Available online at www.HighTechJournal.org

HighTech and Innovation Journal

ISSN: 2723-9535

HighTech and Innovation Journal

Vol. 6, No. 1, March, 2025

Enhancing DBSCAN Accuracy and Computational Efficiency Using Closest Access Point Pre-Clustering for Fingerprint-Based Localization

Abdulmalik Shehu Yaro [1, 2*], Filip Maly [1], Pavel Prazak [1]

[1] *Department of Informatics and Quantitative Methods, Faculty of Informatics and Management, University of Hradec Kralove, 500 03 Hradec Kralove, Czech Republic.*

[2] *Department of Electronics and Telecommunications Engineering, Ahmadu Bello University, Zaria, 810106, Nigeria.*

Received 11 January 2025; Revised 21 February 2025; Accepted 26 February 2025; Published 01 March 2025

Abstract

Within the context of fingerprint database clustering, the density-based spatial clustering of applications with noise (DBSCAN) is notable for its robustness to outliers and ability to handle clusters of different sizes and shapes. However, its high computational burden limits its scalability for dense fingerprint databases. A hybrid two-stage clustering method, the CAP-DBSCAN algorithm, is proposed in this paper, designed to accelerate DBSCAN clustering while ensuring accuracy for fingerprint-based localisation systems. The CAP-DBSCAN algorithm employs the closest access point (CAP) algorithm to pre-cluster the database, while the DBSCAN algorithm performs clustering refinement. It dynamically adjusts the neighborhood radius (Eps) value for each pre-cluster using the k-distance plot method. The performance of the CAP-DBSCAN algorithm is determined across four publicly available received signal strength (RSS)-based fingerprint databases with Euclidean and Manhattan distances as fingerprint similarity metrics. This is benchmarked against the performances of the standard DBSCAN (s-DBSCAN) and k-means++-DBSCAN (k-DBSCAN) algorithms presented in previous research. Simulation results show that the CAP-DBSCAN algorithm consistently outperforms both the s-DBSCAN and k-DBSCAN algorithms, achieving higher silhouette scores, which indicates the generation of more compact and well-defined clusters. Furthermore, the CAP-DBSCAN algorithm demonstrates superior computational efficiency as a result of the CAP algorithm generating well-structured pre-clusters better than those generated by the k-means++ algorithm. This significantly reduces the computational burden of the cluster refinement process. Overall, using Manhattan distance as a fingerprint similarity metric results in the best clustering performance of the CAP-DBSCAN algorithm. These findings underscore the potential of the CAP-DBSCAN algorithm for practical applications in resource-constrained fingerprint-based localization systems.

Keywords: DBSCAN; Computational Efficiency; Proximity-Based Clustering; Fingerprint-Based Localization; Pre-clustering Approach.

1. Introduction

A fingerprint-based localisation system is commonly used for indoor localisation and estimates the position of an indoor target in two phases [1]: the offline phase and the online phase, with the offline phase being the focus of this paper. The offline phase involves creating a fingerprint database [2, 3]. This process begins with the reception and estimation of position-dependent signal parameters, such as received signal strength (RSS), from multiple spatially placed wireless access points (APs) at several reference locations (RLs) [1]. Next, fingerprint vectors are generated, which consist of all RSS measurements collected at each RL. Finally, these fingerprint vectors are stored in a database, commonly referred to as the fingerprint database or radio map, and mapped to their corresponding RLs [1]. In the second

* Corresponding author: abdulmalik.yaro@uhk.cz

http://dx.doi.org/10.28991/HIJ-2025-06-01-022

phase, the online phase, the system estimates or predicts the location of an unknown indoor target using the fingerprint vector generated at the target's location [4]. This is achieved by searching the fingerprint database with a localization matching algorithm, such as the k-nearest neighbors (k-NN) algorithm [5], to identify the RL whose fingerprint vector has the highest similarity to the fingerprint vector obtained at the unknown target's location. The RL of this fingerprint vector is considered the estimated location of the unknown indoor target.

The density of the fingerprint database created at the offline phase of the localisation process has a significant impact on the accuracy with which the location of targets is determined in the online phase [6, 7]. The higher the density, i.e., the more RLs are used in generating the fingerprint database, the greater the localisation accuracy. However, this increases the localization computation time, as the matching algorithm requires more time to search through the fingerprint database to identify the fingerprint vector with the highest similarity. To solve this trade-off, clustering algorithms are used to partition the fingerprint database [8]. The accuracy of the clustering process is critical for improving the localization accuracy of the system. Among the various clustering algorithms, the density-based spatial clustering of applications with noise (DBSCAN) algorithm is preferred due to its ability to handle noise and discover arbitrary cluster shapes [7]. However, the standard DBSCAN (s-DBSCAN) algorithm has several limitations, one of which is its high computational cost, which can hinder its performance in large-scale or irregularly structured fingerprint databases [9, 10].

Several researchers have proposed different methods to reduce the computational burden of the DBSCAN algorithm while at the same time ensuring an accurate clustering process [7, 9–14]. These methods can be categorised into two, namely, algorithm modification and hybrid pre-clustering methods. The algorithm modification methods focus on altering the DBSCAN algorithm to reduce its computational burden. In contrast, the hybrid pre-clustering methods focus on using other clustering algorithms to perform pre-clustering and applying the DBSCAN algorithm to refine the pre-clusters generated [15]. The pre-clustering using the auxiliary algorithms aims to generate well-structured and smaller sub-clusters for the DBSCAN algorithm to process efficiently. To accelerate the clustering process of the DBSCAN algorithm while ensuring accurate clustering, this paper proposes a new hybrid pre-clustering method referred to as the CAP-DBSCAN algorithm. The proposed algorithm uses the closest access point (CAP) algorithm to perform pre-clustering with the DBSCAN algorithm performing cluster refinement. The CAP algorithm is a proximity-based clustering algorithm that groups fingerprint vectors based on the closest wireless APs, which are identified by the wireless AP having the highest RSS value in the fingerprint vector [16, 17]. Performing pre-clustering using the CAP algorithm ensures that well-structured and compacted initial clusters are generated for the DBSCAN algorithm to refine. This reduces the computational burden on the DBSCAN algorithm and ensures robust clustering performance, even for irregularly shaped fingerprint databases.

The contributions of this paper are as follows: (a) the development of a clustering algorithm that integrates the CAP algorithm for pre-clustering with the DBSCAN algorithm for cluster refinement; and (b) the evaluation and identification of the optimal clustering configuration for the CAP-DBSCAN algorithm.

2. Review of Related Works

As previously stated, researchers have proposed various methods to reduce the computational burden of the DBSCAN algorithm, either by modifying the algorithm itself or by hybridizing it with other clustering algorithms. This section reviews and presents previous hybridization methods presented by researchers to reduce the computational burden, accelerate the clustering process, and improve the clustering accuracy of the DBSCAN algorithm.

Thang et al. [15] proposed a method to accelerate the clustering process of the DBSCAN algorithm, referred to as FastDBSCAN. The FastDBSCAN algorithm groups the fingerprint database using the k-means algorithm, and each group is subsequently refined using the DBSCAN algorithm. While the k-means algorithm is effective for clustering fingerprint databases with spherical clusters, its performance degrades when applied to databases with non-spherical clusters or irregular fingerprint vector distributions. Additionally, the clustering performance of the k-means algorithm is highly dependent on the selection of the optimal number of clusters to be generated and fingerprint vectors chosen during the centroid initialization process.

In Gholizadeh et al. [9], the authors presented another method referred to as the k-DBSCAN algorithm, which uses the k-means++ algorithm instead of the k-means algorithm to accelerate the clustering process of the DBSCAN algorithm. In this method, the k-means algorithm is replaced by the k-means++ algorithm, which addresses the poor initialisation of centroids in the standard k-means algorithm, resulting in more reliable and efficient clustering. Nonetheless, both k-means and k-means++ exhibit performance degradation when handling non-spherical clusters or irregular fingerprint vector distributions.

Perafan-Lopez et al. [13] proposed a method called FA+GA-DBSCAN that integrates dimensionality reduction with the DBSCAN algorithm to accelerate the clustering process. The factor analysis (FA) method reduces the dimensionality of the fingerprint database, and then the DBSCAN algorithm is applied to the reduced fingerprint database. This reduces

the overall computational burden of the DBSCAN algorithm. However, the use of the FA method may lead to the loss of valuable information that could be important for the subsequent phase of the fingerprinting-based localization process, specifically the online phase. This loss of information could potentially degrade localization performance.

Another method, namely BIRCHSCAN, was presented by Ventorim et al. [11], which uses the balanced iterative reducing and clustering using hierarchies (BIRCH) algorithm to pre-cluster the fingerprint database before refining using the DBSCAN algorithm. The BIRCH algorithm performs best when fingerprint databases have clusters that are approximately spherical and evenly distributed. However, in scenarios where a fingerprint database contains clusters with varying shapes or densities, the BIRCH algorithm may fail to accurately capture all relevant subclusters, leading to a biased sub-clustering. This will negatively impact the computational time of the DBSCAN algorithm, as it may require more time to process the subclusters.

Kumar & Reddy [10] presented a method called the G-DBSCAN algorithm to accelerate the clustering process of the s-DBSCAN algorithm. The method involves applying grouping partition methods to identify subclusters using nearest neighbors with similar patterns in a specific fingerprint database. The G-DBSCAN algorithm has several limitations, including its reliance on the order in which the patterns are processed, which can affect the clustering results. Additionally, determining threshold distances between clusters requires scanning the entire fingerprint database.

Sridevi & Rajanna [18] proposed the MBK-DBSCAN algorithm to reduce the computational burden of DBSCAN. It first applies the mini-batch K-means (MBK) algorithm for fast, scalable clustering, partitioning large datasets into fixed clusters using mini-batches. DBSCAN then refines these clusters by detecting outliers and improving boundary definitions. While MBK-DBSCAN improves efficiency and accuracy, it faces challenges such as the sensitivity of MBK to centroid initialization.

Cheng et al. [19] propose GB-DBSCAN, a clustering algorithm that integrates granular-ball (GB) representation with DBSCAN to improve efficiency. Instead of clustering individual fingerprint vectors, the GB-DBSCAN algorithm groups nearby fingerprint vectors into GBs using k-nearest neighbors (KNN) and clusters them, significantly reducing computational cost. The approach enhances the clustering time of the DBSCAN algorithm while maintaining accuracy. However, it relies on proper GB formation and may struggle with databases where local density variations make defining GBs challenging.

As seen from the earlier review literature, several methods have been presented to improve the performance of the DBSCAN algorithm by addressing its computational burden. The FastDBSCAN algorithm uses the k-means algorithm to initially group the fingerprint database, but its performance suffers with non-spherical and irregularly distributed clusters. The k-DBSCAN algorithm, which replaces the k-means algorithm with the k-means++ algorithm to improve fingerprint vector centroid initialization, also struggles with non-spherical clusters. The FA+GA-DBSCAN algorithm, which employs dimensionality reduction, results in information loss, which could degrade performance in the online phase. The BIRCHSCAN algorithm that performs pre-clustering using the BIRCH algorithm suffers degraded performance when applied to a fingerprint database with varying cluster shapes and density, leading to biased sub-clustering and increased DBSCAN algorithm clustering time. The G-DBSCAN algorithm, which uses pattern-based grouping to create subclusters, suffers from dependence on fingerprint vector pattern processing order and challenges in determining threshold distances between subclusters. Similarly, the MBK-DBSCAN algorithm, which applies mini-batch K-means (MBK) before DBSCAN to enhance efficiency, is sensitive to MBK centroid initialization, which can affect clustering accuracy. The GB-DBSCAN algorithm, which integrates GB representation with DBSCAN to reduce computational cost, relies on proper GB formation and may struggle with databases where local density variations make defining GBs challenging.

The CAP algorithm considered in this paper for pre-clustering in the CAP-DBSCAN algorithm overcomes some limitations of pre-clustering methods used by other researchers. It is robust in handling non-spherical clusters, unlike the BIRCH algorithm in BIRCHSCAN, the k-means algorithm in FASTDBSCAN, and the k-means++ algorithm in k-DBSCAN. The CAP algorithm is also insensitive to fingerprint vector pattern input order and threshold settings, unlike the grouping partition method in G-DBSCAN. It does not cause fingerprint feature information loss due to dimensionality reduction, as seen in FA+GA-DBSCAN, and does not rely on proper GB formation, unlike GB-DBSCAN.

3. Proposed CAP-DBSCAN Algorithm Clustering Methodology

This section presents the clustering methodology for the proposed CAP-DBSCAN algorithm. As mentioned earlier, the CAP-DBSCAN algorithm integrates the CAP algorithm for pre-clustering and the DBSCAN algorithm for cluster refinement. This approach aims to improve clustering accuracy and reduce the computational burden of the DBSCAN algorithm. The CAP-DBSCAN clustering process consists of two stages, each detailed in the subsections below, with a graphical illustration provided in Figure 1.

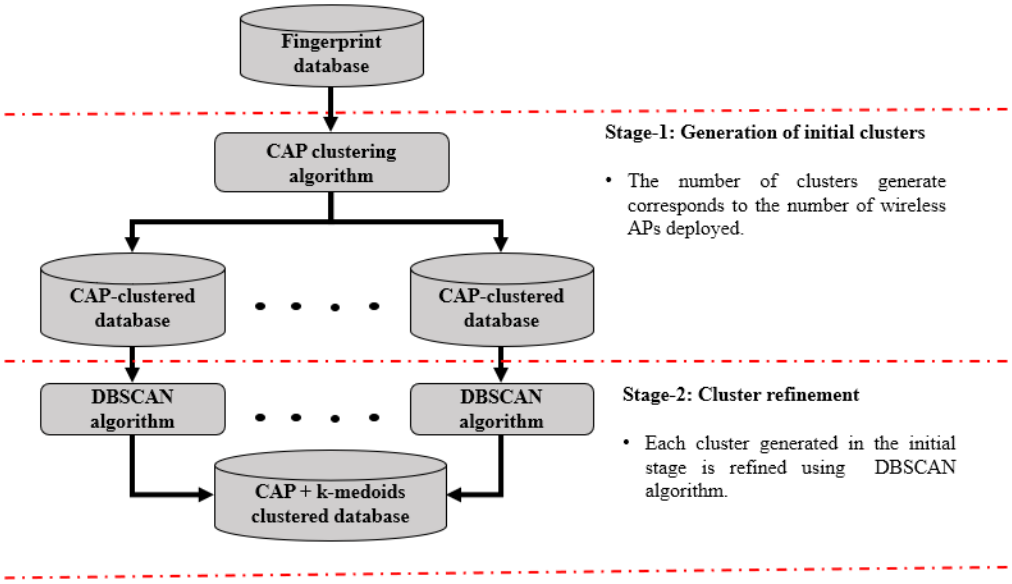

Figure 1. Graphical representation of the CAP-DBSCAN clustering methodology

3.1. First-Stage Clustering Process: Generation of Initial Clusters

As earlier mentioned, the initial clusters for the proposed CAP-DBSCAN algorithm are generated using the CAP algorithm. The CAP algorithm partitions the fingerprint database based on the proximity to the wireless AP, which is identified by the wireless AP with the highest RSS value.

Given a fingerprint database represented as F, containing *N* fingerprint vectors, where each fingerprint vector consists of a set of RSS measurements from *M* wireless APs, the database can be expressed as shown in Equation 1:

$$\mathrm{F} = \{\mathrm{f}_1, \mathrm{f}_2, \mathrm{f}_3, \dots, \mathrm{f}_N\} \quad (1)$$

where: f_i represents the i-th fingerprint vector in the database, for $i = 1,2, \dots, N$. Each fingerprint vector f_i contains RSS measurements from *M* wireless APs, given by: $\mathrm{f}_i = [rss_{i,1}, rss_{i,2}, \dots, rss_{i,M}]$, where $rss_{i,j}$ denotes the RSS value from the j-th AP for the fingerprint vector f_i, with $j = 1,2, \dots, M$

The initial clusters are generated using the steps below:

Step 1: AP selection:

For each fingerprint vector, f_i, determine the index of the wireless AP with the highest RSS value using Equation 2.

$$k_i = \arg\_\max\{\mathrm{f}_i\} \quad (2)$$

where k_i denotes the index of the wireless AP with the maximum RSS value for the fingerprint vector, f_i.

Step 2: Cluster assignment:

Assign f_i to the cluster corresponding to k_i. Thus, the fingerprint database is divided into M clusters, where the *m*-th cluster is defined as:

$$C_m = \{\mathrm{f}_i \in \mathrm{F} | k_i = m\} \quad for\ 1 \le m \le M \quad (3)$$

where C_m represents the set of fingerprint vectors assigned to the m-th cluster and each of the m-th cluster consists of fingerprint vectors in which the AP with the highest RSS value is AP *m*.

The cluster assignment process in Equation 3 ensures that each fingerprint vector is associated with the wireless AP that produces the strongest signal. After partitioning the fingerprints into M clusters by the CAP algorithm, each of the *M* clusters is independently further refined using the DBSCAN.

3.2. Second-Stage Clustering Process: Cluster Refinement

In the second stage of the CAP-DBSCAN algorithm clustering process, each cluster C_j generated by the CAP algorithm in the first stage is further refined using the DBSCAN algorithm. The DBSCAN algorithm uses neighbourhood radius (Esp) and the minimum number of fingerprints required to form a cluster (MinPts) as its clustering input parameters. Consider the *m*-th cluster denoted as C_m such as $C_m \subseteq$ F. Below is a summary of the steps involved in further refining the m-th cluster using DBSCAN algorithm [7].

Step 1: Parameter initialization:

- Define the two parameters, namely, Esp and MinPts. In this paper, MinPts is set to 1 to avoid classifying fingerprint vectors as outliers, allowing clusters to consist of a single fingerprint vector. Also, the optimum value of Esp for each of the *M* clusters is determined using the k-distance plot method [12].

Step 2: Neighbours and core fingerprint vectors identification:

- Determine the similarity value of all possible fingerprint vector pairs in C_m.
- For each fingerprint vector $f_i^{C_m} \in C_m$, identify the fingerprint vectors with a similarity value less than or equal to Esp.
- If the number of fingerprint vectors (including $f_i^{C_m}$) is greater or equal to MinPts, mark $f_i^{C_m}$ as a core fingerprint vector.

Step 3: Cluster expansion:

- For each core fingerprint vector, $f_i^{C_m}$, retrieve all its ε-neighbourhoods, including $f_i^{C_m}$.
- If the neighbouring fingerprint vector, $f_j^{C_m}$ is also a core fingerprint vector, recursively fine-tune and add its ε-neighbourhood fingerprint vectors to the cluster with $f_i^{C_m}$ as the core fingerprint vector.
- If $f_j^{C_m}$ is not yet assigned to any cluster, assign it to the current cluster.

Step 4: Handling border fingerprint vectors:

- If a fingerprint vector is within the ϵ-neighbourhood of a core fingerprint vector but does not meet the core criteria (i.e., it has fewer than MinPts neighbours), mark it as a border fingerprint vector.
- When applicable, assign border fingerprint vectors to the cluster of the nearest core fingerprint vector.

Step 5: Cluster formation:

- As the algorithm progresses, a cluster is formed by each core fingerprint vector and its connected fingerprint vectors.
- The process continues until all fingerprint vectors have been assigned to clusters.

Steps 1 to 5 for the DBSCAN algorithm are used to refine each of the initial M clusters generated by the CAP algorithm. Let C_m^{DBSCAN} be the refined clusters produced by the DBSCAN algorithm within cluster C_m; then fingerprint vectors in C_m^{DBSCAN} are:

$$C_m^{DBSCAN} = \{\text{core fingerprint} \cup \text{border fingerpint}\} \text{ for } 1 \leq m \leq M \tag{4}$$

where a core fingerprint is a fingerprint vector with at least MinPts neighbouring fingerprint vectors within a radius Eps, and a border fingerprint is a fingerprint vector within the ϵ-neighbourhood of a core fingerprint but with fewer than MinPts neighbours.

The final clusters produced by CAP-DBSCAN are the refined DBSCAN clusters within each CAP-generated partition. If there are *M* partitions, the total number of clusters is given by:

$$C^{\text{final}} = \bigcup_{m=1}^{M} C_m^{DBSCAN} \tag{5}$$

Thus, the total number of final clusters is the sum of the clusters identified by DBSCAN within each partition.

3.3. CAP-DBSCAN Algorithm Computation Complexity

In this subsection, the computational complexity (CC) of the proposed CAP-DBSCAN algorithm is determined. For a total of N fingerprint vectors in a database with M wireless APs, the CC for the CAP algorithm in the first stage of the clustering process is obtained as:

$$CC_{CAP} = O(MN) \tag{6}$$

where *N* is the total number of fingerprint vectors and *M* is the total number of wireless APs.

The CAP algorithm in the first stage generates a total of M clusters, where the size of each cluster is approximately N_{C_m}, which means:

$$N = \sum_{m=1}^{M} N_{C_m} \tag{7}$$

where N_{C_m} represents the number of fingerprint vectors in the m-th cluster.

Let C_m be the m-th cluster generated by the CAP algorithm. The CC for refining C_m by the DBSCAN algorithm is obtained as:

$$CC_{DBSCAN}^{C_m} = O\left(N_{C_m}^2\right) \text{ for } 1 \leq m \leq M \tag{8}$$

For a total of M clusters, the total CC by the DBSCAN algorithm is:

$$CC_{DBSCAN} = \sum_{m=1}^{M} CC_{DBSCAN}^{C_m} = O\left(\sum_{m=1}^{M} N_{C_m}^2\right) \tag{9}$$

The overall CC for the CAP-DBSCAN algorithm is obtained as:

$$CC_{CAP-DBSCAN} = CC_{CAP} + CC_{DBSCAN} = O(MN) + O\left(\sum_{m=1}^{M} N_{C_m}^2\right) \tag{10}$$

4. Simulation Results, Comparison and Discussion

The clustering performance of the CAP-DBSCAN algorithm is determined and presented in this section of the paper. First, the simulation parameters and setup are presented, followed by the clustering and CC performance comparison.

4.1. Simulation Parameters and Setup

Four experimentally generated, publicly available RSS-based fingerprint databases are used to evaluate the CAP-DBSCAN algorithm: SEUG_IndoorLoc [20], PIEP_UM_IndoorLoc [21], IIRC_IndoorLoc [22], and MSI_IndoorLoc [23]. These databases differ in wireless technology, number of wireless APs, coverage area, and reference locations (RLs). SEUG_IndoorLoc, PIEP_UM_IndoorLoc, IIRC_IndoorLoc, and MSI_IndoorLoc contain 3, 8, 3, and 11 wireless APs, respectively. In terms of RLs, SEUG_IndoorLoc has 49, PIEP_UM_IndoorLoc has 1000, IIRC_IndoorLoc has 68, and MSI_IndoorLoc has 631. These variations provide diverse environmental scenarios for evaluating the CAP-DBSCAN algorithm. Table 1 summarizes the characteristics of each fingerprint database.

Table 1. Fingerprint database characteristics

Fingerprint Database	Database characteristic			
	Wireless technology	Number of APs (M)	Number of RL (N)	coverage area (m^2)
SEUG_IndoorLoc	Wi-Fi	3	49	33
IIRC_IndoorLoc	Zigbee	3	68	161
PIEP_UM_IndoorLoc	Wi-Fi	8	1000	1000
MSI_IndoorLoc	Wi-Fi	11	631	1000

The choice of fingerprint similarity metric greatly affects the clustering performance of the DBSCAN algorithm. Distance-based similarity metrics, particularly Euclidean and Manhattan distances, are commonly used and are both considered in this study. For clustering performance evaluation, the silhouette score is used. This metric measures how similar a fingerprint vector is to its assigned cluster compared to other clusters. The silhouette score ranges from 1 (indicating well-defined, compact clusters) to -1 (representing poorly defined, overlapping clusters). Table 2 provides a summary of silhouette score ranges and their interpretations in the context of fingerprint database clustering. In this study, a silhouette score threshold of 0.25 is set as the minimum benchmark for acceptable clustering performance. A score above this threshold indicates that the clustering algorithm has successfully produced well-defined and distinct clusters.

Table 2. Silhouette score ranges and their interpretation

Silhouette Score Range	Interpretation
$0.7 \leq S \leq 1$	Very good clustering performance
$0.25 \leq S < 0.7$	Moderate clustering performance
$0.25 < S \leq -1$	Poor clustering performance

The proposed CAP-DBSCAN algorithm is evaluated against the s-DBSCAN and k-DBSCAN algorithms presented by Gholizadeh et al. [9]. To prevent any fingerprint vector from being treated as an outlier, the MinPts parameter is set to 1. For a fair comparison, the clustering performance of all three algorithms is assessed at their respective optimal configurations, determined by the Eps parameter. The optimal Eps values for CAP-DBSCAN, k-DBSCAN, and s-DBSCAN are identified using the k-distance plot method. Additionally, the optimal number of clusters for the k-means++ algorithm in the pre-clustering stage of k-DBSCAN is determined using the elbow method [24].

4.2. Clustering Performance Comparison With S-DBSCAN And K-DBSCAN Algorithms

As previously stated, the proposed CAP-DBSCAN algorithm is evaluated against s-DBSCAN and k-DBSCAN using the silhouette score as the clustering performance metric across the four fingerprint databases summarized in Table 1. Table 3 presents a comparison of the silhouette scores for CAP-DBSCAN, s-DBSCAN, and k-DBSCAN across these databases.

Table 3. Silhouette scores comparison across varying fingerprint databases and similarity metric

Database	Similarity metric	Silhouette scores		
		s-DBSCAN	k-DBSCAN	CAP-DBSCAN
SEUG_IndoorLoc	Euclidean	0.44	0.26	0.57
	Manhattan	0.44	0.26	0.67
IIRC_IndoorLoc	Euclidean	0.02	0.33	0.55
	Manhattan	-0.06	0.28	0.31
PIEP_UM_IndoorLoc	Euclidean	0.20	0.26	0.52
	Manhattan	0.32	0.44	0.68
MSI_IndoorLoc	Euclidean	0.41	0.20	0.64
	Manhattan	0.61	0.56	0.68

As shown in Table 3, the CAP-DBSCAN algorithm exhibits superior clustering performance across all four fingerprint databases and similarity metrics. It consistently achieves higher silhouette scores than both k-DBSCAN and s-DBSCAN, indicating the formation of more well-defined and compact clusters. In the SEUG_IndoorLoc database, all three algorithms produced clusters with silhouette scores above the 0.25 threshold, reflecting good clustering performance. However, the k-DBSCAN algorithm recorded the lowest silhouette scores among the three. In contrast, the CAP-DBSCAN algorithm achieved the highest silhouette scores of 0.57 and 0.67 using Euclidean and Manhattan distances as fingerprint similarity metrics, respectively. These scores represent improvements of approximately 23% and 34% over s-DBSCAN and 54% and 61% over k-DBSCAN, respectively. Overall, for the SEUG_IndoorLoc database, CAP-DBSCAN delivers the best clustering performance, particularly when using Manhattan distance as the fingerprint similarity metric.

For the IIRC_IndoorLoc database, both the k-DBSCAN and CAP-DBSCAN algorithms generated clusters with silhouette scores above the 0.25 threshold using Euclidean and Manhattan distances as fingerprint similarity metrics. In contrast, the s-DBSCAN algorithm produced clusters with scores below this threshold, indicating poor clustering performance. Among the three algorithms, CAP-DBSCAN achieved the highest silhouette scores of 0.55 and 0.31 with Euclidean and Manhattan distances, respectively. This represents a significant improvement of approximately 96% and 120% over s-DBSCAN and 40% and 10% over k-DBSCAN. Overall, the CAP-DBSCAN algorithm demonstrated optimal performance on the IIRC_IndoorLoc database when using Euclidean distance as the similarity metric.

Extending the analysis to the PIEP_UM_IndoorLoc database, both the k-DBSCAN and CAP-DBSCAN algorithms produced clusters with silhouette scores above the defined threshold using Euclidean and Manhattan distances as similarity metrics, indicating well-defined and compact clusters. In contrast, the s-DBSCAN algorithm only achieved scores above the threshold when using Manhattan distance. Among the three algorithms, the proposed CAP-DBSCAN algorithm achieved the highest silhouette scores of 0.52 and 0.68 with Euclidean and Manhattan distances, respectively. Compared to s-DBSCAN and k-DBSCAN, the CAP-DBSCAN algorithm generated clusters that were 62% and 60% more well-defined with Euclidean distance, and 53% and 35% more well-defined with Manhattan distance. Overall, the CAP-DBSCAN algorithm delivered optimal clustering performance on the PIEP_UM_IndoorLoc database when using Manhattan distance as the similarity metric.

Finally, for the MSI_IndoorLoc database, both the s-DBSCAN and CAP-DBSCAN algorithms generated clusters with silhouette scores well above the defined threshold for both similarity metrics. In contrast, the k-DBSCAN algorithm only achieved scores above the threshold when using Manhattan distance. Notably, the CAP-DBSCAN algorithm produced clusters with the highest silhouette scores of 0.64 and 0.68 using Euclidean and Manhattan distances, respectively. This demonstrates its superior ability to generate well-defined and compact clusters compared to both s-DBSCAN and k-DBSCAN. Specifically, the clusters generated by CAP-DBSCAN were 36% and 69% more well-defined and compact than those of s-DBSCAN and k-DBSCAN, respectively, when using Euclidean distance. With Manhattan distance, the improvements were 10% and 18%, respectively. Overall, the CAP-DBSCAN algorithm achieved its best clustering performance on the MSI_IndoorLoc database when using Manhattan distance as the similarity metric.

To further validate the improvement in clustering performance achieved by the CAP-DBSCAN algorithm, a paired t-test was conducted using the silhouette scores from all four fingerprint databases. The test was performed at a significance level of $\alpha = 0.05$ with a degree of freedom of 1. Since the objective was to determine whether the CAP-DBSCAN algorithm exhibits superior clustering performance compared to the s-DBSCAN and k-DBSCAN algorithms, a one-tailed t-test was employed. Table 4 summarizes the p-values obtained from the paired t-tests comparing s-DBSCAN vs. CAP-DBSCAN and k-DBSCAN vs. CAP-DBSCAN.

Table 4. Paired t-test results comparing clustering performance of s-DBSCAN, k-DBSCAN, and CAP-DBSCAN

Comparison	p-value	Significance ($\alpha = 0.05$)
s-DBSCAN vs. CAP-DBSCAN	0.0010	Significant
k-DBSCAN vs. CAP-DBSCAN	0.0012	Significant

At a significance level of $\alpha = 0.05$, the results demonstrate that the CAP-DBSCAN algorithm significantly outperforms both the s-DBSCAN and k-DBSCAN algorithms, with p-values of 0.0010 and 0.0012, respectively—both well below the significance threshold. This provides strong evidence of a statistically significant improvement in the clustering performance of the CAP-DBSCAN algorithm compared to the other two algorithms. These findings further underscore the effectiveness of combining the CAP algorithm for pre-clustering with the DBSCAN algorithm for clustering refinement.

Overall, the analysis of results across the four fingerprint databases underscores the superior clustering performance of the CAP-DBSCAN algorithm, which can be attributed to the effectiveness of the CAP algorithm used for pre-clustering. By leveraging the CAP algorithm, CAP-DBSCAN consistently achieves higher silhouette scores compared to the s-DBSCAN and k-DBSCAN algorithms, demonstrating its ability to generate more well-defined and compact clusters. The statistical significance of these improvements is further confirmed by paired t-test results, with p-values below the significance threshold of 0.05, indicating that the performance gains are robust and not due to random chance. Additionally, the CAP-DBSCAN algorithm performs optimally when using Manhattan distance as the fingerprint similarity metric, suggesting that the databases favor similarity determination based on absolute distance differences rather than Euclidean distance. This aligns with the CAP algorithm's ability to handle complex data structures effectively during the pre-clustering stage.

However, while the CAP-DBSCAN algorithm excels in clustering quality due to its innovative pre-clustering approach, it is equally important to evaluate its computational efficiency. A thorough assessment of the computational burden imposed by CAP-DBSCAN, relative to the s-DBSCAN and k-DBSCAN algorithms, is essential to determine its practicality for real-world applications. This evaluation is presented in the following subsection.

4.3. Time Computational Complexity Comparison With S-DBSCAN And K-DBSCAN Algorithms

Table 5 compares the computational complexity (CC) of the CAP-DBSCAN algorithm with the s-DBSCAN and k-DBSCAN algorithms using big O notation. As previously stated, N represents the total number of fingerprints or RLs in the database, M denotes the number of wireless APs, K is the number of clusters generated by the k-means++ algorithm, N_m is the number of fingerprints within each initial cluster obtained by the CAP algorithm, and t refers to the number of iterations performed by the k-means++ algorithm.

Table 5. CC comparison with other algorithms [9]

Clustering Algorithm	CC
DBSCAN	$O(N^2)$
k-DBSCAN	$O\left(tNK + \left(\frac{N}{K}\right)^2\left(1 + \frac{K(K-1)}{2}\right)\right)$
CAP-DBSCAN	$O\left(MN + \sum_{m=1}^{M} N_{C_m}^2\right)$

where: N is the total number of fingerprint vector in the database, K is the number of cluster generated during the k-DBSCAN pre-clustering stage by the k-means++ algorithm, t is the number of iterations required for clustering in k-DBSCAN algorithm, M is the number of initial clusters formed by the CAP algorithm in CAP-DBSCAN and N_{C_m} is the number of fingerprint vectors in the m-th cluster in CAP-DBSCAN.

Table 6 presents the numerical CC values of the CAP-DBSCAN algorithm compared to the s-DBSCAN and k-DBSCAN algorithms across all four fingerprint databases with varying characteristics, as shown in Table 3.

Table 6. Numerical CC comparison

Database	TIME CC		
	s-DBSCAN	k-DBSCAN	CAP-DBSCAN
SEUG_IndoorLoc	2,401	1,3475	1,000
IIRC_IndoorLoc	37,636	19,400	2004
PIEP_UM_IndoorLoc	1,000,000	508,000	134,440
MSI_IndoorLoc	398,161	206,021	54,234

The CC numerical values in Table 6 show that the CAP-DBSCAN algorithm has the lowest CC values, with significant time reductions compared to the k-DBSCAN and s-DBSCAN algorithms across all four fingerprint databases. For instance, in PIEP_UM_IndoorLoc, which is the largest fingerprint database, the CAP-DBSCAN algorithm has a processing time of about 134,400, compared to 508,000 and 1,000,000 for the k-DBSCAN and s-DBSCAN algorithms, respectively. This means that the CAP-DBSCAN algorithm achieved a percentage processing time reduction of about 277% and 643% in comparison to the k-DBSCAN and s-DBSCAN algorithms, respectively. Similarly, in the MSI_IndoorLoc database, which is the second largest database, the CAP-DBSCAN algorithm has a processing time of about 54,234, which is substantially less than the 206,021 for k-DBSCAN and 398,161 for s-DBSCAN. This translates to about 280% and 634%, respectively, reductions in processing time achieved by the CAP-DBCAN algorithm in comparison to the k-DBSCAN and s-DBSCAN algorithms.

Extending the analysis to the small-scale fingerprint databases, which are the SEUG_IndoorLoc and IIRC_IndoorLoc databases, the CAP-DBSCAN algorithm has the lowest processing time. In the SEUG_IndoorLoc database, the CAP-DBSCAN algorithm has a processing time of about 1,000, which is about 35% and 140% faster than that of the k-DBSCAN and s-DBSCAN algorithms, respectively. Also, in the IIRC_IndoorLoc database, the CAP-DBSCAN algorithm has a processing time of about 2,004, which is about 868% faster than the k-DBSCAN algorithm and 1,778% faster than the s-DBSCAN algorithm. The improved performance achieved by the CAP-DBSCAN algorithm can be attributed to its optimised pre-clustering processing using the CAP algorithm, which generates well-structured and compacted sub-clusters that minimise unnecessary computations. This low computational efficiency is especially valuable in large-scale, fingerprint-based localization systems where computational resources are limited. Overall, the CC analysis across the four fingerprint databases suggests that the CAP-DBCAN algorithm is a robust choice for clustering tasks in resource-constrained fingerprinting-based localization systems. This is due to its reduced computational costs with high clustering accuracy.

In summary, based on silhouette score and CC comparison analysis, the CAP-DBSCAN algorithm has been established as a robust and efficient clustering technique, effectively addressing the limitations of the s-DBSCAN and k-DBSCAN algorithms. The s-DBSCAN algorithm suffers from a high computational burden, which is mitigated in the CAP-DBSCAN and k-DBSCAN algorithms through pre-clustering processing using the CAP and k-means++ algorithms, respectively. However, the performance of the k-DBSCAN algorithm heavily depends on the effectiveness of the k-means++ algorithm, which, in turn, is influenced by factors such as the structure of the fingerprint database and the predefined number of clusters. In contrast, the CAP-DBSCAN algorithm consistently produces well-structured clusters for the DBSCAN algorithm refinement process, regardless of whether the fingerprint database is structured or unstructured. Notably, this is achieved without requiring optimal input parameter selection for the CAP algorithm. These findings highlight the potential of the CAP-DBSCAN algorithm for practical applications in resource-constrained fingerprint-based localization and other applications requiring robust clustering techniques.

5. Conclusion

This paper aims to improve the clustering performance and computational efficiency of the DBSCAN algorithm, particularly when dealing with fingerprint databases of varying densities, as a single Eps value cannot effectively accommodate all cluster types. To address this, the paper proposes a hybrid pre-clustering algorithm called the CAP-DBSCAN algorithm. The CAP-DBSCAN algorithm integrates the CAP algorithm, a wireless AP proximity-based clustering approach, to generate initial clusters. Each of these clusters is subsequently refined using the DBSCAN algorithm, with the Eps value for each cluster determined dynamically using the k-distance plot method. The performance of the CAP-DBSCAN algorithm is evaluated on four fingerprint databases with varying characteristics, using both Euclidean and Manhattan distances as similarity metrics. Simulation results demonstrate that the CAP-DBSCAN algorithm is computationally more efficient compared to the s-DBSCAN and k-DBSCAN algorithms presented in previous studies. Clustering performance, assessed using the Silhouette score as a performance metric, reveals that the CAP-DBSCAN algorithm consistently generates clusters that are better separated and more compact across all four fingerprint databases considered. Additionally, the CAP algorithm's pre-clustering process is independent of input parameters, unlike the k-means++ algorithm employed in the k-DBSCAN algorithm. Overall, the optimal clustering performance of the proposed CAP-DBSCAN algorithm is achieved when Manhattan distance is used as the fingerprint similarity metric.

Since the performance of the DBSCAN algorithm is also influenced by the choice of fingerprint similarity metric, future work will explore the potential of incorporating a pattern-based fingerprint similarity metric. This approach contrasts with the commonly used distance-based metrics and aims to further enhance the effectiveness of the CAP-DBSCAN algorithm.

6. Declarations

6.1. Author Contributions

Conceptualization, A.S.Y.; methodology, A.S.Y.; software, F.M. and P.P.; validation, A.S.Y.; formal analysis, A.S.Y.; investigation, A.S.Y.; resources, F.M. and P.P.; data curation, A.S.Y.; writing—original draft preparation, A.S.Y.; writing—review and editing, A.S.Y. and P.P.; visualization, A.S.Y.; supervision, P.P and F.M.; project administration, F.M. and P.P.; funding acquisition, F.M and P.P. All authors have read and agreed to the published version of the manuscript.

6.2. Data Availability Statement

The data presented in this study are available on request from the corresponding author.

6.3. Funding

This research is supported by the UHK FIM Excellence project run at the Faculty of Informatics and Management, University of Hradec Kralove, Czech Republic.

6.4. Acknowledgments

The authors acknowledge funding from the FIM Excellence project run at the Faculty of Informatics and Management, University of Hradec Kralove, Czech Republic.

6.5. Institutional Review Board Statement

Not applicable.

6.6. Informed Consent Statement

Not applicable.

6.7. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

7. References

[1] Yaro, A. S., Maly, F., & Prazak, P. (2023). A Survey of the Performance-Limiting Factors of a 2-Dimensional RSS Fingerprinting-Based Indoor Wireless Localization System. Sensors, 23(5), 2545. doi:10.3390/s23052545.

[2] Kolakowski, M. (2020). Automatic radio map creation in a fingerprinting-based BLE/UWB localisation system. IET Microwaves, Antennas and Propagation, 14(14), 1758–1765. doi:10.1049/iet-map.2019.0953.

[3] Yaro, A. S., Filip, M., Maly, K., & Prazak, P. (2024). Clustering Performance Analysis of the K-Medoids Algorithm for Improved Fingerprint-Based Localization. Jordan Journal of Electrical Engineering, 10(3), 431–442. doi:10.5455/jjee.204-1703256698.

[4] Shang, S., & Wang, L. (2022). Overview of WiFi fingerprinting-based indoor positioning. IET Communications, 16(7), 725–733. doi:10.1049/cmu2.12386.

[5] Ali, M., Jung, L. T., Abdel-Aty, A. H., Abubakar, M. Y., Elhoseny, M., & Ali, I. (2020). Semantic-k-NN algorithm: An enhanced version of traditional k-NN algorithm. Expert Systems with Applications, 151, 113374. doi:10.1016/j.eswa.2020.113374.

[6] Sadhukhan, P. (2019). Performance analysis of clustering-based fingerprinting localization systems. Wireless Networks, 25(5), 2497–2510. doi:10.1007/s11276-018-1682-7.

[7] Yaro, A. S., Maly, F., Maly, K., & Prazak, P. (2024). Enhancing DBSCAN Clustering for Fingerprint-Based Localization With a Context Similarity Coefficient-Based Similarity Measure Metric. IEEE Access, 12, 117298–117307. doi:10.1109/ACCESS.2024.3446674.

[8] Xu, D., & Tian, Y. (2015). A Comprehensive Survey of Clustering Algorithms. Annals of Data Science, 2(2), 165–193. doi:10.1007/s40745-015-0040-1.

[9] Gholizadeh, N., Saadatfar, H., & Hanafi, N. (2021). K-DBSCAN: An improved DBSCAN algorithm for big data. Journal of Supercomputing, 77(6), 6214–6235. doi:10.1007/s11227-020-03524-3.

[10] Kumar, K. M., & Reddy, A. R. M. (2016). A fast DBSCAN clustering algorithm by accelerating neighbor searching using Groups method. Pattern Recognition, 58, 39–48. doi:10.1016/j.patcog.2016.03.008.

[11] de Moura Ventorim, I., Luchi, D., Rodrigues, A. L., & Varejão, F. M. (2021). BIRCHSCAN: A sampling method for applying DBSCAN to large datasets. Expert Systems with Applications, 184, 115518. doi:10.1016/j.eswa.2021.115518.

[12] Yin, L., Hu, H., Li, K., Zheng, G., Qu, Y., & Chen, H. (2023). Improvement of DBSCAN Algorithm Based on K-Dist Graph for Adaptive Determining Parameters. Electronics (Switzerland), 12(15), 3213. doi:10.3390/electronics12153213.

[13] Perafan-Lopez, J. C., Ferrer-Gregory, V. L., Nieto-Londoño, C., & Sierra-Pérez, J. (2022). Performance Analysis and Architecture of a Clustering Hybrid Algorithm Called FA+GA-DBSCAN Using Artificial Datasets. Entropy, 24(7), 875. doi:10.3390/e24070875.

[14] Bi, J., Cao, H., Wang, Y., Zheng, G., Liu, K., Cheng, N., & Zhao, M. (2022). DBSCAN and TD Integrated Wi-Fi Positioning Algorithm. Remote Sensing, 14(2), 297. doi:10.3390/rs14020297.

[15] Thang, V. V., Pantiukhin, D. V., & Galushkin, A. I. (2016). A hybrid clustering algorithm: The FastDBSCAN. Proceedings - 2nd International Conference on Engineering and Telecommunication, 69–74. doi:10.1109/EnT.2015.31.

[16] Quezada-Gaibor, D., Torres-Sospedra, J., Nurmi, J., Koucheryavy, Y., & Huerta, J. (2021). Lightweight Wi-Fi Fingerprinting with a Novel RSS Clustering Algorithm. 2021 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2021, 1–8. doi:10.1109/IPIN51156.2021.9662612.

[17] Yaro, A. S., Maly, F., Prazak, P., & Malý, K. (2024). Improved Fingerprint-Based Localization Based on Sequential Hybridization of Clustering Algorithms. Emerging Science Journal, 8(2), 394-406. doi:10.28991/ESJ-2024-08-02-02.

[18] Sridevi, K. N., & Rajanna, M. (2025). Hybrid Clustering Framework for Scalable and Robust Query Analysis: Integrating Mini-Batch K-Means with DBSCAN Hybrid Model for Complex Data Clustering. International Journal of Advanced Computer Science and Applications, 16(1), 906–912. doi:10.14569/IJACSA.2025.0160187.

[19] Cheng, D., Zhang, C., Li, Y., Xia, S., Wang, G., Huang, J., Zhang, S., & Xie, J. (2024). GB-DBSCAN: A fast granular-ball based DBSCAN clustering algorithm. Information Sciences, 674, 120731. doi:10.1016/j.ins.2024.120731.

[20] Sadowski, S., Spachos, P., & Plataniotis, K. N. (2020). Memoryless Techniques and Wireless Technologies for Indoor Localization with the Internet of Things. IEEE Internet of Things Journal, 7(11), 10996–11005. doi:10.1109/JIOT.2020.2992651.

[21] Torres-Sospedra, J., Moreira, A., Mendoza-Silva, G. M., Joao Nicolau, M., Matey-Sanz, M., Silva, I., Huerta, J., & Pendao, C. (2019). Exploiting different combinations of complementary sensor's data for fingerprint-based indoor positioning in industrial environments. 2019 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2019, 1–8. doi:10.1109/IPIN.2019.8911758.

[22] Alhmiedat, T. (2023). Fingerprint-Based Localization Approach for WSN Using Machine Learning Models. Applied Sciences (Switzerland), 13(5), 3037. doi:10.3390/app13053037.

[23] Moreira, A., Silva, I., Meneses, F., Nicolau, M. J., Pendão, C., & Torres-Sospedra, J. (2017). Multiple simultaneous Wi-Fi measurements in fingerprinting indoor positioning. 2017 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2017, 2017-January, 1–8. doi:10.1109/IPIN.2017.8115914.

[24] Marutho, D., Handaka, S. H., & Wijaya, E. (2018). The determination of cluster number at k-mean using elbow method and purity evaluation on headline news. 2018 International Seminar on Application for Technology of Information and Communication, 533-538. doi:10.1109/ISEMANTIC.2018.8549752.