

Universiteit Leiden

Master Computer Science

A BLE RSSI Fingerprinting Pipeline for 3D Indoor Localisation on a Crownstone Mesh using Machine Learning

Name:	Praneeth Dathu
Student ID:	s4174089
Date:	11-08-2026
Specialisation:	Artificial Intelligence
1st supervisor:	Dr. Erwin M. Bakker (LIACS)
2nd supervisor:	Prof. dr. M.S. Lew (LIACS)
Daily supervisor:	Teresa Jacqueline van Dam (Almende)

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Contents

List of Figures iii

List of Tables iii

List of abbreviations v

Acknowledgements vi

Abstract vii

1 Introduction 1
- 1.1 Research questions . . . 2
- 1.2 Approach and scope . . . 2
- 1.3 Contributions . . . 3
- 1.4 Thesis outline . . . 4

2 Background 5
- 2.1 Bluetooth Low Energy advertising and RSSI measurements . . . 6
- 2.2 Fingerprinting for indoor localisation . . . 6
- 2.3 Model families for fingerprint classification . . . 8
- 2.4 The split protocol . . . 9
- 2.5 Performance measures . . . 9

3 Related Work 12
- 3.1 Positioning technologies and deployment constraints . . . 12
- 3.2 Learning from RSSI fingerprints: classical models and deep neural networks 12
- 3.3 Physics-based, hybrid, and post-processing methods . . . 13
- 3.4 The research gap . . . 14

4 Initial Study: Room-Level Localisation 16

5 The Crownstone 3D Dataset (CS3D) 18
- 5.1 Deployment and ground-truth grid . . . 18
- 5.2 Collection protocol . . . 22
- 5.3 From raw log to learning examples . . . 23
- 5.4 Scope . . . 24

6 System Methodology 25
- 6.1 Model families . . . 26
- 6.2 Windowed statistical features . . . 29
- 6.3 Physics-derived features and per-receiver calibration . . . 29
- 6.4 Training protocol . . . 32

7 Experimental Setup 33
- 7.1 Evaluation protocol . . . 33
- 7.2 Metrics . . . 34
- 7.3 Experiments on CS3D . . . 35

8 Results **36**
8.1 Experiment 1: Model families . . . 36
8.2 Experiment 2: Best-performing classical versus deep neural network . . . 39
8.3 Experiment 3: Window-size ablation . . . 41
8.4 Experiment 4: Rooms and heights (3D) . . . 43
8.5 Experiment 5: Random versus chronological split . . . 43

9 Evaluation on Public Datasets and Published References **46**
9.1 HDLC published-reference check . . . 46
9.2 Effect of the split protocol . . . 47
9.3 Ablation of the engineered features . . . 47
9.4 Evaluation on the Daniş dataset . . . 47
9.5 Summary . . . 48

10 Discussion and Synthesis **49**
10.1 RQ1: Accuracy . . . 49
10.2 RQ2: Deployment characteristics . . . 50
10.3 RQ3: Evaluation protocol . . . 50
10.4 Synthesis of the findings . . . 50

11 Limitations and Future Work **52**
11.1 Dataset and deployment . . . 52
11.2 Models and training . . . 52
11.3 Evaluation protocol . . . 53
11.4 Measurement and ground truth . . . 53
11.5 Future work . . . 53

12 Conclusion **55**

A Feature and model specification **61**
A.1 Per-anchor path-loss calibration . . . 61
A.2 Model hyperparameters . . . 61
A.3 Hyperparameter sensitivity . . . 63
A.4 Compute environment . . . 64
A.5 ONNX export and edge deployment . . . 64

B Extended results **66**
B.1 Per-fold cross-validation results . . . 66
B.2 Random-split protocol audit . . . 68
B.3 Training-set size . . . 68
B.4 Per-room results . . . 69

C Per-cell error: offline versus live **70**

D Post-processing filters **75**

List of Figures

1 A taxonomy of indoor localisation . . . 5
2 Conceptual view of BLE RSSI reception . . . 7
3 Random and chronological splitting of time-ordered windows . . . 9
4 Crownstone hardware form factors . . . 20
5 BLE asset tag . . . 21
6 The CS3D deployment . . . 21
7 Overview of the CS3D modelling pipeline . . . 26
8 Accuracy and error across model families at $K = 5$ . . . 37
9 ExtraTrees versus Transformer per fold . . . 40
10 Window-size ablation . . . 42
11 Offline per-cell 2D MAE, floor . . . 71
12 Offline per-cell 2D MAE, standing . . . 72
13 Live per-cell 2D MAE, floor . . . 73
14 Live per-cell 2D MAE, standing . . . 74

List of Tables

1 Structure of the thesis. . . . 4
2 Post-processing methods relevant to sequential localisation. . . . 14
3 From room-level to 3D grid-cell localisation . . . 17
4 Comparison of CS3D with public reference datasets . . . 19
5 Example of the CS3D raw log . . . 19
6 CS3D deployment and hardware summary . . . 20
7 CS3D collection protocol and raw-log summary . . . 22
8 Learning representations derived from the CS3D raw log . . . 23
9 Per-stage details of the CS3D modelling pipeline . . . 27
10 The seventeen model families . . . 28
11 The five per-receiver statistical features . . . 30
12 The physics-derived features . . . 30
13 Main results at $K = 5$ . . . 38
14 Main results at $K = 10$ . . . 39
15 Model size and inference latency . . . 41
16 Window-size ablation . . . 42
17 Random versus chronological split for ExtraTrees . . . 44
18 Selection and role of the public datasets . . . 46
19 HDLC published-reference check . . . 47
20 Split-protocol effect on HDLC and CS3D . . . 47
21 Feature ablation across datasets . . . 48
22 Evaluation on the Daniş trajectory dataset . . . 48
23 Synthesis of the research questions . . . 49
24 Per-anchor path-loss calibration . . . 61
25 Neural architectures . . . 62
26 Neural training settings . . . 62
27 Hyperparameter sensitivity . . . 63

28 ONNX against native model size . . . 65
29 Accuracy and error of the ExtraTrees edge variants . . . 65
30 Raspberry Pi ONNX deployment . . . 65
31 Per-fold 2D MAE, K=5 . . . 66
32 Per-fold 2D MAE, K=10 . . . 66
33 Per-fold grid accuracy, K=5 . . . 67
34 Per-fold grid accuracy, K=10 . . . 67
35 Random-split protocol audit . . . 68
36 Training-set size sweep . . . 68
37 ExtraTrees results by room . . . 69
38 Post-processing filters evaluated . . . 75
39 Offline filter results on the static per-cell survey . . . 76
40 Live filter results under a chronological hold-out . . . 76

List of abbreviations

Abbreviation	Meaning
AoA	Angle of Arrival
BLE	Bluetooth Low Energy
CNN	Convolutional Neural Network
CNN-LSTM	Hybrid Convolutional-Recurrent Network
CS3D	Crownstone 3D dataset (introduced in this work)
CSV	Comma-Separated Values
CV	Cross-Validation
FSPL	Free-Space Path Loss
GNSS	Global Navigation Satellite System
HDLC	Hybrid Wi-Fi and BLE fingerprinting dataset (public benchmark)
k-NN	k-Nearest Neighbours
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MB	Megabyte (decimal, 10^6 bytes)
MLP	Multilayer Perceptron
MQTT	Message Queuing Telemetry Transport
NLOS	Non-Line-of-Sight
PINN	Physics-Informed Neural Network
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RQ	Research Question
RSS	Received Signal Strength
RSSI	Received Signal Strength Indicator
SVM	Support Vector Machine
TCN	Temporal Convolutional Network
UART	Universal Asynchronous Receiver/Transmitter
UWB	Ultra-Wideband
WCL	Weighted Centroid Localisation
WLAN	Wireless Local Area Network

Acknowledgements

This thesis represents the culmination of a journey that has not only deepened my academic understanding but also contributed significantly to my personal and professional development. I extend my sincere gratitude to my supervisor, Dr. Erwin M. Bakker (LIACS), for his invaluable guidance and insightful feedback, as discussed throughout this project, which were instrumental in shaping this research.

I am deeply indebted to Almende B.V. for the opportunity to conduct this thesis, and for providing the resources and access to the Crownstone infrastructure upon which this thesis is built. I am particularly grateful to my company supervisor, Teresa Jacqueline van Dam (Almende), whose consistent mentorship, constructive discussions, and unwavering support were vital throughout this project. I also wish to thank the entire Almende team for fostering a welcoming and inclusive research environment. Furthermore, I appreciate the camaraderie of my fellow students at LIACS, with whom I shared this rewarding academic experience.

Exploring the field of machine learning for indoor localisation has been a fascinating endeavour that has taught me the value of persistence and rigour. Finally, this work would not have been possible without the encouragement and support of my family and friends, whose belief in me throughout the past year has been a constant source of strength.

Praneeth Dathu
Leiden, The Netherlands

Abstract

Bluetooth Low Energy (BLE) is widely used for indoor sensing, building automation, and asset tracking, but RSSI-based localisation remains difficult because indoor signal propagation is affected by multipath, fading, and obstruction. This thesis investigates RSSI fingerprinting and introduces the Crownstone 3D (CS3D) benchmark dataset: measurements from one BLE asset, seventeen permanently installed receivers, 42 floor-grid cells, and two tag heights in an active office. Seventeen classical machine-learning and deep-neural-network models were evaluated using chronological cross-validation.

On CS3D, the Transformer model achieved the lowest mean three-dimensional localisation error of 0.32 ± 0.31 m, compared with 0.41 ± 0.26 m for ExtraTrees. The difference was not statistically significant in a paired sign-flip test ($p = 0.187$). In matched-window comparisons, ExtraTrees achieved equal or lower localisation error, showing that the difference between the native configurations cannot be attributed to model architecture alone.

Two public datasets provided external context. On HDLC, the published k-nearest-neighbour mean two-dimensional error of 1.95 m was reproduced at 1.94 m. On the Daniş BLE trajectory dataset, k-nearest neighbours and ExtraTrees obtained errors of 1.36 m and 1.73 m, respectively, compared with the published hidden-Markov-model range of 1.97–2.51 m. Because the protocols and preprocessing differed, these are contextual rather than direct comparisons. On HDLC, random splitting reduced the reported error by up to approximately a factor of two relative to chronological splitting.

On a different-day live replay, the mean three-dimensional localisation error increased to 1.89 m. The same-session results therefore do not demonstrate cross-session generalisation. Overall, reported performance is sensitive to the evaluation protocol and input-window design, while robustness across measurement sessions remains an open challenge.

Keywords: Bluetooth Low Energy, RSSI fingerprinting, 3D indoor localisation, indoor positioning, ensemble learning, deep neural networks, chronological cross-validation, cross-session generalisation

Chapter 1

Introduction

Global navigation satellite systems (GNSS) provide reliable positioning outdoors, but their signals are weak or unavailable inside buildings. Indoor localisation therefore requires alternative technologies [62]. Dedicated systems such as ultra-wideband (UWB) can provide accurate positioning, but require additional hardware and installation effort [1]. A lower-cost alternative is to reuse Bluetooth Low Energy (BLE) infrastructure already installed in many buildings [62, 55].

When fixed BLE receivers record advertisements from a BLE tag, such as a small battery-powered beacon attached to an asset, the RSSI recorded by each receiver is influenced by the tag's position and the surrounding propagation environment. In fingerprinting, machine-learning models learn the mapping from recorded RSSI patterns to spatial positions without first converting RSSI values into distances using a physical propagation model [19, 55]. This thesis investigates whether such an infrastructure can support three-dimensional grid-cell localisation using only BLE RSSI measurements.

Several public datasets are available for RSSI-fingerprinting research [58, 37, 41]. However, the datasets considered here do not jointly provide a permanently installed BLE receiver infrastructure, explicit within-floor height labels, and raw timestamped measurements suitable for chronological evaluation. UJIIndoorLoc is Wi-Fi based and represents vertical position by floor [58]; the Mendoza-Silva BLE dataset is planar and uses beacons deployed specifically for positioning [37]; and HDLC combines Wi-Fi and BLE while also representing vertical position at floor level [41].

To address these gaps, the Crownstone 3D dataset (CS3D) was constructed. The RSSI data were collected in the Almende office in Rotterdam using 17 building-integrated Crownstone receivers at surveyed positions. Fifteen receivers were already installed for building automation, while two were added to improve coverage [35]. These fixed receivers are referred to as *anchors* throughout this thesis. A single BLE tag was placed at surveyed grid positions while the anchors recorded its advertisements over time.

A concern in RSSI-fingerprinting research is that reported performance can be optimistic when temporally related observations occur in both the training and test sets [8, 11]. The Microsoft Indoor Localization Competition also showed that many systems reporting sub-metre accuracy performed less accurately when evaluated in the same unfamiliar environment [34]. The main model comparison in this thesis therefore uses chronological splitting, while random splitting is examined separately to quantify its effect on reported performance.

A second consideration is whether greater model complexity provides a measurable benefit. Classical models such as tree ensembles are often competitive with deep neural networks on tabular data [24]. This thesis evaluates both tabular and sequential RSSI representations and therefore compares classical machine-learning and deep-neural-network models rather than assuming that one model family is inherently more suitable.

This was observed in an earlier study using the same infrastructure, in which the asset was localised at room level by determining which of four rooms it occupied [**?**]. A Temporal Convolutional Network achieved 99.81% accuracy offline but 76.6% during live evaluation. A single random split was used rather than a chronological split. That earlier study motivated the transition from room-level classification to finer-grained three-dimensional grid-cell localisation. In the present study, a chronological evaluation protocol is used.

1.1 Research questions

This thesis examines the localisation accuracy and computational cost of classical machine-learning and deep-neural-network models using a permanently installed BLE receiver infrastructure. The primary research question is:

> ***Using a permanently installed BLE receiver mesh, do deep neural networks localise a BLE asset more accurately than classical machine-learning models, and do any accuracy gains justify their associated training and deployment costs?***

The Crownstone deployment is treated as one instance of a permanently installed BLE receiver infrastructure. Because the required accuracy depends on the application, performance is reported as two-dimensional and three-dimensional localisation error in metres rather than against one fixed accuracy threshold. The primary question is examined through three subquestions:

- **RQ1 (Accuracy).** Under the same evaluation protocol, do deep neural networks achieve lower mean 2D and 3D localisation errors than the best classical machine-learning model?
- **RQ2 (Cost).** What are the deployment costs of each model in terms of inference latency, saved-model size, required input-window length, and preprocessing?
- **RQ3 (Protocol).** How does chronological rather than random data splitting affect the measured performance of the models?

1.2 Approach and scope

To answer the research questions, this thesis develops a common experimental framework for transforming recorded BLE signals into position estimates. Tabular and sequential RSSI representations are used to compare classical machine-learning and deep neural network models on CS3D under chronological cross-validation. Random splitting is evaluated separately to measure its effect on reported performance.

Two public datasets provide external comparison. HDLC was selected because it is a public hybrid Wi-Fi and BLE fingerprinting dataset with a documented k-nearest-neighbour

reference method [41, 42]. The reproduced k-nearest-neighbour implementation obtained a mean two-dimensional error of 1.94 m on this dataset. The Daniş dataset was selected because it provides high-precision continuous ground truth for a moving BLE beacon in an independent building, together with published probabilistic-tracking results [17, 16]. On Daniş, k-nearest neighbours and ExtraTrees obtained mean two-dimensional errors of 1.36 m and 1.73 m, respectively, compared with reported hidden-Markov-model errors of 1.97–2.51 m under a different evaluation protocol. The public-dataset experiments therefore provide contextual comparisons rather than direct leaderboard rankings.

The primary CS3D evidence is deliberately limited to one BLE tag recorded during a single survey session on one office floor. The resulting same-session findings describe this deployment and do not establish cross-session or building-independent generalisation. The public-dataset experiments broaden the evaluation context but do not remove this limitation.

1.3 Contributions

This thesis makes four main contributions:

- **The CS3D dataset.** A three-dimensional grid-cell BLE RSSI fingerprinting dataset collected using 17 permanently installed Crownstone receivers in an active office. It provides ground-truth labels for 42 floor-grid cells at two tag heights, resulting in 84 position classes.
- **A tabular feature pipeline with per-anchor calibration.** A preprocessing and feature-extraction pipeline was developed and evaluated that produces 144 features per window: 85 statistical features and 59 physics-derived features. Path-loss parameters are calibrated separately for each anchor using the recorded data.
- **A systematic comparison of classical and deep-neural-network models.** Seventeen models are evaluated on CS3D using chronological K-fold cross-validation. The Transformer achieves the lowest error in its native configuration, while ExtraTrees is the strongest classical model. Their difference is examined using a paired sign-flip test and matched-window comparisons rather than being interpreted from fold variation alone.
- **Evaluation on public datasets.** Classical methods are evaluated on HDLC and Daniş using protocols appropriate to each dataset. The experiments assess baseline reproducibility, sensitivity to the data-splitting protocol, and performance in an independent building.

1.4 Thesis outline

Table 1 summarises the structure of the thesis.

Table 1: Structure of the thesis.

Chapter	**Content**
Chapter 2	BLE RSSI propagation, fingerprinting, and relevant evaluation pitfalls.
Chapter 3	Related indoor-positioning approaches and the research gap addressed by this thesis.
Chapter 4	The earlier room-level localisation study that motivated the three-dimensional task.
Chapter 5	The CS3D data collection, grid-based ground truth, and resulting windowed datasets.
Chapter 6	Preprocessing, feature extraction, path-loss calibration, and the evaluated model families.
Chapter 7	Evaluation metrics, cross-validation protocol, hypotheses, and experimental design.
Chapter 8	The CS3D model results, paired comparison, matched-window analysis, spatial analysis, and random-versus-chronological comparison.
Chapter 9	Comparisons with published methods on the HDLC and Daniş public datasets.
Chapter 10	Interpretation of the experimental findings and synthesis of the central claim.
Chapter 11	Limitations of the present work and directions for future research.
Chapter 12	Conclusions and answers to the research questions.

Chapter 2

Background

This chapter introduces the main concepts and methods used in this study:

1. Bluetooth Low Energy (BLE) advertising and indoor RSSI variation (Section 2.1);
2. fingerprinting for indoor localisation (Section 2.2);
3. the model families compared in the experiments (Section 2.3);
4. the evaluation protocol and performance measures (Sections 2.4 and 2.5).

The feature computations, model configurations, and evaluation procedures are specified in detail in Chapters 6 and 7.

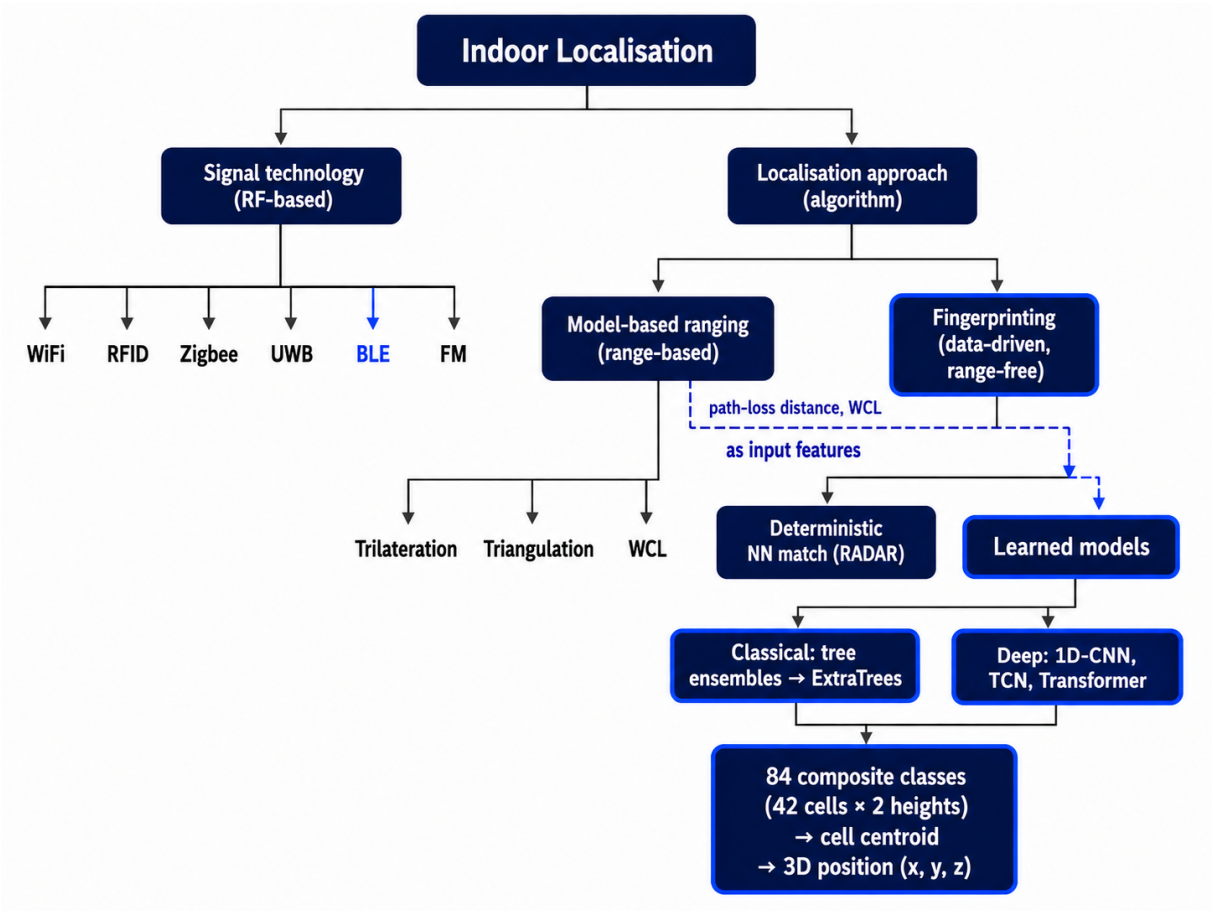


Figure 1: A taxonomy of indoor localisation organised by signal technology and localisation approach. The highlighted path summarises the design choices adopted in this thesis; representative model families are shown.

Figure 1 places the approach used in this thesis within the broader field of indoor localisation. The primary approach is BLE fingerprinting rather than direct model-based ranging. For the tabular models, RSSI statistics are supplemented with physics-derived features, including path-loss distance estimates and a weighted centroid localisation estimate. Classical machine-learning models and deep neural networks predict one of 84 composite position classes, representing 42 floor-grid cells at two tag heights. The predicted class is then mapped to its fixed three-dimensional centroid.

2.1 Bluetooth Low Energy advertising and RSSI measurements

Bluetooth Low Energy positioning can use the advertising mechanism, in which a BLE tag broadcasts short advertisement packets at regular intervals without establishing a connection or pairing. A fixed receiver that receives an advertisement records its Received Signal Strength Indicator (RSSI). RSSI is reported in dBm and provides a receiver-dependent indication of the received signal strength [53]. The localisation methods presented in this thesis derive their position estimates from these RSSI measurements.

Because the receivers only listen for advertisements, a localisation system can reuse BLE receiver infrastructure installed for other purposes [62]. The Crownstone deployment considered in this thesis uses this passive reception mechanism, as described in Chapter 5.

RSSI is a coarse and variable proxy for distance (Figure 2). Although signal strength generally decreases with distance, indoor measurements are affected by reflections from walls, furniture, and metal surfaces, which produce multipath propagation. Large obstacles cause shadowing, small movements produce fast fading, and the human body attenuates the 2.4 GHz signal when it blocks the direct path [19, 55, 62]. Consequently, repeated RSSI measurements at a fixed position form a distribution rather than a single stable value, and distributions from nearby positions may overlap. The methods therefore summarise measurements over short time windows rather than relying on individual packets, as described in Chapter 6.

Bluetooth 5.1 direction finding based on angle of arrival has been investigated for sub-metre positioning, but it requires compatible multi-antenna receiver hardware [13]. The installed Crownstone receivers do not provide this capability. The methods evaluated here therefore use RSSI measurements and RSSI-derived representations rather than angle-of-arrival information.

2.2 Fingerprinting for indoor localisation

Fingerprinting is a data-driven approach to indoor localisation. Instead of converting each signal-strength value directly into a distance, it treats the pattern of RSSI values observed at a position as that position's fingerprint. Fingerprinting normally consists of two phases. During the offline phase, a radio map is constructed by recording fingerprints at known, labelled positions. During the online phase, a fingerprint observed at an unknown position is compared with the radio map to estimate its location. An early example is the RADAR system, which matched an observed RSSI vector to stored fingerprints using

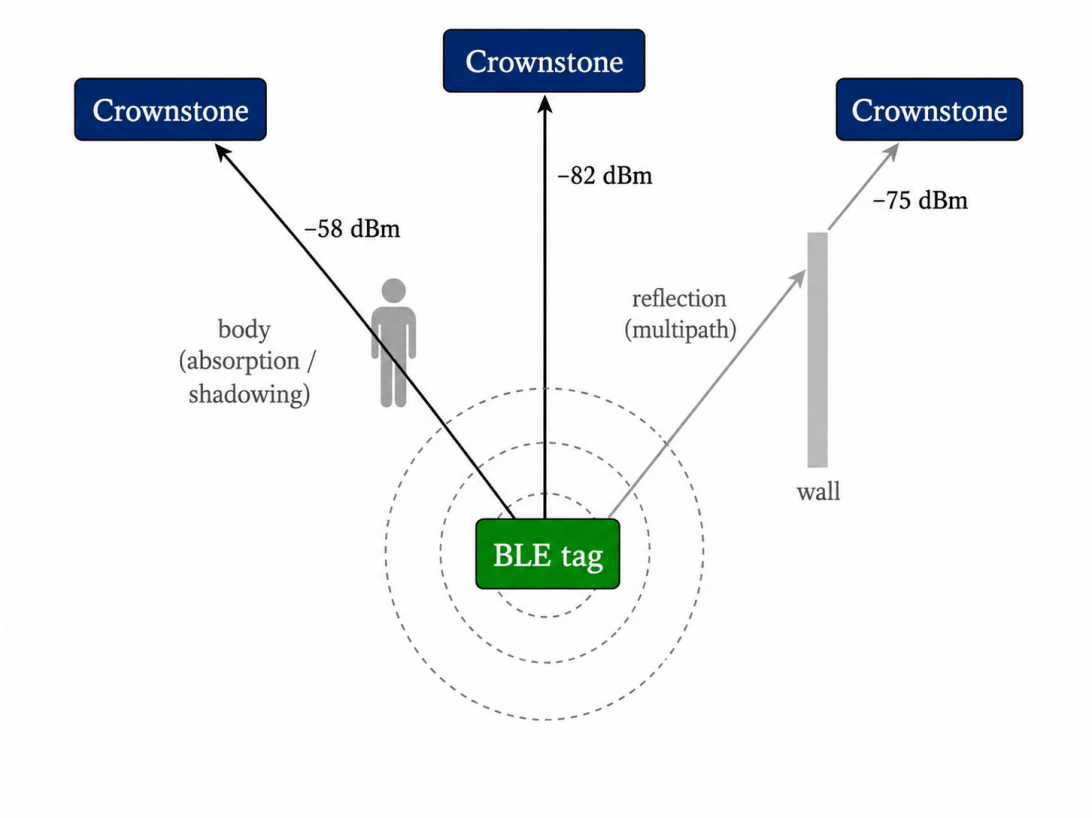


Figure 2: Conceptual view of BLE RSSI reception. A BLE tag broadcasts advertisement packets, and fixed receivers report RSSI values. Body obstruction and reflected signal paths can alter the reported values, so RSSI does not necessarily decrease smoothly with distance. The displayed values are illustrative.

nearest-neighbour search [6]. Because fingerprinting uses signal patterns recorded in the environment, it can accommodate effects such as multipath and shadowing that make direct RSSI-to-distance conversion unreliable [19].

In CS3D, fingerprinting is formulated as supervised classification. Each sample is derived from an RSSI observation window, and each label identifies a surveyed grid cell at one of two tag heights. For example, a window recorded while the tag is located in grid cell D1 at the lower height is assigned the composite class `D1_z0`. A model predicts one of the 84 composite classes rather than a continuous coordinate directly. The highest-scoring class is then decoded to the predefined three-dimensional centre, or centroid, of its grid cell and height.

For the tabular models, RSSI statistics are supplemented with physics-derived features: inverted path-loss distance estimates and a weighted centroid localisation (WCL) estimate. These features allow the models to combine empirical RSSI patterns with approximate propagation-based information. Their computation and per-anchor calibration are described in Chapter 6.

2.3 Model families for fingerprint classification

Representing localisation as classification over grid cells allows standard supervised-learning methods to be applied. The evaluated models are grouped here according to their learning mechanism and input representation. The complete model list and configurations are given in Chapter 6.

- **Classical baselines:** k-nearest neighbours (k-NN), a support vector machine (SVM), and a multilayer perceptron (MLP) [15, 14, 52].
- **Tree ensembles:** random forest, extremely randomised trees (ExtraTrees), XGBoost, LightGBM, and CatBoost. These models operate on the tabular feature representation computed for each observation window [10, 21, 12, 28, 46, 24].
- **Sequence models:** LSTM, one-dimensional CNN, TCN, CNN–LSTM, and Transformer models. These models process each window as an ordered RSSI sequence rather than as one tabular feature vector [26, 7, 59].

Two additional neural models incorporate information about the structure of the localisation problem. The model referred to as the physics-informed neural network (PINN) in this thesis is an MLP trained with an additional loss term based on the WCL position estimate. This term encourages, but does not force, predictions to remain close to the physics-derived estimate [49].

The anchor-attention model instead represents each receiver as an input token containing its mean RSSI and surveyed coordinates. Self-attention is then applied across receivers before the position class is predicted. Chapter 6 provides the complete architectures and training settings.

2.4 The split protocol

The central evaluation concern in this setting is temporal correlation. RSSI windows recorded close together in time often contain similar signal patterns. A random train–test split can therefore place temporally adjacent observations in both sets, leading to overly optimistic performance estimates [8, 11].

The main CS3D model comparison uses chronological within-class K-fold cross-validation. Within each composite class, windows are ordered by time and divided into K contiguous blocks. In each fold, one block is used for testing and the remaining blocks for training. This structure confines train–test adjacency to the two boundaries of the held-out block rather than allowing it throughout the fold.

Figure 3 contrasts chronological and random splitting. The complete splitting procedure, including the training gap used for sequence windows and the paired statistical test, is specified in Chapter 7.

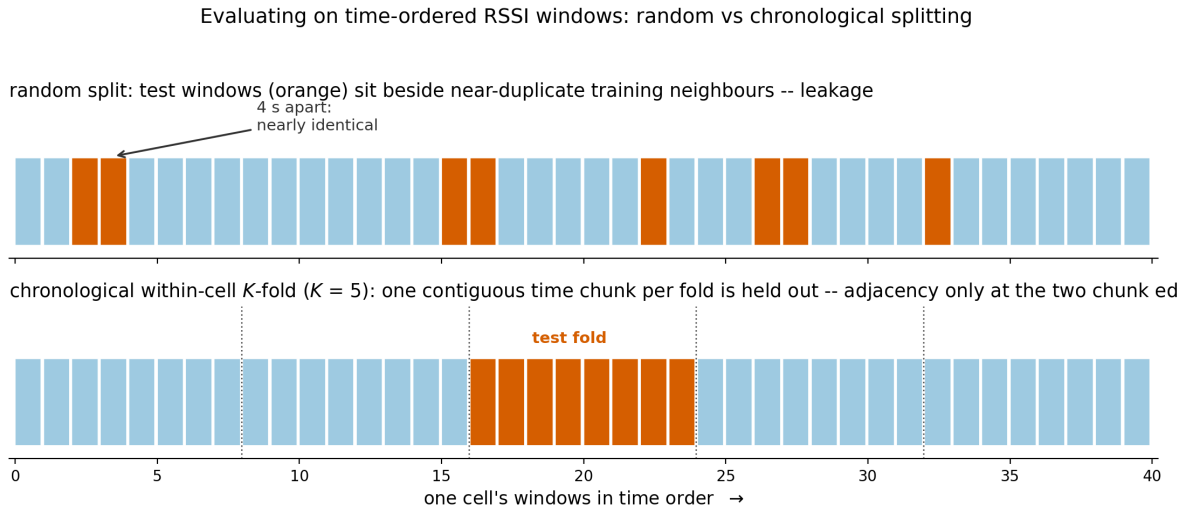


Figure 3: Random and chronological splitting of one class's time-ordered windows into training (blue) and test (orange) sets. In a random split, temporally adjacent windows can occur in both sets. In chronological within-class K-fold cross-validation, one contiguous block is held out in each fold ($K = 5$ shown), so train–test adjacency occurs only near the boundaries of the held-out block.

2.5 Performance measures

For each observation window i, a model produces a score s_{ic} for each of the $C = 84$ composite cell-height classes. The predicted class is the class with the highest score:

$$\hat{c}_i = \underset{c \in \{1,\ldots,C\}}{\arg\max}\, s_{ic}. \tag{1}$$

Each composite class c represents one grid cell at one of the two tag heights and has a predefined three-dimensional centroid

$$\mathbf{m}_c = (x_c, y_c, z_c), \tag{2}$$

where (x_c, y_c) is the planar centroid of the grid cell and z_c is either 0.05 m or 1.00 m. The decoded position prediction for window i is therefore

$$\hat{\mathbf{m}}_i = \mathbf{m}_{\hat{c}_i}. \tag{3}$$

Thus, the models predict a discrete composite class rather than a continuous coordinate directly. The class prediction is subsequently mapped to the fixed centroid of the corresponding grid cell and height level.

The CS3D experiments use three performance measures: grid accuracy, MAE 2D, and MAE 3D.

- **Grid accuracy** is the proportion of test windows for which the predicted composite class equals the true composite class:

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^{N} I(\hat{c}_i = c_i)\,, \tag{4}$$

  where N is the number of test windows and $I(\cdot)$ is the indicator function.

- **MAE 2D** is the mean horizontal Euclidean localisation error between the centroid of the predicted class and the centroid of the true class:

$$\text{MAE}_{\text{2D}} = \frac{1}{N} \sum_{i=1}^{N} \sqrt{(x_{\hat{c}_i} - x_{c_i})^2 + (y_{\hat{c}_i} - y_{c_i})^2}. \tag{5}$$

- **MAE 3D** is the mean three-dimensional Euclidean localisation error:

$$\text{MAE}_{\text{3D}} = \frac{1}{N} \sum_{i=1}^{N} \sqrt{(x_{\hat{c}_i} - x_{c_i})^2 + (y_{\hat{c}_i} - y_{c_i})^2 + (z_{\hat{c}_i} - z_{c_i})^2}. \tag{6}$$

The terms MAE 2D and MAE 3D are retained for consistency with the experimental tables and figures. Mathematically, however, they are mean Euclidean localisation errors rather than coordinate-wise mean absolute errors. Both measures are reported in metres.

Because the reference and predicted positions are drawn from the same set of 84 predefined class centroids, a correct class prediction contributes exactly zero localisation error. The reported errors therefore measure centroid-to-centroid displacement caused by misclassification. They do not include uncertainty about the tag's position within a grid cell and should not be interpreted as unrestricted continuous-position errors.

In the chronological K-fold experiments, each metric is first calculated separately for every fold. The reported central value is the unweighted mean of the fold-level metrics:

$$\bar{q} = \frac{1}{K} \sum_{k=1}^{K} q_k, \tag{7}$$

and fold-level dispersion is reported using the sample standard deviation:

$$s_q = \sqrt{\frac{1}{K-1} \sum_{k=1}^{K} (q_k - \bar{q})^2}. \tag{8}$$

Because fold sizes may differ slightly, $\bar{q}$ is an unweighted mean of the fold-level metrics rather than a pooled per-window result. The reported standard deviation describes variation across folds and is not a confidence interval.

Further details of the chronological splitting procedure, paired statistical test, and experiment-specific reporting are provided in Chapter 7.

Chapter 3

Related Work

This chapter organises related work around three design dimensions that frame the design and evaluation of the CS3D benchmark. positioning technology and receiver deployment, the modelling of RSSI fingerprints, and the use of propagation-based information and post-processing. These dimensions are reviewed in Sections 3.1, 3.2, and 3.3, respectively. Section 3.4 then synthesises the research gap addressed by CS3D.

3.1 Positioning technologies and deployment constraints

Indoor-positioning technologies differ in accuracy, cost, energy use, and infrastructure requirements. General surveys compare technologies including Wi-Fi, Bluetooth Low Energy (BLE), and ultra-wideband (UWB) [62], while BLE-focused surveys examine the advantages and limitations of low-power BLE deployments [55]. UWB can provide high localisation accuracy but commonly requires dedicated receiver hardware [1]. Bluetooth 5.1 direction finding based on angle of arrival similarly requires compatible multi-antenna receivers [13]. Other indoor-positioning research has examined Zigbee and LoRa alongside BLE and Wi-Fi [53], BLE fingerprinting methods [38], and LoRaWAN for longer-range indoor positioning [43].

Receiver deployment is also an important design choice. Some BLE fingerprinting studies use beacons or measurement equipment installed specifically for the experiment [19]. Other research optimises receiver placement to improve localisation performance, including gradient-based placement strategies for BLE anchors [60].

The CS3D deployment instead reuses a Crownstone receiver mesh originally installed for building automation. Most receiver positions were inherited from the building installation, while two receivers were added to improve coverage. The resulting geometry was therefore not globally optimised for localisation. A published description of the Crownstone platform is provided by Marathe [35]. Reusing existing infrastructure reduces the need for additional positioning hardware, but constrains receiver coverage and placement. These characteristics are important when interpreting the CS3D results. The deployment and ground-truth grid are described in Chapter 5.

3.2 Learning from RSSI fingerprints: classical models and deep neural networks

RSSI fingerprinting has been studied using deterministic matching, classical machine-learning models, and deep neural networks. Surveys of machine learning for indoor positioning describe methods ranging from nearest-neighbour matching to learned classifiers

[39, 51, 54]. Their relative suitability depends on the input representation, dataset size, and evaluation protocol.

When an RSSI observation window is represented using summary features, the resulting input is tabular. Grinsztajn et al. found that tree-based models remained strong on medium-sized tabular datasets, including datasets with approximately 10^4 samples [24]. They associated this result with properties such as uninformative features and irregular target functions. The tabular CS3D representation contains 7,006 windows, so this evidence motivates the inclusion of strong tree-ensemble baselines. It does not establish in advance that tree models will outperform deep neural networks on CS3D. Tree and boosted-tree models have also been applied directly to RSSI-fingerprinting tasks [57, 22, 63].

Deep neural networks can operate on tabular features, ordered RSSI sequences, or structured sets of receiver observations. The MLP evaluated in this thesis operates on the tabular representation. Temporal convolutional networks have been evaluated on BLE RSSI sequences [29], while one-dimensional convolutional networks have been used to formulate three-dimensional BLE localisation as classification [61]. Height-aware RSSI modelling has also been studied through empirical propagation models [2]. In related Wi-Fi fingerprinting research, set-based attention models represent detected access points and their RSSI values as unordered inputs [3].

These studies motivate the comparison in this thesis between classical tabular models, neural tabular models, deep sequence models, and an anchor-structured attention model. All models are evaluated within the common CS3D experimental framework. Because their native input representations use different window lengths, a separate matched-window analysis distinguishes architectural differences from input-duration effects, as described in Chapter 8.

3.3 Physics-based, hybrid, and post-processing methods

Propagation models provide an alternative to treating RSSI fingerprinting as entirely data-driven. The log-distance path-loss model estimates distance from signal attenuation, while weighted centroid localisation (WCL) combines receiver positions using signal-dependent weights [50, 9]. Indoor multipath and obstruction limit the reliability of these estimates. The tabular pipeline therefore uses inverted path-loss distances and a WCL estimate as physics-derived input features rather than as final position estimates.

Physics-informed learning incorporates physical assumptions into model training. Physics-informed neural networks were introduced to constrain learning using governing equations [49], and related approaches have been applied to wireless path-loss prediction [33]. The architecture labelled as a PINN in this thesis is more specifically a physics-guided MLP. Its loss contains an auxiliary term based on the WCL position estimate rather than a differential-equation constraint. This distinction is retained when interpreting its results. Hardware- and device-dependent RSSI offsets provide a further reason not to treat propagation-derived quantities as exact physical measurements [47].

Per-window classification does not explicitly enforce continuity between successive position estimates. Several state-estimation methods can be applied after classification:

Table 2: Post-processing methods relevant to sequential localisation.

Method	Role	Reference
Kalman filtering	Recursively combines observations under linear and approximately Gaussian state and noise assumptions.	[27]
Discrete Markov localisation	Propagates a probability distribution over a finite set of candidate locations.	[20]
Particle filtering	Represents potentially non-Gaussian position distributions using weighted samples.	[23, 4]

These filtering families have also been applied in BLE localisation, including Kalman-filter variants, particle-filter localisation, and fusion of BLE with UWB measurements [31, 48, 30]. When a filter interprets classifier outputs as probabilities, the calibration of those outputs becomes relevant. Platt scaling, temperature scaling, and multiclass calibration methods provide possible corrections [45, 40, 25, 32].

Post-processing is excluded from the main model comparison because it adds motion assumptions and configuration choices beyond the per-window localisation task. Six causal post-processing variants are therefore evaluated separately in Appendix D.

3.4 The research gap

Public fingerprinting datasets differ in signal technology, receiver deployment, ground-truth representation, and evaluation structure. UJIIndoorLoc [58] and the Mendoza-Silva BLE dataset [37] provide structural comparisons for signal technology, receiver deployment, and ground-truth representation. HDLC [41, 42] and the Daniş BLE trajectory dataset [17, 16] are used for the external evaluations reported in Chapter 9. Their structural differences from CS3D are summarised in Table 4.

The reviewed literature addresses the main components of this thesis individually:

- permanently installed BLE infrastructure has been documented for the Crownstone building-automation platform [35];
- other BLE studies use purpose-deployed equipment or optimise anchor placement for localisation [19, 60];
- three-dimensional or height-aware BLE localisation has been studied using convolutional and propagation-based models [61, 2];
- classical models and deep neural networks have been compared on tabular and fingerprinting datasets [24, 36]; and
- chronological evaluation has been recommended for temporally ordered observations to reduce dependence between training and test data [8, 11].

Within the literature reviewed for this thesis, no study was identified that combines a BLE receiver mesh inherited primarily from building automation, grid-cell ground truth at two

within-floor heights, multiple classical and deep neural network model families evaluated on one target dataset, and chronological within-class K-fold evaluation with paired model comparison. The contribution of CS3D is this combined dataset and experimental setting rather than any individual component.

Chapter 4

Initial Study: Room-Level Localisation

Phase 1 was an earlier room-level localisation study conducted in the same Almende office using 15 fixed Crownstone receivers. It classified the asset into one of four rooms: Kitchen, Lab, Meeting Room, or Office. In contrast, the present study uses 17 receivers and predicts one of 84 composite classes representing 42 grid cells at two tag heights. The Phase 1 results are included as context rather than as a direct benchmark because both the prediction target and the evaluation protocol differ.

For the tabular pipeline, Phase 1 calculated five RSSI statistics for each receiver over an 8 s window, producing 75 features. The sequence representation contained 30 time bins of 500 ms across the 15 receivers, corresponding to a 15 s input duration. The evaluated methods included classical machine-learning models and temporal neural networks, including an LSTM, a temporal convolutional network (TCN), and a CNN-LSTM. Synthetic Gaussian noise, packet dropout, and receiver-specific bias changes were also used to examine robustness.

Under the clean held-out evaluation, LightGBM achieved 72.68% room-level accuracy, whereas the TCN achieved 99.81%. During live evaluation, the TCN achieved 76.6%. These results suggested that high offline accuracy did not necessarily transfer to a later recording session. Because Phase 1 used a single held-out split, it did not provide a controlled estimate of cross-session degradation.

Phase 1 informed three main changes in the present study. First, the prediction target was refined from four rooms to three-dimensional grid-cell and height classes. Second, the single held-out evaluation was replaced by chronological within-class K-fold cross-validation with fold-level dispersion and a paired statistical test. Third, classical and deep neural network models were evaluated under a common protocol, including a matched-window analysis to control for differences in input duration. Table 3 summarises the main changes.

Table 3: **From room-level to 3D grid-cell localisation.** Main differences between Phase 1 and the present study. The results of the two studies are not directly comparable because their prediction targets and evaluation protocols differ.

Dimension	**Phase 1**	**Present study**
Prediction target	4 room classes	84 composite classes: 42 grid cells at 2 heights
Fixed receivers	15 Crownstone receivers	17 Crownstone receivers
Tabular features	75 statistical features	144 features: 85 statistical and 59 physics-derived
Tabular window	8 s	4 s
Sequence representation	30 time bins $\times$ 15 receivers	30 time bins $\times$ 17 receivers
Evaluation protocol	Single held-out split	Chronological within-class K-fold cross-validation with a paired sign-flip test

Chapter 5

The Crownstone 3D Dataset (CS3D)

This chapter presents the novel Crownstone 3D dataset (CS3D), collected as part of this thesis, and describes its collection protocol. CS3D contains RSSI measurements from a single BLE asset tag in an active office environment (Figure 5). Its advertisements were received by 17 fixed Crownstone receivers, 15 of which were inherited from the building-automation deployment and two of which were added before data collection (Figure 4). The target-deployment experiments reported later use this dataset.

The chapter is organised as follows:

1. the office deployment and ground-truth grid (Section 5.1);
2. the protocol used to collect the raw reception log (Section 5.2);
3. the transformation of the raw log into tabular and sequence learning examples, together with the reference split (Section 5.3);
4. the scope and limitations of the dataset (Section 5.4).

Table 4 compares CS3D with selected public fingerprinting datasets reviewed in Chapter 3. It summarises what each dataset provides and how its instrumentation and ground truth differ from the CS3D setting.

HDLC and Danişş are used in Chapter 9 to compare the best-performing applicable models identified on CS3D with published baselines on public data. Because the datasets use different collection and evaluation protocols, these results provide contextual comparisons rather than direct model rankings.

Table 5 shows an example of the CS3D raw log. Each row represents one receiver's observation of a BLE advertisement and contains its measured RSSI together with the ground-truth position label active at that time.

5.1 Deployment and ground-truth grid

The CS3D deployment covers four rooms in the Almende office in Rotterdam: the Office, Kitchen, L-shaped Lab, and Meeting Room. Figure 6 shows the ground-truth grid and the fixed receiver positions. Table 6 summarises the spatial layout, ground-truth labels, and hardware configuration.

The ground truth comprises 42 grid cells surveyed at two tag heights. Each cell-height pair forms one of the 84 composite classes used by the models. For example, `A1_z0` denotes grid cell A1 at the lower tag height. The two height labels provide a discrete within-floor vertical component; they do not represent continuous vertical localisation. This limitation is discussed in Chapter 11.

Table 4: **Comparison of CS3D with public reference datasets.** Selected fingerprinting datasets are compared by signal modality, infrastructure, ground truth, and their differences from fixed-receiver BLE asset localisation with a within-floor height component.

Dataset	Signal and infrastructure	Ground truth	Difference from CS3D
UJIIndoorLoc [58] (2014)	Wi-Fi measurements collected through a crowdsourced smartphone survey; 520 observed access points	Building, floor, and planar-coordinate labels across three multi-floor buildings	Uses Wi-Fi smartphone measurements; the vertical label is a floor index rather than a within-floor tag height
Mendoza-Silva BLE [37] (2019)	BLE beacons deployed for positioning, with measurements collected using a smartphone	Two-dimensional position and facing direction on one floor	Contains no height component, and its beacon infrastructure was purpose-deployed for positioning
HDLC [41, 42] (2022)	Wi-Fi and BLE fingerprints purpose-collected across three layouts	Reference points recorded on three separate floors	The vertical label is a floor index, and the infrastructure was purpose-collected rather than inherited from building automation
CS3D (this thesis)	BLE asset transmissions received by 17 fixed built-in Crownstone receivers: 15 inherited from the building-automation deployment and two added before data collection	42 grid cells at two within-floor tag heights, forming 84 composite classes; raw timestamped reception records	Limited to one office floor, one BLE tag, one main survey campaign, and two discrete height levels

Contribution within the reviewed datasets. No other dataset identified in the literature reviewed for this thesis combines a fixed BLE receiver mesh inherited primarily from building automation, reception at fixed receivers rather than a mobile survey device, two within-floor tag-height labels, and raw timestamped reception records suitable for chronological window construction.

Table 5: **Example of the CS3D raw log.** The first five rows of the exported raw log (Section 5.2). The value of `assetID` is 1810092 in all five rows and is omitted. Time is expressed as milliseconds since the first displayed record (Δt); the source log stores nanosecond epoch timestamps. The values of `room` and `grid_id` are reproduced as recorded.

Δt (ms)	`channel`	`crownID`	`rssi` (dBm)	`grid_id`	`room`	x (m)	y (m)	`z_level`
0	38	3	−73	D1	LAB	1.0	7.0	0
81	38	10	−82	D1	LAB	1.0	7.0	0
83	39	33	−68	D1	LAB	1.0	7.0	0
84	39	33	−68	D1	LAB	1.0	7.0	0
85	39	9	−80	D1	LAB	1.0	7.0	0

Table 6: **CS3D deployment and hardware summary.** Spatial layout, ground-truth configuration, receiver deployment, asset transmitter, and collection interface.

Aspect	**CS3D configuration**
Site	Almende office, Rotterdam; four rooms with a measured total area of 144.3 m^2
Room allocation	Office: 48.74 m^2, 21 cells; Kitchen: 44.51 m^2, 9 cells; Lab: 39.19 m^2, 10 cells; Meeting Room: 11.85 m^2, 2 cells
Grid	42 cells arranged as seven rows (A–G) and six columns (1–6); nominal cell size 2 m × 2 m
Boundary dimensions	Outermost column width: 1.29 m; outermost row height: 1.65 m; centroid span approximately 9.3 m × 11.7 m
Tag heights	0.05 m and 1.00 m
Composite classes	42 cells × 2 tag heights = 84 classes
Positioning receivers	17 fixed built-in Crownstone modules at surveyed positions: 15 inherited from the building-automation deployment and two added before data collection
Receiver mounting heights	0.10–2.97 m; four receivers mounted at ceiling height
Crownstone form factor	Built-in module used for all 17 positioning receivers; plug-in variant not used in CS3D
Asset transmitter	One KKM F2 Bluetooth Low Energy beacon
Collection interface	CS38 USB sniffer connected to the collection laptop; used to forward asset reports and excluded from the positioning inputs



Figure 4: Crownstone hardware form factors. The form factor used in CS3D is identified in Table 6.



Figure 5: The KKM F2 Bluetooth Low Energy beacon used as the asset transmitter in CS3D.

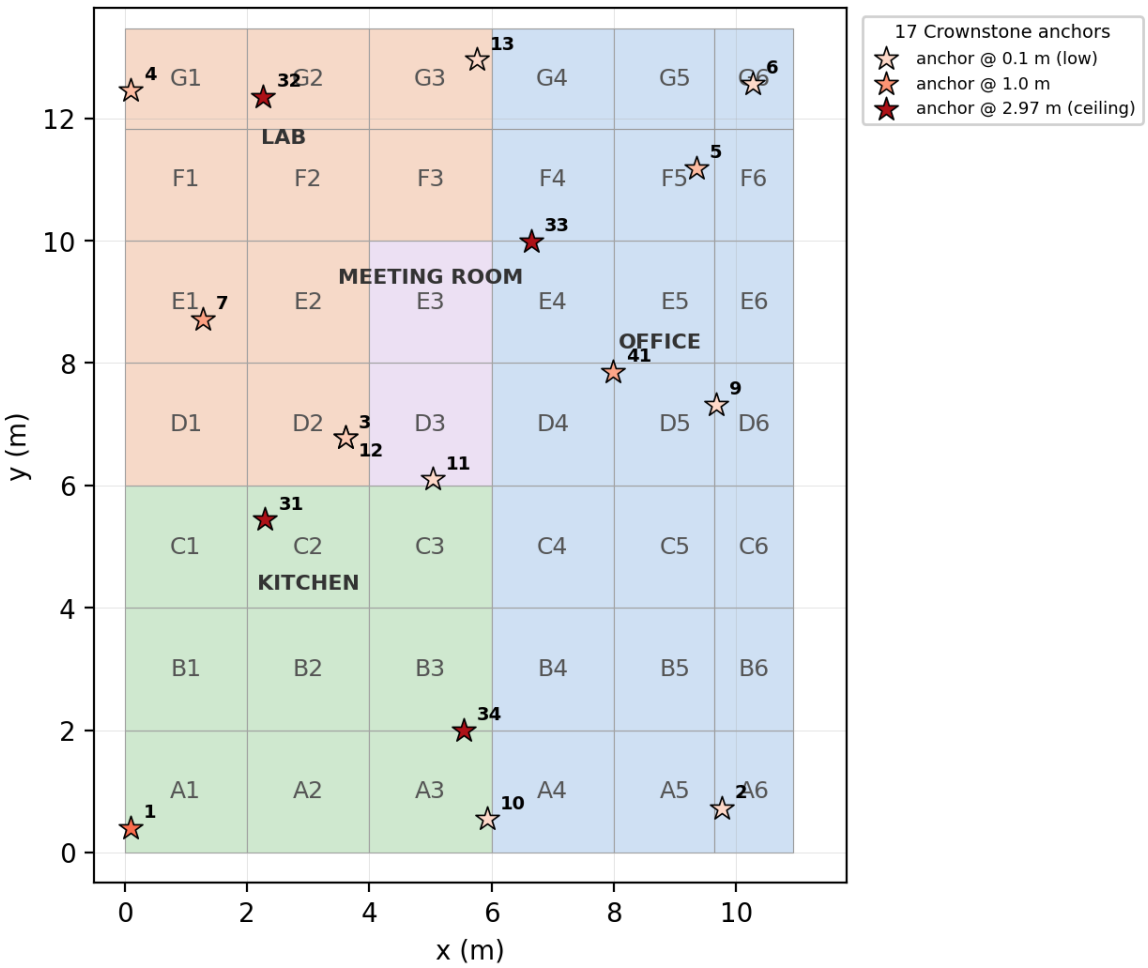


Figure 6: Top-down view of the office deployment, showing the 42 ground-truth grid cells by room and the 17 fixed Crownstone receivers. Receiver markers are shaded according to mounting height; four receivers are ceiling-mounted. Several receivers are located on perimeter walls outside the span of the grid-cell centroids.

5.2 Collection protocol

CS3D was collected by recording receiver reports while one BLE asset tag occupied each labelled cell-height position. The asset tag, ID 1810092, transmitted iBeacon-style advertisements [19] on the three BLE advertising channels. When a Crownstone receiver detected an advertisement, it generated a report containing the receiver identifier, advertising channel, and measured RSSI.

The reports were forwarded through the Crownstone mesh to the CS38 sniffer, transferred over UART to an MQTT broker on the collection laptop, and stored in a time-series database. The raw log was then exported as a comma-separated values file. Each row represents one receiver's observation of an advertisement rather than one unique advertisement transmitted by the tag.

Ground-truth labels were assigned during collection. Before each dwell, the operator selected the active grid cell, tag height, and room using a labelling tool. Subsequent receiver reports inherited this label until the position was marked complete. Reports recorded while the operator moved between positions had no active position label and were excluded from the labelled dataset.

Table 7 summarises the collection protocol and the resulting raw log.

Table 7: **CS3D collection protocol and raw-log summary.** Collection settings, campaign coverage, and the distribution of receiver reports in the labelled raw log.

Aspect	**Collection setting or result**
Asset	One BLE tag, ID 1810092
Broadcast format	iBeacon-style advertisements on the three BLE advertising channels
Ground-truth labelling	Active room, grid cell, and tag height assigned before each dwell; reports recorded during movement between positions were excluded
Dwell protocol	Tag placed or held at the centre of one grid cell at one height for approximately 5 min
Position coverage	84 labelled positions: 42 grid cells at two tag heights
Collection campaign	One main campaign conducted on 26–27 March 2026, with a total campaign duration of 8.45 h
Collection continuity	A progress record stored completed positions and allowed collection to resume after interruptions
Raw receiver reports	213,287 labelled rows; each row contains one receiver's observation of an advertisement
Observed RSSI	-101 to -38 dBm; mean -72.1 dBm and standard deviation 8.65 dB
Reports by room	Office: 108,063; Lab: 50,885; Kitchen: 43,224; Meeting Room: 11,115

5.3 From raw log to learning examples

The raw log contains individual receiver reports, whereas the evaluated models require fixed-size examples. Figure 7 summarises the transformation into separate tabular and sequence representations. These representations use different window lengths and construction rules and are therefore distinct model inputs.

For the tabular representation, the time-ordered reports within each composite cell-height class are divided into non-overlapping 4 s windows. The class-specific construction prevents a window from combining reports with different position labels. A window is retained only when reports from at least six of the 17 positioning receivers are present. This fixed quality rule excludes windows with very limited receiver coverage; the threshold was not selected through an optimisation experiment. This procedure produces 7,006 accepted tabular windows.

For a receiver contributing more than one report to a window, its RSSI series is processed using a Hampel outlier filter [44]. Missing-receiver encoding, statistical summaries, physics-derived features, and the rationale for the 4 s duration are defined in Chapter 6.

The sequence representation is constructed independently. Receiver reports are resampled into 500 ms bins, and a sliding window of 30 bins, corresponding to 15 s, is applied with a stride of five bins. A sequence window is retained only when one composite label accounts for at least 80% of its 30 bins. That dominant composite label is assigned to the retained window. Windows for which no composite label reaches the 80% threshold are excluded. This procedure produces 11,875 sequence windows of shape 30×17.

Table 8 summarises the two derived representations.

Table 8: **Learning representations derived from the CS3D raw log.** The tabular and sequence pipelines use different window construction and acceptance rules. Feature definitions and model-specific processing are provided in Chapter 6.

Representation	**Window construction**	**Acceptance rule**	**Output**
Tabular	Non-overlapping 4 s windows within each composite class	Reports from at least six of the 17 positioning receivers	7,006 windows
Sequence	30 bins of 500 ms, giving 15 s windows with a five-bin stride	One composite label accounts for at least 80% of the bins and is assigned to the window	11,875 windows of shape 30×17

An auxiliary chronological within-class 70/30 partition of the tabular windows contains 4,907 training windows and 2,099 test windows. For each composite class, the earliest 70% of its time-ordered windows are assigned to training and the final 30% to testing. This deterministic partition is retained as a reference split. It prevents temporally adjacent windows from being dispersed randomly across the training and test sets, although train–test adjacency can remain at the split boundary.

The principal experimental results do not rely on this single partition. Instead, the models

are evaluated using the chronological within-class K-fold protocol defined in Chapter 8. Sequence split construction and temporal-gap handling are specified there; model-specific preprocessing and augmentation are defined in Chapter 6.

5.4 Scope

CS3D supports controlled comparison of localisation models using fixed-receiver BLE measurements and discrete cell-height labels. Its scope is limited to one BLE tag, one office floor, one main collection campaign, and two within-floor tag heights. The tag remained at surveyed positions during each dwell, and building occupancy was not recorded.

Consequently, results obtained on CS3D do not by themselves establish generalisation across collection days, buildings, receiver deployments, tag hardware, moving trajectories, or occupancy conditions. The public dataset evaluations broaden the empirical context but do not remove these limitations of the primary CS3D evaluation. Their implications are discussed in Chapter 10.

Chapter 6

System Methodology

This chapter defines the system pipeline applied to the learning examples constructed in Chapter 5. Figure 7 summarises five stages: input data, representations, model families, output decoding, and evaluation. Each model produces scores for the 84 composite cell-height classes. The class with the highest score is selected using argmax and decoded to the predefined three-dimensional centroid of the corresponding cell-height class.

The model-family comparison uses the same class definitions, evaluation metrics, and model-selection rule. All models are evaluated within the same chronological within-class K-fold framework, with representation-specific preprocessing and temporal-gap handling where required. Models that consume the same representation receive the same processed inputs.

However, the tabular, sequence, and anchor-token representations differ in structure, and the tabular and sequence pipelines use different input durations. The native comparison between classical models and deep neural networks therefore cannot be interpreted as an architecture-only comparison. The matched-window analysis in Chapter 8 examines whether the observed performance difference persists when input duration is controlled.

Deployment cost is characterised using three quantities reported in the experiments: model file size, CPU inference latency, and required input duration. The architectural configuration of each model is reported separately in Table 10. Trainable parameter counts are not reported because they were not computed consistently across all model families.

Section 6.1 first presents the model families and their input representations. Section 6.2 defines the windowed statistical features. Section 6.3 defines the physics-derived features and per-receiver calibration. Section 6.4 specifies the training procedure.

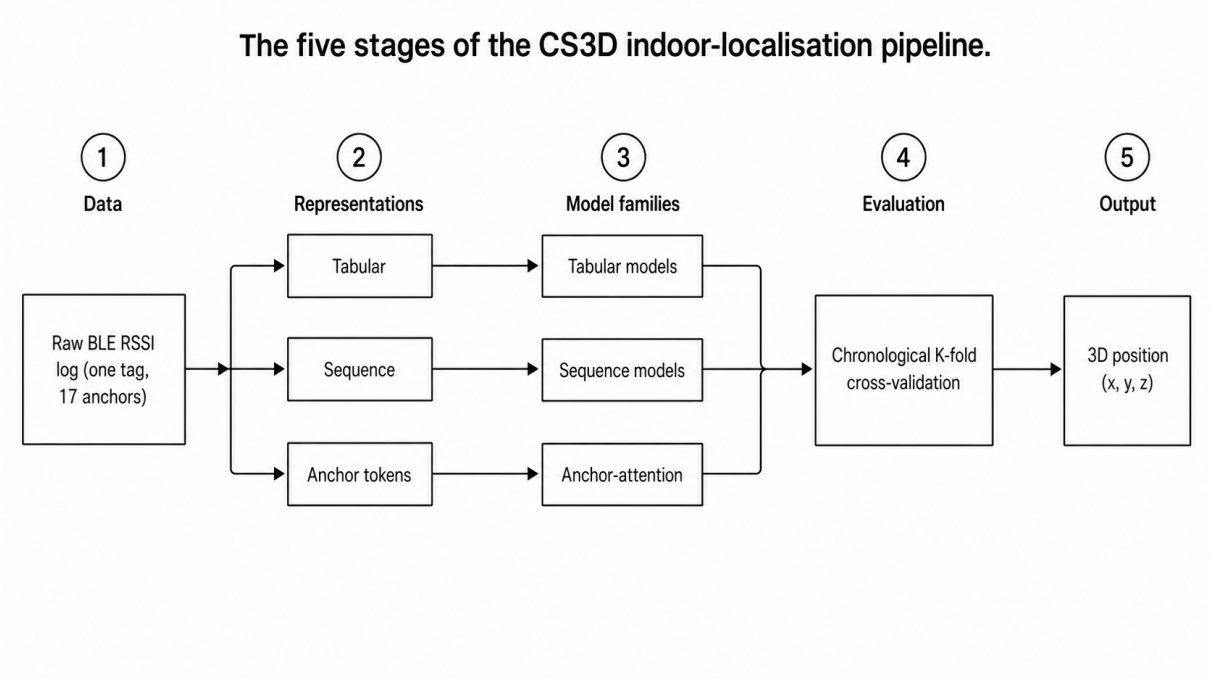


Figure 7: **Overview of the CS3D modelling pipeline.** The raw reception log is converted into tabular, sequence, and anchor-token representations and supplied to 17 model families. Each model produces class scores over the 84 composite classes. The class with the highest score is selected and decoded to its fixed 3D centroid. Table 9 summarises the principal settings for each stage.

6.1 Model families

The 17 model families are grouped by input representation in Table 10. Ten families consume the 144-dimensional tabular vector: k-NN, SVM, MLP, Random Forest, ExtraTrees, LightGBM, XGBoost, CatBoost, DeepMLP, and the implementation labelled PINN. Six families consume the 30×17 sequence representation: LSTM, 1D-CNN, TCN, CNN-LSTM, Transformer, and FusionNet. The remaining family applies attention over 17 anchor tokens rather than over time steps.

The tabular group contains both classical models and neural networks. The native comparison between the strongest classical model and the strongest deep neural network therefore also spans different representations and input durations. It cannot be interpreted as an architecture-only comparison. Chapter 8 reports a matched-window analysis that controls the input duration.

The model labelled PINN in the implementation is a physics-guided MLP, not a differential-equation-based physics-informed neural network. It uses the DeepMLP backbone and adds a weighted-centroid consistency term to the classification loss with $\lambda = 0.1$.

The anchor-attention model embeds each anchor's window-mean RSSI together with its surveyed coordinates. Two self-attention layers allow the 17 anchor tokens to exchange information before classification.

Table 9: **Per-stage details of the CS3D modelling pipeline.** Principal inputs, representations, evaluation settings, and output decoding corresponding to Figure 7.

Stage	**Produces**	**Principal settings**
1. Data	Raw reception log	213,287 receiver reports from 17 positioning receivers and one BLE tag; dataset construction is defined in Chapter 5
2. Representations	Tabular	7,006 non-overlapping 4 s windows; 144 features: 85 statistical and 59 physics-derived
	Sequence	11,875 windows; 30×17 RSSI matrices; 500 ms bins and 15 s input duration
	Anchor tokens	17 tokens containing window-mean RSSI and surveyed receiver coordinates
3. Model families	17 families	10 tabular families, 6 sequence families, and 1 anchor-attention family
4. Evaluation	Per-window class scores	Chronological within-class K-fold evaluation ($K = 5$ and $K = 10$); grid accuracy, 2D MAE, and 3D MAE; paired sign-flip test and random-split audit
5. Decoding	3D position	Argmax selection over 84 composite classes, followed by decoding the selected class to its fixed centroid (x, y, z)

Table 10: **The seventeen model families.** Fixed configurations used in the main comparison, grouped by input representation. Hyperparameter sensitivity is evaluated separately in Appendix A.2.

Model	**Input**	**Fixed configuration**
k-NN	Tabular, 144 features	$k = 5$, distance-weighted voting, Euclidean distance
SVM	Tabular, 144 features	RBF kernel, $C = 10$
MLP	Tabular, 144 features	Hidden layers 256–128, ReLU activation
Random Forest	Tabular, 144 features	300 trees, $\sqrt{d}$ candidate features per split
ExtraTrees	Tabular, 144 features	300 trees, $\sqrt{d}$ candidate features per split
LightGBM	Tabular, 144 features	500 boosting rounds, learning rate 0.05, 31 leaves
XGBoost	Tabular, 144 features	500 boosting rounds, depth 6, learning rate 0.05
CatBoost	Tabular, 144 features	500 iterations, depth 6, learning rate 0.05
DeepMLP	Tabular, 144 features	Hidden layers 256–128–64, batch normalisation, dropout
PINN	Tabular, 144 features	Physics-guided DeepMLP with weighted-centroid auxiliary loss, $\lambda = 0.1$
LSTM	Sequence, 30×17	Two layers, hidden size 128
1D-CNN	Sequence, 30×17	Three convolutional blocks, 32–64–128 channels
TCN	Sequence, 30×17	Four dilated residual blocks, 64 channels
CNN-LSTM	Sequence, 30×17	Convolutional feature extraction followed by a bidirectional LSTM with hidden size 128
Transformer	Sequence, 30×17	Two encoder layers, four attention heads, model width 64
FusionNet	Sequence, 30×17	Mean-RSSI branch combined with a convolutional branch
Anchor-attention	17 anchor tokens	Two self-attention layers, four heads, model width 64

The model breadth serves three purposes. Tree ensembles provide strong baselines for the medium-sized tabular representation. Sequence models test whether temporal RSSI structure provides information beyond fixed window summaries. The physics-guided and anchor-attention variants test whether radio-propagation and receiver-geometry structure improve classification. The results are reported in Chapter 8.

6.2 Windowed statistical features

Each accepted 4 s tabular window is converted into a fixed-length feature vector. This section defines the 85 statistical features. Section 6.3 adds 59 physics-derived features, giving the 144-dimensional representation supplied to the tabular models.

The 4 s duration was fixed before the main model comparison. It shortens the 8 s tabular window used in Phase 1 while retaining multiple receiver reports for window-level feature calculation. It also limits the waiting time before a tabular prediction can be produced. Because the CS3D tag remained stationary during each dwell and windows were constructed within a composite label, accepted windows do not span two surveyed positions. The 4 s duration was a fixed design choice rather than the result of a window-length optimisation. This design does not establish performance for freely moving tags.

For a receiver contributing more than one report to a window, its RSSI series is processed using the Hampel filter described in Section 5.3 [44]. The filter was selected to reduce the effect of isolated RSSI spikes while preserving the normal short-term signal variation within the window. The filtered measurements are then represented by the five statistical features in Table 11. Across 17 receivers, this produces $5 \times 17 = 85$ statistical features.

The five statistical features were selected because they describe different properties of the RSSI measurements within each 4 s window. The mean represents the overall signal level, while the median provides a central value that is less affected by any remaining extreme readings. The standard deviation measures short-term signal variation, and the maximum retains the strongest received signal in the window. The heard flag records whether the receiver detected the tag at all. Together, these features represent signal level, variation, strongest reception, and receiver availability in the CS3D data.

When a receiver contributes no report to a window, the mean, median, and maximum are set to the sentinel value -100 dBm, while the standard deviation and heard flag are set to zero. The heard flag explicitly records whether a receiver contributed a report. The level sentinel does not by itself distinguish a missing receiver from a genuinely observed RSSI close to -100 dBm; this is a limitation of the encoding.

6.3 Physics-derived features and per-receiver calibration

The tabular pipeline adds 59 features derived from radio-propagation models and receiver geometry. These features were selected to add approximate signal-attenuation and spatial information to the statistical RSSI features. The distance and free-space path-loss features describe signal attenuation, the weighted centroid provides a coarse position estimate, and

Table 11: **The five statistical features computed per receiver.** Each of the 17 receivers contributes these summaries, producing 85 statistical features per tabular window.

Feature	Meaning
Mean	Arithmetic mean RSSI within the window
Median	Median RSSI, reducing sensitivity to remaining extreme values
Standard deviation	Variation of the receiver's RSSI measurements within the window
Maximum	Strongest observed RSSI within the window
Heard flag	1 when at least one report from the receiver is present; otherwise 0

the nearest-receiver features identify the receivers estimated to be closest to the tag. The calibration flags record which calibration method was used for each receiver. These values are used as model inputs rather than direct position estimates because indoor RSSI is only a noisy proxy for distance.

Table 12 summarises the five feature groups. All features are calculated from the per-receiver window summaries rather than directly from individual reception records.

Table 12: **The 59 physics-derived features.** Together with the 85 statistical features in Table 11, these form the 144-dimensional tabular representation.

Feature group	Count	Derivation
Receiver-distance estimates	17	Inversion of the per-receiver path-loss model using window-mean RSSI
Free-space path loss	17	Equation (11) evaluated at each estimated distance
Weighted-centroid coordinates	2	Planar weighted-centroid estimate from Equation (12)
Three nearest receivers	6	Scaled identifiers and estimated distances of the three nearest receivers
Calibration flags	17	Indicator of the calibration estimator used for each receiver
Total	59	

The distance proxies use the log-distance path-loss model [50]:

$$\mathrm{RSSI}_i(d) = P_0 - 10\, n_i \log_{10}\left(\frac{d}{d_0}\right), \tag{9}$$

where d is the distance in metres, $d_0 = 1\,\mathrm{m}$, P_0 is the reference RSSI level at d_0 used by the implementation, and n_i is the path-loss exponent fitted for receiver i. Inverting Equation (9) gives

$$d_i = d_0\, 10^{\frac{P_0 - \mathrm{RSSI}_i}{10 n_i}}. \tag{10}$$

The resulting value is an engineered distance proxy, not an exact measurement of physical distance.

The exponent is calibrated separately for each of the 17 receivers by pairing its mean RSSI at a surveyed position with the corresponding three-dimensional receiver-to-tag distance. The implementation uses a close-distance estimate where suitable observations are available and otherwise applies an ordinary least-squares fit of RSSI against log-distance. Five receivers use the close-distance estimator and twelve use the least-squares fallback. A binary calibration flag records which estimator was used; receivers using the fallback are not discarded.

The path-loss parameters are fitted once using all surveyed positions before chronological cross-validation and reused across folds. The calibration can therefore use positions that later contribute test windows. This pre-split calibration is treated as an evaluation limitation in Chapter 10. The fitted per-receiver parameters are reported in Appendix A.1 (Table 24).

The fitted path-loss exponents range from 1.0 to 5.4. Two receivers reach the lower fitting bound of $n = 1.0$.

Free-space path loss provides a reference attenuation value:

$$\mathrm{FSPL}(d) = 20\log_{10}(d) + 20\log_{10}(f) + 20\log_{10}\left(\frac{4\pi}{c}\right), \tag{11}$$

where d is measured in metres, f is the carrier frequency in hertz, and c is the speed of light [50]. The implementation uses 2.44 GHz as a representative frequency near the centre of the BLE advertising band.

The planar weighted-centroid estimate combines the distance proxies as

$$\hat{\mathbf{p}} = \frac{\sum_i w_i \mathbf{p}_i}{\sum_i w_i}, \qquad w_i = \frac{1}{\max(d_i, 0.1)^2}, \tag{12}$$

where $\mathbf{p}_i \in R^2$ is the planar coordinate of receiver i, d_i is its estimated distance, and the 0.1 m lower bound prevents division by zero [9]. The weighted-centroid operation has no learned parameters, but in this pipeline it inherits error from the calibrated RSSI-to-distance proxies. It is therefore used only as an engineered input and as the auxiliary consistency target of the physics-guided MLP.

All estimated distances are clamped to the interval 0.1–20 m. For a receiver absent from a window, the distance feature takes the upper bound. The free-space path-loss features use the same representative 2.44 GHz frequency. The distance, free-space path-loss, weighted-centroid, nearest-receiver, and calibration blocks together produce the 59 features listed in Table 12.

6.4 Training protocol

The main comparison uses fixed, documented configurations rather than performing an unrestricted hyperparameter search separately for every family. A separate sensitivity analysis in Appendix A.2 assesses whether additional tuning materially changes the selected baseline results.

Among the eight baseline tabular pipelines, six use standardised features. Random Forest and ExtraTrees use unscaled inputs. Whenever standardisation is applied, its parameters are estimated using the training portion of the current fold only. Randomised components use seed 42 for reproducibility.

The neural models use Adam with a learning rate of 10^{-3} and weight decay of 10^{-5}. Their principal objective is cross-entropy loss; the physics-guided MLP additionally uses the weighted-centroid consistency term described above. A learning-rate scheduler halves the learning rate when validation accuracy stops improving. Early stopping is performed on a validation subset drawn from the current training fold, and the best-scoring model weights are restored. Family-specific epoch, batch-size, and patience settings are reported in Appendix A.2 (Table 26).

Sequence inputs are standardised per receiver using parameters estimated from the training sequences of the current chronological fold. Four masked copies are generated for each training sequence, with a random 20% of entries in each copy replaced by the sequence sentinel value. This augmentation is applied only to the training data. The copy count and masking proportion are fixed preprocessing choices rather than tuned hyperparameters.

The validation subset for the sequence models is drawn from the augmented training pool. Consequently, the selected stopping epoch is partly determined using augmented sequences. This design choice is considered in Chapter 10.

The main comparison ends with one class prediction per accepted window and its decoded grid-cell centroid. Temporal post-processing over consecutive predictions is evaluated separately in Appendix D. It is excluded from the principal model-family comparison so that all main results refer to the same per-window prediction stage.

Chapter 7 defines the chronological K-fold protocol, evaluation metrics, random-split audit, and paired statistical test.

Chapter 7

Experimental Setup

This chapter specifies the evaluation protocol used to assess predictive accuracy and computational cost, and fixes it before any result is reported. The chapter defines, in order, the cross-validation protocol under which every model is trained and tested (Section 7.1), the metrics that quantify accuracy and localisation error (Section 7.2), and the five experiments conducted on CS3D (Section 7.3). Applying one evaluation procedure to every model is what makes the comparisons in Chapter 8 controlled, and the resulting findings are reported there.

7.1 Evaluation protocol

Because every reported comparison depends on it, this section specifies the split protocol in full. All CS3D results reported in Chapter 8 use the chronological within-class K-fold scheme introduced in Chapter 2. The 7,006 tabular windows are split within each composite class into K chunks that are contiguous in time. Fold f uses chunk f from every class as its test set and uses the remaining chunks for training. Each class therefore appears in every test fold. The test folds form an exact partition of the data; their sizes sum to 7,006. The construction is deterministic and uses no shuffling. Temporal adjacency between training and test windows occurs only at chunk boundaries. The sequence models are evaluated under the same chronological construction over the 11,875 sequence windows. Because consecutive 30-bin windows advance by only five bins, the five training windows immediately preceding each test chunk are removed, so that no earlier training window shares time bins with the test chunk. Windows following a test chunk are not trimmed, so at most the final five windows of each class's test chunk may share bins with a subsequent training window. Both $K = 5$ and $K = 10$ are reported for every CS3D metric. This dual-K requirement applies only to CS3D; the public-dataset evaluations in Chapter 9 use the protocols stated there. Algorithm 1 formalises the construction; the random-split audit in Section 8.5 uses the same procedure, except that the per-class time chunks are replaced by a stratified random partition.

Algorithm 1 Chronological within-class K-fold split. The random-split protocol audit uses the same procedure with one branch modified, as marked.

Input: windows $i = 1 \ldots N$ with composite labels y_i and timestamps t_i; folds K; mode $\in \{\text{CHRONO}, \text{RANDOM}\}$

Output: test sets $T_1, \ldots, T_K$ that partition $\{1 \ldots N\}$, each containing every class

if mode = CHRONO **then**
  for each composite class c **do**
    $I_c \leftarrow$ indices with $y_i = c$, ordered by ascending t_i
    require $|I_c| \geq K$
    split I_c into K contiguous chunks; assign any remainder to the earliest chunks (± 1)
    $g_i \leftarrow k$ for every i in chunk k ▷ per-class time-chunk id
  end for
else
  $g \leftarrow \text{StratifiedKFold}(y, K, \text{shuffle, seed=42})$ ▷ audit: time order discarded; all other settings unchanged
end if
for $f \leftarrow 1$ to K **do**
  $T_f \leftarrow \{i : g_i = f\}$; $\text{train}_f \leftarrow \{1 \ldots N\} \setminus T_f$
  sequence models only: remove the five training windows immediately preceding each test chunk ▷ overlap gap
end for
assert every class appears in every T_f, and $\bigcup_f T_f = \{1 \ldots N\}$
report each metric as mean ± standard deviation over folds; the audit reports $\text{metric}_{\text{RANDOM}} - \text{metric}_{\text{CHRONO}}$ as the inflation

7.2 Metrics

Accuracy alone does not capture how far a misclassification falls from the true position, so three metrics are reported: grid accuracy and the 2D and 3D localisation errors defined in Section 2.5. They are computed as follows. Each predicted class is decoded to its cell centroid, and the localisation error is the distance from that centroid to the window's ground-truth position, so these errors include the centroid quantisation introduced by predicting discrete classes. Height enters the 3D error through the two labelled height levels. Each metric is reported as the mean over folds ± the sample standard deviation across folds. This dispersion describes fold-to-fold variation only and should not be interpreted as a confidence interval. Accuracies are reported as fractions in tables and as percentages in the text. Every reported result uses the same decoder, which maps a window's predicted class probabilities to a single position by selecting the most probable composite class and returning its cell centroid. Let p_c be the predicted probability of composite class c, for $c \in \{1, \ldots, C\}$ with $C = 84$, and let $\mathbf{m}_c$ be the three-dimensional centroid of class c. The predicted class $\hat{c}$ and the decoded position $\hat{\mathbf{x}}$ are

$$\hat{c} = \underset{c \in \{1,\ldots,C\}}{\arg\max}\, p_c, \qquad \hat{\mathbf{x}} = \mathbf{m}_{\hat{c}}, \tag{13}$$

so that each window is assigned the fixed centroid of its predicted cell.

7.3 Experiments on CS3D

Five experiments are conducted on CS3D, all under the protocol of Section 7.1 and the metrics of Section 7.2. Their names are used as the titles of the corresponding result sections in Chapter 8.

1. **Experiment 1: Model families.** All seventeen model families are trained and evaluated under the shared protocol, at both $K = 5$ and $K = 10$, to establish the accuracy ranking and to identify the strongest classical and the strongest DNN model (Section 8.1).

2. **Experiment 2: Best-performing classical versus DNN.** The strongest model from each group is compared with a paired sign-flip test [18] on per-fold 2D error, using 10,000 permutations and a fixed random seed, together with their computational costs. Because the two families summarise the data into different window populations, the per-fold errors are paired by fold index only; the resulting p-value is therefore indicative rather than exact, a point quantified in Section 8.2.

3. **Experiment 3: Window-size ablation.** The two strongest models are re-evaluated at a common window length, to separate the effect of the input-window duration from the effect of the architecture (Section 8.3).

4. **Experiment 4: Rooms and heights (3D).** The error of the strongest classical model is disaggregated by room and by height level, to establish where the remaining error concentrates and how well the vertical dimension is resolved (Section 8.4).

5. **Experiment 5: Random versus chronological split.** Every classical and tabular neural model is re-evaluated under a stratified random split with all other settings identical, quantifying the inflation the split choice introduces (Section 8.5).

Chapter 8

Results

This chapter reports the results of the five CS3D experiments defined in Section 7.3 and conducted under the evaluation protocol specified in Chapter 7. The experiments examine model performance, computational cost, input-window duration, spatial error patterns, and sensitivity to the train–test split. Together, they address the three research questions stated in Chapter 1. Results on the public datasets are reported separately in Chapter 9.

8.1 Experiment 1: Model families

Table 13 reports the $K = 5$ results for the 16 models evaluated under this setting. Figure 8 summarises their grid accuracy and mean 2D localisation error.

The Transformer obtained the lowest mean 2D localisation error, 0.356 m, with a grid accuracy of 88.3%. ExtraTrees obtained the lowest mean 2D localisation error among the classical models, 0.434 m, with a grid accuracy of 84.3%.

The native Transformer and ExtraTrees configurations differ in both input representation and input-window duration. The Transformer uses a 15 s RSSI sequence, whereas ExtraTrees uses a 4 s engineered tabular representation. The matched-window experiment in Section 8.3 compares the models at equal input-window durations.

The five tree-ensemble models obtained similar grid accuracies, ranging from 83.5% to 84.3%. Among the remaining classical baselines, the MLP, SVM, and k-NN obtained grid accuracies of 69.5%, 66.3%, and 38.5%, respectively.

Two architectures did not converge to competitive solutions under the shared training procedure. FusionNet achieved 5.9% grid accuracy at both values of K, while the anchor-attention model achieved 5.4% at $K = 10$. For reference, uniform chance accuracy over 84 classes is approximately 1.2%.

These values are reported as optimisation failures under the tested configuration, not as evidence that the architectures are intrinsically unsuitable for localisation. The shared procedure was retained to keep the principal comparison consistent across model families. A separate model-specific sensitivity analysis is reported in Appendix A.3. The principal result tables retain the shared-procedure values to preserve comparability across model families.

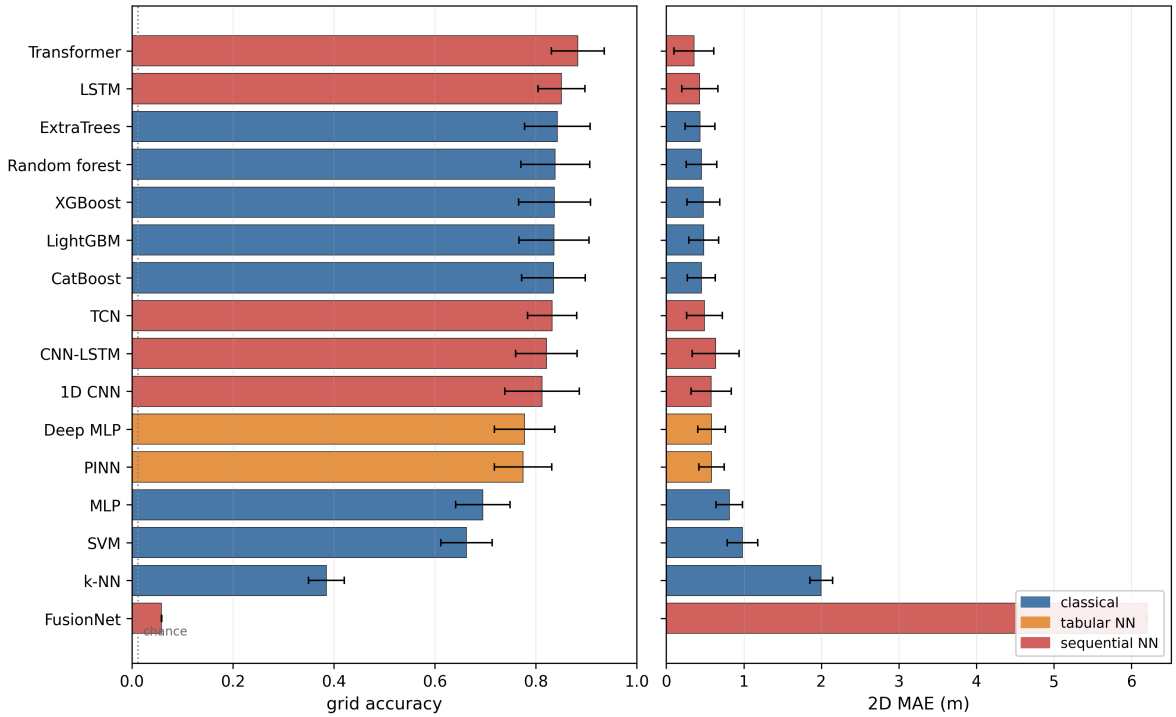


Figure 8: Grid accuracy and 2D mean absolute error for the 16 model families evaluated at $K = 5$, sorted by grid accuracy. Error bars show the sample standard deviation over the five folds. Colour identifies the model group: classical, tabular neural, and sequence neural. The dotted line marks uniform chance accuracy, 1/84. FusionNet is shown at its measured value rather than omitted.

Table 13: **Main results at $K = 5$.** Grid accuracy and centroid-decoded 2D and 3D mean absolute error for the 16 model families evaluated over five chronological folds on CS3D. Values are reported as mean ± sample standard deviation. Boldface marks the best value within each model group and does not imply statistical significance. The anchor-attention model was evaluated only at $K = 10$ and is reported in Table 14. A dagger (†) marks a model that did not converge to a competitive solution under the shared training procedure.

Model	Grid accuracy	2D MAE (m)	3D MAE (m)
ExtraTrees	**0.843 ± 0.065**	**0.434 ± 0.191**	**0.455 ± 0.195**
Random Forest	0.838 ± 0.068	0.452 ± 0.198	0.473 ± 0.202
XGBoost	0.837 ± 0.071	0.477 ± 0.211	0.499 ± 0.217
LightGBM	0.836 ± 0.070	0.483 ± 0.191	0.503 ± 0.196
CatBoost	0.835 ± 0.063	0.452 ± 0.178	0.474 ± 0.184
MLP	0.695 ± 0.054	0.813 ± 0.172	0.854 ± 0.176
SVM	0.663 ± 0.051	0.982 ± 0.197	1.024 ± 0.201
k-NN	0.385 ± 0.035	1.999 ± 0.146	2.071 ± 0.145
DeepMLP	**0.778 ± 0.060**	**0.584 ± 0.178**	**0.616 ± 0.184**
PINN	0.775 ± 0.057	0.584 ± 0.165	0.617 ± 0.171
Transformer	**0.883 ± 0.052**	**0.356 ± 0.256**	**0.371 ± 0.256**
LSTM	0.851 ± 0.046	0.432 ± 0.234	0.453 ± 0.235
TCN	0.832 ± 0.049	0.493 ± 0.233	0.518 ± 0.232
CNN-LSTM	0.821 ± 0.061	0.637 ± 0.304	0.659 ± 0.304
1D-CNN	0.812 ± 0.074	0.579 ± 0.261	0.606 ± 0.269
FusionNet†	0.059 ± 0.000	6.199 ± 0.004	6.249 ± 0.004

Table 14 reports the $K = 10$ evaluation. All eight classical models were re-evaluated, but the table presents only the three leading classical models to keep the comparison concise. Their ordering was ExtraTrees, LightGBM, and XGBoost.

Table 14: **Main results at $K = 10$.** Grid accuracy and centroid-decoded mean 2D and 3D localisation error over ten chronological folds on CS3D. All eight classical models were re-evaluated; the table presents the three leading classical models. The Anchor-attention model was evaluated only at $K = 10$. Values are reported as mean ± sample standard deviation across folds. Boldface marks the best result within each model group and does not imply statistical significance. A dagger (†) marks a model that did not converge to a competitive solution under the shared training procedure.

Model	Grid accuracy	Mean 2D error (m)	Mean 3D error (m)
ExtraTrees	**0.862 ± 0.085**	**0.387 ± 0.254**	**0.405 ± 0.260**
LightGBM	0.859 ± 0.091	0.424 ± 0.273	0.439 ± 0.279
XGBoost	0.859 ± 0.094	0.426 ± 0.297	0.443 ± 0.303
DeepMLP	**0.801 ± 0.080**	**0.523 ± 0.256**	**0.548 ± 0.261**
PINN	0.796 ± 0.075	0.537 ± 0.237	0.563 ± 0.241
Transformer	**0.905 ± 0.070**	**0.312 ± 0.307**	**0.321 ± 0.310**
LSTM	0.877 ± 0.065	0.383 ± 0.288	0.396 ± 0.290
CNN-LSTM	0.855 ± 0.061	0.520 ± 0.295	0.536 ± 0.297
TCN	0.854 ± 0.076	0.434 ± 0.305	0.456 ± 0.311
1D-CNN	0.841 ± 0.066	0.488 ± 0.303	0.512 ± 0.305
FusionNet†	0.059 ± 0.001	6.199 ± 0.006	6.249 ± 0.006
Anchor-attention†	0.054 ± 0.015	6.154 ± 0.139	6.205 ± 0.136

At $K = 10$, the Transformer achieved the highest grid accuracy, 90.5%, and the lowest mean 2D localisation error, 0.312 m. ExtraTrees was the strongest classical model, achieving 86.2% grid accuracy and a mean 2D localisation error of 0.387 m. Compared with the $K = 5$ evaluation, both models obtained lower mean errors, while their fold-level standard deviations were larger. These descriptive differences are reported without attributing them to a specific mechanism.

8.2 Experiment 2: Best-performing classical versus deep neural network

At both values of K, the Transformer obtained a lower mean 2D error than ExtraTrees. The formal paired comparison uses the ten chronological folds from the $K = 10$ evaluation. Per-fold 2D errors are paired by fold index, and a paired sign-flip test evaluates the null hypothesis that the mean paired difference is zero.

The observed mean difference was -0.075 m, favouring the Transformer. The Transformer had lower error in eight of the ten folds. The sign-flip test produced $p = 0.187$, and the bootstrap 95% confidence interval for the mean paired difference was $[-0.176, +0.016]$ m. The null hypothesis is therefore not rejected at the 0.05 significance level. This result does not establish that the two models are equivalent. Figure 9 shows the fold-level errors.

Three qualifications limit the interpretation of this test. First, only ten fold pairs are available, which limits the resolution of the statistical comparison. The confidence interval

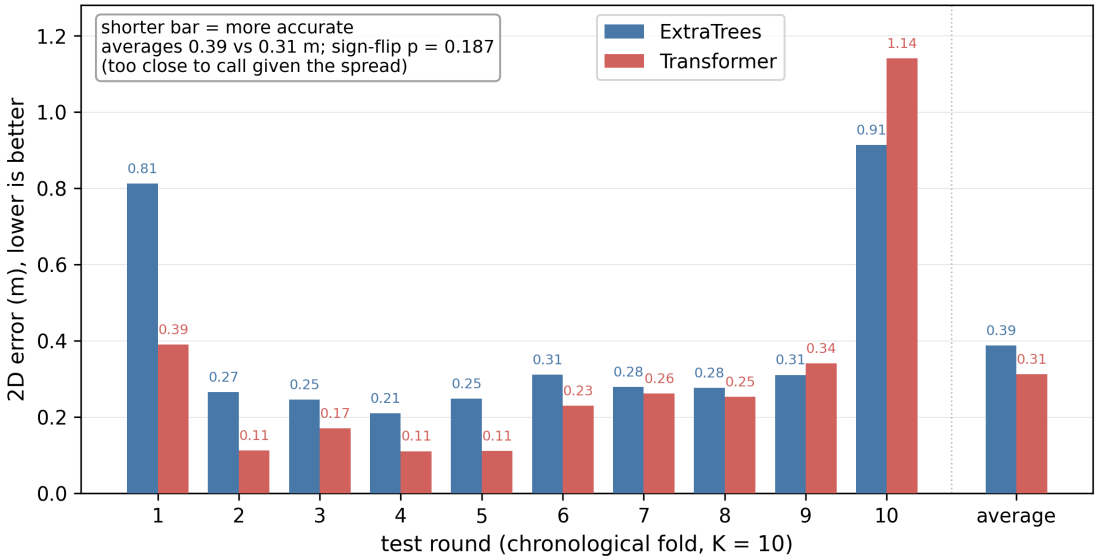


Figure 9: 2D mean absolute error for ExtraTrees and the Transformer in each of the ten chronological folds at $K = 10$. Lower values indicate lower error. The Transformer obtained lower error in eight of the ten folds. The paired sign-flip test did not reject the null hypothesis of zero mean paired difference ($p = 0.187$).

spans zero and remains compatible with both a Transformer advantage and a small difference in the opposite direction.

Second, the models are evaluated on different window populations. Each fold contains up to 1,216 sequence windows for the Transformer and up to 737 tabular windows for ExtraTrees, drawn from corresponding chronological time chunks. The pairing is therefore by fold index rather than by identical examples. Experiment 3 (Section 8.3) addresses this input-window difference.

Third, the pair was selected because it contains the two best-performing observed models. The reported p-value is not adjusted for this data-dependent model selection. The appropriate conclusion is that the paired sign-flip test did not provide sufficient evidence to reject the null hypothesis under this protocol; it does not establish equivalence between the models.

Computational cost is characterised using the three operational quantities defined in Chapter 6: saved-model size, CPU inference latency, and required input-window duration. Table 15 reports saved-model size and single-window latency for all 17 families.

Latency is measured as the median of 100 CPU predictions after ten warm-up calls on one laptop-class machine with an Intel Core i5-1340P processor and 16 GB RAM. The measurements use the saved 84-class models and their implemented scikit-learn or PyTorch serving paths. They are point estimates from one machine; tail latency and variability across hardware are not characterised.

Under this implementation, the Transformer produced a prediction in 1.5 ms from a 0.45 MB checkpoint. The serialised ExtraTrees model occupied 999.5 MB and required 38 ms per prediction through the scikit-learn serving path. The ExtraTrees value reflects

both its 300-tree configuration and its serialisation format; it should not be interpreted as an intrinsic lower bound for decision-forest deployment.

The bagged forests had the largest saved-model sizes and were among the slowest models. The boosted ensembles occupied tens of megabytes, while the neural checkpoints were below 3 MB. At $K = 5$, CatBoost and LightGBM remained within one percentage point of ExtraTrees in grid accuracy while requiring substantially smaller saved artefacts.

Input duration gives a different ordering. ExtraTrees requires one 4 s tabular window, whereas the Transformer requires a 15 s sequence. ExtraTrees therefore requires a shorter observation history before its first prediction can be formed. The deployment implications of this trade-off and the effect of the serving format are discussed in Chapter 10.

Table 15: **Model size and inference latency.** Saved model-artefact size in decimal megabytes and single-window CPU inference latency for all 17 model families. Measurements were made on an Intel Core i5-1340P processor with 16 GB RAM. Latency is the median of 100 timed predictions after 10 warm-up calls, using the saved 84-class models and each library's implemented serving path and stored thread settings.

Model	Size (MB)	Median latency (ms)
ExtraTrees	999.5	38.3
Random forest	442.7	37.6
XGBoost	39.4	22.1
LightGBM	35.4	6.0
CatBoost	21.9	0.4
SVM	12.2	1.3
k-NN	5.7	32.3
MLP	2.0	0.3
DeepMLP	0.35	0.7
PINN	0.35	0.3
Transformer	0.45	1.5
LSTM	0.89	1.1
TCN	0.40	2.3
CNN-LSTM	2.6	8.8
1D CNN	0.21	0.6
FusionNet	0.12	0.9
Anchor-attention	0.44	1.1

8.3 Experiment 3: Window-size ablation

The native comparison in Experiment 2 uses different input durations: 4 s for ExtraTrees and 15 s for the Transformer. It therefore combines differences in model family, input representation, and window duration. To examine the effect of input duration, both models were re-evaluated using 4 s and 15 s windows under the same chronological K-fold protocol. The ExtraTrees 15 s condition and the Transformer 4 s condition are diagnostic re-runs; the native configurations remain ExtraTrees at 4 s and the Transformer at 15 s.

Table 16 and Figure 10 report the $K = 10$ results. Increasing the input duration reduced mean 2D error for both models. ExtraTrees improved from 0.387 m at 4 s to 0.179 m at 15 s, a reduction of 0.209 m based on the unrounded fold values. The Transformer improved from 0.677 m to 0.312 m, a reduction of 0.365 m. For both within-model comparisons, the paired sign-flip test produced $p < 0.005$.

Table 16: **Window-size ablation.** Mean 2D absolute error over ten chronological folds at $K = 10$, reported as mean ± sample standard deviation. ExtraTrees at 4 s and the Transformer at 15 s are the native configurations used in the main comparison. The remaining cells are diagnostic re-runs used to match input duration. Boldface marks the lower mean error within each window duration and does not by itself imply statistical significance.

Window duration	ExtraTrees 2D MAE (m)	Transformer 2D MAE (m)
4 s	**0.387 ± 0.254**	0.677 ± 0.303
15 s	**0.179 ± 0.276**	0.312 ± 0.307

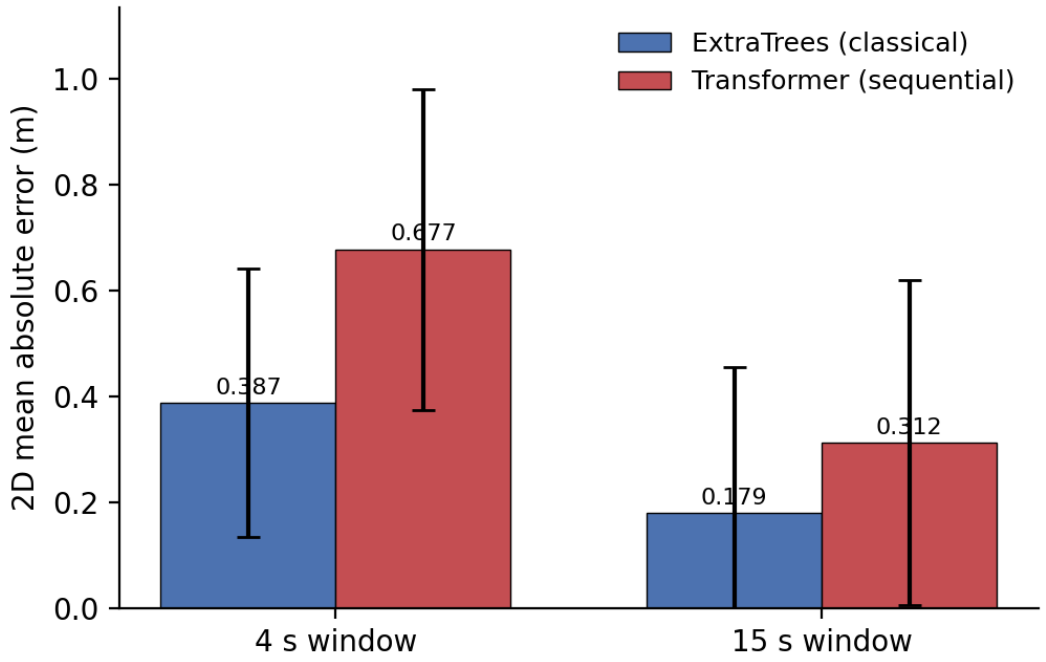


Figure 10: Mean 2D absolute error for ExtraTrees and the Transformer using matched 4 s and 15 s input durations at $K = 10$. Error bars show the sample standard deviation across folds and are not confidence intervals. Both models obtained lower mean error with the 15 s input, and ExtraTrees obtained the lower mean error at both matched durations.

At the matched 15 s duration, ExtraTrees obtained a mean 2D error of 0.179 m, compared with 0.312 m for the Transformer. The primary paired sign-flip test at $K = 10$ produced $p \approx 0.03$. However, the Wilcoxon signed-rank test produced $p = 0.06$, and the corresponding $K = 5$ comparison produced $p = 0.19$. The evidence for a model-family difference at 15 s is therefore sensitive to the statistical test and fold setting and should be treated as suggestive rather than conclusive.

At the matched 4 s duration, ExtraTrees obtained a mean 2D error of 0.387 m, compared with 0.677 m for the Transformer. The paired sign-flip test produced $p < 0.005$. This result must nevertheless be interpreted cautiously because a 4 s Transformer input contains only eight 500 ms time steps, considerably fewer than its native 30-step sequence.

Matching input duration does not make the two model inputs identical. ExtraTrees consumes engineered tabular features, whereas the Transformer consumes an RSSI sequence. Moreover, the statistical comparisons pair fold-level errors rather than predictions on identical windows. Experiment 3 therefore controls input duration but does not isolate architecture from input representation.

The native Transformer advantage reported in Table 14 does not persist when the two models are compared at the same input duration: ExtraTrees has the lower mean error at both 4 s and 15 s. The mixed statistical evidence at 15 s, however, prevents a general claim that ExtraTrees is architecturally superior. The supported conclusion is that input-window duration is an important confounding factor and that the native performance difference cannot be attributed to model architecture alone.

8.4 Experiment 4: Rooms and heights (3D)

The aggregate error was not distributed evenly over the floor. Disaggregating the ExtraTrees results by room (Table 37, Appendix B) shows three rooms performing well: Kitchen, Office, and Lab fell between 83 and 90% accuracy, with mean errors of 0.32 to 0.44 m. The Meeting Room was the clear low-performing outlier: at $K = 5$ it reached 44.5% accuracy with a mean error of 1.245 m, and at $K = 10$ it improved to 59.5% accuracy and 0.944 m. The room contains only two grid cells and the fewest reports of the four rooms (Chapter 5). Each fold therefore contains limited Meeting Room training data, and this limited support is consistent with its lower accuracy.

Height separation, by contrast, is handled effectively. When the ExtraTrees composite prediction is interpreted as a binary height decision, it is correct in 93.8% $\pm$ 1.7% of windows at $K = 5$. The Transformer's mean height error at $K = 10$ is 0.038 ± 0.032 m, against the 0.95 m separation between the two height labels. This explains why the 3D errors of the leading models in Tables 13 and 14 exceed their 2D counterparts by at most approximately two centimetres. Pooling the window errors over the five $K = 5$ test folds provides a distributional view rather than a fold-mean view. In that view, 85.6% of ExtraTrees window errors fall within 1 m and 93.7% within 2 m. The 90th percentile is 2.0 m. The model is usually assigned to the correct cell and is occasionally displaced by one or two cells.

8.5 Experiment 5: Random versus chronological split

Experiment 5 quantifies how the train–test split changes the reported performance on the tabular CS3D windows. The same 7,006 windows, model configurations, and value of $K = 5$ were evaluated under two protocols:

1. the chronological within-class split used for the main results;
2. a stratified random split that preserves class proportions but removes the temporal ordering.

For each metric, the protocol effect is calculated as the random-split result minus the chronological result. A positive difference in grid accuracy indicates higher reported accuracy under random splitting, whereas a negative difference in localisation error indicates lower reported error.

Table 17 presents the audit for ExtraTrees, the best-performing classical model at $K = 5$. Random splitting increased mean grid accuracy from 84.3% to 87.1%, a difference of 2.8 percentage points. Mean 2D error decreased from 0.434 m to 0.371 m, corresponding to an absolute reduction of 0.063 m and a relative reduction of approximately 14.5%. The standard deviation of grid accuracy across folds decreased from 6.5 to 0.7 percentage points.

Table 17: **Random versus chronological split for ExtraTrees.** Results on the same 7,006 tabular windows at $K = 5$. Grid accuracy is reported as mean ± sample standard deviation across folds; 2D MAE is reported as the mean across folds. The difference is calculated as random minus chronological.

Metric	**Chronological**	**Random**	**Difference**
Grid accuracy	$84.3 \pm 6.5\%$	$87.1 \pm 0.7\%$	+2.8 pp
2D MAE	0.434 m	0.371 m	−0.063 m (−14.5%)

The direction of the accuracy effect was consistent across the evaluated tabular models. Random splitting increased mean grid accuracy by up to 2.9 percentage points for the eight classical models and by approximately 2.0–2.2 percentage points for the two tabular neural networks. The complete per-model results are reported in Table 35 of Appendix B.

The two protocols measure different conditions. A stratified random split distributes windows from different recording times across both training and test folds. Temporally adjacent windows can therefore occur on opposite sides of the split, despite being recorded under closely related signal conditions. The chronological protocol instead holds out contiguous time-ordered blocks within each class and evaluates models on temporally separated windows. The random-split results are consequently used as a protocol audit rather than as the main estimate of generalisation across time.

Sequence models are excluded from this audit because their sliding windows overlap substantially. Each sequence spans 15 s and advances by five 500 ms bins, giving a stride of 2.5 s and an overlap of 12.5 s between consecutive windows. A random assignment could therefore place sequences sharing most of their temporal content in both the training and test folds. Such a comparison would combine the effect of random splitting with direct temporal overlap. The chronological sequence split and its train–test gap are defined in Chapter 7.

The audit answers RQ3. Across the evaluated tabular models, random splitting produced higher reported grid accuracy. For ExtraTrees, it also reduced mean 2D error by approx-

imately 14.5% and reduced the fold-level grid-accuracy standard deviation from 6.5 to 0.7 percentage points. The principal conclusions of this thesis are therefore based on the chronological within-class evaluation protocol.

Chapter 9

Evaluation on Public Datasets and Published References

This chapter evaluates k-NN and ExtraTrees on selected public indoor-localisation datasets. The experiments examine four aspects of the CS3D findings: reproduction of a published reference, sensitivity to the data-splitting protocol, the effect of engineered features, and application of the classical pipeline in an independent building. Published results are treated as direct comparisons only when the model, split, and metric are sufficiently aligned; otherwise, they provide contextual evidence.

Table 18 summarises the selected datasets, methods, selection reasons, and supported scope. Sequence models are excluded because the selected public datasets provide fixed-length fingerprints rather than raw timestamped receiver streams from which the CS3D temporal windows can be reconstructed.

Table 18: **Selection and role of the public datasets.** Methods, reasons for selection, and scope of the evidence provided.

Dataset	Methods	Reason for selection	Scope
HDLC [41, 42]	k-NN and ExtraTrees	Provides a published k-NN reference and data suitable for examining split sensitivity	Reference reproduction and split-protocol comparison
Daniş [17, 16]	k-NN and ExtraTrees	Moving BLE beacon with continuous ground truth in an independent office environment	Re-application of the classical pipeline; published HMM results are contextual only
Assayag [5]	ExtraTrees	BLE fingerprints from 15 fixed receivers	Feature-ablation evidence only
Tehran [56]	ExtraTrees	Lower-dimensional processed fingerprint data	Feature-ablation evidence only

9.1 HDLC published-reference check

The HDLC paper reports k-NN results for hybrid Wi-Fi and BLE fingerprints across three layouts and floors [41]. The implementation used in this thesis was evaluated under the corresponding same-layout condition.

Table 19 shows that the reproduced result at $k = 5$ differs from the published value by 0.01 m. This provides a setting-specific consistency check but does not validate the

complete evaluation procedure or the results on other datasets.

Table 19: **HDLC published-reference check.** Mean 2D localisation error under the same-layout condition. Difference is calculated as the reproduced value minus the published value.

Setting	Ours (m)	Published (m)	Difference (m)
k-NN, $k = 1$	2.16	1.85	+0.31
k-NN, $k = 5$	**1.94**	**1.95**	−0.01

9.2 Effect of the split protocol

ExtraTrees was evaluated on HDLC using random and order-preserving splits. Table 20 compares this result with the corresponding CS3D audit. Random splitting reports lower error on both datasets, but the effect size differs. The comparison therefore demonstrates sensitivity to the split rule rather than a universal correction factor.

Table 20: **Split-protocol effect on HDLC and CS3D.** ExtraTrees mean 2D localisation error. Difference is calculated as the order-preserving result minus the random-split result.

Dataset	Random (m)	Order-preserving (m)	Difference (m)
HDLC	≈ 0.17	≈ 0.33	$\approx +0.16$
CS3D	0.371	0.434	+0.063

9.3 Ablation of the engineered features

ExtraTrees was evaluated with raw RSSI and engineered features while the model and split were held fixed within each dataset. HDLC and Tehran use position-held-out GroupKFold evaluation so that complete reference positions are withheld from training.

Table 21 shows changes between −0.014 and +0.039 m. The engineered representation reduces error in four of the five conditions but increases it in one HDLC condition. The observed effects are therefore small and inconsistent.

9.4 Evaluation on the Daniş dataset

The Daniş dataset contains continuous trajectories of a moving BLE beacon in an office of approximately 364 m^2, recorded using 12 fixed Bluetooth sensors and high-precision ground truth [17]. The models were trained and evaluated on this dataset using leave-one-trajectory-out validation; a model trained on CS3D was not transferred directly.

As shown in Table 22, raw-RSSI k-NN obtains the lowest error among the three evaluated configurations. The published HMM range is included only for context because its preprocessing, parameter selection, and evaluation protocol differ [16].

Table 21: **Feature ablation across datasets.** ExtraTrees mean 2D localisation error using raw RSSI and engineered features. Gain is raw-RSSI error minus engineered-feature error, so a positive value favours the engineered representation.

Dataset	RSSI inputs	Raw (m)	Engineered (m)	Gain (m)
CS3D	17	0.447	0.434	+0.012
Tehran	3	3.855	3.815	+0.039
Assayag	15	2.109	2.070	+0.039
HDLC	42	2.079	2.055	+0.024
HDLC	59	1.967	1.981	−0.014

Table 22: **Evaluation on the Daniş trajectory dataset.** Mean 2D localisation error. The published HMM range uses different evaluation conditions and is not directly comparable.

Configuration	Protocol	Mean error (m)
k-NN, raw RSSI	Leave one trajectory out	**1.359**
ExtraTrees, engineered features	Leave one trajectory out	1.732
ExtraTrees, raw RSSI	Leave one trajectory out	1.767
Published HMM [16]	Authors' trajectory-specific optimisation	*1.973–2.514*

9.5 Summary

The HDLC reference check reproduces the published $k = 5$ result within 0.01 m. The split audit shows that random splitting reports lower error on both HDLC and CS3D, although with different effect sizes. The feature ablation provides no consistent evidence that the engineered features materially improve ExtraTrees. On Daniş, raw-RSSI k-NN performs better than the evaluated ExtraTrees configurations.

These results provide external context for the CS3D findings. They do not validate the absolute CS3D errors, establish a universal model ranking, or demonstrate state-of-the-art performance.

Chapter 10

Discussion and Synthesis

This chapter interprets the experimental findings in relation to the three research questions. Table 23 summarises the principal evidence, the conclusions supported by that evidence, and the scope of each conclusion.

Table 23: **Synthesis of the research questions.** Principal evidence, supported interpretation, and scope of each result.

Research question	Principal evidence	Supported interpretation
RQ1: Accuracy	At $K = 10$, the Transformer obtains a mean 2D localisation error of 0.312 m, compared with 0.387 m for ExtraTrees. The paired sign-flip test gives $p = 0.187$. ExtraTrees obtains the lower mean error at both matched input durations.	The experiments do not provide consistent evidence that the evaluated deep neural networks achieve lower localisation error than ExtraTrees. The native comparison is affected by differences in input duration and representation. The conclusion is limited to CS3D and the evaluated protocols.
RQ2: Deployment characteristics	The Transformer has a model file size of 0.45 MB and a measured CPU inference latency of approximately 1.5 ms, with a 15 s input duration. ExtraTrees requires 999.5 MB and approximately 38 ms, with a 4 s input duration.	The measured characteristics reveal a trade-off between observation duration, model storage, and inference latency. They do not identify one model as universally preferable. Training, integration, energy, and maintenance costs were not measured.
RQ3: Evaluation protocol	For ExtraTrees, random splitting changes grid accuracy from 84.3% to 87.1% and mean 2D localisation error from 0.434 m to 0.371 m. The HDLC experiment shows the same direction.	Random splitting reports more favourable performance than chronological splitting. The magnitude of the difference is dataset-specific, and no universal correction factor can be inferred.

10.1 RQ1: Accuracy

The Transformer achieves the lowest mean localisation error in its native configuration. However, the paired sign-flip test does not reject the null hypothesis of no mean fold-level difference between the Transformer and ExtraTrees. The result therefore does not establish that the Transformer is more accurate under the evaluated protocol.

The native comparison also differs in more than model architecture. ExtraTrees uses a 4 s engineered tabular representation, whereas the Transformer uses a 15 s RSSI sequence. In the matched-window experiment, ExtraTrees obtains the lower mean error at both 4 s and 15 s. The native Transformer result therefore cannot be attributed to architecture alone.

These findings do not establish that the models are equivalent, nor do they demonstrate that ExtraTrees is generally superior. They show that, under the evaluated conditions, the evidence is insufficient to support a consistent accuracy advantage for the evaluated deep neural networks over the strongest classical model.

10.2 RQ2: Deployment characteristics

The measured deployment characteristics show different trade-offs. The Transformer has a substantially smaller model file and lower native CPU inference latency, whereas ExtraTrees requires a shorter observation duration before a prediction can be produced.

Reducing the number of ExtraTrees estimators or exporting the model to another format decreases its storage and inference requirements, but also reduces predictive accuracy, as reported in Appendix A.5. The measured comparison therefore concerns a trade-off between model size, inference latency, input duration, and predictive performance.

The analysis does not cover the complete cost of deployment. Training time, data-collection effort, system integration, energy consumption, maintenance, and model-update costs were not measured. Consequently, the results support conclusions only about the reported model file sizes, CPU inference latencies, and input durations.

10.3 RQ3: Evaluation protocol

The random-split audit reports higher grid accuracy and lower localisation error than the chronological protocol. The HDLC evaluation shows the same direction, although the magnitude of the difference is different. This supports the conclusion that reported performance is sensitive to the split rule when temporally related observations are present.

The observed difference should not be interpreted as a fixed estimate of performance inflation. Its magnitude depends on the dataset, sampling process, temporal correlation, and evaluation design.

Sequence models are excluded from the random-split audit because consecutive sequence windows overlap substantially. Random assignment could therefore place windows containing shared signal observations in both the training and test sets. The chronological results, with the specified temporal-gap procedure, remain the basis of the principal conclusions.

10.4 Synthesis of the findings

The combined evidence shows that model architecture alone does not explain the strongest CS3D results. Input duration, input representation, prediction formulation, and evaluation

protocol all affect the measured performance.

The engineered features produce only a small improvement for ExtraTrees on CS3D, and the public-dataset ablations show similarly small and inconsistent changes. The evidence therefore does not demonstrate a substantial general benefit from the engineered representation.

The CS3D grid-classification configuration also produces lower reported errors than the evaluated public-data regression baselines. However, these experiments differ in dataset, model, prediction formulation, ground truth, and evaluation protocol. The difference cannot therefore be attributed to grid classification alone, and the numerical results should not be treated as a direct cross-dataset ranking.

Within the evaluated CS3D setting, no consistent accuracy advantage is established for the deep neural network models over ExtraTrees. The Transformer provides lower model-storage and native inference-latency requirements, whereas ExtraTrees provides a shorter input duration. These are measured trade-offs rather than evidence that either model is universally preferable.

The broader methodological conclusion is that model-family comparisons in RSSI localisation should control or explicitly report differences in input duration, representation, decoding, and evaluation protocol. Model selection should be based on an order-preserving evaluation and repeated using the intended deployment hardware and operating conditions.

Chapter 11

Limitations and Future Work

The findings of this thesis are bounded by the dataset, model-training procedure, evaluation protocol, and ground-truth design. This chapter defines those limitations and identifies the corresponding directions for future work.

11.1 Dataset and deployment

The main evidence comes from one BLE tag, one office floor, and one principal survey session. The results therefore do not establish generalisation across buildings, receiver layouts, devices, or recording days.

The Daniş transfer experiment illustrates this limitation. Under leave-one-trajectory-out evaluation, raw-RSSI k-NN obtained a lower mean error than feature-engineered ExtraTrees, 1.359 m versus 1.732 m (Chapter 9). A model that performs strongly on CS3D therefore need not retain the same ranking in another environment.

The spatial task is also restricted. The vertical component contains two discrete tag heights within one floor rather than a continuous vertical coordinate. Room coverage is imbalanced, with only two of the 42 cells located in the Meeting Room (Table 6), and this imbalance is not explicitly corrected during training.

11.2 Models and training

The main comparison uses one shared training procedure and a fixed experimental budget. This improves comparability but may not optimise every architecture equally. FusionNet and the anchor-attention model failed to converge under the main procedure and were reported at their collapsed values. Their improvement under model-specific tuning (Appendix A.3) indicates optimisation sensitivity rather than demonstrating an inherent lack of model capacity.

The sequence models are evaluated only on CS3D. The selected public fingerprint datasets do not provide raw timestamped streams from which the same temporal windows can be reconstructed. Their results therefore cannot be transferred directly using the sequence pipeline.

Cross-session robustness is also limited. During the different-day live replay, ExtraTrees obtained 16.4% cell accuracy and the Transformer approximately 2.9% (Appendix C). The offline error levels and model ranking therefore do not transfer reliably to a later recording session.

11.3 Evaluation protocol

The comparison between the strongest classical and neural models uses ten folds, which limits statistical power. These models were selected after the family-level results were observed, and the paired test does not correct for this selection step. Its result should therefore be interpreted as evidence for this specific comparison rather than as a confirmatory test covering all model families.

Path-loss parameters were initially calibrated before the folds were formed. Repeating the calibration within each fold changed ExtraTrees 2D error by approximately 0.01 m and changed $K = 5$ grid accuracy by 0.3 percentage points, with no accuracy change at $K = 10$. The audit therefore found no material effect on the conclusions, although fold-local calibration is methodologically preferable.

The sequence split uses a one-sided five-window training gap. Residual temporal overlap may remain at the opposite boundary of a held-out chunk. The sequence results may therefore retain limited temporal dependence. A random split is not reported for these models because overlapping sequences could place substantially shared signal content in both training and test folds.

11.4 Measurement and ground truth

The dataset contains one BLE tag type, so device-specific RSSI offsets are not evaluated [47]. Ground-truth positions were placed manually during collection and may contain small, unquantified placement deviations.

Both predictions and labels are represented by fixed cell centroids. The reported distance therefore measures error between predicted and labelled centroids and does not separately quantify within-cell positioning error. Consequently, the evaluation describes discrete grid localisation rather than unrestricted continuous localisation.

Results from different datasets are not directly interchangeable because their buildings, receiver layouts, labels, and evaluation protocols differ. The public-dataset results provide contextual and transfer evidence rather than a common leaderboard.

11.5 Future work

Future work should prioritise cross-session and cross-environment validation before increasing model complexity.

- **Cross-session and cross-building validation.** Collect data across multiple days, buildings, layouts, and receiver geometries, and report leave-one-session-out and leave-one-building-out performance.
- **Deployment-shift robustness.** Evaluate controlled changes in occupancy, furniture placement, receiver availability, and signal conditions.

- **Cross-session adaptation.** Investigate recalibration, domain adaptation, continual learning, and drift detection to reduce different-day performance degradation.
- **Model optimisation.** Use architecture-specific tuning under equal compute budgets, multiple random seeds, and predefined model-selection rules.
- **Sequence evaluation.** Apply strict two-sided temporal gaps and develop sequence models that tolerate missing or unheard receivers during live operation.
- **Cross-device calibration.** Collect data from multiple tag and receiver units, quantify device-specific RSSI offsets, and evaluate calibration or device-invariant representations.
- **Ground truth and vertical localisation.** Use higher-precision tracking, add more heights and floors, and report centroid discretisation and model error separately.
- **Survey and deployment efficiency.** Explore trajectory-assisted or semi-supervised labelling and extend the system to multiple simultaneously tracked assets on edge hardware.

Chapter 12

Conclusion

This study investigated BLE-based 3D indoor localisation using an existing Crownstone mesh network. It contributed the CS3D dataset and compared classical and deep neural models under a chronological evaluation protocol.

The Transformer achieved the lowest native mean 2D error, 0.312 m, compared with 0.387 m for ExtraTrees. However, the primary paired test did not establish this difference, and ExtraTrees obtained the lower mean error when the models were compared at matched input durations. The results therefore provide no consistent evidence that greater neural complexity improves localisation accuracy under the evaluated conditions. They also show that model selection involves different deployment trade-offs: the Transformer has a smaller saved artefact and lower native inference latency, whereas ExtraTrees requires a shorter observation window.

The evaluation protocol materially affected the reported performance. For ExtraTrees, random splitting reduced mean 2D error from 0.434 m to 0.371 m and produced substantially lower fold-level variation than the chronological protocol. The public-data evaluation showed the same direction on HDLC, although with a different effect size. These findings support the use of order-preserving evaluation when temporal correlation is present.

The main methodological conclusion is that model architecture cannot be interpreted independently of input duration, representation, prediction formulation, and evaluation protocol. In these experiments, engineered features provided only small and inconsistent improvements, while the grid-classification pipeline produced substantially lower error than the evaluated regression baselines. Because these configurations differ in more than one component, the improvement cannot be attributed to grid classification alone.

The conclusions are limited to one office, one BLE tag, and one principal survey session. The marked degradation observed during different-day live replay shows that cross-session robustness remains unresolved. Chapter 11 describes these limitations and the corresponding future work. Overall, the thesis shows that careful evaluation and pipeline design are at least as important as model complexity for the investigated deployment.

References

[1] Abdulrahman Alarifi, AbdulMalik Al-Salman, Mansour Alsaleh, Ahmad Alnafessah, Suheer Al-Hadhrami, Mai A. Al-Ammar, and Hend S. Al-Khalifa. Ultra wideband indoor positioning technologies: Analysis and recent advances. *Sensors*, 16(5):707, 2016.

[2] Wilford Arigye, Qiaolin Pu, Mu Zhou, Waqas Khalid, and Muhammad Junaid Tahir. RSSI fingerprint height based empirical model prediction for smart indoor localization. *Sensors*, 22(23):9054, Nov 2022.

[3] Aris J. Aristorenas. Permutation-invariant transformer neural architectures for set-based indoor localization using learned RSSI embeddings, 2025.

[4] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[5] Yuri Assayag, Horacio Oliveira, Max Lima, João Junior, Mateus Preste, Leonardo Guimarães, and Eduardo Souto. Indoor environment dataset based on RSSI collected with Bluetooth devices. *Data in Brief*, 55:110692, 2024.

[6] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)*, volume 2, pages 775–784, 2000.

[7] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271, 2018.

[8] Christoph Bergmeir and José M. Benítez. On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191:192–213, 2012.

[9] Jan Blumenthal, Ralf Grossmann, Frank Golatowski, and Dirk Timmermann. Weighted centroid localization in Zigbee-based sensor networks. In *IEEE International Symposium on Intelligent Signal Processing*, pages 1–6, 2007.

[10] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[11] Vitor Cerqueira, Luís Torgo, and Igor Mozetič. Evaluating time series forecasting models: an empirical study on performance estimation methods. *Machine Learning*, 109(11):1997–2028, 2020.

[12] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[13] Marco Cominelli, Paul Patras, and Francesco Gringoli. Dead on arrival: An empirical study of the Bluetooth 5.1 positioning system. In *Proceedings of the 13th International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH)*, pages 13–20, 2019.

[14] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[15] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[16] F. Serhan Daniş, Cem Ersoy, and A. Taylan Cemgil. Probabilistic indoor tracking of Bluetooth Low-Energy beacons. *Performance Evaluation*, 162:102374, 2023.

[17] F. Serhan Daniş, A. Teoman Naskali, A. Taylan Cemgil, and Cem Ersoy. An indoor localization dataset and data collection framework with high precision position annotation. *Pervasive and Mobile Computing*, 81:101554, 2022.

[18] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[19] Ramsey Faragher and Robert Harle. Location fingerprinting with bluetooth low energy beacons. *IEEE Journal on Selected Areas in Communications*, 33(11):2418–2428, 2015.

[20] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.

[21] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[22] Ashkan Goharfar, Jaber Babaki, Mehdi Rasti, and Pedro H. J. Nardelli. Indoor positioning via gradient boosting enhanced with feature augmentation using deep learning, 2022.

[23] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2):107–113, 1993.

[24] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *Advances in Neural Information Processing Systems (NeurIPS) 35, Datasets and Benchmarks Track*, 2022.

[25] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 1321–1330, 2017.

[26] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[27] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[28] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems (NeurIPS 2017)*, volume 30, 2017.

[29] Chia Wei Khor and Nur Syazreen Ahmad. BLE-based indoor localization with temporal convolutional network. *Journal of Engineering Research*, 2026. ScienceDirect PII S2307187725000549.

[30] Marcin Kołakowski. Kalman filter based localization in hybrid BLE-UWB positioning system. In *2017 IEEE International Conference on RFID Technology and Application (RFID-TA)*, pages 290–293, 2017.

[31] Marcin Kołakowski. Comparison of extended and unscented kalman filters performance in a hybrid BLE-UWB localization system. In *2020 23rd International Microwave and Radar Conference (MIKON)*, pages 122–126, 2020.

[32] Meelis Kull, Miquel Perelló-Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. Beyond temperature scaling: Obtaining well-calibrated multiclass probabilities with Dirichlet calibration. In *Advances in Neural Information Processing Systems (NeurIPS 2019)*, volume 32, 2019.

[33] Steffen Limmer, Alberto Martinez Alba, and Nicola Michailow. Physics-informed neural networks for pathloss prediction. In *2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP)*, 2023.

[34] Dimitrios Lymberopoulos and Jie Liu. The Microsoft indoor localization competition: Experiences and lessons learned. *IEEE Signal Processing Magazine*, 34(5):125–140, 2017.

[35] Vishakha K. Marathe. Indoor in-network asset localization using Crownstone network. Master's thesis, Delft University of Technology, 2023.

[36] Sara Martín-Frechina, Esther Dura, Ignacio Miralles, and Joaquín Torres-Sospedra. From fingerprinting to advanced machine learning: A systematic review of Wi-Fi and BLE-based indoor positioning systems. *Sensors*, 25(22):6946, 2025. PMC ID: PMC12656469.

[37] Germán Martín Mendoza-Silva, Miguel Matey-Sanz, Joaquín Torres-Sospedra, and Joaquín Huerta. BLE RSS measurements dataset for research on accurate indoor positioning. *Data*, 4(1):12, 2019.

[38] Filippo Milano, Helbert da Rocha, Marco Laracca, Luigi Ferrigno, António Espírito Santo, José Salvado, and Vincenzo Paciello. BLE-based indoor localization: Analysis of some solutions for performance improvement. *Sensors*, 24(2):376, 2024.

[39] Ahasanun Nessa, Bhagawat Adhikari, Fatima Hussain, and Xavier N. Fernando. A survey of machine learning for indoor positioning. *IEEE Access*, 8:214945–214965, 2020.

[40] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 625–632, 2005.

[41] Aina Nadhirah Nor Hisham, Yin Hoe Ng, Chee Keong Tan, and David Chieng. Hybrid Wi-Fi and BLE fingerprinting dataset for multi-floor indoor environments with different layouts. *Data*, 7(11):156, November 2022.

[42] Aina Nadhirah Nor Hisham, Yin Hoe Ng, Chee Keong Tan, and David Chieng. Hybrid Wi-Fi and BLE fingerprinting dataset for multi-floor indoor environments with different layouts, November 2022. Dataset deposit; CC-BY-4.0; HDLC.zip 7.5 MB, MD5 ca63cab48f521ab47821f16852f45a74.

[43] Nahshon Mokua Obiri and Kristof Van Laerhoven. Environment-aware indoor LoRaWAN ranging using path loss model inversion and adaptive RSSI filtering, 2025.

[44] Ronald K. Pearson, Yrjö Neuvo, Jaakko Astola, and Moncef Gabbouj. Generalized Hampel filters. *EURASIP Journal on Advances in Signal Processing*, 2016(1):87, 2016.

[45] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Alexander J. Smola, Peter L. Bartlett, Bernhard Schölkopf, and Dale Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74, Cambridge, MA, 1999. MIT Press.

[46] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: Unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[47] Jingshi Qian, Nobuyoshi Komuro, Won-Suk Kim, and Younghwan Yoo. Enhanced position estimation via RSSI offset correction in BLE fingerprinting-based indoor positioning. *Future Internet*, 17(10):440, 2025.

[48] Jingshi Qian, Jiahe Li, Nobuyoshi Komuro, Won-Suk Kim, and Younghwan Yoo. Improved particle filter in machine learning-based BLE fingerprinting method to reduce indoor location estimation errors. *Future Internet*, 16(6):211, 2024.

[49] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[50] Theodore S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, 2 edition, 2002.

[51] Priya Roy and Chandreyee Chowdhury. A survey of machine learning techniques for indoor localization and navigation systems. *Journal of Intelligent and Robotic Systems*, 101(3):63, 2021.

[52] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[53] Sebastian Sadowski and Petros Spachos. RSSI-based indoor localization with the Internet of Things. *IEEE Access*, 6:30149–30161, 2018.

[54] Shuai Shang and Lixin Wang. Overview of WiFi fingerprinting-based indoor positioning. *IET Communications*, 2022.

[55] Taolei Shi and Wei Gong. A survey of Bluetooth indoor localization. *arXiv preprint arXiv:2404.12529*, 2024.

[56] Behrad Mousaei Shir-Mohammad, Behzad Moshiri, and Abolfazl Yaghmaei. Topology-aware hybrid Wi-Fi/BLE fingerprinting via evidence-theoretic fusion and persistent homology. arXiv preprint arXiv:2510.16557, 2025. Code and data: `https://github.com/behradbx/Tehran-Indoor-Localization`.

[57] Navneet Singh, Sangho Choe, Rajiv Punmiya, and Navneesh Kaur. XGBLoc: XGBoost-based indoor localization in multi-building multi-floor environments. *Sensors*, 22(17):6629, Sept 2022.

[58] 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)*, pages 261–270, 2014.

[59] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS 2017)*, volume 30, 2017.

[60] Jinjin Yan, Manyu Zhang, Fuqiang Gu, and You Li. A novel deployment strategy based on improved gradient-based optimizer for BLE anchor nodes. *Transactions in GIS*, 29:e70012, 2025.

[61] Shangyi Yang, Chao Sun, and Youngok Kim. Indoor 3D localization scheme based on BLE signal fingerprinting and 1D convolutional neural network. *Electronics*, 10(15):1758, July 2021.

[62] Faheem Zafari, Athanasios Gkelias, and Kin K. Leung. A survey of indoor localization systems and technologies. *IEEE Communications Surveys and Tutorials*, 21(3):2568–2599, 2019.

[63] Lieping Zhang, Yao Chen, Xiaolin Gao, Xinpeng Zheng, and Cui Zhang. An algorithm for indoor positioning based on LightGBM + ExtraTrees chaotic weighted ensemble: evaluation and comparison. *The Journal of Supercomputing*, 81(7), May 2025.

Appendix A

Feature and model specification

A.1 Per-anchor path-loss calibration

Table 24 lists the per-anchor parameters of the log-distance path-loss calibration described in Section 6.3.

Table 24: **Per-anchor log-distance path-loss calibration**, fitted from CS3D. P_{tx} is the reference power and n the path-loss exponent; N is the number of surveyed positions used. Anchors whose reference power was estimated from at least two close-range positions are flagged reliable; the rest fall back to a least-squares fit and are flagged unreliable. These parameters drive the physics features of Table 12.

Anchor	P_{tx} (dBm)	n	N	Method	Reliable
1	-64.16	1.000	11	joint_ols	no
2	-60.02	1.689	18	median_close	yes
3	-60.27	1.572	35	median_close	yes
4	-57.36	2.066	14	median_close	yes
5	-57.98	1.965	30	median_close	yes
6	-58.99	1.437	17	joint_ols	no
7	-58.75	2.276	26	median_close	yes
9	-61.28	1.334	31	joint_ols	no
10	-66.60	1.000	23	joint_ols	no
11	-63.53	1.085	37	joint_ols	no
12	-58.46	1.668	35	joint_ols	no
13	-62.70	1.998	27	joint_ols	no
31	-34.45	5.413	24	joint_ols	no
32	-36.47	5.242	17	joint_ols	no
33	-52.46	2.676	34	joint_ols	no
34	-41.70	3.888	23	joint_ols	no
41	-62.20	1.125	38	joint_ols	no

A.2 Model hyperparameters

The main comparison holds every value at the fixed, documented settings introduced in Section 6.4, with the classical configurations as listed in Table 10. Beyond those, the tree ensembles take unscaled features and the remaining classical models standardised features. The standardiser is fitted on the training portion only. The boosted ensembles hold out 10% of each training portion for early stopping. Parameters not listed are left at their library defaults. Table 25 lists the neural architectures and Table 26 their training settings. A separate per-model hyperparameter search is reported at the end of this section (Section A.3, Table 27).

Table 25: **Architecture of the neural families.** Parameters not listed are library defaults.

Model	Architecture
Deep MLP	$144 \to 256 \to 128 \to 64 \to 84$; batch norm; dropout 0.30, 0.20, 0.10
PINN	Deep MLP backbone with a weighted-centroid physics penalty, $\lambda = 0.10$
1D CNN	Conv 32-64-128 (kernels 3, 5, 3) with batch norm and pooling; head dropout 0.3
LSTM	2 layers, hidden 128, dropout 0.3
CNN-LSTM	per-step Conv (32, 64) then BiLSTM (2 layers, hidden 128, concat 256); dropout 0.3
TCN	4 dilated causal blocks (dilations 1, 2, 4, 8), kernel 3, weight norm; dropout 0.10
Transformer	2 layers, $d_{\text{model}} = 64$, 4 heads, $d_{\text{ff}} = 256$, dropout 0.10; learnable positional encoding
FusionNet	statistical MLP branch ($\to 64$) and convolutional branch ($\to 64$), late fusion; dropout 0.3
Anchor-attention	per-anchor embedding Linear($1 \to 32$) with surveyed (x, y, h); 2 attention layers, $d = 64$, 4 heads, dropout 0.10

All neural families share the optimisation recipe of Section 6.4. The learning-rate schedule is a ReduceLROnPlateau of factor 0.5 and patience 8 to 10, and anchor-attention additionally clips gradients to a maximum norm of 1.0. The sequence models consume windows of 30 bins (15 s) over the 17 anchors at a 500 ms bin and a stride of 5 bins, z-scored per anchor, with a fourfold dropout augmentation that masks 20% of entries. The per-model epochs, batch size, learning rate, and early-stopping patience are given in Table 26.

Table 26: **Training settings of the neural families.** All share the recipe stated in the text (Adam, weight decay 10^{-5}, ReduceLROnPlateau, 15% validation slice, seed 42); the columns list the values that differ between models. Parameters not listed are library defaults.

Model	Epochs	Batch	Learning rate	Early-stop patience
Deep MLP	300	128	10^{-3}	30
PINN	300	128	10^{-3}	30
1D CNN	200	64	10^{-3}	20
LSTM	80	128	10^{-3}	12
CNN-LSTM	300	64	10^{-3}	30
TCN	300	64	10^{-3}	30
Transformer	300	64	10^{-3}	30
FusionNet	120	64	5×10^{-4}	25
Anchor-attention	300	128	10^{-3}	30

A.3 Hyperparameter sensitivity

To confirm that the fixed settings above do not bias the model comparison, every one of the seventeen families was additionally tuned with an Optuna tree-structured Parzen estimator search of thirty trials. Each trial was scored by composite 84-class grid accuracy, averaged over the same chronological $K = 5$ folds, features, and sequence windows as the main comparison. The standardised classical models (SVM, k-NN, MLP) were searched inside a per-fold standardiser, matching their wrappers. For each neural-network model the documented default configuration was evaluated as the first trial, so its tuned value in Table 27 cannot fall below its default within the search. The classical searches did not enqueue the default, although every classical tuned value still exceeded its default in practice. Table 27 reports, per model, the fixed-default grid accuracy, the best tuned grid accuracy, and their difference.

Table 27: **Tuned versus fixed-default grid accuracy.** For each model, the best composite 84-class grid accuracy from an Optuna search (30-trial target), against the fixed default used in the main comparison. Both numbers are means over the same chronological within-class $K = 5$ folds, and Δ is tuned minus default. The rows fall into two groups that are not directly comparable: the classical and tabular models are scored on 4 s windows (about 1,433 test windows per fold), while the sequence and anchor models are scored on 15 s windows (about 2,375 test windows per fold). Accuracies should therefore be compared within a group, not across the two groups. A dagger (†) marks searches that did not reach the 30-trial budget within the cluster walltime (CNN-LSTM 6, Transformer 19, anchor-attention 11 trials); these tuned values are lower bounds. Differences below about one point lie within the fold-to-fold standard deviation of roughly 0.05 to 0.07 and are not treated as significant. Each neural-network search evaluated the model's own default as its first trial, so a neural tuned value cannot fall below its default; the classical searches did not.

Model	Family	Window	Default	Tuned	Δ
ExtraTrees	Classical	4 s	0.843	0.847	+0.004
XGBoost	Classical	4 s	0.837	0.857	+0.020
LightGBM	Classical	4 s	0.836	0.862	+0.026
CatBoost	Classical	4 s	0.835	0.856	+0.021
Random Forest	Classical	4 s	0.838	0.841	+0.003
MLP	Classical	4 s	0.695	0.724	+0.029
SVM	Classical	4 s	0.663	0.670	+0.007
k-NN	Classical	4 s	0.385	0.634	+0.249
DeepMLP	Tabular NN	4 s	0.778	0.797	+0.019
PINN	Tabular NN	4 s	0.775	0.800	+0.025
LSTM	Sequence NN	15 s	0.851	0.907	+0.056
1D CNN	Sequence NN	15 s	0.812	0.839	+0.027
TCN	Sequence NN	15 s	0.832	0.893	+0.061
CNN-LSTM	Sequence NN	15 s	0.821	0.821†	+0.000
Transformer	Sequence NN	15 s	0.883	0.894†	+0.011
FusionNet	Sequence NN	15 s	0.059	0.845	+0.786
Anchor-Attention	Anchor NN	15 s	0.059	0.495†	+0.436

The table must be read within a single input protocol. The classical and tabular models use 4 s windows. The sequence and anchor models use 15 s windows over a larger test set, so the tuned column is comparable only within a protocol, not across the whole table. Three patterns follow. First, the families that already converge move by at most a few points. The bagged tree ensembles barely change (ExtraTrees gains +0.004 and random forest +0.003), the boosted ensembles gain about two points (LightGBM is the strongest tuned tree at 0.862), and the tabular networks gain about two points. These gaps among the strong tabular models are within the baseline fold-to-fold standard deviation of about 0.05 to 0.07, so they are not treated as significant. Among the sequence models the LSTM and TCN gain the most, about five to six points, but stay within the band already covered by the default sequence models. Second, large gains occur only where a default was clearly poor. The k-nearest-neighbour accuracy rises from 0.385 to 0.634 once the neighbour count moves from five into the low twenties, and the two models that collapsed under the shared recipe, FusionNet and anchor-attention, recover from near-chance (≈ 0.06) to 0.845 and 0.495. Because these defaults collapsed almost identically across folds (standard deviation ≈ 0.0004), the recovery reflects an optimisation and stability problem rather than a real capacity gain. Third, the two heaviest sequence searches (CNN-LSTM and Transformer) and the anchor search did not use their full trial budget, so those tuned values are lower bounds. None of the tuned settings were adopted; the main comparison and the deployed model keep the fixed defaults of Section 6.4.

A.4 Compute environment

Every randomised component uses a fixed seed of 42. Classical models were trained on CPU; the neural models were trained on the ALICE cluster GPU partitions using the project virtual environment. The preprocessing, training, and evaluation code is maintained in the project repository.

A.5 ONNX export and edge deployment

The sizes in Table 15 are serialised scikit-learn and PyTorch artefacts. For deployment on resource-constrained edge hardware, every model is also exported to the ONNX format and run under the ONNX Runtime, free of scikit-learn and PyTorch. The conversion is faithful. The full ExtraTrees ONNX matches the scikit-learn forest to a maximum class-probability difference of 6.4×10^{-7} and reproduces 99.90% of the top-1 predictions over the 2,099-window test split (two near-tie flips). All seventeen families reproduce their predictions exactly on a 500-window live audit. The ONNX model is therefore the same model, not an approximation. No quantisation is applied, and all exports are float32 at opset 17.

ONNX conversion itself loses nothing, so the only accuracy cost of edge deployment comes from trimming the forest to fit a smaller memory budget. Table 29 measures that cost under the same chronological within-class $K = 5$ protocol used throughout.

Table 30 reports a deployment of the ONNX models on a Raspberry Pi 4 (1.85 GB RAM, ONNX Runtime 1.26). The full 300-tree forest needs about 3 GB resident and does not

Table 28: **ONNX against native model size**, in decimal MB. Native sizes match Table 15. The 100- and 50-tree rows are trimmed ExtraTrees sub-forests for edge deployment. Classical models shrink under ONNX; the small neural models are essentially unchanged.

Model	Native (MB)	ONNX (MB)
ExtraTrees (300 trees)	999.5	761.0
ExtraTrees (100 trees)	–	242.1
ExtraTrees (50 trees)	–	121.0
XGBoost	39.4	11.5
Transformer	0.45	0.62
LSTM	0.89	0.89

Table 29: **Accuracy and error of the ExtraTrees edge variants**, chronological within-class $K = 5$, mean $\pm$ std across folds. The 300-tree forest is the thesis result; trimming to 100 or 50 trees shrinks the ONNX file at a small accuracy cost. The full ONNX preserves this 84.3%, since it is the exact model.

Variant	Trees	ONNX (MB)	Grid acc	2D MAE (m)	Δacc
ExtraTrees (300 trees)	300	761	0.843 ± 0.065	0.434 ± 0.191	–
ExtraTrees (100 trees)	100	242	0.836 ± 0.064	0.457 ± 0.174	-0.007
ExtraTrees (50 trees)	50	121	0.821 ± 0.066	0.510 ± 0.187	-0.022

fit. The trimmed 50-tree forest (121 MB) loads in about 79 s and then predicts in 70 microseconds, while the Transformer (0.6 MB) loads in 0.1 s and predicts in about 1 ms. The device runs all four models concurrently at a 1.37 GB peak and 7.2 ms per cycle. Even the near-gigabyte ExtraTrees forest is therefore deployable at the edge through the ONNX Runtime, with a small, measured accuracy cost and real-time latency.

Table 30: **ONNX Runtime deployment on a Raspberry Pi 4** (1.85 GB RAM, ONNX Runtime 1.26). Size is the ONNX file; cold-start load is the one-time startup cost and is paid only once at launch, whereas the recurring per-prediction cost is the latency column; steady RAM is the resident set after load; latency is single-window inference, except the final row which is one full cycle over all four models. †The concurrent row is a warm load, the ExtraTrees forest already being cached, so it is far below its 79 s cold start. Single device, single session.

Model (ONNX)	Size (MB)	Cold-start load (s)	Steady RAM (MB)	Latency (ms)
ExtraTrees (50 trees)	121	79	520	0.07
XGBoost	11.5	1.0	198	3.6
Transformer	0.60	0.1	49	1.0
LSTM	0.87	0.0	51	1.9
All four concurrent	–	$11^{\dagger}$	648	7.2

Appendix B

Extended results

B.1 Per-fold cross-validation results

The main text reports each metric as a fold mean with its standard deviation. Tables 31 to 34 give the underlying per-fold values for the chronological within-class protocol, at $K = 5$ and $K = 10$. The fold means reproduce the headline numbers of Chapter 8.

Table 31: **Per-window 2D mean absolute error (m) per fold, chronological within-class $K = 5$.** Lower is better. Final column is mean ± standard deviation across folds.

Model	F1	F2	F3	F4	F5	mean ± std
ExtraTrees	0.659	0.262	0.318	0.309	0.625	0.434 ± 0.191
Random forest	0.710	0.278	0.294	0.365	0.615	0.452 ± 0.198
CatBoost	0.663	0.271	0.327	0.378	0.621	0.452 ± 0.178
LightGBM	0.746	0.308	0.368	0.370	0.625	0.483 ± 0.191
XGBoost	0.764	0.309	0.327	0.342	0.643	0.477 ± 0.211
MLP	0.988	0.624	0.720	0.728	1.003	0.813 ± 0.172
SVM	1.192	0.835	0.863	0.816	1.202	0.982 ± 0.197
k-NN	2.109	1.870	1.864	1.961	2.190	1.999 ± 0.146
Transformer	0.292	0.131	0.212	0.360	0.788	0.356 ± 0.256
LSTM	0.409	0.212	0.292	0.427	0.821	0.432 ± 0.234
TCN	0.476	0.329	0.336	0.427	0.894	0.493 ± 0.233
CNN-LSTM	0.774	0.358	0.417	0.538	1.100	0.637 ± 0.304
1D CNN	0.891	0.410	0.355	0.403	0.836	0.579 ± 0.261
FusionNet	6.203	6.199	6.202	6.201	6.192	6.199 ± 0.004

Table 32: **Per-window 2D mean absolute error (m) per fold, chronological within-class $K = 10$.** Final column is mean ± standard deviation across folds.

Model	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	mean ± std
ExtraTrees	0.812	0.265	0.246	0.210	0.248	0.312	0.279	0.276	0.311	0.914	0.387 ± 0.254
LightGBM	0.957	0.328	0.341	0.197	0.196	0.339	0.329	0.313	0.337	0.903	0.424 ± 0.273
XGBoost	1.015	0.327	0.287	0.208	0.222	0.320	0.312	0.312	0.305	0.955	0.426 ± 0.297
Transformer	0.390	0.112	0.171	0.111	0.111	0.230	0.262	0.253	0.341	1.140	0.312 ± 0.307
LSTM	0.488	0.275	0.208	0.163	0.235	0.283	0.262	0.317	0.449	1.150	0.383 ± 0.288
TCN	0.593	0.247	0.286	0.202	0.140	0.356	0.495	0.346	0.466	1.208	0.434 ± 0.305
CNN-LSTM	0.654	0.478	0.395	0.298	0.256	0.449	0.382	0.387	0.626	1.280	0.520 ± 0.295
1D CNN	0.671	0.267	0.412	0.304	0.171	0.393	0.587	0.359	0.481	1.242	0.488 ± 0.303
FusionNet	6.210	6.202	6.205	6.198	6.190	6.196	6.196	6.198	6.203	6.194	6.199 ± 0.006
Anchor-attention	5.758	6.202	6.205	6.198	6.190	6.196	6.196	6.198	6.203	6.194	6.154 ± 0.139

Table 33: **Grid accuracy per fold, chronological within-class $K = 5$.** The final column reports the mean ± standard deviation across folds.

Model	F1	F2	F3	F4	F5	mean ± std
ExtraTrees	0.749	0.896	0.887	0.882	0.800	0.843 ± 0.065
Random forest	0.739	0.892	0.892	0.873	0.796	0.838 ± 0.068
CatBoost	0.742	0.895	0.881	0.857	0.801	0.835 ± 0.063
LightGBM	0.730	0.894	0.877	0.880	0.800	0.836 ± 0.070
XGBoost	0.730	0.894	0.882	0.882	0.797	0.837 ± 0.071
MLP	0.625	0.754	0.714	0.729	0.655	0.695 ± 0.054
SVM	0.593	0.708	0.692	0.696	0.623	0.663 ± 0.051
k-NN	0.351	0.416	0.418	0.397	0.344	0.385 ± 0.035
Transformer	0.871	0.937	0.919	0.886	0.802	0.883 ± 0.052
LSTM	0.824	0.905	0.889	0.846	0.792	0.851 ± 0.046
TCN	0.803	0.856	0.883	0.855	0.763	0.832 ± 0.049
CNN-LSTM	0.776	0.873	0.875	0.842	0.739	0.821 ± 0.061
1D CNN	0.692	0.856	0.865	0.859	0.790	0.812 ± 0.074
FusionNet	0.058	0.059	0.059	0.059	0.059	0.059 ± 0.000

Table 34: **Grid accuracy per fold, chronological within-class $K = 10$.** The final column reports the mean ± standard deviation across folds.

Model	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	mean ± std
ExtraTrees	0.691	0.894	0.914	0.922	0.910	0.892	0.895	0.896	0.891	0.713	0.862 ± 0.085
LightGBM	0.661	0.890	0.899	0.922	0.928	0.891	0.888	0.897	0.892	0.722	0.859 ± 0.091
XGBoost	0.659	0.887	0.915	0.914	0.910	0.899	0.904	0.897	0.895	0.707	0.859 ± 0.094
Transformer	0.838	0.954	0.933	0.952	0.943	0.938	0.925	0.927	0.910	0.728	0.905 ± 0.070
LSTM	0.795	0.891	0.907	0.924	0.898	0.924	0.924	0.913	0.870	0.728	0.877 ± 0.065
TCN	0.738	0.886	0.898	0.921	0.928	0.885	0.825	0.882	0.873	0.704	0.854 ± 0.076
CNN-LSTM	0.766	0.844	0.881	0.896	0.917	0.885	0.890	0.894	0.846	0.731	0.855 ± 0.061
1D CNN	0.747	0.879	0.840	0.872	0.917	0.877	0.823	0.880	0.870	0.704	0.841 ± 0.066
FusionNet	0.058	0.058	0.058	0.058	0.059	0.059	0.059	0.060	0.059	0.060	0.059 ± 0.001
Anchor-attention	0.012	0.058	0.058	0.058	0.059	0.059	0.059	0.060	0.059	0.060	0.054 ± 0.015

B.2 Random-split protocol audit

Table 35 expands the protocol audit of Section 8.5. It compares the eight classical families under the chronological within-class $K = 5$ protocol against a stratified random $K = 5$ split of the same windows, all else identical. The two tabular neural models show the same direction (Section 8.5). The sequence models are excluded from the table because a random split over their overlapping windows is an optimistic ceiling, not a comparable measurement.

Table 35: **Protocol audit: chronological versus random $K = 5$.** The same windows, models, and seed 42 are used; only the split changes. The random split reports higher accuracy for every family (Δacc) and lower error, because windows close in time can appear on both sides of the split. The random audit was run only at $K = 5$.

Model	Acc chrono	Acc random	Δacc	2D chrono	2D random
ExtraTrees	84.3 ± 6.5	87.1 ± 0.7	+2.8	0.434 ± 0.191	0.371 ± 0.033
Random forest	83.8 ± 6.8	86.0 ± 0.9	+2.2	0.452 ± 0.198	0.406 ± 0.030
CatBoost	83.5 ± 6.3	85.6 ± 0.9	+2.1	0.452 ± 0.178	0.410 ± 0.031
LightGBM	83.6 ± 7.0	86.5 ± 1.2	+2.9	0.483 ± 0.191	0.409 ± 0.046
XGBoost	83.7 ± 7.1	86.0 ± 0.8	+2.4	0.477 ± 0.211	0.411 ± 0.033
MLP	69.5 ± 5.4	71.0 ± 1.2	+1.5	0.813 ± 0.172	0.790 ± 0.028
SVM	66.3 ± 5.1	67.9 ± 1.2	+1.6	0.982 ± 0.197	0.951 ± 0.041
k-NN	38.5 ± 3.5	39.6 ± 1.3	+1.1	1.999 ± 0.146	1.969 ± 0.089

B.3 Training-set size

Table 36 reports how ExtraTrees accuracy changes with the training fraction under the chronological within-class $K = 5$ protocol.

Table 36: **ExtraTrees performance by training-set size**, chronological within-class $K = 5$, reported as mean ± standard deviation across folds. Accuracy rises steeply up to about 20% of the data and then flattens, suggesting that CS3D is not strongly data-limited at full size.

Fraction	n_{train}	Grid acc	2D MAE (m)
0.05	273	58.0 ± 4.7	1.213 ± 0.146
0.10	558	67.5 ± 5.9	0.914 ± 0.175
0.20	1118	76.0 ± 5.7	0.672 ± 0.187
0.30	1682	78.3 ± 6.2	0.592 ± 0.182
0.40	2240	80.0 ± 6.9	0.556 ± 0.193
0.50	2802	81.7 ± 5.8	0.509 ± 0.160
0.60	3365	81.9 ± 6.3	0.486 ± 0.176
0.75	4200	83.4 ± 6.7	0.451 ± 0.181
0.90	5046	84.0 ± 6.1	0.446 ± 0.183
1.00	5605	84.3 ± 6.5	0.434 ± 0.191

B.4 Per-room results

Table 37 disaggregates the ExtraTrees grid accuracy and 2D mean absolute error by room, at both values of K, under the chronological within-class protocol. The Meeting Room, which contains only two of the 42 cells, is the clear low-performing outlier. This is discussed in Section 8.4.

Table 37: **ExtraTrees results by room.** Grid accuracy and 2D mean absolute error by room at both values of K, as mean ± sample standard deviation over folds. The Meeting Room contains two of the 42 cells and contributed the fewest raw reports.

	$K = 5$		$K = 10$	
Room	Accuracy	2D MAE (m)	Accuracy	2D MAE (m)
Kitchen	0.901 ± 0.059	0.318 ± 0.182	0.901 ± 0.093	0.321 ± 0.251
Office	0.868 ± 0.065	0.389 ± 0.175	0.879 ± 0.087	0.348 ± 0.252
Lab	0.831 ± 0.107	0.444 ± 0.387	0.855 ± 0.154	0.393 ± 0.549
Meeting Room	0.445 ± 0.101	1.245 ± 0.291	0.595 ± 0.117	0.944 ± 0.443

Appendix C

Per-cell error: offline versus live

Figures 11 to 14 show the per-cell 2D mean absolute error of ExtraTrees under argmax decoding as spatial maps over the 7-row by 6-column grid, for the offline held-out fold and the cross-session live replay, at both heights. Each panel is shown at full width, one per page, to keep the per-cell values legible in print. Cells marked with an asterisk were not visited in the relevant session. The Office region, columns 4 to 6, carries most of the live-side error, whereas the Lab region, rows D to G and columns 1 to 3, shows the smallest offline-to-live gap. The live panels are drawn from a single replay session, so the spatial pattern indicates where covariate shift is most visible rather than supporting a multi-session claim.

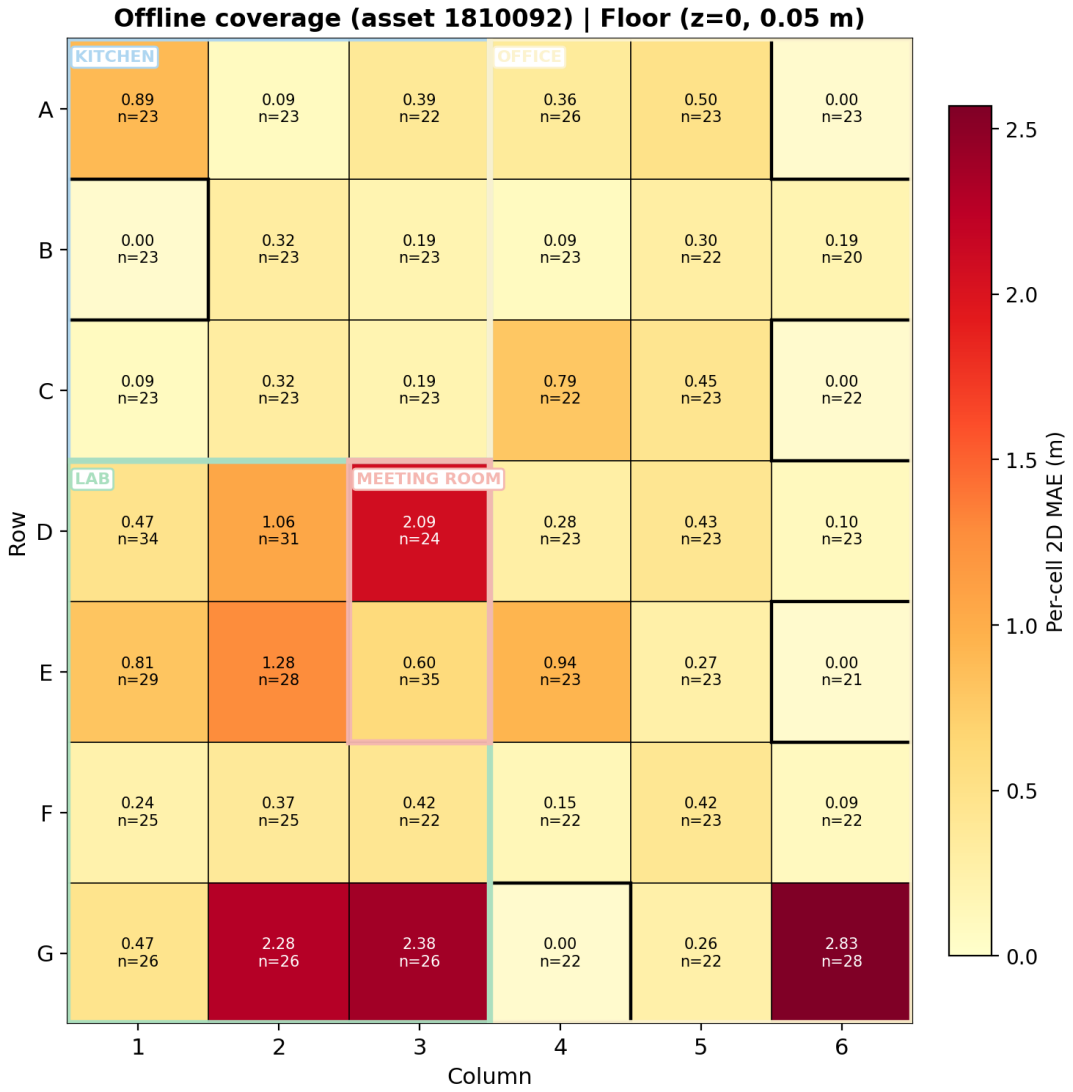


Figure 11: Per-cell 2D mean Euclidean error of ExtraTrees under argmax decoding. Offline single-session held-out fold, floor height ($z = 0,\ 0.05\,\mathrm{m}$). Each cell shows its MAE in metres and sample count n; asterisks mark cells absent from the held-out fold. Asset 1810092.

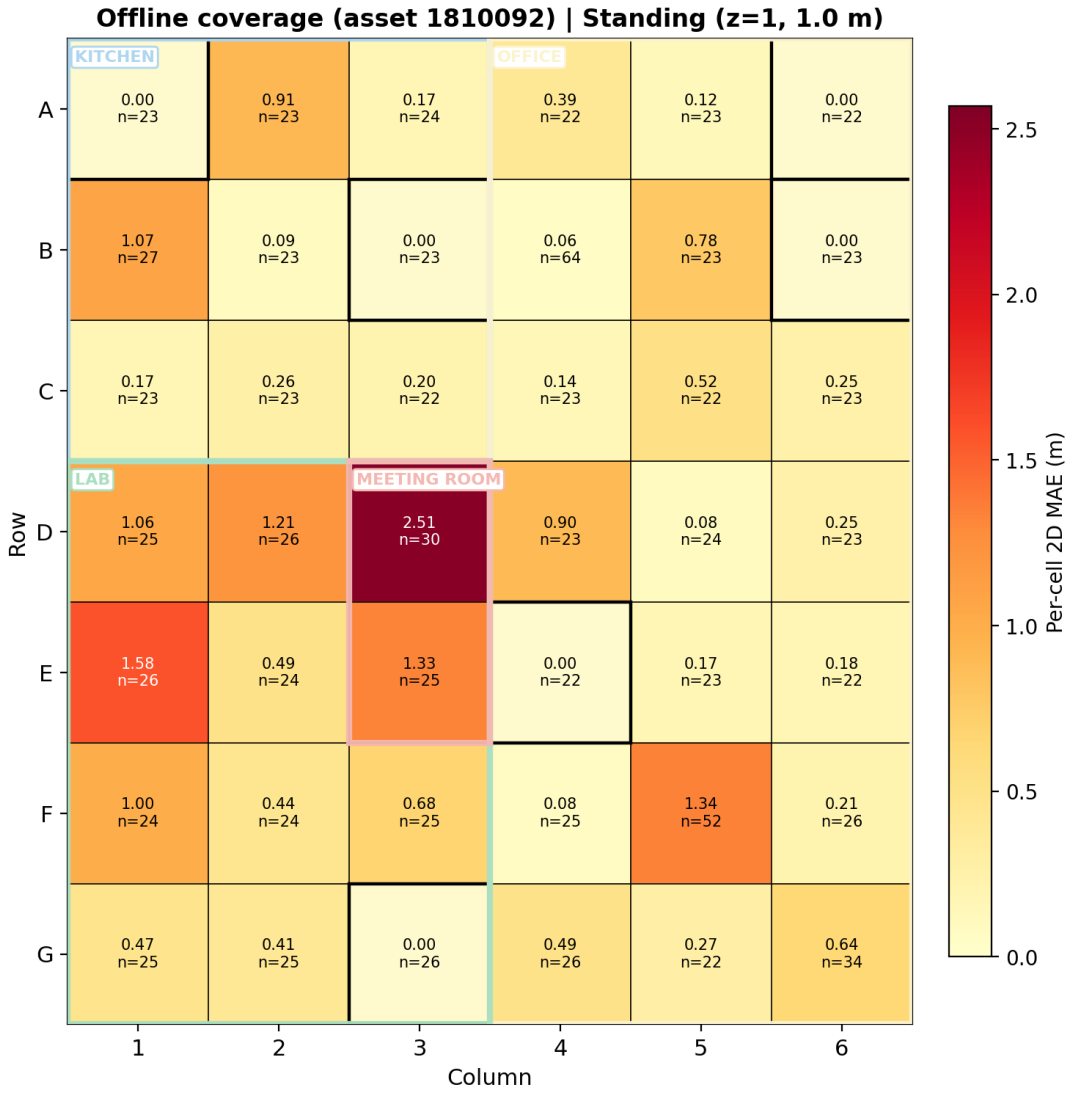


Figure 12: Per-cell 2D mean Euclidean error of ExtraTrees under argmax decoding. Offline single-session held-out fold, standing height ($z = 1$, 1.0 m). Asset 1810092.

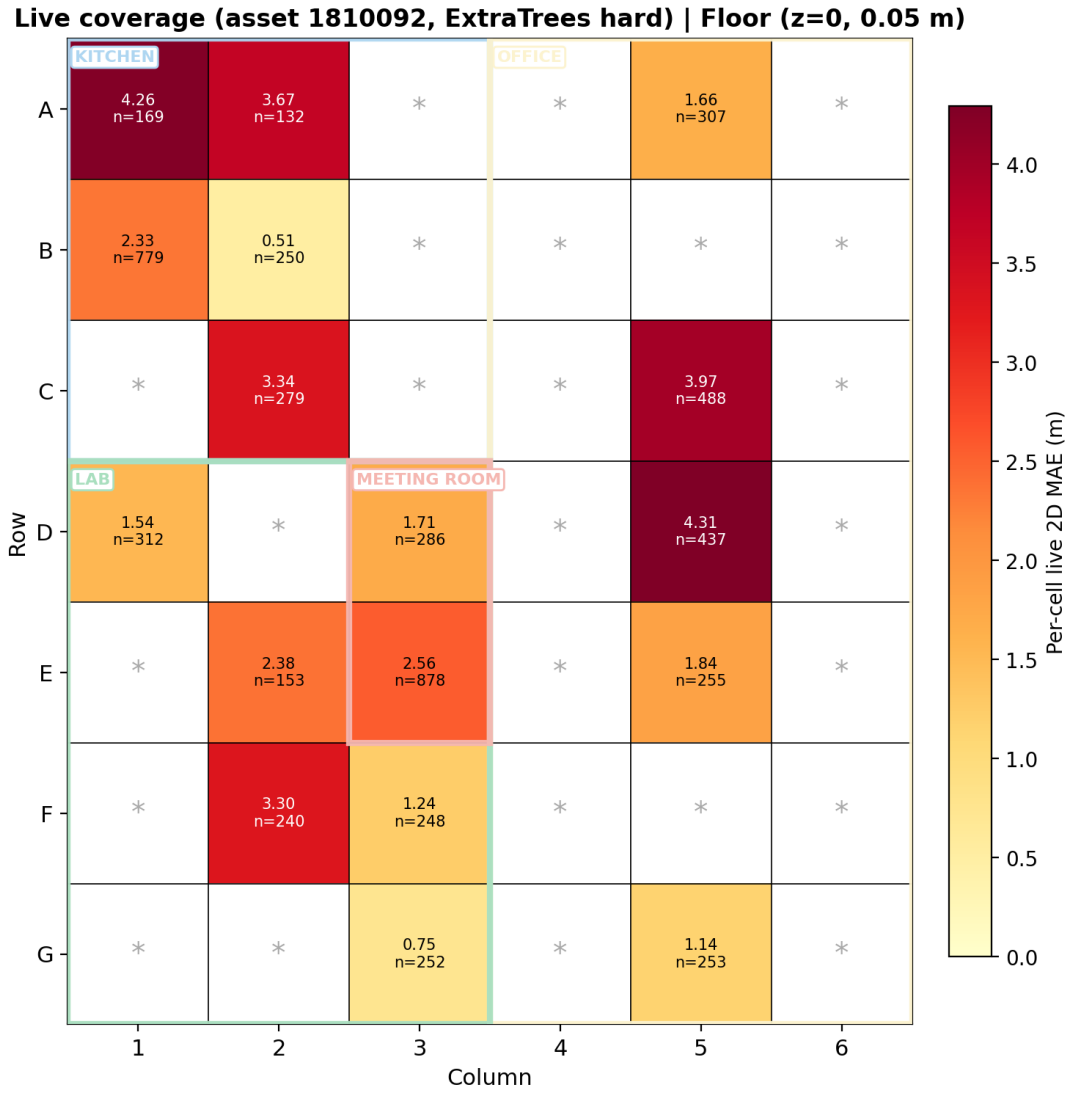


Figure 13: Per-cell 2D mean Euclidean error of ExtraTrees under argmax decoding. Cross-session live replay recorded on a different day, floor height ($z = 0$). The errors are larger than in the offline fold, consistent with covariate shift between the survey and the live session. This is one session; multi-session validation remains future work. Asset 1810092.

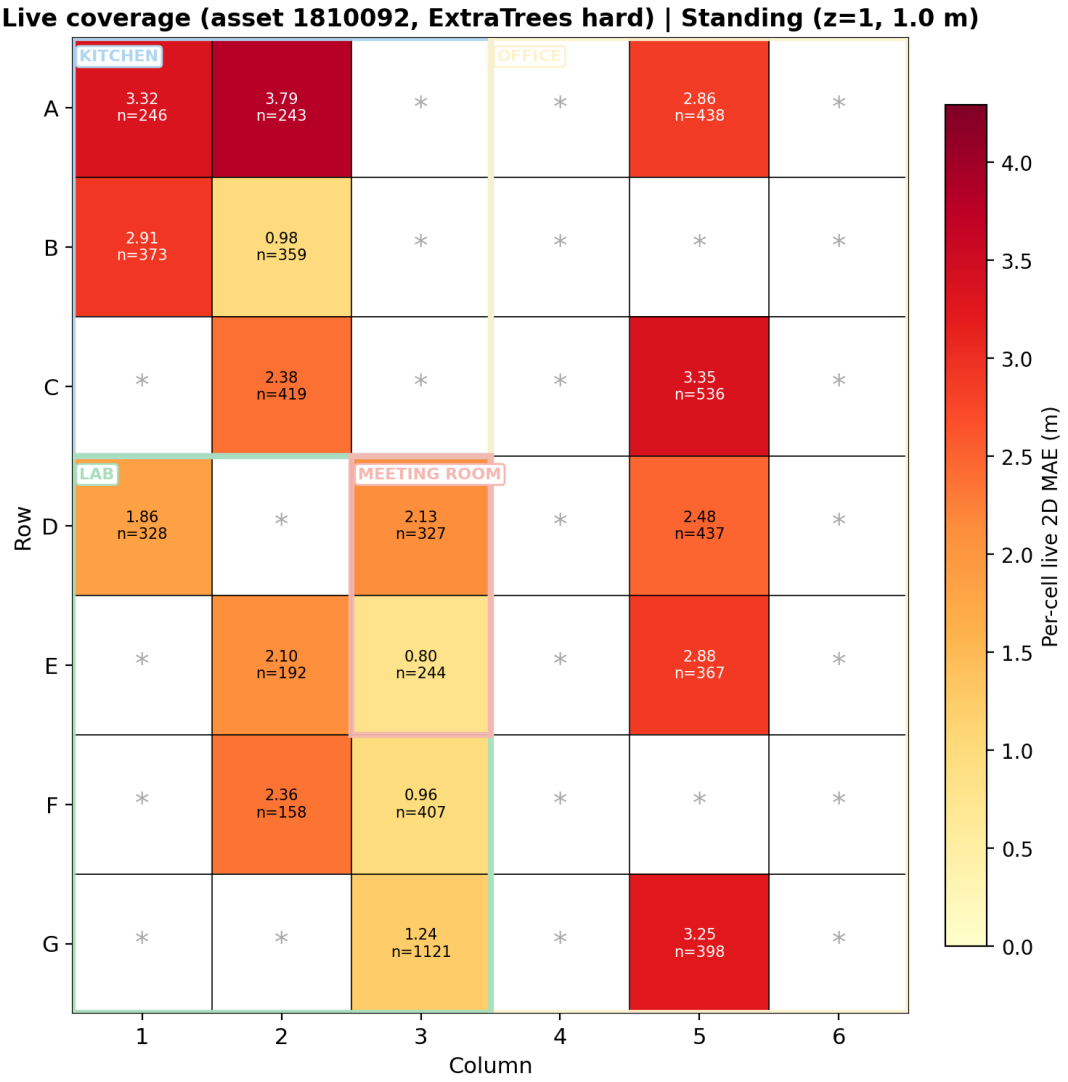


Figure 14: Per-cell 2D mean Euclidean error of ExtraTrees under argmax decoding. Cross-session live replay recorded on a different day, standing height ($z = 1$). Asset 1810092.

Appendix D

Post-processing filters

A per-window classifier treats each prediction independently, but a tagged asset moves continuously, so consecutive predictions are correlated. A post-processing stage can use this temporal structure to smooth the predicted trajectory. This appendix implements and evaluates six such filters on top of the ExtraTrees prediction stream. All six are strictly causal and use past samples only, making them valid for live streaming. The main comparison in Chapter 8 stops at the per-window prediction. This appendix reports the filters separately because, under the live chronological protocol, none yields a statistically distinguishable improvement.

Table 38: The six causal post-processing filters. Filters acting on coordinates smooth the (x, y, z) output; filters acting on the probability vector operate on the 84-class probability vector before centroid mapping.

Filter	Input	Operation
Kalman (3D CV)	(x, y, z)	constant-velocity state-space smoother
Particle (3D CV)	(x, y, z)	Monte Carlo belief; resample when ESS $< N/2$
HMM forward	84-class proba	distance-weighted grid HMM with probability emission
EMA-softmax	84-class proba	exponential moving average of class probabilities
One-Euro	(x, y, z)	speed-adaptive low-pass filter
Fixed-Lag Viterbi	84-class proba	k-step look-back most-likely sequence

Protocol. Each filter's hyperparameters were tuned using 5-fold contiguous time-split cross-validation, following single-session evaluation practice in indoor positioning [34]. Two evaluations are reported. The first tunes and scores the filters on the offline 84-class test split (2,099 windows, ExtraTrees base), which is the per-cell chronological hold-out, namely the chronological tail of each cell's survey block. The second uses a single live replay session split chronologically. Hyperparameters are tuned on the first 9,849 windows and frozen, then applied to the last 2,463 windows, which were not seen during tuning. The live evaluation reports 95% moving-block bootstrap confidence intervals for the mean 3D error.

Table 39: **Offline static-survey test** (ExtraTrees base, 2,099 windows, per-cell chronological split). The raw baseline is the unfiltered per-window prediction, decoded by argmax to its cell centroid (Equation 13), which is the decoder used for all main results; it gives 0.578 m in 3D (0.558 m in 2D), and this lies within one fold standard deviation of the 0.455 ± 0.195 m K-fold result of Table 13. Because the survey holds the tag stationary within each cell, the apparent filter gains are a static-survey artefact analysed below; the deployment-faithful comparison is the live moving-tag hold-out of Table 40.

Filter	3D MAE (m)	p75 (m)	height acc. (%)
raw	0.578	0.000	92.90
HMM	**0.274**	0.000	97.67
Fixed-Lag Viterbi	0.285	0.000	97.14
Particle	1.166	1.556	96.62
Kalman	1.205	1.639	97.05
One-Euro	1.239	1.648	96.76
EMA-softmax	1.325	1.729	94.04

Table 40: **Live chronological hold-out** (single replay session, 2,463 hold-out windows). Hyperparameters are tuned on the first 80% by time and then frozen. The raw hold-out MAE is 1.890 m, with a 95% bootstrap interval of $[1.498, 2.477]$. Every filter interval overlaps the raw interval, so no filter is statistically distinguishable from the unfiltered output in this session.

Filter	3D MAE (m)	95% bootstrap CI	Δ vs raw (m)
EMA-softmax	1.744	$[1.367, 2.302]$	-0.145
Kalman	1.791	$[1.403, 2.363]$	-0.099
HMM	1.791	$[1.337, 2.414]$	-0.099
Particle	1.805	$[1.422, 2.373]$	-0.084
Fixed-Lag Viterbi	1.814	$[1.390, 2.390]$	-0.076
One-Euro	1.824	$[1.432, 2.410]$	-0.066

Offline gains do not transfer to the live replay. The contrast between Tables 39 and 40 is the main result of this appendix. Offline, the HMM and Fixed-Lag Viterbi appear to reduce the error to 0.274 and 0.285 m, with a 75th-percentile error of zero, a reduction of about 0.30 m against the raw baseline of 0.578 m. Even this residual gain is an artefact of the static per-cell survey: when sorted by time, adjacent offline windows often correspond to the same surveyed cell, so a high-persistence HMM with $p_{\text{stay}} = 0.95$ tends to remain at that cell and is rarely penalised. In the live session, the tag moves between cells, the same high p_{stay} becomes disadvantageous, and the apparent advantage disappears. Every filter interval overlaps the raw baseline. A latency-versus-accuracy sweep of the Fixed-Lag Viterbi look-back, from zero to ten seconds, supports this interpretation, changing the live error by only 0.024 m across the whole range. The offline advantage therefore reflects the stationary survey geometry rather than a property that survives on a moving tag. The evaluation in this appendix rests on a single replay session, so generalisation across sessions and days is not measured and remains future work.