# An Enhanced Density-based Clustering Algorithm for the Autonomous Indoor Localization

Yaqian Xu, Rico Kusber, and Klaus David
Chair for Communication Technology, University of Kassel
Wilhelmshöher Allee 73, Kassel 34119, Germany
Email: comtec@uni-kassel.de

*Abstract*—**Indoor localization applications are expected to become increasingly popular on smart phones. Meanwhile, the development of such applications on smart phones has brought in a new set of potential issues (e.g., high time complexity) while processing large datasets. The study in this paper provides an enhanced density-based cluster learning algorithm for the autonomous indoor localization algorithm DCCLA (Density-based Clustering Combined Localization Algorithm). In the enhanced algorithm, the density-based clustering process is optimized by "skipping unnecessary density checks" and "grouping similar points". We conducted a theoretical analysis of the time complexity of the original and enhanced algorithm. More specifically, the run times of the original algorithm and the enhanced algorithm are compared on a PC (personal computer) and a smart phone, identifying the more efficient density-based clustering algorithm that allows the system to enable autonomous Wi-Fi fingerprint learning from large Wi-Fi datasets. The results show significant improvements of run time on both a PC and a smart phone.**

***Keywords- Time complexity of algorithms; Run time of algorithms; Density-based clustering algorithm; Fingerprinting-based indoor localization.***

## I. INTRODUCTION

A user's location is a powerful context for many emerging location-based applications. Locating users by utilizing smart phones has become an essential function. Among the diverse approaches of indoor localization, detecting a user's location from the Wi-Fi radio environment is regarded as a promising option, one interesting approach being called Wi-Fi fingerprinting-based technique. The basic idea is to discover the Wi-Fi signal characteristics in certain locations to form the "fingerprints (*FPs*)" of these locations. The location is then recognized when the real-time measurement matches a particular fingerprint. In recent years, various fingerprinting-based indoor localization approaches, such as UnLoc [1], ARIEL [2], have been developed to make use of the smart phones to localize themselves.

Particularly, one of the potential indoor localization approaches on smart phones is called DCCLA (Density-based Clustering Combined Localization Algorithm) [3], which is proposed by ComTec, Kassel University. It is an autonomous indoor localization system on smart phones to learn fingerprints from Wi-Fi received signal strength indicators (*RSSIs*) from surrounding access points (APs) in an unsupervised manner.

The algorithm automatically discovers clusters and generates fingerprints from the raw Wi-Fi datasets directly without explicit pre-deployment effect of data annotation or a prior assumption about the number of clusters. For the building of a Wi-Fi fingerprint database, DCCLA performs density-based clustering, namely, DCCLA cluster learning algorithm [3] [4], to discover the "high-density" clusters. It works based on the assumption that the periodically collected Wi-Fi data are similar in a significant place (e.g., home, office), presenting a "high-density" distribution. Unlike the above-mentioned approaches [1] [2], DCCLA enables generating a complete Wi-Fi fingerprint database on a smart phone without communication to a server or a PC, with the consideration of privacy that people are not willing to share their locations, especially in their private lives.

DCCLA discovers Wi-Fi fingerprints based on density-based clustering. However, the density-based clustering algorithms typically have a fundamental limitation: high time complexity while dealing with large-scale data [5]. This limitation raises an efficiency problem of building fingerprints for DCCLA, because the size of a database containing the raw Wi-Fi datasets is likely to be large. The run time of the algorithm increases as the input dataset size increases. Thus, it is necessary to check time complexity and provide an efficient density-based clustering algorithm for indoor localization, especially when it is running on a smart phone.

In this paper, we propose an enhanced density-based cluster learning algorithm for DCCLA with improvements of the algorithm efficiency to build the fingerprint database. In the enhanced algorithm, the process of density-based clustering is optimized by "skipping unnecessary checks" and "grouping similar points".

We define evaluation terms and conduct a theoretical analysis of algorithm efficiency in terms of time complexity. To test our analysis results, we design experiments, which compare the algorithm run time on a PC (personal computer) and a smart phone. Datasets collected in a real-world scenario with multiple APs are used. As a result, we identify the more efficient density-based cluster learning algorithm that allows DCCLA to enable autonomous Wi-Fi fingerprint learning from large numbers of Wi-Fi datasets.

This paper is organized as follows. Section II introduces the core idea of DCCLA and the original DCCLA cluster learning algorithm. Section III elaborates possible improvements for the enhanced density-based cluster learning algorithm, as well as the time complexity of the original and the enhanced algorithm. In Section IV, the experimental comparisons and evaluations are given. Finally, the conclusion is presented in Section V.

MOBILWARE 2013, November 11-12, Bologna, Italy

DOI 10.4108/icst.mobilware.2013.254284

## II. INTRODUCTION OF THE ORIGINAL ALGORITHM

### A. DCCLA

Often, a person spends most of his time in a few specific places, such as "home", "office", "meeting room", "café" and so on, which are mostly significant for him. It is observed the RSSIs from a Wi-Fi AP are generally similar in a place. If a person stays in a place for a while, his smart phone scans the surrounding APs to record the RSSIs periodically. Thus, the RSSIs from an AP present a high "density" distribution. The high "density" characteristic offers us the opportunity to discover the fingerprints in an unsupervised manner by an autonomic system.

The autonomous indoor localization algorithm DCCLA is developed based on the "density" observation. It automatically learns the fingerprints of places where a person goes to, and recognizes them when he returns to these places. It has turned out DCCLA to be a promising option for indoor localization [6].

### B. DCCLA Cluster Learning Algorithm [3]

The key of automatically learning fingerprints in DCCLA is the cluster learning algorithm, which is a modified version of the density-based clustering algorithm DBSCAN [7] to suit the collected RSSI data type. It is used to discover the high-density RSSI range as clusters from collected RSSIs. To make this paper here self-contained, we present parts of the original cluster learning algorithm [3] necessary to understand the improvements.

**Definition 1**: (RSSI point) An RSSI point ($P_{ik}$) is a record when the smart phone scans the surrounding APs. It includes the timestamp ($t_k$), the MAC (Medium Access Control) address ($MAC_i$) of the Wi-Fi AP ($AP_i$) and the corresponding RSSI value ($RSSI_{ik}$).

**Definition 2**: (neighborhood of $P_{ik}$) A collection of RSSI points with the same MAC address as $P_{ik}$ within a neighborhood range ($NR$) is the neighborhood of $P_{ik}$, denoted by $N(P_{ik})$. Here, we define $P_{ik}$ itself belongs to $N(P_{ik})$ as well.

$$N(P_{ik}) = \{P_{im} \in D \mid 0 \le RSSI_{im} - RSSI_{ik} \le NR\} \ (i,k,m \in \mathbb{N}^*)$$

The *neighborhood range* ($NR$) is one parameter, used to delimit the range of the neighborhood starting from the RSSI point ($P_{ik}$). The unit of $NR$ is $dB$.

**Definition 3**: (neighborhood density) Neighborhood density of $P_{ik}$ is the number of RSSI points belonging to the neighborhood of $P_{ik}$, indicated by $\boldsymbol{\rho(P_{ik})}$.

Two parameters are introduced to determine the criterion of high density. One is the *neighborhood range (NR)* as introduced. Another is the *minimum number of RSSI points (MinPts)*. It is a natural number which is introduced to determine if $\rho(P_{ik})$ is high enough to create a cluster. For an RSSI point ($P_{ik}$), the neighborhood density is high enough to create a cluster if the following criterion is satisfied:

$$\rho(P_{ik}) \ge MinPts \ (i,k \in \mathbb{N}^*) \quad (1)$$

**Definition 4**: (cluster) A cluster ($\mathbf{C_i^o}$) is a collection of RSSI points within a range of high density, associated with $\mathbf{MAC_i}$, which can be indicated by:

$$C_i^o = \{MAC_i, RSSI_{ib}, RSSI_{ie}\} \ (o,i,b,e \in \mathbb{N}^*)$$

$RSSI_{ib}$ represents the beginning and $RSSI_{ie}$ represents the end RSSI value of the high-density range (e.g. $C_i^o = \{00{:}aa{:}00{:}62{:}c6{:}00, -56dB, -62dB\}$).

The range of a cluster is not delimited by the $NR$ of the RSSI point, which is used to create the cluster. After a cluster is created, the algorithm checks if the cluster can be extended by checking if the criterion of *cluster-extension* is satisfied.

**Definition 5**: (cluster-extension) A cluster ($\mathbf{C_i^o}$) can be extended if the two conditions are met:

- $P_{im} \in C_i^o$
- $\rho(P_{im}) \ge MinPts$

The neighborhood of $P_{im}$ is merged into the cluster ($C_i^o$). This cluster ($C_i^o$) is extended.

The basic idea to generate a cluster is as follows [3]. For each RSSI point $P_{ik}$, the algorithm calculates $\rho(P_{ik})$. If $\rho(P_{ik})$ is lower than $MinPts$, the algorithm continues to check the next unchecked RSSI point. Otherwise, $N(P_{ik})$ is either used to create a new cluster if $P_{ik}$ does not belong to any existing cluster, or merged to an existing cluster if $P_{ik}$ belongs to the existing cluster. The set of RSSI points not belonging to any cluster is defined as *noise*. The pseudo code of the DCCLA cluster learning algorithm is shown in Figure 1.

Input: collected RSSI points.
Output: a set of all learned clusters $\{C_i^o, \ldots, C_j^q\}$.

1) Separate the collected RSSI points into datasets, each dataset with a unique MAC address $MAC_i$.
2) Order each dataset to form a *list* $L_i$ with increasing RSSI values.
3) Label each *RSSI point* ($P_{ik}$) on each $L_i$ as unchecked
4) **for** each ordered $L_i$, **do**
    1) **while** there exist an unchecked $P_{ik}$, **do**
        1) Calculate the *neighborhood density of* $P_{ik}$ ($\rho(P_{ik})$)
        2) **if** $\rho(P_{ik})$ is smaller than *MinPts* , **then**
            1) Label $P_{ik}$ as checked.
            2) **continue** the while loop
        3) **else if** $P_{ik}$ belongs to an existing cluster $C_i^o$, **then**
            1) Merge neighborhood of $P_{ik}$ to $C_i^o$.
        4) **else**
            1) Create a new cluster $C_i^p$.
        5) **end if**
        6) Label $P_{ik}$ as checked.
    2) **end while**
5) **end for**

Figure 1 the cluster learning algorithm pseudo code [3]

## III. ENHANCED ALGORITHM DESCRIPTION AND ANALYSIS OF ALGORITHMS

### A. Evaluation Terms

Before presenting the possible improvements, we introduce related evaluation terms referring to the algorithm efficiency.

**Time complexity**: The time complexity of an algorithm quantifies the amount of time to perform an algorithm. It is determined in terms of its growth-rate that indicates how fast the algorithm's time requirement grows as the size of input grows [8]. Real world problems need to be solved as fast as possible and then output the desired results. A convention called *O*-notation (or "big-*O*-notation") is utilized to represent different time complexity classes that indicated how the time increases with the size of an input dataset ($n$).

**Run time**: The run time of an algorithm is the time needed to execute the algorithm. In the following experimental evaluation, we use *t(n)* to represent the run time of an algorithm, where $n$ is the size of the input dataset.

In order to emphasize the differences between the original and the enhanced algorithms, the input datasets used in the following algorithms are under the assumption of being processed by step 1), 2) and 3) in the DCCLA cluster learning algorithm pseudo code in Figure 1, containing lists of "unchecked" RSSI points with increasing RSSI values, separated by MAC addresses. The lists of "unchecked" RSSI points are regarded as the input datasets in the following time complexity analysis, while the outputs datasets are the learned clusters.

### B. Time Complexity of DCCLA Cluster Learning Algorithm

Given a dataset with $n$ RSSI points, the DCCLA cluster learning algorithm starts with the first point in the dataset and checks its neighborhood density. The density check is done for every RSSI point, starting from the beginning to the end until all clusters are found out. In this paper, the term DCCLA specially indicates the original DCCLA cluster learning algorithm.

The time complexity of the DCCLA is:

$$\boldsymbol{O}\ (\boldsymbol{n} * \boldsymbol{m})\ (\boldsymbol{n}, \boldsymbol{m} \in \mathbb{N}^*, \boldsymbol{m} \leq \boldsymbol{n})$$

$n$ is the size of the input dataset and $m$ is the average size of the neighborhood for all RSSI points. The growth-rate of the clustering algorithm is (n*m) that the run time of the algorithm increases as a function of the size of the input dataset. For the worst case, the time complexity is $O(n^2)$ as the value of $m$ equals to the value of $n$.

### C. Possible Improvement - Skipping Unnecessary Density Checks

To improve the algorithm efficiency, one possible solution is to reduce the amount of RSSI points to be checked. In other words, it reduces the frequency of density checks. In the original DCCLA, for points belonging to an existing cluster, the algorithm performs density checks on them to determine if the cluster can be extended. However, not all points are necessary to be checked. Among these points, there should be some critical points, which can be used to maximally extend the cluster. If the algorithm figures out critical points, only critical points are checked to maximally extend the cluster. Density checks for other points do not change the cluster extension result. Therefore, the unnecessary density checks can be skipped.

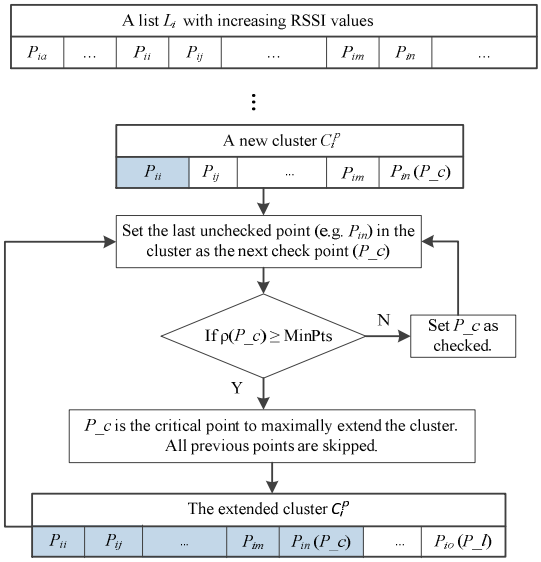


Figure 2 the flowchart of the algorithm $DCCLA_1$

An efficient method to find the critical points in an existing cluster is to perform the density check in a reverse sequence, starting from the last point of an existing cluster.

The algorithm with the improvement of "skipping unnecessary density checks" is denoted by $DCCLA_1$. The flowchart of $DCCLA_1$ is depicted in Figure 2. We consider a list $L_i$ with increasing RSSI values as an example input. The background blue color identifies the point being checked or skipped. $P\_c$ is used to mark the next check point, which is dynamically revised in the cluster learning process. The algorithm performs density checks starting from the last point. If the neighborhood density of $P\_c$ is higher than *MinPts*, it is the critical point to maximally extend the cluster. All previous points in the cluster are skipped. Otherwise, $DCCLA_1$ performs the density check in a reverse sequence, until either the cluster is extended, or the previous point is already checked.

The time complexity of $DCCLA_1$ is:

$$\boldsymbol{O}\ (\boldsymbol{o} * \boldsymbol{m})\ (\boldsymbol{o}, \boldsymbol{m} \in \mathbb{N}^*, \boldsymbol{m}, \boldsymbol{o} \leq \boldsymbol{n})$$

In the equation, $o$ is the number of checked RSSI points in the dataset. The time complexity of $DCCLA_1$ is $o/n$ times of that of the original DCCLA, where $n$ is the number of all RSSI points. From the equation we can see, the run time of $DCCLA_1$ depends on the number of checked RSSI points $m$, not on the number of input RSSI points $n$. The worst case efficiency of $DCCLA_1$ is equal to that of the original DCCLA when all RSSI points are actually checked ($o/n=1$). In most cases of our empirical study, $m$ is smaller than $n$. For the average case in our experimental evaluation, the value of $m$ is in the range $(0.5 * n \geq m \geq 0.1 * n)$.

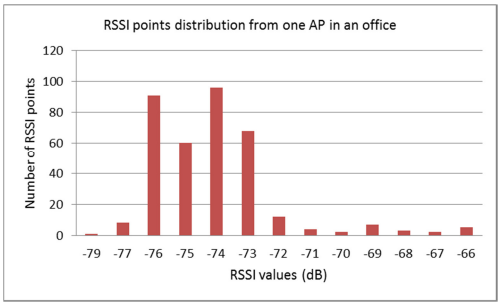


Figure 3 RSSI points distribution from one AP in an office

### D. Possible Improvement – Grouping Similar Points

If a user stays in a place for a while, the collected RSSI points from one AP are quite similar to each other. Figure 3 depicts the RSSI points distribution from a given AP (with $MAC_i$: 00:12:7f:ce:8f:a0) collected in 30 minutes in an office. 354 RSSI points are collected with RSSI values in the range of [-79*dB*, -66*dB*]. Thus, we pro-process input data by grouping similar points. The RSSI points with the same RSSI value are grouped as an RSSI group.

**Definition 6**: (RSSI group) An RSSI group ($G_{ik}$) is a collection of RSSI points, which have the same MAC address ($MAC_i$) and RSSI value ($RSSI_{ik}$). The duplication degree of an RSSI group is the number of RSSI points in the RSSI group. For example, for $G_{ik}$ with $RSSI_{ik}$=-76dB in Figure 3, the duplication degree is 92.

**Definition 7**: (neighborhood density of an RSSI group) The neighborhood density of $G_{ik}$ is the number of RSSI points in its neighbor groups, associated with the same MAC address and the neighborhood range.

It is not hard to figure out that the neighborhood density of an RSSI group is the same as the neighborhood density of any RSSI point in the RSSI group. The calculation is more efficient by using the duplication degree. The algorithm with the improvement of "grouping similar points" is denoted by $DCCLA_2$.

The basic idea of $DCCLA_2$ is to group the RSSI points and calculate the duplication degree. For each RSSI group $G_{ik}$, $DCCLA_2$ calculates the neighborhood density of $G_{ik}$. The criteria of creating or extending a cluster are the same as we introduced in Section II.

The run time complexity is:

$$\boldsymbol{O}\left(\frac{\boldsymbol{n}}{\boldsymbol{p}} * \frac{\boldsymbol{m}}{\boldsymbol{p}} + \boldsymbol{n}\right) (\boldsymbol{m}, \boldsymbol{n} \in \mathbb{N}^*, \boldsymbol{m} \leq \boldsymbol{n}, \boldsymbol{p} \geq \boldsymbol{1})$$

$p$ is the average duplication degree. At the example above, 354 RSSI points are distributed in 13 RSSI groups. The average duplication degree p is 354/13=27.23. The frequency and time of density checks both reduce to *1/p* of DCCLA. However the calculation of the duplication degree increases the time complex as an additional summand $n$, since each RSSI point needs to be grouped. In most cases, the run time of $DCCLA_2$ is much smaller than that of DCCLA.

The growth-rate of $DCCLA_2$ does not only depend on the size of the input dataset, but also on the number of the RSSI groups. As we studied, the range of RSSI values from the Cisco APs used in our experiments is [-100dB, -45dB]. Though a user stays in a place longer, the increase of dataset size does not lead to a linear increase of time complexity for $DCCLA_2$, since the duplication degree $p$ may increase simultaneously.

### E. Enhanced Clustering Algorithm with Two Improvement

Two possible improvements – skipping unnecessary density checks and grouping similar points – can be combined to further improve the algorithm efficiency. The algorithm with two improvements is denoted by $DCCLA_3$. The time complexity of $DCCLA_3$ is:

$$\boldsymbol{O}\left(\boldsymbol{q} * \frac{\boldsymbol{m}}{\boldsymbol{p}} + \boldsymbol{n}\right) (\boldsymbol{q}, \boldsymbol{m}, \boldsymbol{n} \in \mathbb{N}^*, \boldsymbol{o}, \boldsymbol{m} \leq \boldsymbol{n}, \boldsymbol{p} \geq \boldsymbol{1})$$

$q$ is the number of checked RSSI groups. Compared to the improvement of $DCCLA_2$, the number of checked RSSI group ($q$) is smaller than or equal to the number of complete RSSI groups ($n/p$), which are checked in $DCCLA_2$. For the worst case, the time complexity of $DCCLA_3$ is equal to that of $DCCLA_2$ when every RSSI group is checked.

Both two possible improvements aim to improve the algorithm efficiency by reducing the frequency or time of density checks, but maintaining the clustering results. In other words, the improvements do not affect the localization accuracy.

## IV. Experimental Evaluation

In order to verify the algorithm efficiency improvement with respect to the run time, comparison experiments have been carried out.

We selected two domains, a PC with windows 7 platform and a smart phone Galaxy S3 with Android (version 4.1.2) platform for the intended investigations. We run the algorithms by using real-world datasets, which are the same as used in the previous experimental evaluation [3] for the original algorithm DCCLA. The RSSI points from available APs in the surrounding are measured at an interval of 5 seconds.

The algorithms are coded in Java. Java version "1.7.0_07" with the allocation of Heap Size (Xmx: maximum java heap size = 512M) for JVM (Java Virtual Machine) is used on the PC. For tests on the smart phone Galaxy S3, 456MB free RAM (random access memory) can be used.

### A. Influence of the Dataset Size

In this evaluation, we explore the run time of the original and enhanced algorithms by using datasets with increasing size. Based on the results in [3], *NR*=3 is used as one parameter. Another parameter *MinPts* is dynamically adjusted according to the collection time – one-third of the total scans. The data collection takes place at the office area of our department, which is located on the second floor of a three-story office building. The RSSI points from available APs in the surrounding are collected at a collection interval of 5 seconds. The data collection times in the experimental evaluation are decided based on our empirical study when we test our algorithm in practice. For test on a PC, the collection

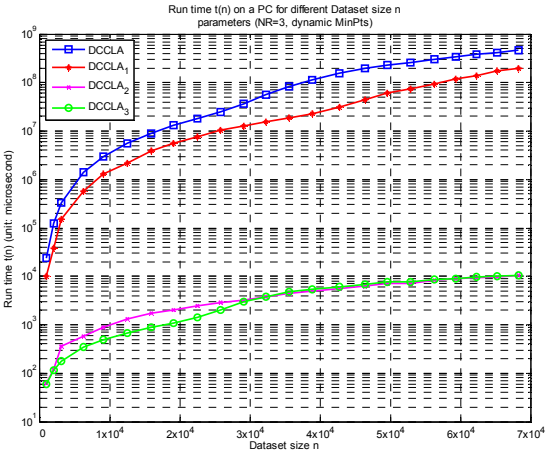


Figure 4 the run time on a PC for different dataset size

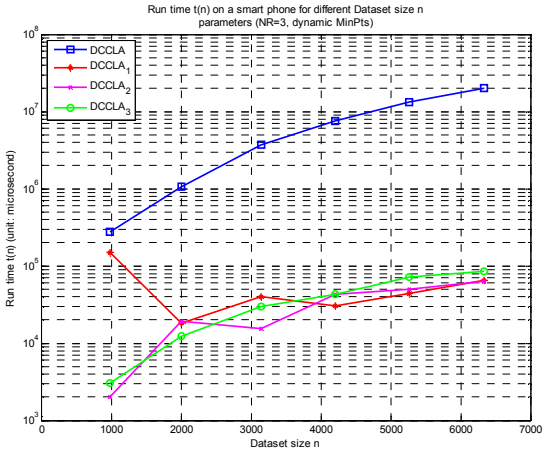


Figure 5 the run time on a smart phone for different dataset size

times of datasets increase from 10 minutes to 10 hours (under the assumption that a user does not stay in a place longer than 10 hours). The dataset size increases from around $0.1*10^4$ to $7*10^4$. Figure 4 presents the run time (*t(n)*: y-axis) on a PC for different dataset size (*n*: x-axis). The unit of the run time is microsecond ($10^{-6}$ second). Different colors represent different trial results respectively using DCCLA, $DCCLA_1$, $DCCLA_2$, and $DCCLA_3$. The tests are repeated on a smart phone by increasing the collection time from 10 minutes to 1 hour, which are the optimal collection times to obtain high localization accuracy by smart phones in our empirical study. The size of the datasets increases from 1000 to 7000. The results are shown in Figure 5.

The increase of dataset size results in an increasing run time for all four trials using DCCLA, $DCCLA_1$, $DCCLA_2$, and $DCCLA_3$. The results on a smart phone do not stably increase, due to the relative limited processing capability with respect to a smart phone processor and potential interference caused by other running applications or the smart phone system.

For the same datasets, the run times of trials using $DCCLA_2$ and $DCCLA_3$ on a PC have a significant decrease. The trial using $DCCLA_1$ also shows slightly improvement on the time complexity. On a smart phone, the run times of trials

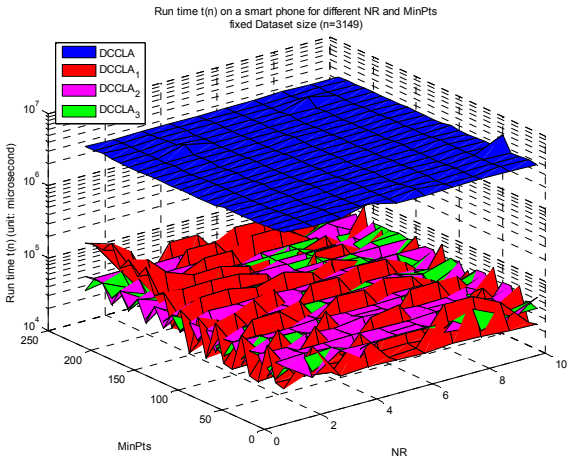


Figure 6 the run time on a smart phone for different parameters

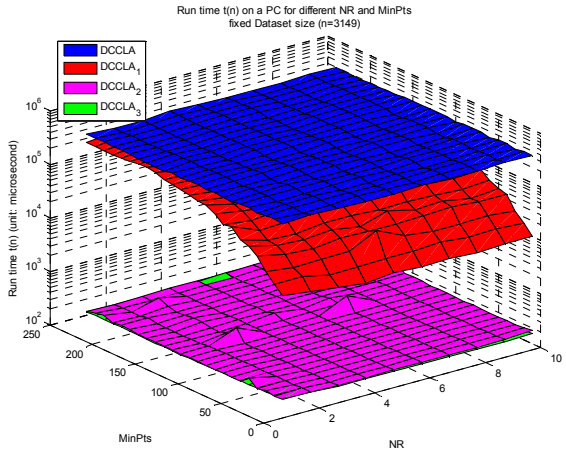


Figure 7 the run time on a PC for different parameters

using $DCCLA_1$, $DCCLA_2$ and $DCCLA_3$, even not stable, also show improved performances.

In the trials using $DCCLA_2$ and $DCCLA_3$, the run times, due to the grouping pre-process, typically grow more slowly than that of trials using DCCLA and $DCCLA_1$, especially when the size of the input dataset becomes large. It indicates that the growth-rate of $DCCLA_2$ and $DCCLA_3$ does not depend on the size of the input dataset, but on the number of the RSSI groups. Consequently, the grouping pre-process is a necessary step for applying density-based clustering algorithm to learn places from Wi-Fi data.

Comparing the performance on the PC between the trials using $DCCLA_2$ and $DCCLA_3$, the run time of $DCCLA_3$ is slightly less than that of $DCCLA_2$ when size of the input dataset is smaller than $3*10^4$. However, the decrease does not continue when the size of the input dataset becomes large. That is caused by the "if-else" statement when the algorithms determine the new check group. In $DCCLA_2$ and DCCLA, they always perform density check on the next group or point. In $DCCLA_3$ and $DCCLA_1$, the set of the next check group or point depends on if the critical group or point is figured out, being judged by an "if-else" statement. The "if-else" statement

in a recursive cluster learning loop does not affect the time complexity, but slightly affects the actual run time.

### B. Influence of the Parameters

In the experiments above, the tests are performed with a combination of parameters (*NR*=3, Dynamic *MinPts*) for the cluster learning process. We investigated the influence of the parameters when a fixed dataset size (*n*=3149) is used. The parameters to be investigated are: neighborhood range (*NR*) and the minimum number of RSSI points (*MinPts*). In this evaluation, the values of *NR* and *MinPts* are changed in each repetition of the cluster learning process. Selected *NR* values are [1, 2, 3, 4, 5, 6, 7, 8, 9, 10 *dB*]. Selected *MinPts* values range from 10 to 240 with an increment of 10.

By changing the parameters of the cluster learning algorithm, the run times, shown in Figure 7 and Figure 6, are not affected, except for some unreasonable combination of two parameters (e.g., extremely small *MinPts* and large *NR*). The unreasonable combination of two parameters would not be considered in the real-world scenarios. As we studied, the density-based clustering results are usually sensitive to the choice of two parameters. However, the evaluation turns out that the run time is not sensitive to the alternation of two parameters when the density-based clustering is applied in the place learning system.

### C. Discussion

Observed from above evaluations, $DCCLA_1$, $DCCLA_2$ and $DCCLA_3$ are able to provide a decrease of the run time on both a PC and a smart phone. $DCCLA_2$ and $DCCLA_3$ are seen to show a significant improvement on algorithm efficiency as maintaining the clustering results. On a PC, the run time of trials using $DCCLA_2$ and $DCCLA_3$ is at least 2 to 4 orders of magnitude faster than that of the original DCCLA when the collection time of an input dataset is between 10 minutes to 10 hours. The decrease of run time becomes more obvious as the size of an input dataset becomes larger. On a smart phone, the run time of trials using $DCCLA_2$ and $DCCLA_3$ is about 2 orders of magnitude faster than that of the original algorithm DCCLA when the collection time of an input dataset is between 10 minutes to 1 hour. For example, when the collection time is half an hour (n=3149), the run time reduces from 0.1 second to $10^{-4}$ second by using $DCCLA_3$ instead of DCCLA. Similarly, on a smart phone, the run time reduces from around 10 seconds to $10^{-2}$ second. When the collection time is 10 hours, $DCCLA_3$ can excute the clustering process on a PC in about 10 seconds.

As we introduced in the theoretical analysis, the algorithm with two possible improvements presents the same localization accuracy as the original one. The accuracy in this experimental scenario was investigated in the previous works [6] [3]. When the input datasets were collected in half an hour in every room, and the collection interval was 5 seconds, adjacent rooms can be correctly learned and localized with accuracy between 97% and 100% when at least 3 APs are available.

Based on the theoretical analysis of the time complexity and the experimental evaluation of the run time, we identify the algorithm $DCCLA_3$ with two improvements as the enhanced density-based clustering algorithm for the autonomous indoor localization.

## V. Conclusion

In this paper, we have presented an enhanced density-based clustering algorithm for the indoor localization algorithm DCCLA (Density-based Clustering Combined Localization Algorithm). In the enhanced algorithm, the cluster learning process is optimized by doing a pre-process of grouping and skipping unnecessary density checks. On a PC, the run time of the enhanced algorithm is 2 to 4 orders of magnitude faster than that of the original algorithm in our experimental evaluation. On a smart phone, the run time is about 2 orders of magnitude faster than that of the original algorithm. As a result, the enhanced density-based clustering algorithm is an optimal clustering algorithm as it presents significantly improved efficiency performance on both PC and smart phone platforms.

## Acknowledgment

This paper is funded by the German “Bundesministerium für Wirtschaft und Technologie” by funding the project “pinta – Pervasive Energie durch internetbasierte Telekommunikationsdienste”, reference number 01ME11027. The authors are responsible for the content of this publication and would like to acknowledge the contributions of their colleagues.

## References

[1] H. Wang, S. Sen, A. Elgohary, M. Farid and M. C. R. R. Youssef, "No need to war-drive: unsupervised indoor localization," in MobiSys '12, 2012.

[2] Y. Jiang, X. Pan, K. Li, Q. Lv, R. P. Dick, M. Hannigan and L. Shang, "ARIEL: Automatic Wi-Fi based Room Fingerprinting for Indoor Localization," in UbiComp ’12, 2012.

[3] Y. Xu, S. L. Lau, R. Kusber and K. David, "DCCLA: Autonomous Indoor Localization Using Unsupervised Wi-Fi Fingerprinting," in CONTEXT'13, Annecy, France, 2013.

[4] S. L. Lau, Y. Xu and K. David, "Novel indoor localisation using an unsupervised Wi-Fi signal clustering method," in Future Network & Mobile Summit, 2011.

[5] Y. Wu, J. Guo and X. Zhang, "A linear DBSCAN algorithm based on LSH," in Machine Learning and Cybernetics, 2007 International Conference on, 2007.

[6] Y. Xu, S. L. Lau, R. Kusber and K. David, "An Experimental Investigation of Indoor Localization by Unsupervised Wi-Fi Signal Clustering," in Future Network and Mobile Summit, 2012.

[7] E. Ester, H. Kriegel, J. Sander and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in 2nd International Conference on Knowledge Discovery and Data Mining, 1996.

[8] C. A. Shaffer, Data Structures and Algorithm, 3.2 (Java Version) ed., Dover Publications, 2012.