Cluster Filtered KNN: A WLAN-Based Indoor Positioning Scheme

Jun MA, Xuansong LI, Xianping TAO and Jian LU
State Key Laboratory for Novel Software Technology
Department of Computer Science and Technology, Nanjing University
NO.22 Han Kou Road, Nanjing 210093, China
{majun, lixs, txp} @ics.nju.edu.cn, lj@nju.edu.cn

Abstract

Location Based Service (LBS) is one kind of ubiquitous applications whose functions are based on the locations of clients. The core of LBS is an effective positioning system. As wireless LAN (WLAN) costs less and is easy to access, using WLAN for indoor positioning has been widely studied recently. K Nearest Neighbors (KNN) is one of the basic deterministic fingerprint based algorithms and widely used for WLAN-based indoor positioning. However, KNN takes all the nearest K neighbors for calculating the estimated result, which could be improved if some selective work could be done to those neighbors beforehand. In this paper we propose a new scheme called "Cluster Filtered KNN" (CFK). CFK utilizes clustering technique to partition those neighbors into different clusters and chooses one cluster as the delegate. In the end, the final estimate can be calculated only based on the elements of the delegate. With experiments, we found that CFK does outperform KNN.

1. Introduction

With the rapid development of wireless communication technology and the popularity of various new digital devices, study on ubiquitous computing has been getting more and more attention. Applications in ubiquitous computing environment have some new characteristics, such as mobility and self-adaptability. *Location Based Service (LBS)* is one kind of these applications whose functions are based on the locations of clients, such as kid's safety care system [1] and location-aware information retrieval system [2]. Positioning System is the core of *LBS*. Its precision dramatically affects the quality and effectivity of *LBS*.

Even though Global Position System (GPS) has been widely used for positioning purpose, it is virtually impossible to locate an object in an indoor environment. Many techniques, such as Infrared Ray (IR), Ultrasonic, RFID and Bluetooth, have been utilized for indoor positioning. ORL's Active Badge System [3] makes use of IR. But as it requires line-of-sight pass between client's Badge and sensors, the application scope of the system is limited. The Cricket Location Support System [4] and The Bat location system [5] deduce the position of a client based on the propagation time of ultrasonic. Systems of this kind could get extremely quite good position result [5]; while on the other hand, large-scale basic infrastructures (Beacons, Listeners, Receivers etc.) have to be deployed, resulting in extra massive cost. LANDMARC [6] adopts RFID to locate indoor clients. However, the high price of RFID reader makes this technology hard to be widely used in practice at this moment. BIPS [7] proposes an indoor positioning system based on the Bluetooth technology

Wireless LAN (WLAN) costs less and is easy to use. In recent years, there has been an increasing awareness of the potential of using WLAN for indoor positioning, and many WLAN-based laboratorial systems have been developed [8-17]. *K Nearest Neighbors (KNN)* is one of the basic fingerprint based algorithms and generally used for WLAN-based indoor positioning. However, *KNN* takes all the nearest K neighbors into account while calculating the estimated result, which can be improved if some selective work could be done to those neighbors beforehand. In order to deal with this problem and to improve the precision, in this paper we propose a new scheme called "*Cluster Filtered KNN*" *(CFK)*, which uses clustering to filter out some of the neighbors. With experiments, we found that, to some extent, *CFK* does perform better than

978-1-4244-2100-8/08/$25.00 ©2008 IEEE

KNN in most cases.

This paper is organized as follows: Section 2 briefly introduces the keystones and main methods of indoor positioning based on WLAN and some related works. Section 3 presents our "*Cluster Filtered KNN*" *(CFK)*. After demonstrating the experimental results and a demonstration application in Section 4，we conclude this paper and provide an outlook of future work in Section 5.

2. Background and Related Works

Generally speaking, WLAN is not made for the purpose of positioning, but for wireless communication and data transmission. However, with the *Received Signal Strength (RSS)* of each wireless access point (AP), the location of a mobile client can be deduced. To date, nearly all of indoor positioning systems using WLAN are based on *RSS*, and according to [8][10] and [17] they could be essentially divided into two categories: propagation-based and training-based (or fingerprint-based).

With the geometry information of the building, propagation-based positioning systems (.e.g. [8]) first use a signal propagation model to convert *RSS* to a distance measurement; and then apply some methods (.e.g. trilateration) to compute the location of a mobile client. Nevertheless, the signal attenuation of each AP is not only dependent on the distance, but is also affected by many environmental factors (.e.g. people, walls, and humidity) [14]. As a result, it is impractical to obtain such a general signal propagation model that simulates the real world situation well enough, and then the precision declines.

To get better precision, training-based technology gets more and more attention. Normally, systems of this kind consist of two main phases: training phase and estimation phase. The object of training phase is to build a fingerprint (or sample space) database. To generate the database, *Reference Points* (RPs) in the area where the positioning system covers are carefully selected at the very beginning. Then, at each RP, *RSSs* of all APs are collected and converted to a certain form and finally recorded in the database. In the *estimation* phase, to locate a client at a given place *L*, the system measures the *RSS*s of all APs at *L,* and then compares the collected *RSS*s with the data recorded in the database according to a given algorithm to evaluate the location of *L*.

Compared to propagation-based model, using fingerprint can avoid the hard work of obtaining a sufficient good and general propagation model.

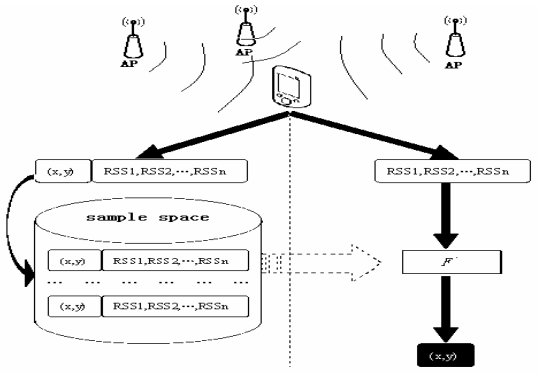


Figure 1. Deterministic training-based approach

However, fingerprint-based way has its inherent disadvantages: the significant overhead for building the sample space, and furthermore, when the environment changes, the outdated sample space might result in poor system performance. Many studies have been done to deal with this problem. For instance, [15] sets up a fixed differential correction base station in the same environment to compensate for the influence of susceptible radio frequency on the user device. The approach taken by [13] is to use different inter-RP distances for different areas according to the importance of each area. Yet [12] takes advantage of *Inverse Distance Weighting (IDW)* and *Universal Kriging (UK)* to automatically generate a large-scale sample space from a relative small one, which could save 25%-50% building cost without decreasing the system accuracy.

Some training-based systems apply a probabilistic approach, which construct a probability distribution over the target's location for the physical area making up the environment [16]. Most other systems [11][12][13][17] choose the deterministic approach(see Fig.1), where the entries (or samples) in the fingerprint database are generally in the form of "$\{RP_i.x, RP_i.y, RSSV_i\}$". Here $RP_i.x$ and $RP_i.y$ stand for the location of RP_i, and $RSSV_i$ is a vector "$\{RSS_{i1}, RSS_{i2}...RSS_{ij},...,RSS_{in}\}$", where RSS_{ij} refers to the average *RSS* of AP_j at RP_i and n is the number of APs used. Systems of this kind generally utilize "*K Nearest Neighbors (KNN)"* or its variation "*K Weighted Nearest Neighbors (KWNN)"* as the algorithm F (see Fig.1) .With *KNN*, K (K≥2) nearest neighbors are firstly chosen from the fingerprint according to the Euclidean distance between the *RSSV* of each *SP(Sample Point)* and the one got at the client's location L:

$$Dis(RSSV_i, RSSV_L) = \sqrt{\sum_{k=1}^{n} (RSS_{ik} - RSS_{Lk})^2}$$

The (weighted) average of the coordinates (x,y) of these K neighbors can be then calculated as the estimation of the mobile client's location. If K=1, *KNN* turns to *Nearest Neighbors (NN)*.

As the signal attenuation of wireless AP is influenced by many factors, there might be chances that the *RSSV*s obtained at two different locations are almost the same. Obviously, using *NN* which considers only the *SP* with the nearest *RSSV*, the estimated location could be far from the actual, while *KNN* takes more nearby ones and could get a better result. In order to illustrate this, Figure 2(a) gives a concrete example. Given the client's location *A* and the corresponding $RSSV_A$. And *n1, n2, n3* are the locations of nearest (according to the *RSSV* distance) three **SP**s *(SP_1, SP2, SP3)*, where the following condition holds:

$$Dis(RSSV_A, SP_1.RSSV) < Dis(RSSV_A, SP_2.RSSV) < Dis(RSSV_A, SP_3.RSSV)$$

Then the estimated location of *A* would be *n1* if *NN* is used, whereas *KNN* (K=3) could get an intuitively better one, *e1*.

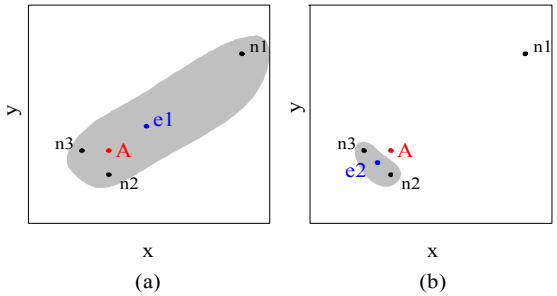


Figure 2. K-nearest neighbors

However, *KNN* takes all the nearest K neighbors to calculate the estimated result, while not all of them do contribute. If we could select some of these K neighbors before calculation, a more accurate estimate could be found. As shown in Figure 2(b), if we could construct a sample group with respect to the grey area, we could get a much better estimate *e2*. Being aware of this, we propose a new scheme called *"Clustering Filtered KNN" (CFK)*, which uses clustering to filter out some of the useless neighbors.

3. CFK

For a given location *L* and the corresponding $RSSV_L$, the process of *CFK* is presented in Table 1. *CFK* first gets the K neighbors as same as *KNN* does. Yet unlike *KNN* taking all the nearest K neighbors (*SP*s) into account for calculating the estimated result, *CFK* uses clustering technique (according to the physical location of each neighbor) to partition these neighbors (*SP*s) into several disjoint clusters and only one cluster is chosen as the delegate while the others are filtered out. Finally, the estimate is calculated based on the elements of the delegate.

Table 1. Clustering Filtered KNN

1) Go through the sample space ***S***, find the set $\boldsymbol{K\_Set_L}$ which consists of the nearest K samples.
2) Apply some clustering algorithm ***CF*** to $\boldsymbol{K\_Set_L}$, partition $\boldsymbol{K\_Set_L}$ into several *clusters*.
3) According to the selecting rule set ***R*** , select one *Cluster* as delegate, *say* ***C****;*
4) Take the average of the coordinates (x,y) of all samples in C as the estimate the location (x,y) :

$$L.x = \frac{\sum_{SP_i \in C} SP_i.x}{|C|} \qquad L.y = \frac{\sum_{SP_i \in C} SP_i.y}{|C|}$$

ps: $|\boldsymbol{C}|$ stands for the cardinality of ***C***

There are many clustering algorithms [18], while *Hierarchical Clustering* [19] is a basic and classic one. Moreover, it could be distinguished into two main types: *Agglomerative* and *Divisive*. In *agglomera*tive clustering, each object is initially placed in its own group. The "nearest" pair of groups is then merged into a single group. This procedure continues until a threshold distance is reached. Table 2 shows the basic steps of *agglomerative HC*. On the other hand, *Divisive* h*ierarchical clustering* does the reverse by starting with all objects in one group and subdividing them into smaller pieces.

Table 2. Hierarchical Clustering

1) Place each object initially in its own group.
2) Combine the two "closest" groups into a single group.
3) Recalculate the distance for each pair of groups.
4) If the minimum distance $\boldsymbol{d}_{\min}$ is greater than the threshold ***T***, stop; otherwise, repeat step2 and

With the difference of measures of the distance between two groups (Table 2, step3), *agglomerative Hierarchical Clustering* can be further divided into several subtypes: *single-linkage,* where the distance between two clusters is measured by the shortest distance from any element of one cluster to any element of the other one, *complete-linkage,* which chooses the greatest element distance, and *average-linkage,* which uses the average element distance.

4. Experiments and Analysis

To show the effect of *Clustering Filtered KNN (CFK)*，we build a test bed in our lab, where the blueprint is presented in Figure 3. We deploy five *Linksys-WAP54G-CN* wireless APs (AP1, AP2, AP3, AP4 and AP5) and select totally 70 *Reference Points* over room C, D, F and hallway while the distance between each two neighbor RPs with in the same area (hallway, room C, D or F) is φ=1.2 meters. And two *HP iPAQ hx2700* PDAs are used to record the *RSSs* of APs.

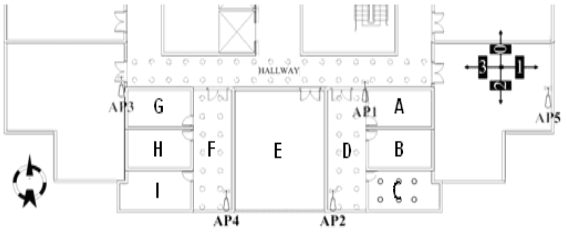


Figure 3. Test bed

During the *Sample Space Construction* phase, we collect *RSSs* from all APs (every 500ms) concurrently at each reference point RP_i. For a certain AP_j, we record 20 *RSS*s, and take the average of them as the representative "RSS_{ij}". With the representatives of all APs, the *RSS* vector $RSSV_i\{RSS_{i1},RSS_{i2},...,RSS_{ij},...,RSS_{in}\}$ could be constructed. Finally, put the vector and the actual location of the RP together, we get one sample $SP\{RP_i.x, RP_i.y, RSSV_i\}$ for RP_i. As the direction influences *RSS*, four directions (as 0,1,2,3 shown in Fig.3) are taken into account. Then, 4 samples could be gathered correspondingly per RP, and a sample space with 280 samples is constructed for all the 70 RPs.

To generate test cases, we randomly select two directions at every RP; and for each direction, we collect 10 *RSS*s of every AP periodically (every 500ms). Just like the construction of a sample, we get the $RSSV_i$ vector and one test case $TC_i=\{RP_i.x, RP_i.y, RSSV_i\}$. Therefore, 140 test cases are obtained for all 70 RPs.

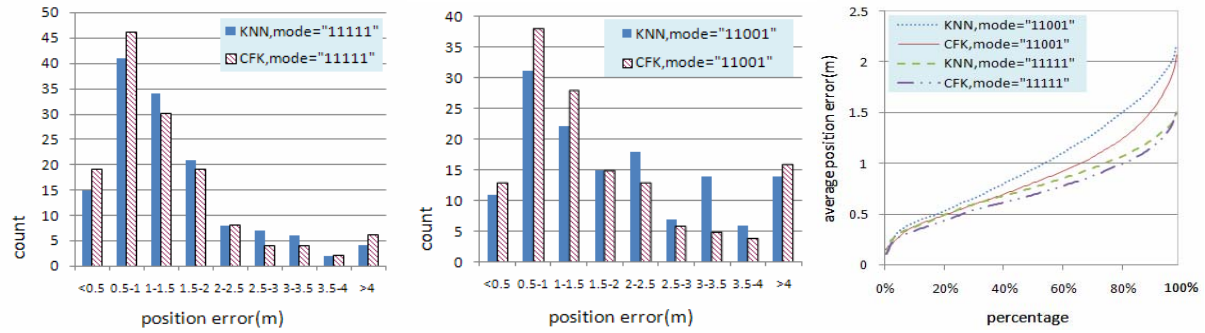
For the purpose of showing the influence of the num of APs, we test subsets of all the five APs in our experiments, and we call each subset as a "mode". For example, mode"11001" means only the *RSSs* of AP1, AP2 and AP5 are used to deduce the estimated location，while mode "11111"stands for using *RSSs* of all the five APs.

4.1. Effect and Efficiency

As shown in Table 1, *CFK* is mainly determined by two parts: the clustering algorithm *CF* and the cluster selecting rule set *R*. In the experiments, we choose *average-linkage agglomerative Hierarchical Clustering (HC)* as the *CF*. At the same time, *R* is relatively simple, which consist of two rules: (1) *select*

Table 3. Average position errors of *KNN, CFK* (Unit=m)

K	SCHEME	mode="11111"				mode="11001"			
		20%	60%	95%	100%	20%	60%	95%	100%
2	KNN	0.44	0.87	1.43	1.64	0.42	0.94	1.9	2.23
	CFK 1/2φ	0	0.58	1.32	1.57	0.09	0.95	2.15	2.55
	CFK 1φ	0	0.58	1.3	1.56	0.18	0.95	2.13	2.53
	CFK 2φ	0.16	0.68	1.37	1.61	0.31	0.92	2.08	2.48
	CFK 3φ	0.24	0.73	1.36	1.6	0.35	0.92	2.06	2.46
	CFK 4φ	0.31	0.77	1.36	1.59	0.37	0.92	2.04	2.44
	CFK 5φ	0.42	0.85	1.43	1.65	0.37	0.92	2.02	2.42
	CFK 6φ	0.42	0.85	1.41	1.64	0.37	0.91	2	2.4
3	KNN	0.45	0.79	1.34	1.54	0.47	1	1.91	2.2
	CFK 1/2φ	0	0.62	1.36	1.6	0.09	0.94	2.13	2.53
	CFK 1φ	0.12	0.61	1.27	1.51	0.25	0.92	2.03	2.38
	CFK 2φ	0.3	0.66	1.28	1.52	0.34	0.92	2.01	2.35
	CFK 3φ	0.43	0.74	1.33	1.58	0.43	0.93	1.93	2.28
	CFK 4φ	0.43	0.74	1.33	1.57	0.47	0.94	1.88	2.22
	CFK 5φ	0.43	0.77	1.34	1.56	0.47	0.96	1.88	2.25
	CFK 6φ	0.43	0.78	1.33	1.54	0.47	0.99	1.91	2.26
4	KNN	0.42	0.81	1.34	1.53	0.48	1.04	1.91	2.18
	CFK 1/2φ	0	0.63	1.36	1.6	0	0.85	1.96	2.31
	CFK 1φ	0.27	0.68	1.34	1.59	0.32	0.96	2.06	2.4
	CFK 2φ	0.4	0.74	1.36	1.6	0.38	0.85	1.9	2.25
	CFK 3φ	0.39	0.76	1.37	1.61	0.42	0.89	1.93	2.27
	CFK 4φ	0.36	0.75	1.34	1.58	0.45	0.91	1.84	2.19
	CFK 5φ	0.37	0.77	1.3	1.51	0.46	0.98	1.92	2.26
	CFK 6φ	0.37	0.77	1.31	1.52	0.46	0.99	1.92	2.26
5	KNN	0.47	0.8	1.31	**1.48**	0.51	1.06	1.9	**2.16**
	CFK 1/2φ	0	0.59	1.25	1.46	0	0.89	1.9	2.25
	CFK 1φ	0.3	0.69	1.29	1.52	0.37	0.99	2.03	2.37
	CFK 2φ	0.41	0.72	1.23	**1.43**	0.45	0.9	1.94	2.29
	CFK 3φ	0.41	0.73	1.26	1.47	0.46	0.92	1.91	2.25
	CFK 4φ	0.42	0.75	1.27	1.48	0.5	0.96	1.86	2.2
	CFK 5φ	0.44	0.78	1.29	1.47	0.51	0.98	1.85	2.19
	CFK 6φ	0.43	0.76	1.26	**1.43**	0.5	0.99	1.85	2.19
6	KNN	0.48	0.83	1.32	1.5	0.49	1.06	1.89	**2.16**
	CFK 1/2φ	0	0.66	1.33	1.54	0.04	0.89	1.86	2.2
	CFK 1φ	0.35	0.75	1.35	1.6	0.38	0.97	1.94	2.27
	CFK 2φ	0.42	0.74	1.3	1.51	0.46	0.88	1.81	2.15
	CFK 3φ	0.39	0.73	1.23	1.44	0.46	0.88	1.85	2.18
	CFK 4φ	0.42	0.76	1.28	1.49	0.47	0.9	1.81	2.15
	CFK 5φ	0.45	0.79	1.3	1.45	0.49	0.97	1.85	2.15
	CFK 6φ	0.46	0.8	1.31	1.44	0.47	0.96	1.82	2.15
7	KNN	0.49	0.85	1.34	1.52	0.53	1.1	1.91	2.17
	CFK 1/2φ	0	0.63	1.33	1.55	0.13	0.94	2.05	2.41
	CFK 1φ	0.35	0.71	1.26	1.48	0.43	0.97	1.94	2.28
	CFK 2φ	0.42	0.72	1.29	1.51	0.44	0.86	1.77	2.1
	CFK 3φ	0.42	0.75	1.28	1.49	0.48	0.9	1.79	2.12
	CFK 4φ	0.44	0.78	1.29	1.5	0.49	0.92	1.74	2.07
	CFK 5φ	0.47	0.82	1.29	1.48	0.5	0.96	1.75	2.07
	CFK 6φ	0.47	0.83	1.26	1.45	0.49	0.97	1.73	**2.06**
8	KNN	0.5	0.83	1.34	1.5	0.57	1.11	1.94	2.2
	CFK 1/2φ	0	0.64	1.32	1.55	0	0.9	2.09	2.45
	CFK 1φ	0.34	0.67	1.23	1.45	0.39	0.91	2.02	2.37
	CFK 2φ	0.42	0.74	1.27	1.49	0.48	0.87	1.86	2.21
	CFK 3φ	0.44	0.75	1.28	1.49	0.47	0.9	1.88	2.21
	CFK 4φ	0.44	0.78	1.28	1.49	0.47	0.88	1.77	2.11
	CFK 5φ	0.46	0.8	1.28	1.48	0.5	0.94	1.81	2.15
	CFK 6φ	0.48	0.81	1.27	1.44	0.51	0.96	1.82	2.15
9	KNN	0.51	0.82	1.35	1.51	0.57	1.12	1.97	2.21
	CFK 1/2φ	0	0.64	1.36	1.54	0	0.87	2.11	2.47
	CFK 1φ	0.35	0.74	1.37	1.55	0.41	0.92	2.2	2.55
	CFK 2φ	0.45	0.77	1.33	1.55	0.5	0.87	1.85	2.2
	CFK 3φ	0.47	0.78	1.36	1.6	0.45	0.85	1.86	2.21
	CFK 4φ	0.49	0.8	1.36	1.59	0.48	0.89	1.81	2.16
	CFK 5φ	0.49	0.81	1.3	1.56	0.51	0.94	1.81	2.16
	CFK 6φ	0.51	0.82	1.28	1.53	0.5	0.95	1.82	2.17



(a) Error distribution, mode= "11111" (b) Error distribution, mode= "11001" (c) Average position error

Figure 4. Experimental results
T=4φ, K=7, mode= "11111", "11001"

the cluster with more elements; (2) *select the cluster with smaller average RSSV distance if two clusters' cardinalities are equal*.

Table 4. Median position errors (Unit=m)

mode		11001								11111							
scheme \ k		2	3	4	5	6	7	8	9	2	3	4	5	6	7	8	9
KNN		1.61	1.78	1.87	1.93	1.87	2	1.98	1.96	1.37	1.23	1.33	1.35	1.33	1.37	1.29	1.35
CFK	*1/2φ*	*1.7*	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.2	1.2	1.2	1.2	1.2	1.2	1.2	*1.7*
	1φ	*1.7*	1.7	*1.8*	1.91	*1.9*	1.7	1.7	1.8	1.33	*1.32*	1.33	*1.36*	*1.38*	1.29	1.28	*1.36*
	2φ	*1.7*	*1.69*	1.63	1.7	1.6	1.55	1.55	1.41	1.33	1.2	1.22	1.14	1.28	1.18	1.25	1.24
	3φ	*1.7*	1.57	1.64	1.7	1.49	1.51	1.6	1.41	1.34	1.21	1.29	1.18	1.28	1.18	1.15	1.23
	4φ	*1.65*	1.57	1.59	1.7	1.57	1.49	1.51	1.51	1.34	1.2	1.29	1.28	1.29	1.26	1.25	1.28
	5φ	*1.65*	1.65	1.73	1.74	1.67	1.69	1.66	1.69	1.37	1.22	1.29	1.3	1.29	1.26	1.27	1.33
	6φ	1.61	1.74	*1.81*	1.76	1.67	1.69	1.68	1.7	1.37	1.22	1.29	1.28	1.3	1.27	1.26	1.35

Table 4 shows the ***medians*** of position errors got by *KNN* and *CFK* under many different situations. It's obvious that the position error medians got by *CFK* are generally better than those got by *KNN*.

Table 3 presents the ***average position errors*** (***APE***) of the best 20%, 60%, 95% and 100% test cases[①].We can find that nearly all the ***APE***(20%)s and ***APE***(60%)s and most of the ***APE***(95%)s got by *CFK* show improvements over the counterparts of *KNN,* especially when *K* is great enough. To make this more clear，we present a concrete example in Figure 4 and Table 5, where *T*=4*φ*, *K*=7.

As shown in Figures 4(a) and (b), on the one hand,

Table 5. Improvements of *CFK*
T=4φ, K=7, mode= "11111", "11001"

SCHEME	mode="11111"				mode="11001"			
	20%	60%	95%	100%	20%	60%	95%	100%
KNN	0.49	0.85	1.34	1.52	0.53	1.1	1.91	2.17
CFK(T=4φ)	0.44	0.78	1.29	1.5	0.49	0.92	1.74	2.07
Improvements	10.20%	8.24%	3.73%	1.32%	7.55%	16.36%	8.90%	4.61%

① The experimental results of all test cases are arranged in ascending order according to their position errors; "the average position errors of the best 20% test cases" means the average of the first 20% ordered test cases' position errors. We use the form "***APE***(20%)" as short for it.

CFK improves most of the test cases' position precisions and obviously increases the number of test cases whose position errors are relatively small; while on the other hand, it also increases a few test cases' position errors. After all, the average position error got by *CFK* is desirable (Fig.4(c)).

To show the efficiency of *CFK*, Table 6 illustrates the execution times of both the schemes with *K*=2-9. The tests are carried out on a *lenovo* notebook PC with one *Intel Pentium M processor* (1500MHz) and 512M memory, while the OS is *Windows XP Professional Edition with SP2*. The schemes are executed 400 times for all 140 test cases, and the total computation times are gathered as shown in Table 6. From the results, we could make the following comments. First, with the growing of *K*, the computation times increase accordingly for both the schemes. This is caused by the increasing time of finding the *K* nearest neighbors. Second, *CFK* takes more time than *KNN* does, and this extra time, say *Γ*, is the very time introduced by the *CF* algorithm. Furthermore, Γ is no more than few tens of milliseconds for all 140 test cases (about 1/10 millisecond (s) for each one), and is not noticeable when compared with the time for collecting *RSS* and building K_Set_L. What is more important is that the average extra time for each test case is not affected by the size of the sample space, it is just a function of *K*.

Table 6. Total computation times for running all 140 test cases 400 times (Unit=second)

mode		11001								11111							
scheme \ k		2	3	4	5	6	7	8	9	2	3	4	5	6	7	8	9
KNN		345	345	344	346	346	347	347	348	350	350	350	352	351	353	352	353
CFK	*1/2φ*	345	348	348	350	350	352	356	357	352	353	353	355	356	358	361	363
	1φ	346	347	348	350	351	354	357	359	352	352	354	355	357	360	363	365
	2φ	346	347	349	350	352	355	357	361	351	353	354	356	358	360	363	367
	3φ	346	347	349	350	352	356	357	361	351	353	354	355	359	361	363	366
	4φ	345	347	349	350	352	354	358	361	351	353	355	356	359	361	363	367
	5φ	345	347	348	350	353	355	357	360	350	352	354	356	358	361	363	366
	6φ	345	347	349	350	353	355	358	362	351	353	354	356	358	360	364	366

Third, *CFK* takes more time in mode "11111". As in mode "11001" where less APs are used, the corresponding K_Set_L gathered for a given test case *TC* is relatively more dispersive, and then it takes less time for d_{min} to get greater than T (see Table 2, step4). Fourth, the threshold T doesn't influence the execution time so much.

4.2. Discussion

In this section, we illustrate some of the factors affecting the performance of *CFK*. The first one is ***the Number of APs.*** From our experimental results shown above, it's quite clear that (1)the results of using five APs are much better than those of using three APs and (2)the effect of *CFK* is more obvious in mode "11001" than that in mode "11111". As more APs are used, for a given test case *TC*, the corresponding K_Set_L is then relatively more centralized and accurate, and so be the final estimate. Moreover, running clustering on such a centralized K_Set_L will not make any significant improvements.

The second factor is ***the Number of Neighbors (K).*** As *K* increases, the results (averages of all test cases) of both *KNN* and *CFK* generally become better at first, but then get worse. When *K* is small, only few neighbors with short *RSSV* distances are included in K_Set_L, then the estimate based on such a small K_Set_L is relatively more sensitive to mismatching of neighbors. The case gets worse if less APs are considered, where the possibility of mismatching is greater. When *K* grows, more neighbors are included in K_Set_L, and some mismatched neighbors might be filtered out by clustering when using *CFK*. Especially, when less APs are used, the *K* neighbors might be more likely to be dispersive, and the advantages of *CFK* are much more remarkable. However, the number of *"real"* neighbors of one test case is limited. IF *K* grows greater than a given threshold, then more "*fake*" neighbors (samples that are actually not so close to the location of the test case) will be selected in K_Set_L. As a result, the estimated location would become less accurate. That is why at 100% test cases, the proposed scheme does not perform well than *KNN* when K=9. Based on our experiments, we conclude that *CFK* works relatively well when *K=5, 6, 7, 8* for mode "11111" and when *K=6, 7, 8* for mode "11001".

The Threshold (T) used in *HC* is another main factor. In our experiments, the selecting rule set R is mainly based on the cardinality of each cluster. When the threshold of *CFK* is small, K_Set_L will be divided into many small clusters. If we are lucky enough, the cluster with the greatest cardinality is the very closest one to the test case, then the final estimated location of one test case could be quite accurate, however the cardinalities of these clusters are ordinarily small and sometimes identical, which may increase the possibility of mischoosing of the delegate cluster and result in a poor overall precision. Take $T=0.5\varphi$ for instance, almost all the ***APE***(20%)s under different situations ($K=2$ *to* 9) are nearly 0 in both modes, while the ***APE***(100%)s are generally worse than their counterparts of *KNN*.

Cases turn different if the threshold becomes greater. In this case, small clusters gathered above with a small T could be further grouped into some bigger ones. This makes the clusters' cardinalities more variant, and reduces the feasibility of mischoosing, which results in the improvement of the overall precision. However, as more elements are included in the delegate cluster, many TCs' estimated locations might be not far from the corresponding results got by *KNN* and some extreme cases may even be identical. As shown in Table 3, when $T=6\varphi$, the ***APE***(100%)s are usually better than those of *KNN* for both modes, whereas the ***APE***(20%)s and ***APE***(60%)s are slightly improved when compared with their counterparts.

From our experimental results, we have the following observations. First, when more APs are used, there are less constraints on threshold range, it could be set among $[2\varphi, 6\varphi]$, while values among $[2\varphi, 4\varphi]$ are more preferred, because running *CFK* with such a threshold could not only keep the overall average error low but also improve the position precisions of many test cases. Second, when less APs are used, the threshold should be set a little greater ($[4\varphi, 6\varphi]$) owing to the dispersive distribution of neighbors in K_Set_L.

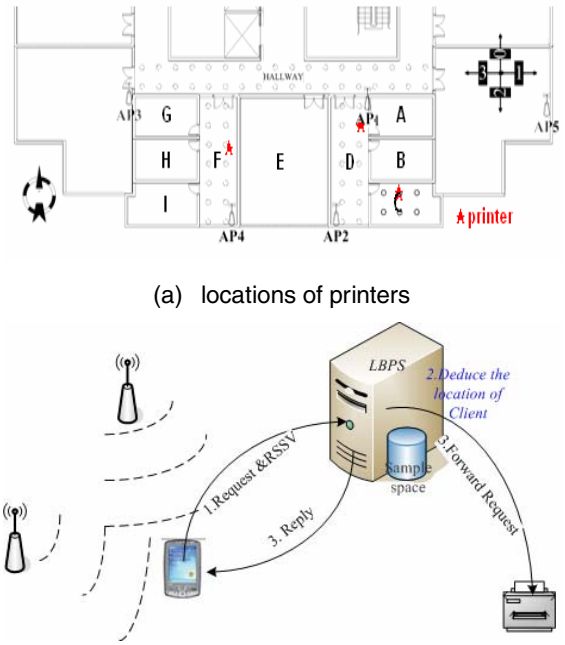
Besides the main factors mentioned above, there are some other factors which might influence the proposed scheme. Take *the distribution of elements in* K_Set_L for instance, as K_Set_L is constructed through the similarity of *RSSV*, the elements of it might be far from the test case or be rather dispersive, resulting in a drifting result even after *CFK*. Furthermore, *the characteristic of clustering algorithm CF* affects as well**.** For example, we have also tested *K-means Clustering* as *CF* (not included in this paper). However, due to its stiffness, the results are not as obvious as those of *Hierarchical Clustering*. In addition, the *cluster selecting rule set* R applied in our experiments is mainly based on the cardinality of each cluster and is relatively simple. However, many other measurements (.e.g. the density of each cluster) and their combinations might be desirable as well.

4.3. Demonstration Application

Based on our proposal, we've developed a simple demonstration application as shown in Figure 5. The main purpose is to provide printing service for a moving client, and we call this application as a *location base print service (LBPS)*.

Three LAN-connected printers are installed in room C, D and F separately. The locations of the printers are stored in a *LBPS* server, which holds the sample space as well. A client wanders around with a PDA, and wants a document being printed, so the client submits the printing request accompanied with the current *RSSV* gathered by the PDA to the *LBPS* server. Upon receiving the printing request and *RSSV*, the server utilizes the proposed scheme *CFK* to deduce the location of the client. With the estimated location, it forwards the request to the nearest printer and at the same time it sends a reply message to the client showing on which printer the document is being printed.

With respects to our experiments, we find that the positioning precision provided by *CFK* is good enough for this application and that the *LBPS* system works fine.



(a) locations of printers

(b) LBPS

Figure 5. Application scenario

5. Conclusion and Future Work

The performances of *LBS* applications are mainly dependent on the precisions of their *Positioning Systems*. As *Wireless LAN (WLAN)* costs less and is easy to access when compared with other indoor positioning techniques (IR, Ultrasonic, RFID, etc.), using WLAN for indoor positioning has become one hotspot of research.

In this paper, we briefly introduced the keystones and main methods of WLAN-based indoor positioning. We further pointed out the typical scheme *KNN* could be improved if some selective preprocessing could be done. To validate this, we proposed a scheme called *"Clustering Filtered KNN"*, which utilizes clustering to filter out some of the neighbors. And finally, with the experimental results, we showed that *CFK* does outperform *KNN* in most cases.

Our future work will focus on following directions. First, try to apply *Inverse Distance Weighting* (*IDW*) and *Universal Kriging*(*UK*)[12] to reduce the work of building sample space and generate enough fine grained samples, which might make the use of clustering more remarkable and improve quality of clusters got by *CFK*. Second, attempt some other clustering algorithms as well as other techniques other than clustering. Third, search after other measurements and combine some of them to dig out fine enough cluster selecting rule set ***R***. Fourth, make efforts to find a way to rank the *RSSs* from different APs. Through our experiments, the *RSSs* from all AP have the same priorities. We expect that the *RSSs* from some APs might contribute more to the positioning progress while some others do not.

Based on these work, we will make efforts to establish a reliable enough indoor positioning system and finally put it in use for our further study or everyday life.

Acknowledgements

The work of this paper is funded by 863 Program of China (2007AA01Z178, 2007AA01Z140), NSFC (60736015, 60721002) and NSFJ (BK2006712).

References:

[1] K. Takata, J.H. Ma, and B.O. Apduhan, "A Dangerous Location Aware System for Assisting Kids Safety Care", *Proceedings of AINA 2006, Volume 1*, 2006, pp. 657-662.

[2] J.N. Cao, K.M. Chan, G.Y.K. Shea, M.Y. Guo, "Location-Aware Information Retrieval for Mobile Computing", *Proceedings of EUC 2004*, In: L.T. Yang et al.(Eds.) *Lecture Notes in Computer Science, Vol. 3207*, Springer-Verlag, Aizu-Wakamatsu City, Japan, August 2004, pp.450-459.

[3] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The Active Badge location system", *ACM Transactions on Information Systems*, *Volume 10, Issue 1*, ACM Press, January 1992, pp. 91-102.

[4] N.B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket location-support system", *Proceedings of MOBICOM 2000*, Boston, August 2000, pp. 32–43.

[5] A. Harter, A. Hopper, P. Steggles, A.Ward, and P. Webster, "The anatomy of a context-aware application", *Proceedings of MOBICOM 1999*, Seattle, August 1999, pp. 59–68.

[6] L.M. Ni, Y. Liu. Y.C. Lau, and A.P. Patil, "LANDMARC: Indoor Location Sensing Using Active RFID", *Proceedings of IEEE PerCom 2003*, Dallas, Texas, March 2003, pp. 407- 415.

[7] R. Bruno and F. Delmastro, "Design and Analysis of a Bluetooth-Based Indoor Localization System", *Proceedings of PWC 2003*, Venice, Italy, September 2003, pp. 711-725.

[8] P. Bahl and V.N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system", *Proceedings of IEEE INFOCOM 2000*, Los Alamitos, CA, March 2000, pp. 775-784.

[9] H. Lim, L.-C. Kung, J.C. Hou, and H. Luo, "Zero-configuration, robust indoor localization: Theory and experimentation", *Proceedings of IEEE INFOCOM 2006*, Barcelona, April 2006, pp. 1-12.

[10] M. Emery and M.K. Denko, "IEEE 802.11 WLAN based Real-time Location Tracking in Indoor and Outdoor Environments", *Proceedings of CCECE 2007,* Vancouver, April 2007, pp. 1062-1065.

[11] R. Zhou, "Architecture and Implementation of Indoor Wireless Position System", MASTER'S THESIS, Institute for Computer Science, University of Freiburg, Germany, January 2005.

[12] B. Li, Y. Wang, H.K. Lee, A.G. Dempster, and C. Rizos, "Method for yielding a database of location fingerprints in WLAN**",** *IEE Proceedings-Communications, Volume 152, Issue 5*, October 2005, pp. 580-586.

[13] P. Prasithsangaree, P. Krishnamurthy, and P. Chrysanthis, "On Indoor Position Location with Wireless LANs", *Proceedings of IEEE PIMRC 2002, volume 2,* Lisbon, Portugal, September 2002, pp. 720-724.

[14] Y.-C. Chen, J.-R. Chiang, H.-H. Chu, P. Huang, and A.W. Tsui, "Sensor-assisted wi-fi indoor location system for adapting to environmental dynamics", *Proceedings of ACM MSWiM 2005*, Montreal, Quebec, Canada, October 2005, pp.118-125.

[15] Y. Wang, X. Jia, and C. Rizos, "Two New Algorithm for Indoors Wireless Positioning System (WPS)", *Proceedings of 17th Int. Tech. Meeting of the Satellite Division of the U.S. Institute of Navigation,* Long Beach, California, September 2004, pp. 1988-1997.

[16] V. Seshadri, G.V. Záruba, and M. Huber, "A Bayesian Sampling Approach to In-door Localization of Wireless Devices Using Received Signal Strength Indication", *Proceedings of IEEE PerCom 2005*, Kauai Island, Hawaii, March 2005, pp. 75-84.

[17] B. Li, J. Salter, A. G. Dempster, and C. Rizos, "Indoor Positioning Techniques Based on Wireless LAN", *Proceedings of AusWireless 2006*, Sydney, Australia, March 2006.

[18] J. B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations", *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics an*d Probability*, Volume 1*, University of California Press, Berkeley, 1967, pp.281-297.

[19] S. Johnson, "Hierarchical clustering schemes", *Psychometrika*, September 1967, pp. 241-254.