Cloud-based Indoor Positioning Platform for Context-adaptivity in GNSS-denied Scenarios

Doctoral Thesis

Darwin Patricio Quezada Gaibor

Supervisors: Prof. Joaquín Huerta (Universitat Jaume I)

Dr. Joaquín Torres-Sospedra (University of Minho)

Prof. Jari Nurmi (Tampere University)

Prof. Yevgeni Koucheryavy (Tampere University)

This thesis has been completed in a joint/double Doctoral Degree programme at Universitat Jaume I, Spain and Tampere University, Finland.

Castelló de la Plana (Spain)
February 2023

Cloud-based Indoor Positioning Platform
for Context-adaptivity in GNSS-denied Scenarios

Report submitted by Darwin Patricio Quezada Gaibor in order to be eligible for a joint/double doctoral degree awarded by the Universitat Jaume I and Tampere University

UNIVERSITAT JAUME·I

European Joint Doctorate Marie Sklodowska-Curie in
A Network for Dynamic Wearable Applications with Privacy Constraints (A-WEAR)

Universitat Jaume I – Doctoral School

Tampere University

Doctoral programme in Dynamic Wearable Applications with Privacy Constraints

Tampere University

DARWIN PATRICIO QUEZADA GAIBOR
Digitally signed by DARWIN PATRICIO|QUEZADA GAIBOR
Date: 2023.01.31 13:55:13 +01'00'

Darwin Patricio Quezada Gaibor

Firmado digitalmente por JOAQUIN|HUERTA|GUIJARRO
Nombre de reconocimiento (DN): cn=JOAQUIN|HUERTA|GUIJARRO, serialNumber=04571672P, givenName=JOAQUIN, sn=HUERTA GUIJARRO, ou=CIUDADANOS, o=ACCV, c=ES
Fecha: 2023.02.01 21:11:43 +01'00'

Prof. Joaquín Huerta Guijarro

Assinado por: **JOAQUIN TORRES SOSPEDRA**
Num. de Identificação: PASES-PAK129655
Data: 2023.02.02 09:40:09 +0000

Dr. Joaquín Torres-Sospedra

Jari Nurmi
Digitally signed by Jari Nurmi
DN: cn=Jari Nurmi, c=FI, o=Tampere University, ou=ITC Faculty, email=jari.nurmi@tuni.fi
Date: 2023.02.02 08:39:20 +02'00'

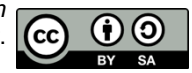
Prof. Jari Nurmi

Prof. Yevgeni Koucheryavy

Castelló de la Plana, February 2023

This dissertation is funded by the European Union's Horizon 2020 Research and Innovation programme under the Marie Skłodowska Curie grant agreement No. 813278, A-WEAR.

Cloud-based Indoor Positioning Platform for Context-adaptivity in GNSS-denied Scenarios. Copyright © 2023 Darwin Quezada-Gaibor. This work is licensed under CC BY 4.0.



DARWIN PATRICIO QUEZADA GAIBOR

Cloud-based Indoor Positioning Platform for Context-adaptivity in GNSS-denied Scenarios

ACADEMIC DISSERTATION

To be presented, with the permission of

the Doctoral School of Universitat Jaume I, and of the Faculty of Information Technology and Communication Sciences of Tampere University,

for public discussion at Universitat Jaume I,

Av. Vicent Sos Baynat, s/n 12071, Castelló de la Plana, Spain,

On March 31st 2023.

ACADEMIC DISSERTATION

Universitat Jaume I, Doctoral School
Spain

Tampere University, Faculty of Information Technology and Communication Sciences
Finland

Responsible supervisor	Professor Joaquín Huerta Universitat Jaume I Spain	
Supervisor(s)	Dr. Joaquín Torres Sospedra Universitat Jaume I Spain	Professor Jari Nurmi Tampere University Finland
	Professor Yevgeni Koucheryavy Tampere University Finland	
Pre-examiner(s)	Dr. Manon Kok TU Delf Netherlands	Dr. Alfonso Bahillo Martínez University of Valladolid Spain
	Dr. Juan Jesús García Domínguez University of Alcalá Spain	Dr. Manuel Francisco Dolz Zaragozá Universitat Jaume I Spain
Opponent(s)	Professor Luca de Nardis Sapienza University of Rome Italy	

The originality of this thesis has been checked using the Turnitin Originality Check service.

Copyright © 2023 Author

ISBN 978-952-03-2759-0 (print)
ISBN 978-952-03-2760-6 (pdf)
http://urn.fi/URN:ISBN:978-952-03-2760-6

2023

A mi amada familia.

ACKNOWLEDGEMENTS

This research work was carried out at Universitat Jaume I - Castellón de la Plana, Spain and Tampere University - Tampere, Finland, between 2019 and 2022. It has been a rewarding experience, both personally and professionally, where I have had the opportunity to know and work with excellent researchers and friends who have contributed in one way or another to this dissertation. Here I want to express my warm gratitude to them.

Firstly, I want to express my sincere thanks to my supervisors, Dr. Joaquín Torres-Sospedra (Ximo), Prof. Joaquín Huerta, Prof. Jari Nurmi and Prof. Yevgeni Koucheryavy, for their support and valuable advice during these years. Additionally, I would like to gratefully acknowledge funding from the European Union's Horizon 2020 Research and Innovation programme under the Marie Skłodowska Curie grant agreement No. 813278, A-WEAR; this work would not be possible without this funding.

I would also like to take the opportunity to thank all the staff at the A-WEAR project and Geotec for their help and assistance during my time in Spain and Finland, – they became a second family to me. I would like to thank my colleagues and co-authors of many of our articles, Dr. Antonino Crivello, Dr. Francesco Furfari, Prof. Elena Simona Lohan, and Lucie and Roman Klus.

I would like to extend my sincere gratitude and love to my dad Franco (of blessed memory), my mum Marlene, my sisters Ximena and Katty, and my dear Marina, for all your support. There are no words to express my deep gratitude to you for your valuable advice and undying patience.

Last but not least, I want to express my thanks to my friends. In particular, my dear friends Padma, Ellis, Noemi, Janeth, Carlos, and Christian, for your sincere friendship and help.

Castelló de la Plana, January 2023
Darwin Quezada Gaibor

ABSTRACT

The demand for positioning, localisation and navigation services is on the rise, largely owing to the fact that such services form an integral part of applications in areas such as human activity recognition, robotics, and eHealth. Depending on the field of application, these services must accomplish high levels of accuracy, massive device connectivity, real-time response, flexibility, and integrability. Although many current solutions have succeeded in fulfilling these requirements, numerous challenges remain in terms of providing robust and reliable indoor positioning solutions.

This dissertation has a core focus on improving computing efficiency, data preprocessing, and software architecture for Indoor Positioning Systems (IPSs), without throwing out position and location accuracy. Fingerprinting is the main positioning technique used in this dissertation, as it is one of the approaches used most frequently in indoor positioning solutions. The dissertation begins by presenting a systematic review of current cloud-based indoor positioning solutions for Global Navigation Satellite System (GNSS) denied scenarios. This first contribution identifies the current challenges and trends in indoor positioning applications over the last seven years (from January 2015 to May 2022).

Secondly, we focus on the study of data optimisation techniques such as data cleansing and data augmentation. This second contribution is devoted to reducing the number of outliers fingerprints in radio maps and, therefore, reducing the error in position estimation. The data cleansing algorithm relies on the correlation between fingerprints, taking into account the maximum Received Signal Strength (RSS) values, whereas the Generative Adversarial Network (GAN) network is used for data augmentation in order to generate synthetic fingerprints that are barely distinguishable from real ones. Consequently, the positioning error is reduced by more than 3.5% after applying the data cleansing. Similarly, the positioning error is reduced in 8 from 11 datasets after generating new synthetic fingerprints.

The third contribution suggests two algorithms which group similar fingerprints

into clusters. To that end, a new post-processing algorithm for Density-based Spatial Clustering of Applications with Noise (DBSCAN) clustering is developed to redistribute noisy fingerprints to the formed clusters, enhancing the mean positioning accuracy by more than 20% in comparison with the plain DBSCAN. A new lightweight clustering algorithm is also introduced, which joins similar fingerprints based on the maximum RSS values and Access Point (AP) identifiers. This new clustering algorithm reduces the time required to form the clusters by more than 60% compared with two traditional clustering algorithms.

The fourth contribution explores the use of Machine Learning (ML) models to enhance the accuracy of position estimation. These models are based on Deep Neural Network (DNN) and Extreme Learning Machine (ELM). The first combines Convolutional Neural Network (CNN) and Long short-term memory (LSTM) to learn the complex patterns in fingerprinting radio maps and improve position accuracy. The second model uses CNN and ELM to provide a fast and accurate solution for the classification of fingerprints into buildings and floors. Both models offer better performance in terms of floor hit rate than the baseline (more than 8% on average), and also outperform some machine learning models from the literature.

Finally, this dissertation summarises the key findings of the previous chapters in an open-source cloud platform for indoor positioning. This software developed in this dissertation follows the guidelines provided by current standards in positioning, mapping, and software architecture to provide a reliable and scalable system.

RESUMEN

La demanda de servicios de posicionamiento, localización y navegación va en aumento, en gran medida debido a que dichos servicios forman parte integral de aplicaciones en áreas como el reconocimiento de la actividad humana, la robótica y el eHealth. Dependiendo del campo de aplicación, estos servicios deben lograr altos niveles de precisión, conectividad masiva de dispositivos, respuesta en tiempo real, flexibilidad e integrabilidad. Aunque muchas de las soluciones actuales han logrado cumplir estos requisitos, siguen existiendo numerosos retos a la hora de proporcionar soluciones de posicionamiento en interiores robustas y fiables.

Esta tesis se centra en mejorar la eficiencia computacional, el preprocesamiento de datos y la arquitectura de software en sistemas de posicionamiento en interiores, sin dejar de lado la precisión en la posición y localización. Fingerprinting es la técnica principal de posicionamiento utilizada en esta disertación, ya que es uno de los enfoques utilizados con mayor frecuencia en las soluciones de posicionamiento en interiores. La tesis comienza con una revisión sistemática de las soluciones actuales de posicionamiento en interiores basadas en la nube para escenarios en dónde las señales de los sistemas satelitales de navegación globales –GNSS por sus siglas en inglés– son poco accesibles. Esta primera contribución identifica los retos y tendencias actuales en aplicaciones de posicionamiento en interiores durante los últimos siete años (desde enero de 2015 hasta mayo de 2022).

En segundo lugar, nos centramos en el estudio de técnicas de optimización de datos como la limpieza y el aumento de datos. Esta segunda contribución está dedicada a reducir el número de huellas atípicas en los mapas de radio y, por tanto, a reducir el error en la estimación de la posición. El algoritmo de limpieza de datos se basa en la correlación entre fingerprints, teniendo en cuenta los valores máximos de la intensidad de señal recibida, mientras que la Red Generativa Antagónica –GAN por sus siglas en inglés– se utiliza para el aumento de datos, teniendo como fin el generar fingerprints sintéticos apenas distinguibles de los reales. En consecuencia, el error de

posicionamiento es reducido en más de 3, 5% tras aplicar la limpieza de datos. Del mismo modo, el error de posicionamiento se reduce en 8 de 11 conjuntos de datos tras generar nuevos fingerprints.

En la tercera contribución se propone dos algoritmos que agrupan fingerprints similares en clusters. Para ello, se desarrolla un nuevo algoritmo de posprocesamiento para el algorithmo de agrupamiento espacial basado en la densidad de aplicaciones con ruido –DBSCAN por sus siglas en inglés– con el fin de redistribuir las huellas dactilares ruidosas a los clusters formados, mejorando la precisión media de posicionamiento en más de 24% en comparación con un DBSCAN simple. También se introduce un nuevo algoritmo de agrupamiento ligero, que agrupa fingerprints similares basándose en los valores máximos de la intensidad de señal recibida y el identificador del punto de acceso. Este nuevo algoritmo de clustering reduce el tiempo empleado para formar los clusters en más de 60% en comparación con dos algoritmos tradicionales de clustering.

La cuarta contribución explora el uso de modelos de aprendizaje automático –Machine Learning (ML)– para mejorar la precisión de la estimación de la posición. Estos modelos se basan en redes neuronales profundas –DNN por sus siglas en inglés– y de aprendizaje extremo –ELM por sus siglas en inglés–. El primero combina las redes convolucionales –CNN por sus siglas en inglés– y de memoria a corto plazo –LSTM por sus siglas en inglés– para aprender los patrones complejos de los mapas de radio de fingerprints y mejorar la precisión de la posición. El segundo modelo utiliza CNN y ELM para proporcionar una solución rápida y precisa para la clasificación de fingerprints en edificios y pisos. Ambos modelos ofrecen un mejor rendimiento en términos de tasa de aciertos en piso en comparación con la línea de base (más de 8% de media), y también superan a algunos modelos de aprendizaje automático de la literatura.

Por último, esta tesis resume las principales conclusiones de los capítulos anteriores en una plataforma en la nube de código abierto para el posicionamiento en interiores. El software desarrollado en esta tesis sigue las directrices proporcionadas por los estándares actuales en posicionamiento, cartografía y arquitectura de software para proporcionar un sistema fiable y escalable.

CONTENTS

List of Figures

List of Tables

ACRONYMS

AP	Access Point
APC	Affinity Propagation Clustering
API	Application Programming Interface
AR	Augmented Reality
BIM	Building Information Modeling
BLE	Bluetooth Low Energy
BSSID	Basic Service Set Identifier
CC	Cloud Computing
CDF	Cumulative Distribution Function
cGAN	Conditional Generative Adversarial Network
CNN	Convolutional Neural Network
CSI	Channel State Information
DBSCAN	Density-based Spatial Clustering of Applications with Noise
DNN	Deep Neural Network
EC	Edge Computing
ELM	Extreme Learning Machine
FC	Fog Computing
FPC	FingerPrinting Clustering
GAN	Generative Adversarial Network
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GSSL	Graph-based Semi-Supervised Learning

HTTP	HyperText Transfer Protocol
IaaS	Infrastructure as a Service
ILS	Indoor Location System
INS	Indoor Navigation System
iNaaS	Indoor Navigation as a Service
IoT	Internet of Things
IPS	Indoor Positioning System
IP	Internet Protocol
ISO	International Organization for Standardization
JWT	JSON Web Token
***k*-NN**	*k*-Nearest Neighbor
KF	Kalman filter
KPCA	Kernel Principal Component Analysis
LoST	Location-to-Service Translation Protocol
LSTM	Long short-term memory
MEC	Multi-access Edge Computing
MC	Mist Computing
MCC	Mobile Cloud Computing
MR	Mixed Reality
MSA	Microservice Architecture
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Squared Error
MVC	Model–View–Controller
MVVM	Model–View–Viewmodel
MQTT	Message Queuing Telemetry Transport
NFC	Near-field Communication
NLOS	Non-Line-Of-Sight

NN	Neural Network
OAS	OpenAPI Specification
OBEX	OBject EXchange
OGC	Open Geospatial Consortium
OS	Operating System
PaaS	Platform as a Service
PIR	Passive Infrared
POI	Point-of-Interest
PRISMA	Preferred Reporting Items for Systematic reviews and Meta-Analyses
QoE	Quality of Experience
QoS	Quality of Service
REST	REpresentational State Transfer
RF	Radio Frequency
RFID	Radio Frequency Identifier
RNN	Recurrent Neural Network
ROS	Random Over Sampling
RSS	Received Signal Strength
RSSI	Received Signal Strength Indicator
SAE	Stacked Autoencoder
SaaS	Software as a Service
SIP	Session Initiation Protocol
SLFN	Single Hidden Layer Feedforward Neural Network
SMOTE	Synthetic Minority Oversampling
SOA	Service Oriented Architecture
SLAM	Simultaneous Localization and Mapping
SSID	Service Set Identifier
SSL	Secure Sockets Layer

SVM	Support Vector Machine
TLS	Transport Layer Security
TCP	Transport Control Protocol
ToA	Time of Arrival
UDP	User Datagram Protocol
UHF	Ultra High Frequency
UI	User Interface
UWB	Ultra Wideband
VHF	Very High Frequency
VLC	Visible Light Communication
VoIP	Voice over IP
VPS	Virtual Private Server
VR	Virtual Reality
Wi-Fi	IEEE 802.11 Wireless LAN
WLAN	Wireless LAN
WPAN	Wireless Personal Area Network
XML	Extensible Markup Language
Wi-Fi	IEEE 802.11 Wireless LAN
XMPP	Extensible Messaging and Presence Protocol

NOMENCLATURE

The most common symbols used in this dissertation are listed below:

$\mathcal{A}$	Number of APs
$\mathcal{R}$	Real numbers
$\mathcal{T}_{TR}$	Training dataset
$\mathcal{T}_{\mathcal{TC}}$	Real numbers
$\aleph$	Maximum number of valid RSS values
$\mathcal{X}$	Matrix of AP's identifiers
$\mathfrak{I}$	Current match percentage between samples
J	Previous match percentage between samples
b	Bias term
C	Cluster
D	Distance matrix between the position of the synthetic and the real fingerprints
d	Distance between the position of the synthetic and the real fingerprints
H	Hidden layer output matrix (ELM)
$\mathbf{H}^{\dagger}$	Moore-Penrose Pseudoinverse
$h(\cdot)$	Activation function
NC	New clusters
o	Number of outliers detected
T	Target output (ELM)
S_l	Set of samples in the l-th cluster
$r(\cdot)$	Coefficient correlation between centroids
$\overline{x}$	Average samples in all the cluster

$\mathbf{w}$	Input weights ELM
β	Output weights of the ELM
Ξ	Regularisation term ELM
Ψ	Radio map/fingerprinting dataset
Ψ_{XTR}	Fingerprint training set
Ψ_{yTR}	Labels training set
Ψ_{XF}	Augmented training set
Ψ_{yF}	Labels of the augmented training set
Ψ_{SF}	Synthetic fingerprints
$\widehat{\Psi}$	Normalised radio map using *unit norm*
ω_{fp}	Average number of samples per reference point
ε_{3D}	Mean 3D positioning error
ε_{2D}	Mean 2D positioning error
ζ_b	Building hit rate
ζ_f	Floor hit rate
$\eth_{TE}$	Prediction time
$\eth_{TR}$	Training time
e	Mathematical constant e
γ	Non-detected values in the radio map
ρ	Threshold – minimal correlation between samples (data cleansing algorithm)
ρ_{DBSCAN}	Threshold DBSCAN
σ	Standard deviation
ι	Multiplication factor
η_{sf}	Number of synthetic fingerprints to be generate with the GAN network.
η_d	List of distances to select relevant synthetic fingerprints
η_i	Number of iterations for each distance in η_d
η_{NMRSS}	Number of maximum RSS values

ϕ	Number of clusters formed by the clustering algorithms.
Γ	Latent space used in the GAN model to generate the synthetic fingerprints
$\varkappa$	List of the shortest distances
ς	Relation between the i-*th* distance ($\varkappa_i$) and the shortest distance ($max(\varsigma)$)
ϖ	Cluster's centroid
ϖ_s	Centroid of small clusters
$\|\cdot\|$	Euclidean norm

1 INTRODUCTION

Position and navigation systems have been sought by human beings since recorded history began. Astronavigation has given way to modern indoor and outdoor positioning systems based on cutting-edge technologies such as Global Positioning System (GPS), Galileo, and all Global Navigation Satellite System (GNSS) constellations. Satellite navigation systems have become a reference point for positioning and navigation outdoors, owing to their global coverage, and widespread usage in open-source and commercial applications. GNSS is not the only technology available outdoors; there are various technologies based on magnetic fields, radiofrequency, and others, which are currently used for positioning and navigation. Like outdoor positioning systems, Indoor Positioning Systems (IPSs) have evolved over the years, gaining popularity in industry and academia. Currently, IPS or Indoor Location System (ILS) utilise the advantages of Internet of Things (IoT), computing paradigms (Cloud, Edge, Fog, Mist, etc.), 5G technology, Wireless LAN (WLAN), augmented, virtual and mixed reality [1, 2, 3, 4]. As a result, these IPSs provide more accurate, robust, reliable, and resilient services.

1.1 Background

Cloud-based indoor positioning systems provide positioning, localisation, and navigation services through the Internet [5, 6]. Generally, these services are delivered as a Software as a Service (SaaS), freeing the user from complex installations and administration. The use of the cloud and other computing paradigms to deploy indoor positioning systems offers high availability, computation and storage capabilities, and ubiquitous computation – all of which are highly sought-after in the avoidance of overloading the user device [7, 8]. In the last decade, the demand for SaaS has gained great popularity, comprising most of the cloud workload and compute instances. The workload is thus being migrated from traditional on-premises data centres to

the cloud [9]. The growth in demand for cloud services goes hand-in-hand with the incremental increase of wearable and IoT devices.

Despite computing paradigms providing high performance, indoor positioning systems must be as efficient as the computing paradigms to harness all these benefits. Truly efficient indoor positioning systems require computational efficiency, position accuracy, and standardisation. Computing efficiency can be achieved by employing robust lightweight algorithms to determine the user or device position. The complexity of indoor environments makes positioning accuracy an ongoing challenge. Some technologies, however, such as Ultra Wideband (UWB) or camera-based have been able to provide centimetre and even millimetre level accuracy [10, 11, 12]. Along with positioning technologies, the combinations of techniques, methods, and algorithms have enabled a significant improvement in positioning accuracy.

The capacity of cloud-based indoor positioning platforms to evolve and adapt has allowed the introduction of Machine Learning (ML) techniques and models to solve regression and classification problems, acquiring high levels of accuracy regardless of the scenario. For instance, Convolutional Neural Networks (CNNs) have been used to learn complex patterns in fingerprinting datasets, which led to a reduction in positioning error, as well as an improvement in the floor-building hit rate in multi-building and multi-story environments [13].

Many of the cloud-based systems currently offering the above characteristics are commercial IPSs (i.e., proprietary software), which, although often being more robust than open-source platforms, do not have the advantage of publicly available source codes. For example, indoo.rs® [14] provides a real-time indoor positioning and navigation solution which supports iBeacons and smartphone sensors to estimate position and provide navigation services. This proprietary solution also offers additional services such as indoor mapping –Simultaneous Localization and Mapping (SLAM) – and indoor analytics). In contrast, Mpeis, Roussel, Kumar, Costa, LaoudiasDenis, Capot-Ray, and Zeinalipour-Yazti [15] offers an open-source indoor positioning platform which provides localisation, navigation tracking, and analytics services. Similarly, De Nardis, Caso, and Di Benedetto [1] provide an open-source indoor positioning platform which is focused on IoT devices. One of the main advantages of anyplace platform is its support of IoT and mobile devices. Desirable characteristics of open-source indoor positioning platforms include:

- The use of standards (e.g., International Organization for Standardization (ISO), IndoorGML, etc.);
- Easy-to-adopt new indoor positioning technologies, techniques, and models;
- Provision of efficient algorithms;
- Simple to build and easy to maintain;
- Fail tolerance and robustness;
- Fulfilment of privacy and security policies;
- Independent services which can be deployed in most computing paradigms;
- Real-time position, localisation, and navigation;
- Well-documented.

As the desirability of several of the above characteristics has been reported extensively in the literature, researchers are already presenting various kinds of software (e.g., proprietary, open-source, freeware, shareware, etc.) that incorporate these characteristics.

1.2 Motivation

The integration of lightweight indoor positioning algorithms, machine learning techniques, and established standards (e.g., ISO, IndoorGML, etc.) creates reliable and robust open-source IPSs. For example, the use of standards facilitates interoperability between systems without the need to significantly alter the source code, reducing integration time and improving user experience. The use of lightweight algorithms will allow us to deploy certain processes on resource-constrained platforms and/or devices such as IoT devices. There is, however, a trade-off between computational efficiency and position accuracy. This trade-off is one of the biggest challenges faced by providers of indoor positioning, localisation and/or navigation solutions.

It is also important to consider the complexity of indoor environments and the interoperability between systems. These considerations raise the following question: "*How can we provide a robust open-source indoor positioning solution which can be used in multiple scenarios and inter-operate with other platforms with minimal consumption of computational resources?*".

Currently, many researchers have put a great deal of effort into addressing the indoor positioning challenges mentioned in previous paragraphs. The resulting research studies provide deep analysis of indoor positioning algorithms, privacy & security, data optimisation, and machine learning models [16, 17, 18]. Moreover, the open-source community has contributed to maintaining, improving, and adding functionalities to the open-source solutions. Motivated by this research and the contributions of open-source communities, this thesis provides an open-source indoor positioning platform which combines data optimisation methods, algorithms' optimisation and machine learning models.

1.3 Research Questions and Contributions

The aim of this dissertation is to develop a cloud platform for context-adaptive positioning and localisation on wearable devices. To accomplish this objective, it is necessary to analyse the components that make up an IPS/ILS in order to offer a robust solution in terms of accuracy, power consumption, and usability. Consequently, we have devised the following research questions:

- *What are the current trends and challenges of cloud-based indoor positioning platforms?* (Chapter 2)
- *Can data pre-processing techniques enhance the quality of indoor positioning data?* (Chapter 4)
- *Can the computational load in indoor positioning algorithms be reduced without significantly affecting the positioning accuracy?* (Chapters 5 and 6)
- *Can machine learning models provide the flexibility and robustness needed to function in heterogeneous GNSS-denied scenarios?* (Chapter 6)
- *Can current standards focused on indoor positioning, indoor maps, and software help to enhance the integrability and robustness of IPS?* (Chapter 7)

In order to achieve the aforementioned objective, the main contributions of this dissertation are summarised as follows:

- The dissertation begins with a systematic review of recent advances in cloud-based indoor positioning platforms. The review aims to isolate the aspects of

indoor positioning that continue to be the most challenging. The focus of this dissertation is on identifying solutions for some of these challenges.

- The WLAN fingerprinting technique is commonly implemented in many proprietary and open-source indoor positioning solutions. Generally, this technique uses a radio map, which may contain outliers samples that can affect the position estimation. These outliers are due to undesirable fluctuations in the signal strength caused by factors such as the multipath effect, Non-Line-Of-Sight (NLOS), and other factors caused by the diversity of indoor environments. Based on the correlation among the Received Signal Strength (RSS) measurements and Access Points (APs), a new data cleansing algorithm is designed to remove irrelevant fingerprints from the datasets (radio maps). This algorithm does not just remove outliers, but also improves the position accuracy and speeds up the positioning prediction. Additionally, we introduce a data augmentation model based on Generative Adversarial Networks (GANs) to generate artificial fingerprints and enrich the radio map.
- The large computational load required for many indoor positioning algorithms and/or techniques makes them unsuitable for deployment in resource-constrained devices. Efficient algorithms not only reduce the computational load, but also provide a speedy position estimation. However, there is a trade-off between computational efficiency and accuracy, therefore, makes the search for equilibrium between accuracy and power consumption a hot topic in the research field. In the light of this, we introduce two new algorithms and models to scale down the computational load and boost the position accuracy. The first model is devoted to grouping similar fingerprints based on RSS measurements stored in the radio map with the aim of diminishing the search time in the online phase of fingerprinting. The second is used to estimate the user location (building and floor).
- GNSS-denied scenarios are considered by the research community one of the most complex environments for positioning purposes, particularly when radiofrequency-based technologies are used to estimate device position. Radiofrequency signals can be highly affected by numerous factors increasing the positioning error. There are certain patterns in signal propagation, and these can be learnt using, for instance, deep learning models such as CNNs. These Deep Neural Network (DNN) models allow the extraction of meaningful

information from radio maps. On the basis of the mentioned above, we suggest two models based on DNN and Extreme Learning Machine (ELM) to learn the complex patterns in the radio maps in order to provide a robust solution for indoor positioning and localisation.

- Established expertise can be integrated into our research work with the use of standards. These standards contain a detailed set of guidelines and technical specifications used to increase the reliability, reproducibility, replicability, and effectiveness of any system or service. In this case, standards like IndoorGML, ISO/IEC 18305:2016, and ISO/IEC/IEEE 42020:2019 help enhance the quality of the indoor positioning platform developed in this dissertation, as well as its integrability with other systems.

1.4 Outline of thesis

The outline of this dissertation is detailed below:

Chapter 2 presents a systematic review of current cloud-based indoor positioning platforms. This review follows the Preferred Reporting Items for Systematic reviews and Meta-Analyses (PRISMA) model's guidelines for ensuring work undertaken is reproducible and replicable. A comprehensive analysis of the main findings of the work reviewed is detailed in this chapter, as well as the current challenges of IPSs.

Chapter 3 details the materials and methods used in this dissertation. Firstly, we introduce the positioning technique and datasets used to evaluate the proposed algorithms and ML models. Then, we introduce the positioning algorithm used as the baseline, namely *k*-Nearest Neighbor (*k*-NN), along with the positioning results obtained with 26 public datasets.

Chapter 4 introduces two new data optimisation algorithms, the first a data cleansing algorithm designed to remove outliers fingerprints from radio maps in order to improve their quality. This data cleansing algorithm is tested with 26 open-access datasets to evaluate its efficiency. The second contribution is a data augmentation model which combines the advantages of GAN architecture to create realistic fingerprints with DNN to find patterns in the radio maps.

Chapter 5 presents two approaches to reducing computational load. Firstly,

we introduce a new algorithm to rejoin fingerprints detected as outliers to the formed clusters. This post-processing algorithm uses the distance matrix to determine the correlation between the samples classified as outliers and the samples in the clusters. The second approach presents a new lightweight clustering algorithm devoted to joining similar fingerprints into clusters, leading to faster run times than traditional clustering algorithms.

Chapter 6 focuses on position estimation. Two ML models are proposed to estimate the device's position and location, offering a generalised solution which can be used in heterogeneous environments. These ML models combine the benefits of DNN and ELM to provide a robust solution with reference to time response and positioning accuracy.

Chapter 7 describes the platform architecture, components, and standards used to implement the proposed cloud-based indoor positioning platform. This platform includes the models and algorithms introduced in Chapters 4–6.

Chapter 8 concludes this research work and presents the main conclusions of this thesis.

2 INDOOR POSITIONING CLOUD PLATFORMS: A SYSTEMATIC REVIEW

This chapter consists of a review of the current state-of-the-art in cloud-based indoor positioning platforms, beginning with an overview of indoor positioning solutions, followed by an account of the research method used in the review. The chapter concludes with the major findings of the systematic review and a brief discussion of them.

The main contributions of this chapter are the follows:

- A systematic review of current cloud-based indoor positioning platforms and research works.
- Discussion of main challenges and future trends.

2.1 Introduction

GNSS has become a key technology for positioning, localisation and tracking, given its global coverage and high precision. Despite the advantages of GNSS, its performance is significantly affected in some scenarios, such as indoor environments or urban canyons [19]. In such scenarios GNSS signals remain largely unavailable, reducing the positioning accuracy.

Multiple technologies have been deployed in indoor environments to overcome the limitations of GNSS in these scenarios [20, 21], such as IEEE 802.11 Wireless LAN (Wi-Fi), Bluetooth, UWB, Visible Light Communication (VLC), and infrared. The technologies employed most frequently for indoor positioning and localisation are Wi-Fi [22, 23, 24] and Bluetooth Low Energy (BLE) [25, 26], due to their cost, availability, and the fact that many devices support them. Despite the multiple advantages provided by these technologies, a clear alternative to GNSS for indoor environments is yet to emerge.

Similarly, multiple solutions have been proposed to meet the needs of positioning services indoors. These positioning or localisation systems can be classified into 4 groups: device-free, device-based, infrastructure-free, and infrastructure-based, each one having advantages and limitations. Currently, many of them have been deployed on the cloud to avoid processing overheads in the user device and provide high-quality services. Additionally, the cloud cuts out the need to acquire hardware and software to install the IPSs/ILSs. Cloud Computing (CC) has thus become the preferred option for deploying indoor positioning and localisation systems as a service.

Generally, all components of indoor positioning technologies, including computing paradigms, are constantly evolving to offer a better and more accurate solution. This continuous and rapid development of IPS makes necessary an updated review prior to any discussion of future positioning platforms. This chapter offers an updated systematic review of current challenges and trends related to cloud-based indoor positioning platforms or systems. It also includes concepts related to indoor positioning, computing paradigms, mobile devices, network protocols, and standards.

2.2 Background

According to the National Human Activity Pattern Survey (NHAPS), people in the United States spend more than 80% of their total time indoors, and Canada shares a similar pattern [27]. Other countries may spend an equivalent amount of time indoors, and this could be the reason why IPS/ILS are increasing in demand. Indoor positioning services are currently employed in health care applications, entertainment, sports, and manufacturing [28, 29, 30]. Despite the demand for indoor positioning services being on the rise, challenges remain in relation to positioning accuracy, data optimisation, and security & privacy, among others.

As previously mentioned, indoor positioning systems can be classified into 4 primary groups: device-free, device-based, infrastructure-free and infrastructure-based. Device-free indoor localisation consists of determining the user position without the need to carry any specific device. Conversely, device-based indoor localisation actively involves the user device in the localisation process [20]. Likewise, infrastructure-based indoor localisation requires the use of indoor positioning technologies to determine the device position, while infrastructure-free does not require the configuration or deployment of any locator device deployed at a venue.

As part of IPSs, we also have indoor positioning techniques, algorithms, and methods that are as diverse as indoor positioning technologies. The algorithms or models used to estimate the user position may run on the user device or on external devices such as IoT devices and servers (local or cloud). Since many of these algorithms are very complex, they require high computational resources. A desire to design a platform that does not require excessive computational resources has led the authors to choose the cloud or other similar computing paradigms to deploy their indoor positioning solutions regardless of the type of IPS [17, 31, 11].

Nevertheless, cloud computing has limitations which should be carefully examined. Although cloud computing provides large storage and processing capabilities necessary for big data analysis, having cloud-client architecture increases the latency and reduces the time response. Moreover, cloud computing tends to be more vulnerable to security & privacy issues [32].

Cloud-based indoor positioning and localisation platforms can be extremely complex in their architecture, relying on multiple technologies, protocols, positioning algorithms, and additional components. Many of them are based on client-server architecture (see Fig. 2.1), where all information is collected by mobile devices and then sent to the cloud to estimate the device position. Generally, IPS on cloud platforms provide services such as positioning, localisation, navigation, routing and mapping. More advanced platforms also provide some additional services based on the positioning engine, including human pattern recognition, monitoring of patients, and contact tracing.

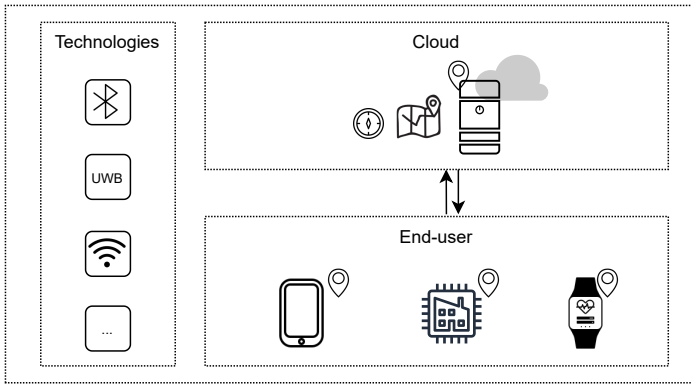


Figure 2.1 General representation of cloud-based IPS/ILS.

2.3 Research Methodology

The systematic review presented in this chapter is based on the Preferred Reporting Items for Systematic reviews and Meta-Analyses (PRISMA) guidelines [33], which consist of a 27 items checklist together with a flow diagram divided into 3 parts (previous studies; identification of new studies via databases and registers; and identification of the new studies via other methods). This methodology is characterised by an exhaustive scan of published papers using keywords which will assist in answering the research questions defined. The results retrieved are subsequently filtered using inclusion and exclusion criteria to determine which reports will ultimately be included in the review.

This section updates the key findings of the published systematic review [32] with reference to research articles published between January 2021 and May 2022.

2.3.1 Research questions

In order to identify the most relevant works, we set the following main research question (MRQ):

MRQ What are the possible gaps or issues in Cloud Platforms for positioning and navigation in GNSS-denied environments?

To keep the core of the previous work, we used the research questions (RQs) established in [32]. This allows clear analysis of the progress made in the research field between 2015 and May 2022:

RQ1 Are the main computing paradigms used in current indoor positioning platforms?

RQ2 What network protocols do the current platforms use to provide reliable services?

RQ3 Do the current platforms permit heterogeneous positioning technologies for GNSS-denied scenarios?

RQ4 Do the current platforms adapt to different scenarios?

RQ5 What are the improvements made in similar studies in the field of cloud-based indoor positioning solutions?

RQ6 How is the standardisation aspect dealt with across different platforms?

These research questions help us to determine the current state of cloud-based indoor positioning solutions. We can then establish the research questions which guide this dissertation based on the analysis of current challenges and future trends.

2.3.2 Keywords

The proper selection of keywords helps us to retrieve the most relevant research works related to our research field. Therefore, we established the following keywords according to 3 criteria: environment, infrastructure, and system.

Table 2.1 Criteria and keywords

Criteria	Search Keywords
Environment	Indoor*, GNSS-denied
Infrastructure	Cloud, Edge, Fog, MIST, computing, platform
System	Position*, location, localisation

Table 2.1 shows the keywords selected for this research process. The wildcard pattern (* in the queries) refers to one or more characters. For instance, position* matches anything starting with *position* such as positions, positioning, etc.

2.3.3 Search Query

The keywords defined in the previous step are used to form the research queries, which are then used in two well-known search engines (*Web Of Science* and *SCOPUS*) to find relevant works in the field of the systematic review.

Web Of Science Query to extend [32]:

```
TS=((((( cloud OR edge OR fog OR mist ) AND ( computing OR paradigm )
OR platform ) AND ( indoor* OR gnss-denied ) AND ( position* OR location
OR localisation ))) Timespan:  2021-2022
```

SCOPUS Query to extend [32]:

```
TITLE-ABS-KEY((((( cloud OR edge OR fog OR mist ) AND ( computing
```

```
OR paradigm )) OR platform ) AND ( indoor* OR gnss-denied )
AND ( position* OR location OR localisation )))
AND ( LIMIT-TO ( PUBYEAR , 2021 ) OR LIMIT-TO ( PUBYEAR , 2022 ))
```

The defined queries will return a list of studies, including books, journal and conference papers. Since not all of the retrieved studies are relevant, the PRISMA model proposes tangible steps towards a better selection of germane works.

2.3.4 Study selection

This section details the 6 stages undertaken to select the studies analysed: record identification, record screening, reports sought for retrieval, reports assessed for eligibility, new studies included in the review, and total studies reviewed.

Stage 1: Record Identification SCOPUS and Web of Science engines contain research works from a variety of sources, including conference papers, journals, books, and also research studies indexed in other search engines and repositories (e.g., IEEExplore, SpringerLink, ArXiv, etc.). SCOPUS and Web of Science engines thus are used to search for relevant studies for this review. After merging the results recovered from both the SCOPUS and Web of Science, a reference manager software is used to store, eliminate duplicate records and study classification.

Stage 2: Records Screening and Selection Criteria In this stage, the records obtained in the previous step are filtered using inclusion criteria (IC) and exclusion criteria (EC) listed below:

IC1 Full research works written in English

IC2 Research works dealing with platforms supporting positioning

EC1 Works not dealing with any computing paradigm (e.g., Cloud computing) or GNSS-denied scenarios

EC2 Works not published in peer-reviewed international journals or conference proceedings

EC3 Studies not dealing with wearable devices (we consider smartphones as wearable devices)

EC4 Studies not dealing with positioning, localisation or navigation

EC5 Studies not written in English

To ensure that all works extracted fulfil the *IC* and *EC* defined above, it is necessary to proceed with a systematic revision of titles and abstracts to determine whether the works are *ACCEPTED* or *REJECTED* (i.e., papers excluded are also labelled with the *EC*). Overall, only 13% of the new studies fulfilled the inclusion criteria.

Stage 3: Reports sought for retrieval This stage refers to the number of records obtained in the previous stage for full-text screening (records screened - records excluded). The total number of works sought for retrieval is 24.

Stage 4: Reports assessed for eligibility In this stage, we proceed with the systematic revision of each study sought for retrieval (review the full text).

Stage 5: New studies included in review The number of papers excluded is removed from the total number of studies reviewed for eligibility. In total, 24 new records are added to this review.

Stage 6: Total studies in review A total of 106 works (journal, magazine, and conference works) are included in this review, 83 from [32] and the 23 recent works. Figure 2.2 shows the PRISMA diagram and the results of each stage described above.

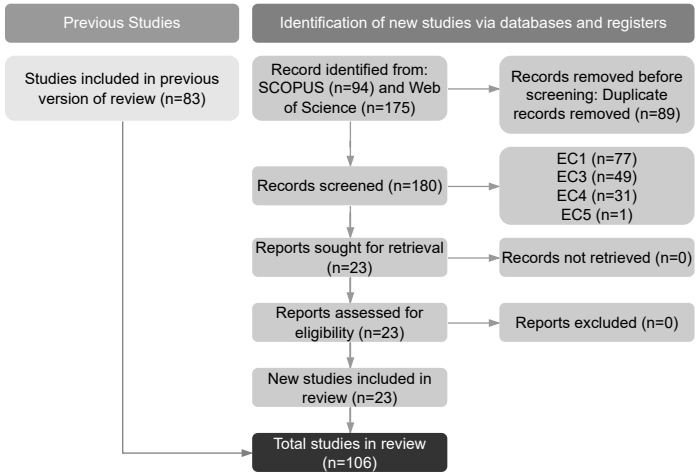


Figure 2.2 PRISMA Flow Diagram with the results obtained in each stage of the studies selection.

2.3.5 Overview of the studies classification and selection

At the end of the process, only 106 studies fulfilled all the criteria established and were, therefore, analysed (see Figure 2.2). The temporal distribution and type of research work are shown in Figure 2.3.

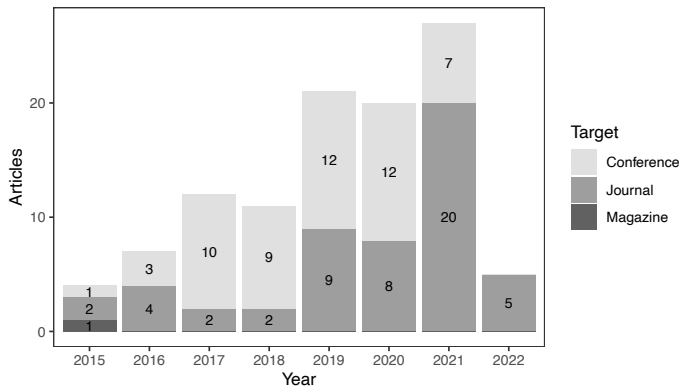


Figure 2.3 Distribution of the selected studies per year and target.

Of the 106 research studies selected, 54 were published in conferences, followed by 51 works published in different journals, and just a single magazine article. It is important to note that, since 2019, the number of papers related to this research field published in journals increased from 8 in 2020 to 20 in 2021. The total number of relevant papers listed for 2022 refers only to those published prior to May 2022.

2.3.6 Data extraction

This process is devoted to collecting all the relevant information from the 106 research works selected. This information covers the following aspects: computing paradigms used in recent IPS/ILS (RQ1), network protocols (RQ2), positioning/localisation technologies (RQ3), testing and deployment scenarios (RQ4), main goals and results achieved in each research work (RQ5) and the standards used in each IPS/ILS (RQ6). The key findings of this review are reflected in Section 2.4 and the papers analysed in Tables A.5–A.7 included in the Appendix A.1.

2.4 Results

This section presents a complete analysis of the 106 studies selected in relation to the research questions defined in Section 2.3.

2.4.1 Computing paradigms used in current indoor positioning platforms (RQ1)

The emergence of new computing paradigms has gone some way to meeting the needs of numerous devices and systems, especially in terms of computing and storage resources, security & privacy, and connectivity. These benefits have led to the deployment of systems and applications in the available computing paradigms, such as cloud computing. This increased use of cloud computing applies to Indoor Positioning System (IPS).

The works analysed featured 6 computing paradigms which are detailed below:

Cloud Computing (CC) The aim of this computing paradigm is to extend storage and computational capabilities to the Internet [34]. Generally, these services are allocated in large data centres which meet high standards. Cloud computing usually provides 3 main services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and SaaS. However, every day new services are emerging to satisfy customer needs, such as the Indoor Navigation as a Service (iNaaS) [5].

In the case of SaaS, all services are managed by the service provider. In PaaS, only data and applications are managed by the user. Finally, in IaaS, the user is in charge of the application, data, Operating System (OS), and middleware, the remaining services, such as virtualisation, storage, servers and networking, are managed by the cloud provider.

IPS/ILS have adopted these, offering navigation and localisation solutions as a service, such as the one developed by [5]. The cloud thus receives all the information collected by the users to retrieve the navigation instructions. Similarly, [35] provided a service to evaluate IPS/ILS, thus developers can test the accuracy of their solutions.

Given the numerous services that can be deployed in the cloud, some researchers have opted to deploy their indoor positioning/localisation, navigation and tracking systems in the cloud in order to avoid running heavy processes in the end-user device [6, 37, 8, 4, 38, 39, 40, 41, 42, 36].

Mobile Cloud Computing (MCC) This computing paradigm differs from the Cloud in that it combines Mobile computing with Cloud computing, meaning that, some IPS/ILS processes are executed in the user's mobile device, whilst processes which place a high demand on computational resources are executed in the Cloud to minimise the power and resource consumption in end-user devices.

According to Khan, Othman, Madani, and Khan [43], the main objectives of Mobile Cloud Computing (MCC) are related to *energy-consumption*, *performance*, *multi-objective MCC model* and *constrain devices*. These objectives are shown in research studies related to indoor positioning; for example, Huang, Zhao, Li, and Xu [44] suggested a novel indoor positioning solution, which offers a low-energy consumption on mobile devices without negatively affecting the position accuracy. It consists of dividing environments into n subareas and applying state controls in each of them. Similarly, Noreikis, Xiao, and Ylä-Jääski [45] provided efficient indoor positioning in terms of memory consumption and performance. The authors provided a vision-based indoor navigation solution, which offloads computing-intensive processes to the cloud.

Fog Computing (FC) Fog Computing (FC) was designed to decentralise systems or processes, the decentralisation of computational load being one of their main advantages, along with low latency and fast response time [46, 47].

Sciarrone, Fiandrino, Bisio, Lavagetto, Kliazovich, and Bouvry [48] deployed their platform in this computing paradigm in order to conserve power consumption for the user device. When the device's battery power fell below a pre-defined level, the computational load of the algorithm *FingerPrinting* (P-FP) would be automatically distributed to n devices nearby, reducing the energy consumption by more than 80%.

FC also enables massive device connectivity and centralizes the computing capabilities closer to the user devices, which improves response time. This advantage has been exploited by [49] and [16]. The platform developed by [49] permits thousands of devices to be connected to their indoor positioning platform without reducing its performance. Similarly, [16] used the FC to reduce computational load and provide massive device connectivity. In this case, the authors went a step further and used the storage capabilities of cloud computing to store historical data.

Mist Computing Generally, Mist Computing (MC) is used in cooperation with other computing paradigms such as FC. Like FC, this computing paradigm greatly

alleviates computational load and extends the computational and storage capabilities of FC. Various processes may take place in this computing paradigm, such as pattern recognition, data analysis, predictions, and position estimation [50]. In addition to the points mentioned above, this computing paradigm is used to reduce security & privacy issues, as in most cases, the data is not exposed to the cloud and is instead processed locally.

Edge Computing It is known for its high computational resources, low latency, massive device connectivity, and mobility support [34, 47, 51, 52]. Additionally, Edge Computing (EC) is geographically close to the data source or end-user, allowing real-time analytics and processes. Many research works have explored this paradigm with reference to these characteristics. For example, Liu, Si, Xu, He, and Zhang [53] used EC to store indoor positioning data and estimate the position of the user's device. Ben Ali, Hashemifar, and Dantu [31] proposed an Edge-SLAM system which distributes the positioning process between the user's device and the EC to reduce the computational load on the mobile device.

The disadvantage of the device connectivity associated with EC is the potential for data leakages and security issues that this can present. Several authors have proposed different mechanisms to tackle this potential problem and guarantee the security & privacy of indoor applications. Zhang, Chen, Peng, and Jiang [18] offered a new method based on the differential privacy-preserving model to ensure privacy during the offline stage of fingerprinting. Similarly, Liu and Yan [54] provided a security layer for EC for outsourced IPS using a verification schema.

Additionally, EC has also been combined with other computing paradigms such as MC and CC, demonstrating that the key characteristics of each computing paradigm can be exploited by IPSs [55].

Multi-access Edge Computing In contrast with EC, Multi-access Edge Computing (MEC) allows extending the network capabilities of mobile network operators or service providers. As a result, this computing paradigm or architecture has the characteristics of EC, plus the enhanced network capabilities of mobile networks. The advantages of MEC thus have caught the eye of the research community in the field of indoor positioning, increasing the number of researchers using MEC to deploy their IPS. For instance, Santa, Fernandez, Ortiz, Sanchez-Iborra, and Skarmeta [56]

used this computing paradigm to offload certain processes from the wearable device to MEC servers, exploiting the advantages of MEC and 5G technology.

MEC can also be combined with other computing paradigms such as mobile-cloud and cloud computing, according to the needs of the IPS/ILS. e.g., Horsmanheimo, Lembo, Tuomimaki, Huilla, Honkamaa, Laukkanen, and Kemppi [3] used mobile-MEC, where the positioning service was deployed in a MEC server with 5G capabilities. Carrera V., Zhao, Wenger, and Braun [11] combined MEC and CC where MEC is in charge of real-time localisation and CC of storage the localisation data.

2.4.2 Network protocols used in current Cloud-based Indoor Positioning Platforms (RQ2)

Network protocols establish a set of rules to ensure communication between devices and/or systems in the same network. Although there are multiple network protocols, it should be taken into consideration that only those protocols used in current research works retrieved from the study selection are covered here. These protocols are divided into 4 groups: communication, security, IoT, and other protocols used in the studies analysed.

Communication Protocols They are a set of rules which determine how data is transmitted and exchanged across the network, allowing reliable communication between devices. Currently, there are many communication protocols, such as HyperText Transfer Protocol (HTTP) and Message Queuing Telemetry Transport (MQTT), which are selected according to the software requirements.

The papers analysed mainly used at least one of the following 5 communications protocols: HTTP as part of the Transport Control Protocol (TCP)/Internet Protocol (IP) protocol suite, User Datagram Protocol (UDP), WebSocket, OpenFlow and OBject EXchange (OBEX). HTTP is used to exchange information between the web server (in this case, the IPS) and the web client. Some authors used REpresentational State Transfer (REST) architecture to exchange data between client and server through HTTP [35, 57, 58, 7, 59, 15, 60, 6], but REST is not limited to HTTP. For example, Sykes [61] used a REST Application Programming Interface (API) service to update the user location when the user reaches a place.

Limitations of HTTP include multiple underlying TCP connections for each client and each message, with the client-side having to maintain a mapping from

outgoing and incoming connections. The WebSocket protocol overcomes these limitations, providing reliable connections between server and client. Modern IPS are using this protocol to connect the client and the web core [62, 11].

In view of the rapid increase of wearable and IoT devices consuming positioning and localisation services, it is essential to use lightweight protocols. For this reason, UDP protocol is becoming increasingly popular in IPS oriented for wearable devices. For instance, Chen, Hsieh, Liao, and Yin [63] used UDP protocol to upload the data collected with the wristband to the server in order to estimate the user position.

OpenFlow protocol has also been used in current IPS implementations. This protocol is used to determine the path for message forwarding; part of the routing decision is thus done in the OpenFlow switches, and part in a controller, allowing the optimisation of network resources. Guo, Zhao, Wang, Liu, and Qiu [64] used this protocol to exchange position information between the OpenFlow switch and the FC layer.

OBEX protocol allows efficient exchange of binary data between Bluetooth-enabled devices. Li, Wu, Wang, Chu, and Liu [49] used this protocol to share position information between the user device and Lbeacons.

Security Protocols The primary weakness of cloud-based indoor positioning applications is the potential security & privacy breaches that can arise, given that the users have to share sensitive data with the cloud to estimate their position. During the communication between the user and the server, the user data may be susceptible to multiple attacks such as spoofing [65, 66]. Additionally, attackers can generate and emit fake RSS values to affect the position estimation.

Researchers have proposed and implemented multiple mechanisms for the provision of secure indoor positioning solutions. For example, Transport Layer Security (TLS) protocol has been implemented to ensure that communication between client and server is secure, encrypting the communication between both sides [67]. This cartographic protocol uses two extra protocols to guarantee the confidentiality and integrity of communication, namely TLS record and TLS handshake protocols. TLS handshake protocol controls the session negotiation, whereas TLS record protocol ensures the confidentiality of the communication by using two mechanisms: symmetric key cryptography and checksum [68].

IoT protocols The increasing demand for localisation and tracking services by IoT devices means that new protocols have been designed to facilitate lightweight and efficient communication among devices. Some of the most common IoT protocols employed in the selected research studies are: MQTT and Extensible Messaging and Presence Protocol (XMPP).

MQTT protocol is composed of 3 main components: broker, publisher, and subscriber [69]. The broker takes the responsibility of dispatching the messages among clients. The publisher represents any device (client) that transmits messages through the broker, and the subscriber is the (client) device connected to the broker. In the case of indoor positioning applications, this protocol is used for its low power consumption, which makes it suitable for wearable and IoT devices [37].

The indoor positioning solutions analysed are using MQTT protocol to encrypt real-time communications, dispatch RSS values, and communicate the user device with different computing paradigms such as EC [70, 37, 8, 46, 50, 71, 72].

XMPP is another IoT protocol used in the analysed studies. This is a flexible and decentralised protocol developed for instant messaging and video calls. Additionally, this protocol allows the exchange of messages between Jabber/XMPP clients and servers [69]. [73] used the XMPP protocol to select the most suitable portable sensing units (PSUs) and communicate them with the XMPP server, using the k-d tree algorithm. The authors thus provide a fast and reliable indoor positioning system [73].

Additional protocols In the hope of providing efficient indoor positioning solutions, researchers have tested different communication protocols, including protocols used in Voice over IP (VoIP) such as Session Initiation Protocol (SIP), and also Location-to-Service Translation Protocol (LoST) protocol for location-based services that use a geographic location.

2.4.3 Do the current platforms permit heterogeneous positioning technologies for GNSS-denied scenarios? (RQ3)

Current cloud-based indoor positioning solutions provide support to a variety of positioning technologies that contribute to the estimation of user position. These positioning technologies are diverse, providing different levels of accuracy depending

on user requirements. For instance, we have technologies based on radio frequency, light and sound, among others.

This section describes all the positioning technologies featured in the works analysed, which have been classified into 5 categories: radiofrequency technologies, magnetic field, inertial sensors, computer vision, sound, and optical technologies. This section also covers the techniques and algorithms used with the technologies.

2.4.3.1 Radiofrequency Technologies

Radiofrequency-based technologies are commonly used in indoor positioning applications because of the prevalence of radio frequency signals in indoor and outdoor environments; signals from cellular base stations, Wi-Fi, UWB and Bluetooth. Although most of them have not been designed for positioning purposes, they are currently used to support indoor positioning platforms.

IEEE 802.11 Wireless LAN (Wi-Fi) Wi-Fi is a wireless technology supported by numerous devices such as wearable, IoT devices, and computers [6, 21]. This technology belongs to the IEEE 802.11 standards, and it is commonly used for indoor positioning platforms due to its availability and low cost [74].

Along with Wi-Fi technology, multiple algorithms and techniques were presented within the literature. For example, the fingerprinting technique, which is divided into two phases: the offline where RSS measurements are collected at reference points in order to build a radio map, and the online phase where RSS measurements are collected (unknown positions) to estimate the device position using matching algorithms [75, 56]. RSS values can also be used to estimate user position by applying signal propagation models [76].

In order to offer a most robust solution, some authors have combined Wi-Fi fingerprinting with other techniques such as Time of Arrival (ToA). ToA is also known as time-of-flight and is used to measure the exact time that a signal is sent from the source and the time to reach the target device. Lemic, Handziski, Wirström, Van Haute, De Poorter, Voigt, and Wolisz [77] used these two techniques to evaluate indoor positioning algorithms.

Some authors have also combined Wi-Fi technology with different methods and techniques, acquiring differing levels of accuracy. For instance, Konstantinidis, Demetriades, and Pericleous [78] acquired $\approx 1\,\mathrm{m}$ by combining Wi-Fi with Multi-Objective

Fingerprint Selection Optimization Problem (MO-FSOP). Chen, Hsieh, Liao, and Yin [63] combined fuzzy logic and genetic algorithms, obtaining a mean positioning error of 2 m on average.

Bluetooth Bluetooth is a Wireless Personal Area Network (WPAN) for short-range communications. The latest versions of Bluetooth offer low-energy consumption suitable for power-constrained devices. Additionally, Bluetooth v5.1 includes some improvements for indoor positioning, providing centimetre level accuracy [79, 37].

Generally, Bluetooth-based indoor positioning solutions require the deployment of BLE devices (e.g., beacons, iBeacons, custom emitters) in the environment to estimate the device position [40, 80]. The distance between two or more Bluetooth devices can be determined using, i.e., the RSS of BLE advertisements.

The selected papers use two protocols based on BLE: iBeacon developed by Apple company, and Eddystone developed by Google. These two protocols provide proximity services using 4 regions to determine the proximity of devices: *unknown* (device not ranged), *immediate* (between 0 m and ≈1 m), *near* (between 1 m and ≈3 m) and *far* (between 3 m and ≈50 m) region [21, 61]. However, the complexity of indoor environments may affect the accuracy of the proximity estimation.

Some researchers have proposed the use of filtering techniques such as Winsorization, Trimmed Mean, and Kalman filter (KF) [60, 81, 81, 71] to reduce the errors produced by undesirable fluctuations in signal propagation. Calibration techniques are also used to offer a robust solution in terms of positioning accuracy [40].

As with Wi-Fi, fingerprinting can be used with BLE to estimate the device position. Other techniques such as trilateration, multilateration, and triangulation have been also used in the studies analyses [71, 81, 82]. Trilateration uses 3 reference points to compute the device's position. Multilateration determines the position of a target point by measuring the time of arrival of the signal transmitted by the base stations. Triangulation uses angles to determine user position [20, 83].

Unlike Wi-Fi, the accuracy achieved with BLE may range in the centimetre level. e.g., Li, Cao, Liu, Zhang, Hu, and Yao [37] combined KF, Long short-term memory (LSTM) + Tri (Multi-Weighted-Centroid) acquiring centimetre level accuracy (≈ 0.86 m).

Ultra Wideband (UWB) This radio frequency technology provides a wide spectrum and high bandwidth, which make it suitable for different applications such as indoor positioning [10, 21, 84].

Modern indoor positioning applications use this technology to provide centimetre-level accuracy. For instance, UWB has been used in applications to analyse mobility patterns in people with dementia [10]. In this case, the authors combined UWB, ML (Support Vector Machine (SVM) + k-NN) and EC for this analysis.

Carrera V., Zhao, Wenger, and Braun [11] combined 3 ML algorithms CART, KStart, and a Multilayer Perceptron (MLP) network, acquiring 0.59 m of accuracy. Similarly, Barua, Dong, Al-Turjman, and Yang [10] used EC to run complex algorithms and determine the user position and provide quick response (real-time application).

Cellular/Mobile Networks Cellular is a technology for mobile communications which allows a large number of devices to connect simultaneously. This technology has evolved over the years, distinguishing different generations, from 1G to 5G. However, only the last 4 generations have been used for indoor positioning [73, 3, 85].

Generally, this technology is combined with BLE, Wi-Fi or others in order to reduce the position error. For example, Santa, Fernandez, Ortiz, Sanchez-Iborra, and Skarmeta [56] proposed an positioning solution for indoor and outdoor environments, which supports: mobile networks, GPS, Wi-Fi, BLE and Near-field Communication (NFC), acquiring an accuracy of 4.61 m on average. Similarly, Kulshrestha, Saxena, Niyogi, and Cao [73] combined the technologies of 3 mobile networks, GPS and Wi-Fi, offering a novel solution for indoor and outdoor environments. They tested the proposed approach in a real scenario, detecting the last location of a person and the number of people in the environment.

IEEE 802.15.4 – Zigbee Zigbee is a wireless technology developed to enable low-power consumption, IoT and machine-to-machine communications. Zigbee supports centralised and decentralised networks, adapting to the network requirements. This technology is becoming more and more used in smart environments [83].

Zigbee networks combined with other technologies, such as Bluetooth, have been used in the literature for indoor positioning purposes. For instance, Li, Wu, Wang, Chu, and Liu [49] combined BLE-based technologies and Zigbee for fine indoor

localisation. Zigbee star network thus connects Lbeacons and is used for monitoring and initialisation. Likewise, Chen and Huang [86] used the Zigbee network to efficiently coordinate the messages from the connected devices or terminals.

Radio Frequency Identifier (RFID) This technology operates in different frequencies, including low, high, Very High Frequency (VHF), Ultra High Frequency (UHF), and microwave frequency [21, 87]. RFID tags are generally classified into: active, passive, and semi-passive. The key difference is given by the number of components. For instance, active RFID tags consist of an integrated circuit, an onboard transmitter, a battery, and an antenna. Passive tags are composed of two components, an integrated circuit and an antenna. The semi-passive tag is composed of an integrated circuit, a battery, and an antenna [87].

Fang, Cho, Zhang, and Perez [88] implemented a passive RFID-based indoor localisation system. In the proposed system, the antenna is responsible for the reading of tags data, and the reader transmits the data to the cloud using Wi-Fi access points. This system tracks workers according to their proximity to the RFID antennas, acquiring an accuracy rate of 88.1% approximately. Datt, Senapathi, and Mirza [89] stated that RFID technology can be useful for indoor navigation where accuracy and real-time communication is the primary requirement.

2.4.3.2 Magnetic Field

A particular benefit of using magnetic fields for indoor positioning applications is that they are stable over time. The use of magnetic fields prevents the need to deploy additional devices, as in the case of BLE technology. Similar to Radio Frequency (RF)-based indoor positioning technologies, fingerprinting technique can be used along with this technology, as each scenario has special characteristics in terms of magnetic intensity (i.e., ferromagnetic materials used in constructions such as buildings give rise to unique magnetic levels).

Liu, Guo, Yang, Shi, and Chen [90] proposed a novel geomagnetism-based indoor navigation application using the fingerprinting technique. In the offline phase, the geomagnetic signals are collected in pre-established reference points to build a geomagnetic dataset. During the online phase, new measurements are collected in order to estimate the user position using 3 algorithms; Dynamic time warping (DTW), particle filter (PF), and k-NN.

2.4.3.3 Inertial sensors

This technology is also called Indoor Navigation System (INS), which uses data obtained from the gyroscope and accelerometer. This technology is frequently used in collaboration with other technologies such as Wi-Fi and/or BLE to diminish the error accumulated during its use [83, 42, 15, 45, 91]. Carrera V., Zhao, Wenger, and Braun [11] provided a new real-time indoor positioning solution that combines CC, MEC, UWB and INS. As a result, Carrera V., Zhao, Wenger, and Braun [11] obtained a tracking error of 0.44 m on average.

Nikolovski, Lameski, and Chorbev [62] put forward a robust system for Ambient-Assisted living. The authors used INS for falling detection. As INS tends to accumulate errors, the author used multiple filtering techniques (e.g., KF) to reduce the error.

2.4.3.4 Computer Vision-based Technology

Computer vision technology has become a cornerstone for Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR), is gaining widespread acceptance in indoor applications. This technology attempts to acquire visual information from images or videos obtained from cameras at a pixel level [83]. In the case of indoor navigation, computer vision is used to obtain information from the environment, such as obstacle recognition and pathfinding [92, 45].

As indoor applications based on computer vision usually require high computational resources, some of the positional processes are offloaded to the cloud or other computing paradigms to avoid an overload in the user device. For instance, the application developed by [93] used the cloud to process the images captured by the wearable belt designed by the authors. As a result of their proposal, the authors determined the obstacle proximity using two main technologies; computer vision and sonar technology.

Computer vision technology is considered one of the most accurate technologies for indoor positioning, acquiring centimetre-level accuracy. e.g., Zhao, Xu, Qi, Hu, Wang, and Runge [94] obtained a position accuracy of 60 cm in an area of 7.2 m $\times$ 5.2 m with similar results in 3 different environments where the proposed solution was tested.

2.4.3.5 Sound-based technologies

Sound-based technologies can be divided into two groups based on the frequency, audible sound-based technology and ultrasound. Audible sound technologies operate within frequencies in the range of the human audibility (20 kHz) [21]. Whereas ultrasound technology operates in frequencies greater than 20 kHz, which are not detectable for the human ear.

In some indoor navigation solutions, these technologies are used to detect objects. For instance, Silva and Wimalaratne [93] used two ultrasonic sensors to detect obstacles in the venue. As mentioned above, these technologies were combined with computer vision to provide an efficient indoor navigation solution for visually impaired people.

2.4.3.6 Optical technologies

The popularity of this technology for indoor positioning applications is on the rise, as it remains unaffected by electromagnetic interference, and boasts low power consumption (e.g., VLC). These characteristics make this technology suitable for resource-constrained devices. Additionally, technologies like VLC provide high levels of position accuracy [12]. Despite these advantages, optical technologies may be affected by other factors such as ambient light noise, multipath, and delays in communication.

In the case of optical technologies, modulated and unmodulated light is used in indoor positioning applications. Unmodulated light uses sensors to quantify the light intensity. For instance, Liu, Jiang, Jiang, Liu, Ma, Jia, and Xiao [98] proposed a new indoor navigation solution that combines unmodulated light and inertial sensors. The authors measure the light intensity of the luminaries deployed in the environment, so peaks in the intensity are associated with virtual graphs to provide the navigation path to the user, resulting in an average accuracy of 90%.

Table 2.2 summarises the indoor positioning technologies analysed in this chapter, the range, accuracy reported in the analysed studies and their power consumption. As shown, UWB, camera-based, and ultrasound may provide centimetre level accuracy in the position estimation, whereas the remaining positioning technologies are in the range of meters.

Table 2.2 Indoor positioning technologies and their characteristics.

Tech. Group	Technology	Range	Accuracy	Power cons.
Radiofrequency	Mobile network [95]	500m - 80km [a]	<50m [20]	Moderate-low
	Wi-Fi [96]	< 100m [b]	avg. > 1m [97, 53, 63]	Moderate
	Bluetooth [79, 44, 37]	v2.1-4.0 → 100m, v5.0 → 400m	avg. > 1.5m [44, 8]	Low
	UWB [20]	10-20m	median < 50cm [20]	Low
	Zigbee [95]	100m	median < 5m [20]	Low
	RFID	200m	median < 3m [88]	Low
Magnetic Field	-	-	median < 5m [20]	Low
Inertial sensors	Gyroscope, accelerometer, etc.	-	<5m [88] [c]	Low
Vision	Camera	-	avg. ≈ 20cm [31]	High
Sound	Ultrasound [21]	< 20m	median < 10cm [20]	Low
	Audible Sound	-	-	Low
Optical	Light	-	-	Low

[a] is based on the mobile network generation (3G-5G)
[b] it relies on the standard (e.g., IEEE 802.11a, IEEE 802.11g, etc)
[c] the error may increase according to the distance travelled.
Accuracy as reported in the analysed research works. Table reproduced with permission from [32].

2.4.4 Do the current platforms adapt to different scenarios? (RQ4)

Considering that indoor scenarios are heterogeneous and complex, it is important to analyse factors such as the number of buildings and floors, accuracy required, and purpose (e.g., navigation, localisation or tracking) prior to deploying a certain IPS. It is, therefore, essential to test the proposed indoor positioning/localisation solutions in multiple scenarios in order to evaluate their scalability, robustness, and adaptability.

This section provides an analysis of the selected research works from the perspective of indoor positioning software, environment/venue, and devices.

Indoor Positioning Platform Currently, the hosting models provided by the cloud permit a fast and straightforward deployment of applications within minutes. For example, in PaaS, most services are managed by the cloud provider (e.g., OS, virtualisation, middleware, storage, etc.). The developer, therefore, is only in charge of the software and data.

Currently, many service providers also offer additional services such as API gate-

ways, databases as a service, cloud functions, and data analysis. These services have been exploited by current indoor positioning solutions, shortening the coding and deployment time. For example, Terán, Carrillo, and Parra [38] and P. Álvarez and N. Hernández and Fco. Javier Fabra and M. Ocaña [6] developed their solution using: S3 Service, Amazon Dinamo DB, Amazon API Gateway and Amazon Machine Learning. Although these services help when planning to deploy an indoor positioning platform, these IPSs/ILSs should be independent and capable of being implemented in any service provider or local infrastructure. Additionally, these indoor positioning platforms must be modular and flexible, allowing the addition of new and old technologies.

Environment In the light of the complexity of GNSS-denied scenarios, the proposed indoor positioning solutions should be tested in multiple scenarios to verify their robustness and adaptability. Some of the research papers analysed, however, used only a single scenario for evaluation [55, 38, 44], and just a few articles tested their solutions in multiples scenarios, including outdoor environments [61, 99, 15, 100]. The test area used in each research work also differs, with some cloud-based indoor positioning solutions being tested in areas of a few square meters, while others were tested in large areas of 562 000 m^2, 1000 m^2, 20 000 m^2, 1000 m^2 [6, 98, 101].

The type of scenario in which the system is tested is of paramount importance in providing an efficient indoor positioning solution. 5 scenarios were most frequently used within the works analysed in order to test their solutions: Universities [8], shopping malls [6], libraries [99], and residential buildings [46]. It is important to highlight that the number of people present in the testing environment can affect the accuracy of the positioning platform; thus, determining if the proposed indoor positioning solution is suitable for crowded places is also relevant.

Client The user device has been the case of study in different research works in the field of indoor positioning, as accurate indoor positioning applications may consume high computational resources from the user device. This excessive consumption of the computational resources of the user device speeds up the deterioration of the device and its battery. That is why some research studies centred on efficient computation and how to offload computing-intensive process [6, 31, 48]. For instance, Santa, Fernandez, Ortiz, Sanchez-Iborra, and Skarmeta [56] used the EC to offload

computing-intensive processes from the user device to this computing paradigm to alleviate the computational load exerted by indoor positioning applications.

Efficient ways to provide positioning services to users are also analysed by the research community. Currently, clients can consume indoor positioning and localisation services through different methods offered by modern IPS [59]. For example, data and services are exposed to clients through APIs [40, 2, 92, 100, 7, 59, 41, 42, 15, 57, 58], web services [38], and HTTP(s) request [75], which allow a straightforward implementation from the client side. Considering this, some authors have developed indoor positioning applications that interact with the cloud using these services (e.g., maps, messages, routes, positioning, localisation services, etc.) [59, 7].

2.4.5 What improvements were done in similar studies (RQ5)

The studies analysed were focused on covering existing issues or limitations of current cloud-based indoor positioning platforms. Thus, some authors are focused on, for example, improving the positioning accuracy, reducing computational load, or providing a useful application for the end-user. To analyse the objectives more commonly addressed in the selected research works, we have created a common framework which summarises each of the improvements outlined in each study. These objectives are described below.

Here, it is necessary to clarify that localisation and positioning are not used as synonyms in this research work, as they have different meanings. We have adopted the definitions provided by Sithole and Zlatanova [102], where a position is represented by a coordinate (latitude, longitude and altitude or x, y, z), whereas a location represents a physical space (e.g., class A113, Auditorium, etc.). Therefore, localisation constrains the position of a user/device to a specific place or area instead of providing its exact coordinates.

- Computational efficiency [6, 70, 75, 31, 3, 48, 42, 94, 38, 64] refers to the methods used by the authors to reduce the amount of computational resources used by the positioning/localisation/navigation algorithms.
- Interoperability [71, 7, 58, 59] represents the capability of indoor positioning platforms to interact with other systems and technologies. It is particularly important in view of the heterogeneity of indoor applications and because other systems use the offered positioning, localisation, and navigation services.

- Positioning [17, 64, 3, 15, 72, 44]: refers to any improvement in the reduction of positioning error.
- Usability [103, 2, 60, 105, 62, 86, 80, 89, 35, 73, 57, 77, 104, 59]: refers to the quality of user's experience and how easy-to-use is the suggested positioning/localisation/navigation solution.
- Localisation [61, 55, 37, 92, 46, 88, 10, 5, 81, 106, 101, 11, 76, 44, 107]: makes reference to recent improvements in relation to localisation.
- Cost [6, 70, 82, 53, 63]: Cost is always a consideration when designing indoor positioning systems; that is why this parameter has been taken into account in some research works.
- Navigation [8, 108, 93, 109, 100, 90, 110, 89, 45, 15]: considers whether proposed solutions offer navigation and routing services.
- Scalability [49, 88, 100, 17]: examines the capability of the IPS to adapt to different scenarios, technologies and support concurrent connections without reducing performance.
- Low latency [50, 80, 89]: is one of the primary factors to consider when developing indoor positioning systems, more when fast response and real-time communication are required.
- Energy efficiency [97, 98, 111, 45, 44]: This item covers the methods used by authors to reduce energy consumption while performing a process. It is especially important in resource-constrained devices.
- Reliability [88, 77]: represents how well an indoor positioning system performs over time under different conditions and scenarios in relation to accuracy.
- Tracking [101]: considers whether the analysed indoor positioning solutions provide tracking services.
- Evaluation [35]: considers whether the studies analysed used any method to evaluate the performance of the solutions proposed, and whether the IPS follows any standard for the evaluation.
- Privacy [18]: relates to the mechanisms developed and implemented in each research work with a view to protecting the privacy of user information, such as the differential-privacy algorithm developed by [18].

- Security [16, 67, 99, 54, 112]: relates to the algorithms, methods and protocols developed or used by researchers to reduce security gaps in indoor positioning applications.

2.4.6 How is the standardization aspect focused on different platforms? (RQ6)

Nowadays, positioning, localisation and/or navigation services are consumed by a great variety of systems such as fitness applications, manufacturing and even between IPS. To achieve this level of integration IPS must fulfil specific standards established by standardisation entities such as ISO. These standards contain a set of technical specifications developed by a community of experts in a specific area.

In this section, we analyse the standards used in current indoor positioning systems based on 5 components: maps, positioning technologies, evaluation, and software architecture.

Maps Currently, multiple standards are available for outdoor and indoor maps. These provide a set of specifications to exchange spatial data under a common framework, including geospatial data representation, symbols, and coordinate systems, among others [113, 32]. Some of the most well-known standards are provided by the Open Geospatial Consortium (OGC). This entity provides a set of free and public standards to support new technologies and their interoperability. IndoorGML is part of the standards offered by OGC, and it corresponds to a set of specifications for data models and Extensible Markup Language (XML) schemes for indoor spatial information. This standard thus covers the representation of cellular space, semantic representation, geometric representation, topological representation, and multi-layered representation, e.g., XML: `<xs:complexType name="IndoorFeaturesType">`. There are other schemas provided by other entities, such as OpenStreetMaps. In the case of Indoor OpenStreetMaps, it includes a complete set of tags for indoor mapping. e.g., `stairs=yes, indoor=room, amenity=restaurant`. Some authors are also using Building Information Modeling (BIM) [88, 80] to manage indoor information under the ISO 19650.

In the research works analysed, some authors used OGC standards to manage and represent indoor and outdoor maps [57]. Here we can observe the use of third-party maps which already fulfil some standards, e.g., Google Maps [61, 60, 109, 100, 89, 7, 2], and also the use of custom maps [8, 40, 53, 99, 63, 3, 71, 101]

Position Technologies Section 2.4.3 provided an analysis of indoor position technologies used in the selected studies. In general, most of the indoor positioning technologies used in the research studies analysed fulfil specific standards. For instance, Bluetooth under the standard IEEE 802.15.1, Zigbee - IEEE 802.15.4, UWB built upon the standard IEEE 802.15.4z, Wi-Fi - IEEE 802.11. In the case of RFID, it is in several standards such as EPCglobal and ISO 18000.

Evaluation methods Currently, ISO/IEC 18305:2016 Information technology – Real time locating systems – Test and evaluation of localisation and tracking systems is one of the standards used to evaluate indoor positioning system [114, 32]. This standard consists of a set of guidelines for the standardisation of IPS/ILS which have been widely discussed within the indoor positioning community [115, 116]. ISO/IEC 18305:2016 includes metrics to evaluate the performance of indoor positioning applications, considerations of security & privacy, taxonomies and methodologies; highlighting the importance of testing IPS to provide reproducible and replicable solutions.

Software architecture Software design and structure are fundamental to achieving reliable, efficient, and robust IPSs. The software architecture also determines the scalability of the indoor positioning solution, meaning this topic has been addressed in some of the analysed papers. In general, the studies analysed use different design patterns, such as, monolithic architecture, Microservice Architecture (MSA), Model–View–Viewmodel (MVVM), Model–View–Controller (MVC)), and Cloud native architecture. For example, Mpeis, Roussel, Kumar, Costa, LaoudiasDenis, Capot-Ray, and Zeinalipour-Yazti [15] applied MSA pattern in their indoor navigation platform. [5, 7, 58] opted by Service Oriented Architecture (SOA) and [45, 2, 8, 105, 38] used cloud-native architecture.

MSA architecture divides the system into small independent microservices which interact between them and other services through their interfaces using lightweight protocols (e.g., XMPP, MQTT, WebSockets, etc.). In contrast, monolithic architecture is built as a single unit. SOA implements services which use their interfaces to communicate with other services. MVC design pattern shares similar characteristics with MSA, being both decoupled architectures. However, an app using MVC is based on 3 components, model, view and controller, whereas a microservice-based app is divided into small specialised services. MVVM is commonly used to separate

the control layer from the view layer. In the case of cloud-native architecture, it exploits the benefits of the cloud to provide scalable and resilient applications. This architecture uses other patterns such as MSA.

Part of these design patterns are described in ISO/IEC/IEEE 42020:2019 – Software, systems and enterprise — Architecture processes [117]. This standard includes a detailed description of the types of architectures used in software development, process, software implementation, and other relevant information.

2.5 Discussion of the state-of-the-art

Section 2.4 addressed 6 research questions using the methodological procedure described in the PRISMA model. This section summarises the key findings of the previous section and examines the future trends and current challenges.

2.5.1 Computing paradigms and improvements (RQ1 and RQ5)

Section 2.4.1 featured a full review of the computing paradigms used in the selected research studies. The review concluded that CC (49 articles [62, 35, 77, 2, 108, 57, 58, 75, 40, 105, 17, 100, 5, 42, 38, 104, 6, 71, 59, 72, 70, 37, 81, 15, 107, 118, 119, 36, 39, 85, 120, 12, 121, 91, 122, 123, 121, 124, 125, 126, 127, 128, 129, 130, 120, 131, 91, 132, 32]) and MCC (24 articles [103, 88, 80, 89, 101, 45, 82, 99, 63, 78, 67, 109, 90, 98, 106, 7, 111, 44, 61, 60, 92, 93, 86, 73]) are the most frequently used computer paradigms (see Fig. 2.4), being present in at least one research study per year during the relevant period. CC, is often combined with other computing paradigms such as MEC (4 articles [56, 3, 76, 133]), FC (7 articles [48, 49, 64, 16, 134, 135, 136]), and EC (14 articles [53, 18, 31, 110, 54, 52, 51, 142, 143, 137, 138, 139, 140, 141]).

The use of these computing paradigms addresses IPS needs such improvement of positioning accuracy, reduction of deployment costs, reduction of energy consumption in the user device, and the provision of privacy & security (see Fig. 2.5). In general, CC and MCC is used to provide a better experience to the end-user (20 articles [103, 62, 77, 2, 57, 105, 80, 89, 104, 59, 61, 60, 86, 73, 118, 119, 36, 123, 124, 129] focus on usability). EC is used to diminish the computational intensity in the user device (6 articles energy efficiency [31, 94, 51, 142, 138, 139]). The combination of CC-EC, FC-CC and MEC-CC is used to improve the localisation accuracy. FC is mostly

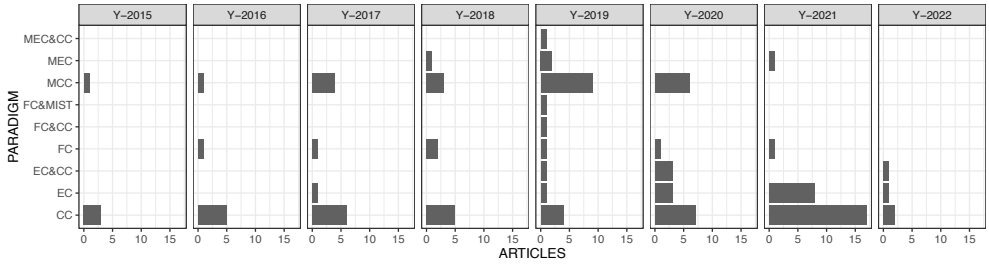


Figure 2.4 Cloud Platform by Year.

used to provide energy efficiency and security. FC and MC are combined to provide a fast response to the end-user, commonly used in real-time applications. MEC is used to reduce the localisation error and computational load in the user device.

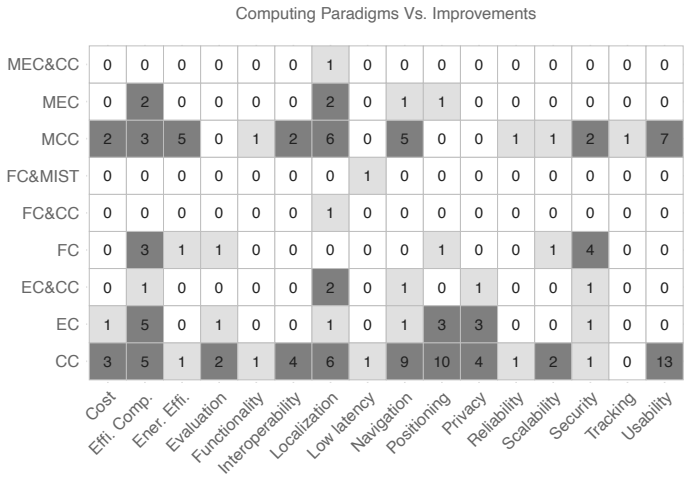


Figure 2.5 Main goals of the analysed studies classified by computing paradigm. Reproduced with the permission from [32].

Section 2.4, demonstrated that the algorithms proposed or tested often required high computational and storage resources, leading to opt to offload computing-intensive processes to some of the computing paradigms analysed in the previous section. As a result, the authors provided an IPS/ILS more efficient and robust.

Fig. 2.5 also shows what objectives are the most achieved in the analysed works. The main goals thus are usability with 20 articles addressing this goal [103, 62, 77, 2, 57, 105, 80, 89, 104, 59, 61, 60, 86, 73, 118, 119, 36, 123, 124, 129], in second place is computational efficiency with 19 articles [48, 75, 80, 89, 45, 56, 42, 38, 6, 64, 3, 70, 31, 94, 134, 51, 142, 138, 139]. And third, we have localisation with 18

articles [101, 5, 55, 46, 106, 11, 76, 44, 61, 37, 92, 10, 81, 107, 85, 120, 133, 138]. It is important to highlight that some of the papers examined addressed more than one goal, e.g., cost and efficient computation, efficient computation and positioning, and so on.

2.5.2 Network protocols (RQ2)

The selection of network protocols (see Fig. 2.6) is essential to any cloud-based indoor positioning solution to ensure the robustness, security, and scalability of the proposed IPSs. As discussed in previous paragraphs, the authors selected protocols according to the user requirements and services provided by the IPS. For instance, if the IPS provides real-time functionalities, the authors opted for lightweight protocols such as UDP, MQTT. Some of the proposed IPS/ILS also provides encrypted end-to-end communication by adding an Secure Sockets Layer (SSL)/TLS protocols. Custom algorithms, methods or devices to add a security & privacy layer to the indoor positioning solutions have also been offered by researchers [67, 99, 67, 16, 54].

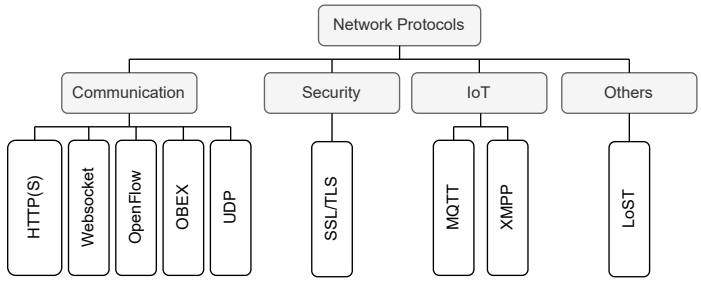


Figure 2.6 Network protocols employed in the research works selected.

In spite of the relevance of network protocols in indoor positioning applications, most of the papers analysed in this review do not provide this information. Overall, less than 40% of the reviewed papers reported the type of protocol used in their systems.

2.5.3 Indoor positioning technologies (RQ3)

The solutions proposed in the literature selected for analysis made use of a wide range of indoor positioning technologies, including Wi-Fi, BLE, and ultrasound, among others (see Fig. 2.7). Although there are many indoor positioning technologies,

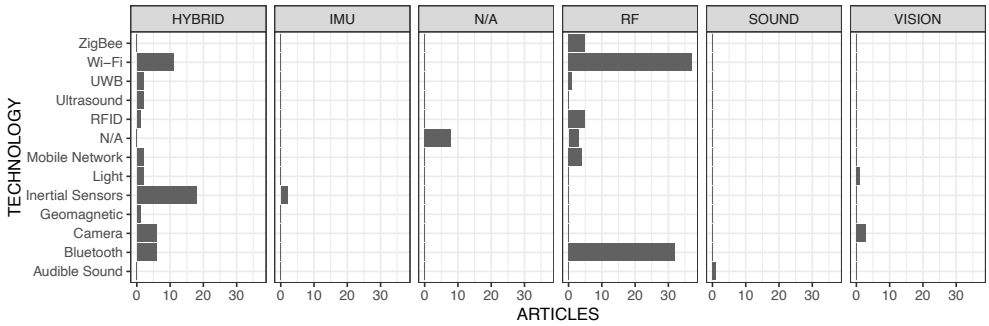


Figure 2.7 Indoor positioning technologies used in the research works selected.

Wi-Fi is still the most popular indoor positioning technology, which was used in 47 research studies [77, 88, 48, 75, 105, 17, 53, 100, 99, 63, 56, 6, 55, 78, 67, 46, 18, 64, 90, 3, 106, 11, 7, 76, 59, 111, 72, 16, 73, 15, 107, 119, 39, 52, 136, 51, 121, 91, 144, 121, 142, 125, 143, 138, 140, 141, 131]. Bluetooth-based is the second-ranked indoor positioning technology, with 38 articles [2, 57, 40, 17, 100, 80, 119, 82, 49, 99, 5, 56, 42, 38, 55, 67, 46, 109, 50, 71, 44, 61, 60, 8, 37, 110, 81, 73, 15, 120, 122, 144, 124, 142, 127, 128, 130, 120]. Inertial sensor-based indoor positioning solutions came in third place, with 20 articles [62, 17, 100, 45, 42, 109, 90, 98, 11, 7, 111, 15, 36, 12, 91, 121, 139, 127, 140, 91].

Expansion in the use of laser, ultrasound or UWB in indoor positioning solutions is currently limited by the small number of user devices which support them. Although most of these technologies provide better performance than Wi-Fi in terms of positioning accuracy, not all users can access them through their mobile devices, unlike in the case of Wi-Fi and BLE.

Section 2.4.3 and Fig. 2.7 demonstrated that indoor positioning platform usually combines indoor positioning technologies to provide an accurate solution to the end-user. For instance, Wi-Fi + BLE [144, 145] and Wi-Fi + inertial sensors [140]. The researchers not only used or combined different positioning technologies, but also numerous custom algorithms and ML-based models (e.g., *k*-NN, LSTM, CNN, etc.).

2.5.4 Cloud-based indoor positioning platforms - scenarios (RQ4)

In order to ascertain whether current indoor positioning solutions are robust enough to be used in multiple scenarios, we analysed 3 related components: indoor positioning platform, client, and environment (see section 2.4.4)

Section 2.4.4 shows that current indoor positioning platforms are exploiting most of the resources provided by the cloud and similar computing paradigms. The advantages of deploying these platforms on the cloud are the high availability of services, high computational resources to run complex position algorithms and offload the computational load from the user device to the cloud, and the relative ease of deployment. Additionally, some cloud providers offer services that facilitate the development and deployment of indoor positioning platforms.

In the case of the environment (GNSS-denied environments), many indoor positioning solutions have been tested in more than one scenario, including multistory buildings and crowded places. These scenarios presented the conditions necessary to test the robustness and flexibility of the proposed indoor positioning applications to adapt to multiple scenarios and conditions. The authors thus provided a comprehensive analysis of the positioning error, computational load, time response, and other parameters essential to analyse the performance of the proposed system.

Finally, we can observe that many of the proposed indoor positioning solutions have an application for web or mobile, which can be used in some of the most well-known operating systems (e.g., android and iOS). Additionally, a few works provided wearable devices such as a belt for positioning and navigation, demonstrating the scope of the proposed solutions.

2.5.5 Standardization (RQ6)

As previously mentioned, standardisation is important to enhance the interoperability, innovation, quality, and compatibility of IPS/ILS with other systems and technologies. Section 2.4.6 addressed the standardisation aspect of cloud-based indoor positioning platforms with a focus on 3 components: position technologies, maps, evaluation methods and software architecture.

Although standards for indoor maps are currently available for public use (e.g., IndoorGML and Indoor OpenStreetMaps), the authors in the papers analysed did not specify the kind of standard used in their implementations. However, in the case

of outdoor maps, some authors mentioned the used of OGC and BIM standards in their indoor positioning platforms.

Similarly, standards to test and evaluate indoor positioning solutions were raised infrequently or were not mentioned at all in the works analysed. For instance, the standard ISO/IEC 18305:2016 [114] was used in only one paper of 106 research articles, in spite of the fact that ISO/IEC 18305:2016 provides a set of guidelines to evaluate the performance of indoor positioning platforms. The use of this standard would allow us to evaluate and compare indoor positioning solutions under a common framework.

In the research papers analysed, the authors generally use 5 design patterns: monolithic, MSA, MVC, MVVM, and cloud-native patterns. MSA is one of the most frequently applied in enterprise commercial and open-source solutions. The second one is SOA, which is also divided into services where the application component provides certain services to other components via communication protocols such as those analysed in section 2.4.2.

2.5.6 Current challenges

Challenges related to software Software architecture is a pivotal pillar of a well-designed indoor positioning solution. It plays an important role in the robustness, scalability, and usability of the platform. Considering this, some authors followed architecture patterns such as MSA or SOA. Nevertheless, some characteristics of a robust indoor position solution remain unresolved in many applications, for instance, fault tolerance, reliability, and interoperability.

Challenges related to standardisation Previous sections presented some of the standards available to evaluate indoor positioning systems (ISO/IEC 18305:2016 [114]), to provide indoor and outdoor maps (e.g., OGC and IndoorGML), for indoor positioning technologies (e.g., IEEE 802.11x and Zigbee standard) and for software architecture. These standards should be considered during the design of an indoor positioning/localisation/navigation solution in order to enhance its quality and interoperability. Despite the efforts in standardising several aspects related to indoor positioning, most of the analysed papers do not use (or mention) any of the existing standards. This lack of implementation or even discussion of available standards in

the design of indoor positioning solutions may reduce their capacity to interoperate with other systems in a straightforward manner.

2.5.7 Future Trends

From Fig. 2.3, we can infer a clear trend of using the cloud or similar computing paradigms to deploy indoor positioning platforms. The trend is due to the numerous advantages provided by computing paradigms. Additionally, there is a growing trend of providing indoor positioning, localisation, navigation, and tracking systems as a service that can be used or integrated with other systems. This last trend can be observed in the integration of localisation services with smart parking applications, smart cities, industry 4.0, eHealth applications, among many others.

2.6 Summary

This chapter has presented an introduction to cloud-based indoor positioning solutions, along with a systematic review of current research studies in this field. This chapter has introduced different concepts, technologies, techniques, standards, and network protocols that are particularly relevant during the design, development, evaluation, and implementation of indoor positioning applications. The systematic review presented in this chapter analysed 106 research works selected through a rigorous procedure, following the PRISMA guidelines to offer a reproducible and replicable work.

Through this review, we have observed that combining techniques, methods, and positioning technologies help to improve accuracy in position estimation, achieving centimetre-level accuracy in the best case. Similarly, the use of computing paradigms provides various benefits to indoor positioning solutions, such as high computational and storage resources, availability, redundancy, and low latency, among others. These benefits have led to an increase in the use of computing paradigms in indoor positioning solutions. Additionally, we have observed a lack of the use of standards in the development of indoor positioning solutions. The use of standards is crucial to providing well-tested solutions that are easy to integrate with other solutions.

3 RESEARCH MATERIALS AND METHODS

This chapter provides a description of the datasets and the baseline algorithms used in this dissertation. The chapter is composed of the following parts:

- Introduction to fingerprinting technique and radio maps.
- Description of baseline algorithms.
- Summary.

3.1 Introduction

The evaluation of algorithms, methods and/or models with multiple and heterogeneous datasets provides a good understanding of the efficiency of the proposed algorithms. The research community has put a great deal of effort into ensuring that datasets are available, allowing researchers to test and train their solutions using the same data distribution and under the same conditions as previous researchers, allowing direct comparisons.

In light of the mentioned above, this dissertation uses multiple fingerprinting datasets to evaluate the performance of the algorithms and models proposed in each chapter, offering a comprehensive analysis of their advantages and drawbacks. The research community can thus replicate and reproduce the experiments carried out in this dissertation.

We used a basic but efficient baseline, namely k-NN, in order to compare and evaluate the robustness of the proposed algorithms. This baseline has previously been used in many research works [146], performing better than complex Neural Networks (NNs), in some cases.

The following paragraphs outline the fingerprinting technique, radio maps, and the k-NN baseline used in this dissertation.

3.2 Fingerprinting Technique

WLAN-based indoor positioning solutions have been widely used in commercial and open-source applications because of their flexibility and the availability of wireless signals in most scenarios [147, 148]. RSS fingerprinting techniques have been implemented in many applications, due to their simplicity and the fact that most users' devices already support some radiofrequency-based technologies such as BLE and Wi-Fi. However, this technique suffers from limitations caused by the degradation of signal quality in indoor environments, and the frequent updates of the radio map [149] caused, especially, in case of significant changes in the environment [150].

Generally, the fingerprinting technique is divided into two phases: offline and online. In the offline phase, we collect RSS measurements (in the case of radiofrequency-based technologies) at pre-established reference points to create a radio map. During the online phase, the position is estimated using the RSS measurements collected in unknown positions, a matching algorithm (e.g., *k*-NN) and the radio map (see Fig. 3.1).

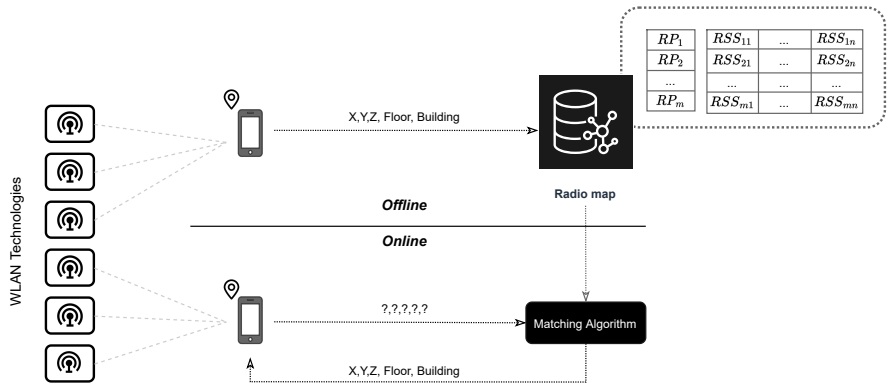


Figure 3.1 WLAN Fingerprinting schema

Due to the importance of the radio map in correctly estimating device position, multiple techniques have been developed by the research community in order to reduce the positioning error. For instance, feature extraction [151], data compression [152], data transformation [153] and data cleansing [154]. These techniques improve the characteristics of the dataset; reducing anomalies, outliers, and incomplete data.

Commonly, a radio map is composed by m fingerprints and n APs ($\Psi \in \mathcal{R}^{m \times n}$), its representation is given in Eq. (3.1).

$$\Psi = \begin{bmatrix} \psi_{11} & \psi_{12} & \cdots & \psi_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{m1} & \psi_{m2} & \cdots & \psi_{mn} \end{bmatrix}_{m \times n} \tag{3.1}$$

where, ψ_{ij} represents an RSS value in the i-th position ($i = 1, 2, 3, ..., m$) and transmitted by the n-AP ($j = 1, 2, 3, ..., n$). Each fingerprint in the radio map is linked with a reference point/position (e.g., x, y, z, floor and building).

3.3 Radio maps

This thesis uses 26 publicly available fingerprinting datasets (9 BLE and 17 Wi-Fi datasets), collected in heterogeneous environments and with different devices. In general, two data collection strategies or techniques have been used to collect the datasets: professional and crowdsourced. In professional datasets, one person or more collects the fingerprints at pre-established reference points in a systematic manner, e.g., the dataset provided in [148] was systematically collected over a period of 15 months. Crowdsourced datasets are collected by multiple users, and, in general, there are no well-defined reference points to collect the data. Therefore, each user collects fingerprints at different points which are not necessarily the same, e.g., in datasets provided by [155], the data collection involved multiple users taking measurements in different points of the building and using an Android application, namely "TUT Wi-Fi Positioning" to determine the user position.

It is also important to highlight that each dataset was collected under different conditions; for instance, UEXB1–3 were collected during holiday time to reduce the level of interference produced by people.

Table 3.1 demonstrates the diversity of the datasets used to test the proposed algorithms. This table contains the name of the dataset, year of publication, number of samples/fingerprints in the training dataset (Ψ_{TR}), number of samples/fingerprints in the test dataset (Ψ_{TE}), number of access points ($\mathscr{A}$), approximate dimension of the IPS deployment area (*Area*), the average number of fingerprints per reference point (ωfp), indoor positioning technology (*Tech.*), number of buildings ($\#b$), number of

floors ($\#f$), environment (*Env.*) –e.g., indoor (In), outdoor (Out), or both (In-Out), and, finally, the reference of each dataset.

Table 3.1 Datasets Parameters.

Dataset	Year	$\lvert\Psi_{TR}\rvert$	$\lvert\Psi_{TE}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\overline{\omega}_{fp}\rvert$	Area	Tech.	$\#b$	$\#f$	Env.	Ref.
LIB 1	2018	576	3120	174	12	308.4 m^2	Wi-Fi	1	2	In	[148]
LIB 2	2018	576	3120	197	12	308.4 m^2	Wi-Fi	1	2	In	[148]
MAN 1	2008	14300	460	28	110	221 m^2	Wi-Fi	1	1	In	[156]
MAN 2	2008	1300	460	28	10	221 m^2	Wi-Fi	1	1	In	[156]
TUT 1	2013	1476	490	309	1	9000 m^2	Wi-Fi	1	4	In	[157]
TUT 2	2013	584	176	354	1	14 000 m^2	Wi-Fi	1	3	In	[157]
TUT 3	2017	697	3951	992	1	8000 m^2	Wi-Fi	1	5	In	[155]
TUT 4	2017	3951	697	992	1	8000 m^2	Wi-Fi	1	5	In	[155]
TUT 5	2018	446	982	489	1	12 325 m^2	Wi-Fi	1	3	In	[158]
TUT 6	2020	3116	7269	652	1	12 100 m^2	Wi-Fi	1	4	In	[159]
TUT 7	2020	2787	6504	801	1	8400 m^2	Wi-Fi	1	3	In	[159]
UJI 1	2014	19861	1111	520	21	108 703 m^2	Wi-Fi	3	4-5	In	[160]
UJI 2	2017	20972	5179	520	11	108 703 m^2	Wi-Fi	3	4-5	In	[161]
UTS 1	2019	9108	388	589	6	44 000 m^2	Wi-Fi	1	16	In	[162]
TIE 1	2021	10633	613	613	1	5432 m^2	Wi-Fi	1	6	In	[163]
SAH 1	2021	9291	156	775	1	4184 m^2	Wi-Fi	1	3	In	[163]
OFIN	2021	1537	659	1380	13	380 m^2	Wi-Fi	–	–	Out	[164]
UJIB 1	2019	732	900	24	31	176.02 m^2	BLE	1	1	In	[165]
UJIB 2	2019	576	240	22	24	151.07 m^2	BLE	1	1	In	[165]
UEXB 1	2020	417	102	30	3	1000 m^2	BLE	1	3	In	[166]
UEXB 2	2020	552	138	30	3	1800 m^2	BLE	1	3	In	[166]
UEXB 3	2020	552	138	30	3	5800 m^2	BLE	2	3	In-Out	[166]
OFINB 1	2021	349	149	1044	11	97 m^2	BLE	–	–	Out	[164]
OFINB 2	2021	258	111	806	11	92 m^2	BLE	–	–	Out	[164]
OFINB 3	2021	396	169	2253	11	25 m^2	BLE	–	–	Out	[164]
OFINB 4	2021	371	159	490	11	172 m^2	BLE	–	–	Out	[164]

As can be observed in Table 3.1, the authors of each dataset provide their manner of dividing the datasets into training and testing sets. For example, in LIB1–2, the number of training samples is less than the testing samples ($\approx$ 15% and $\approx$ 85%, respectively), in TUT1 is $\approx$ 75% training and testing set $\approx$ 25%, and in UTS1 is $\approx$ 85% training and $\approx$ 5% in testing. We have thus used the same training and testing sets in order to ensure the reproducibility and replicability of the results provided within this thesis. However, in OFIN datasets, the samples were redistributed, given that the function used in the source code provided by the authors does not properly distribute the samples into training and testing. A proper distribution of fingerprints into training and testing sets should not have the same reference points and/or fingerprints

in both sets. If the samples or reference points are in both sets risks of overoptimistic position estimation.

In the case of TUT1–2 datasets, the authors averaged the RSS values in a grid size of 1 m, and in TUT5, the cell average was of 5 m [146].

Fig. 3.2 shows the 3D (left) and 2D (right) positions of the reference points in the training sets of 2 datasets, TUT3 and LIB1. These datasets are an example of heterogeneity in the datasets used in this dissertation. For instance, (a–b) shows the reference points in the training dataset of TAU3 (TUT3 used crowdsourced technique to collect the data) and (c–d) of LIB1 datasets (LIB1 is a professional dataset collected systematically on the floor 3 and 5 at UJI's library).

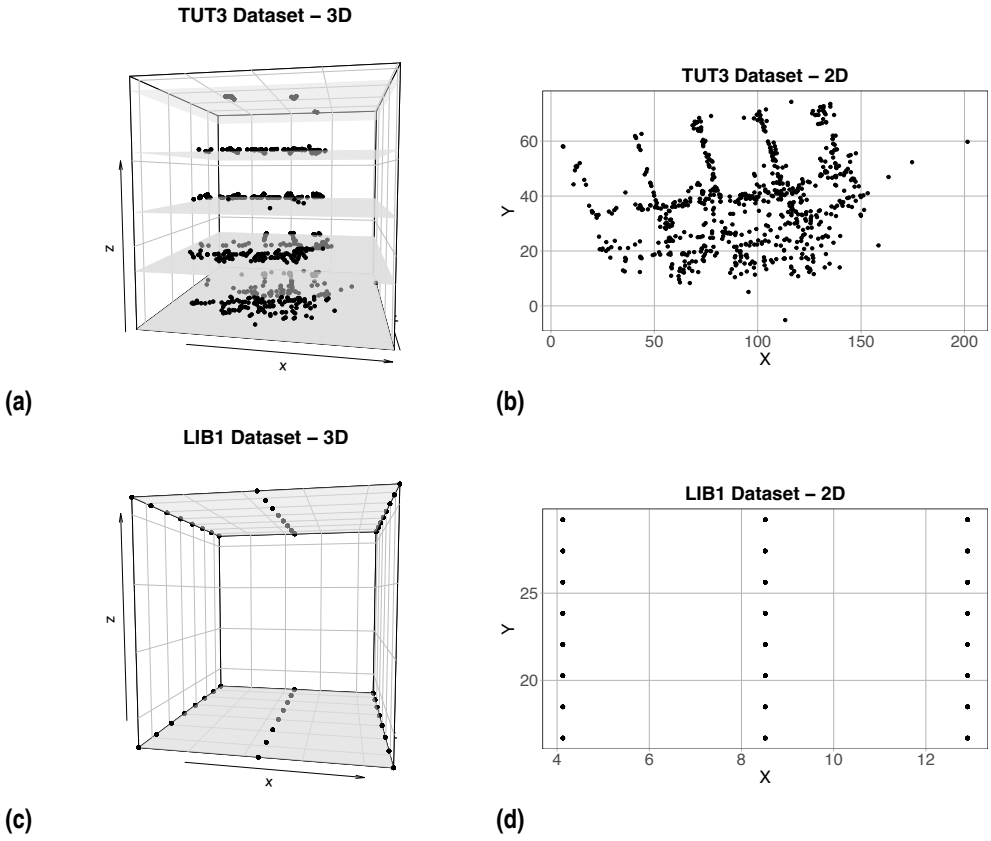


Figure 3.2 3D (left) and 2D (right) representation of the reference points in 3 datasets. (a–b) TUT3 and (e–f) LIB1 datasets.

Table 3.2 exhibits a summary of the statistical description of the datasets (training and testing sets) described above. This table shows the percentage of null values in

the dataset (%γ) (–i.e., in most of these datasets 100 represents a non detected value), mean, standard deviation (σ), min RSS value, maximum RSS values, 25 percentile (P_{25}), 50 percentile (P_{50}), 75 percentile (P_{75}), and 95 percentile (P_{95}). As can be observed from this table, there is a high percentage of null values in the datasets, e.g., in LIB1, 88.53% of the total RSS values are null, 49.76% in MAN2 and 97.24% in OFINB3. LIB1–2 and UEXB1–3 are the datasets with a lower number of null values (less than 45%).

Table 3.2 Statistical description of datasets.

Dataset	%γ	MEAN	σ	MIN	MAX	P_{25}	P_{50}	P_{75}	P_{95}
LIB1	88.53	-80.97	9.75	-98.00	-38.00	-88.00	-84.00	-77.00	-60.00
LIB2	90.23	-80.09	9.79	-99.00	-35.00	-87.00	-83.00	-75.00	-59.00
MAN1	63.06	-75.02	11.49	-100.00	-41.00	-85.00	-76.00	-67.00	-55.00
MAN2	49.76	-77.53	11.50	-95.50	-44.20	-88.00	-79.00	-68.90	-57.11
TUT1	89.98	-76.01	12.46	-100.00	-20.00	-86.00	-79.00	-69.00	-50.63
TUT2	88.18	-80.85	11.97	-100.00	-26.00	-89.00	-85.00	-75.00	-56.00
TUT3	95.01	-76.99	10.59	-102.00	-14.00	-84.00	-79.00	-72.00	-55.00
TUT4	95.01	-76.99	10.59	-102.00	-14.00	-84.00	-79.00	-72.00	-55.00
TUT5	92.39	-72.16	10.63	-93.00	-22.00	-80.00	-74.50	-66.61	-51.00
TUT6	94.67	-72.99	9.87	-94.00	-27.00	-80.00	-75.00	-67.00	-54.00
TUT7	96.62	-73.34	10.11	-94.00	-27.00	-81.00	-76.00	-67.00	-54.00
UJI1	96.54	-78.46	12.59	-104.00	0.00	-88.00	-82.00	-72.00	-54.00
UJI2	96.61	-77.28	12.78	-104.00	0.00	-87.00	-80.00	-70.00	-53.00
UTS1	93.99	-78.24	8.57	-96.00	-37.00	-85.00	-80.00	-73.00	-62.00
TIE01	93.10	-74.57	9.68	-94.00	-27.00	-82.00	-77.00	-69.00	-55.00
SAH1	95.76	-74.91	9.92	-94.00	-27.00	-82.00	-78.00	-69.00	-55.00
OFIN1	95.79	-82.91	8.25	-99.00	-38.00	-89.00	-85.00	-78.00	-67.00
UJIB1	30.03	-79.29	11.98	-103.00	-33.00	-89.00	-81.00	-72.00	-57.00
UJIB2	44.10	-74.45	7.86	-94.00	-44.00	-80.00	-75.00	-69.00	-60.00
UEXB1	40.37	-94.73	11.29	-109.00	-62.34	-109.00	-92.07	-85.63	-78.33
UEXB2	42.02	-95.70	11.06	-109.00	-62.34	-109.00	-93.73	-86.64	-79.13
UEXB3	35.12	-102.38	10.12	-109.00	-53.50	-109.00	-109.00	-94.76	-81.44
OFINB1	96.46	-92.15	7.98	-112.00	-71.00	-100.00	-90.00	-86.00	-81.00
OFINB2	95.83	-89.53	9.70	-113.00	-53.00	-97.00	-89.00	-84.00	-72.00
OFINB3	97.23	-92.25	7.65	-111.00	-69.00	-100.00	-90.00	-86.00	-81.00
OFINB4	96.08	-90.57	8.43	-113.00	-59.00	-97.00	-90.00	-85.00	-77.00

As can be shown from table 3.2, the statistical properties are diverse among the datasets demonstrating their heterogeneity. Only 2 datasets, TUT3 and TUT4, show similar statistical characteristics, the same mean, standard deviation (σ), minimum and maximum RSS values, and same percentiles. In UEXB1–3 can be observed that the maximum RSS values do not correspond to integer values, and it is because the

authors used the average of all signals transmitted by the same beacon. Those values were used to build the radio maps.

3.4 Baseline Algorithm

k-NN is the main core IPS used as the baseline to compare the performance of the proposed algorithms in terms of positioning error, floor and building hit rate, and processing time. *k*-NN is a non-parametric unsupervised ML algorithm used to solve regression and classification problems. This algorithm is also known as a proximity algorithm due to the fact that it is based on the assumption that similar points can be localized close to one another. Thus, the aim of *k*-NN is to identify the nearest neighbours to a certain point. Basically, this algorithm requires 2 input parameters, k and the distance metric. k determines the number of neighbours to be taken into account in the classification and/or regression process, and the distance metrics allow us to determine the distance between the data points.

In general, we use the *simple configuration* of *k*-NN described in [146]. It includes $k = 1$, Manhattan distance (see Eq. 3.2) and positive data representation (see Eq. 3.3) [167].

$$D_{Manhattan}(O, \mathcal{P}) = \sum_{i=1}^{n} |O_i - \mathcal{P}_i| \tag{3.2}$$

where O and $\mathcal{P}$ are two vectors, and n represents the number of values in the vector. In the case of fingerprinting, each vector represents a fingerprint, and m corresponds to the number of APs in the dataset.

$$Positive_i(\Psi) = \begin{cases} \psi_{ij} - min(\Psi), & \text{If } \psi_{ij} \text{ is not a non detected value,} \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

where, $min(\Psi)$ represents the minimum RSS value in the radio map.

3.5 Experiments and Results

The experiments were performed using a computer with the following characteristics: Intel® Core™ i7-8700T @ 2.40 GHz and 16 GB of RAM, the operating system is Fedora Linux 32, and the software used is Python 3.9. The library used is Scikit-learn

(Sklearn). In order to compare the performance of the processes algorithms, we use the following parameters: mean 3D positioning error (ε_{3D}), mean 2D positioning error (ε_{2D}), prediction time (δ_{TE}), building hit rate (ζ_b) and floor hit rate (ζ_f). Normalised values are denoted with the tilde symbol ($< \tilde{metric} >$) over each metric.

Table 3.3 Results of the baseline method, k-NN with simple configuration [146] ($k = 1$, Manhattan distance and Positive data representation)

	Baseline k-NN (Simple Configuration)					Baseline — Normalised values				
Dataset	ζ_b [%]	ζ_f [%]	ε_{2D} [m]	ε_{3D} [m]	δ_{TE} [s]	$\tilde{\zeta}_b$ [–]	$\tilde{\zeta}_f$ [–]	$\tilde{\varepsilon}_{2D}$ [–]	$\tilde{\varepsilon}_{3D}$ [–]	$\tilde{\delta}_{TE}$ [–]
LIB1	–	99.840	3.035	3.043	0.531	–	1	1	1	1
LIB2	–	97.724	4.031	4.197	0.608	–	1	1	1	1
MAN1	–	100.000	2.877	2.877	0.376	–	1	1	1	1
MAN2	–	100.000	2.467	2.467	0.034	–	1	1	1	1
TUT1	–	90.000	8.623	9.601	0.401	–	1	1	1	1
TUT2	–	72.727	11.218	12.893	0.073	–	1	1	1	1
TUT3	–	91.622	8.926	9.594	5.035	–	1	1	1	1
TUT4	–	95.265	6.152	6.406	5.424	–	1	1	1	1
TUT5	–	88.391	6.387	6.924	0.393	–	1	1	1	1
TUT6	–	99.986	1.959	1.959	27.612	–	1	1	1	1
TUT7	–	99.185	2.110	2.351	27.429	–	1	1	1	1
UJI1	99.190	87.759	7.718	10.829	21.674	1	1	1	1	1
UJI2	100.000	85.345	7.742	8.052	108.441	1	1	1	1	1
UTS1	–	92.784	7.769	8.757	4.076	–	1	1	1	1
TIE1	–	60.000	4.248	6.548	0.669	–	1	1	1	1
SAH1	–	46.795	8.110	9.048	2.204	–	1	1	1	1
OFIN	–	–	2.386	2.386	2.646	–	1	1	1	1
UJIB1	–	–	3.076	3.076	0.044	–	1	1	1	1
UJIB2	–	–	4.898	4.898	0.008	–	–	1	1	1
UEXB1	–	90.196	3.493	3.708	0.005	–	1	1	1	1
UEXB2	–	94.203	4.396	4.649	0.006	–	1	1	1	1
UEXB3	100.000	76.667	6.617	7.104	0.002	1	1	1	1	1
OFINB1	–	–	3.984	3.984	0.103	–	–	1	1	1
OFINB2	–	–	3.181	3.181	0.046	–	–	1	1	1
OFINB3	–	–	1.696	1.696	0.299	–	–	1	1	1
OFINB4	–	–	3.883	3.883	0.061	–	-	1	1	1
Avg.	–	–	–	–	–	1	1	1	1	1

3.6 Summary

The following paragraph summarizes the material and methods used throughout this thesis.

- The algorithms and models developed in this dissertation are evaluated with 26 publicly available datasets, including BLE and Wi-Fi datasets. These datasets were collected in different environments, such as universities and libraries. Additionally, to test the proposed algorithms' performance, we use k-NN with *simple configuration* as the baseline.

4 DATA OPTIMISATION

Data optimisation determines to a large extent the accuracy of indoor positioning applications. The examination of the techniques used to enhance the quality of indoor positioning data is therefore of paramount relevance to providing highly accurate solutions. This chapter presents a data cleansing algorithm and a data augmentation model based on ML models. The outline of this chapter is as follows:

- A general introduction of data optimisation in indoor positioning.
- Description of the data cleansing algorithm proposed, including experiments and results.
- Introduction to a data augmentation model based on GAN, including experiments and results.
- Discussion of the results obtained.
- A summary of the key findings.

4.1 Introduction

In light of the exponential development and growth of wearable and IoT devices using positioning and localisation services [168], data preprocessing and data analysis play an important role in providing high-quality indoor positioning applications to the end-user. This massive device connectivity generates large amounts of data that should be preprocessed in order to extract the most relevant information. Generally, the data collected is likely to contain incomplete data, irrelevant observations, and duplicate records, all of which can affect position estimation [169, 170]. The data collected should therefore be preprocessed prior to applying any positioning algorithm or technique in order to enhance position accuracy.

In the case of RF-based indoor positioning solutions, radio frequency signals are often disturbed by factors such as multipath effects, NLOS, and occlusion, among

others [171, 172]. If not filtered out, removed, or otherwise preprocessed, these adverse factors produce undesirable fluctuations in the signal strength that will likely lead to a high error rate in position estimation. Considering that the fingerprinting technique is often used (but is not limited to) with RF-based indoor positioning technologies, these undesirable fluctuations may be present in the radio map built in the offline phase of fingerprinting.

With the goal of removing some noisy fingerprints, we propose a new data cleansing model for indoor positioning fingerprinting datasets. The model enables indoor positioning platforms to obtain meaningful data and estimate the user position in an efficient way using consistent quality data.

Once noisy fingerprints are removed from the datasets, the dataset size may be significantly reduced. This reduced size may be a negative factor when training ML-based positioning models, given that most ML algorithms require large amounts of high-quality data to be trained. On this basis, we suggest a data augmentation model based on GAN to generate new realistic artificial fingerprints and augment the radio map. This new synthetic data contains multiple characteristics of real fingerprints, making them almost indistinguishable from real data. Additionally, the data augmentation model reduces the time-consuming process of manual data collection.

4.2 Data Cleansing

There are various processes that form part of data cleansing, such as managing missing data, discarding duplicate records, and removing corrupted data. Here we concentrated on enhancing the quality of the data by removing outliers.

Some authors have already proposed methods to enhance the quality of radio maps. Lin, Jiang, Yus, Bouloukakis, Chio, Mehrotra, and Venkatasubramanian [173], for example, proposed a new three-step method: firstly, unlabelled data is completed using additional measurements. Secondly, the coarse localisation is carried out, and, finally, the fine location is estimated using a probabilistic model. However, the authors failed to manage outliers in the dataset. Similarly, Khalajmehrabadi, Gatsis, and Akopian [170] completed missed data by using information obtained from the neighbouring APs. Additionally, the authors also provided an outlier detection algorithm, which was able to detect anomalies produced by, for example, data poisoning attacks. Some

authors have suggested data cleansing models for crowdsourced Wi-Fi fingerprinting datasets to fill missed RSS values [154].

Unlike the mentioned above, we propose a data cleansing model which filters data based on the correlation between fingerprints. The aim is to remove irrelevant fingerprints, improve the quality of the radio map, and reduce the positioning error. The proposed algorithm is divided into five parts as described below:

(i) **Number of valid RSS:** This step is devoted to determining the maximum number of RSS values ($\aleph$) in the dataset. Thus, only the valid RSS measurements are selected, and the non-detected values (γ) are ruled out.

$$\begin{aligned} \nu_i &= len(\Psi_i) | \forall \psi_{ij} \neq \gamma \\ \aleph &= max(\nu) \end{aligned} \tag{4.1}$$

where, $\nu \in \mathcal{R}^m$ is a vector with the number of valid RSS values of the i-th fingerprint, $i = 1, 2, 3, ..., m$ and $j = 1, 2, 3, ..., n$.

(ii) **Sort and replace RSS values:** In the second step, the RSS values are sorted in descending order and then replaced by their corresponding AP identifier.

$$\begin{aligned} \mathcal{X}_i &= sort(\Psi_i, descending) \\ x_{ij} &\leftarrow AP_j \end{aligned} \tag{4.2}$$

where, $\mathcal{X}$ is a matrix consisting of AP identifiers instead of the RSS values. The RSS value in the i-th position and transmitted by the j-th AP is replaced by the j-th AP identifier. Here, only the first $\aleph$-columns are selected for analysis.

$$\mathcal{X} = \begin{bmatrix} x_{11}, x_{12}, & \dots & x_{1\aleph} \\ x_{21}, x_{22}, & \dots & x_{2\aleph} \\ \vdots & \ddots & \vdots \\ x_{m1}, x_{m2}, & \dots & x_{m\aleph} \end{bmatrix}_{m \times \aleph}$$

(iii) **Match percentage between samples:** This stage is devoted to computing the correlation between samples (i.e., it corresponds to the match percentage between samples.). We first set a threshold (ρ), which represents the minimal

correlation between two samples. Once the threshold is defined, the correlation among all samples in $\mathcal{X}$ is computed under the following conditions:

$$\mathfrak{I}_i = \begin{cases} \jmath_i, & \text{if } \mathfrak{I}_i < \rho \text{ or } \jmath_i > \mathfrak{I}_i \text{ or } \mathcal{X}_i = \mathcal{X}_l \\ \frac{len(\mathcal{X}_i \cap \mathcal{X}_l)}{\aleph} * 100, & \text{otherwise} \end{cases} \tag{4.3}$$

where $\jmath_i$ represents the previous match percentage between the i-th and the l-th sample, whereas ($\mathfrak{I}_i$) is the current match percentage between them

(iv) **Removing fingerprints:** Finally, all samples with zero match percentage are removed from the original radio map (training dataset), given that they do not fulfil the conditions established before.

$$\Psi_i \in \Psi_c \forall 1 \leq i \leq m | \mathfrak{I}_i \neq 0 \tag{4.4}$$

All steps described in the previous paragraphs are summarised in Algorithm 4.1. The input parameters are: the training dataset (Ψ_{TR}), the non-detected value (γ), and threshold ρ. The output is the cleansed dataset (Ψ_c).

4.2.1 Experiments and Results

4.2.1.1 Experiment setup

In order to carry out the experiments, we used 26 public datasets introduced in Chapter 3. Similarly, the hardware, software and baseline used to perform the experiments were described in Chapter 3.

The parameters used in the data cleansing algorithm are: non-detected value equal to +100, the threshold for each dataset is computed as the correlation between samples described in the previous section, and the maximum number of RSS values ($\aleph$) in the dataset. Moreover, we use positive data representation in all datasets [167].

Additionally, we use k-NN as the core IPS to test positioning accuracy obtained with the radio maps before and after the data cleansing. The benchmark is k-NN using the *simple configuration* and with the original datasets (see Chapter 3). The parameters used to compare the results are: mean 3D positioning error (ε_{3D}), mean 2D positioning error (ε_{2D}), prediction time (δ_{TE}), training time (δ_{TR}), building hit

Algorithm 4.1 Data cleansing algorithm. Reproduced with the permission from [174].

Input : Ψ_{TR}, γ, ρ
Output : Ψ_c
$\Psi \leftarrow \Psi_{TR}$ $\Psi_c \leftarrow \Psi_{TR}$
/* 1. Number of valid RSS values */
$\nu_i = len(\Psi_i) | \forall \psi_{ij} \neq \gamma$
/* Maximum number of valid RSS values. */
$\aleph = max(\nu)$
/* 2. Sort and replace RSS values. */
$\chi_i = sort(\Psi_i, descending)$
$x_{ij} \leftarrow AP_j$
$\chi \in \mathcal{R}^{m \times \aleph}$
/* 3. Match percentage between samples */
for *i=1 to m* **do**
  for *l=1 to m* **do**
    $\Im'_i = \frac{len(\chi_i \cap \chi_l)}{\aleph} * 100$
    if $\chi_i \neq \chi_l$ & $\jmath_i < \Im'_i$ & $\Im'_i > \rho$ **then**
      $\Im_i = \Im'_i$
    end
  end
  $\jmath_i \leftarrow \Im_i$
end
/* 4. Remove samples with zero match percentage */
for *i=1 to m* **do**
  if $\Im_i == 0$ **then**
    DEL (Ψ_{ci})
  end
end

rate (ζ_b) and floor hit rate (ζ_f). Additionally, this method has been compared with two widely-used data cleansing methods.

4.2.1.2 Results

Table 4.1 shows the results obtained before and after applying the proposed data cleansing algorithm. The results obtained with the cleansed datasets were normalised to better visualise the variation in the training dataset ($\mathcal{T}_{TR}$), building hit rate (ζ_b), floor hit rate (ζ_f), mean 3D positioning error (ε_{3D}), mean 2D positioning error (ε_{2D}) and prediction time (δ_{TE}).

Table 4.1 demonstrates that the proposed algorithm was able to remove noisy fingerprints in 22 datasets without negatively affecting the positioning estimation. The data cleansing algorithm could not, however, find any irrelevant fingerprints in four of the datasets (datasets denoted by "✕" symbol) due to the complexity and nature of the radio maps. In some cases, the number of fingerprints removed from a dataset

Table 4.1 1-NN using the cleansed dataset.

		Data cleansing + 1-NN					
Dataset	ρ [%]	$\bar{\Psi}_{TR}$ [–]	$\bar{\zeta}_b$ [–]	$\bar{\zeta}_f$ [–]	$\bar{\varepsilon}_{2D}$ [–]	$\bar{\varepsilon}_{3D}$ [–]	$\bar{\delta}_{TE}$ [–]
LIB1	33	0.844	–	1	0.998	0.998	0.843
LIB2	40	0.589	–	1.02	0.888	0.858	0.589
MAN1	34	0.961	–	1	0.981	0.981	0.914
MAN2	45	0.952	–	1	0.989	0.989	0.927
TUT1	35	0.674	–	1.014	0.903	0.873	0.769
TUT2	30	0.664	–	1.039	0.964	0.939	0.66
TUT3	2	0.983	–	1.003	0.99	0.978	0.984
TUT4	1	0.996	–	1	0.998	0.999	0.992
TUT5	21	0.798	–	1.001	0.969	0.956	0.809
TUT6	2	0.997	–	1	0.99	0.99	0.997
TUT7	×	×	×	×	×	×	×
UJI1	20	0.877	1.008	1.03	1	0.828	0.877
UJI2	20	0.873	1	1.022	0.978	0.96	0.876
UTS1	20	0.923	–	1.008	1.002	0.962	0.888
TIE1	50	0.366	–	1.2	0.927	0.977	0.357
SAH1	25	0.829	–	1.123	1.1	0.957	0.812
OFIN	20	0.926	–	1	0.995	0.995	0.935
UJIB1	×	×	×	×	×	×	×
UJIB2	30	0.741	–	–	0.999	0.999	0.729
UEXB1	×	×	×	×	×	×	×
UEXB2	16	0.924	–	1.008	0.982	0.98	0.934
UEXB3	×	×	×	×	×	×	×
OFINB1	30	0.232	–	–	0.836	0.836	0.253
OFINB2	12	0.883	–	–	0.9	0.9	0.855
OFINB3	40	0.398	–	–	0.927	0.927	0.387
OFINB4	10	0.965	–	–	0.921	0.921	0.857
Avg.	–	0.791	1.004	1.028	0.965	0.946	0.784

was less than 1%, such as in the case of TUT4 and TUT6. Conversely, some datasets saw a reduction of more than 60% in some cases (OFINB1, OFINB3 and TIE1). In the 22 datasets where the algorithm removed outliers, there was a slight improvement in positioning accuracy. For instance, there was an average improvement of 3.5% and 5.4% in the 2D and 3D positioning errors, respectively. The floor hit rate improved by almost 3% on average, and the building hit rate remained similar.

Figure 4.1 shows the data distribution of LIB2 (a), MAN1 (b), UJI1 (c) datasets before and after the data cleansing. The solid line represents the data distribution prior to data cleansing, showing a major concentration in the non-detected value. The dotted line shows the data distribution in the cleansed dataset. As shown, the

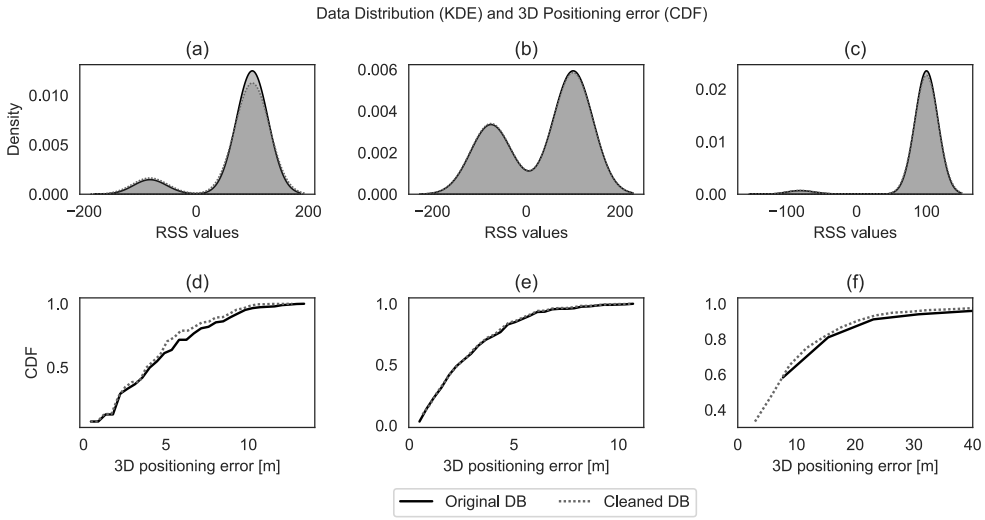


Figure 4.1 Representation of the data distribution using Kernel Density Estimation (KDE) and the distribution of the mean 3D error using the Cumulative Distribution Function (CDF).

data distribution is slightly affected, decreasing the number of non-detected values after using the data cleansing algorithm. Additionally, Figure 4.1 (d-f) (LIB2, MAN1, UJI1 datasets, respectively) shows the 3D positioning performance when the original dataset is used to estimate the user position and when the dataset is pre-processed using the proposed data cleansing algorithm. After applying the cleansing algorithm, there is a slight improvement in the positioning performance.

The proposed cleansing algorithm was also compared with two methods widely used for outlier detection. In the first method (based on the standard deviation of the data), we compute the mean ($\overline{\Psi}$) and the standard deviation (σ_Ψ) of the dataset, and then we define the threshold ($\rho_{m1} = \sigma_\Psi * \iota$, where ι is a multiplication factor) used to compute the minimum ($min\_val = \overline{\Psi} - \rho_{m1}$) and the maximum value ($max\_val = \overline{\Psi} + \rho_{m1}$) required to determine if a fingerprint is an outlier or not. The outlier is thus the fingerprint which mean value ($\overline{\Psi}_i, i = 1, 2, 3, ..., m$) is less than the min_val or greater than max_val. To carry out this experiment, we used ι equal to 0.12.

The second method is Isolation Forest, which is an unsupervised model used to detect anomalies/outliers in different fields, e.g., network intrusion and fraudulent bank transactions. This ML model is built using the decision tree algorithm, and its design is based on the assertion that "outliers are few and different". Based on this

Table 4.2 Comparison of Data Cleansing: Standard Deviation and Vs. Isolation Forest.

	STD Method						Isolation Forest Method					
Dataset	$\bar{\Psi}_{TR}$ [–]	$\bar{\zeta}_b$ [–]	$\bar{\zeta}_f$ [–]	$\bar{\varepsilon}_{2D}$ [–]	$\bar{\varepsilon}_{3D}$ [–]	$\bar{\delta}_{TE}$ [–]	$\bar{\Psi}_{TR}$ [–]	$\bar{\zeta}_b$ [–]	$\bar{\zeta}_f$ [–]	$\bar{\varepsilon}_{2D}$ [–]	$\bar{\varepsilon}_{3D}$ [–]	$\bar{\delta}_{TE}$ [–]
LIB1	0.852	–	0.998	1.057	1.058	0.869	0.939	–	0.995	1.020	1.027	0.959
LIB2	0.847	–	1.017	0.983	0.954	0.853	0.991	–	0.993	1.068	1.077	1.037
MAN1	0.690	–	–	0.982	0.982	0.654	0.893	–	–	0.999	0.999	0.850
MAN2	0.746	–	–	0.925	0.925	0.829	0.896	–	–	0.993	0.993	0.960
TUT1	0.719	–	0.989	0.850	0.851	0.713	×	×	×	×	×	×
TUT2	0.765	–	1.141	1.036	0.914	0.698	0.986	–	1.000	1.000	1.000	0.893
TUT3	1.000	–	0.941	1.040	1.112	0.619	0.986	–	0.992	0.992	1.008	0.997
TUT4	0.582	–	0.910	1.326	1.580	0.561	0.988	–	0.995	1.005	1.011	0.966
TUT5	0.789	–	0.960	1.053	1.057	0.800	×	×	×	×	×	×
TUT6	0.686	–	0.990	2.349	2.385	0.692	×	×	×	×	×	×
TUT7	0.859	–	0.992	1.279	1.444	0.855	×	×	×	×	×	×
UJI1	0.897	1.008	1.012	1.018	0.856	0.897	×	×	×	×	×	×
UJI2	0.897	1.000	1.000	0.999	0.999	0.892	1.000	1.000	1.000	1.000	1.000	1.007
UTS1	0.664	–	1.006	1.014	0.958	0.642	×	×	×	×	×	×
TIE1	0.637	–	0.833	1.405	1.139	0.612	×	×	×	×	×	×
SAH1	0.868	–	1.151	1.271	1.076	0.850	×	×	×	×	×	×
OFIN	0.994	–	–	1.000	1.000	1.005	×	×	×	×	×	×
UJIB1	0.320	–	–	1.033	1.033	0.320	0.698	–	–	1.026	1.026	0.541
UJIB2	0.130	–	–	1.150	1.150	0.250	0.500	–	–	1.043	1.043	0.512
UEXB1	0.487	–	0.837	1.098	1.255	0.451	0.854	–	0.978	1.116	1.133	0.546
UEXB2	0.270	–	0.893	1.485	1.517	0.401	0.848	–	1.015	1.084	1.056	0.754
UEXB3	0.354	1.000	0.804	1.217	1.436	0.748	0.913	1.000	0.978	1.039	1.022	0.901
OFINB1	0.307	–	–	0.798	0.798	0.308	0.905	–	–	1.000	1.000	0.931
OFINB2	0.595	–	–	1.047	1.047	0.590	0.992	–	–	1.000	1.000	0.969
OFINB3	0.807	–	–	1.002	1.002	0.743	×	×	×	×	×	×
OFINB4	0.581	–	–	0.951	0.951	0.522	×	×	×	×	×	×
Avg.	0.667	1.003	0.985	1.130	1.134	0.668	0.893	1.000	0.994	1.026	1.026	0.855

premise, Isolation Forest was implemented to detect outliers in radiofrequency-based fingerprinting datasets. As with the previous experiments, we used Python 3.9 and Sklearn library to perform this experiment and analysis. The hyperparameters used are the number of estimators equal to 1000 and a random state equal to 1102.

Table 4.2 exhibits the results of the application of the standard deviation (STD) method and Isolation Forest to detect outliers in 26 fingerprinting datasets. The first method, described in previous paragraphs, demonstrates good performance in detecting anomalies in some of the datasets (MAN1–2, TUT1, UJI2, OFINB1, OFINB4). In all other cases, the floor hit rate and/or 3D and 2D positioning error increased in comparison with the baseline. In the case of the Isolation Forest, the

algorithm was not able to find outliers in 11 datasets. It was able to remove outliers in only six of the datasets without negatively affecting the positioning accuracy (MAN1–2, TUT2, UJI2, OFINB1–2). As shown in Table 4.1, the proposed method outperforms some of the current leading methods used to detect outliers.

4.3 Data Augmentation

Data augmentation is one of the primary techniques used to generate new data artificially. This technique is helpful in increasing the dataset size, avoiding overfitting, and improving ML model's performance [175]. This technique commonly uses existing data to generate new modified data, thus enriching the dataset. Given the advantages of data augmentation, this technique has been covered in numerous research papers in the field, mostly with the fingerprinting technique [176, 177].

Generally, building an accurate radio map is laborious and time-consuming, sometimes taking weeks or even months to complete. Despite this, it is important that the task is repeated from time-to-time (e.g., every three months) in order to keep the radio map updated. In order to reduce the manual labour and the time required to build a radio map, deep learning and GAN networks are used to generate synthetic data, which is indistinguishable from real data [178, 176]. Although GAN network is relatively new in the field of ML, it has already been used for indoor positioning.

In order to reduce the amount of manual labour involved in updating and maintaining indoor positioning radio maps, we propose the combinations of Conditional Generative Adversarial Network (cGAN) and an algorithm to select the most realistic synthetic fingerprints. This proposed combination is tested with 11 datasets to determine its robustness and performance with heterogeneous datasets. Unlike previous research works using GAN in indoor positioning, we propose a complete framework for data augmentation, indoor positioning, and a new algorithm to select the most suitable synthetic fingerprints to enrich the dataset (hereinafter namely SURIMI). The following paragraphs describe in detail the first two components of the SURIMI framework, the data augmentation model and the algorithm to select realistic synthetic fingerprints.

4.3.1 Generative Adversarial Network (GAN)

Generative adversarial networks are comprised of two components: a discriminator and a generator (see Fig. 4.2). The generator is devoted to learning the data distribution of the input data in order to generate realistic synthetic data. The discriminator is devoted to determining if the input data is real or synthetic. Moreover, this network is trained in an "adversarial" manner [179], meaning that the performance of the discriminator increases at the cost of decreasing the capability of the generator and vice-versa. The GAN is mathematically represented as follows:

$$\min_{G} \max_{D} V(D, G) = \mathbb{E}_{\Psi \sim p_{data}(\Psi)} [\log D(\Psi|y)] + \mathbb{E}_{z \sim p_z(z)} [\log 1 - D(G(\Psi|y))] \quad (4.5)$$

where, $V(D, G)$ represents the value function, G is the generator, and D is the discriminator of the GAN network. Ψ represents the fingerprints, y the classes in the dataset, and z the noise values. $p_{data}(\Psi)$ means a probability distribution over the data Ψ and $p_z(z)$ is the probability distribution of the noise.

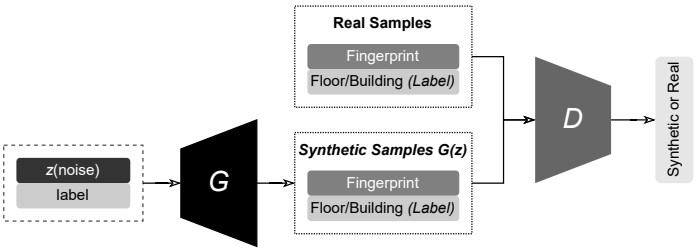


Figure 4.2 cGAN for WLAN fingerprints generation. Reproduced from [180].

Fig. 4.2 shows a general diagram of a cGAN for fingerprint data generation. The input of the generator is the noise (z) with dimensionality n and random labels, or all labels equal to zero. Once the noise passes through the generator, new synthetic data is generated. Finally, the discriminator tries to determine if the input is real or not. As discussed, the cGAN is trained in an adversarial manner; thus, there is a point where the discriminator cannot distinguish between synthetic and real fingerprints.

Previous papers have analysed the use of GAN for data augmentation in relation to indoor positioning data. For instance, Li, Qu, Liu, Zhou, Sun, Sigg, and Li [176] offered a new data augmentation model namely Amplitude-Feature Deep Convolutional GAN (AF-DCGAN). In this case, the authors converted the Channel State Information (CSI) radio map into an amplitude feature map and then augmented

it using the (AF-DCGAN) model. As a result, the authors increased the size of the training dataset and improved the positioning accuracy without the need for further data collection. In the same vein, Njima, Chafii, Chorti, Shubair, and Poor [178] combined a GAN and DNN to generate new synthetic RSS values and improve the positioning accuracy. The authors use the RSS values instead of CSI information to enhance the training set. Additionally, [178] established two criteria for selecting the most relevant synthetic fingerprints, thus reducing the positioning error by more than 15% in comparison with the benchmark.

Unlike previous studies, we used a cGAN to generate synthetic fingerprints for multi-building and multi-floor environments. The algorithm proposed for selecting the most suitable artificial fingerprints is based on the physical distance between the real fingerprints (point in the training set) and the generated fingerprints.

Our proposed cGAN is composed of thirteen layers in the generator model (see Fig. 4.3).

- One embedding layer, which is used to convert categorical inputs into continuous variables (e.g., floors: [[0],[2]] will be converted to [[0.15, 0.14], [0.63, -0.23]]). Vector representations thus allow the neuronal network to compute angles and projections, which are some of the primary operations in many machine learning algorithms.
- Two dense layers are used to connect their preceding layer deeply. Additionally, these layers empower the network's ability to learn the patterns of radio maps.
- Two reshape layers which are in charge for changing the shape of the input layer. If the neural network does not receive the data with the expected shape, it will not be able to process the data or may lead to incorrect results.
- Four LeakyReLu activation function layers, which are used to prevent the zero output from the neurons (dying state), allowing negative values to pass through in a controlled manner.
- Three Conv1DTranspose layers are used to upsample the input data to have a more detailed and large output.
- A Conv1D is used to downsampling the input data.

Unlike the generator, the discriminator (see Fig. 4.4) is comprised of sixteen layers described below.

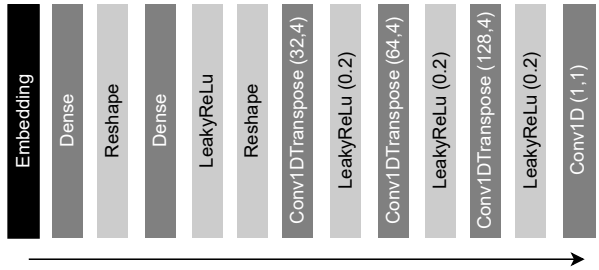


Figure 4.3 cGAN synthetic fingerprints generation - Generator model. Reproduced from [180].

- An embedding layer is used to convert categorical inputs into continuous variables as in the generator.
- Two dense layers, the first one is used to connect the preceding layer deeply and the last dense layer to compute the prediction of the overall network.
- A reshape layer is used to change the shape of the input data in the same way as in the generator.
- Five convolutional layers are employed to downsampling the input data and learn or highlight some features that are important for the classification of fingerprints.
- Five LeakyReLu activation function layers, which are used to prevent the zero output from the neurons as in the generator.
- A flatten layer is used to convert the feature map into a one-dimensional vector and pass it to the final layer or network, which is in charge of calculating the output of the overall network.
- A dropout layer to prevent overfitting and improve the generalization of the proposed model.

In order to train the data augmentation model, we used three methods: the first one divides the dataset into buildings, the second divides the dataset into floors and the third uses the whole dataset.

4.3.2 Synthetic Fingerprints Selection

As the generator only learns from fingerprints within the radio map, some synthetic fingerprints can appear in unrealistic places. To avoid this problem, we have proposed

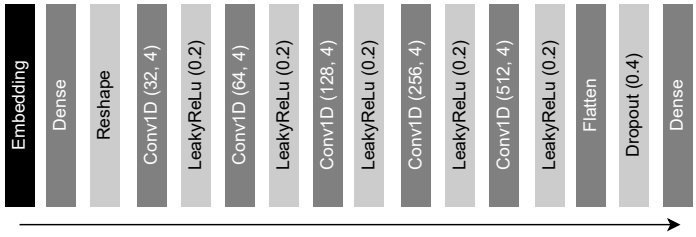


Figure 4.4 cGAN synthetic fingerprints generation - Discriminator model. Reproduced from [180].

a novel algorithm to select the most realistic fingerprints based on the actual distance between the real position of the fingerprints in the training dataset and the position of the new fingerprints. This fingerprint selection leads to a better position estimation and an accurate radio map. The steps to select the most relevant fingerprints are described below.

4.3.2.1 Synthetic fingerprints generation

Firstly, we generate a latent space (i.e., it is a Gaussian distribution with 0 mean and unit σ), which is used as the input of the generator model. The latent space (Γ) generated is a $\eta_{sf} \times n$(where, η_{sf} represents the number of synthetic fingerprints and n the number of features) matrix and from this latent space are generated the new training fingerprints (Ψ'_{XF}) and labels (Ψ'_{yF}).

4.3.2.2 Estimating the position of the synthetic fingerprints

CNN-LSTM model is used to solve regression (position estimation) and classification (classify the fingerprints into floor and building) problems. Hence, the CNN-LSTM is trained using the training set and then is used to predict the position of each synthetic fingerprint generated in the previous step. The CNN-LSTM model is further explained in detail in the following chapters.

4.3.2.3 Computing the distance between real and synthetic fingerprints

Once the synthetic fingerprints are generated and their positions have been estimated, we compute the distance between the positions of the real fingerprints and the positions of the synthetic ones in order to build the distance matrix (D, $\mathcal{R}^{\eta_{sf} \times m}$, where η_{sf} is

the number of synthetic fingerprints and m is the number of real fingerprints in the training dataset) –i.e., the euclidean distance is used to compute the distance between the positions of the fingerprints (see Eq. 4.6).

$$d = d(\Psi'_{yF}, \Psi_{yTR}) = \sqrt{\sum_{i=0}^{p} |\Psi'_{yFi} - \Psi_{yTRi}|^2} \tag{4.6}$$

where, d represents the distance between the position of the synthetic (Ψ'_{yF}) and the real fingerprints Ψ_{yTR}. p is the dimensionality of the position vector (2D or 3D).

4.3.2.4 Selecting relevant synthetic fingerprints

In order to select the most relevant fingerprints we must establish the distance or distances ($\eta_d = [\eta_{d1}, \eta_{d2}, \eta_{d3}, ..., \eta_{dr}]$, where r is the number of the predefined distances) used as a threshold between the position of the real and synthetic fingerprints. Therefore, only the fingerprints in the range of 0 to η_{di} metres are selected.

$$\Psi_{XFS\_i}, \Psi_{yFS\_i} \leftarrow \Psi'_{XFi}, \Psi'_{yFi} \Leftrightarrow \exists d_{ij}, d_{ij} \leqslant \eta_{di} \tag{4.7}$$

where, Ψ_{XFS} represents the synthetic fingerprints selected and Ψ_{yFS_i} their corresponding labels. d_{ij} is the distance in the i-*th* and j-*th* position in the distance matrix (D).

To select the most relevant synthetic fingerprints and enrich the radio map, the proposed algorithm (see Algorithm 4.2) requires five input parameters: the training radio map (Ψ_{XTR}) and their labels – x, y, x, floor and building – (Ψ_{yTR}); the number of synthetic fingerprints will be generated in each iteration (η_{sf}); a list of distances used to select the most realistic fingerprint (η_d); and the number of iterations for each distance in (η_i). As can be observed, for each distance in η_d we generate η_{fs} synthetic fingerprints η_i times. We select the most relevant fingerprints from the synthetic fingerprints generated in each iteration using the steps described above. The outputs of the proposed algorithm are the new training dataset Ψ_{XF} (i.e., the selected synthetic fingerprints and the real ones) and the corresponding labels Ψ_{yF}.

Algorithm 4.2 Fingerprints selection and data augmentation.

Input : Ψ_{XTR}, Ψ_{yTR}, η_d, η_{nfs}, η_i,
Output : Ψ_{XF}, Ψ_{yF}
Function `FPSelecionAugmentation`(Ψ_{XTR}, Ψ_{yTR}, η_d, η_i, η_{nfs}):
    for *dist in* η_d **do**
        for *iter* = 0 *to* η_i **do**
            `/* Latent space Γ */`
            $\Gamma \in \mathcal{R}^{\eta_{sf} \times n}$
                `/* Labels for the latent space */`
            $\Gamma_l \in \mathcal{R}^{\eta_{sf}}$
                `/* 1. Synthetic fingerprints generation */`
            Ψ'_{XF} = cgan.predict($[\Gamma_p, \Gamma_l]$)
                `/* 2. Estimating the position of the synthetic fingerprints (x, y, z, floor and building) (`Ψ'_{yF}`) */`
            Ψ'_{yF} = cnn_lstm.predict(Ψ'_{XF})
                `/* 3. Computing the distance between real and synthetic fingerprints. */`
            $D_{ij} = D(\Psi'_{yF}, \Psi_{yTR})$, $D_{ij} \in \mathcal{R}^{\eta_{sf} \times m}$
                `/* 4. Selecting relevant synthetic fingerprints */`
            $\Psi_{XFS\_i}, \Psi_{yFS\_i} \leftarrow \Psi'_{XFi}, \Psi'_{yFi} \Leftrightarrow \exists d_{ij}, d_{ij} \leqslant \eta_{di}$
        end
        `/* Adding the new fingerprints to the output */`
        $\Psi_{XF} = [\Psi_{XF}, \Psi_{XFS}]$
        $\Psi_{yF} = [\Psi_{yF}, \Psi_{yFS}]$
    end
    `/* Concatenate the new real fingerprints with the artificial fingerprints */`
    $\Psi_{XF} = [\Psi_{XTR}, \Psi_{XF}]$
    $\Psi_{yF} = [\Psi_{yTR}, \Psi_{yF}]$
    return Ψ_{XF}, Ψ_{yF}
End Function

4.3.3 Experiments and results

4.3.3.1 Experiments setup

The experiments were carried out using the hardware, software, and datasets described in Chapter 3. NumPy and TensorFlow v. 2.6.0 libraries were used for numerical computation and ML. The proposed data augmentation model has been tested with eleven multi-floor datasets, two of them multi-building and multi-floor datasets, as described in Table 3.1. Single-floor datasets were omitted, given that Synthetic Minority Oversampling (SMOTE) and Random Over Sampling (ROS) oversampling techniques used to compare against our proposal do not support single-class datasets.

To test the accuracy of both the proposed data augmentation model and the algorithm to select the most relevant fingerprints, we have developed a DNN which combines a CNN and LSTM model. This DNN model is used to estimate the user

position (x, y, z, floor and building). The baseline is the plain k-NN with *simple configuration* as described in Chapter 3. In order to minimise fluctuations in the radio map and differences between features, we applied the powed data representation [167] (see Eq. 4.8) before training the models.

$$Powed = \begin{cases} 0, & \text{if } \psi_{ij} = 0, \\ \left(\frac{\psi_{ij} - min(\Psi)}{-min(\Psi)}\right)^{e}, & \text{otherwise} \end{cases} \tag{4.8}$$

where, Ψ is the radio map; $min(\Psi)$ is the lowest RSS value in Ψ; ψ_{ij} is the RSS value in the i-th position and transmitted by the j-th AP, and e is the constant e.

Once the data representation has been selected, the next step is to set up the hyperparameters of the cGAN model, which can be suitable for any dataset. In this case, only the batch size and number of epochs will be set in this step. To fine-tune these hyperparameters, we selected three public datasets, on which to apply the data augmentation and positioning models. Additionally, three different methods have been proposed to train the cGAN network, which are described below:

Method 1 (M1): Training by building This method is only used if the dataset contains samples from multiple buildings, e.g., UJI1–2. The dataset is divided into buildings, and the cGAN model is trained using the floor label as the condition.

Method 2 (M2): Training by floor This method is similar to method 1, but instead of training the model per building, it is trained by floor (i.e., the condition is the building label).

Method 3 (M3): Full dataset training The cGAN model is trained using the whole dataset. The discriminator/condition in the cGAN model is the floor label.

4.3.3.2 Results

This section presents the results of applying the cGAN data augmentation model to generate synthetic fingerprints. The first step is to set up the hyperparameters used in the cGAN model, then choose the best method to be used in the eleven public datasets described in Chapter 3. To determine these hyperparameters and methods,

Table 4.3 Training parameters for the cGAN using different methods and hyperparameters and the primary results. Reproduced with the permission from [180].

	Training Parameters					Positioning		
Database	Epoc.	BS	Method	η_i	η_d	Ψ_{SF}	ε_{2D}[m]	ε_{3D}[m]
UJI 1	14	64	M1	10	1-5	155	10.01	10.07
	14	64	M1	10	1-10	1415	10.64	10.70
	14	64	M2	10	1-5	213	10.55	10.60
	14	64	M2	10	1-10	1398	11.18	11.25
	14	64	M3	10	1-5	288	10.85	10.91
	14	64	M3	10	1-10	11 138	11.00	11.06
	14	128	M1	10	1-5	311	10.67	10.73
	14	128	M1	10	1-10	2088	11.80	11.84
	14	128	M2	10	1-5	311	11.25	11.31
	14	128	M2	10	1-10	2088	11.93	11.99
	14	128	M3	10	1-5	132	10.97	11.02
	14	128	M3	10	1-10	1677	11.60	11.66
TUT 3	14	64	M2	10	1-5	848	10.23	10.44
	14	64	M2	10	1-10	5175	9.62	9.72
	14	64	M3	10	1-5	3951	9.67	9.76
	14	64	M3	10	1-10	2912	10.13	10.23
	14	128	M2	10	1-5	600	11.41	11.51
	14	128	M2	10	1-10	3498	9.26	9.34
	14	128	M3	10	1-5	449	11.50	11.61
	14	128	M3	10	1-10	4847	9.77	9.86
UTS	14	64	M2	10	1-5	266	7.21	7.57
	14	64	M3	10	1-5	389	7.48	7.68

we selected three datasets that were used widely in previous research studies: UJI 1 (UJIIndoorLoc dataset), TUT 3 (Tampere dataset) and UTS dataset.

Table 4.3 shows the results of training the GAN model with fourteen epochs, and a batch size of 64 or 128. This table also contains the hyperparameters to select the most relevant fingerprints. The results include the number of synthetic fingerprints (Ψ_{SF}) and the mean 2D and 3D positioning error (ε_{2D} and ε_{3D}, respectively).

Method 1 provided a generally good performance in UJI1 dataset using 14 epochs, a batch size equal to 64, 10 iterations and a distance between 1 m to 5 m. The positioning error obtained when using this configuration was approximately 10 m. When the batch size is 128, the lowest mean 2D positioning error obtained was 10.67 m and the 3D error was 10.73 m, which is higher than using a batch size equal to 64.

The results obtained with Method 2 demonstrate a better performance than the

Method 1 in UJI1 with *epoch* = 14, *batch_size* = 64, $\eta_i = 10$ and $\eta_d = 1 - 5$. On the contrary, in TUT3, the lowest positioning error was obtained with a *batch_size* = 128, $\eta_i = 10$ and $\eta_d = 1 - 10$

The third method used to train the GAN network, Method 3, offered a good performance in UJI1 and TUT3 datasets when we used the following parameters: *epoch* = 14, *batch_size* = 64, $\eta_i = 10$ and $\eta_d = 1 - 5$.

Method 2 with *epoch* = 14, *batch_size* = 64, $\eta_i = 10$ and $\eta_d = 1 - 5$ provided greater accuracy than Method 1 and Method 3 in most cases. We thus selected the second method, Method 2, with the parameters mentioned above to train the cGAN and perform its final evaluation in the extended set that includes the 11 multi-floor datasets.

Fig. 4.5 compares the performance of the proposed model with two well-known data oversampling techniques; ROS and SMOTE. The x-axis represents the 11 selected datasets, and the y-axis their average normalised 3D mean positioning error. This figure demonstrates that the proposed data augmentation model offered a superior performance in terms of 3D positioning accuracy than ROS and SMOTE in 8 datasets. Only in LIB2 and TUT2 datasets was the mean 3D positioning error higher than ROS and SMOTE techniques. There were also two datasets where the ROS and SMOTE techniques did not generate any synthetic fingerprints (LIB1–2). In a few cases, ROS, SMOTE, and cGAN have similar mean 3D positioning error.

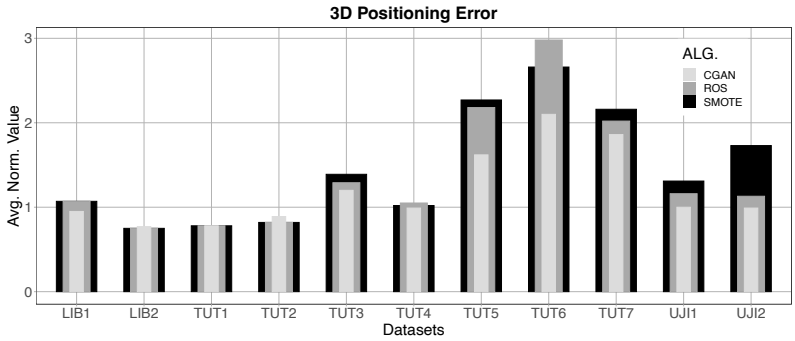


Figure 4.5 Normalized mean 3D positioning error of the cGAN (light grey), ROS (dark gray) and SMOTE (black).

Fig. 4.6 shows the distribution of the mean 3D positioning error using a boxplot and Cumulative Distribution Function (CDF). The first plot, Fig. 4.6(a), shows the distribution of the mean 3D positioning error using a boxplot, showing the median (middle point of the box), the upper quartile (top point of the box), lower quartile

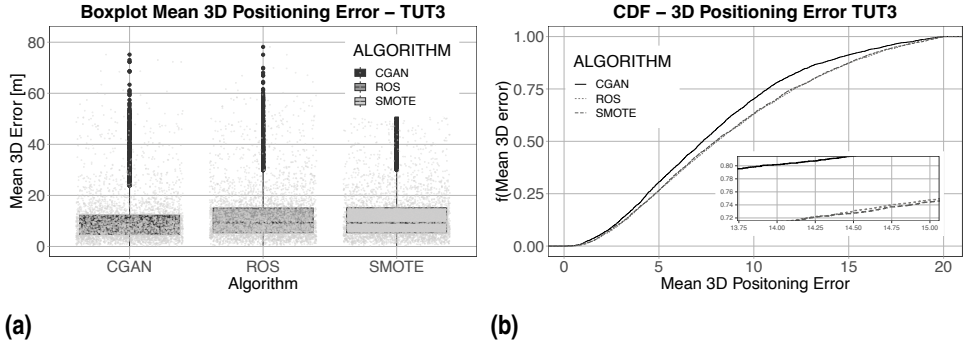


Figure 4.6 Distribution of the Mean 3D Positioning Error of TUT3 dataset using a boxplot (a) and a CDF (b).

(lower point of the box), maximum error (line over the box), minimum error (line below the box), and the outliers (black dots over the maximum value). The light grey dots represent a concentration of errors. The error obtained when we used the proposed cGAN model to generate new fingerprints was lower than when ROS and SMOTE oversampling technique was used. The difference between the three methods, however, was relatively small in the case of TUT3 dataset. The second plot (see Fig. 4.6(b)) shows the cumulative probability of the 3D positioning error. As can be observed from this figure, the proposed data augmentation model (cGAN) provided a better performance than the traditional models (ROS and SMOTE).

Overall, the proposed data augmentation model is almost 14% and 18% more accurate than ROS and SMOTE, respectively.

4.4 Discussion

This chapter introduced two new data optimisation techniques along with performance analysis using heterogeneous radio maps. These two techniques have been widely employed in different research areas, including indoor positioning for the enhancement of data quality, as well as to improve the performance of ML models. Considering that the fingerprinting technique is frequently utilised in indoor positioning applications, it is crucial to provide efficient data optimisation techniques that can reduce the complexity of fingerprinting datasets.

The complexity of fingerprinting datasets is caused, inter alia, by the undesirable fluctuations in signal strength emitted by the access points. These undesirable fluctua-

tions can affect the accuracy of the position estimation ending up in large positioning errors. That is why we have proposed a novel data cleansing algorithm to remove fingerprints labelled as outliers from the radio maps. The proposed algorithm was tested with 26 public datasets (Wi-Fi and BLE), resulting in an improvement in the positioning accuracy (5.3% on average). There are, however, some cases (TUT7, UJIB1, UEXB1, and UEXB3) where the proposed data cleansing algorithm could not detect any outlier without affecting the positioning error regardless of the hyperparameters' values used in the algorithm.

The standard deviation-based method and Isolation Forest could not remove "outliers" without affecting the accuracy of positioning estimation in most of the datasets (see Table 4.2). The standard deviation-based method only detected outliers in 30% of the datasets without increasing the positioning error. The Isolation Forest method could detect outliers in 23% of the datasets efficiently (i.e., without increasing the positioning error), and failed to detect any outlier in 43% of the datasets.

In addition to detecting outliers in the offline phase of fingerprinting, it is essential to filter signals during the online phase. However, detecting outliers in the online phase is often more complex given that positioning, localisation and/or navigation systems require a quick response in the online phase, more so in real-time applications. Another option is to combine different techniques to crosscheck if a fingerprint is an outlier. In this way, removing relevant fingerprints from datasets can be avoided.

The second data optimisation technique suggested in this dissertation relates to data augmentation. In some cases, the number of fingerprints in the radio map proves insufficient to train machine learning models or accurately estimate the user or device positioning. We have therefore proposed a new combination of deep learning models and GAN architectures.

Since the inception of GAN architecture, it has caught the eye of many researchers due to its novel capabilities to generate "synthetic" samples and images that can be indistinguishable from real ones. As a result, multiple researchers have used this architecture in different fields, and new GAN networks have been proposed to overcome some of the limitations of the original model. StyleGAN, for instance, was designed to learn high-level attributes from the input data, obtaining a high-quality output, such as realistic faces. This GAN characteristic caught the attention of researchers in the field of indoor positioning.

On this basis, Section 4.3 provided a comprehensive analysis of the proposed data

augmentation model with 11 public datasets, in addition to a new straightforward algorithm to select the most relevant artificial fingerprints, taking the real position (x, y, z, floor and building) of the training samples as a reference (i.e., the positions of the synthetic fingerprints are estimated using the CNN-LSTM model). The proposed data augmentation model works efficiently in different environments with minimal changes. Despite all the parameters established in the algorithm (see Algorithm 4.2) being fundamental to selecting the most realistic fingerprints, it is important to choose the correct distance (η_d) given that large distances between the positions of the real and the synthetic fingerprints may lead to an increase in positioning error. For example, using a distance below 1 m in LIB1–2 provides lower positioning errors than when using distances above 3 m. Additionally, we have to highlight that a large number of reference points per square meter may not be helpful in reducing the positioning error, which is also mentioned in [181].

The experiments carried out for this dissertation revealed that changes in the number of epochs, batch size, or the number of layers may affect the generator's behaviour, but not necessarily diminish the network performance. This means that small changes in the network can produce new and original fingerprints that can improve the quality of the radio map. For instance, when the batch size was changed from 64 to 128 in Method 2, the positioning error improved for dataset TUT3

It is important to highlight that having multiple fingerprints per reference point significantly increases the accuracy of DNN. e.g., the positioning error obtained in LIB1–2 is lower than in TUT6–7. The average number of fingerprints per reference point is 12 in LIB1–2 and 1 in TUT6–7. However, this is only in the case of DNN. Using k-NN, the positioning accuracy is similar in these datasets.

This chapter has demonstrated the advantages of two data optimisation techniques, and how these techniques assist in providing robust IPSs. The data optimisation has thus been shown to be even more important than the most powerful algorithm or ML model used to predict the user or device position.

4.5 Summary

The performance benefits of the proposed data cleansing and data augmentation model are summarised in the following paragraphs.

- The proposed data cleansing algorithm allowed us to remove "outliers" that can

affect the position estimation, reducing the data size by up to more than 30% without increasing positioning error. In the worst instance, just a few outliers were removed from the dataset without affecting the positioning accuracy. At the same time, the time response was reduced in the online phase of the fingerprinting technique (14% approx.), given that the number of samples was reduced in the training dataset. The proposed data cleansing algorithm outperformed two well-known methods; the standard deviation-based method and the Isolation Forest.

- The second technique introduced in this section was data augmentation. In this case, we used a cGAN architecture to generate synthetic fingerprints. We also proposed a new algorithm to select the most suitable synthetic fingerprints based on the physical distance between the real fingerprints and the synthetic fingerprints generated by the cGAN. This two components are an integral part of the proposed SURIMI framework for data augmentation and indoor positioning.
- The proposed data augmentation model was tested with eleven publicly available datasets and compared with two traditional oversampling methods used most frequently within the literature. The data augmentation model introduced in this chapter generated more realistic fingerprints than using ROS and SMOTE technique, which allows improving the position accuracy in 8 from 11 datasets. Moreover, we can distinguish a significant improvement in most of the datasets, reducing the positioning error up to 24% in some datasets.

This chapter has extended the results of the proposed Data Cleansing model that was published in [174], and the proposed Data Augmentation model that was published in [180].

5 ALGORITHM OPTIMISATION

Along with data optimisation, algorithm optimisation is integral to the performance of any indoor positioning solution or platform. Optimizing algorithms not only improves the accuracy of position estimation but also can reduce the computational load exerted by positioning systems in resource-constrained devices such as wearable and IoT devices. This chapter introduces a post-processing algorithm for Density-based Spatial Clustering of Applications with Noise (DBSCAN) and a new clustering algorithm for fingerprinting. The key contributions of this chapter are as follows:

- Performance analysis of a new post-processing algorithm for DBSCAN to join "noisy" samples to formed clusters.
- Introduction of a novel clustering algorithm based on the maximum RSS values, with experiments in 26 fingerprinting datasets.
- General discussion on the proposed algorithms and methods performance.
- A summary of the key findings.

5.1 Introduction

The fingerprinting technique requires accurate radio maps, which may contain thousands of RSS values needed to estimate the user position. Generally, these radio maps are constantly growing, as they need to be regularly updated, i.e., they grow to adapt to radio environment changes and keep the accuracy of the IPSs. This incremental increase in radio map size directly affects the response time in the operational phase. In view of this continuous increase in indoor positioning data, several scholars have offered techniques or algorithms to manage this data in a more effective way and provide a fast position estimation to the end-user. For instance, the use of clustering algorithms to divide the dataset into small subsets, thus reducing position estimation time in the online phase.

These algorithms used to manage big datasets allow us to offer accurate indoor positioning services, Quality of Service (QoS), Quality of Experience (QoE), and low power consumption. Currently, there are many research articles in the field of indoor positioning which address computational load and data management. For instance, Torres-Sospedra, Quezada-Gaibor, Mendoza-Silva, Nurmi, Koucheryavy, and Huerta [182] offered three new variants to improve the performance of K-Means when it is used with Wi-Fi fingerprinting. Here the authors reduced the computational time whilst simultaneously offering a better distribution of fingerprints through all clusters, reducing the positioning error.

Overall, many of the solutions proposed to manage big datasets in fingerprinting use supervised ML models such as CNN or Recurrent Neural Network (RNN) to transfer most of the computational load to the offline phase of fingerprinting (training phase). Thus, position estimation in the online phase takes less time. Unsupervised ML models are also commonly used to reduce the computational load and time response, for instance, by using clustering algorithms such as the example mentioned above.

5.2 Clustering and Fingerprinting

The use of clustering algorithms arises from the need to manage low and high-dimensional data and classify it into multiple subgroups. In general, clustering algorithms group samples or data points into small groups on the basis of similar characteristics. The parameters used to determine the similarity between samples differ greatly among clustering algorithms. For example, some clustering algorithms group samples based on the density of data points (e.g., DBSCAN), distance/similarity in the feature space (e.g., k-Means, c-Means, etc.), or using probabilistic models.

In the case of the fingerprinting technique, clustering algorithms are also used to group similar fingerprints into clusters. This entails dividing the radio map into several reduced radio maps (clusters), thus reducing the computational load in the online phase of fingerprinting. The complexity of signal propagation and radio maps, however, means that traditional clustering algorithms might not be efficient enough to classify the fingerprints accurately.

On this basis, various authors have proposed modifications to traditional algorithms or even new clustering algorithms to group similar fingerprints in a more

efficient manner, improving the position estimation and/or reducing the time response. For example, Bundak, Abd Rahman, Abdul Karim, and Osman [183] used a clustering algorithm to optimise the reference points and then select the ten most relevant reference points using a rank algorithm, reducing the positioning error by 31%. In the same fashion, Bi, Cao, Wang, Zheng, Liu, Cheng, and Zhao [184] used a clustering algorithm to improve the position estimation. The authors combined three different distance metrics with DBSCAN in order to group the most relevant nearest reference points, obtaining a more stable position estimation by more than 60%.

The following paragraphs outline clustering algorithms used most frequently with the fingerprinting technique.

- k-Means: is an unsupervised clustering algorithm characterised by its simplicity. It is used to discover underlying patterns and group similar data points. Usually, k-Means generates k random centres to which the nearest data points are assigned [185]. The centroid is iteratively updated until the centroid does not (significantly) change.
- Fuzzy c-Means: Unlike k-Means, c-Means generates fuzzy clusters where a data point may belong to one or more clusters, often leading to overlapping clusters. Like the aforementioned clustering algorithm, c- Means is sensitive to outliers [186].
- k-Medoids: This clustering algorithm is similar to k-Means but is more resilient to outliers. In this case, the centroid is represented by one data point in the cluster [187].
- Affinity Propagation Clustering (APC): Basically, this algorithm uses two messages to determine the membership of data points to a certain cluster. These messages identify whether a data point can be considered as an exemplar or not [188].
- DBSCAN: This clustering is used to group similar samples based on two parameters: the minimum number of points ($minPts$) and the minimum distance between samples (Eps). Unlike the clustering algorithms described above, DBSCAN is also used to find outliers in datasets, which are commonly labelled with 0 or -1 [189].

These clustering methods are commonly applied in the offline phase of the fingerprinting technique in order to form clusters. In the online phase, the matching

algorithm performed the coarse and fine-grained searches. During the coarse search, the matching algorithm selects the most relevant cluster or clusters for incoming fingerprints. Once the cluster or clusters are selected, the fine-grained search is performed. In this last search, the incoming fingerprint is compared with all the samples in the selected cluster to find the most similar fingerprint and thus estimate the position.

5.2.1 Improving DBSCAN

As part of the optimisation algorithms developed in this dissertation, we introduce a novel post-processing algorithm for DBSCAN. This method aims to join samples labelled as noise in order to diminish the position error when DBSCAN is used with the fingerprinting technique. The combination of DBSCAN plus the proposed post-processing algorithm will also reduce the time response in the online phase of the fingerprinting being useful for real-time applications.

As previously mentioned, DBSCAN is resilient to outliers. In optimal conditions, DBSCAN removes only those samples that might poison the radio map, thus acquiring high accuracy levels after clustering. However, the complexity of radio maps makes them challenging for many clustering and data cleansing algorithms, leading to the labelling of some useful fingerprints as outliers. The proposed post-processing algorithm reduces the probability of removing useful fingerprints from the radio map.

The following paragraphs describe in detail the post-processing algorithm proposed:

5.2.1.1 Step one - Threshold

Firstly, we define a threshold, (ρ_{DBSCAN}), that determines the percentage of noise permitted in a dataset. The threshold may vary from one dataset to another given the heterogeneity of the fingerprint datasets.

5.2.1.2 Step two - Computing the distance matrix

Secondly, we form the distance matrix ($D \in \mathcal{R}^{m \times m}$) by computing the distance between samples in the training set. This matrix is used in the following steps to search the samples with the lowest distance to the "outliers".

Once the distance matrix is computed, we get the shortest distance ($\varkappa_i, i = 1, 2, 3, ..., m$) between samples (see Eq. 5.1). Then, we compute the relation between the general and the shortest distances (ς), using Eq. 5.2.

$$\varkappa_i = min(D_{ij}) \tag{5.1}$$

$$\varsigma_i = \frac{\varkappa_i}{max(\varkappa)} * 100 \tag{5.2}$$

5.2.1.3 Step three - Joining "outliers" to the formed clusters

In this step, the noisy samples or outliers are added to one of the formed clusters only if the outlier meets the following condition: the relation between the shortest and general distances must be less than or equal to the pre-defined threshold ($\varsigma_i \leq \rho_{DBSCAN}$). We then search for the sample with the lowest distance in the distance matrix – which was not labelled as an outlier – in order to join the outlier to it.

Algorithm 5.1 illustrates the whole process described above. The input parameters are the cluster indexes for each observation ($IDX \in \mathcal{R}^{m \times 1}$), training dataset ($\Psi_{TR}$), and the defined threshold (ρ_{DBSCAN}). The output is the new IDX vector denoted by IDX'. In $\mathcal{I}$, we store the index of the "noisy" samples which fulfil the following condition $\mathcal{I}_i \leftarrow IDX_i \Leftrightarrow \exists IDX_i, (IDX_i = -1 \wedge \varsigma_i \leq \rho_{DBSCAN}), i = 1, 2, 3, ..., m$. Then, we search the sample index with the lowest distance between the "noisy" samples and the "non-noisy" sample. Finally, the noisy samples selected are labelled with the IDX label of the nearest sample, and the centroid of the cluster is recomputed.

Algorithm 5.1 DBSCAN post-processing function.

Input : $IDX, \Psi_{TR}, \rho_{DBSCAN}$
Output : IDX'
Function `DbscanPostprocessing`($IDX, \Psi_{TR}, \rho_{DBSCAN}$):
    $IDX' \leftarrow IDX$
        `/* Distance matrix */`
    $D \in \mathcal{R}^{m \times m}$
        `/* General distance */`
    $\varkappa_i = min(D_i)$
        $\varsigma_i = \frac{\varkappa_i}{max(\varkappa)} * 100$
        `/* Selecting noisy samples with a high level of correlation */`
    $\mathcal{T}_i \leftarrow IDX_i \Leftrightarrow \exists IDX_i, (IDX_i = -1 \wedge \varsigma_i \leq \rho_{DBSCAN})$
      for *i in* $\mathcal{T}$ **do**
        `/* Sorting distances in an ascending manner */`
        index, S_{D_i} = sort(D_i, 'asc')
          for *j in index* **do**
            if $IDX'_j \neq -1$ **then**
                $IDX'_i = IDX'_j$
                  break
            end
        end
    end
End Function

5.2.2 Experiments and Results

5.2.2.1 Experiments setup

The experiments were carried out using the datasets, computer, and baseline referred to in Chapter 3. Similarly, the software used was Python 3.9, and *Sklearn* library in order to run the experiments.

The performance of the proposed algorithm was compared with the plain *k*-NN and the original DBSCAN algorithm without any modification. The parameters used to compare the different approaches with the proposed post-processing algorithm were: mean 2D positioning error (ε_{2D}), mean 3D positioning error (ε_{3D}), building hit rate (ζ_b), floor hit rate (ζ_f) and testing time or prediction time (δ_{TE}). As in the Chapter 4, we have used normalised values.

To run DBSCAN clustering, the first step is to find the optimal values of *Eps* and *minPts*. Given the heterogeneous distribution of the fingerprints in the dataset, we use small values of *minPts* ($0 < minPts < 5$). In this case, we used *k*-NN and the rule-of-the-elbow method to determine the optimal *Eps* value. In general, the point where a sharp change in the distance occurs serves as the threshold (suggested value), and from this point, we can find the most suitable or optimal *Eps* value. To find

the optimal value, we run this method several times, adding 5 to the suggested value until finding the values which provide a good distribution of fingerprints among the clusters and, therefore, the lowest positioning error. For instance, Fig. 5.1 shows the suggested value for the *Eps* parameter and the selected value (optimal value). Fig. 5.1 (a) shows the analysis for UJI1 and Fig. 5.1 (b) for UTS1 dataset.

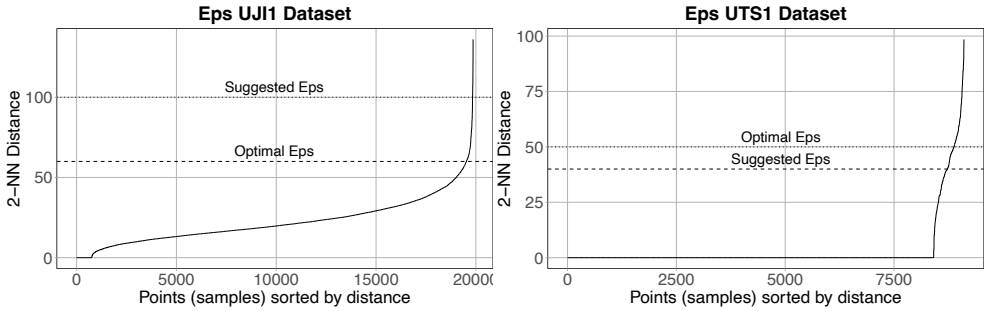


Figure 5.1 Using k-NN to find the optimal value of *Eps*. (a) and (b) show the suggest and the optimal values of UJI1 and UTS1 training sets, respectively.

Table 5.1 Parameters of DBSCAN (*Eps* and *minPts*) and post-processing algorithm (ρ_{DBSCAN}).

Dataset	*Eps*	*minPts*	ρ_{DBSCAN}	Dataset	*Eps*	*minPts*	ρ_{DBSCAN}	Dataset	*Eps*	*minPts*	ρ_{DBSCAN}
LIB1	50	2	50	TUT6	90	2	50	UJIB1	50	2	70
LIB2	80	4	80	TUT7	90	2	50	UJIB2	40	4	70
MAN1	20	2	50	UJI1	100	2	50	UEXB1	40	2	70
MAN2	20	2	50	UJI2	100	2	50	UEXB2	80	2	70
TUT1	290	2	50	UTS1	50	2	80	UEXB3	40	2	70
TUT2	290	2	50	TIE1	180	2	50	OFINB1	180	2	70
TUT3	240	2	50	SAH1	180	2	60	OFINB2	20	2	70
TUT4	600	2	60	OFIN	100	2	60	OFINB3	100	2	70
TUT5	140	2	50					OFINB4	60	2	70

5.2.2.2 Results

This section analyses the results obtained with the proposed post-processing algorithm.

Table 5.1 shows the hyperparameters *Eps* and *minPts* used by DBSCAN and threshold (ρ_{DBSCAN}) used by the post-processing algorithm to form the clusters and redistribute the "noisy" samples. The threshold is directly proportional to the outliers

permitted in the dataset. The higher the threshold, the higher the number of outliers allowed in the dataset.

Table 5.2 Comparing DBSCAN with DBSCAN + the Post-processing Algorithm.

	DBSCAN							DBSCAN + Post-processing						
Dataset	ϕ [−]	o [%]	$\bar{\zeta}_b$ [−]	$\bar{\zeta}_f$ [−]	$\bar{\varepsilon}_{2D}$ [−]	$\bar{\varepsilon}_{3D}$ [−]	$\bar{\delta}_{TE}$ [−]	ϕ [−]	o [%]	$\bar{\zeta}_b$ [−]	$\bar{\zeta}_f$ [−]	$\bar{\varepsilon}_{2D}$ [−]	$\bar{\varepsilon}_{3D}$ [−]	$\bar{\delta}_{TE}$ [−]
LIB1	195	61	-	0.996	1.063	1.066	0.981	15	61	-	0.998	1.022	1.024	1.084
LIB2	366	34	-	0.997	1.055	1.038	0.934	35	34	-	1.013	1.002	0.972	1.065
MAN1	9559	678	-	-	1.230	1.230	0.618	19	678	-	-	1.126	1.126	0.668
MAN2	1	118	-	-	0.993	0.993	1.142	1	118	-	-	0.993	0.993	1.133
TUT1	431	74	-	0.961	1.179	1.072	0.572	1	74	-	0.993	1.059	0.960	0.679
TUT2	18	117	-	1.086	1.121	0.983	0.788	7	117	-	1.102	1.103	0.967	0.879
TUT3	510	34	-	0.833	1.748	1.653	0.438	6	34	-	0.912	1.224	1.158	0.566
TUT4	429	151	-	0.950	1.240	1.191	0.598	5	151	-	0.970	1.188	1.154	0.671
TUT5	267	50	-	0.744	1.948	1.829	0.765	27	48	-	0.971	1.218	1.143	1.094
TUT6	2224	178	-	0.973	3.725	3.739	0.221	29	178	-	0.997	1.368	1.370	0.244
TUT7	1314	295	-	0.984	2.388	2.151	0.231	8	295	-	0.989	1.435	1.293	0.271
UJI1	4664	1342	1.003	0.979	1.349	0.975	0.094	61	1890	1.007	1.022	1.146	0.826	0.153
UJI2	5417	1375	1.000	0.997	1.090	1.067	0.077	49	1375	1.000	0.924	1.044	1.034	0.094
UTS1	196	1557	-	0.994	1.076	0.999	0.183	208	1557	-	0.994	1.076	0.999	0.153
TIE1	285	588	-	0.133	1.340	1.067	0.104	53	159	-	-	1.547	1.171	0.144
SAH1	44	114	-	1.000	0.981	0.959	0.223	12	114	-	1.000	0.977	0.955	0.278
OFIN	1516	17	-	-	2.130	2.130	0.156	45	17	-	-	1.102	1.102	0.690
UJIB1	321	137	-	-	1.001	1.001	1.232	22	137	-	-	0.955	0.955	1.367
UJIB2	370	57	-	-	1.016	1.016	1.186	21	57	-	-	1.004	1.004	1.379
UEXB1	44	22	-	0.978	1.092	1.064	1.180	9	22	-	0.978	1.004	0.981	1.303
UEXB2	7	2	-	0.985	1.050	1.002	1.246	8	9	-	0.985	1.139	1.089	1.341
UEXB3	108	18	1.000	0.848	1.366	1.318	1.188	5	18	1.000	0.891	1.299	1.247	1.327
OFINB1	24	24	-	-	0.844	0.844	0.951	7	24	-	-	0.844	0.844	1.038
OFINB2	186	34	-	-	1.050	1.050	0.785	186	34	-	-	1.048	1.048	0.874
OFINB3	188	59	-	-	0.941	0.941	0.349	105	59	-	-	0.925	0.925	0.405
OFINB4	46	68	-	-	0.999	0.999	2.873	46	68	-	-	0.999	0.999	3.307
Avg.	-	-	1.001	0.908	1.347	1.284	0.735	-	-	1.002	0.984	1.109	1.052	0.854

Table 5.2 shows the normalised results of combining: 1) DBSCAN; 2) k-NN and DBSCAN, 3) post-processing; and 4) k-NN. ϕ represents the number of clusters formed by the DBSCAN clustering algorithm. o is the number of "outliers" detected by the clustering algorithm. If the number of "outliers" is different than zero, it shows that one cluster contains the outliers (samples labelled with −1) and the remaining clusters the valid fingerprints.

When the plain DBSCAN algorithm was used to divide the datasets into small subsets, there was an increment in the mean positioning error of more than 35% and

28% in the 2D and 3D positioning errors, respectively. Similarly, the floor hit rate was negatively affected by almost 10%. The testing time, however, was reduced by more than 26% when compared with the baseline.

Although the positioning error increased in most cases, there are some datasets where the positioning error was reduced after applying DBSCAN without the post-processing algorithm. For instance, in OFINB1 (BLE dataset), the positioning error was reduced by almost 15%. In MAN2, SAH1, OFINB3 and OFINB4, the reduction in the positioning error was less relevant.

When we applied the proposed post-processing algorithm, some of the "noisy" samples were joined to clusters according to the conditions defined in the previous section. This post-processing algorithm allowed us to reduce the positioning error by more than 20% in comparison with the DBSCAN algorithm. However, the 2D positioning error was still greater than the baseline by 10.9%. Additionally, the testing time increased by 12% approx. compared to the DBSCAN without post-processing but remained lower than the baseline.

Fig. 5.2 shows the data distribution among the clusters of UEXB1 (a) and OFIN1 (b) datasets. The x-axis shows the formed clusters and the y-axis represents the number of samples (log scale) in the datasets. The black and grey bars represent the number of samples in each cluster when using DBSCAN and DBSCAN + post-processing algorithm, respectively.

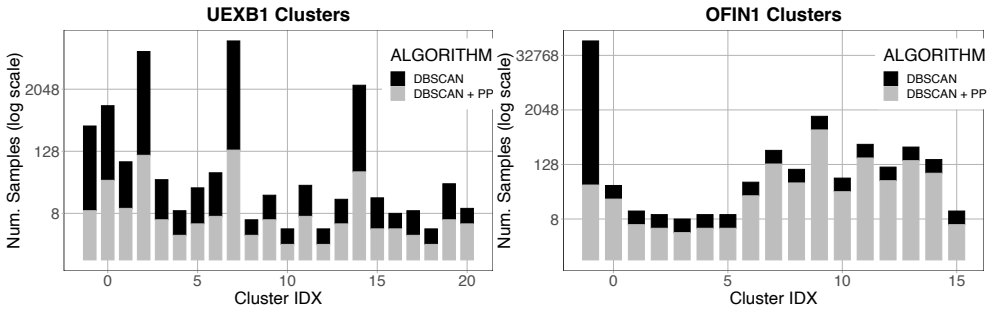


Figure 5.2 Label distribution among the clusters using the DBSCAN and DBSCAN + the post-processing algorithm. (a) shows the number of clusters as well as the data distribution of UEXB1 dataset. (b) shows the data distribution of OFIN1 dataset.

In relation to data distribution, the samples might be unevenly distributed among the clusters. This may be due to the complexity of the radio maps, or because the input

parameters of DBSCAN were improperly selected. For instance, in OFIN1 dataset, the vast majority of samples were labelled as noise, and the remaining clusters contain only one or two relevant samples (see Fig. 5.2). However, the "noisy" samples are successfully redistributed among the DBSCAN clusters when they are post-processed, reducing the positioning error.

Overall, the post-processing algorithm improves the performance of DBSCAN when applied to fingerprint datasets. It is important to highlight here that testing time is reduced only when the number of samples in the datasets is significant. Otherwise, the DBSCAN and the post-processing algorithm do not provide an advantage in terms of time response in the online phase of fingerprinting.

5.2.3 New Clustering Algorithm for Fingerprinting

The use of positioning and localisation services in resource-constrained devices makes the development of efficient algorithms essential, leading multiple authors to propose algorithms to reduce the computational load – and thus the power and memory consumption needed for running positioning and localisation services on these devices. In a similar vein, we introduce FingerPrinting Clustering (FPC), a new lightweight clustering algorithm based on the maximum RSS values and the AP identifier. FPC allows us to fastly divide fingerprinting datasets into small subsets in four stages:

5.2.3.1 Step one – Creating the initial clusters

The initial clusters are built using each fingerprint's maximum RSS values. If the maximum RSS values of different fingerprints belong to the same AP, these fingerprints will form a cluster. This step can be mathematically expressed as follows:

$$\Psi_i \in C_l \Leftrightarrow max(\Psi_i) \in AP_l \tag{5.3}$$

where, Ψ_i is the i-*th* fingerprint in the radio map, C_l represents the l-*th* cluster, $max(\Psi_i)$ is the maximum RSS values of the i-*th* fingerprint, and AP_l is the l-*th* AP.

5.2.3.2 Step two – Computing the centroids

The second step is devoted to computing the centroids of the initial clusters, which corresponds to the mean of all samples in the cluster.

$$\varpi_l = \frac{1}{|S_l|} \sum_{\Psi \in S_l} \Psi \tag{5.4}$$

where, $\varpi_l \in \mathcal{R}^{1 \times n}$ and $S_l \in \mathcal{R}^{m' \times n}$ (m' is the number of samples and n the number of features) are the centroid and set of samples of the l-*th* cluster (C_l), respectively.

5.2.3.3 Step three – Combining small clusters

In this step, small clusters are combined according to their level of correlation. A cluster is small when it has fewer samples than the average among all the clusters. In order to join small clusters, we compute the correlation between the centroids of all the small clusters using the coefficient correlation (see Eq. 5.5).

$$r(\varpi_{sA}, \varpi_{sB}) = \frac{cov(\varpi_{sA}, \varpi_{sB})}{\sigma_{\varpi_{sA}} \sigma_{\varpi_{sB}}} \tag{5.5}$$

where, $r(\varpi_{sA}, \varpi_{sB})$ represents the coefficient correlation between the centroid of the small cluster ϖ_{sA} and ϖ_{sB}. $cov(\cdot)$ is the covariance between the centroids of two small clusters. σ represents the standard deviation.

5.2.3.4 Step four – Updating the centroids

Once all the samples are distributed among clusters, the next step is to recompute the centroid. As in the second step, we use the mean to recompute or update the centroid. This centroid is used in the coarse search of fingerprinting.

Algorithm 5.2 summarises the steps carried out to form the clusters using FPC. This algorithm requires two inputs: the training dataset (Ψ_{TR}) and the number of maximum RSS values (η_{MRSS}). The outputs are the cluster indexes (IDX) and the centroids (ϖ). Once we have the initial clusters with their centroids, we can compute the average number of samples in all the clusters ($\bar{x}$); we then get all the small clusters ($\varpi_{si} \leftarrow C_l \Leftrightarrow num(S_l) < \bar{x}$). The next step is to build the coefficient correlation matrix ($\mathcal{M}$) using Eq. 5.5, which is used to determine the level of correlation

Algorithm 5.2 FingerPrinting Clustering Algorithm.

Input : Ψ_{TR}, η_{MRSS}
Output : IDX, ϖ
$\Psi \leftarrow \Psi_{TR}$
/* 1. Creating the initial clusters. */
$\Psi_i \in C_l \Leftrightarrow max(\Psi_i) \in AP_l$
/* 2. Computing the centroids. */
$\varpi_l = \frac{1}{|S_l|} \sum_{\Psi \in S_l} \Psi$
/* 2.1. Determine the average number of samples in the cluster. */
$\bar{x} = \frac{1}{M} \sum_{l=1}^{M} S_l$
/* 2.2. Get small clusters */
$\varpi_{si} \leftarrow C_l \Leftrightarrow size(S_l)[0] < \bar{x}$
/* 3. Combining small clusters */
/* 3.1. Compute the correlation between the centroids of all small clusters. */
$\mathcal{M} = corr\_matrix(r(\varpi_{sA}, \varpi_{sB}))$
k = number of clusters +1
if $size(\varpi_s)[0]! = 0$ **then**
  while $size(\varpi)[0]! = 0$ **do**
    for $i = 1$ *to* $size(\varpi)[0]$ **do**
      /* 3.2. Group the clusters with a high level of correlation. */
      $NC_k = \varpi_{si} \cup \varpi_{index(max(\mathcal{M}_i))}$
      $k = k + 1$
    end
    $\varpi_{si} \leftarrow NC_p \Leftrightarrow size(NC_l)[0] < \bar{x}$
    $\mathcal{M} = corr\_matrix(r(\varpi_{sA}, \varpi_{sB})$
  end
end
/* 3.3. Join the first clusters whose number of samples is greater than $\bar{x}$ with the new clusters NC. */
$S \leftarrow S \cup NC$
$IDX \leftarrow l$
/* 4. Update the centroids. */
$\varpi_l = \frac{1}{|S_l|} \sum_{\Psi \in S_l} \Psi$

between centroids. Finally, the small clusters are combined according to the level of correlation computed in the previous step, creating a new cluster (NC). These new clusters are joined to the first clusters whose number of samples is greater than $\bar{x}$, and the centroid is updated.

5.2.4 Experiments and Results

5.2.4.1 Experiments setup

The experiments were carried out using the hardware, datasets, and baseline described in Chapter 3. The clustering and post-processing methods were implemented in octave 6.2.0.

In order to test the performance of the proposed clustering algorithm, we ran the FPC with the parameters described in Table 5.3. Once the FPC generated the clusters, we used the number of clusters generated by the FPC to set the hyperparameters of k-Means and c-Means (k and c, respectively).

Table 5.3 Parameters - Clustering Algorithms.

Dataset	k-Means k	c-Means c	FPC η_{MRSS}	Dataset	k-Means k	c-Means c	FPC η_{MRSS}	Dataset	k-Means k	c-Means c	FPC η_{MRSS}
LIB1	15	15	1	TUT6	1077	1077	4	UJIB1	161	161	4
LIB2	15	15	1	TUT7	946	946	3	UJIB2	200	200	4
MAN1	49	49	3	UJI1	1463	1463	3	UEXB1	50	50	2
MAN2	22	22	4	UJI2	2970	2970	4	UEXB2	56	56	2
TUT1	86	86	1	UTS1	1049	1049	4	UEXB3	19	19	1
TUT2	137	137	2	TIE1	2994	2994	4	OFINB1	27	27	2
TUT3	120	120	1	SAH1	3275	3275	4	OFINB2	7	7	2
TUT4	571	571	2	OFIN	154	154	4	OFINB3	24	24	2
TUT5	158	158	2					OFINB4	28	28	2

The proposed FPC clustering algorithm was compared to k-Means and c-Means clustering (parameters listed in Table 5.3). The clustering models were compared in terms of mean 2D positioning error (ε_{2D}), mean 3D positioning error (ε_{3D}), the time required to form the clusters, and the time required to estimate the user position (δ_{TE}).

5.2.4.2 Results

This section reports the results of applying the proposed clustering algorithm (FPC) with 26 public datasets and against two traditional clustering algorithms.

Table 5.4 shows the results of combining k-NN with k-Means, c-Means and FPC. The results shown in the table correspond to the normalised mean 2D and 3D positioning errors. Overall, the best positioning accuracy is obtained with k-Means, and the lowest execution time (δ_{TE}) is acquired with FPC.

Comparing k-Means with FPC, the mean 2D positioning error obtained with FPC is 28% higher than k-Means and 32% in the mean 3D positioning error. However, the increase in error is offset by the reduction in the processing time, which was reduced by 61.29%. Similarly, the positioning error acquired with c-Means is 12% better than the error obtained with FPC, but the processing time is much smaller (82.79% less).

Table 5.4 Comparison: k-Means, c-Means and FPC.

	k-Means				c-Means				FPC			
Dataset	ϕ [−]	$\bar{\varepsilon}_{2D}$ [−]	$\bar{\varepsilon}_{3D}$ [−]	δ_{TE} [s]	ϕ [−]	$\bar{\varepsilon}_{2D}$ [−]	$\bar{\varepsilon}_{3D}$ [−]	δ_{TE} [s]	ϕ [−]	$\bar{\varepsilon}_{2D}$ [−]	$\bar{\varepsilon}_{3D}$ [−]	δ_{TE} [s]
LIB1	15	1.04	1.05	0.16	15	1.03	1.03	0.33	15	0.99	0.99	0.15
LIB2	15	1.02	0.99	0.10	14	0.97	0.96	0.27	15	1.08	1.20	0.06
MAN1	49	0.99	0.99	0.05	43	1.04	1.04	0.27	49	0.96	0.96	0.06
MAN2	22	0.99	0.99	0.05	19	1.00	1.00	0.30	22	0.99	0.99	0.12
TUT1	86	0.90	0.96	0.36	4	0.98	0.96	1.46	86	1.08	1.21	0.11
TUT2	137	1.00	1.06	1.10	10	1.05	1.05	3.98	137	0.96	1.05	0.66
TUT3	120	1.02	1.05	0.15	2	1.16	1.22	0.62	120	1.35	1.48	0.07
TUT4	571	1.07	1.07	2.02	2	1.20	1.24	4.38	571	1.53	1.92	0.68
TUT5	158	1.00	0.98	0.41	2	1.10	1.13	1.43	150	1.18	1.18	0.18
TUT6	1077	1.23	1.24	0.38	3	2.19	2.29	1.10	1076	3.17	3.21	0.29
TUT7	946	1.07	1.09	0.33	17	1.50	1.69	0.77	896	2.18	2.48	0.16
UJI1	1463	1.01	0.95	2.90	8	1.40	1.47	2.27	1463	1.19	1.09	0.58
UJI2	2970	0.99	1.08	1.33	12	1.01	1.05	1.34	2970	1.07	1.75	0.49
UTS1	1049	1.00	1.00	3.48	2	1.00	1.00	10.74	1049	0.97	0.96	0.89
TIE1	2994	0.86	1.01	152.44	4	1.15	0.98	332.62	2815	1.01	0.83	51.12
SAH1	3275	0.58	0.93	45.26	14	0.51	0.87	53.49	3275	0.84	1.01	23.25
OFIN	154	1.01	1.01	1.51	60	1.30	1.30	11.88	154	1.17	1.17	0.29
UJIB1	161	1.02	1.02	0.18	54	1.10	1.10	1.03	159	1.04	1.04	0.20
UJIB2	200	1.03	1.03	0.30	6	1.09	1.09	2.57	200	1.09	1.09	1.03
UEXB1	50	1.01	1.06	0.53	37	1.12	1.32	4.17	50	1.54	1.56	0.53
UEXB2	56	0.98	0.98	0.34	36	1.22	1.26	3.66	56	1.37	1.34	0.35
UEXB3	19	0.99	1.02	0.47	4	1.02	1.03	5.27	19	1.33	1.34	0.93
OFINB1	27	0.90	0.90	1.22	12	1.30	1.30	19.26	27	1.51	1.51	0.44
OFINB2	7	1.01	1.01	0.69	7	0.97	0.97	8.38	7	1.08	1.08	0.52
OFINB3	24	1.03	1.03	1.08	5	1.08	1.08	17.17	24	1.18	1.18	0.50
OFINB4	28	1.13	1.13	0.81	9	1.47	1.47	0.76	28	1.25	1.25	0.70
Avg.	-	0.99	1.02	8.37	-	1.15	1.19	18.83	-	1.27	1.34	3.24

Fig. 5.3 shows the time required by the clustering algorithms used in the experiments to form the clusters – labelling the training samples and computing the centroid. The x-axis represents the dataset used to test the processed clustering algorithm, and the y-axis, the time (logarithmic scale) required to form the clusters (in seconds). As can be observed from this figure, FPC is faster than all the traditional clustering algorithms (c-Means and k-Means), but there are some cases where traditional algorithms are slightly faster than FPC, e.g., MAN1–2, TUT6, UEXB2–3 and UJIB1.

Figure 5.3 Execution time to form the clusters using FPC, c-Means and k-Means.

5.3 Discussion

This chapter introduced two new algorithms linked to clustering, which are devoted to reducing the positioning error and the computational load. The first section suggested a new post-processing algorithm for DBSCAN, which joins some "noise" samples (outliers) to the formed clusters, reducing the positioning error in comparison with the DBSCAN without any modification or post-processing. The second part of the chapter introduced a new lightweight clustering algorithm for fingerprinting, which is devoted to reducing the computational cost of forming clusters.

As can be observed from Table 5.2, the post-processing methods reduced the mean position error of DBSCAN by almost 24% and improved the floor hit rate by more than 2% on average. The main reason for these improvements is the redistribution of the "noisy" samples to the formed clusters. For instance, the mean 2D positioning error was reduced by 8% in TUT1 after redistributing all samples labelled as "noisy". There were other cases where not all "noisy" samples were redistributed to the formed clusters, e.g., in UJIB1, only 40% of noisy samples were joined.

This post-processing method provides a better distribution of samples among the formed clusters in comparison with the original DBSCAN clustering (see Fig. 5.2). The post-processing algorithm does not, however, guarantee a homogeneous distribution of the noisy samples, as shown in Fig. 5.2.

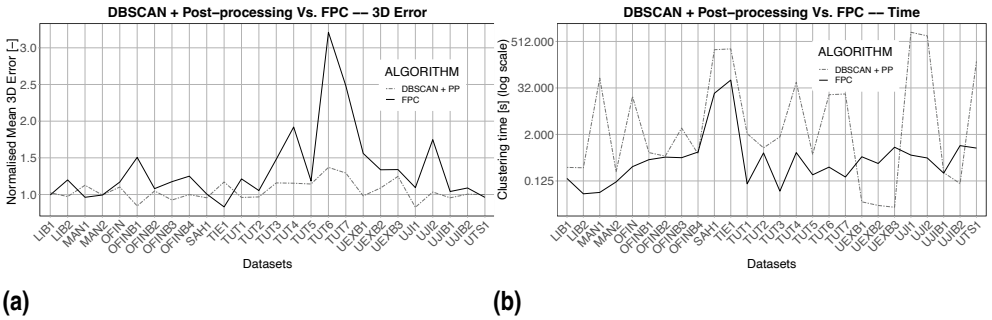
Although the post-processing algorithm reduced the positioning error in most datasets (compared with the plain DBSCAN), there were some datasets where the positioning error increased, such as in UJI2 and TIE1. In the case of UJI2, 15% of

the "noisy" samples were redistributed to the formed clusters, increasing the mean 2D positioning error by more than 3%.

This chapter also provided a straightforward method to join fingerprints based on the maximum RSS values. Overall, FPC is 61.29% faster than *k*-Means and 82.79% faster than *c*-Means. However, the low computational load was at the expense of an increment in the positioning error.

The computational load was greatly reduced in large datasets. For instance, in UJI1 dataset, *k*-Means required 2.90 s to form 1463 clusters, *c*-Means required 2.27 s, and FPC only required 0.58 s to form the same number of cluster, reducing the computational load needed to estimate position by more than 70% (see Table 5.2). Similarly, the time required to estimate the position was significantly reduced in UJI2, UTS1, TIE1 and SAH1, where there are more than 9000 samples in each dataset.

When the number of samples is not significant, the performance of combining *k*-Means and *k*-NN was better than combining FPC and *k*-NN in some cases. However, these small datasets do not require the use of clustering algorithms, e.g., UEXB1–2 where the number of samples is less than 600. In some cases, the time required to compute the user position combining clustering and *k*-NN may be higher than only using *k*-NN.



(a) **(b)**

Figure 5.4 Comparison between DBSCAN + post-processing and FPC agorithms. (a) shows the normalised mean 3D positioning error and (b) the time required to form the clusters.

Fig. 5.4 shows the comparison between DBSCAN + post-processing and FPC clustering algorithms. As can be observed from Fig. 5.4 (a), DBSCAN + post-processing algorithm is more accurate than FPC by almost 30%, offering a better distribution of fingerprints among the clusters. Nevertheless, it is important to consider that FPC requires less time to form the clusters compared with DBSCAN + post-processing (see Fig. 5.4 (b)). On average FPC is more than 90% faster than

DBSCAN + post-processing, making it more computationally efficient. Given that both algorithms were developed using different programming languages, the time to form the clusters may slightly differ.

We have to take into account that none of the clustering algorithms tested provided an equal distribution among the clusters. However, FPC reduces the possibility of having numerous small clusters, joining them according to the level of correlation between centroids.

The computational efficiency provided by FPC makes it suitable for use in resource-constrained devices or even in applications where an accurate position is not required, e.g., to localise classrooms, offices, or boarding gates in airports.

5.4 Summary

This section summarises the primary results of the proposed post-processing and clustering algorithm.

- The proposed post-processing algorithm for DBSCAN clustering successfully redistributed the most relevant "noisy" fingerprints into the formed clusters, reducing the mean positioning error by more than 20% in comparison with DBSCAN without post-processing. Additionally, the proposed combination (DBSCAN and post-processing) reduced the time required to estimate the position in the online phase by more than 14% compared to the baseline.
- In contrast with traditional clustering algorithms, the core FingerPrinting Clustering (FPC) is not based on any distance metric, only on the maximum RSS values of each fingerprint in the dataset. As a result of using FPC clustering, the average time required to form the cluster was reduced significantly when compared with k-Means and c-Means, $\approx$ 61.29% and $\approx$ 82.79%, respectively. However, the low computational load is at the cost of a low level of accuracy (i.e., the positioning error increased by more than 28%).

This chapter has extended the results of the proposed post-processing model applied to DBSCAN that was published in [190], and the proposed FPC clustering model that was published in [191].

6 POSITIONING AND LOCALISATION

This chapter introduces two ML models for the estimation of user/device position and location. These models combine DNN and ELM architectures to learn complex patterns in fingerprinting datasets. This chapter is organised as follows:

- A general introduction to the proposed ML models for fingerprinting.
- Presentation of DNN (CNN-LSTM) model for indoor positioning, using fingerprinting datasets. We also include the performance analysis of the proposed model with 23 public fingerprinting datasets.
- Description and performance analysis of a lightweight CNN-ELM for multi-building and multi-floor classification.
- A summary of the proposed ML models and results.

6.1 Introduction

MLs models used in current applications are diverse, from basic fully connected neural networks to complex networks with hundreds of layers and neurons. This neural networks have been used to learn the complex pattern of radio maps (e.g., CNN) [162], extract local features (e.g., LSTM) [192], and split the radio map into subgroups using traditional clustering algorithms such as DBSCAN [184].

These ML models fall into three categories: supervised, semi-supervised, and unsupervised. Researchers in the field of indoor positioning are currently using many ML models from those categories to reduce positioning error, improve the classification of fingerprints into floor and building, and for security & privacy. For example, Shen, Li, Chen, and Wang [193] proposed a new algorithm to build an accurate radio map from crowdsourced data. In this case, the authors used HDBSCAN to extract the most relevant fingerprints and balance the path travelled by the users. Wang, Wang, Zhao, and Zhang [194] suggested an improvement to Graph-based

Semi-Supervised Learning (GSSL) in order to improve the positioning accuracy and reduce the manual labour of data collection. The authors combined the improved GSSL with double-weighted k-NN, obtaining a confidence level of 95% in the best instance. Song, Fan, Xiang, Ye, Liu, Wang, He, Yang, and Fang [162] provided a new CNN-based indoor positioning model to detect the floor and estimate the user position accurately. The authors combined Stacked Autoencoder (SAE) and CNN, obtaining a floor hit rate of 95% and a mean 2D positioning error of 11.78 meters in the UJIIndoorLoc dataset.

Most of the related work in fingerprinting and ML has focused on processing radio maps and estimating a position. However, obtaining a high level of accuracy with fingerprinting is still a challenge due to undesirable fluctuations in the RSS values and the heterogeneity of available datasets. Therefore, it is important to research new models that can overcome these limitations. Despite these challenges, fingerprinting is still the technique employed most frequently owing to its low cost and the ubiquitous distribution of wireless access points in indoor and outdoor environments.

In this chapter, we introduce two ML models that exploit the advantages of DNN and ELM in order to provide a robust and generalised solution for fingerprinting-based indoor positioning applications. The first model combines CNN and LSTM to estimate the user or device position (positioning model of the SURIMI framework introduced in Chapter 4) Quezada-Gaibor, Torres-Sospedra, Nurmi, Koucheryavy, and Huerta [180]. The second model uses CNN and ELM to provide a fast and accurate solution for classifying fingerprints into floor and building Quezada-Gaibor, Torres-Sospedra, Nurmi, Koucheryavy, and Huerta [13]. Unlike other models featured within the literature, the two models suggested here have been tested with multiple datasets, demonstrating their robustness for use in different and heterogeneous indoor environments.

6.2 CNN-LSTM model for Position Estimation

Indoor positioning based on the fingerprinting technique presents both opportunities (e.g., easy to implement) and challenges (e.g., fluctuations in RSS), many of which have been extensively analysed by the research community [195]. Researchers have suggested some models and algorithms to circumvent the limitations of the finger-

printing technique. For example, the use of CNN and LSTM models have been proposed to overcome noise and uncertainty in radio maps [178].

CNNs have been widely used for regression and classification problems in different areas, such as image processing. These NNs are often composed of one or more layers for feature extraction and fully connected layers to determine the network's output [196]. Similarly, LSTM networks are part of NN, which belongs to RNN. Unlike CNN, LSTM networks recognise patterns in sequential data, taking into account the time and data sequence [192].

This combination of CNN and LSTM has already been used within the research area to improve positioning accuracy. For instance, Wang, Luo, Wang, Li, Zhao, and Huang [197] proposed a spatial-temporal positioning algorithm which combines CNN and LSTM. The CNN model is used to extract features from the radio map and the LSTM for temporal feature extraction, overcoming the gradient explosion in long-time series. As a result, the positioning error was reduced by more than 50% compared to well-known previous works.

The CNN-LSTM model has also been used with other indoor positioning technologies such as infrared sensors, e.g., Ngamakeur, Yongchareon, Yu, and Sheng [198] used Passive Infrared (PIR) sensor, as the primary indoor positioning technology to predict user position. Along with PIR technology, the authors used the CNN-LSTM model for feature discovery and spatial-temporal feature extraction, as well as the research work described above.

This dissertation differs from existing research in that it provides a comprehensive analysis of a new CNN-LSTM model with 23 public Wi-Fi and BLE fingerprinting datasets collected in indoor and outdoor environments (see Chapter 3). The model proposed is compared with a CNN model from the literature.

6.2.1 Model Description

The suggested CNN-LSTM model aims to learn complex patterns and the changes in the signal fluctuation of the radio maps to reduce the positioning error. This model is used in this dissertation to address two classification problems and one regression problem.

The first model dealt with the classification of fingerprints into buildings. It is composed of the following layers: a *Conv1D* layer responsible for extracting spatial information from the radio map; a *MaxPool1D* which is used to compute the

maximum value for each patch in the activation map; a *dropout* layer, which deactivates some neurons during the training stage, used to avoid overfitting. Then, a Flatten is used to convert a multidimensional input into a one-dimensional representation. These fourth layers are wrapped by a time-distributed layer, which allows applying a layer to every data slice of the input.

The second classification model contains an additional block of a *Con1D*, *MaxPool1D* and a *dropout* layer before the LSTM. This model is devoted to classifying the samples into floors.

The regression model is similar to the classification models, but it contains three blocks of a *Con1D*, *MaxPool1D* and a *dropout* layer before the LSTM. This model is used to estimate the user/device position (x, y, z or latitude, longitude, and altitude).

Fig. 6.1 shows the layers, parameters, and values of the three models described above. CF represents the number of floors, CB is the number of buildings, and AF is the activation function. This figure also contains the training parameters of each model, along with their values. In addition to the dropout layers used in each model, the early stopping optimisation technique was used to avoid overfitting.

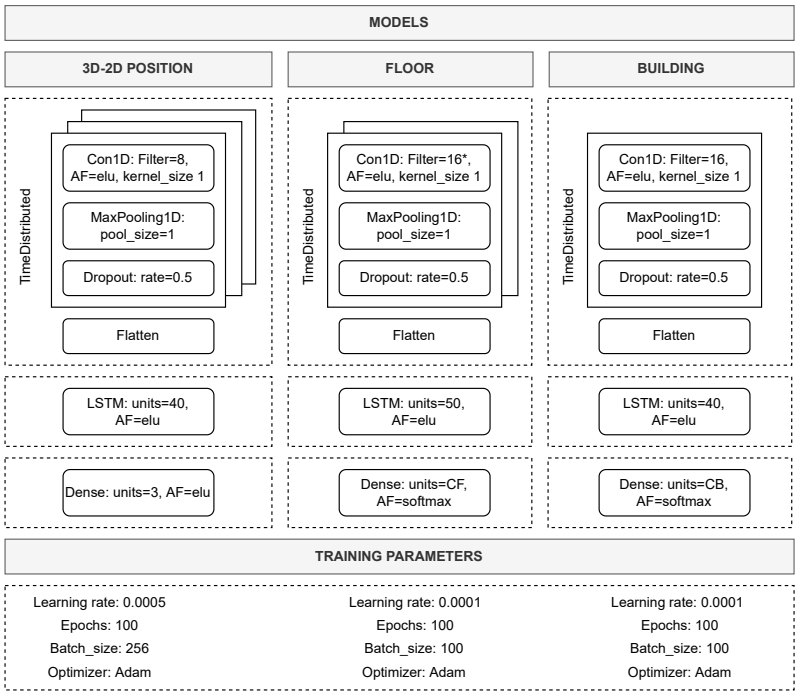


Figure 6.1 CNN-LSTM models, layers and hyperparameters.

6.2.2 Model training and Position Estimation

Chapter 3 described the fingerprinting technique, which consists of two phases: offline and online. In this case, the same indoor positioning technique is used, but unlike the diagram presented in Chapter 3, the CNN-LSTM model is included in the offline phase of fingerprinting technique. During the online phase, the trained CNN-LSTM is used to predict or estimate the user position.

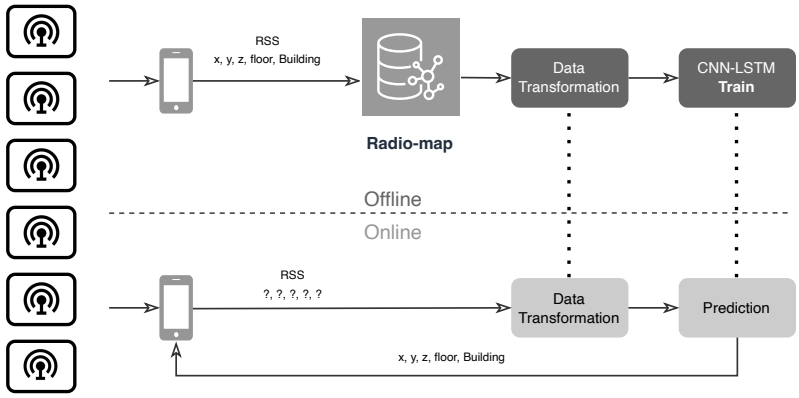


Figure 6.2 Fingerprinting-based indoor positioning using the proposed CNN-LSTM model.

Fig. 6.2 shows the offline and online phases of the fingerprinting technique using the proposed CNN-LSTM. In the offline phase, the data transformation and the training of the proposed model are carried out. The data transformation consists of changing the data representation using Eq. 4.8 [167]. During the online phase, the incoming data will be changed using powed data representation. This "new data" is used to predict the user position (x, y, z, floor and building).

6.2.3 Experiments and results

6.2.3.1 Experiments setup

The experiments were carried out using the hardware and the baseline described in Chapter 3. The CNN-LSTM model was implemented using Python 3.9 and TensorFlow library. This model has been tested with 23 public datasets described in Table 3.1. In this case, UEXB1–3 datasets were not included beacause CNNLoc, a

state-of-the-art CNN model obtained from the literature and used to compare our ML model, was not able to process them.

CNNLoc combines an SAE and a CNN for feature extraction [162]. The source code is available in the authors' repository for public use [199]. In this case, we made minor changes to estimate the mean 3D position, as the original source code only provides the mean 2D positioning error. The hyperparameters are the same as the provided by the authors (Mean Squared Error (MSE) loss function, Adam Optimizer, 40 epochs, dropout rate equal to 0.7, and a learning rate of 0.0001).

In order to compare the performance of the CNNLoc and the CNN-LSTM model, we used the following parameters: building hit rate (ζ_b), floor hit rate (ζ_f), mean 3D positioning error (ε_{3D}) and mean 2D positioning error (ε_{2D}). As in previous experiments, we provide the normalized values for the comparison. For this experiment, the training datasets were divided into validation and training, 10% and 90%, respectively, in order to train both models on equal terms. In this case, it is very important to ensure that validation points are not in the training and test datasets.

6.2.3.2 Results

Table 6.1 shows the results of testing the CNNLoc and CNN-LSTM models with 23 datasets. Comparing the CNNLoc with the baseline, the floor hit rate is 3% better than when using the *k*-NN with *simple configuration*. Conversely, the mean 2D and 3D positioning accuracy decreased by 69.7% on average. In datasets like LIB2, OFIN and UJIB1–2, the performance of the CNNLoc was higher than the baseline. For example, in the LIB2 dataset, the mean 2D error was reduced by more than 10% and the mean 3D positioning error by around 8%. The lowest positioning error was acquired in the mean 2D positioning error of UJIB1–2 and OFINB3.

The CNN-LSTM provided a better floor hit rate than the baseline and the CNNLoc by 9.1% on average. The positioning error increased by more than 28% compared with the baseline. However, it is more than 18% lower than the CNNLoc. The positioning error was lower than the baseline in seven datasets (LIB1–2, MAN1–2, TUT1, UJIB1 and OFINB3), and higher in the remaining datasets. However, there were three datasets where the positioning error increased more than twice compared with the baseline; this phenomenon occurred in both the CNNLoc and CNN-LSTM models.

Table 6.1 Comparison: CNNLoc Vs. CNN-LSTM.

	CNNLoc				CNN-LSTM			
Dataset	$\bar{\zeta}_b[-]$	$\bar{\zeta}_f[-]$	$\bar{\varepsilon}_{2D}[-]$	$\bar{\varepsilon}_{3D}[-]$	$\bar{\zeta}_b[-]$	$\bar{\zeta}_f[-]$	$\bar{\varepsilon}_{2D}[-]$	$\bar{\varepsilon}_{3D}[-]$
LIB1	–	1.000	1.079	1.188	–	0.992	0.937	0.999
LIB2	–	1.019	0.882	0.916	–	1.016	0.725	0.739
MAN1	–	–	1.083	1.742	–	–	0.839	0.839
MAN2	–	–	1.204	1.997	–	–	0.868	0.868
TUT1	–	0.971	1.411	1.289	–	1.027	0.987	0.899
TUT2	–	1.266	1.823	1.589	–	1.172	1.182	1.033
TUT3	–	0.981	1.996	1.871	–	1.004	1.290	1.211
TUT4	–	0.971	2.032	1.966	–	1.002	1.077	1.044
TUT5	–	1.116	2.338	2.162	–	1.108	1.924	1.780
TUT6	–	0.999	4.793	4.807	–	0.999	2.740	2.746
TUT7	–	0.974	5.373	4.835	–	0.992	2.456	2.212
UJI1	1.005	1.082	2.333	1.666	1.008	1.068	1.399	0.997
UJI2	1	0.982	1.797	1.735	1	1.059	1.044	1.011
UTS1	–	0.986	1.021	0.997	–	1.011	1.032	0.945
TIE1	–	0.367	2.393	1.601	–	1.100	2.887	2.037
SAH1	–	1.041	1.537	1.399	–	1.726	1.094	1.028
OFIN	–	–	0.894	1.083	–	–	2.571	2.571
UJIB1	–	–	0.637	1.397	–	–	1.001	1.001
UJIB2	–	–	0.499	0.932	–	–	0.820	0.820
OFINB1	–	–	1.167	1.228	–	–	1.050	1.050
OFINB2	–	–	0.944	0.944	–	–	1.760	1.760
OFINB3	–	–	0.573	0.978	–	–	0.795	0.795
OFINB4	–	–	1.216	1.284	–	–	1.261	1.261
Avg.	–	1.030	1.697	1.722	–	1.091	1.380	1.289

6.3 CNN-ELM Model for Fingerprints Classification.

Section 6.2 provided an analysis of the combination of two ML models, CNN and LSTM, and how they are used in indoor positioning solutions to acquire a high level of positioning accuracy. Although the previous model, CNN-LSTM, provides good general performance, it requires high computational resources for training, rendering it unsuitable for resource-constrained devices. Therefore, we investigated other state-of-the-art approaches to be included in fingerprinting.

In this section, we introduce a new combination based on deep learning and Extreme Learning Machine (ELM) for fingerprint classification in terms of building and floor. The model provides a balanced trade-off between accuracy and power consumption.

Unlike the CNN or LSTM models, ELM does not use a backpropagation algorithm or similar to adjust the output weights; rather, they are analytically determined using *Moore-Penrose Pseudoinverse*. The input weights of the ELM do not necessarily have to be updated or adjusted [200]. The training and prediction phase is thus faster than traditional neural networks. This NN can be used for classification, regression, dimensionality reduction, and even clustering [201, 202, 203, 204, 205].

The use of ELM for indoor positioning has different objectives. Zou, Lu, Jiang, and Xie [206] used an online sequential extreme learning machine (OS-ELM) to learn the environment dynamics and reduce the manual time-consuming data collection required for fingerprinting. ELM has also been combined with other algorithms to reduce the effects of noise in radio maps. Gan, Khir, Witjaksono Bin Djaswadi, and Ramli [207] used multi-kernel ELM for its robustness and better classification performance and combined it with Kernel Principal Component Analysis (KPCA) to extract the coarse features of the radio map.

In contrast with previous works, we combine a low-level CNN model to extract the meaningful features from the radio map and ELM to classify the fingerprints into building and floor. We thus provide a lightweight and accurate hybrid model which can be used in multiple scenarios.

6.3.1 Model Description

This section describes the CNNs, the basics of ELM and the proposed architecture.

6.3.1.1 Data preparation

Data preparation techniques in current use include data transformation, data cleansing, and validation. These techniques can be used to improve the quality of the radio maps and, therefore, improve the learning capabilities of the ML models. In this case, two data preparation techniques were applied to the radio maps. The first technique is devoted to changing the data representation using *powed* data representation [167] in the same manner as in the previous model (CNN-LSTM). The second technique is data normalisation, which adds an additional statistical constraint to the data. Here,

unit norm normalisation was applied to the radio maps using Eq. 6.1.

$$\widehat{\Psi}_j = \frac{\Psi_j}{\|\Psi_j\|}, j = 1, 2, ..., n \tag{6.1}$$

where, $\widehat{\Psi}_j$ represents the normalised values of the j-*th* (AP), Ψ_j represents all the RSS values in the j-*th* feature (AP) in the radio map, and $\|\Psi_j\|$ is the Euclidean norm of Ψ_j feature.

6.3.1.2 Convolutional Neural Network (CNN) Model

The proposed feature learning model is composed of three layers devoted to extracting the most relevant characteristics from the radio map. The first layer is a *Conv1D* to extract the spatial characteristics of the dataset. The second layer is a *Pooling1D* used to reduce the dimensionality of the data according to the defined pool size. Finally, a *batch_flatten* used to transform each sample of the batch into a 2D vector. The hyperparameters for this model based on CNN are reported in Fig. 6.3.

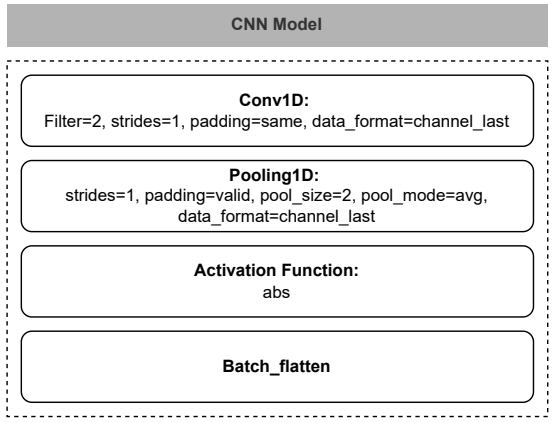


Figure 6.3 CNN parameters.

6.3.1.3 Extreme Learning Machine (ELM) Basics

ELM is the algorithm proposed by [200] to train Single Hidden Layer Feedforward Neural Network (SLFN) networks. Unlike traditional learning algorithms such as back-propagation, ELM determines the output weight analytically using *Moore-*

Penrose Pseudoinverse as was previously mentioned. The input weights and bias terms are randomly generated and do not necessarily have to be adjusted. Considering N arbitrary distinct samples $(\psi_i, \mathbf{t}_i)$, where $\psi_i = [\psi_{i1}, \psi_{i2}, ..., \psi_{in}]^T \in \mathcal{R}^n$ is i-*th* fingerprint and $\mathbf{t}_i = [t_{i1}, t_{i2}, t_{i3}, ..., t_{im}]^T \in \mathcal{R}^m$ $(i = 1, 2, ..., N)$ is the i-*th* target (e.g., floor or building). In this case, the aim is to find the correlation between the input (ψ_i) and the target $(\mathbf{t}_i)$. The ELM is similar to solving a least squares problem [200, 208]. The SLFN is represented as follows.

$$\sum_{i=1}^{L} \beta_i h(\mathbf{w}_i \cdot \psi_j + b_i) \triangleq t_j, j = 1, 2, ..., N \tag{6.2}$$

where $\beta_i = [\beta_{i1}, \beta_{i2}, ..., \beta_{im}]^T$ is the weight vector which connects the i-*th* hidden node with the output layer. $h(\cdot)$ is the activation function. $\mathbf{w}_i = [w_{i1}, w_{i2}, ..., w_{in}]^T \in \mathcal{R}^n$ represent the input weights, which connects the input with hidden neurons. $b_i \in \mathcal{R}$ is the bias term, and L is the number of hidden neurons. $\mathbf{w}_i \cdot \psi_j$ represents the dot product between weights and the input [200]. Eq. 6.2 can also be represented as follows:

$$\mathbf{T} = \mathbf{H}\beta \tag{6.3}$$

where $H \in \mathcal{R}^{N \times L}$ is the hidden layer output matrix. $\beta \in \mathcal{R}^{L \times m}$ represents the output weights of the ELM, which connect the hidden neurons with the output and $\mathbf{T} \in \mathcal{R}^{N \times m}$ is the target matrix,

$$\mathbf{H} = \begin{bmatrix} h(\mathbf{w}_1 \cdot \psi_1 + b_1) & \dots & h(\mathbf{w}_L \cdot \psi_1 + b_L) \\ \vdots & \ddots & \vdots \\ h(\mathbf{w}_1 \cdot \psi_N + b_1) & \dots & h(\mathbf{w}_L \cdot \psi_N + b_L) \end{bmatrix}_{N \times L} \tag{6.4}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix} \text{ and } \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix} \tag{6.5}$$

The linear system (Eq. 6.3) can be solved using using *Moore-Penrose Pseudoinverse* [200].

$$\beta = \mathbf{H}^{\dagger}\mathbf{T} \tag{6.6}$$

where, $\mathbf{H}^{\dagger}$ is the *Moore-Penrose Pseudoinverse* of $\mathbf{H}$. $H^{\dagger} = (H^T H)^{-1} H^T$ when $H^T H$ is not singular, otherwise, $H^{\dagger} = H^T (HH^T)^{-1}$. A regularisation term (Ξ) can be added to achieve better performance in the ELM.

$$\beta = \left(\mathbf{H}^T \mathbf{H} + \frac{1}{\Xi} \right)^{-1} \mathbf{H}^T \mathbf{T} \tag{6.7}$$

The ELM network supports regular and non-regular activation functions (e.g., sigmoid, sine, cosine, etc.) [200]. In this case, the hyperbolic tangent sigmoid (tansig) is used as the primary activation function.

$$tansig = \frac{2}{(1 + exp(-2 * (\mathbf{w}_i \cdot \psi_j + b_i)))} - 1 \tag{6.8}$$

Finally, the 8-bits fixed-point quantisation is used to minimise the power consumption of the ELM as [209]. In this case, 8-bits quantisation uses integers rather than floating-point and integer math instead of floating-point math, which reduces the number of bits used for mathematical operations and storage. This compact representation allows us to reduce the requirements in terms of memory and computational resources, minimising power consumption.

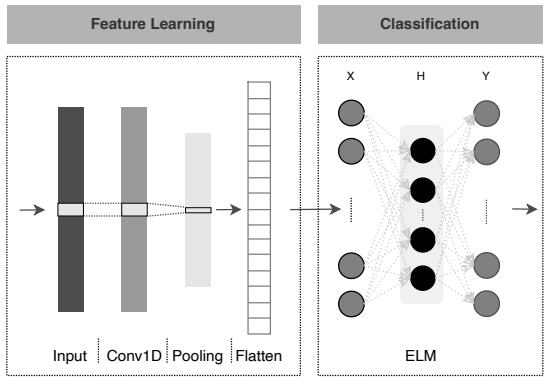


Figure 6.4 CNN-ELM model. Reproduced with the permission from [13].

Fig. 6.4 shows the proposed CNN-ELM model described in previous paragraphs. The left part represents the CNN model used for feature learning, and the right part shows the ELM model used for classification (building and floor).

6.3.1.4 CNN-ELM Indoor Localisation

Two fingerprinting schemes were introduced in previous chapters and sections. The first is a general schema, in which the matching algorithm determines the user position in the online phase (see Fig. 3.1). The second schema adds two components; the data transformation and the CNN-LSTM model for position estimation.

In contrast with the above-mentioned models, the new schema contains three processes: the data transformation, the CNN model and the ELM model. The training of the CNN and ELM models are carried out in the offline phase, whereas the data transformation is used in both phases. The evaluation using the CNN model, and the classification using the ELM model take place in the online phase. The output of this process is the classification of fingerprints into floor and building. In order to classify the fingerprints, the model uses the hidden layer output matrix (H) and the output weights (β) (see Fig. 6.5).

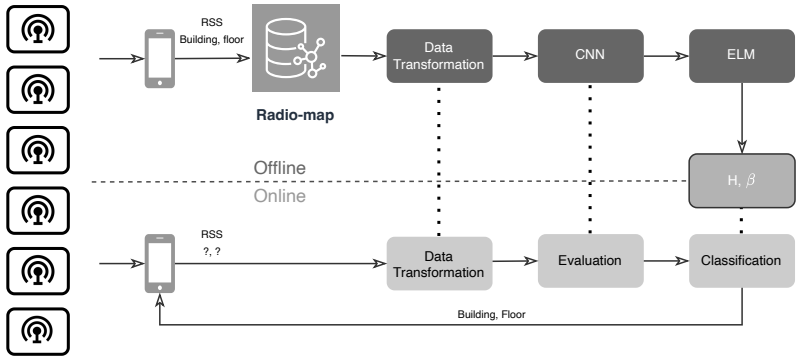


Figure 6.5 CNN-ELM indoor localisation schema.

6.3.2 Experiments and results

6.3.2.1 Experiments setup

The experiments were carried out using the same hardware, software, and baseline as in the previous experiments. The models were implemented in Python 3.9 using NumPy and Keras libraries and tested with 14 the public datasets described in Table 3.1. In this case, only multi-building and multi-floor datasets are selected to test the proposed CNN-ELM model.

The CNN-ELM model is compared against ELM, CNNLoc and 1-NN (baseline). The floor hit rate (ζ_f), building hit rate (ζ_f), training time ($\eth_{TR}$) and testing time ($\eth_{TE}$) are used to compare the performance of the proposed model.

In order to reduce the computational load, the CNN model is implemented using Keras backend in the same fashion as the implementation provided in [209].

The CNN-ELM is run by increasing the number of neurons in the hidden layer in intervals of 5 until reaching the number of features in the dataset where the model is evaluated. The process ends when the lowest localisation error is acquired. The regularisation term is selected from among four values: 1, 0.1, 0.5, 0.001 and 0.0001. These parameters provide the best results in terms of localisation. The optimal hyperparameter values for the ELM-based model in each dataset are provided in Table 6.2.

Table 6.2 Hyperparamter values for the ELM-based model.

Dataset	LIB1	LIB2	TUT1	TUT2	TUT3	TUT4	TUT5	TUT6	TUT7	UJI1	UJI2	UTS1	TIE1	SAH1
neurons	105	105	75	160	235	275	195	450	200	530	215	275	5	230
Ξ	0.05	0.01	0.1	0.01	0.05	0.05	0.01	0.1	1	0.1	0.01	0.01	0.0001	0.001

6.3.2.2 Results

Table 6.3 CNN-ELM vs. AFARLS. Reproduced with the permission from [13].

Approach	Database	Parameters	ζ_b[%]	ζ_f[%]	$\eth_{TR}$[sec]	$\eth_{TE}$[sec]
AFARLS [207]	UJI 1	$L = 1000$	100%	95.41%	84.68	0.21
	TUT 3	$L = 1000$	–	94.18%	2.40	0.57
CNN-ELM	UJI 1	$L = 530$	100%	92.26%	0.26	0.03
	TUT 3	$L = 235$	–	93.27%	0.22	0.10

Firstly, the proposed model is tested against AFARLS [207] with only two datasets: UJI1 (UJIIndoorLoc dataset) and TUT3 (Tampere dataset), which are used by the authors to test their approaches. Overall, the CNN-ELM provides almost the same accuracy as AFARLS (less than 3% of difference), but with less hidden neurons and lower computational cost (see Table 6.3). In this case, we have to consider that the time provided by the authors [207] includes the time to predict the 2D position (x and y)

Table 6.4 Comparison: CNNLoc, ELM and CNN-ELM

	CNNLoc				ELM				CNN-ELM			
Dataset	$\bar{\zeta}_b[-]$	$\bar{\zeta}_f[-]$	$\bar{\eth}_{TR}[-]$	$\bar{\eth}_{TE}[-]$	$\bar{\zeta}_b[-]$	$\bar{\zeta}_f[-]$	$\bar{\eth}_{TR}[-]$	$\bar{\eth}_{TE}[-]$	$\bar{\zeta}_b[-]$	$\bar{\zeta}_f[-]$	$\bar{\eth}_{TR}[-]$	$\bar{\eth}_{TE}[-]$
LIB1	-	0.9974	1	0.8023	-	0.9977	0.0105	0.0662	-	1.0010	0.0897	0.1855
LIB2	-	1.0039	1	0.7579	-	1.0098	0.0119	0.0763	-	1.0141	0.0244	0.0764
TUT1	-	0.9841	1	1.1417	-	0.9909	0.0042	0.0716	-	1.0136	0.0066	0.0742
TUT2	-	1.2656	1	2.6492	-	1.2344	0.0120	0.3239	-	1.2657	0.0147	0.3879
TUT3	-	0.9707	1	0.1423	-	1.0174	0.0138	0.0193	-	1.0180	0.0156	0.0202
TUT4	-	0.9548	1	0.0455	-	0.9895	0.0022	0.0054	-	1.0060	0.0042	0.0056
TUT5	-	1.1094	1	0.7165	-	1.1037	0.0121	0.0704	-	1.1118	0.0201	0.0676
TUT6	-	0.9978	1	0.0419	-	0.9963	0.0033	0.0023	-	0.9955	0.0079	0.0048
TUT7	-	0.9631	1	0.0349	-	0.9836	0.0029	0.0020	-	0.9839	0.0052	0.0024
UJI1	1.0054	1.0841	1	0.0299	1.0073	0.9846	0.0007	0.0013	1.0082	1.0513	0.0010	0.0016
UJI2	1.0000	1.0104	1	0.0072	0.9996	1.0543	0.0005	0.0004	1.0000	1.0884	0.0011	0.0004
UTS1	-	0.9278	1	0.1317	-	1.0028	0.0011	0.0070	-	1.0278	0.0019	0.0149
TIE1	-	0.3667	1	0.4844	-	1.5333	0.0004	0.0432	-	1.3333	0.0008	0.0657
SAH1	-	1.0959	1	0.2867	-	1.4795	0.0003	0.0138	-	1.5480	0.0005	0.0146
Avg.	1.0027	0.9808	1	0.5194	1.0034	1.0984	0.0054	0.0502	1.0041	1.1042	0.0138	0.0658

Table 6.4 shows the results of CNNLoc, ELM and CNN-ELM tested with 14 datasets. The baseline for the training time is the time obtained with the CNNLoc, given that the *k*-NN does not have a training stage. In general, the time required to train the ELM model is more than 99% faster than CNNLoc, due to the fact that the ELM model does not use the traditional backpropagation or similar algorithms.

The results show that the floor hit rate acquired with the CNNLoc is lower than the baseline by almost 2% on average, meaning the floor hit rate is slightly higher than the baseline in only 6 of 14 datasets. The prediction or testing time has reduced by more almost 48% compared with the baseline in the online phase of fingerprinting.

In the case of the ELM model, the floor hit rate is almost 9.8% better than the baseline. The training time is more than 99% faster than the CNNLoc, and the testing time is 95% lower than the baseline.

The CNN-ELM provides the overall best-normalised floor hit rate, 10.04% better than the baseline and 12.4% better than CNNLoc. The building hit rate is practically the same in UJI1 and UJI2. The training time is approximately 98% lower than the CNNLoc, but is approximately 2.5 times higher than the ELM. The classification time of testing fingerprints is slightly lower than the ELM, but much better than the baseline and CNNLoc.

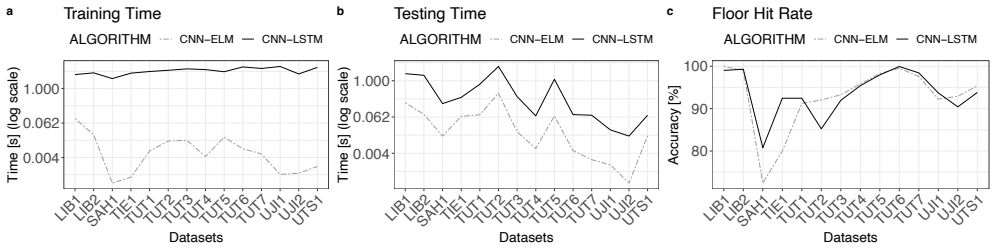


Figure 6.6 Comparing CNN-ELM and CNN-LSTM. (a) Training time, (b) Testing time and (c) Floor hit rate.

Fig. 6.6 shows the performance of the CNN-ELM and CNN-LSTM models in terms of classification accuracy, training and testing time. Fig. 6.6a shows the time required to train both models, where the x-axis represents the dataset and the y-axis the time in seconds (log scale). As can be observed from this figure, the CNN-ELM model needs less than 1 second to be trained; meanwhile, the CNN-LSTM requires several seconds or even minutes to be trained. For instance, to the train the CNN-ELM with UJI1 dateset takes 0.24 s and the CNN-LSTM needs 1669.6 s. Correspondingly, the time employed by the CNN-LSTM is much higher than the CNN-ELM in all cases (Fig. 6.6b). However, the CNN-LSTM is more accurate than the CNN-ELM model by more than 1% on average (see Fig. 6.6c). In spite of the low complexity of the CNN-ELM, it is more accurate than the CNN-LSTM in 50% of datasets.

6.4 Discussion

The use of ML models has gained widespread popularity in the field of indoor positioning. Currently, architectures like DNN and RNN are often used to extract meaningful information from the datasets, learning patterns, and outlier detection. Only a few researchers, however, have provided a comprehensive analysis of their model with multiple and heterogeneous datasets.

This chapter introduced two models for positioning and localisation, which were extensively tested in multiple datasets. The first model combined CNN and LSTM architecture to estimate the mean 2D and 3D position and classify the fingerprints into building and floor. This model was compared against the baseline (k-NN with simple configuration) and the CNNLoc model. Overall, the proposed CNN-LSTM

is better than the CNNLoc, but its accuracy is still lower than the baseline (average results). We must, however, consider that *k*-NN is not computationally efficient in the online phase of fingerprinting, given that the algorithm compares the incoming fingerprint with all samples in the radio map, which is time-consuming. Conversely, both the CNNLoc and CNN-LSTM are trained in the offline phase of fingerprinting, with the prediction being carried out in the online phase, providing a fast response. It is essential when there are large datasets with hundreds and thousands of fingerprints.

It is well-known that one of the factors that can affect the performance of ML models is the amount of data used to train the models. Fingerprinting is not the exception to this premise. In fingerprinting is essential to have enough samples to train the ML models used to estimate the user/device position to provide high levels of accuracy. It is important to emphasise that the experiments were carried out using heterogeneous datasets, allowing us to provide an exhaustive performance analysis of the proposed indoor positioning model. Table 6.1 shows that the CNN-LSTM model performs better with professional datasets (i.e., datasets collected systematically). However, the error obtained with crowdsourced datasets is acceptable, taking into account that those datasets are considered imbalanced datasets (i.e., not an equal number of samples per reference point and not an equal number of samples per building and floor).

The second ML model introduced in this chapter combines DNN and ELM in order to provide a fast and accurate localisation estimation. In contrast with the previous model, the time required to train the CNN-ELM is much less than the CNNLoc and CNN-LSTM because the CNN model is implemented using a low-level library, and the output weights of the ELM model are determined analytically. Additionally, the CNN and ELM models are less complex than the CNN-LSTM and CNNLoc (i.e., less number of layers and neurons). Although the CNN-ELM is a lightweight algorithm, its classification performance is comparable to complex models like AFARLS and CNNLoc.

As can be observed from Tables 6.1 and 6.4, the models based on CNNs are, in general, more efficient in solving classification than regression problems in the case of fingerprinting. Overall, the hitting rate is higher than *k*-NN, but the position accuracy is lower than the baseline. There are some factors that may affect to a large extent the accuracy of regression models using DNN, but with less impact on the

performance of the classification models and also in algorithms like k-NN. Some of them are listed below.

- The number of fingerprints per reference point.
- Distribution of reference points in the venue (e.g., the distance between samples).
- Complexity of the physical environment.
- The existence of identical reference points on different floors may lead to an increase in positioning error.

It is essential to highlight that data representation plays a fundamental role in acquiring high levels of accuracy. Consequently, we changed from the original data representation of the dataset to powed data representation in order to reduce the fluctuations in the RSS values. As a result, the ML models introduced in this chapter can efficiently extract meaningful information from datasets.

6.5 Summary

The key findings of this chapter are summarised in the following paragraphs.

- The proposed model based on CNN-LSTM for positioning has proven its flexibility to work with heterogeneous datasets from different environments and positioning technologies (e.g., Wi-Fi and BLE). It has provided better performance than some DNN-based indoor positioning solutions proposed in the literature. e.g., the mean 3D positioning error of CNN-LSTM is approx. 18% less than the CNNLoc on average.
- The proposed model based on CNN-ELM for building and floor classification is lightweight. It is an 8.8% more accurate than k-NN, 13.4% than CNNLoc and $\approx$ 1% than ELM. Along with the improvement in the localisation estimation, the training and testing time was faster than k-NN and CNNLoc, although slightly slower than ELM. The proposed CNN-ELM model provides an excellent trade-off among all evaluation metrics.

This chapter has extended the results of the proposed positioning model based on CNN-LSTM that was published in [180], and the proposed CNN-ELM model for Multi-building and Multi-floor classification that was published in [13].

7 CLOUD-BASED INDOOR POSITIONING PLATFORM

This chapter describes the architecture of the proposed open-source indoor positioning platform. This platform exploits the benefits of the algorithms and methods described in previous chapters in order to provide scalable, resilient software with a high fault tolerance. The chapter details the design, protocols, standards, and services provided by the proposed solution.

This chapter covers the following points:

- Key considerations in the design and development of the proposed indoor positioning platform.
- Details of the methodology and architecture followed when developing the platform.
- Software architecture.

7.1 Introduction

The increased demand for positioning and localisation services in IoT and wearable devices requires the provision of a new software architecture which fulfils all the requirements of scalable and robust applications. Indoor positioning and localisation applications with some of these features have been developed by researchers such as Mpeis, Roussel, Kumar, Costa, LaoudiasDenis, Capot-Ray, and Zeinalipour-Yazti [15], who proposed a new IoT localisation architecture for smart environments, enabling positioning and localisation for IoT devices and ultimately acquiring an accuracy of 2 m approximately.

Anyplace is not the only open-source platform for indoor positioning; there are also other indoor positioning applications or frameworks such as FIND [210] and indoorlocation [211]. These platforms support most of the standard technologies

for indoor positioning, e.g., Wi-Fi and BLE. Although each platform provides multiple services, their components or services cannot be deployed independently as microservices without refactoring the source code; a serious limitation in terms of scalability. In contrast with the solutions mentioned above, we propose an indoor positioning platform based on microservices to offer a scalable solution.

In order to provide a flexible, reusable, and scalable cloud-based indoor positioning platform, we propose a new architecture for indoor positioning and location platforms based on microservices. This architecture is designed with reference to the guidelines provided in the available standards for indoor positioning, mapping, and software architecture. This platform is designed to have independent services which can be deployed and used with other systems. For instance, some components can be reused in contact-tracing applications, autonomous navigation, and indoor parking, among others. It will reduce the development and deployment time of other systems in a similar way to the services provided by open-source platforms. The platform also takes into account standards, protocols, data pre-processing, accuracy, and positioning technologies.

7.2 Indoor Positioning Platform – Main considerations

In order to develop a resilient open-source indoor positioning platform, we focused on five key considerations that are included in the proposed platform:

7.2.1 Standardisation

IPSs have been widely studied within industry and academia over the last decade, however, just a few authors have touched upon which, if any, standards were followed in the development of IPSs. The failure to use standards becomes a challenge when it is necessary to implement those platforms in multiple environments or to integrate them with other systems such as eHealth applications, tourism systems, or smart parking.

7.2.2 Scalability

The proposed indoor positioning platform needs to be modular, and each component (microservice) must scale on demand. This is especially crucial when hundreds

of thousands of devices are simultaneously consuming positioning and localisation services. The proposed platform is designed to support vertical and horizontal scalability. Regarding vertical scalability, this platform can be deployed in low and high-profile devices such as servers or small IoT devices. Horizontal is more complex than vertical scalability, as the proposed platform must be able to split the workload between different infrastructures.

7.2.3 Portability

Portability is important as it enables systems or applications to be deployed on multiple devices, servers, or computers without losing functionality. The applications can thus be programmed once and work everywhere, as is the case with microservices and APIs, which can also be consumed and deployed everywhere (inside and outside the organisation). The microservices in the proposed platform enable it to work independently of other services; they can be installed in multiple computing paradigms without altering the behaviour of the platform.

7.2.4 Usage experience

In general, to provide an optimum user experience, systems must be intuitive, user-friendly (i.e., interface/services well designed and easy to understand), and provide QoS. The design of this platform takes into account the usage experience in the design of the services and documentation.

7.2.5 Privacy & Security

During the design of this platform, we took into consideration current issues regarding security & privacy; we included authorisation and authentication mechanisms to access the platform. In order to authenticate users, we implemented the authentication mechanism provided in Appwrite (user verification and authentication using an email account). The access to services provided by the proposed platform is given by two predefined roles (administrator and general user). This platform uses the SSL protocol to add it a security & privacy layer to the communication between the user and the application. These security mechanisms reduce security risks such as unauthorised access to the platform and ensure the privacy of the data managed by the platform.

7.3 Software Development Methodology

The first step in developing the proposed platform was determining the "best" software development methodology for use throughout the process. This step is crucial to developing well-tested software which has a high fault tolerance. During this process, each component goes through the six stages of the development process, to avoid errors in the final version. The methodology selected was DevOps Software Development Model [212], which is relatively new in comparison with waterfall or agile methodology, and allows rapid development of software and more reliability.

7.4 Architecture

7.4.1 Microservices & Clean Architecture

Microservices architecture has been adopted by multiple companies such as Netflix, Uber, eBay, Amazon [213]. These companies are characterised by their large number of users around the world, and the consequent millions of requests received per second. Here, infrastructure and software are designed in such a way as to support the traffic without affecting the service provided to the end-user. Microservices are thus used to offer scalable platforms with high availability.

In this microservice architecture pattern, it is important to split the software into small services which can interact with other services [214]. The failure of a single component will only affect one service, with all other services remaining intact. This will allow the developers to quickly determine the source of any problems. Additionally, each microservice can be deployed using Docker containers in servers (local or cloud) and embedded devices.

Fig. 7.1 shows a general schema of the microservices used in the proposed platform. These microservices are divided into three primary categories: experience, process, and core microservices. "Experience" microservices are the services consumed by the applications (e.g., web, mobile, and/or desktop applications) through the API gateway. "Process" microservices are the intermediaries between the core and the experience microservices, and can link more than one core service. Finally, "core" microservices are specialised services that coordinate data from the datasets. Each microservice can be connected to a specific database, or different core microservices

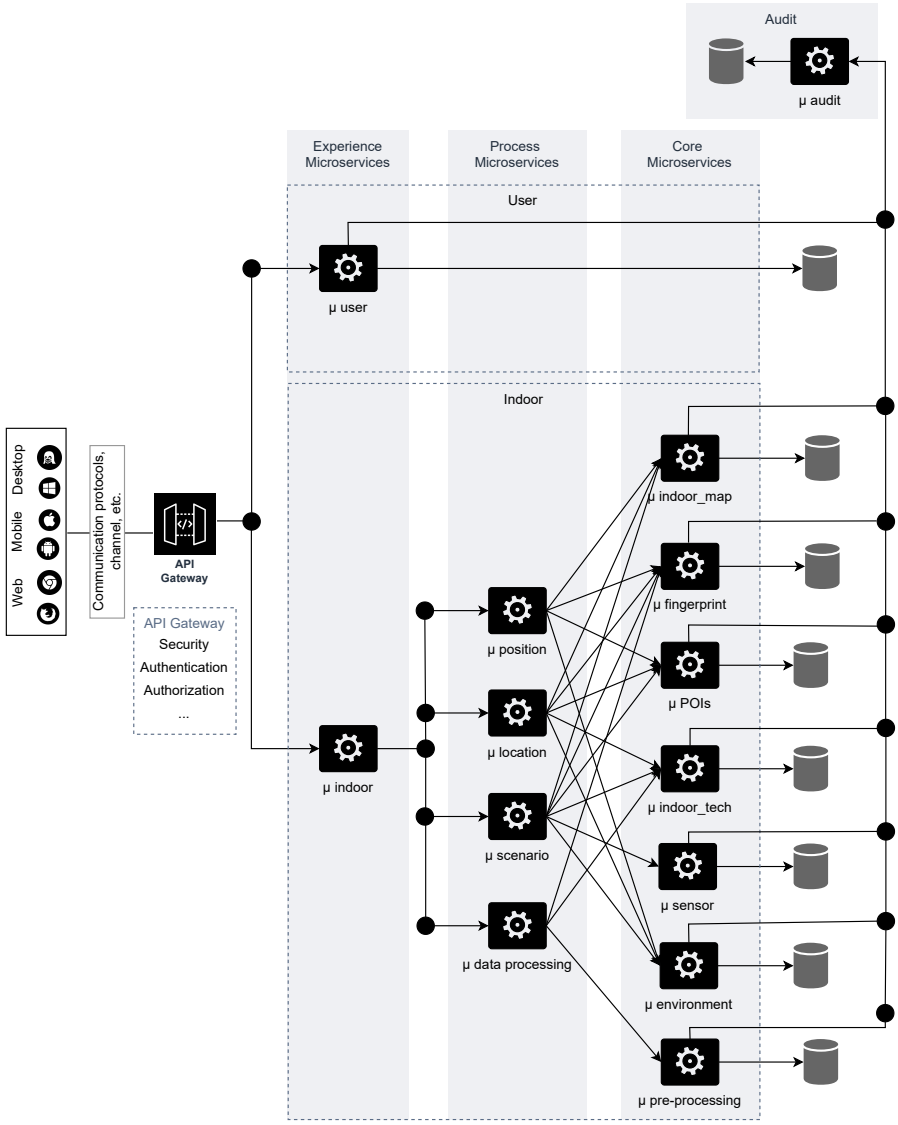


Figure 7.1 Cloud-based Indoor Positioning Platform – Architecture.

can share one database. Additionally, there is one microservice in charge of collecting logs (audit logs) generated for each core microservice.

Additionally, we implemented clean architecture in each microservice in order to have well-organised, testable, and maintainable software. The clean architecture was suggested by Robert C. Martin, with the idea of integrating the prime characteristics of different software architecture patterns (e.g., hexagonal architecture) into a

single architecture. This architecture pattern is composed of four "layers" or circles (although additional "layers" can be added according to the software requirements), governed by the dependency rule, which states that dependencies can go only from the external layer to the internal layer and not vice versa.

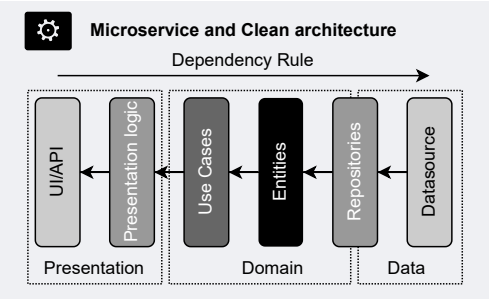


Figure 7.2 Structure of each microservice.

Fig. 7.2 shows the general schema used to develop the microservices. Each microservice has three primary layers: the presentation layer, which is composed of User Interface (UI) in the case of the frontend and API endpoint in the case of the microservice; the domain layer, which is composed of the "uses cases", the entities and repositories; and the data layer which contains the implementation of the repositories of the domain layer and data sources (e.g., databases, sensor, API, etc.).

Fig. 7.3 shows the structure of microservice directory. It follows the structure of clean architecture described above. Each microservice contains four key packages. Firstly, the core package where decorators and exceptions are developed. The second package contains the data sources, models, and the implementation of the repositories defined in the domain layer. In the domain package, we can find the entities (i.e., the class or classes of the data used in the microservice), the repositories (abstract classes) that define the methods used in the microservice, and the "use cases", as the name implies those are the use cases of the microservice, e.g., create an environment, update the user's information, etc. Finally, the presentation package contains the dependency injection (dependency rule) and the endpoints in the case of the backend (microservices). Additionally, certain microservices incorporate two packages: "algorithm" and "script". The package "algorithm", as its name suggests, contains some of the algorithms or models developed through this thesis, and the script package contains general functions to verify the state of the database and tables.

```
.
├── application                              # Contains the microservice code
│   ├── algorithms                           # * algorithms used in the ms
│   ├── core                                 # Util resources
│   │   ├── decorators
│   │   │   └── jwt_managet.py               # API authetication using JWT
│   │   ├── exceptions
│   │   │   ├── exceptions.py                # API exceptions
│   │   │   └── collection_exceptions.py     # DB - collections exception
│   ├── data                                 # Data layer
│   │   ├── datasource
│   │   │   ├── datasource.py                # Datasource abstract class
│   │   │   └── datasource_impl.py           # Datasource Implementation
│   │   ├── model
│   │   │   └── <name>_model.py              # Model's name
│   │   ├── repository
│   │   │   └── repository_impl.py           # Implementation repository
│   ├── domain                               # Domain layer
│   │   ├── entity
│   │   │   └── <name>_entity.py             # Entity
│   │   ├── repository
│   │   │   └── <name>_repository.py         # Model's name
│   │   ├── use_cases                        # Specific use cases
│   │   │   ├── create_use_case.py
│   │   │   ├── get_use_case.py
│   │   │   ├── delete_use_case.py
│   │   │   └── update_use_case.py
│   ├── presentation                         # Presentation layer
│   │   ├── data_injection
│   │   │   └── injection_container.py
│   │   ├── endpoints                        # API endpoints
│   │   │   └── <name>_endpoint.py
│   │   ├── req_body                         # Request body
│   │   │   └── <name>_body.py
│   ├── scripts                              # General scripts (DB initializations, etc.)
├── .env                                     # Environement setup
├── .flaskenv                                # Flask environment variable
├── confing.py                               # Configuration file
├── docker-compose.yml                       # YAML file to configure the application's services
├── Dockerfile                               # Commands to assemble the image
├── requirements.txt                         # Requirements (Python libraries)
├── pyproject.toml                           # Pytest configuration
├── pytest.ini                               # Pytest ini
├── run.py                                   # Core file, run the microservice
└── README.md                                # Readme please
```

Figure 7.3 Microservice – directory structure using clean architecture.

7.4.2 Standards

This platform was developed and tested following the guidelines provided by the standard ISO/IEC 18305:2016, IEEE 42020-2019 - ISO/IEC/IEEE International Standard and IndoorGML OGC. The standard ISO/IEC 18305:2016 was used to test and evaluate the performance of the platform. For example, the performance of the algorithms suggested in previous chapters was measured using the mean error

(see Eq. 7.1) and the floor prediction probability (see Eq. 7.2), which are described in the standard. Similarly, we followed the standard to measure the latency of the system as well as its resilience.

$$\varepsilon = \frac{1}{N} \sum_{i=1}^{N} error_i \tag{7.1}$$

$$P_F = \frac{N_F}{N} \tag{7.2}$$

where ε is the mean error, N is the number of testing points, and $error_i$ represents the i-*th* error. P_F represents the floor probability detection, and N_F is the number of times that the floor was predicted correctly.

IEEE 42020-2019 - ISO/IEC/IEEE International Standard offers guidelines for proper implementation of software architecture, which can be used for enterprise applications, software, and services [117]. This standard helped us to establish the architecture elaboration plan, viewpoints, model kinds, view models, and descriptions.

Along with the standards mentioned, the proposed indoor positioning platform implements the IndoorGML, the OGC standard for indoor spatial information [215]. The proposed platform uses the terminology, representation of indoor space, and geo-tagging as provided by the standard.

7.4.3 Communication Protocols

Based on the analysis provided in Section 2.4.2, HTTP was selected to connect the client with the cloud-based indoor positioning server, and TLS protocol has been used to add a security layer to the communication.

7.4.4 Positioning Technologies

The proposed platform supports two indoor positioning technologies, Wi-Fi and BLE. These two technologies were selected because they are supported by most wearable devices. The platform is able to support additional technologies when further microservices are added.

7.4.5 Database

In this section, we describe the databases used to store all information from the indoor environment, users, maps, etc. As each microservice manages different types of data (e.g., geospatial information), we selected two open-source databases that can be installed on a variety of platforms and devices, such as Linux, Windows, etc. First, we chose RethinkDB, which is used for scalable real-time web applications and supports geospatial information [216]. Appwrite, which is an open-source backend server for mobile and web applications, is another database used with this positioning platform. Appwrite offers multiple services, including databases, user authentication, storage, functions, security & privacy, and geolocalisation that can be accessed through a set of APIs well-described in its documentation [217].

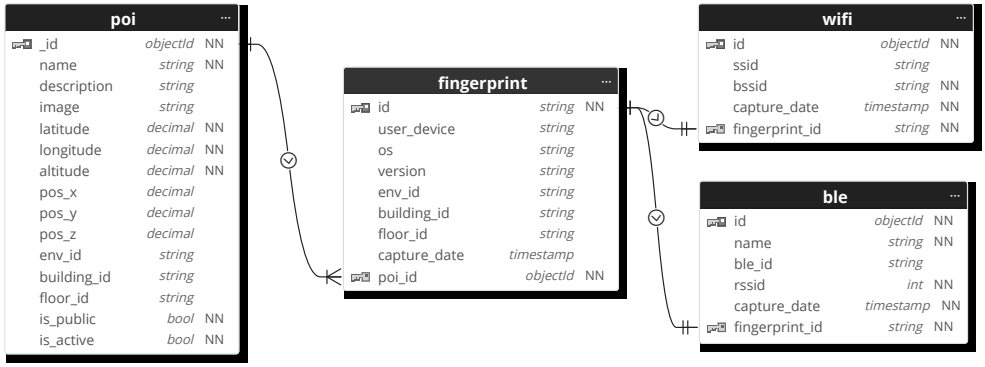


Figure 7.4 Example of the four data models used in the proposed indoor positioning platform.

Fig. 7.4 shows four data models (entities) defined to store the Point-of-Interests (POIs), fingerprints of Wi-Fi and BLE data (the remaining tables are included in Appendix A.2). Each entity contains the fields, primary key, type of data, and relationship with other entities. For instance, the "poi" entity has a one-to-many relationship with "fingerprint", and a one-to-one relationship with "Wi-Fi" and "BLE" entities.

7.4.6 Backend

7.4.6.1 Programming Language

The backend was developed using Python programming language, which is a high-level and interpreted programming language. Python was selected due to its flexibility, e.g., to develop web applications, for data analysis, data visualisation, task automation, and machine learning. Here, Python is used principally to develop the APIs, the machine learning models, and the algorithms described in the previous chapters. Along with Python, Sklearn, NumPy, Pandas, TensorFlow, and Flask were utilised.

7.4.6.2 APIs

The APIs allows the interaction between services, microservices, and systems. Each API developed with this platform belong to one microservice. For example, the position microservice contains an API whose endpoint is "*/api/v1/position/{env_id}*" (see Table 7.1). This API request the following information from device in order to return a position estimate: mobile phone model, OS version, list of RSS values from both Wi-Fi and BLE. In the case of Wi-Fi, the information contains the Service Set Identifier (SSID), Basic Service Set Identifier (BSSID), and Received Signal Strength Indicator (RSSI), whereas the BLE information consists of the device id, name and RSSI. The positioning API responds with the 3D position (latitude, longitude and altitude), the building and the floor.

Table 7.1 API – endpoints example.

Microservice	Request Method	Endpoint
Positioning	POST	/api/v1/position/{*env_id*}
Environment	POST	/api/v1/environment
	GET	/api/v1/environment/all
	GET	/api/v1/environment/{*env_id*}
	GET	/api/v1/environment/{*name*}/name
	...	
Building	POST	/api/v1/building/{*env_id*}
	GET	/api/v1/building/{*building_id*}
	GET	/api/v1/building/{*name*}/name
	DELETE	/api/v1/building/{*building_id*}/delete
	PUT	/api/v1/building/{*building_id*}/update

Table 7.1 shows an example of three of microservices developed in this indoor positioning platform (version 1). The first column contains the name of the microservice, then the request method (e.g., *POST*, *GET*, *DELETE* and *PUT*). The last column contains the endpoints. In general, the endpoints are composed of the word "api", API version "v1", the microservice name (e.g., position, environment, poi, etc.), and the id and request method. The full list of APIs included in the first version is in Appendix A.3.

In order to consume the APIs, the user must be authenticated using the JSON Web Token (JWT) generated by the Appwrite authentication service.

7.4.7 Documentation

The APIs in this platform are documented using OpenAPI Specification (OAS), which is defined as a standard for HTTP APIs. Users or computers can thus understand the services developed in this application without needing to access the source code. Additionally, the source code contains a *README.md* file per microservice; this *README.md* file contains the software requirements, specifications and the procedure to install and run each microservice.

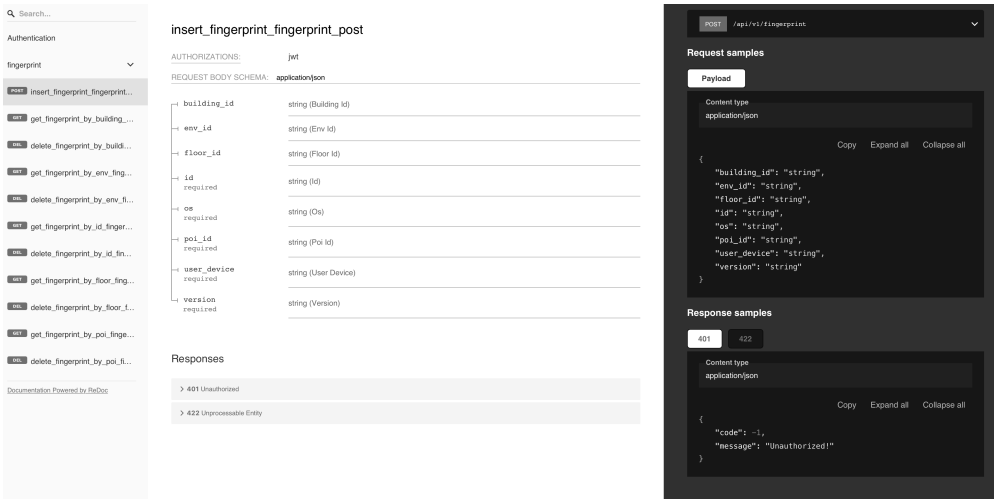


Figure 7.5 Fingerprint API.

Figure 7.5 shows the documentation of the "fingerprint" API using RapiDoc framework. This framework allows the developers and users to interact with API in a simple and interactive way. Through this API documentation, the developer can

access and test all the endpoints provided within the microservice. Moreover, this documentation contains "examples" with the data type and structure.

We also provide access to the software documentation and source code through a web application, namely *indoorSky.cloud*. This documentation contains guidance on deploying the platform (containers of each microservice), how to begin using APIs, how the microservices are structured (see Fig. 7.3), information on each microservice (i.e., their endpoints and how to consume their services), a glossary of terms, versions, release notes, access to the source code, and how to contribute to this open-source project (see Fig. 7.6).

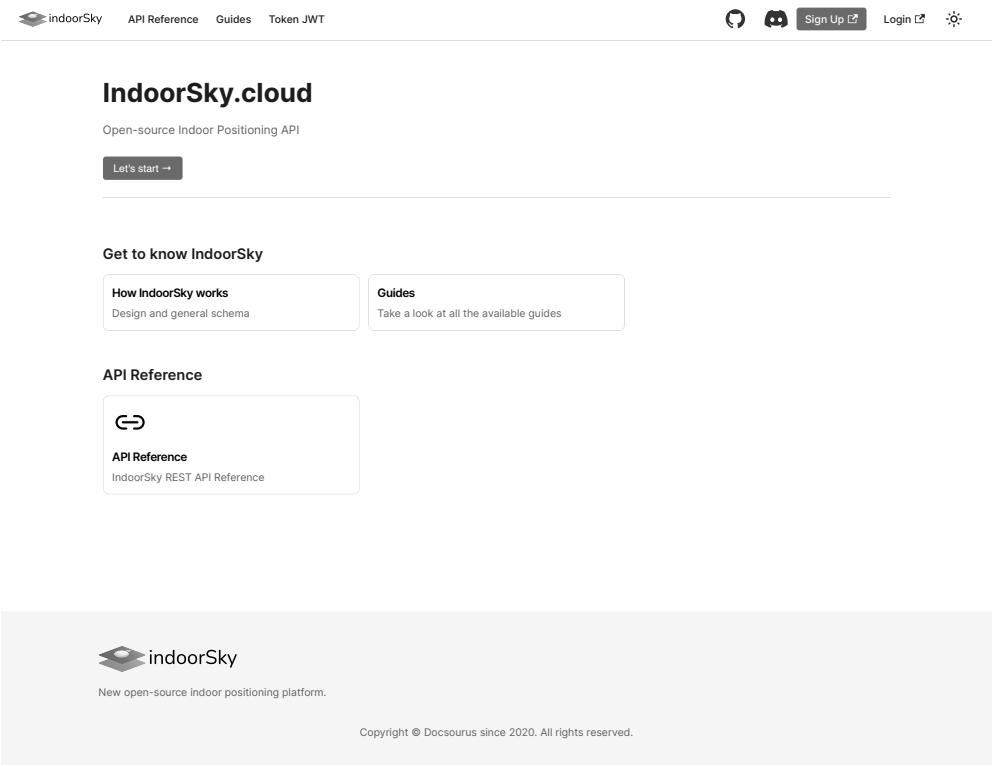


Figure 7.6 Web documentation.

7.4.8 Access

The source code and documentation are open-access and open-source and, therefore, they are publicly available through the GitHub repository [218] and website https://www.insky.cloud. This software is licensed under a Creative Commons Attribution

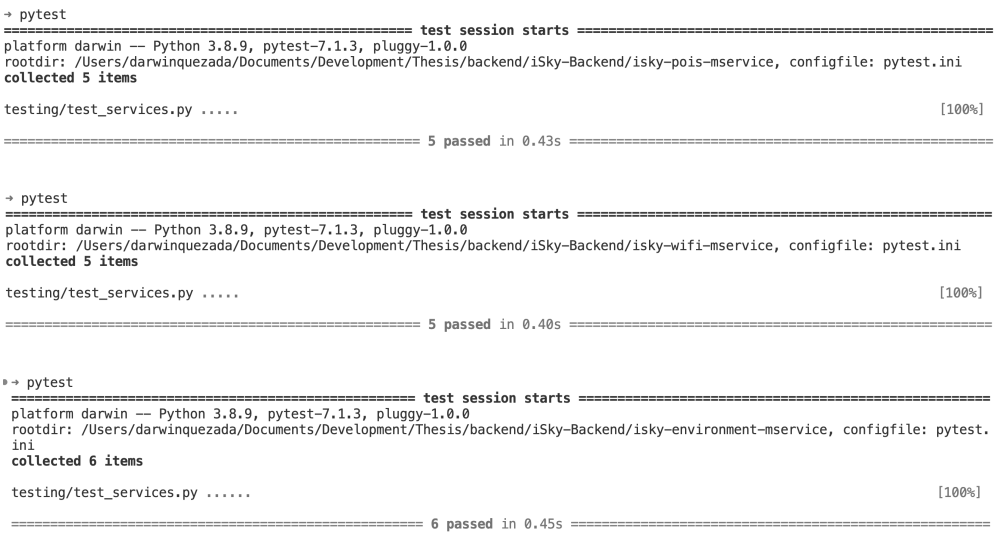


Figure 7.7 Results of functional testing using Pytest framework.

4.0 International License (CC BY 4.0), providing the maximum freedom to share and adapt it for any purpose.

7.5 Performance Analysis

In order to test the performance of the proposed cloud-based indoor positioning platform, the platform has been deployed in a Virtual Private Server (VPS) with the following characteristics: 6 vCPU Cores, 16 GB RAM and Ubuntu 20.04 OS. The performance of the software developed was evaluated in terms of latency (i.e., the time between the request and the response). The tool used to evaluate the performance is JMeter, which simulates a heavy load on the server or each microservice. Additionally, functional testing was carried out on each service using the Pytest framework.

Fig. 7.7 shows the results of running the functional tests in three microservices (POI, Wi-Fi and environment). These tests help to validate whether each service is working in accordance with the specifications. This figure also demonstrates that the endpoints of those microservices successfully passed the test. This test was carried out on each microservice offered within this positioning platform.

Fig. 7.8 shows the results of running 1000 requests to the cloud platform in a ramp-up period of 1 s. The x-axis shows the timestamp, and the y-axis the latency in

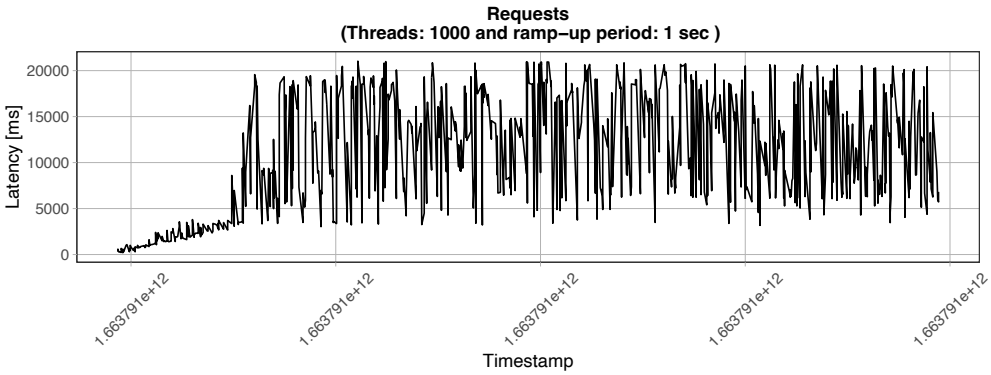


Figure 7.8 Latency request cloud-based indoor positioning platform.

milliseconds. The average latency is 11.455 s, maximum 21.028 s, minimum 128×10^{-3} s with a standard deviation of 6.222 s.

7.5.1 Empirical Test

This section provides a brief example of how the proposed microservices work. This example was carried out using the API web interface, as described in previous sections.

Firstly, the software provided contains an authentication API which uses the JWT generated by the authentication Appwrite backend. Fig. 7.9 shows the response of the authentication using the token. The response contains information about the user, such as id, email, name, phone, status, etc. The Appwrite backend provides this information and it is part of the API offered by it [217].

The token generated enables access to all the APIs provided with the positioning platform. Fig. 7.10 shows the field used for the authentication. In this platform, all the API web interfaces contain this authentication mechanism to avoid unauthorised access to these services.

Once we have access to the platform, we can create the environments where the IPS was deployed. Fig. 7.11 shows an example of two environments added to the system, UJI and Tampere University, respectively. Through the "environment" API can be administrated the environments (create, update, delete). The environments API endpoints are explained in the web documentation.

The next step is to add the buildings linked to the environment created in the previous step. The buildings belonging to the environments can be administrated

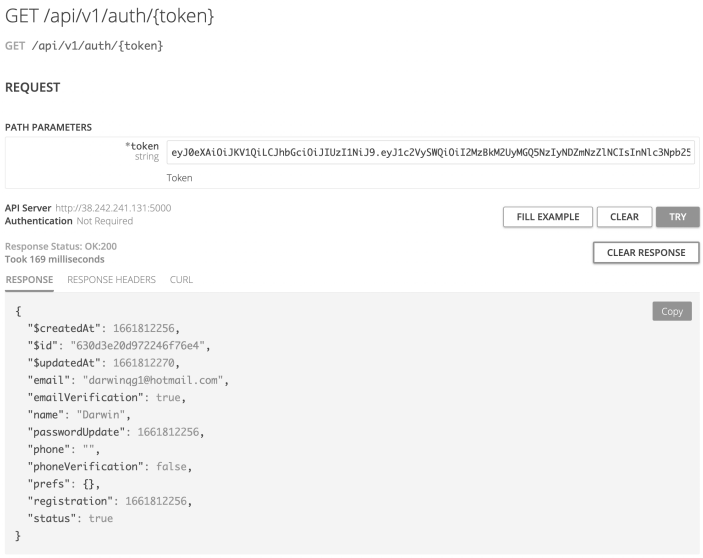


Figure 7.9 Get user information using the JWT.

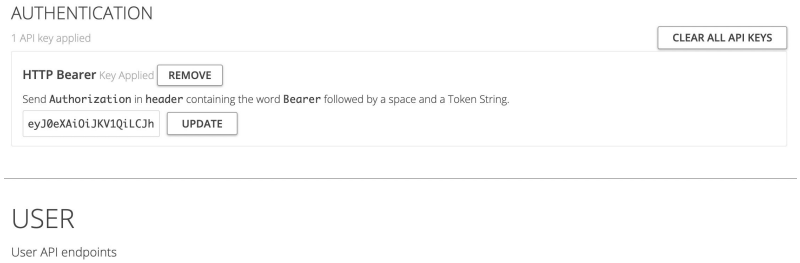


Figure 7.10 Authentication in each microservice using JWT.

through the "building" API provided with the software. Fig. 7.12 shows an example of how to add a new building to the environment. As can be observed from this figure, the "building" API is linked with the "environment" API through the "env_id" key.

Floors can be added to each building using the "floor" API (see Fig. 7.13). The "building" and floor APIs are linked through the "building_id". The user can determine if the floor is public or not, and also if it is currently enabled ("is_active").

The next step is to add the Point-of-Interest (POI) to the environment. The reference points required to build the radio map in the offline phase of fingerprinting are added to the database using this endpoint. Fig. 7.14 illustrates the parameters of

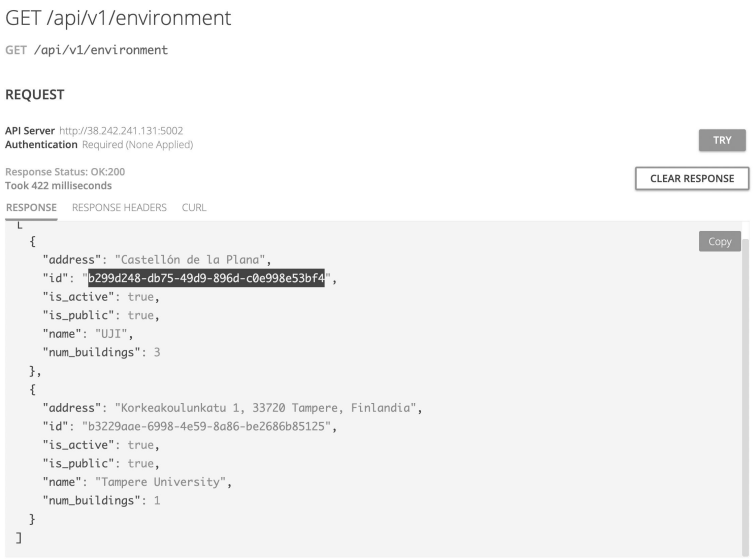


Figure 7.11 Environment API.

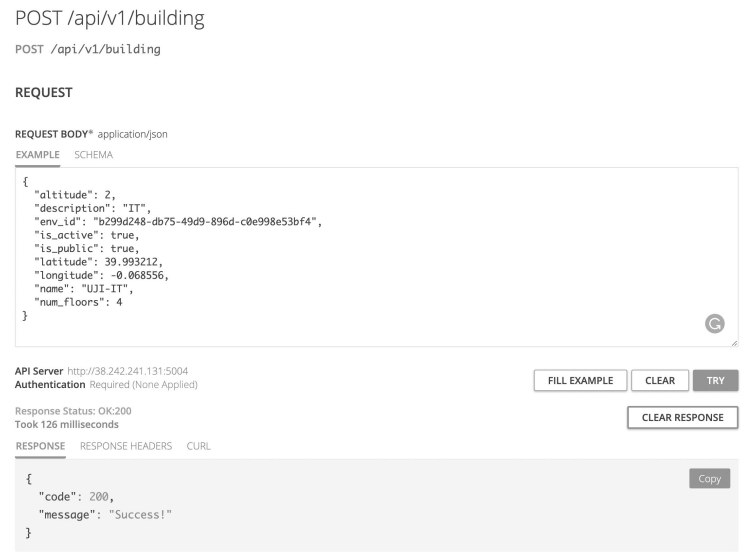


Figure 7.12 Building API.

this API, e.g., the POI's geographical position (latitude, longitude, altitude), name, description and local coordinates (x, y, z), and other fields.

In order to collect and manage the fingerprints, the proposed platform contains the "fingerprint" API (see Fig. 7.15), which allows us to add, remove, and retrieve

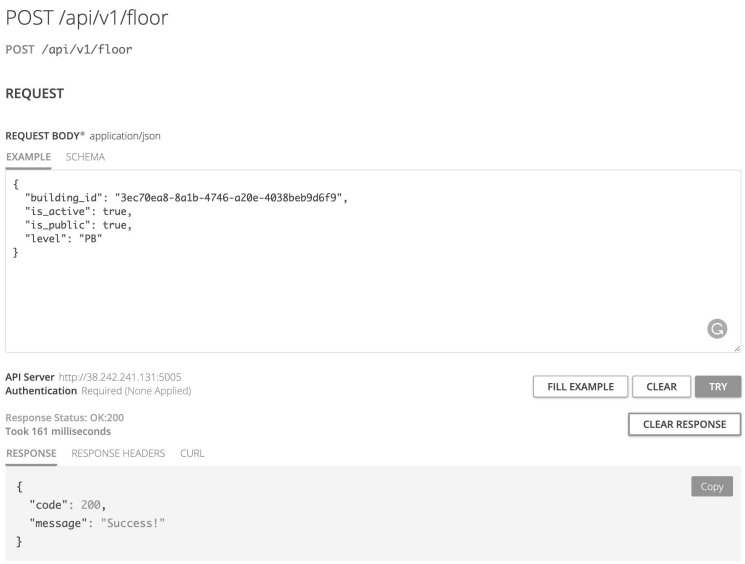


Figure 7.13 Floor API.

Figure 7.14 POI API.

fingerprints. The information stored in the database are the “env_id”, “building_id” (optional), “floor_id” (optional), device used to collect the fingerprints (user_device), “os” operation system and version.

This platform stores the fingerprints in different tables according to the technology.

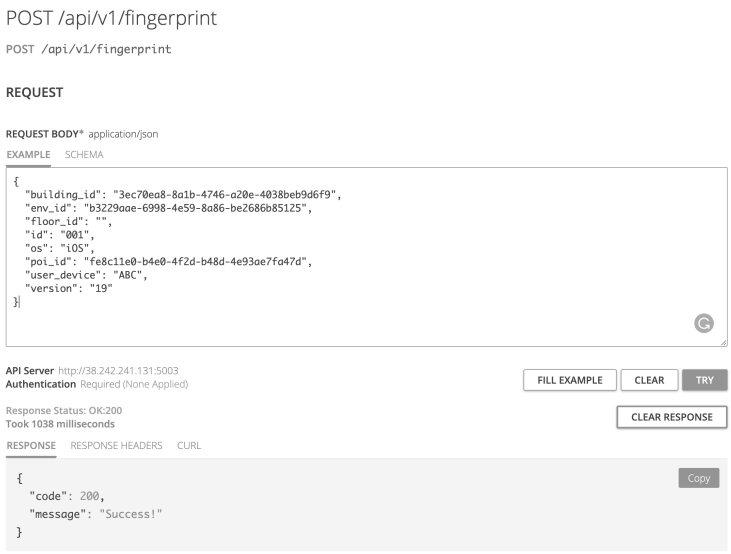


Figure 7.15 Fingerprint API.

For instance, the Wi-Fi fingerprints (RSS measurements) are stored in the wifi-fingerprint table. When the user wishes to add new technologies, this is just a matter of creating a new microservice, and the values collected will be stored in a new table. The tables are linked with the fingerprint table through the "fingerprint_id': we can thus collect fingerprints from different technologies (e.g., Zigbee, mobile networks, LoRaWAN, etc.). Fig. 7.16 show the Wi-Fi API used to manage the data collected of Wi-Fi technology.

Finally, we use the positioning API to estimate the user position. Example data was uploaded to the system to test the microservice (see Fig. 7.17).

This is just one example of the use of the services provided by the proposed indoor positioning platform. Details of all microservices and how to use them is contained in the source code and documentation.

7.6 Discussion

Microservices architecture allows us to easily and quickly deploy services that can scale according to software and user requirements. We can thus deploy these services on different devices or computing paradigms. Additionally, we can select only those services which are required to build a new application. For instance, we can use

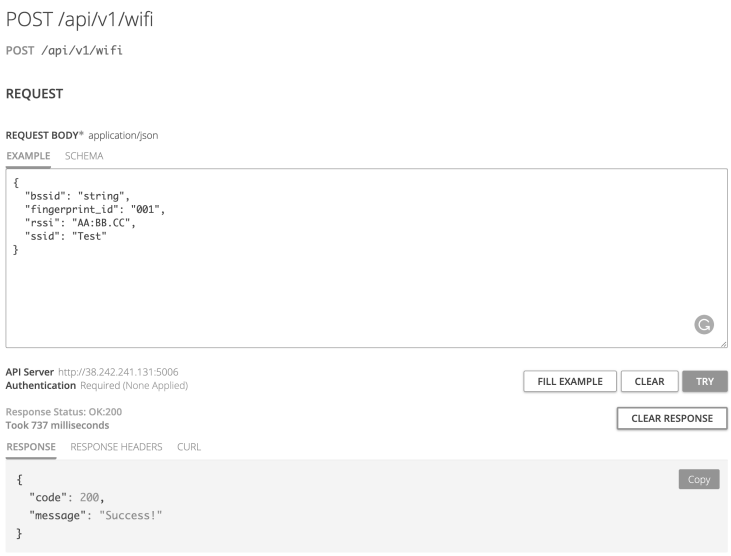


Figure 7.16 Wi-Fi fingerprint API.

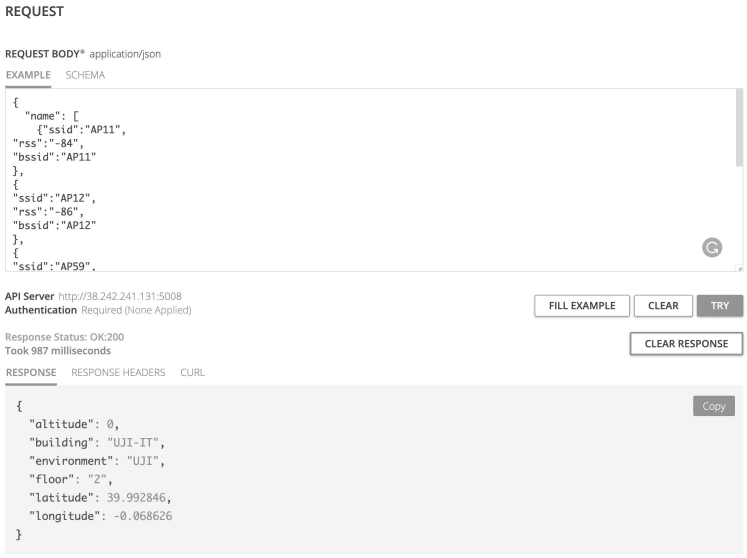


Figure 7.17 Positioning API.

only the POI microservice to display information points in a museum application. Microservices do, however, increase the complexity of software development due to the communication and integration layer that these services necessitate.

From the point of view of integrability in indoor positioning, having multiple

microservices may facilitate the integration of the multiple IPS/ILS through their services. For example, if developers wish to add further functionalities, it is simply a matter of adding new microservices, or consuming the services provided by another IPS. In such a way, the microservices of other positioning systems can be reused, reducing the developing time. Additionally, the use of microservices architecture in IPS allows for flexible and maintainable applications. In this dissertation, microservice architecture goes hand-in-hand with clean architecture, which allows for easy source code maintenance, as each layer in the microservice is well delineated and connected using the dependency injection.

In order to facilitate the integration and use of this positioning platform, a detailed documentation is provided via sources such as Rapidoc, and web documentation, meaning that the user (software developer) can easily understand how this platform was developed and how it can be integrated with other systems.

It is essential to highlight that the algorithms included in this platform and the software have been tested with multiple datasets and tools, ensuring the quality of the proposed platform. Despite the extensive testing, the chance remains that bugs may exist within the platform. In order to prevent any problems on this front, we have opened multiple channels for others to contribute to this project. For example, the web documentation contains links to Github and Slack, where any bugs or issues in the code can be reported. Through these channels and repositories, the community can download and pull requests.

7.7 Summary

This section summarises the key considerations in developing the positioning platform, as well as the results obtained with it.

- The proposed indoor positioning platform exploits the advantages of microservices architecture and clean architecture to provide a maintainable and testable system. The use of microservices enables the deployment of the services on different devices, as well as their integration with other microservices and systems with minimal effort. The platform was tested using functional testing, and by simulating heavy load to ensure the quality and proper operation of this indoor positioning platform.
- A special emphasis was placed on the documentation for this platform. The

web documentation developed explains the functionalities, structure, and APIs. Additionally, the APIs can be accessed through a web interface using RapiDoc, where the user can explore and test the services provided within each microservice.

The source code and documentation can be downloaded from [218]. The API and web documentation can be accessed through the following link: https://www.insky.cloud.

8 CONCLUSIONS AND FUTURE WORK

This section outlines the conclusions drawn from our results with reference to the research questions listed in Chapter 1.

8.1 Answering the research questions

Here, the answers to the research questions formulated in Chapter 1 are provided.

- *What are the current trends and challenges of cloud-based indoor positioning platforms?*

  New indoor positioning solutions, including applications and technologies, emerge every year, offering new services to the end-user, or addressing limitations of the current software. Chapter 2 thus consists of a systematic review of research papers published between January 2015 and May 2022. The review identifies current challenges and trends in cloud-based indoor positioning solutions. The challenges identified with overwhelming frequency during the review were related to software design and the consideration standards in current indoor positioning solutions. The review identified a clear trend of making use of the cloud or similar computing paradigms (e.g., EC, Mist, FC, etc.) to deploy indoor positioning/localisation/navigation systems, given their high computational and storage capabilities.

- *Can data pre-processing techniques enhance the quality of indoor positioning data?*

  In general, fingerprinting-based indoor positioning solutions require the building of a radio map for the estimation of user/device position during the operational phase. This radio map may contain random fluctuations produced by multiple factors such as NLOS and multipath that can affect the position estimation. It is essential to detect these undesirable fluctuations, as well as samples that do not contribute to an accurate position estimation. Chapter 4

shows the results of applying a data cleansing algorithm to the radio, which results in a reduction of the positioning error, processing time, and dimension of the dataset. Additionally, the data augmentation model reduced the position estimation in 8 of 11 datasets. The use of pre-processing techniques thus offered an optimisation of the data and an improvement in the performance of positioning algorithms.

- *Can the computational load in indoor positioning algorithms be reduced without significantly affecting the positioning accuracy?*

  In Chapter 5, we introduced two models to enhance the position accuracy and reduce the computational load. The first model suggested in Chapter 5 improved position estimation by post-processing the noisy samples detected by DBSCAN, at the cost of a slight increase in the time required to estimate position. Conversely, the second algorithm provided a reduction in the processing time but increased the positioning error. In the same fashion, Chapter 6 provides a localisation algorithm that outperforms the baseline in terms of position accuracy and processing time. This last algorithm offers a balance between computational load and accuracy.

- *Can machine learning models provide the flexibility and robustness needed to function in heterogeneous GNSS-denied scenarios?*

  The heterogeneity of indoor environments makes them a challenging scenario for many indoor positioning solutions, including ML-based indoor positioning models. Several extensively tested supervised and unsupervised ML models, such as k-NN, CNN, CNN-LSTM, ELM, CNN-ELM, and clustering algorithms have been proposed throughout this work. The evaluation was carried out with more than ten heterogeneous datasets obtained from different environments, showing the performance of each model in terms of positioning accuracy and computational time. Unsupervised models like k-Means offered better performance than other clustering algorithms in most of the datasets, joining near samples into clusters and with minimum errors. Supervised models such as CNN-ELM efficiently solved classification problems, reducing the error in the classification of fingerprints into building and floor in more than 80% of the datasets compared with the baseline.

- *Can standards focused on indoor positioning, indoor maps, and software help to enhance the integrability and robustness of IPS?*

  The use of standards allows systems to speak the same "language", facilitating integration between systems. The smooth integration between systems is one of the major advantages of providing systems which follow the guidelines published by entities like ISO and/or OGC. Additionally, the use of standards enhances the quality of indoor positioning systems, as well as their continuous improvement. The software developed with this thesis followed the standards mentioned in Chapter 2 and 7.

8.2 Impact of publications and supporting materials

This dissertation extends some of the articles accepted in journals [32] and conferences [190, 191, 174, 13, 180], including two papers awarded as best student and runner-up papers. The first award-winning paper was presented at ICUMT 2020 conference, where it won best student paper; it was extended in Chapter 5. The second award-winning work is the SURIMI framework for data augmentation and indoor positioning, which was presented at the IPIN conference 2022, where it was designated runner-up best paper (see Chapters 4 and 6).

The algorithms provided in each publication were widely tested with multiple public datasets in order to provide a complete analysis of the advantages and drawbacks of each model or algorithm. Additionally, the source code is publicly available under the CC BY 4.0 license.

In addition to the publications mentioned above, I have collaborated on various works related to this Thesis. Although those works have not been explicitly described in this Thesis,

- Potortì, Torres-Sospedra, Quezada-Gaibor, Jiménez, Seco, Pérez-Navarro, Ortiz, Zhu, Renaudin, Ichikari, Shimomura, Ohta, Nagae, Kurata, Wei, Ji, Zhang, Kram, Stahlke, Mutschler, Crivello, Barsocchi, Girolami, Palumbo, Chen, Wu, Li, Yu, Xu, Huang, Liu, Kuang, Niu, Yoshida, Nagata, Fukushima, Fukatani, Hayashida, Asai, Urano, Ge, Lee, Fang, Jie, Young, Chien, Yu, Ma, Wu, Zhang, Wang, Fan, Poslad, Selviah, Wang, Yuan, Yonamoto, Yamaguchi, Kaichi, Zhou, Liu, Gu, Yang, Wu, Xie, Huang, Zheng, Peng, Jin, Wang, Luo, Xiong, Bao, Zhang, Zhao, Yu, Hung, Antsfeld, Chidlovskii, Jiang, Xia,

Yan, Li, Dong, Silva, Pendão, Meneses, Nicolau, Costa, Moreira, De Cock, Plets, Opiela, Džama, Zhang, Li, Chen, Liu, Yean, Lim, Teo, Lee, and Oh [219] presents the results for the IPIN 2020 Indoor Localisation Competition. Our contribution related chiefly to data collection and evaluation in Track 3 about smartphone-based indoor positioning. In this Thesis, we took an eye on the sensing technologies, data format, suggested models, accuracy and real-time restrictions in the competition when designing the platform provided in Chapter 7.

- Klus, Quezada-Gaibor, Torres-Sospedra, Lohan, Granell, and Nurmi [220] exploits the advantages of combining multiple machine learning models into a novel cascade algorithm for regression and classification. The proposed model was tested in fourteen public datasets which are also used in this dissertation.
- Torres-Sospedra, Silva, Klus, Quezada-Gaibor, Crivello, Barsocchi, Pendão, Lohan, Nurmi, and Moreira [221] defines the aggregated metrics for those cases where the comparison of indoor positioning systems involves a set of diverse datasets. In this Thesis, we have adopted the proposed way to provide normalised values, easing the comparison of different models as shown in Chapters 4–6.
- Torres-Sospedra, Aranda, Álvarez, Quezada-Gaibor, Silva, Pendão, and Moreira [222] took advantage of the noise present in the RSS to improve the positioning accuracy. In this Thesis, we have worked to optimize data and reduce that noise in Chapter 4.
- In Furfari, Crivello, Baronti, Barsocchi, Girolami, Palumbo, Quezada-Gaibor, Mendoza Silva, and Torres-Sospedra [223] provides a novel approach for indoor localisation that enables cooperation between heterogeneous localisation solutions. In this research work, the authors explained in detail the proposed architecture, components and services. The solution proposed by the authors inspired the development of the cloud-based indoor position platform introduced in Chapter 7.
- Torres-Sospedra, Quezada-Gaibor, Mendoza-Silva, Nurmi, Koucheryavy, and Huerta [224] defines the three new approaches to deal with clusters generated with *k*-Means. In this Thesis, as a follow-up to that work, we have worked on two clustering models in Chapter 5.

8.3 Future Work

Although this dissertation has provided algorithms to circumvent some of the current challenges or gaps in indoor positioning (e.g., data preprocessing, computational load and software design), there are still numerous challenges in indoor positioning that could not be addressed in this research. For example, challenges related to security & privacy (partially addressed in this dissertation, using authentication and SSL protocol to ensure the communication between the user and the cloud).

There is room to research new methods to offer a seamless transition between indoor and outdoor positioning services. Additionally, it is important to research new ways to combine current indoor positioning technologies in order to improve position accuracy. Although these are not new topics in the field, a definitive solution is yet to be found.

REFERENCES

[1] Luca De Nardis, Giuseppe Caso, and Maria Gabriella Di Benedetto. "ThingsLocate: A ThingSpeak-Based Indoor Positioning Platform for Academic Research on Location-Aware Internet of Things". In: *Technologies* 7.3 (2019). ISSN: 2227-7080. DOI: 10.3390/technologies7030050. URL: https://www.mdpi.com/2227-7080/7/3/50.

[2] David Martín, Carlos Lamsfus, and Aurkene Alzua-Sorzabal. "A cloud-based platform to develop context-aware mobile applications by domain experts". In: *Computer Standards & Interfaces* 44 (2016), pp. 177–184.

[3] S. Horsmanheimo, S. Lembo, L. Tuomimaki, S. Huilla, P. Honkamaa, M. Laukkanen, and P. Kemppi. "Indoor Positioning Platform to Support 5G Location Based Services". In: *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*. 2019, pp. 1–6.

[4] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao. "Application of Machine Learning in Wireless Networks: Key Techniques and Open Issues". In: *IEEE Communications Surveys Tutorials* 21.4 (2019), pp. 3072–3108. DOI: 10.1109/COMST.2019.2924243.

[5] P Konstantopoulos, E G M Petrakis, and S Sotiriadis. "INaaS: Indoors navigation as a service on the cloud and smartphone application". In: *2018 IEEE 39th Sarnoff Symposium, Sarnoff 2018*. 2018.

[6] P. Álvarez and N. Hernández and Fco. Javier Fabra and M. Ocaña. "A cloud-based parallel system for locating customers in indoor malls". In: *IPIN*. 2019.

[7] Dario Facchinetti, Giuseppe Psaila, and Patrizia Scandurra. "Mobile cloud computing for indoor emergency response: the IPSOS assistant case study". In: *Journal of Reliable Intelligent Environments* 5 (Sept. 2019).

[8] M. Fazio, A. Buzachis, A. Galletta, A. Celesti, and M. Villari. "A proximity-based indoor navigation system tackling the COVID-19 social distancing measures". In: *2020 IEEE Symposium on Computers and Communications (ISCC)*. 2020, pp. 1–6.

[9] System CISCO. *Cisco Annual Internet Report (2018–2023)*. 2018. URL: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf.

[10] A. Barua, C. Dong, F. Al-Turjman, and X. Yang. "Edge Computing-Based Localization Technique to Detecting Behavior of Dementia". In: *IEEE Access* 8 (2020), pp. 82108–82119. DOI: 10.1109/ACCESS.2020.2988935.

[11] J. L. Carrera V., Z. Zhao, M. Wenger, and T. Braun. "MEC-based UWB Indoor Tracking System". In: (2019), pp. 138–145.

[12] Xiangyu Liu, Lei Guo, Helin Yang, and Xuetao Wei. "Visible Light Positioning Based on Collaborative LEDs and Edge Computing". In: *IEEE Transactions on Computational Social Systems* (2021), pp. 1–12. DOI: 10.1109/TCSS.2021.3109631.

[13] Darwin Quezada-Gaibor, Joaquín Torres-Sospedra, Jari Nurmi, Yevgeni Koucheryavy, and Joaquín Huerta. "Lightweight Hybrid CNN-ELM Model for Multi-building and Multi-floor Classification". In: *2022 International Conference on Localization and GNSS (ICL-GNSS)*. 2022, pp. 01–06. DOI: 10.1109/ICL-GNSS54081.2022.9797021.

[14] GmbH indoo.rs®. *Indoo.rs*. 2022. URL: https://indoo.rs.

[15] Paschalis Mpeis, Thierry Roussel, Manish Kumar, Constantinos Costa, Christos LaoudiasDenis, Denis Capot-Ray, and Demetrios Zeinalipour-Yazti. "The Anyplace 4.0 IoT Localization Architecture". In: *2020 21st IEEE International Conference on Mobile Data Management (MDM)*. 2020, pp. 218–225. DOI: 10.1109/MDM48529.2020.00045.

[16] W. Li, Z. Su, R. Li, K. Zhang, and Q. Xu. "Abnormal Crowd Traffic Detection for Crowdsourced Indoor Positioning in Heterogeneous Communications Networks". In: *IEEE Transactions on Network Science and Engineering* 7.4 (2020), pp. 2494–2505.

[17] C. Yi, W. Choi, L. Liu, and Y. Jeon. "Cloud-Based Positioning Method with Visualized Signal Images". In: *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*. 2017, pp. 122–129.

[18] X. Zhang, Q. Chen, X. Peng, and X. Jiang. "Differential Privacy-Based Indoor Localization Privacy Protection in Edge Computing". In: *2019 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld / SCALCOM / UIC / ATC / CBDCom / IOP / SCI)*. 2019, pp. 491–496.

[19] Yucheng Guo, Rou You, G Yuchen, You Rou, Guo Yucheng, and You Rou. "A New Approach of Location and Navigation Services System Design In Complex Indoor Scenes Based on the Android Mobile Computing Platform". In: 2018-Septe (2017). Ed. by Yucheng, G. ISSN: 2379-3724.

[20] German Mendoza-Silva, Joaquin Torres-Sospedra, and Joaquin Huerta. "A Meta-Review of Indoor Positioning Systems". In: *SENSORS* 19.20 (Oct. 2019).

[21] Faheem Zafari, Athanasios Gkelias, and Kin K. Leung. "A Survey of Indoor Localization Systems and Technologies". In: *IEEE COMMUNICATIONS SURVEYS AND TUTORIALS* 21.3 (2019), 2568–2599.

[22] Chokatsu Yara, Yuta Noriduki, Shigenori Ioroi, and Hiroshi Tanaka. "Design and implementation of map system for indoor navigation-An example of an application of a platform which collects and provides indoor positions-". In: International Symposium on Inertial Sensors and Systems (2015), pp. 70–73. ISSN: 2377-3464.

[23] Vaclav Kaczmarczyk, Radek Kuchta, Zdenka Kuchtova, Jaroslav Kadlec, and Ondrej Bastan. "Data processing platform for indoor localization framework". In: *IFAC PAPERSONLINE* 51.6 (2018), pp. 508–513. ISSN: 2405-8963.

[24] Hyojeong Shin, Yohan Chon, Yungeun Kim, and Hojung Cha. "A Participatory Service Platform for Indoor Location-Based Services". In: *IEEE PERVASIVE COMPUTING* 14.1 (2015), pp. 62–69. ISSN: 1536-1268.

[25] K M Anandkumar, A Krishnan, G Deepakraj, and N Nishanth. "Remote controlled human navigational assistance for the blind using intelligent computing". In: (2017).

[26] T T Lu, Y J Liu, J J Ciou, and C H Lu. "Feature selection difference matching method for indoor positioning". In: (2017).

[27] Neil E. Klepeis, William C. Nelson, and John P. Robinson Wayne R. Ott. "The National Human Activity Pattern Survey (NHAPS): A resource for assessing exposure to environmental pollutants". In: *Journal of Exposure Analysis and Environmental Epidemiology* 11 (2001). DOI: {https://doi.org/10.1038/sj.jea.7500165}.

[28] Johannes Wichmann. "Indoor positioning systems in hospitals: A scoping review". In: *DIGITAL HEALTH* 8 (2022), p. 20552076221081696. DOI: 10.1177/20552076221081696.

[29] Priya Roy and Chandreyee Chowdhury. "A survey on ubiquitous WiFi-based indoor localization system for smartphone users from implementation perspectives". In: *CCF Transactions on Pervasive Computing and Interaction* (Jan. 2022). DOI: 10.1007/s42486-022-00089-3.

[30] Maria Posluk, Jesper Ahlander, Deep Shrestha, Sara Modarres Razavi, Gustav Lindmark, and Fredrik Gunnarsson. *5G Deployment Strategies for High Positioning Accuracy in Indoor Environments*. 2021. DOI: 10.48550/ARXIV.2105.09584. URL: https://arxiv.org/abs/2105.09584.

[31] Ali J. Ben Ali, Zakieh Sadat Hashemifar, and Karthik Dantu. "Edge-SLAM: Edge-Assisted Visual Simultaneous Localization and Mapping". In: *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*. MobiSys '20. Toronto, Ontario, Canada: Association for Computing Machinery, 2020, pp. 325–337. ISBN: 9781450379540. URL: https://doi.org/10.1145/3386901.3389033.

[32] Darwin Quezada-Gaibor, Joaquín Torres-Sospedra, Jari Nurmi, Yevgeni Koucheryavy, and Joaquín Huerta. "Cloud Platforms for Context-Adaptive Positioning and Localisation in GNSS-Denied Scenarios—A Systematic Review". In: *Sensors* 22 (Dec. 2021), p. 110. DOI: 10.3390/s22010110.

[33] Matthew J. Page, Joanne E. McKenzie, Patrick M. Bossuyt, Isabelle Boutron, Tammy C. Hoffmann, Cynthia D. Mulrow, Larissa Shamseer, Jennifer M. Tetzlaff, Elie A. Akl, Sue E. Brennan, Roger Chou, Julie Glanville, Jeremy M. Grimshaw, Asbjørn Hróbjartsson, Manoj M. Lalu, Tianjing Li, Elizabeth W. Loder, Evan Mayo-Wilson, Steve McDonald, Luke A. McGuinness, Lesley A.

Stewart, James Thomas, Andrea C. Tricco, Vivian A. Welch, Penny Whiting, and David Moher. "The PRISMA 2020 statement: An updated guideline for reporting systematic reviews". In: *International Journal of Surgery* 88 (2021), p. 105906. ISSN: 1743-9191. DOI: https://doi.org/10.1016/j.ijsu.2021.105906. URL: https://www.sciencedirect.com/science/article/pii/S1743919121000406.

[34] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P. Jue. "All one needs to know about fog computing and related edge computing paradigms: A complete survey". In: *JOURNAL OF SYSTEMS ARCHITECTURE* 98 (Sept. 2019), 289–330. ISSN: 1383-7621.

[35] T. V. Haute, E. D. Poorter, F. Lemic, V. Handziski, N. Wirström, T. Voigt, A. Wolisz, and I. Moerman. "Platform for benchmarking of RF-based indoor localization solutions". In: *IEEE Communications Magazine* 53.9 (2015), pp. 126–133.

[36] B. Vamsi Krishna and K. Aparna. "IoT-Based Indoor Navigation Wearable System for Blind People". In: *Artificial Intelligence and Evolutionary Computations in Engineering Systems*. Ed. by Subhransu Sekhar Dash, Paruchuri Chandra Babu Naidu, Ramazan Bayindir, and Swagatam Das. Singapore: Springer Singapore, 2018, pp. 413–421. ISBN: 978-981-10-7868-2.

[37] Z. Li, J. Cao, X. Liu, J. Zhang, H. Hu, and D. Yao. "A Self-Adaptive Bluetooth Indoor Localization System using LSTM-based Distance Estimator". In: *2020 29th International Conference on Computer Communications and Networks (ICCCN)*. 2020, pp. 1–9.

[38] M. Terán, H. Carrillo, and C. Parra. "WLAN-BLE Based Indoor Positioning System using Machine Learning Cloud Services". In: *2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA)*. 2018, pp. 1–6.

[39] Nan Jiang, Guangjie Dong, Yiying Hu, Li Gao, and Jing Chen. "Toward Efficient Indoor Positioning for Cloud Services in SIoT". In: *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*. 2020, pp. 124–132. DOI: 10.1109/CLOUD49709.2020.00030.

[40] G. H. Flores, T. D. Griffin, and D. Jadav. "An iBeacon Training App for Indoor Fingerprinting". In: *2017 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. 2017, pp. 173–176.

[41] Jun Zhong, Shasha Zhao, Xuefeng Han, Yongfeng Liu, and Kai Guo. "Research on Indoor and Outdoor Positioning System for Special Population". In: *IOP Conference Series: Materials Science and Engineering* 719 (Jan. 2020), p. 012055.

[42] J. A. D. C. Jayakody, I. Murray, J. Hermann, S. Lokuliyana, and V. R. Dunuwila. "Tempcache: A Database Optimization Algorithm for Real-Time Data Handling in Indoor Spatial Environments". In: *2018 13th International Conference on Computer Science Education (ICCSE)*. 2018, pp. 1–6.

[43] Atta ur Rehman Khan, Mazliza Othman, Sajjad Ahmad Madani, and Samee Ullah Khan. "A Survey of Mobile Cloud Computing Application Models". In: *IEEE Communications Surveys Tutorials* 16.1 (2014), pp. 393–413. DOI: 10.1109/SURV.2013.062613.00160.

[44] Junjian Huang, Yubin Zhao, XiaoFan Li, and Cheng-Zhong Xu. "Ultra-Low Power Localization System Using Mobile Cloud Computing". In: *Cloud Computing – CLOUD 2019*. Ed. by Dilma Da Silva, Qingyang Wang, and Liang-Jie Zhang. Cham: Springer International Publishing, 2019, pp. 1–10. ISBN: 978-3-030-23502-4.

[45] Marius Noreikis, Yu Xiao, and Antti Ylä-Jääski. "SeeNav: Seamless and Energy-Efficient Indoor Navigation using Augmented Reality". In: Oct. 2017, pp. 186–193.

[46] Sasa Pesic, Milenko Tosic, Ognjen Iković, Milos Radovanovic, Mirjana Ivanovic, and Dragan Boscovic. "BLEMAT: Data Analytics and Machine Learning for Smart Building Occupancy Detection and Prediction". In: *International Journal on Artificial Intelligence Tools* 28 (Sept. 2019), p. 1960005.

[47] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan. "Fog Computing: Survey of Trends, Architectures, Requirements, and Research Directions". In: *IEEE Access* 6 (2018), pp. 47980–48009.

[48] A. Sciarrone, C. Fiandrino, I. Bisio, F. Lavagetto, D. Kliazovich, and P. Bouvry. "Smart Probabilistic Fingerprinting for Indoor Localization over Fog Computing Platforms". In: *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*. 2016, pp. 39–44.

[49] C. C. Li, P. Wu, H. Wang, E. T. H. Chu, and J. W. S. Liu. "Building/environment Data/information System for Fine-Scale Indoor Location Specific Services". In: *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*. 2018, pp. 1–8.

[50] P. Battistoni, M. Sebillo, and G. Vitiello. "Experimenting with a Fog-computing Architecture for Indoor Navigation". In: *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*. 2019, pp. 161–165.

[51] Qianwen Ye, Hongxia Bie, Kuan Ching Li, Xiaochen Fan, Liangyi Gong, Xiangjian He, and Gengfa Fang. "EdgeLoc: A Robust and Real-time Localization System Towards Heterogeneous IoT Devices". In: *IEEE Internet of Things Journal* (2021), pp. 1–1. DOI: 10.1109/JIOT.2021.3101368.

[52] Jian An, Zhenxing Wang, Xin He, Xiaolin Gui, Jindong Cheng, and Ruowei Gui. "Know Where You Are: A Practical Privacy-Preserving Semi-Supervised Indoor Positioning via Edge-Crowdsensing". In: *IEEE Transactions on Network and Service Management* (2021), pp. 1–1. DOI: 10.1109/TNSM.2021.3107718.

[53] S. Liu, P. Si, M. Xu, Y. He, and Y. Zhang. "Edge Big Data-Enabled Low-Cost Indoor Localization Based on Bayesian Analysis of RSS". In: *2017 IEEE Wireless Communications and Networking Conference (WCNC)*. 2017, pp. 1–6.

[54] S. Liu and Z. Yan. "Verifiable Edge Computing for Indoor Positioning". In: *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. 2020, pp. 1–6.

[55] C. Liu, C. Wang, J. Luo, and Q. He. "A Cooperative Indoor Localization Enhancement Framework on Edge Computing Platforms for Safety-Critical Applications". In: *2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*. 2019, pp. 372–377.

[56] José Santa, Pedro Fernandez, Jordi Ortiz, Ramon Sanchez-Iborra, and Antonio Skarmeta. "Offloading Positioning onto Network Edge". In: *Wireless Communications and Mobile Computing* 2018 (Oct. 2018).

[57] Stelios Thomopoulos, Christina Karafylli, Maria Karafylli, Dionysis Motos, Vassilis Lampropoulos, Kostantinos Dimitros, and Christos Margonis. "WayGoo: a platform for geolocating and managing indoor and outdoor spaces". In: May 2016, p. 984218.

[58] Long Niu, Sachio Saiki, Shinsuke Matsumoto, and Masahide Nakamura. "WIF4InL: Web-based integration framework for Indoor location". In: *International Journal of Pervasive Computing and Communications* 12 (Apr. 2016), pp. 49–65.

[59] M. Raspopoulos, N. Paspallis, and P. Kaimakis. "PINSPOT: An open platform for intelligent context-based indoor positioning". In: 2019.

[60] G. SUCIU, C. BALANEAN, A. PASAT, C. ISTRATE, H. IJAZ, and R. MATEI. "A new concept of smart shopping platform based on IoT solutions". In: *2020 12th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. 2020, pp. 1–4.

[61] Edward Sykes. "A context-aware system using mobile applications and beacons for on-premise security environments". In: *Journal of Ambient Intelligence and Humanized Computing* 11 (Nov. 2020).

[62] Vlatko Nikolovski, Petre Lameski, and Ivan Chorbev. "Cloud Based Patient Monitoring Platform Using Android Smartphone Sensors". In: *Cybernetics and Information Technologies* 15.7 (2015), pp. 109–119.

[63] C. Chen, C. Hsieh, Y. Liao, and T. Yin. "Implementation of Wearable Devices and Indoor Positioning System for a Smart Hospital Environment". In: *2018 International Symposium in Sensing and Instrumentation in IoT Era (ISSI)*. 2018, pp. 1–5.

[64] Y. Guo, L. Zhao, Y. Wang, Q. Liu, and J. Qiu. "Fog-Enabled WLANs for Indoor Positioning". In: *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*. 2019, pp. 1–5.

[65] J. Cho, J. Yu, S. Oh, J. Ryoo, J. Song, and H. Kim. "Wrong Siren! A Location Spoofing Attack on Indoor Positioning Systems: The Starbucks Case Study". In: *IEEE Communications Magazine* 55.3 (2017), pp. 132–137.

[66] A. Ye, Q. Li, Q. Zhang, and B. Cheng. "Detection of Spoofing Attacks in WLAN-Based Positioning Systems Using WiFi Hotspot Tags". In: *IEEE Access* 8 (2020), pp. 39768–39780.

[67] Jacob Biehl, Adam Lee, and Gerry Filby. "Anchor of trust: towards collusion-resistant trusted indoor location for enterprise and industrial use". In: *Personal and Ubiquitous Computing* 24 (Oct. 2020).

[68] IETF. *The Transport Layer Security (TLS) Protocol Version 1.2*. Last accessed 16 March 2021. 2008. URL: https://tools.ietf.org/html/rfc5246#page-26.

[69] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications". In: *IEEE Communications Surveys Tutorials* 17.4 (2015), pp. 2347–2376.

[70] C. Navya, S. Salvi, N. D. Jacob, and S. Kumar. "A ROOF Computing Architecture based Indoor Positioning System for IoT Applications". In: *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. 2020, pp. 19–24.

[71] Angelos Chatzimichail, Christos Chatzigeorgiou, Athina Tsanousa, Dimos Ntioudis, Georgios Meditskos, Fotis Andritsopoulos, Christina Karaberi, Panagiotis Kasnesis, Dimitrios Kogias, Georgios Gorgogetas, Stefanos Vrochidis, Charalampos Patrikakis, and Ioannis Kompatsiaris. "Internet of Things Infrastructure for Security and Safety in Public Places". In: *Information* 10 (Oct. 2019), p. 333.

[72] Luca De Nardis, Giuseppe Caso, and Maria-Gabriella Di Benedetto. "ThingsLocate: A ThingSpeak-Based Indoor Positioning Platform for Academic Research on Location-Aware Internet of Things". In: *Technologies* 7 (July 2019), p. 50.

[73] T. Kulshrestha, D. Saxena, R. Niyogi, and J. Cao. "Real-Time Crowd Monitoring Using Seamless Indoor-Outdoor Localization". In: *IEEE Transactions on Mobile Computing* 19.3 (2020), pp. 664–679.

[74] Jiang Xiao, Zimu Zhou, Youwen Yi, and Lionel M. Ni. "A Survey on Wireless Indoor Localization from the Device Perspective". In: *ACM Comput. Surv.* 49.2 (June 2016). ISSN: 0360-0300. URL: https://doi.org/10.1145/2933232.

[75] S. M. J. Sadegh, S. Shahidi, and S. Valaee. "An efficient database management for cloud-based indoor positioning using Wi-Fi fingerprinting". In: *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. 2017, pp. 1–6.

[76] W. Li, Z. Chen, X. Gao, W. Liu, and J. Wang. "Multimodel Framework for Indoor Localization Under Mobile Edge Computing Environment". In: *IEEE Internet of Things Journal* 6.3 (2019), pp. 4844–4853.

[77] F. Lemic, V. Handziski, N. Wirström, T. Van Haute, E. De Poorter, T. Voigt, and A. Wolisz. "Web-based platform for evaluation of RF-based indoor localization algorithms". In: *2015 IEEE International Conference on Communication Workshop (ICCW)*. 2015, pp. 834–840.

[78] Andreas Konstantinidis, Aphrodite Demetriades, and Savvas Pericleous. "A Multi-Objective Indoor Localization Service for Smartphones". In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. SAC '19. Limassol, Cyprus: Association for Computing Machinery, 2019, pp. 1174–1181. ISBN: 9781450359337. URL: https://doi.org/10.1145/3297280.3297395.

[79] Bluetooth. *Bluetooth 5 Go Faster. Go Further.* Last accessed 16 March 2021. 2021. URL: https://www.bluetooth.com/wp-content/uploads/2019/03/Bluetooth_5-FINAL.pdf.

[80] Jochen Teizer, Mario Wolf, Olga Golovina, Manuel Perschewsk, Markus Propach, Herms-Matthias Neges, and Markus König. "Internet of Things (IoT) for Integrating Environmental and Localization Data in Building Information Modeling (BIM)". In: July 2017.

[81] Iuliana Marin, Maria-Iuliana Bocicor, and Arthur-Jozsef Molnar. "Indoor Localization Techniques Within a Home Monitoring Platform". In: *Evaluation of Novel Approaches to Software Engineering*. Ed. by Ernesto Damiani, George Spanoudakis, and Leszek A. Maciaszek. Cham: Springer International Publishing, 2020, pp. 378–401. ISBN: 978-3-030-40223-5.

[82] W. Chen, L. Chen, W. Chang, and J. Tang. "An IoT-based elderly behavioral difference warning system". In: *2018 IEEE International Conference on Applied System Invention (ICASI)*. 2018, pp. 308–309.

[83] Ramon Brena, Juan García-Vázquez, Carlos Galván Tejada, D. Munoz, Cesar Vargas-Rosales, James Fangmeyer Jr, and Alberto Palma. "Evolution of Indoor Positioning Technologies: A Survey". In: *Journal of Sensors* 2017 (Mar. 2017).

[84] Fekher Khelifi, Abbas Bradai, Abderrahim Benslimane, Priyanka Rawat, and Mohamed Atri. "A Survey of Localization Systems in Internet of Things". In: *MOBILE NETWORKS & APPLICATIONS* 24.3, SI (June 2019), 761–785. ISSN: 1383-469X.

[85] Nicola Blefari-Melazzi, Stefania Bartoletti, Luca Chiaraviglio, Flavio Morselli, Eduardo Baena, Giacomo Bernini, Domenico Giustiniano, Mythri Hunukumbure, Gürkan Solmaz, and Kostas Tsagkaris. "LOCUS: Localization and analytics on-demand embedded in the 5G ecosystem". In: *2020 European Conference on Networks and Communications (EuCNC)*. 2020, pp. 170–175. DOI: 10.1109/EuCNC48522.2020.9200961.

[86] Jun-Yan Chen and Long Huang. "Design of Elderly Care System Integrated with SLAM Algorithm". In: *Security with Intelligent Computing and Big-data Services*. Ed. by Ching-Nung Yang, Sheng-Lung Peng, and Lakhmi C. Jain. Cham: Springer International Publishing, 2020, pp. 503–513. ISBN: 978-3-030-16946-6.

[87] Christoph Jechlitschek. "A survey paper on Radio Frequency Identification (RFID) trends". In: (Jan. 2010).

[88] Yihai Fang, Yong K. Cho, Sijie Zhang, and Esau Perez. "Case Study of BIM and Cloud-Enabled Real-Time RFID Indoor Localization for Construction Management Applications". In: *JOURNAL OF CONSTRUCTION ENGINEERING AND MANAGEMENT* 142.7 (June 2016). ISSN: 0733-9364.

[89] Sheetal Datt, Mali Senapathi, and Farhaan Mirza. "IO Vision - An Integrated System to Support the Visually Impaired". In: Dec. 2017.

[90] D. Liu, S. Guo, Y. Yang, Y. Shi, and M. Chen. "Geomagnetism-Based Indoor Navigation by Offloading Strategy in NB-IoT". In: *IEEE Internet of Things Journal* 6.3 (2019), pp. 4074–4084.

[91] Martin Štancel, Ján Hurtuk, Michal Hulič, and Jakub Červeňák. "Indoor Atlas Service as a Tool for Building an Interior Navigation System". In: *Acta Polytechnica Hungarica* 18.9 (2021). Cited by: 1; All Open Access, Bronze Open Access, pp. 87–110. DOI: 10.12700/APH.18.9.2021.9.6.

[92] Rani Keerthana, G. Priyadarshini, Shriram Vasudevan, Harii Shree, and Karthik Venkatachalam. "An intelligent and interactive AR-based location identifier for indoor navigation". In: *International Journal of Advanced Intelligence Paradigms* 15 (Jan. 2020), p. 32.

[93] Chathurika Silva and Prasad Wimalaratne. "Context-Aware Assistive Indoor Navigation of Visually Impaired Persons". In: *Sensors and Materials* 32 (Apr. 2020), p. 1497.

[94] W. Zhao, L. Xu, B. Qi, J. Hu, T. Wang, and T. Runge. "Vivid: Augmenting Vision-Based Indoor Navigation System With Edge Computing". In: *IEEE Access* 8 (2020), pp. 42909–42923.

[95] Q. M. Qadir, T. A. Rashid, N. K. Al-Salihi, B. Ismael, A. A. Kist, and Z. Zhang. "Low Power Wide Area Networks: A Survey of Enabling Technologies, Applications and Interoperability Needs". In: *IEEE Access* 6 (2018), pp. 77454–77473.

[96] Sandra Sendra, Miguel Garcia-Pineda, Carlos Turro, and Jaime Lloret. "WLAN IEEE 802.11 a/b/g/n indoor coverage and interference performance study". In: *International Journal on Advances in Networks and Services* 4 (Jan. 2011).

[97] Savvas Pericleous, Andreas Konstantinidis, and Aphrodite Demetriades. "A Multi-Objective Indoor Localization Service for Smartphones". In: Apr. 2019.

[98] W. Liu, H. Jiang, G. Jiang, J. Liu, X. Ma, Y. Jia, and F. Xiao. "Indoor Navigation With Virtual Graph Representation: Exploiting Peak Intensities of Unmodulated Luminaries". In: *IEEE/ACM Transactions on Networking* 27.1 (2019), pp. 187–200.

[99] Tianyang Cao, T. Xue, L. Hong, H. Zhou, and Y. Song. "Generation of an Indoor 2D Map and Track Encryption Based on Mobile Crowdsourcing". In: *International Journal of Engineering and Technology(UAE)* 7 (Sept. 2018), pp. 4–19.

[100] M.C. Rodriguez-Sanchez and Juan Martinez-Romo. “GAWA – Manager for accessibility Wayfinding apps”. In: *International Journal of Information Management* 37.6 (2017), pp. 505–519. ISSN: 0268-4012. URL: https://www.sciencedirect.com/science/article/pii/S026840121630812X.

[101] Christy Sujin, Kousalya Govardhanan, and Maode Ma. “Markovian model based indoor location tracking for Internet of Things (IoT) applications”. In: *Cluster Computing* 22 (Sept. 2019).

[102] George Sithole and Sisi Zlatanova. “POSITION, LOCATION, PLACE AND AREA: AN INDOOR PERSPECTIVE”. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* III-4 (June 2016), pp. 89–96.

[103] Her-Tyan Yeh, B. Chen, and B. Wang. “A City Parking Integration System Combined with Cloud Computing Technologies and Smart Mobile Devices.” In: *Eurasia journal of mathematics, science and technology education* 2016 (2016), pp. 1231–1242.

[104] Eunyoung Cho, Sangjoon Park, Jehyeok Rew, Changjun Park, Soowhan Lee, and Youngmong Park. “Towards a Sustainable Open Platform for Location Intelligence and Convergence”. In: Oct. 2018, pp. 1411–1413.

[105] Y. Nikoloudakis, E. Markakis, G. Mastorakis, E. Pallis, and C. Skianis. “An NF V-powered emergency system for smart enhanced living environments”. In: *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. 2017, pp. 258–263.

[106] M. Tsai, J. Luo, M. Yang, and N. Lo. “Location Tracking and Forensic Analysis of Criminal Suspects’ Footprints”. In: *2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT)*. 2019, pp. 210–214.

[107] Z. Khaliq, P. Mirdita, A. Refaey, and X. Wang. “Unsupervised Manifold Alignment for Wifi RSSI Indoor Localization”. In: *2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*. 2020, pp. 1–7.

[108] Yan Li, Jong-Hyuk Park, and Byeong-Seok Shin. “A shortest path planning algorithm for cloud computing environment based on multi-access point topology analysis for complex indoor spaces”. In: *The Journal of Supercomputing* 73 (July 2017).

[109] G. Berkovich, D. Churikov, J. Georgy, and C. Goodall. "Coursa Venue: Indoor Navigation Platform Using Fusion of Inertial Sensors with Magnetic and Radio Fingerprinting". In: *2019 22th International Conference on Information Fusion (FUSION)*. 2019, pp. 1–6.

[110] M. Fazio, A. Celesti, and M. Villari. "Improving Proximity Detection of Mesh Beacons at the Edge for Indoor and Outdoor Navigation". In: *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. 2020, pp. 1–6.

[111] L. Gong, Y. Zhao, C. Xiang, Z. Li, C. Qian, and P. Yang. "Robust Light-Weight Magnetic-Based Door Event Detection with Smartphones". In: *IEEE Transactions on Mobile Computing* 18.11 (2019), pp. 2631–2646.

[112] Yongkang Wang, Chunxia Chen, and Qijie Jiang. "Security algorithm of Internet of Things based on ZigBee protocol". In: *CLUSTER COMPUTING-THE JOURNAL OF NETWORKS SOFTWARE TOOLS AND APPLICATIONS* 22.6 (Nov. 2019), 14759–14766. ISSN: 1386-7857.

[113] K. Li, S. Zlatanova, J. Torres-Sospedra, A. Perez-Navarro, C. Laoudias, and A. Moreira. "Survey on Indoor Map Standards and Formats". In: *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2019, pp. 1–8.

[114] ISO. *Information technology – Real time localization System – test and evaluation of localization and tracking systems*. Standard. Geneva, CH: International Organization for Standardization, Nov. 2016.

[115] Francesco Potortì, Antonino Crivello, Paolo Barsocchi, and Filippo Palumbo. "Evaluation of Indoor Localisation Systems: Comments on the ISO/IEC 18305 Standard". In: *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2018, pp. 1–7.

[116] Francesco Furfari, Antonino Crivello, Paolo Barsocchi, Filippo Palumbo, and Francesco Potortì. "What is next for Indoor Localisation? Taxonomy, protocols, and patterns for advanced Location Based Services". In: *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2019, pp. 1–8.

[117] iso. “ISO/IEC/IEEE International Standard - Software, systems and enterprise – Architecture processes”. In: *ISO/IEC/IEEE 42020:2019(E)* (2019), pp. 1–126.

[118] Amir Atabekov, Jing He, and Patrick Otoo Bobbie. “Internet of Things-Based Framework to Facilitate Indoor Localization Education”. In: *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 2. 2016, pp. 269–274. DOI: 10.1109/COMPSAC.2016.143.

[119] Kaikai Liu, Navjot Warade, Tejas Pai, and Keertikeya Gupta. “Location-aware smart campus security application”. In: *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/ UIC/ATC/CBDCom/IOP/SCI)*. 2017, pp. 1–8. DOI: 10.1109/UIC-ATC.2017.8397588.

[120] Quanyi Hu, Feng Wu, Raymond K. Wong, Richard C. Millham, and Jinan Fiaidhi. “A novel indoor localization system using machine learning based on bluetooth low energy with cloud computing”. In: *Computing* (2021). Cited by: 3. DOI: 10.1007/s00607-020-00897-4.

[121] Hongbo Jiang, Wenping Liu, Guoyin Jiang, Yufu Jia, Xingjun Liu, Zhicheng Lui, Xiaofei Liao, Jing Xing, and Daibo Liu. “Fly-Navi: A Novel Indoor Navigation System With On-the-Fly Map Generation”. In: *IEEE Transactions on Mobile Computing* 20.9 (2021), pp. 2820–2834. DOI: 10.1109/TMC.2020.2990446.

[122] Iuliana Marin and Arthur-Jozsef Molnar. “Evaluation of Indoor Localisation and Heart Rate Evolution”. In: *Computational Science and Its Applications – ICCSA 2021*. Ed. by Osvaldo Gervasi, Beniamino Murgante, Sanjay Misra, Chiara Garau, Ivan Blečić, David Taniar, Bernady O. Apduhan, Ana Maria A. C. Rocha, Eufemia Tarantino, and Carmelo Maria Torre. Cham: Springer International Publishing, 2021, pp. 75–89. ISBN: 978-3-030-86976-2.

[123] Lun-Ping Hung, Nan-Chen Hsieh, and Chien-Liang Chen. “A RFID-Based Infection Prevention and Control Mechanism in Aged Care Living Residences”. In: *Mobile Networks and Applications* 27.1 (2022). Cited by: 0; All Open Access, Bronze Open Access, pp. 33–46. DOI: 10.1007/s11036-020-01707-z.

[124] Mauro Borrazzo, Giuseppe D'Ambrosio, Vittorio Scarano, and Carmine Spagnuolo. "TraceMeNow: an Open-Source Software Framework for Indoor Localization Applications". In: vol. 3097. Cited by: 0. 2021.

[125] Zhiheng Wang, Yanyan Xu, Yuejing Yan, Yiran Zhang, Zheheng Rao, and Xue Ouyang. "Privacy-preserving indoor localization based on inner product encryption in a cloud environment". In: *Knowledge-Based Systems* 239 (2022). Cited by: 0. DOI: 10.1016/j.knosys.2021.108005.

[126] Shushu Liu and Zheng Yan. "Efficient Privacy Protection Protocols for 5G Enabled Positioning in Industrial IoT". In: *IEEE Internet of Things Journal* (2022). Cited by: 0; All Open Access, Green Open Access. DOI: 10.1109/JIOT.2022.3161148.

[127] Xiaoying Wang, Xiaodong Zhang, Chenxi Zu, Zijiang Yang, Guohua Bian, Yongbiao Zhang, Weiqi Ruan, Benquan Wu, Xiaoqi Wu, Lianxiong Yuan, Qingwu Wu, and Qintai Yang. "An accurate cloud-based indoor localization system with low latency". In: *International Journal of Intelligent Systems* 37.8 (2022). Cited by: 0, pp. 4794–4809. DOI: 10.1002/int.22740.

[128] Ji Li, Gopalkarur Rathinam, and A. Sigappi. "IoT in a museum for interactive experience design". In: *Annals of Operations Research* (Nov. 2021). DOI: 10.1007/s10479-021-04419-z.

[129] D. Geetha, A. Logarathinam, and R. Parvathi. "Fire Escape Route Tracing Using Sensors, Artificial Intelligence, and Azure Cloud". In: *Lecture Notes in Electrical Engineering* 792 (2022). Cited by: 0, pp. 95–103. DOI: 10.1007/978-981-16-4625-6_10.

[130] Ketan Ramaneti, Nikita Mohanty, and Vinoth Babu Kumaravelu. "IoT based 2D Indoor Navigation System using BLE Beacons and Dijkstra's Algorithm". In: Cited by: 0. 2021. DOI: 10.1109/ICCCNT51525.2021.9580047.

[131] Paul Mirdita, Zain Khaliq, Ahmed Refaey Hussein, and Xianbin Wang. "Localization for Intelligent Systems Using Unsupervised Learning and Prediction Approaches". In: *IEEE Canadian Journal of Electrical and Computer Engineering* 44.4 (2021), pp. 443–455. DOI: 10.1109/ICJECE.2021.3062971.

[132] Naser El-Sheimy and You Li. "Indoor navigation: state of the art and future trends". In: *Satellite Navigation* 2.1 (2021). Cited by: 30; All Open Access, Gold Open Access. DOI: 10.1186/s43020-021-00041-3.

[133] Shuchen Zhao. "The Cloudlet MEC Enabling ISAR Based Indoor Localization and Navigation System Using Passive UHF RFID". In: Cited by: 0. 2021, pp. 331–336. DOI: 10.1109/IAECST54258.2021.9695729.

[134] Kriti Bhargava and Stepan Ivanov. "A fog computing approach for localization in WSN". In: *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. 2017, pp. 1–7. DOI: 10.1109/PIMRC.2017.8292245.

[135] Weiwei Li, Kuan Zhang, Zhou Su, Rongxing Lu, and Ying Wang. "Anomalous Path Detection for Spatial Crowdsourcing-Based Indoor Navigation System". In: *2018 IEEE Global Communications Conference (GLOBECOM)*. 2018, pp. 1–7. DOI: 10.1109/GLOCOM.2018.8647174.

[136] Liang Xie, H. Tom Luan, Zhou Su, Qichao Xu, and Nan Chen. "A Game Theoretical Approach for Secure Crowdsourcing-Based Indoor Navigation System with Reputation Mechanism". In: *IEEE Internet of Things Journal* (2021), pp. 1–1. DOI: 10.1109/JIOT.2021.3111999.

[137] Xiangyu Liu, Lei Guo, Helin Yang, and Xuetao Wei. "Visible Light Positioning Based on Collaborative LEDs and Edge Computing". In: *IEEE Transactions on Computational Social Systems* 9.1 (2022). Cited by: 2, pp. 324–335. DOI: 10.1109/TCSS.2021.3109631.

[138] Qianwen Ye, Hongxia Bie, Kuan-Ching Li, Xiaochen Fan, Liangyi Gong, Xiangjian He, and Gengfa Fang. "EdgeLoc: A Robust and Real-Time Localization System Toward Heterogeneous IoT Devices". In: *IEEE Internet of Things Journal* 9.5 (2022). Cited by: 5, pp. 3865–3876. DOI: 10.1109/JIOT.2021.3101368.

[139] Mohammed Alloulah and Lauri Tuominen. "IMULet: A Cloudlet for Inertial Tracking". In: Cited by: 0. 2021, pp. 50–56. DOI: 10.1145/3446382.3448364.

[140] Taehun Yang, Sang-Hoon Lee, and Soochang Park. "Ai-aided individual emergency detection system in edge-internet of things environments". In: *Electronics (Switzerland)* 10.19 (2021). Cited by: 1; All Open Access, Gold Open Access. DOI: 10.3390/electronics10192374.

[141] Zhonglin Ding, Wei Cao, Zheng Zhang, Zhikun Wang, Yi Zhang, and Shuang Yang. "Edge Computing Empowered Indoor Positioning: A Brief Introduction". In: Cited by: 0. 2021. DOI: 10.1109/WCSP52459.2021.9613444.

[142] Marwa Zamzam, Tallal Elshabrawy, and Mohamed Ashour. "A Minimized Latency Collaborative Computation Offloading Game Under Mobile Edge Computing for Indoor Localization". In: *IEEE Access* 9 (2021), pp. 133861–133874. DOI: 10.1109/ACCESS.2021.3115157.

[143] Jian An, Zhenxing Wang, Xin He, Xiaolin Gui, Jindong Cheng, and Ruowei Gui. "Know Where You are: A Practical Privacy-Preserving Semi-Supervised Indoor Positioning via Edge-Crowdsensing". In: *IEEE Transactions on Network and Service Management* 18.4 (2021). Cited by: 1, pp. 4875–4887. DOI: 10.1109/TNSM.2021.3107718.

[144] Xuejun Zhang, Fucun He, Qian Chen, Xinlong Jiang, Junda Bao, Tongwei Ren, and Xiaogang Du. "A Differentially Private Indoor Localization Scheme with Fusion of WiFi and Bluetooth Fingerprints in Edge Computing". In: *Neural Comput. Appl.* 34.6 (Mar. 2022), pp. 4111–4132. ISSN: 0941-0643. DOI: 10.1007/s00521-021-06815-9. URL: https://doi.org/10.1007/s00521-021-06815-9.

[145] Xiaoying Wang, Xiaodong Zhang, Chenxi Zu, Zijiang Yang, Guohua Bian, Yongbiao Zhang, Weiqi Ruan, Benquan Wu, Xiaoqi Wu, Lianxiong Yuan, Qingwu Wu, and Qintai Yang. "An accurate cloud-based indoor localization system with low latency". In: *International Journal of Intelligent Systems* 37.8 (2022), pp. 4794–4809. DOI: https://doi.org/10.1002/int.22740. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/int.22740. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/int.22740.

[146] Joaquín Torres-Sospedra, Philipp Richter, Adriano Moreira, Germán M. Mendoza-Silva, Elena Simona Lohan, Sergio Trilles, Miguel Matey-Sanz, and Joaquín Huerta. "A Comprehensive and Reproducible Comparison of Clustering and Optimization Rules in Wi-Fi Fingerprinting". In: *IEEE Transactions on Mobile Computing* 21.3 (2022), pp. 769–782. DOI: 10.1109/TMC.2020.3017176.

[147] Yaoxin Duan, Kam-Yiu Lam, Victor C. S. Lee, Wendi Nie, Kai Liu, Hao Li, and Chun Jason Xue. "Data Rate Fingerprinting: A WLAN-Based Indoor Positioning Technique for Passive Localization". In: *IEEE Sensors Journal* 19.15 (2019), pp. 6517–6529. DOI: 10.1109/JSEN.2019.2911690.

[148] Germán Martín Mendoza-Silva, Philipp Richter, Joaquín Torres-Sospedra, Elena Simona Lohan, and Joaquín Huerta. "Long-Term WiFi Fingerprinting Dataset for Research on Robust Indoor Positioning". In: *Data* 3.1 (2018). ISSN: 2306-5729. DOI: 10.3390/data3010003. URL: https://www.mdpi.com/2306-5729/3/1/3.

[149] Shilong Dai, Liang He, and Xuebo Zhang. "Autonomous WiFi Fingerprinting for Indoor Localization". In: *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*. 2020, pp. 141–150. DOI: 10.1109/ICCPS48487.2020.00021.

[150] Ivo Silva, Cristiano Pendão, Joaquín Torres-Sospedra, and Adriano Moreira. "Quantifying the Degradation of Radio Maps in Wi-Fi Fingerprinting". In: *2021 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2021, pp. 1–8. DOI: 10.1109/IPIN51156.2021.9662558.

[151] Oluwaseyi Paul Babalola and Vipin Balyan. "WiFi Fingerprinting Indoor Localization Based on Dynamic Mode Decomposition Feature Selection with Hidden Markov Model". In: *Sensors* 21.20 (2021). ISSN: 1424-8220. DOI: 10.3390/s21206778. URL: https://www.mdpi.com/1424-8220/21/20/6778.

[152] Lucie Klus, Darwin Quezada-Gaibor, Joaquín Torres-Sospedra, Elena Simona Lohan, Carlos Granell, and Jari Nurmi. "Rss fingerprinting dataset size reduction using feature-wise adaptive k-means clustering". In: *2020 12th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. IEEE. 2020, pp. 195–200.

[153] Joaquín Torres-Sospedra, Joan P. Avariento, David Rambla, Raúl Montoliu, Sven Casteleyn, Mauri Benedito-Bordonau, Michael Gould, and Joaquín Huerta. "Enhancing integrated indoor/outdoor mobility in a smart campus". In: *International Journal of Geographical Information Science* 29 (2015), pp. 1955–1968.

[154] Jing Sun, Baoezeng Wang, Xiaoxu Song, and Xiaochun Yang. "Data Cleaning for Indoor Crowdsourced RSSI Sequences". In: Aug. 2021, pp. 267–275. ISBN: 978-3-030-85898-8. DOI: 10.1007/978-3-030-85899-5_20.

[155] Elena Simona Lohan, Joaquín Torres-Sospedra, Helena Leppäkoski, Philipp Richter, Zhe Peng, and Joaquín Huerta. "Wi-Fi Crowdsourced Fingerprinting Dataset for Indoor Positioning". In: *Data* 2.4 (2017). ISSN: 2306-5729. DOI: 10.3390/data2040032. URL: https://www.mdpi.com/2306-5729/2/4/32.

[156] T. King, T. Haenselmann, and W. Effelsberg. "On-demand fingerprint selection for 802.11-based positioning systems". In: *2008 International Symposium on a World of Wireless, Mobile and Multimedia Networks*. June 2008, pp. 1–8.

[157] Shweta Shrestha, Jukka Talvitie, and Elena Simona Lohan. "Deconvolution-based indoor localization with WLAN signals and unknown access point locations". In: *2013 International Conference on Localization and GNSS (ICL-GNSS)*. 2013, pp. 1–6. DOI: 10.1109/ICL-GNSS.2013.6577256.

[158] Philipp Richter, Elena Simona Lohan, and Jukka Talvitie. *WLAN (WiFi) RSS database for fingerprinting positioning*. Version 1.0.0. Zenodo, Jan. 2018. DOI: 10.5281/zenodo.1161525. URL: https://doi.org/10.5281/zenodo.1161525.

[159] Lohan. *Additional TAU datasets for Wi-Fi fingerprinting- based positioning*. Version v1, 11.05.2020. Zenodo, May 2020. DOI: 10.5281/zenodo.3819917. URL: https://doi.org/10.5281/zenodo.3819917.

[160] Joaquín Torres-Sospedra, Raúl Montoliu, Adolfo Martínez-Usó, Joan P. Avariento, Tomás J. Arnau, Mauri Benedito-Bordonau, and Joaquín Huerta. "UJIIndoorLoc: A new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems". In: *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2014, pp. 261–270. DOI: 10.1109/IPIN.2014.7275492.

[161] Joaquín Torres-Sospedra, Adriano J. C. Moreira, Stefan Knauth, Rafael Berkvens, Raúl Montoliu, Óscar Belmonte, Sergi Trilles, Maria João Nicolau, Filipe Meneses, António D. Costa, Athanasios Koukofikis, Maarten Weyn, and Herbert Peremans. "A realistic evaluation of indoor positioning systems based on Wi-Fi fingerprinting: The 2015 EvAAL-ETRI competition". In: *J. Ambient Intell. Smart Environ.* 9 (2017), pp. 263–279.

[162] Xudong Song, Xiaochen Fan, Chaocan Xiang, Qianwen Ye, Leyu Liu, Zumin Wang, Xiangjian He, Ning Yang, and Gengfa Fang. "A Novel Convolutional Neural Network Based Indoor Localization Framework With WiFi Fingerprinting". In: *IEEE Access* 7 (2019), pp. 110698–110709. DOI: 10.1109/ACCESS.2019.2933921.

[163] Elena Simona Lohan, Joaquín Torres-Sospedra, and Alejandro Gonzalez. *WiFi RSS measurements in Tampere University multi- building campus, 2017*. Version 1. Zenodo, Aug. 2021. DOI: 10.5281/zenodo.5174851. URL: https://doi.org/10.5281/zenodo.5174851.

[164] Fahad Alhomayani and Mohammad H. Mahoor. "OutFin, a multi-device and multi-modal dataset for outdoor localization based on the fingerprinting approach". In: *Scientific Data* 8.1 (Feb. 2021). DOI: 10.1038/s41597-021-00832-y. URL: https://doi.org/10.1038%5C%2Fs41597-021-00832-y.

[165] Germán Martín Mendoza-Silva, Miguel Matey-Sanz, Joaquín Torres-Sospedra, and Joaquín Huerta. "BLE RSS Measurements Dataset for Research on Accurate Indoor Positioning". In: *Data* 4.1 (2019). ISSN: 2306-5729. DOI: 10.3390/data4010012. URL: https://www.mdpi.com/2306-5729/4/1/12.

[166] Fernando J. Aranda, Felipe Parralejo, Fernando J. Álvarez, and Joaquín Torres-Sospedra. "Multi-Slot BLE Raw Database for Accurate Positioning in Mixed Indoor/Outdoor Environments". In: *Data* 5.3 (2020). ISSN: 2306-5729. DOI: 10.3390/data5030067. URL: https://www.mdpi.com/2306-5729/5/3/67.

[167] Joaquín Torres-Sospedra, Raúl Montoliu, Sergio Trilles, Óscar Belmonte, and Joaquín Huerta. "Comprehensive analysis of distance and similarity measures for Wi-Fi fingerprinting indoor positioning systems". In: *Expert Systems with Applications* 42.23 (2015), pp. 9263–9278. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2015.08.013. URL: https://www.sciencedirect.com/science/article/pii/S0957417415005527.

[168] Aleksandr Ometov, Viktoriia Shubina, Lucie Klus, Justyna Skibińska, Salwa Saafi, Pavel Pascacio, Laura Flueratoru, Darwin Quezada Gaibor, Nadezhda Chukhno, Olga Chukhno, Asad Ali, Asma Channa, Ekaterina Svertoka, Waleed Bin Qaim, Raúl Casanova-Marqués, Sylvia Holcer, Joaquín Torres-Sospedra, Sven Casteleyn, Giuseppe Ruggeri, Giuseppe Araniti, Radim Bur-

get, Jiri Hosek, and Elena Simona Lohan. "A Survey on Wearable Technology: History, State-of-the-Art and Current Challenges". In: *Computer Networks* 193 (2021), p. 108074. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2021.108074. URL: https://www.sciencedirect.com/science/article/pii/S1389128621001651.

[169] Youxiong Wu and Zhaohui Li. "An Effective Algorithm for Detecting and Eliminating Wi-Fi Fingerprint Outliers". In: *2017 5th International Conference on Enterprise Systems (ES)*. 2017, pp. 270–275. DOI: 10.1109/ES.2017.51.

[170] Ali Khalajmehrabadi, Nikolaos Gatsis, and David Akopian. "Structured Group Sparsity: A Novel Indoor WLAN Localization, Outlier Detection, and Radio Map Interpolation Scheme". In: *IEEE Transactions on Vehicular Technology* PP (Oct. 2016). DOI: 10.1109/TVT.2016.2631980.

[171] Alexandra Zayets and Eckehard Steinbach. "Robust WiFi-based indoor localization using multipath component analysis". In: *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2017, pp. 1–8. DOI: 10.1109/IPIN.2017.8115943.

[172] Pu Wang, Toshiaki Koike-Akino, and Philip V. Orlik. "Fingerprinting-Based Indoor Localization with Commercial MMWave WiFi: NLOS Propagation". In: *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. 2020, pp. 1–6. DOI: 10.1109/GLOBECOM42002.2020.9348144.

[173] Yiming Lin, Daokun Jiang, Roberto Yus, Georgios Bouloukakis, Andrew Chio, Sharad Mehrotra, and Nalini Venkatasubramanian. "Locater: Cleaning Wifi Connectivity Datasets for Semantic Localization". In: *Proc. VLDB Endow.* 14.3 (Nov. 2020), pp. 329–341. ISSN: 2150-8097. DOI: 10.14778/3430915.3430923. URL: https://doi.org/10.14778/3430915.3430923.

[174] Darwin Quezada-Gaibor, Lucie Klus, Joaquín Torres-Sospedra, Elena Simona Lohan, Jari Nurmi, Carlos Granell, and Joaquín Huerta. "Data Cleansing for Indoor Positioning Wi-Fi Fingerprinting Datasets". In: *2022 23rd IEEE International Conference on Mobile Data Management (MDM)*. 2022, pp. 349–354. DOI: 10.1109/MDM55031.2022.00079.

[175] Connor Shorten and Taghi M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning". In: *Journal of Big Data* 6.1 (2019), pp. 1–48. DOI: 10.1186/S40537-019-0197-0.

[176] Qiyue Li, Heng Qu, Zhi Liu, Nana Zhou, Wei Sun, Stephan Sigg, and Jie Li. "AF-DCGAN: Amplitude Feature Deep Convolutional GAN for Fingerprint Construction in Indoor Localization Systems". In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 5.3 (2021), pp. 468–480. DOI: 10.1109/TETCI.2019.2948058.

[177] J. H. Seong and D. H. Seo. "Selective Unsupervised Learning-Based Wi-Fi Fingerprint System Using Autoencoder and GAN". In: *IEEE Internet of Things Journal* 7.3 (2020), pp. 1898–1909. DOI: 10.1109/JIOT.2019.2956986.

[178] Wafa Njima, Marwa Chafii, Arsenia Chorti, Raed M. Shubair, and H. Vincent Poor. "Indoor Localization Using Data Augmentation via Selective Generative Adversarial Networks". In: *IEEE Access* 9 (2021), pp. 98337–98347. DOI: 10.1109/ACCESS.2021.3095546.

[179] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. *Generative Adversarial Networks*. 2014. DOI: 10.48550/ARXIV.1406.2661. URL: https://arxiv.org/abs/1406.2661.

[180] Darwin Quezada-Gaibor, Joaquín Torres-Sospedra, Jari Nurmi, Yevgeni Koucheryavy, and Joaquín Huerta. "SURIMI: Supervised Radio Map Augmentation with Deep Learning and a Generative Adversarial Network for Fingerprint-based Indoor Positioning". In: *2022 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2022.

[181] Giuseppe Caso, Luca De Nardis, Filip Lemic, Vlado Handziski, Adam Wolisz, and Maria-Gabriella Di Benedetto. "ViFi: Virtual Fingerprinting WiFi-Based Indoor Positioning via Multi-Wall Multi-Floor Propagation Model". In: *IEEE Transactions on Mobile Computing* 19.6 (2020), pp. 1478–1491. DOI: 10.1109/TMC.2019.2908865.

[182] Joaquín Torres-Sospedra, Darwin Quezada-Gaibor, Germán M. Mendoza-Silva, Jari Nurmi, Yevgeni Koucheryavy, and Joaquín Huerta. "New Cluster Selection and Fine-grained Search for k-Means Clustering and Wi-Fi Fingerprinting". In: *2020 International Conference on Localization and GNSS (ICL-GNSS)*. 2020, pp. 1–6. DOI: 10.1109/ICL-GNSS49876.2020.9115419.

[183] Caceja Elyca Anak Bundak, Mohd Amiruddin Abd Rahman, Muhammad Khalis Abdul Karim, and Nurul Huda Osman. "Fuzzy rank cluster top k Euclidean distance and triangle based algorithm for magnetic field indoor positioning system". In: *Alexandria Engineering Journal* 61.5 (2022), pp. 3645–3655. ISSN: 1110-0168. DOI: https://doi.org/10.1016/j.aej.2021.08.073. URL: https://www.sciencedirect.com/science/article/pii/S1110016821005883.

[184] Jingxue Bi, Hongji Cao, Yunjia Wang, Guoqiang Zheng, Keqiang Liu, Na Cheng, and Meiqi Zhao. "DBSCAN and TD Integrated Wi-Fi Positioning Algorithm". In: *Remote Sensing* 14.2 (2022). ISSN: 2072-4292. DOI: 10.3390/rs14020297. URL: https://www.mdpi.com/2072-4292/14/2/297.

[185] David Arthur and Sergei Vassilvitskii. "K-means++: The advantages of careful seeding." In: *In Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '07*, 2007, pp. 1027–1035.

[186] James C. Bezdek, Robert Ehrlich, and William Full. "FCM: The fuzzy c-means clustering algorithm". In: *Computers & Geosciences* 10.2 (1984), pp. 191–203. ISSN: 0098-3004. DOI: https://doi.org/10.1016/0098-3004(84)90020-7. URL: https://www.sciencedirect.com/science/article/pii/0098300484900207.

[187] Danyang Cao and Bingru Yang. "An improved k-medoids clustering algorithm". In: *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*. Vol. 3. 2010, pp. 132–135. DOI: 10.1109/ICCAE.2010.5452085.

[188] Delbert Dueck. "Affinity propagation: Clustering data by passing messages". In: *PhD thesis* (Jan. 2009).

[189] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[190] Darwin Quezada-Gaibor, Lucie Klus, Joaquiín Torres-Sospedra, Elena Simona Lohan, Jari Nurmi, and Joaquín Huerta. "Improving DBSCAN for Indoor Positioning Using Wi-Fi Radio Maps in Wearable and IoT Devices".

In: *2020 12th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. 2020, pp. 208–213. DOI: 10.1109/ICUMT51630.2020.9222411.

[191] Darwin Quezada-Gaibor, Joaquín Torres-Sospedra, Jari Nurmi, Yevgeni Koucheryavy, and Joaquín Huerta. "Lightweight Wi-Fi Fingerprinting with a Novel RSS Clustering Algorithm". In: *2021 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2021, pp. 1–8. DOI: 10.1109/IPIN51156.2021.9662612.

[192] Zhenghua Chen, Han Zou, JianFei Yang, Hao Jiang, and Lihua Xie. "WiFi Fingerprinting Indoor Localization Using Local Feature-Based Deep LSTM". In: *IEEE Systems Journal* 14.2 (2020), pp. 3001–3010. DOI: 10.1109/JSYST.2019.2918678.

[193] Xingfa Shen, Chuang Li, Weijie Chen, and Yongcai Wang. "MapICT: Unsupervised Radio-Map Learning From Imbalanced Crowd-Sourced Trajectories". In: *IEEE Sensors Journal* 22.3 (2022), pp. 2399–2408. DOI: 10.1109/JSEN.2021.3133865.

[194] Donglin Wang, Ting Wang, Feng Zhao, and Xuetao Zhang. "Improved Graph-Based Semi-Supervised Learning for Fingerprint-Based Indoor Localization". In: *2018 IEEE Global Communications Conference (GLOBECOM)*. 2018, pp. 1–6. DOI: 10.1109/GLOCOM.2018.8647621.

[195] Antoni Pérez-Navarro, Joaquín Torres-Sospedra, Raul Montoliu, Jordi Conesa, Rafael Berkvens, Giuseppe Caso, Constantinos Costa, Nicola Dorigatti, Noelia Hernández, Stefan Knauth, Elena Simona Lohan, Juraj Machaj, Adriano Moreira, and Pawel Wilk. "1 - Challenges of Fingerprinting in Indoor Positioning and Navigation". In: *Geographical and Fingerprinting Data to Create Systems for Indoor Positioning and Indoor/Outdoor Navigation*. Ed. by Jordi Conesa, Antoni Pérez-Navarro, Joaquín Torres-Sospedra, and Raul Montoliu. Intelligent Data-Centric Systems. Academic Press, 2019, pp. 1–20. ISBN: 978-0-12-813189-3. DOI: https://doi.org/10.1016/B978-0-12-813189-3.00001-0. URL: https://www.sciencedirect.com/science/article/pii/B9780128131893000010.

[196] Wafa Njima, Marwa Chafii, Ahmad Nimr, and Gerhard Fettweis. "Deep Learning Based Data Recovery for Localization". In: *IEEE Access* 8 (2020), pp. 175741–175752. DOI: 10.1109/ACCESS.2020.3026615.

[197] Rongrong Wang, Haiyong Luo, Qu Wang, Zhaohui Li, Fang Zhao, and Jingyu Huang. "A Spatial–Temporal Positioning Algorithm Using Residual Network and LSTM". In: *IEEE Transactions on Instrumentation and Measurement* 69.11 (2020), pp. 9251–9261. DOI: 10.1109/TIM.2020.2998645.

[198] Kan Ngamakeur, Sira Yongchareon, Jian Yu, and Quan Z. Sheng. "Deep CNN-LSTM Network for Indoor Location Estimation using Analog Signals of Passive Infrared Sensors". In: *IEEE Internet of Things Journal* (2022), pp. 1–1. DOI: 10.1109/JIOT.2022.3183148.

[199] Xudong Song, Xiaochen Fan, Xiangjian He, Chaocan Xiang, Qianwen Ye, Xiang Huang, Gengfa Fang, Liming Luke Chen, Jing Qin, and Zumin Wang. *Cnnloc: Deep-learning based indoor localization with wifi fingerprinting*. Version 2.0.4. July 2019. DOI: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00139. URL: https://github.com/XudongSong/CNNLoc-%20Access.git.

[200] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. "Extreme learning machine: Theory and applications". In: *Neurocomputing* 70 (2006), pp. 489–501. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2005.12.126. URL: https://www.sciencedirect.com/science/article/pii/S0925231206000385.

[201] Hualong Yu, Xibei Yang, Shang Zheng, and Changyin Sun. "Active Learning From Imbalanced Data: A Solution of Online Weighted Extreme Learning Machine". In: *IEEE Transactions on Neural Networks and Learning Systems* 30.4 (2019), pp. 1088–1103. DOI: 10.1109/TNNLS.2018.2855446.

[202] Meiyi Li, Weibiao Cai, and Qingshuai Sun. "Extreme Learning Machine for Regression Based on Condition Number and Variance Decomposition Ratio". In: *Proceedings of 2018 International Conference on Mathematics and Artificial Intelligence*. 2018, pp. 42–45. DOI: 10.1145/3208788.3208794. URL: https://doi.org/10.1145/3208788.3208794.

[203] Jichao Chen, Yijie Zeng, Yue Li, and Guang-Bin Huang. "Unsupervised feature selection based extreme learning machine for clustering". In: *Neurocomputing* 386 (2020), pp. 198–207. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2019.12.065. URL: https://www.sciencedirect.com/science/article/pii/S0925231219317746.

[204] Liyanaarachchi Lekamalage Chamara Kasun, Yan Yang, Guang-Bin Huang, and Zhengyou Zhang. "Dimension Reduction With Extreme Learning Machine". In: *IEEE Transactions on Image Processing* 25.8 (2016), pp. 3906–3918. DOI: 10.1109/TIP.2016.2570569.

[205] Xiaoxuan Lu, Han Zou, Hongming Zhou, Lihua Xie, and Guang-Bin Huang. "Robust Extreme Learning Machine With its Application to Indoor Positioning". In: *IEEE Transactions on Cybernetics* 46 (Jan. 2015), pp. 1–1. DOI: 10.1109/TCYB.2015.2399420.

[206] Han Zou, Xiaoxuan Lu, Hao Jiang, and Lihua Xie. "A Fast and Precise Indoor Localization Algorithm Based on an Online Sequential Extreme Learning Machine". In: *Sensors* 15.1 (2015), pp. 1804–1824. ISSN: 1424-8220. DOI: 10.3390/s150101804. URL: https://www.mdpi.com/1424-8220/15/1/1804.

[207] Hengyi Gan, Mohd Haris Bin Md Khir, Gunawan Witjaksono Bin Djaswadi, and Nordin Ramli. "A Hybrid Model Based on Constraint OSELM, Adaptive Weighted SRC and KNN for Large-Scale Indoor Localization". In: *IEEE Access* 7 (2019), pp. 6971–6989. DOI: 10.1109/ACCESS.2018.2890111.

[208] Xiaoxuan Lu, Han Zou, Hongming Zhou, Lihua Xie, and Guangbin Huang. "Robust Extreme Learning Machine With its Application to Indoor Positioning". In: *IEEE Transactions on Cybernetics* 46 (2016), pp. 194–205.

[209] Radu Dogaru and Ioana Dogaru. "BCONV - ELM: Binary Weights Convolutional Neural Network Simulator based on Keras/Tensorflow, for Low Complexity Implementations". In: *2019 6th International Symposium on Electrical and Electronics Engineering (ISEEE)*. 2019, pp. 1–6. DOI: 10.1109/ISEEE48094.2019.9136102.

[210] FIND. *FIND The Framework for international Navigation and Discovery*. https://www.internalpositioning.com/ [Accessed 2023-01-31]. 2020.

[211] indoorlocation.io. *The first open-source unified framework to manage locations coming from any Indoor Positioning Technology.* https://www.indoorlocation.io/ [Accessed 2023-01-31]. 2020.

[212] C. A. Cois, J. Yankel, and A. Connell. "Modern DevOps: Optimizing software development through effective system interactions". In: *2014 IEEE International Professional Communication Conference (IPCC)*. 2014, pp. 1–7. DOI: 10.1109/IPCC.2014.7020388.

[213] R. Pietrantuono, S. Russo, and A. Guerriero. "Run-Time Reliability Estimation of Microservice Architectures". In: *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*. 2018, pp. 25–35.

[214] H. Calderón-Gómez, L. Mendoza-Pittí, M. Vargas-Lombardo, J. M. Gómez-Pulido, J. L. Castillo-Sequera, J. Sanz-Moreno, and G. Sención. "Telemonitoring System for Infectious Disease Prediction in Elderly People Based on a Novel Microservice Architecture". In: *IEEE Access* 8 (2020), pp. 118340–118354. DOI: 10.1109/ACCESS.2020.3005638.

[215] SWG IndoorGML. *IndoorGML*. http://www.indoorgml.net/ [Accessed 2023-01-31]. 2022.

[216] RethingDB. *The open-source database for the realtime web*. https://rethinkdb.com/ [Accessed 2023-01-31]. 2020.

[217] Appwrite. *Appwrite*. https://www.appwrite.io/ [Accessed 2023-01-31]. 2022.

[218] Quezada Darwin. *Cloud-based indoor positioning platform*. Version 1.0. Oct. 2022. URL: https://github.com/darwinquezada/indoorsky.

[219] Francesco Potortì, Joaquín Torres-Sospedra, Darwin Quezada-Gaibor, Antonio Ramón Jiménez, Fernando Seco, Antoni Pérez-Navarro, Miguel Ortiz, Ni Zhu, Valerie Renaudin, Ryosuke Ichikari, Ryo Shimomura, Nozomu Ohta, Satsuki Nagae, Takeshi Kurata, Dongyan Wei, Xinchun Ji, Wenchao Zhang, Sebastian Kram, Maximilian Stahlke, Christopher Mutschler, Antonino Crivello, Paolo Barsocchi, Michele Girolami, Filippo Palumbo, Ruizhi Chen, Yuan Wu, Wei Li, Yue Yu, Shihao Xu, Lixiong Huang, Tao Liu, Jian Kuang, Xiaoji Niu, Takuto Yoshida, Yoshiteru Nagata, Yuto Fukushima, Nobuya Fukatani, Nozomi Hayashida, Yusuke Asai, Kenta Urano, Wenfei Ge, Nien-Ting Lee, Shih-Hau Fang, You-Cheng Jie, Shawn-Rong Young, Ying-Ren

Chien, Chih-Chieh Yu, Chengqi Ma, Bang Wu, Wei Zhang, Yankun Wang, Yonglei Fan, Stefan Poslad, David R. Selviah, Weixi Wang, Hong Yuan, Yoshitomo Yonamoto, Masahiro Yamaguchi, Tomoya Kaichi, Baoding Zhou, Xu Liu, Zhining Gu, Chengjing Yang, Zhiqian Wu, Doudou Xie, Can Huang, Lingxiang Zheng, Ao Peng, Ge Jin, Qu Wang, Haiyong Luo, Hao Xiong, Linfeng Bao, Pushuo Zhang, Fang Zhao, Chia-An Yu, Chun-Hao Hung, Leonid Antsfeld, Boris Chidlovskii, Haitao Jiang, Ming Xia, Dayu Yan, Yuhang Li, Yitong Dong, Ivo Silva, Cristiano Pendão, Filipe Meneses, Maria João Nicolau, António Costa, Adriano Moreira, Cedric De Cock, David Plets, Miroslav Opiela, Jakub Džama, Liqiang Zhang, Hu Li, Boxuan Chen, Yu Liu, Seanglidet Yean, Bo Zhi Lim, Wei Jie Teo, Bu Sung Lee, and Hong Lye Oh. "Off-Line Evaluation of Indoor Positioning Systems in Different Scenarios: The Experiences From IPIN 2020 Competition". In: *IEEE Sensors Journal* 22.6 (2022), pp. 5011–5054. DOI: 10.1109/JSEN.2021.3083149.

[220] Lucie Klus, Darwin Quezada-Gaibor, Joaquín Torres-Sospedra, Elena Simona Lohan, Carlos Granell, and Jari Nurmi. "Towards Accelerated Localization Performance Across Indoor Positioning Datasets". In: *2022 International Conference on Localization and GNSS (ICL-GNSS)*. 2022, pp. 1–7. DOI: 10.1109/ICL-GNSS54081.2022.9797035.

[221] Joaquín Torres-Sospedra, Ivo Silva, Lucie Klus, Darwin Quezada-Gaibor, Antonino Crivello, Paolo Barsocchi, Cristiano Pendão, Elena Simona Lohan, Jari Nurmi, and Adriano Moreira. "Towards Ubiquitous Indoor Positioning: Comparing Systems across Heterogeneous Datasets". In: *2021 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2021, pp. 1–8. DOI: 10.1109/IPIN51156.2021.9662560.

[222] Joaquín Torres-Sospedra, Fernando J. Aranda, Fernando J. Álvarez, Darwin Quezada-Gaibor, Ivo Silva, Cristiano Pendão, and Adriano Moreira. "Ensembling Multiple Radio Maps with Dynamic Noise in Fingerprint-based Indoor Positioning". In: *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*. 2021, pp. 1–5. DOI: 10.1109/VTC2021-Spring51267.2021.9448947.

[223] Francesco Furfari, Antonino Crivello, Paolo Baronti, Paolo Barsocchi, Michele Girolami, Filippo Palumbo, Darwin Quezada-Gaibor, Germán M. Mendoza Silva, and Joaquín Torres-Sospedra. "Discovering location based ser-

vices: A unified approach for heterogeneous indoor localization systems". In: *Internet of Things* 13 (2021), p. 100334. ISSN: 2542-6605. DOI: https://doi.org/10.1016/j.iot.2020.100334. URL: https://www.sciencedirect.com/science/article/pii/S2542660520301657.

[224] Joaquín Torres-Sospedra, Darwin Quezada-Gaibor, Germán M. Mendoza-Silva, Jari Nurmi, Yevgeni Koucheryavy, and Joaquín Huerta. "New Cluster Selection and Fine-grained Search for k-Means Clustering and Wi-Fi Fingerprinting". In: *2020 International Conference on Localization and GNSS (ICL-GNSS)*. 2020, pp. 1–6. DOI: 10.1109/ICL-GNSS49876.2020.9115419.

APPENDIX A APPENDIX

A.1 Systematic Review

Table A.1 Analysed articles Chapter 2 – Reproduced with the permission from [32].

ARTICLE	YEAR	TECHNOLOGY	TECHNIQUE	ALGORITHM	INDOOR	OUTDOOR	AREA	METRIC/ERROR
[103]	2015	RFID	N/A	N/A	✓	✓	N/A	N/A
[62]	2015	Inertial Sensors, Camera	Fusion techniques	Filter-base (low and high-pass filter)	✓	✗	N/A	N/A
[35]	2015	N/A	Range based, Range free, Fingerprinting	N/A	✓	✗	N/A	Mean error $2.82m$ to $4.26m$
[77]	2015	Wi-Fi	Fingerprinting, ToA	LQI, Pompeiu-Hausdorff	✓	✗	$22.5m^2$, $11m^2$ and $5m^2$ per point	Mean error $2.82m$ to $4.26m$
[2]	2016	Bluetooth	N/A	N/A	✓	✓	N/A	N/A
[108]	2016	N/A	Path planning	Multi-access Point (MaP algorithms)	✓	✗	N/A	N/A
[88]	2016	Wi-Fi, RFID	Proximity	N/A	✓	✗	$32m \times 12m$ and $21m \times 20m$	
[48]	2016	Wi-Fi	Fingerprinting	Probabilistic-Fingerprinting (P-FP)	✓	✗	N/A	N/A
[57]	2016	Bluetooth	N/A	N/A	✓	✓	N/A	N/A
[58]	2016	N/A	N/A	k-NN	✓	✗	N/A	N/A
[118]	2016	ZigBee	Multilateration	N/A	✓	✗	N/A	N/A
[75]	2017	Wi-Fi	Fingerprinting	k-NN	✓	✗	N/A	N/A
[40]	2017	Bluetooth	Proximity, Fingerprinting	k-NN	✓	✗	N/A	N/A
[105]	2017	Wi-Fi, Mobile Network	Statistical Approximation, AAL	N/A	✓	✓	N/A	N/A
[17]	2017	Wi-Fi, Inertial Sensors, Bluetooth, Mobile Network	Deep Learning, Signal visualization, Scene Analysis, Triangulation	N/A	✓	✓	N/A	COEX env. Mean error 4.16m, Store 3.54m
[53]	2017	Wi-Fi	Probabilistic, Bayesian theory	N/A	✓	✗	$62.22m \times 10.23m$	Average error from 1m to 2m
[100]	2017	Bluetooth, Inertial Sensors, Mobile Network, Wi-Fi, Camera	Path planning	N/A	✓	✓	Indoor $175m^2$, Outdoor 15km, $125m^2$	1-3m
[80]	2017	Bluetooth	N/A	N/A	✓	✗	N/A	N/A
[89]	2017	Camera, RFID	ToF	N/A	✓	✓	N/A	NA
[101]	2017	N/A	Probabilistic	Markov	✓	✗	$12m \times 12m$	N/A
[45]	2017	Camera, Inertial Sensors	Image Based, Structure from Motion (SfM) technique, Path planning	N/A	✓	✓	$150m^2$	$1m$
[119]	2017	Wi-Fi, Bluetooth	Fingerprinting, ML, Trilateration	SVM	✓	✗	N/A	Average distance error 11.48 ft.
[134]	2017	N/A	ML	Genetic Algorithm	✓	✓	N/A	Accuracy > 98%

Table A.2 Analysed articles Chapter 2 – Continuation – Reproduced with the permission from [32].

ARTICLE	YEAR	TECHNOLOGY	TECHNIQUE	ALGORITHM	INDOOR	OUTDOOR	AREA	METRIC/ERROR
[82]	2018	Bluetooth	Geometric approach, triangulation	N/A	✓	✗	N/A	N/A
[49]	2018	ZigBee, Bluetooth	Proximity, Waypoint-based navigation	N/A	✓	✗	3 meters	
[99]	2018	Bluetooth, Wi-Fi	Probabilistic	N/A	✓	✗	$8m \times 8m$ and $44m \times 44m$	Maximum error 5.94%
[63]	2018	Wi-Fi	Fuzzy logic, Trilateration, Fingerprinting	Genetic algorithms	✓	✗	N/A	Mean error ≈ 2.11 m± 0.6m
[5]	2018	Bluetooth	Proximity	N/A	✓	✗	N/A	N/A
[56]	2018	Wi-Fi, Bluetooth, RFID, Cellular	Fingerprint, Proximity	N/A	✓	✓	N/A	Mean error 4.62 m ± 0.31 m
[42]	2018	Bluetooth, Inertial Sensors	ML, image processing	Brute-Force Marching and ORB descriptors	✓	✗	N/A	N/A
[38]	2018	Bluetooth	ML	k-NN, SVM	✓	✓	64m2	1m
[104]	2018	N/A	N/A	N/A	✓	✓	N/A	N/A
[135]	2018	N/A	N/A	Hidden Markov Model	✓	✗	N/A	N/A
[36]	2018	Camera, Ultrasound	inertial Sensors	N/A	✓	✗	Accuracy > 97%	
[6]	2019	Wi-Fi	ML	Support Vector Regression	✓	✗	In a mall, $2500m^2$, and $562000m^2$	N/A
[55]	2019	Wi-Fi, Bluetooth, ZigBee	N/A	RACIL algorithm	✓	✗	Exp. $100m^2$, Real $2m \times 40m$	Simulated $0.2m$ to $1.1m$, Real $0.4m$ to $1.6m$
[78]	2019	Wi-Fi	ML, Fingerprinting	Multi-Objective Evolutionary Algorithm, W k-NN	✓	✗	N/A	Average error $1m$
[67]	2019	Wi-Fi, Bluetooth,	Proximity	N/A	✓	✗	N/A	N/A
[46]	2019	Bluetooth, Wi-Fi	ML	LSTM	✓	✗	$68m \times 16m$, $34 \times 16m$, $26.5m \times 16m$, $19m \times 16m$	N/A
[109]	2019	Bluetooth, Wi-Fi, Inertial Sensors	Fingerprinting, PDR, Map Matching	Particle Filter	✓	✗	N/A	Mean error $2.34m$
[18]	2019	Wi-Fi	ML	ELM-based	✓	✗	$12m \times 6m$, $8.7m \times 55m$	$15m$
[50]	2019	Bluetooth	Proximity	N/A	✓	✗	N/A	N/A
[64]	2019	Wi-Fi	Probabilistic	Motley Keenan	✓	✗	N/A	N/A
[90]	2019	Wi-Fi, Inertial Sensors, Geomagnetic	Deterministic	k-NN, Dynamic Time Warping (DTW),PF (Particle Filter)	✓	✗	N/A	$5cm$
[98]	2019	Light, Inertial Sensors	N/A	Peak Intensity detection, IIR, Filter, DTW	✓	✗	$1000m^2$, $20000m^2$, $800m^2$	Accuracy 98
[3]	2019	Wi-Fi	Neural Networks, Image Based	Genetic Algorithm	✓	✓	$\approx 4Km$	$1m$ to $5m$
[71]	2019	Bluetooth	ML, Probabilistic, Winsorization technique	Trimmed mean	✓	✓	$10m \times 4m$, $20 \times 2m$	$1m$
[106]	2019	Wi-Fi	Triangulation	N/A	✓	✗	$120m \times 120m$	$< 5.09m$
[11]	2019	UWB,Inertial Sensors,Wi-Fi	ML, Markov	N/A	✓	✗	$39m \times 18m$	Accuracy 90
[7]	2019	Wi-Fi, Inertial Sensors	Proximity	Nearest-checkpoint identification	✓	✓	N/A	N/A
[76]	2019	Wi-Fi	Experience-based	Heuristic algorithm, GBOMD, EBOP	✓	✗	N/A	N/A
[59]	2019	Wi-Fi	Fingerprinting	k-NN, etc.	✓	✗	N/A	N/A
[111]	2019	Wi-Fi, Inertial Sensors	N/A	Light-Weight Magnetic-Based Door Event Detection method	✓	✗	N/A	Detection accuracy 90
[72]	2019	Wi-Fi	Fingerprinting	W k-NN	✓	✗	$42m \times 12m$	

Table A.3 Analysed articles Chapter 2 – Continuation – Reproduced with the permission from [32].

ARTICLE	YEAR	TECHNOLOGY	TECHNIQUE	ALGORITHM	INDOOR	OUTDOOR	AREA	METRIC/ERROR
[44]	2019	Bluetooth	N/A	Bounding Box Algorithm	✓	✗	$36m \times 36m$	Average error $1.55m$
[61]	2020	Bluetooth	Proximity	N/A	✓	✓	$42.5m^2$	Mean accuracy 97.7
[60]	2020	Bluetooth	N/A	N/A	✓	✓	N/A	N/A
[8]	2020	Bluetooth	Proximity	N/A	✓	✗	N/A	$\approx 2.6m$
[70]	2020	Audible Sound	ML	k-NN, SVM, Naïve Bayes (NB)	✓	✗	-	Accuracy 71
[37]	2020	Bluetooth	ML, Trilateration	N/A	✓	✗	$12m \times 16m$	RMSE $0.86m$
[16]	2020	Wi-Fi	Markov model	N/A	✓	✗	N/A	N/A
[92]	2020	Camera	AR technique	N/A	✓	✓	N/A	N/A
[93]	2020	Camera, Ultrasound	Fuzzy logic, image processing	N/A	✓	✓	N/A	N/A
[86]	2020	ZigBee	N/A	Oriented FAST and Rotate BRIEF (ORB) algorithm	✓	✗	N/A	N/A
[10]	2020	UWB	ML, image processing	Brute-Force Marching and ORB descriptors	✓	✗	$10m \times 10m \times 3.3m$	N/A
[31]	2020	Camera	Visual-SLAM	N/A	✓	✗	N/A	Mean error $\approx 20cm$
[110]	2020	Bluetooth	ML, Proximity, Trilateration,	LSTM, RNN	✓	✓	N/A	N/A
[81]	2020	Bluetooth	ML	N/A	✓	✓	$2.50m \times 3.29m$, $2.50m \times 1.00m$, $2.34m \times 2.21m$, $5.60m \times 7.80m$, $1.60m \times 5.60m$	Average error $35.23cm$ ± 11.86
[73]	2020	Wi-Fi, Bluetooth, Mobile Network	N/A	k-NN, k-d Tree	✓	✓	$1.48km^2$	N/A
[15]	2020	Wi-Fi, Inertial Sensors, Bluetooth, UWB	ML	N/A	✓	✓	N/A	N/A
[107]	2020	Wi-Fi	ML, Fingerprinting	Manifold Alignment algorithm	✓	✗	68.9 ft × 52.5 ft	N/A
[54]	2020	N/A	ML, Fingerprinting	k-NN, SVM, NN, RF, MLP	✓	✗	N/A	N/A
[94]	2020	Camera	ML	DNN	✓	✓	42m×37m, 17m×13m, 8m×5m	$60cm$
[39]	2020	Wi-Fi	multidimensional spatial similarity (MDSS), k-NN	N/A	✓	✗	10m × 10m	Positioning error from 0.037 to 0.269 m
[85]	2020	Mobile Network	eMBB, mMTC, URLLC	N/A	✓	✗	N/A	N/A
[120]	2021	Bluetooth	ML	ANN-SVM, KWNN	✓	✗	N/A	Accuracy > 91%
[52]	2021	Wi-Fi	Fingerprinting	kNN, RLAEW	✓	✗	N/A	Mean error 2.67m
[136]	2021	Wi-Fi	Fingerprinting	Reputation Mechanism	✓	✗	N/A	N/A
[51]	2021	Wi-Fi	Fingerprinting	Dynamic Routing Algorithm of CapsNet	✓	✗	N/A	Average localization error 7.93m
[12]	2021	Light	N/A	Visible Light Positioning algorithm	✓	✗	3.3m × 3.15 m	Positioning error from 3 to 6 m
[121]	2021	Wi-Fi	N/A	classical multidimensional scaling (CMDS)	✓	✗	$2400m^2$	80 percentile 3m
[91]	2021	Inertial Sensors	Pattern matching technique	Dijkstra's algorithm	✓	✗	N/A	mean error 7.39m
[122]	2021	Bluetooth	N/A	Levenberg-Marquardt algorithm	✓	✗	5m × 5m approx.	Mean error < 1.7m
[123]	2021	RFID	Fingerprinting	kNN	✓	✗	11.2 × 13.40m	lowest error 78.3cm
[121]	2021	Wi-Fi, Inertial Sensors	classical multidimensional scaling	classical multidimensional scaling	✓	✗	3 Buildings, 2400m2	maximum error approx 4m
[133]	2021	RFID	N/A	N/A	✓	✗	N/A	N/A

Table A.4 Analysed articles Chapter 2 – Continuation – Reproduced with the permission from [32].

ARTICLE	YEAR	TECHNOLOGY	TECHNIQUE	ALGORITHM	INDOOR	OUTDOOR	AREA	METRIC/ERROR
[124]	2021	Bluetooth	N/A	Pathloss model	✓	✗	N/A	N/A
[121]	2021	Wi-Fi, Inertial Sensors	classical multi-dimensional scaling	classical multi-dimensional scaling	✓	✗	3 Buildings, 2400m2	maximum error approx 4m
[133]	2021	RFID	N/A	N/A	✓	✗	N/A	N/A
[124]	2021	Bluetooth	N/A	Pathloss model	✓	✗	N/A	N/A
[142]	2021	Wi-Fi, Bluetooth, ZigBee	N/A	Indoor Localisation Latency Centralised Offloading (ILLCO)	✓	✗	50 x 50m	N/A
[125]	2021	Wi-Fi	N/A	Inner Product Encryption (IPE)	✓	✗	N/A	Time response 0.1s, positioning error < 6m
[143]	2021	Wi-Fi	Fingerprinting	KNN	✓	✗	UJIIndoorloc 108,703 m2	One building average error 2.67m
[137]	2021	Light	N/A	VLP	✓	✗	N/A	positioning error > 5m
[126]	2021	N/A	Trilateration	VANET positioning based, Security protocols, Pub-pos and Pri-pos	✓	✓	N/A	Accuracy < 1.5 m
[139]	2021	Inertial Sensors	ML	N/A	✓	✗	N/A	95 percentile localisation error 12 cm
[127]	2021	Bluetooth&Inertial Sensors	location retrospective adjustment	location retrospective adjustment	✓	✗	9,000 m2	average accuracy 0.65m
[128]	2021	Bluetooth	ML	Kalman Filter	✓	✗	N/A	Accuracy 97.1%
[140]	2021	Wi-Fi&Inertial Sensors	ML	SVM, CNN	✓	✗	3 x 2.5 and 6 x 3m	Accuracy 96.5%
[130]	2021	Bluetooth	Dijkstra	Dijkstra	✓	✗	43.4 x 110.3m	N/A
[141]	2021	Wi-Fi	Fingerprinting	kNN,etc.	✓	✗	N/A	positioning error < 2.32m
[120]	2021	Bluetooth	Fingerprinting, ML	SVM, KNN	✓	✗	N/A	Localisation accuracy 92.79%
[131]	2021	Wi-Fi	ML	SVM, KNN, MLP, ARIMA	✓	✗	N/A	N/A
[91]	2021	Inertial Sensors	Dijkstra	Dijkstra	✓	✗	N/A	deviation < 1m
[132]	2021	N/A	N/A	N/A	N/A	N/A	N/A	N/A
[144]	2022	Wi-Fi&Bluetooth	Fingerprinting	adp-FSELM	✓	✗	7.4 x 12.5, and 14.95 x 31.8 m	error > 10m
[138]	2022	Wi-Fi	Fingerprinting	CapsNet	✓	✗	UJIIndoorloc 108,703 m2	average error 7.93
[129]	2022	N/A	ML	N/A	✓	✗	N/A	N/A
[32]	2022	N/A	N/A	N/A	N/A	N/A	N/A	N/A

A.2 Database

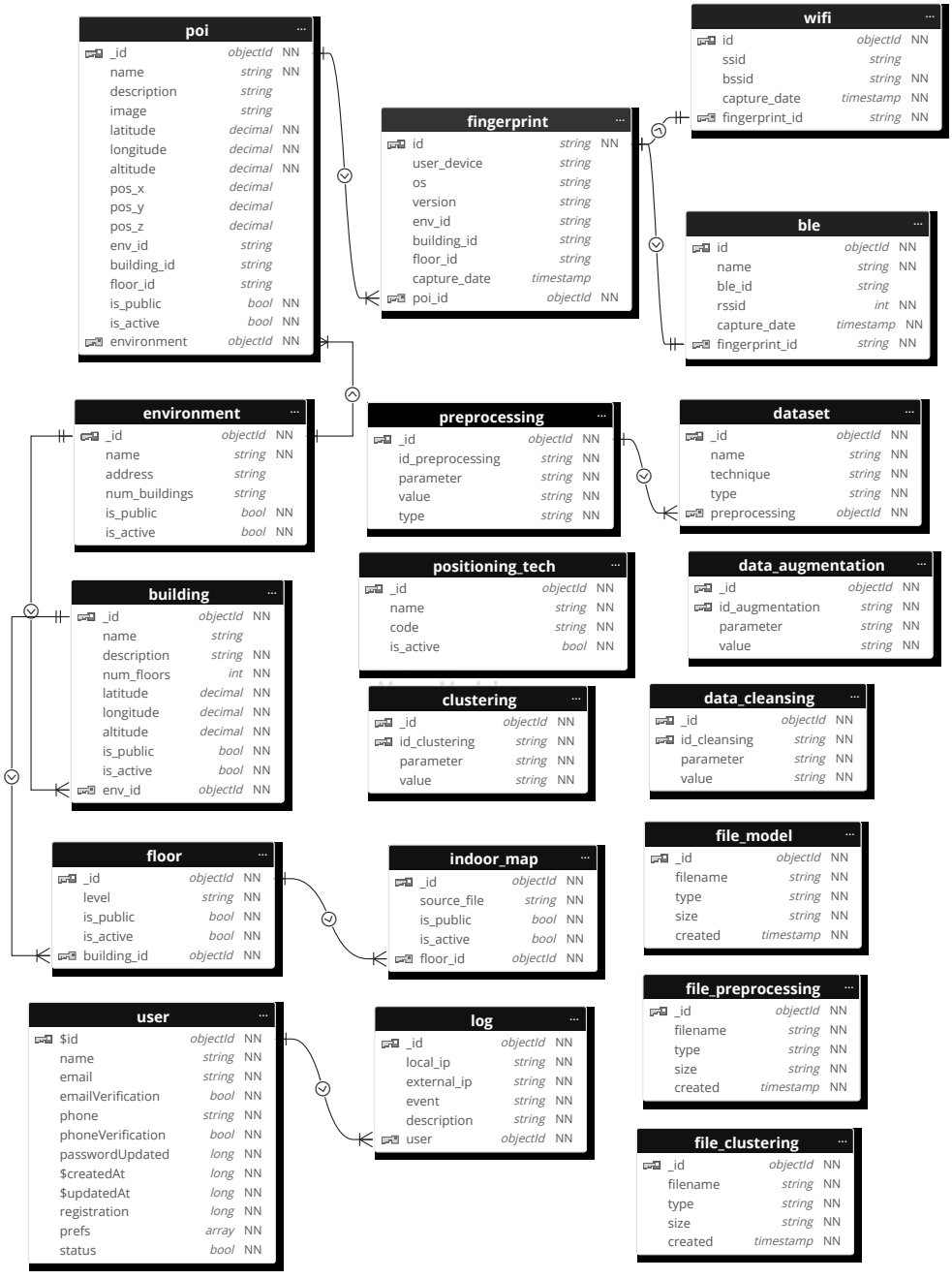


Figure A.1 Database – tables.

A.3 APIs

Table A.5 API - endpoints

Microservice	Request Method	Endpoint
Authentication	GET	/api/v1/auth/{*token*}
User	POST	/api/v1/user
	GET	/api/v1/user/{*user_id*}
	DELETE	/api/v1/user/user_id/delete
	PUT	/api/v1/user/user_id/update/email
	PUT	/api/v1/user/user_id/update/email_verification
	PUT	/api/v1/user/user_id/update/name
	PUT	/api/v1/user/user_id/update/password
	PUT	/api/v1/user/user_id/update/phone
	PUT	/api/v1/user/user_id/update/phone_verification
	PUT	/api/v1/user/user_id/update/status
Positioning	POST	/api/v1/position/{*env_id*}
Localisation	POST	/api/v1/localisation/{*env_id*}
Environment	POST	/api/v1/environment
	GET	/api/v1/environment/all
	GET	/api/v1/environment/{*env_id*}
	GET	/api/v1/environment/{*name*}/name
	DELETE	/api/v1/environment/{*env_id*}/delete
	PUT	/api/v1/environment/{*env_id*}/update
Building	POST	/api/v1/building/{*env_id*}
	GET	/api/v1/building/{*building_id*}
	GET	/api/v1/building/{*name*}/name
	DELETE	/api/v1/building/{*building_id*}/delete
	PUT	/api/v1/building/{*building_id*}/update
Floor	POST	/api/v1/floor/
	GET	/api/v1/floor/{*floor_id*}
	GET	/api/v1/floor/{*level*}/level
	DELETE	/api/v1/floor/{*floor_id*}/delete
	PUT	/api/v1/floor/{*floor_id*}/update
POIs	POST	/api/v1/poi/
	GET	/api/v1/poi/{*poi_id*}
	GET	/api/v1/poi/{*name*}/name
	DELETE	/api/v1/poi/{*poi_id*}/delete
	PUT	/api/v1/poi/{*poi_id*}/update

Table A.6 API - endpoints – Continuation

Microservice	Request Method	Endpoint
Fingerprints	POST	/api/v1/fingerprint/
	GET	/api/v1/fingerprint/{*env_id*}/environment
	GET	/api/v1/fingerprint/{*building_id*}/building
	GET	/api/v1/fingerprint/{*floor_id*}/floor
	GET	/api/v1/fingerprint/{*fingerprint_id*}
	DELETE	/api/v1/fingerprint/{*env_id*}/environment/delete
	DELETE	/api/v1/fingerprint/{*building_id*}/building/delete
	DELETE	/api/v1/fingerprint/{*floor_id*}/floor/delete
	DELETE	/api/v1/fingerprint/{*fingerprint_id*}/delete
WiFi	POST	/api/v1/wifi
	GET	/api/v1/wifi/{*wifi_id*}
	GET	/api/v1/wifi/{*fingerprint_id*}/fingerprint
	DELETE	/api/v1/wifi/{*wifi_id*}/delete
	DELETE	/api/v1/wifi/{*fingerprint_id*}/fingerprint/delete
BLE	POST	/api/v1/ble
	GET	/api/v1/ble/{*ble_id*}
	GET	/api/v1/ble/{*fingerprint_id*}/fingerprint
	DELETE	/api/v1/ble/{*ble_id*}/delete
	DELETE	/api/v1/ble/{*fingerprint_id*}/fingerprint/delete
Positioning Tech.	POST	/api/v1/pos_tech
	GET	/api/v1/pos_tech/{*pos_tech_id*}/
	GET	/api/v1/pos_tech/{*name*}/name
	DELETE	/api/v1/pos_tech/{*pos_tech_id*}/delete
	PUT	/api/v1/pos_tech/{*pos_tech_id*}/update
Data Preprocessing	POST	/api/v1/preprocess
	GET	/api/v1/preprocess/{*env_id*}/environment
	GET	/api/v1/preprocess/{*preprocessing_id*}
	DELETE	/api/v1/preprocess/{*preprocessing_id*}/delete
Data Cleansing	POST	/api/v1/cleansing/clean_dataset
	GET	/api/v1/cleansing/{*cleansing_id*}
	GET	/api/v1/cleansing/{*env_id*}/environment
	GET	/api/v1/cleansing/{*name*}/name
	DELETE	/api/v1/cleansing/{*cleansing_id*}/delete
Data Augmentation	POST	/api/v1/augmentation/data_augmentation
	GET	/api/v1/augmentation/{*augmentation_id*}
	GET	/api/v1/augmentation/{*env_id*}/environment
	GET	/api/v1/augmentation/{*name*}/name
	DELETE	/api/v1/augmentation/{*augmentation_id*}/delete

Table A.7 API - endpoints – Continuation

Microservice	Request Method	Endpoint
Train Models	POST	/api/v1/model/cnn_lstm
	POST	/api/v1/model/cnn_elm
	GET	/api/v1/model/{*model_id*}
	GET	/api/v1/model/{*name*}/name
	DELETE	/api/v1/model/{*model_id*}/delete
Audit	POST	/api/v1/audit/
	GET	/api/v1/audit/{*audit_id*}