is that the portable device can locate itself even if it loses network connection and it does not need to keep on inquiring the location server. Therefore, positioning can be done in a more teal-time manner. For example, if a

person walks at a speed of 4km/hr and we require a positioning precision of 1m, then the portable device needs to inquire the server approximately 66.6 times per minute. When the network connection is unreliable, maintaining such a level of precision is difficult.

Moving the positioning task to the client side may face two challenges. First, the data(which we call *cache*) to be stored at the client should be kept as small as possible. Second, a cache management strategy is needed when a user moves out of the region for which his current local cache data is valid, in which case the local cache needs to be updated. For the first issue, we will adopt the *discriminant function (DF)*-based localization methodology [12], which relies on an path loss model technique and is believed to place less memory demand on the positioning job. For the second, we will propose an efficient solution in Section 3.

For completeness, we review the DF-based methodology below. The basic idea behind the DF-based localization is to construct a *continuous* and *differentiable* discriminant function $f : \mathcal{F} \mapsto \mathbb{R}^+$ from the given $\mathcal{L}$, $\mathcal{V}$, and $\boldsymbol{s}$. Then, apply an optimization search algorithm to find the location in $\mathcal{F}$ that minimizes f. Instead of evaluating the difference between $\boldsymbol{s}$ and each vector in $\mathcal{V}$, the search algorithm is expected to quickly find a location $\ell_{est} \in \mathcal{F}$ that is close to the optimal location $\ell^* = \arg\min_{\ell \in \mathcal{F}} f(\ell)$. Specifically, it will exploit the *gradient descent search* [13] to find the optimizer of the discriminant function. This scheme is an iterative process which can quickly converge to the minimizer of the given discriminant function. Two possible ways to define the discriminant function were proposed, called *Newton-PL* and *Newton-INT* . They are based on the path loss model and the interpolation technique, respectively. Both of them have the continuity and the differentiability properties and satisfy $f(\ell_i) \approx h(\ell_i)$, $i = 1..m$. The latter implies that the functions f and h have similar decreasing trends.

Newton-PL designs its discriminant function by estimating the path loss of the environment to simulate the channel condition. From path loss, it expects

to measure the amount of signal strength degradation from each beacon at each location in $\mathcal{F}$. In the training phase, some path loss-related parameters will be computed. In the positioning phase, a discriminant function will be constructed according to the vector $\boldsymbol{s}$ and these parameters. From this function, the estimated location ℓ_{est} will be computed by means of the optimization search algorithm. By transforming training data to the path loss-related parameters, the data needed for positioning can be greatly reduced.

Newton-INT adopts the *inverse distance weighted interpolation* (or so-called *Shepard interpolation* [14]), a numerical data-fitting approach. Similar to Newton-PL, Newton-INT also needs to infer the received power model $P_r(\ell, b_j)$ for each b_j from the training data. The main idea is to use a weighted function to predict the signal strength at each location $\ell \in \mathcal{F}$. We assume that training locations that are closer to ℓ will contribute more influence on the received signal strength at ℓ. Although Newton-INT doesn't have the data reduce ability like Newton-PL, it can achieve better accuracy especially under complex environments. Therefore, we will implements both algorithms in this paper.

Chapter 3

The Proposed Cache Management Scheme

By adopting DF-based localization methodology, we can reduce data size to an extent, but the overall data are still not small enough to fit into a portable device when the the scale of $\mathcal{F}$ becomes the country scale. Downloading partial data could be one possible solution, but it will yield other new questions like what to be downloaded or what will happen when the downloaded data is invalid. Therefore, our main objective is to build a cache management scheme that help portable devices choose the data they need when they start the positioning system. Also, we will help portable devices update their cache when they find they are going to reach out of the region where their local cache is valid. There are also two phases in our algorithm, *off-line training* and *online positioning* phases. In the first phase, we collect data as usual, but cluster all these data according to some constraints. During the second phase, portable device will download partial data relative to its current position for position usage. After several positioning, it may find the data it owned is not enough for further positioning; this is when the update happens.

3.1 The Off-line Training Phase

During the off-line training phase, we collect training data first as usual and then generate related parameters $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$. Then, we observed that in the positioning phase not all parameters are needed to construct the discriminant function f. Both received power model $P_r(\ell, b_j)$ are inverse proportional to the distance from the ℓ to the beacon or the training location. This means that beacons or locations too far away from the location of the portable device would have very few effect on its signal strength and we can just ignore it. For this reason, we will split the sensing field $\mathcal{F}$ into several small area $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$; we refer to each one of it as a *tile* . The shape or the size of a tile can vary. For simplicity, we will use the square shape and an uniform size in this paper. Fig. 3.1 is an example of the partition process.

Since we split the field $\mathcal{F}$ into several tile, we need to have a connection between each tile t and the parameters p because what the portable device downloads is the parameter not the tile. The tile is like the index key to help we find the parameters we need. For Newton-PL, if we want to construct a discriminant function inside an area, we need the parameters from all of the beacons that can be heard inside. Instead of doing the test at every place inside an area, the signal strength vector of the training data give us a hint. Therefore, a beacon's parameter will be put in a tile if one of the training locations inside that tile has received the signal

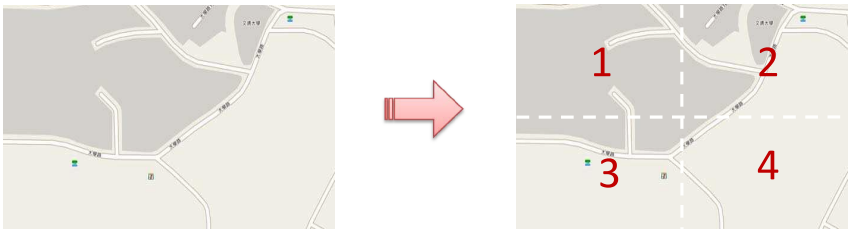

Figure 3.1: An example of area partition. The original field was split into 4 smaller area.

from that beacon. This can be expressed by m:

$$m(t_i) = \{p_j\}, \text{ if } \ell_i \text{ is inside } t_i \text{ and } v_i \text{ has } v_{i,j}. \tag{3.1}$$

where t_i is the requesting tile. For Newton-INT, the received signal strength function is generated by interpolating signal strength of neighboring training data. Therefore, a training location data will be put in a tile if it is inside that tile. This can be expressed by m:

$$m(t_i) = \{p_i\}, \text{ if } \ell_i \text{ is inside } t_i. \tag{3.2}$$

3.2 The Online Localization Phase

During the online localization phase, the portable device first reports its rough position and its available storage size to the server. The rough position can be easily obtained by using GSM Cell-ID or any other primitive way like simply letting user chose its area. Then, the server will try to recognize its belonging tile and the surrounding tiles. The parameter of these tiles can be look as a basic data unit for portable device. The portable device will be able to position himself as long as he has the parameters of the belonging tile. But once he steps out the positioning-able area, the positioning error will increase rapidly due to the lack of parameters. In order to prevent this situation from happening, we let portable device download the parameters of the surrounding tile . Once he reaches out the central tile, he will need to download new data but still be positioning-able at the same time. After basic data unit is ready, the server will or will not add up more tile's parameter according to portable device's available storage size or user's requirement. Anyhow, the overall tiles will form a small field. The dimension of this field will accompany with all previous parameters to be send to the portable device. Purpose of transferring the dimension of this field will be explained later. Fig. 3.2 shows an example of the initialization.

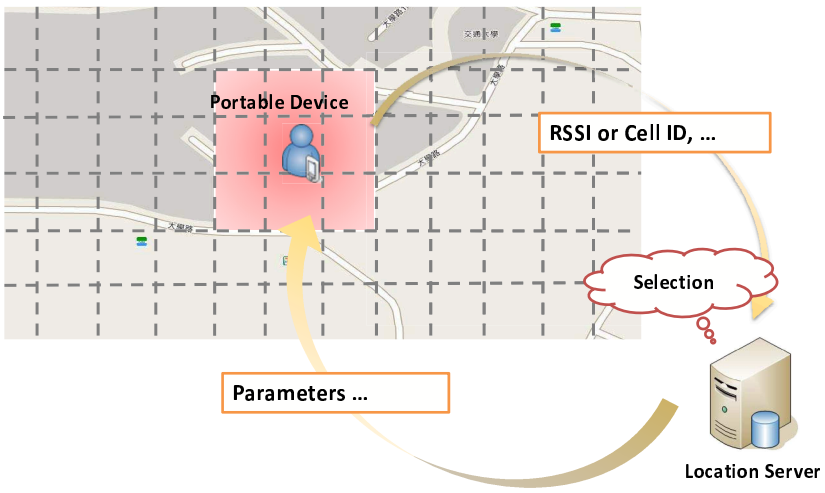

Figure 3.2: An illustration of the positioning system initialization on the portable device.

After downloads all these data, the portable device will be capable of positioning by itself. The only job it need to do is to plug-in current received $\boldsymbol{s}$ into the received power model $P_r(\ell, b_j)$, then constructs the discriminant function f and applies an optimization search algorithm to find the location. Sooner or later, the portable device will detect that it is going to reach out the field. It will have to report to the server and downloads new data again. We refer to this process as *cache update*. This process can be further divided into two parts: *deletion* and *selection*.

3.2.1 Deletion

Since the storage is limited, we can never add new data in whenever old data are still there. Otherwise, we will face the problem of out of the memory. Every time we want to do cache update, we need to delete old tile data first. Then the problem comes: What are we going to delete first? The simplest answer is by deleting all data and ask for whole new data we need for further positioning usage. But this could just waste a lot of still usable data and increase network communication cost. By observing that the field of user owned tiles usually overlaps with the last

time's field, we know we can keep the overlapping tiles and delete those not. The dimension of the field can help portable device finds out what is needed and what is not. After deleting those not needed, the portable device can simply report other needed tiles to the server.

3.2.2 Selection

As mentioned previously, we let overall tiles downloaded composed of at least one data unit. More tiles can be added according to any different requirements. We refer to the adding process as *selection* . In the following, we introduce three different selection methods including one basic method and two advanced methods.

Basic Selection

The basic selection do the same thing as the first time positioning. The selection process starts from the belonging tile of the portable device and selects 8 surrounding tiles too. Through out whole positioning process, the portable device will always download a basic data unit only. This is the most fundamental selection method. But since the storage size limit may exceed the size of a basic data unit and our goal is try to reduce network connection time and be as independent as possible, we should try to use a smarter selection method.

Greedy Selection

As implied by the name, the greedy selection just simply tries to select as more tile as it can. The selection process starts from the belonging tile of the portable device. Choosing the surrounding tile one after another until the required size of the parameters meets the limit of the storage size of the portable device. Fig. 3.3 is an example of greedy selection under different storage constraints. Assume the storage requirement for each tile is 1 MB, the portable device can download at

most 9 tiles when its available storage is 9 MB. But when its available storage size increases to 25 MB, it will download at most 25 tiles.

The advantage of greedy selection is it is suitable for most situation. But since people tend to move in a fix direction for short period, this will waste most of the data not in the current moving direction from the belonging tile. The connection time may be reduced, but overall hit rate may decrease.

Selection using Mobility Pattern Prediction

If we can predict where user will be later, then we may be able to select tiles based on this predicted position. There are many techniques about mobility pattern prediction, we will adopt Vector-Based tracking [15] in this paper. In vector tracking, the future positions of a moving object are given by a linear function of time, i.e., by a start position and a velocity vector. The user's current velocity could be calculated by his current position and previous position.

$$V_x = (\ell_{ix} - \ell_{(i-1)x})/(t_i - t_{(i-1)}) \tag{3.3}$$

$$V_y = (\ell_{iy} - \ell_{(i-1)y})/(t_i - t_{(i-1)}) \tag{3.4}$$

A velocity vector consists of speed and heading. The heading helps us select tiles along which direction. The speed helps us determine how far do we need for

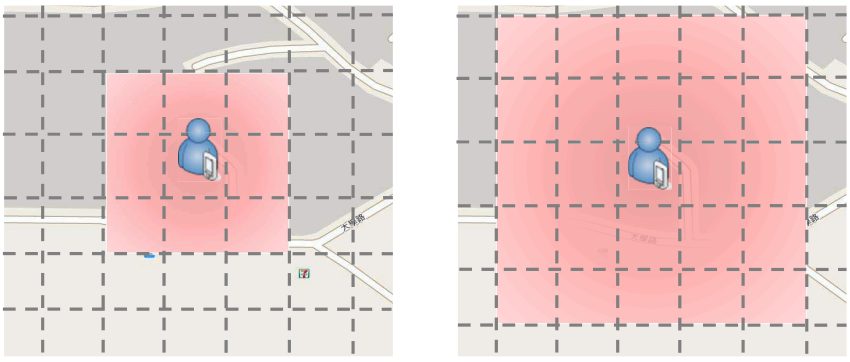

Figure 3.3: An illustration of the greedy selection under different storage constraints.

selection. If the user moves fast, he will need more tiles in order to reduce the need for update. On the contrary, fewer tiles would be needed for a user moves at a slow pace.

Fig. 3.5 is an example of the selection using mobility pattern prediction. The blue area is the portable device currently owned tiles and the red arrow shows the moving direction of the portable device. The tiles inside the pink area are selected according to the velocity of the portable device and will be downloaded later. The shape of the field formed by all the tiles selected will no longer be like a square due to the mechanism of its selection.

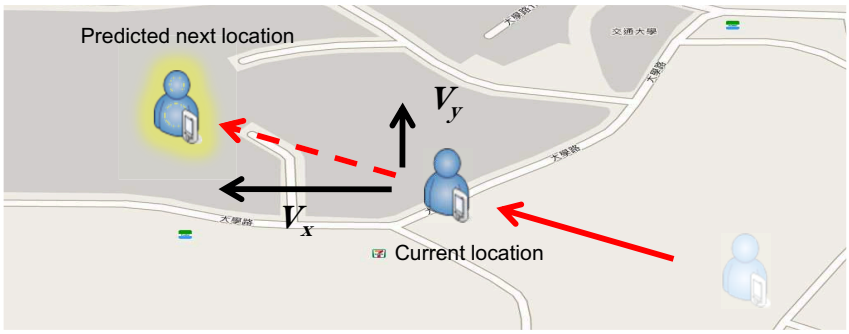

Figure 3.4: An illustration of the velocity calculation.

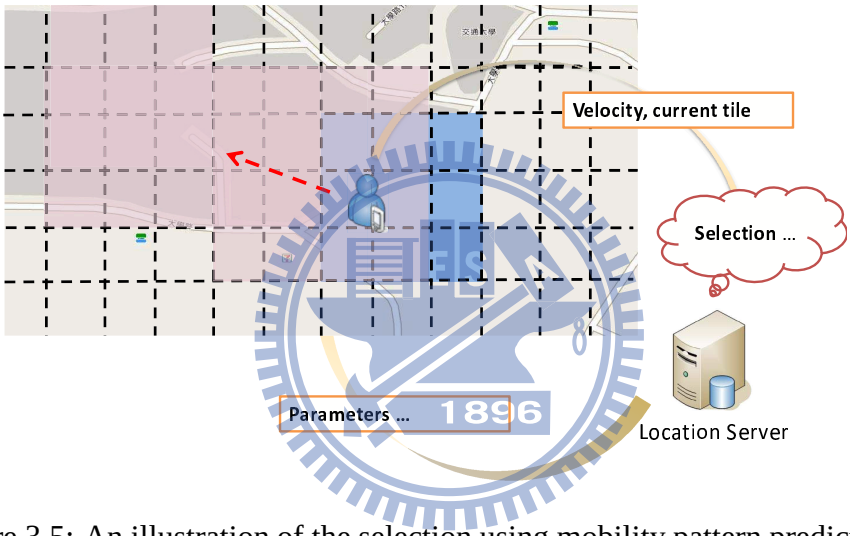

Figure 3.5: An illustration of the selection using mobility pattern prediction.

Chapter 4

Simulation Results

We consider a sensing field of size 500×500 m^2. There are 121 beacons placed at $(0+50\times i, 0+50\times j, 2)$, $i = 0..10$ and $j = 0..10$. To simulate a more realistic environment, some walls are given. Training locations are placed at grid locations where the grid size is a 5. We collect 100 training data and then average them at each training location. We modify the RIM [11] to model the decay of signal strengths:

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

where P_t^{VSP} is the transmit power, which may vary among different hardware, $PL^{DOI}(\ell, b_j)$ is the path loss which can reflect non-isotropic and continuous properties, $PL^{OBS}(\ell, b_j)$ is the path loss which can simulate the signal attenuation caused by the obstacles in the environment, and $N(0, \sigma_f)$ is a zero-mean normal random variable with a standard deviation σ_f to stand for dynamic noise.

In RIM, impacts of hardware difference and remaining battery on transmit power is modeled by the *variance of sending power (VSP)*, e.g.,

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

where P_t denotes the initial transmit power and $N(0, \mathsf{VSP})$ is a zero-mean normal random variable with a standard deviation VSP. The parameter VSP controls

the degree of variance of sending power among different beacons. Each beacon randomly selects its P_t^{VSP} when the simulation starts.

In real-world experiments, the irregularity of signal fading is a common phenomenon. However, most path loss models do not take this non-isotropic property into consideration. To capture this effect, RIM imports the *degree of irregularity (`DOI`)* to control the amount of path loss in different directions, e.g.,

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

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

$$PL(d) = PL(d_0) + 10\phi \log(\frac{d}{d_0}), \tag{4.4}$$

and the coefficient K_i is to model the level of irregularity at degree i ($i = 0..359$) such that

$$K_i = \begin{cases} 1 & \text{if } i = 0 \\ K_{i-1} \pm W(0, \sigma_d, \gamma) \times \texttt{DOI} & \text{if } i = 1..359 \end{cases} \tag{4.5}$$

where $|K_0 - K_{359}| \leq \texttt{DOI}$ and $W(0, \sigma_d, \gamma)$ is a zero-mean Weibull random variable. The parameter `DOI` controls the allowable difference of two successional degrees[1]. 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, they should have dramatic signal strength attenuation. In our simulations, the modified RIM model takes this property into consideration. Specifically, there are some walls defined in our simulation environment. The path loss $PL^{OBS}(\ell, b_j)$ stands for the amount of signal strengths

[1]The 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.

absorbed by the obstacles between the transmitter b_j and the receiver at ℓ. We adopt the concept of *wall attenuation factor (WAF)* proposed in [4]:

$$PL^{OBS}(\ell, b_j) = \max(N_{obs}, C) \times \mathsf{WAF}, \tag{4.6}$$

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, C is the maximum number of obstacles which can influence $PL^{OBS}(\ell, b_j)$, and WAF is a system parameter which denotes the amount of signal attenuation caused by one obstacle. Note that different materials have different WAF values.

In order to simulate a real moving pattern, an object moving by a waypoint model is simulated. Fig. 4.1 shows an example of how we walk. Blue lines represent the walls and red lines are the simulated path. It will switch between the moving and pausing states. In the moving state, the object will uniformly select a destination in the sensing field and moves to it at the speed 1 m/sec. After reaching the destination, the object will switch to the pausing state and stay at this location for 3 sec. The tracked object measures the signal strengths of all beacons every 1 sec. The total simulation time for the whole trip is $2,194$ sec. The default simulation parameters are set to $P_t = 15$ dBm, $d_0 = 1$ m, $PL(d_0) = 37.3$ dBm, $\phi = 4$, $\sigma_f = 2$, $\mathsf{VSP} = 0.2$, $\mathsf{DOI} = 0.005$, $\mathsf{WAF} = 3.1$, $C = 4$, $\sigma_d = 0.1$, $\gamma = 1$, $\Delta\ell_{min} = 1$ m, and the grid size is 5 m.

The performance study includes the comparisons of Newton-PL, Newton-INT, and the NNSS positioning algorithm [4] with two advanced selection algorithms, greedy and prediction. This makes total six combinations. Three performance metrics are considered: positioning error , update count, and average data size per update.

Fig. 4.2 evaluates three positioning algorithms under different levels of noise. Intuitively, no matter which positioning algorithm is adopted, the positioning error should be increased in a noisier environment. Fig. 4.2 illustrates that three algorithms have similar increasing trend. Different selection algorithms do not

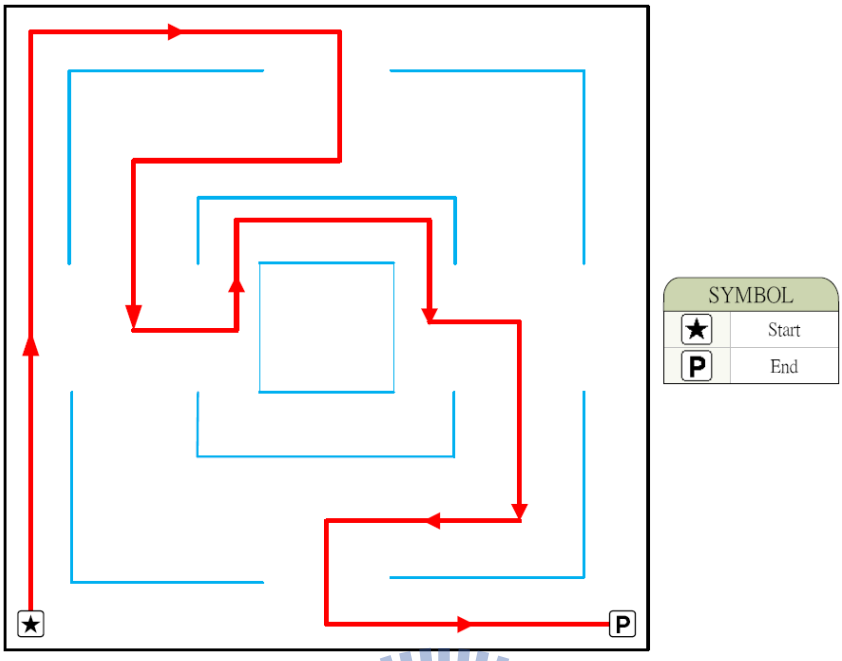

Figure 4.1: Simulation environment

affect the positioning error. It shows that a basic data unit is good enough for positioning.

Fig. 4.3 compares update count for three positioning algorithms with two selection algorithms. Overall, selection using prediction is better than greedy selection since it tries to select more useful data. Therefore, less update is needed. But when noise level becomes larger, the update count using prediction will become greater too. On the other hand, the update count using greedy selection stays the same. This can be explained by more false prediction incurred from higher positioning error. But greedy selection selects tile omnidirectionally and becomes resistible to higher positioning error.

In Fig. 4.4, the average size per update of the data that portable device has to download is shown. By transferring raw training data into simple coefficients of beacon, the Newton-PL algorithm only needs very few memory space com-

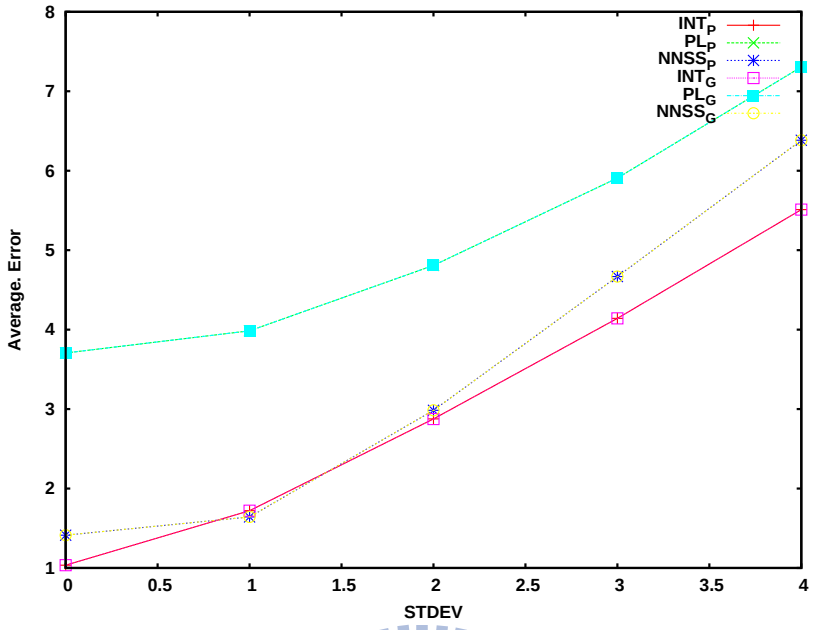

Figure 4.2: Comparisons of positioning error for NNSS, Newton-PL, and Newton-INT under different noise levels with different selection algorithms.

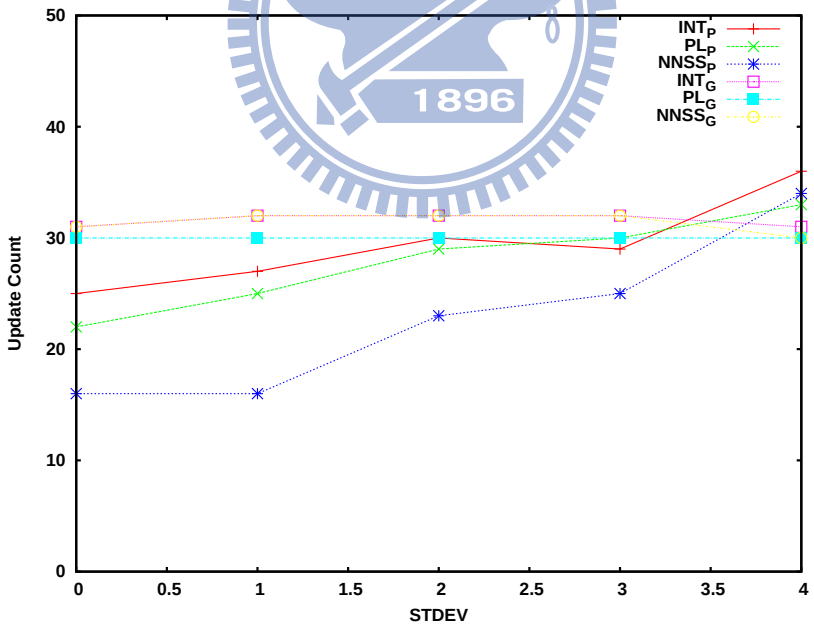

Figure 4.3: Comparisons of update count for NNSS, Newton-PL, and Newton-INT under different noise levels with different selection algorithms.

pared to other two algorithms. For Newton-INT, the additional gradient coefficient makes it need more memory space than NNSS.

In summary, after evaluating the algorithms NNSS, Newton-PL, and Newton-INT under different noise level, we have some suggestions. First of all, if device available storage is the major concern, Newton-PL is an acceptable choice because it takes very few memory space and also provides rational positioning accuracy. However, if the portable device is a high-end device and needs more accurate localization service in a well-trained and complex environment, Newton-INT can satisfy this requirement.

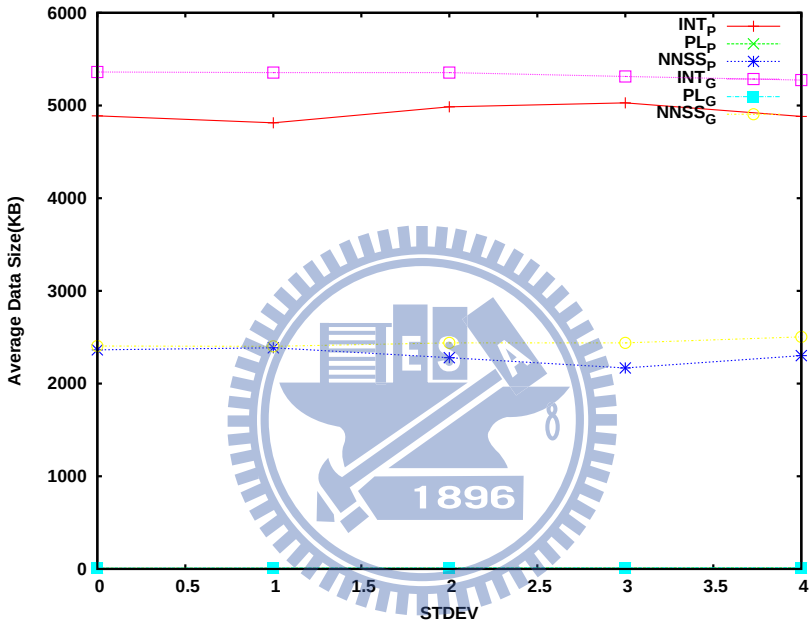

Figure 4.4: Comparisons of average data size for NNSS, Newton-PL, and Newton-INT under different noise levels with different selection algorithms.

Chapter 5

Experimental Results

The proposed cache management scheme is implemented and integrated to a *discriminant function (DF)*-based localization system. In this section, we will describe the experimental settings and show the experimental results.

5.1 Experimental Settings

We conducted the experiments on the fifth floor of our department building, where the Department of Computer Science and Engineering, Chiao Tung University, is located. The building is deployed with an IEEE 802.11b
g wireless network in the 2.4 GHz frequency bandwidth. The layout of the floor is shown in Fig. 5.1. This area measures $60m \times 30m$.

We developed a simple Visual C#-based application to assist us in the process of gathering fingerprints. To record a fingerprint, we first identify the current position by clicking on a map of the building. The application then records the signal strengths reported by the 802.11 card using Intel Mobile Platform SDK [16] provided by Intel. Fig. 5.2 is a screen shot of the Visual C#-based application. Each points are located 2 to 3 meters apart. We collected 50 measurements per location. For evaluation, we wait for an hour and do the collection process again. We only collect 15 locations and 10 measurements per location to compare with the first training data set. The collecting results for both trail are shown in Fig. 5.2

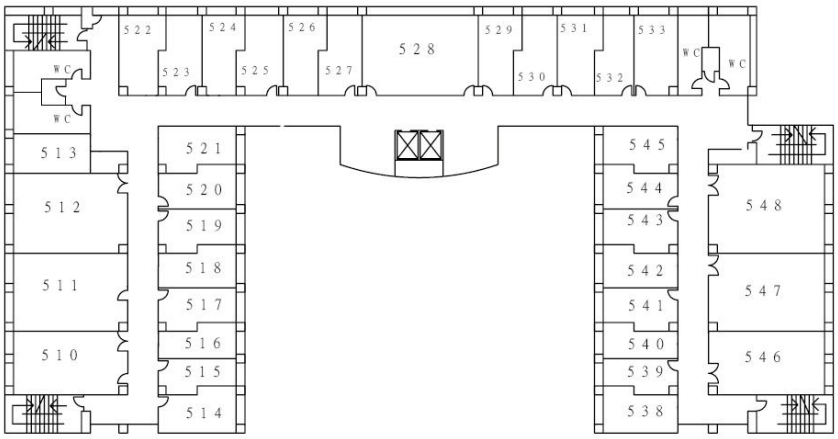

Figure 5.1: The layout of the fifth floor of the Department of Computer Science and Engineering, Chiao Tung University.

and Fig. 5.3.

5.2 Experimental Results

We set the tile size to $10m \times 10m$ and compare both Newton-PL and Newton-INT with or without basic cache management scheme. Table 5.1 shows the result. As we can see, Newton-INT performs better than Newton-PL basically. With the cache management scheme, error distance for Newton-INT was decreased. The reason for this is because the discriminant function will not be influenced by training data too far away. But for Newton-PL, the coefficients of the beacons downloaded were less related to the position of the portable device, hence shows less differences.

	Newton-INT	Newton-PL	Cached Newton-INT	Cached Newton-PL
Error Distance(m)	5.46	6.10	4.83	6.42

Table 5.1: Comparison of error distance under different algorithms.

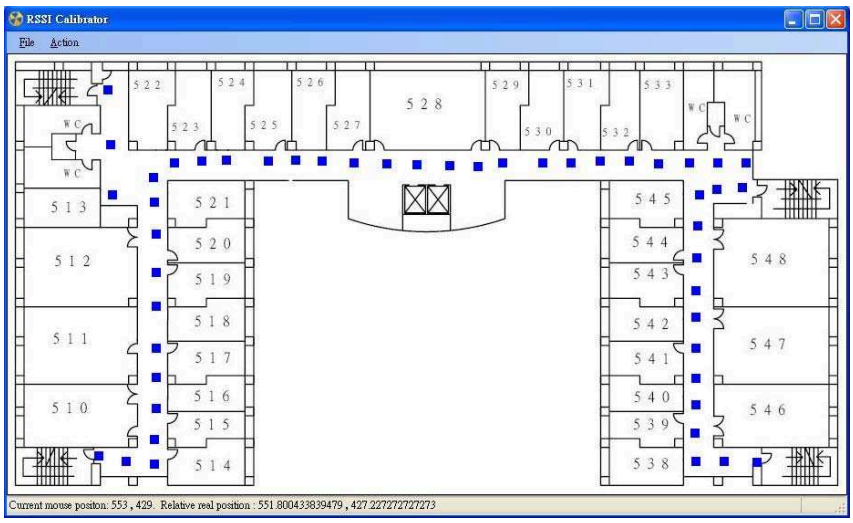

Figure 5.2: The collecting result of the training data set.

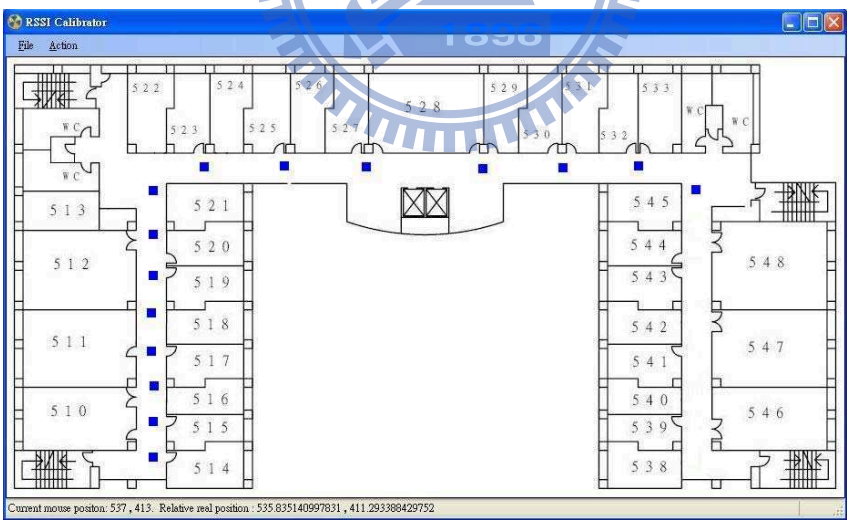

Figure 5.3: The collecting result of the comparing data set.

Chapter 6

Conclusions

In this paper, we have presented a novel cache management scheme, which can help client-based pattern-matching positioning system. It has large potential to be applied in portable device. By pushing on-line search part from the server side to the client side, the portable devices would be able to position itself even when it is not connected to the network. During the on-line search process, much storage can be saved because the large volume of training data has been transformed to a few scalars of the discriminant function at the off-line stage. This relieves the problem of limited storage resource on the portable device. With two advanced selection algorithms, overall update count is reduced by 99%, from 2196 times to 30 times. We believe these features can greatly support real-time LBSs in large-scale environments [17].

Bibliography

[1] P. Enge and P. Misra, "Special Issue on GPS: The Global Positioning System," *Proc of the IEEE*, vol. 87, no. 1, pp. 3–15, 1999.

[2] Skyhook wi-fi positioning system. [Online]. Available: http://www.skyhookwireless.com/

[3] Navizon peer-to-peer wireless positioning. [Online]. Available: http://www.navizon.com/

[4] P. Bahl and V. N. Padmanabhan, "RADAR: An In-building RF-based User Location and Tracking System," in *IEEE INFOCOM*, vol. 2, 2000, pp. 775–784.

[5] T. Roos, P. Myllymäki, H. Tirri, P. Misikangas, and J. Sievänen, "A Probabilistic Approach to WLAN User Location Estimation," *Int'l Journal of Wireless Information Networks*, vol. 9, no. 3, pp. 155–164, 2002.

[6] V. Seshadri, G. V. Záruba, and M. Huber, "A Bayesian Sampling Approach to In-door Localization of Wireless Devices Using Received Signal Strength Indication," in *IEEE PERCOM*, 2005, pp. 75–84.

[7] M. Brunato and R. Battiti, "Statistical Learning Theory for Location Fingerprinting in Wireless LANs," *Computer Networks*, vol. 47, no. 6, pp. 825–845, 2005.

[8] 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*, vol. 18, no. 9, pp. 1181–1193, 2006.

[9] S.-P. Kuo, B.-J. Wu, W.-C. Peng, and Y.-C. Tseng, "Cluster-Enhanced Techniques for Pattern-Matching Localization Systems," in *IEEE Int'l Conf. on Mobile Ad-hoc and Sensor Systems (MASS)*, 2007.

[10] 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*, vol. 20, no. 5, pp. 678–684, 2008.

[11] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Impact of Radio Irregularity on Wireless Sensor Networks," in *ACM MobiSys*. ACM Press New York, NY, USA, 2004, pp. 125–138.

[12] S.-P. Kuo, "Design and implementation of pattern-matching localization in large-scale wireless networks," Ph.D. dissertation, National Chiao Tung University, 2008.

[13] E. K. Chong and S. H. Zak, *An Introduction to Optimization*, 2nd ed. John Wiley adn Sons, Inc, 1995.

[14] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *in Proceedings of the 23rd ACM National Conference*. New York, NY, USA: ACM Press, 1968, pp. 517–524.

[15] A. Civilis, C. S. Jensen, and S. Pakalnis, "Techniques for efficient road-network-based tracking of moving objects," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 5, pp. 698–712, 2005.

[16] Mobile platform software development kit 1.3. [Online]. Available: http://ossmpsdk.intel.com/index.php

[17] S.-P. Kuo, S.-C. Lin, B.-J. Wu, Y.-C. Tseng, and C.-C. Shen, “GeoAds: A Middleware Architecture for Music Service with Location-Aware Advertisement,” in *IEEE Int’l Conf. on Mobile Ad-hoc and Sensor Systems (MASS)*, 2007.