TinyML-Empowered Indoor Positioning with Light

A Study on the Impact of LED Aging and Failure

Joey Wenyi Li[1]

Supervisor(s): Qing Wang[1]**, Ran Zhu**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

Name of the student: Joey Wenyi Li
Final project course: CSE3000 Research Project
Thesis committee: Qing Wang, Ran Zhu, Ranga Rao Venkatesha Prasad

Abstract

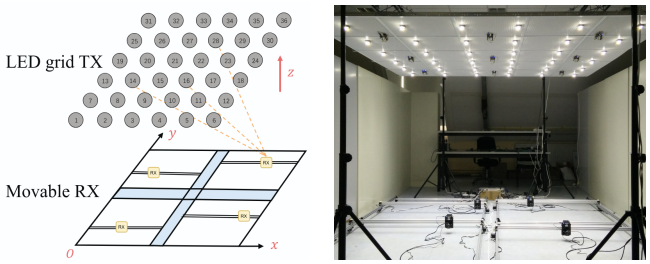
Visible light positioning (VLP) enables accurate indoor localization by leveraging a dense deployment of LEDs in future lighting infrastructure, but its widespread adoption is hindered by two key challenges: the need for densely sampled fingerprint datasets and performance degradation due to LED aging or failure. In this work, we propose a VLP framework that reduces reliance on dense fingerprinting and remains robust over time without requiring manual re-fingerprinting. Using a dataset acquired from the DenseVLC testbed, we evaluate preprocessing techniques that enhance positioning accuracy under noisy received signal strength (RSS) measurements. To address long-term reliability, we introduce a simulation framework that models LED degradation and sudden failures. Most importantly, we present an online learning approach that dynamically adapts the positioning model in response to environmental and infrastructure changes. In our simulations, this approach maintains the original level of accuracy despite aging effects. In some cases, it yields up to a 95% improvement when evaluated over longer timespans. Furthermore, our preprocessing contributions have led to a 30% improvement to baseline performance without aging. Our results demonstrate a path toward scalable, self-sustaining VLP systems suitable for real-world deployment.

1 Introduction

Accurate indoor positioning is a critical enabler for a wide range of modern applications, from smart homes and factories to augmented reality and robotics. While outdoor systems like Global Positioning System (GPS) and BeiDou (BDS) provide global coverage, they fail to deliver the precision and reliability required in indoor environments. Visible light positioning (VLP) systems have emerged as a viable alternative due to their high accuracy and low power consumption, while also being affordable.

VLP encompasses a range of techniques, with state-of-the-art methods including time-difference-of-arrival (TDOA) [4] and received signal strength (RSS) [10]. However, these techniques often rely on densely sampled fingerprint datasets, which are collections of signal measurements taken at many known locations throughout an area. Creating such datasets is labor-intensive and expensive, hindering large-scale deployment. Recent research such as the work by Zhu *et al.* [11] partially remedies this by using data preprocessing techniques to enable sparse fingerprint learning, but the reliance on signal measurements remains. Moreover, VLP systems can degrade over time due to LED aging or failures, necessitating regular recalibration to maintain performance.

To address these limitations, our research investigates how to reduce the dependency on dense data collection while improving the long-term robustness of VLP systems. Specifically, we explore data preprocessing techniques that can construct accurate and reliable fingerprints from noisy input data.



(a) Schematic (b) Testbed

Figure 1: DenseVLC system setup [1].

We also aim to simulate and mitigate the impact of degraded or malfunctioning LEDs on positioning accuracy, an essential step toward scalable, robust and long-lasting indoor localization solutions.

In this work, we make three primary contributions. First, we conduct a comparative evaluation of data preprocessing strategies aimed at improving positioning accuracy with noisy fingerprint datasets. Second, we introduce an online learning approach that dynamically adapts the positioning model in response to infrastructure degradation, reducing the need for manual recalibration and supporting robust, long-term deployment. Third, we develop a simulation framework to model the effects of LED aging and sudden failures, enabling systematic testing of system resilience.

The rest of the paper is structured as follows: Section 2 discusses related work and the dataset. Section 3 presents our data cleaning and augmentation pipeline, the aforementioned online learning approach and the LED aging simulation. Section 4 briefly describes the experimental setup, while Section 5 goes over the results and their analysis. Section 6 goes over TinyML deployment, where we demonstrate that our approach performs effectively in resource-constrained environments. Section 7 addresses ethical considerations and reproducibility, while Sections 8 and 9 go over the discussion and conclusion respectively.

2 Background

There have been significant advancements in the field of visible light positioning (VLP) in recent years. Several comprehensive surveys, such as those by Zhu *et al.* [12], Zhuang *et al.* [13], Do and Yoo [3], and Luo *et al.* [9], have documented these developments. Our research builds on the work of Zhu *et al.* [11], who used the DenseVLC [1] testbed to collect a high-resolution RSS fingerprint dataset for position prediction. We extend their approach by introducing several enhancements to the RSS-based positioning method and evaluating its performance under more realistic conditions. The following subsections provide a detailed overview of the most relevant prior work.

2.1 DenseVLC Dataset

The DenseVLC [1] dataset is a high-resolution indoor positioning dataset collected in a controlled environment using the DenseVLC testbed. A mobile receiver samples received signal strength (RSS) values from 36 ceiling-mounted LED

transmitters arranged in a grid layout. The covered area measures 3 m $\times$ 3 m, with LEDs positioned on a height adjustable ceiling and spaced at 0.5 m intervals. Figure 1 presents both a schematic and a photograph of the experimental setup.

Each data point in the dataset consists of RSS values from all 36 LED transmitters, paired with ground-truth position information. The data was collected using four mobile receivers, each covering an area of approximately 1.2 m by 1.2 m. Measurements were taken at 1 cm intervals along both the x and y axes, resulting in 121×121 points sampled per receiver. For each position, three measurements were recorded at two different heights relative to the LEDs: 176 cm and 192 cm. In total, the dataset comprises $121 \cdot 121 \cdot 4 \cdot 3 \cdot 2 = 351{,}384$ samples.

In our work, we utilize this dataset to simulate sparse data conditions by subsampling the original grid and evaluate how well different preprocessing and learning strategies can recover accurate position estimates. Additionally, we use the complete dataset as a baseline for comparison and as a reference for modeling LED degradation scenarios.

2.2 Data Cleaning and Augmentation

The work of Zhu *et al.* [11] represents a significant step forward in visible light positioning (VLP) by demonstrating that accurate indoor localization is achievable even with sparse RSS fingerprint data. Their approach differs from traditional methods that rely on densely sampled fingerprints, proposing instead to use data cleaning and augmentation in order to augment sparsely sampled data into a dense dataset.

Using the DenseVLC dataset as a foundation, Zhu *et al.* first start by cleaning this data. This is done by calculating the continuity at a local point for every sample using Equation 1.

$$\Delta I_{x,y} = \left\| I_{x,y} - \frac{1}{|\mathbf{I_S}|} \sum_{I \in \mathbf{I_S}} I \right\| \tag{1}$$

Here, $I_{x,y}$ denotes the RSS at location (x, y), and $\mathbf{I_S}$ is the set of neighboring RSS values. Samples with excessively high continuity scores, indicating inconsistency, are removed.

These removed values are then reconstructed using nearby clean points and the Lambertian diffuse model, a physical model that approximates how light intensity varies with angle and distance. The model is given by the Equation 2.

$$I^r = \begin{cases} I^t A(d) \cos^m(\phi) \cos(\psi), & 0 \leq \psi \leq \Psi_c \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The final reconstructed RSS value is obtained by averaging the estimates computed from multiple nearby points.

Next, the researchers proposed a method to increase the density of a dataset. This augmentation process works analogously to the cleaning process, where we interpolate missing values using the Lambertian model, as described in Equation 2. This can, for example, interpolate values from a 16 cm interval dataset to a 1 cm interval.

We adopt their method as a baseline in our research, both to evaluate our proposed preprocessing strategies and to benchmark performance under simulated LED degradation. Furthermore, we use their augmentation in our online learning approach in order to deploy on TinyML hardware.

3 Methodology

In the previous section, we have seen that prior work has had promising results in the field of VLP. However, there has not been much work in regard to the effects of LED degradation during real-world deployment. In the following subsections, we will first discuss some changes made to the data preprocessing pipeline. Second, we design a resource-efficient neural network that reduces model size while improving positioning accuracy. Third, we propose an online learning framework that enables the model to adapt to LED aging and failures. Finally, we develop a simulation to replicate realistic LED degradation and flickering for long-term testing. Together, these advances create a robust, adaptive VLP system fit for real-world, long-term use on constrained hardware.

3.1 Improved LED Position

Previous pipelines generally assume precise knowledge of LED positions. However, these positions may be subject to errors or uncertainties in real-world settings. By leveraging the radial decay of LED illumination, we fit circles to a band of RSS data values within a defined threshold range to infer light source locations directly from raw signal strength measurements. These estimates can be made from raw data, improving interpolation in later stages of the pipeline.

We propose the usage of a simple RANSAC algorithm [5], as described in Algorithm 1, due to its robust nature against outliers.

Algorithm 1: Basic RANSAC Algorithm for Circle Fitting using Three-Point Sampling

Input: Array P of 2D points, threshold δ, number of trials m
Output: Best fitting circle center (x_0, y_0) and radius r

$(x_{best}, y_{best}) \leftarrow (0, 0)$;
$r_{best}, s_{best} \leftarrow 0$;
for $i \leftarrow 1$ **to** m **do**
  Select 3 random, non-collinear points from P: (p_1, p_2, p_3);
  Compute circle (x, y, r) passing through (p_1, p_2, p_3);
  $s \leftarrow 0$;
  foreach $p \in P$ **do**
    if $|\|p - (x_0, y_0)\| - r| < \delta$ **then**
      $s \leftarrow s + 1$;
  if $s > s_{best}$ **then**
    $(x_{best}, y_{best}) \leftarrow (x, y)$;
    $r_{best}, s_{best} \leftarrow r, s$;
return $(x_{best}, y_{best}, r_{best})$;

An example of how this algorithm is used to calculate LED positions compared to classical statistical methods, such as the Kåsa method [8], is illustrated in Figure 2. The RANSAC-based approach demonstrates superior robustness in the presence of noise, enabling accurate LED position estimation even prior to data cleaning.

3.2 Improved Data Cleaning

To enhance data quality and improve model performance under noisy fingerprinting conditions, we propose a modi-

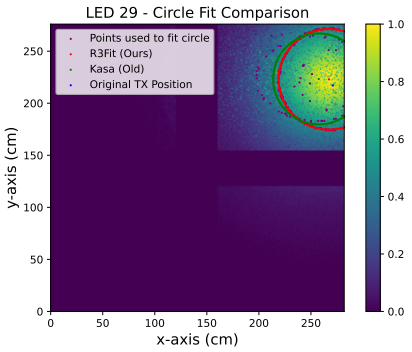


Figure 2: Comparison of circle fitting results for LED 29 using the RANSAC-based method and the Kåsa method [8].

fied scoring function that accounts for small local illumination variations. We then interpolate missing values using a Lambertian-based model.

Given the illumination intensity $I_{x,y}$ at position (x, y) and a neighborhood I_N centered around it, our baseline scoring metric is defined as:

$$S_{x,y} = \left\| I_{x,y} - \frac{1}{|I_N|} \sum_{I \in I_N} I \right\| \tag{3}$$

To better normalize this score with respect to the local context, we propose an improved variant:

$$S'_{x,y} = \frac{I_{x,y} - \frac{1}{|I_N|} \sum_{I \in I_N} I}{\left[\frac{1}{|I_N|} \sum_{I \in I_N} I \right]^{\nu}} \tag{4}$$

Here, ν is a tunable bias hyperparameter that adjusts the influence of the surrounding illumination. This formulation improves the ability to differentiate signal anomalies whose absolute difference is small.

Additionally, we apply inverse distance weighted (IDW) interpolation to further smooth and reconstruct signal patterns in low-density areas, replacing the baseline approach which relies on simple mean-based interpolation.

Figure 3 illustrates the visual impact of our cleaning methods on LEDs 18 and 7, demonstrating the progression from raw data to the cleaned versions using both the old and improved approaches. For LEDs with relatively low noise, such as LED 18, the differences are minimal. However, in cases with a low signal-to-noise ratio like LED 7, our improved methods clearly make a significant difference.

3.3 Improved Data Augmentation

Our data augmentation strategy follows the approach of Zhu *et al.* [11], with a single modification: we use the LED positions estimated in Section 3.1 instead of assuming known ground-truth positions.

3.4 Improved Model

To reduce computational overhead and enable efficient on-device inference, we design a lightweight neural network architecture inspired by the building blocks of MobileNetV3 [7]. The model replaces the baseline MLP with a residual MLP architecture composed of two bottleneck blocks. Each bottleneck consists of a projection layer that compresses the input by approximately 90%, followed by a non-linear activation (H-Swish), and an expansion layer that restores the original dimension. This is followed by a residual connection adding the original input, and a final H-Swish activation. The network takes a 36-dimensional input, which is first normalized to have unit L2 norm, i.e., $\|\mathbf{x}\|_2 = 1$. The normalized input is then projected into a 256-dimensional latent space, followed by a ReLU activation and the two bottleneck blocks. A final linear layer maps the output to a 2D coordinate prediction. This design significantly reduces parameter count while maintaining accuracy, making it suitable for real-time deployment on constrained devices such as the Raspberry Pi Pico.

3.5 Online Degradation Scalar Learning

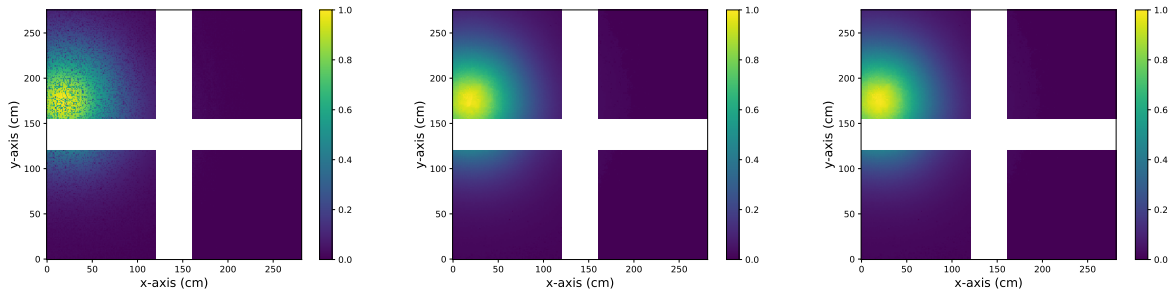
In order to keep our solution robust against the effects of LED degradation, we propose an online learning framework. The modular design of this framework enables easy integration with any existing VLP model that uses RSS, making implementation straightforward and flexible. Algorithm 2 shows the pseudo algorithm, explaining the basic details.

Algorithm 2: Online Scalar Adaptation for LED Degradation

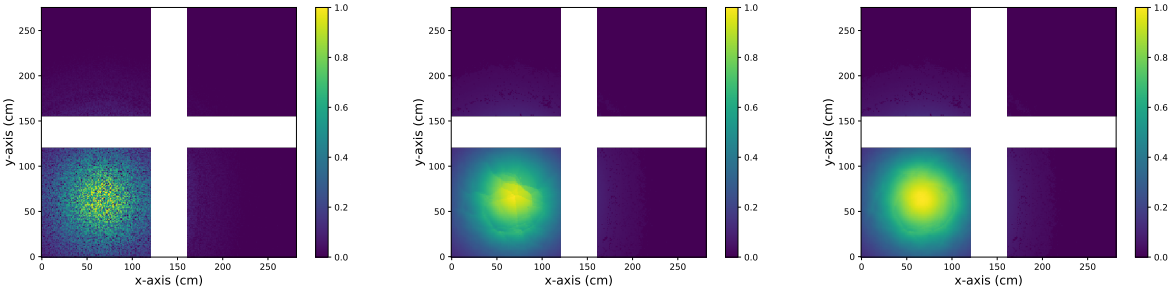
Input: Incoming RSS sample $X_t \in \mathbb{R}^{36}$, prediction model $\mathcal{M}$, window size n
Output: Updated scaling vector $\boldsymbol{\alpha} \in \mathbb{R}^{36}$

Initialize scaling vector $\boldsymbol{\alpha} \leftarrow \mathbf{1}_{36}$;
Initialize prediction buffer $B \leftarrow [\,]$;
Initialize reference buffer $R \leftarrow [\,]$;
while *new RSS sample* X_t *arrives* **do**
  Scale input: $X'_t \leftarrow \boldsymbol{\alpha} \odot X_t$;
  Predict position: $\hat{Y}_t \leftarrow \mathcal{M}(X'_t)$;
  Append $(X_t, \hat{Y}_t)$ to B;
  if *length of* $B \geq n$ **then**
    Obtain reference positions $Y_{t-n+1:t}$ for $X_{t-n+1:t}$;
    for $i = 1$ **to** 36 **do**
      Fit a line: $Y \approx a'_i \cdot \hat{Y}_i$;
      Update scalar: $\alpha_i \leftarrow \alpha_i \cdot a'_i$;
    Clear buffers $B \leftarrow [\,]$, $R \leftarrow [\,]$;

The algorithm begins by initializing a scaling vector that contains one scalar value per LED transmitter. Each incoming RSS sample is first scaled using this vector and then passed through the base positioning model to generate a predicted location. These predictions, along with the corresponding raw RSS inputs, are temporarily stored in a buffer. Once a predefined number of samples n have been collected, the algorithm retrieves the corresponding ground-truth positions and fits a separate linear model for each of the 36 LED channels. An example fit for this can be seen in Figure 4.



(a) Raw LED 18 (b) Baseline clean LED 18 (c) Improved clean LED 18



(d) Raw LED 7 (e) Baseline clean LED 7 (f) Improved clean LED 7

Figure 3: Comparison of data cleaning results for two example LEDs (LED 18 and LED 7). Subfigures (a) and (d) show the raw input data, (b) and (e) show the results of the baseline cleaning method as described by Zhu *et al.* [11], and (c) and (f) show the results using the proposed improved cleaning method.

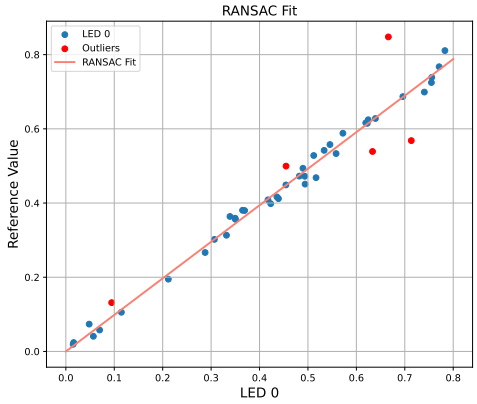


Figure 4: Example line fit for LED 0

Each linear model captures the coefficient between the raw RSS values and the expected signal under normal conditions. This is exactly the inverse of the factor by which the LEDs are degraded. The slope of each fitted line is then used to update the associated scalar in the scaling vector. This process enables the system to continuously adapt to changes in LED output, such as gradual dimming or sudden shifts, without requiring manual recalibration or retraining of the base model.

The choice to use a RANSAC line-fitting algorithm has been made based on experimental results. When comparing RANSAC to, for example, MSE line-fitting we can see RANSAC is more stable and less prone to outliers.

3.6 LED Aging Simulation

To evaluate the effectiveness of our online adaptation method in realistic conditions, we simulate LED aging and degradation effects directly on the input data. Since our dataset lacks long-term data points capturing natural LED wear, we introduce a synthetic degradation layer that mimics how LEDs lose brightness over time.

Our simulation operates by applying per-LED degradation factors to the raw RSS values in the dataset. For each LED i, we first define a degradation constant $k_i \in (0, \infty)$. The degradation at time t is computed using the exponential decay model shown in Equation 5, as introduced in the TM-21 standard [6].

$$L_i(t) = e^{-k_i t} \tag{5}$$

At any given time t, we scale the original RSS value from LED i, denoted as $X_i^{\text{original}} \in \mathbb{R}^{36}$, by $L_i(t)$. This gives us the simulated degraded signal.

$$X_i^{\text{degraded}}(t) = L_i(t) \cdot X_i^{\text{original}} \tag{6}$$

In order to further promote robustness, the simulation also introduces two additional noise components: random flickering and additive Gaussian noise. This means that the degraded RSS function can be rewritten as a random variable.

$$X_i^{\text{degraded}}(t) = L_i(t) \cdot \xi_i(t) \cdot X_i^{\text{original}} + \varepsilon_i(t) \tag{7}$$

where $\xi_i(t) \sim \text{Bernoulli}(p)$ models random LED flickering with flicker probability $q = 1 - p$, and $\varepsilon_i(t) \sim \mathcal{N}(0, \sigma^2)$ represents additive Gaussian noise with standard deviation σ.

At each timestep t, we sample m points. These points are then degraded using Equation 7, forming the set of points σ_t. This set is then fed into the model to simulate the continuous data flow typical of a regularly used VLP system. After processing the sample set, we test the accuracy of the model using a test set. This approach allows us to realistically simulate and measure the model's performance over any future time period.

4 Experimental Setup

The following provides a brief overview of the experimental setup and the rationale behind the choices. Additional details and parameters for reproduction can be found in the Appendix. The complete code base, including the experimental setup and models, is made publicly available[1].

4.1 Dataset and Environment

All experiments are conducted using the DenseVLC dataset [1]. For further information about this dataset, please see Section 2.1. We split the dataset into 80% for training, 10% for validation, and 10% for testing. The training set is used to optimize model parameters, the validation set is used for tuning hyperparameters and monitoring for overfitting, and the test set is held out entirely for final performance evaluation to ensure unbiased assessment.

Our experiments are run on a workstation with Ryzen 7 8845HS and an RTX 4070, using PyTorch 2.7.0, and all models are trained and evaluated using the same computational environment to ensure consistency.

4.2 Baseline Model Setup

As our primary model baseline, we adopt the multilayer perceptron (MLP) architecture defined by Zhu *et al.* [11], which consists of five hidden layers with 256, 512, 1024, 512, and 256 units respectively, using ReLU activations. The model is trained using the Adam optimizer with a learning rate of 1×10^{-4}, a batch size of 64, and for 25 epochs.

4.3 Data Preprocessing and Parameter Settings

To evaluate the contribution of our data cleaning and augmentation methods, we first apply the preprocessing pipeline, as described by Zhu *et al.*, on the original dataset without degradation. We then apply our modified preprocessing pipeline to the same original dataset and compare the resulting model performance to that achieved using the baseline approach. This allows us to isolate the effect of preprocessing from other factors such as LED aging.

As described in Section 3.2, we use a scoring-based outlier removal step with a fixed bias of $\nu = 0.25$, selected based on empirical performance on the validation set.

4.4 Improved Model

In order to evaluate our improved model, as introduced in Section 3.4, we train it under the same conditions as the baseline MLP. This includes using the same training and test splits, optimizer settings, batch size, and number of epochs to ensure a fair comparison. The dataset consists of the preprocessed data obtained using the best-performing method identified in Section 5.1.

[1]The full source code, along with instructions, is available at: https://github.com/einstein8612/VLP

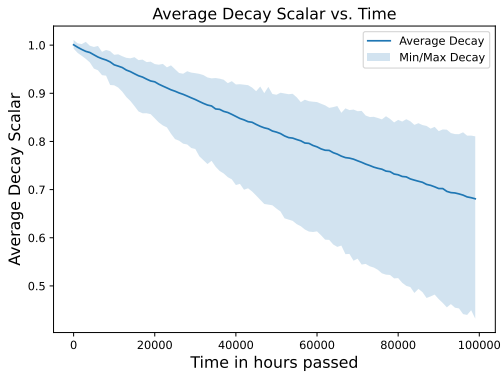


Figure 5: This plot shows the average decay over time, with a shaded region for the observed min and max values.

4.5 Degradation Simulation

To simulate realistic long-term LED aging, we use the simulation model as described in Section 3.6. The decay constants k_i are randomly generated, such that the LEDs retain approximately 90% of their original brightness over a period ranging from 10,000 to 50,000 hours. These values reflect real-world LED performance, with the LEDs in our dataset reportedly reaching this threshold at approximately 33,000 hours. This behavior is illustrated in Figure 5. We set the probability to flicker to $q = 0.001$.

4.6 Reproducibility and Multiple Runs

To ensure the robustness and reliability of our results, all experiments are repeated across five independent runs using different random seeds. For each configuration, we report the mean and standard deviation of the positioning error. This procedure helps mitigate the impact of randomness in initialization and training, providing a more stable and representative performance estimate.

4.7 Evaluation Metrics

We report positioning accuracy using the mean Euclidean distance between predicted and ground-truth coordinates. For time-dependent evaluations, especially under simulated LED degradation, we plot results as a function of time t to capture performance trends over the system's lifespan.

5 Results

This section presents the experimental results in three parts: first, a comparison of preprocessing strategies; second, an evaluation of our positioning model against the baseline; and third, an assessment of our method's performance under LED degradation scenarios compared to the baseline.

5.1 Improved Data Cleaning

As outlined in the methodology section, we introduced three improvements to the data cleaning pipeline: first, estimating LED positions earlier in the process; second, modifying the scoring function to better distinguish small, non-continuous signal variations; and third, replacing the mean with inverse distance weighted interpolation for reconstructing signals based on the Lambertian model. Table 1 presents the impact of both their individual contributions and combinations on model accuracy. The results show the mean Euclidean error along with the standard deviation over 5 independent runs.

Table 1: Impact of Preprocessing Improvements on Model Accuracy

Preprocessing Step	**Error (mm)**	**Std (mm)**
Baseline (Zhu *et al.*) [11]	17.77	2.33
Early LED Position	**13.49**	**1.64**
Modified Scoring	18.56	3.97
IDW	15.98	3.63
Early LED Position + Scoring	31.68	20.63
Early LED Position + IDW	**12.13**	2.31
Scoring + IDW	17.25	**2.12**
All Three Combined	17.88	4.16

From the results, we observe that using both early LED position estimation and inverse distance weighted interpolation improve model accuracy by over 30%. In contrast, the modified scoring function appears to negatively impact performance. This likely stems from the fact that low-intensity, outer signals contribute less to positioning accuracy, while the modified scoring function disproportionately penalizes high-intensity regions that are more critical for precise localization.

5.2 Improved Model

Table 2 summarizes the positioning error results for the baseline MLP model by Zhu *et al.* [11], compared our improved residual MLP model. The results show the mean Euclidean error along with the standard deviation over 5 independent runs.

Table 2: Positioning error comparison of different models evaluated on the cleaned dataset described in Section 4.4.

Model	**Error (mm)**	**Std (mm)**	**Params**
Baseline (Zhu *et al.*) [11]	12.13	2.31	1323010
Residual MLP (Ours)	**11.28**	**1.59**	**36148**

The improved residual MLP model achieves a small reduction in mean positioning error compared to the baseline, highlighting the effectiveness of our architectural redesign. In addition, the model size is reduced by approximately 97%, making it highly suitable for deployment on resource-constrained edge devices.

5.3 Online Degradation Scalar Learning

As introduced in Section 3.5, we propose an online learning framework to adaptively compensate for LED degradation during deployment. This approach continuously updates a scaling vector per LED based on incoming RSS samples and corresponding position predictions, using a robust RANSAC line fitting technique to estimate degradation factors without manual recalibration.

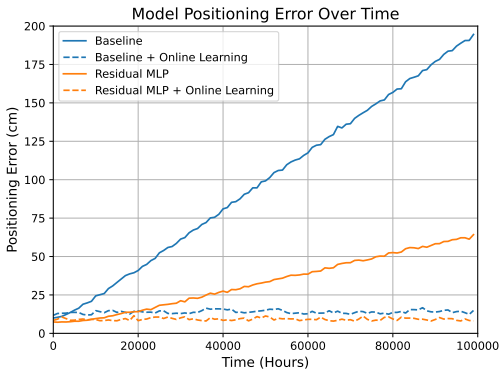


Figure 6: Positioning error over time for the baseline and the proposed model, each evaluated with and without online learning. In both cases, the error remains stable over time when our online learning method is applied.

Figure 6 shows how online learning scores against both the baseline and the model proposed in Section 3.4. The normalization layer in the proposed model initially helps with robustness against minor degradation, but we can clearly see that it is not sustainable. In contrast, the online learning method maintains a stable error throughout the entire simulated period for both models, indicating that its effectiveness is consistent and not dependent on the underlying model architecture.

In Table 3, we present some key statistics at relevant timestamps. The baseline model, without any adaptation, starts with a relatively low initial error of 9.97 mm. However, this error rapidly starts deteriorating and eventually reaches 194.53 mm by $t = 100{,}000$. The online learning models exhibit slightly higher initial errors compared to their respective foundational models. This is due to the fact that before evaluation, the model undergoes a warm-up phase with 50 randomly selected samples, during which it updates its parameters. This early adaptation occasionally leads to slight overcompensation, causing a minor initial accuracy drop. Nevertheless, by the end of the simulation, both online learning models achieve an error below 10% of the baseline's, demonstrating that this small initial increase is justified by substantial long-term performance gains.

Online Learning Hyperparameter Tuning and Trade-Off Analysis

In the previous section, we demonstrated that the proposed online learning method significantly outperforms the baseline once the LEDs have degraded sufficiently. However, this method depends on several hyperparameters, which we tuned by evaluating their performance within our simulation. The results reveal key insights, especially for scenarios where factors beyond accuracy are important.

A summary of the hyperparameter tuning results can be

Table 3: Different model accuracy over time. "Mean" is the average positioning error across time. "Std" is the standard deviation. "CV" is the coefficient of variation (std/mean), indicating performance stability. Lower is better for all metrics.

	Error (mm) and % improvement vs Baseline					Statistics		
Model	t=0	t=25000	t=50000	t=75000	t=100000	Mean	Std	CV
Baseline (Zhu *et al.*) [11]	9.97	48.78	98.60	145.16	194.53	98.86	55.01	0.56
Residual MLP (Ours)	7.43 (-25.5%)	16.88 (-65.4%)	32.72 (-66.8%)	47.78 (-67.1%)	64.22 (-67.0%)	33.35	17.48	0.52
Baseline + Online Learning	11.71 (+17.5%)	13.23 (-72.9%)	14.66 (-85.1%)	12.99 (-91.1%)	14.87 (-92.4%)	13.94	1.06	**0.08**
Residual MLP + Online Learning	**8.45** (-15.3%)	**10.59** (-78.3%)	**9.96** (-89.9%)	**9.10** (-93.7%)	**8.12** (-95.8%)	**9.23**	**0.86**	0.09

found in Table 4, showing the impact of varying samples per timestep, timestep size, and the number of samples used for line fitting on model performance and stability.

Table 4: Effect of online learning hyperparameters on performance. We define a *balanced* setting as one that achieves $\geq$90% of the best observed accuracy.

Parameter	**Range**	**Best Accuracy**	**Balance**
Samples/timestep	10–100	100	30
Timestep size (h)	100–10000	100	1500
Line fit samples	10–50	50	10

From these experiments, several important insights emerge. First, we can enable deployment on low-memory hardware by reducing the number of samples per timestep, as even a small sample count can yield acceptable accuracy. Second, the timestep size must remain relatively small. If too large, LED degradation between updates becomes too severe to compensate effectively. Finally, only a few samples are needed to fit the degradation scalar accurately. Since this step scales linearly with memory usage, using fewer line fit samples allows us to allocate more memory to timestep samples instead, improving overall stability.

While the optimal parameters for maximum accuracy are logical, this analysis provides the necessary guidance for achieving a balance between performance and resource efficiency.

6 Deployment on a Microcontroller

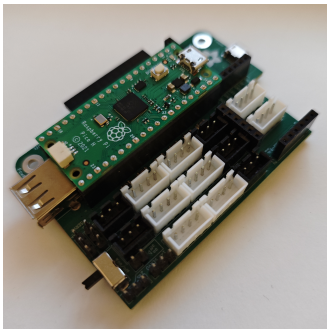


Figure 7: Raspberry Pi Pico

To evaluate the real-world feasibility of our system, we deployed the trained model onto a Raspberry Pi Pico microcontroller using the TensorFlow Lite Micro framework [2]. This platform provides a streamlined inference pipeline for quantized neural networks optimized for resource-constrained environments.

6.1 Quantization and Memory Optimization

Our improved model that we discussed in Section 3.4 has a size of 147 KB pre-quantization. This allows us to directly quantize and deploy it on TinyML hardware, such as the Raspberry Pi Pico. As we were well below the required size, however, we made some slight changes to the model to allow it to quantize better. Specifically, we adjusted the bottleneck compression ratio from 90% to 75%, and replaced H-Swish activations with standard ReLU functions. These changes slightly increase computational cost, but improve quantization performance by preserving more information per layer, resulting in higher accuracy in the quantized case. The final size of this altered model was 304 KB pre-quantization.

This model, trained in 32-bit floating point precision, was quantized to 8-bit integers using post-training quantization through TensorFlow. This reduced the size of the model from 304 KB to 103 KB. The quantized model maintained comparable inference accuracy, with the full-precision model having an accuracy of around 9 mm while the quantized model achieves an accuracy of around 12 mm. When comparing the quantized model to our full-precision model meant for deployment and our best model as described in Section 3.4, we observe a precision loss of approximately 3 mm and 5 mm, respectively.

In order to leverage our online learning framework, we used a compact data representation strategy based on the existing data augmentation pipeline, as discussed in Section 2.2. Specifically, the original fingerprint data was downsampled by a factor of 4 along both axes, reducing the total size from 21.3 MB to just 1.06 MB. Only the downsampled data is stored in the binary, and during inference, the required samples are augmented on-demand. This not only enables the full framework to run entirely on-edge, but also supports our objective of reducing the complexity and overhead of the fingerprinting process.

6.2 Inference Performance

Inference using the quantized model executes in approximately 8.3 ms per sample on the Pico, including input preprocessing and postprocessing steps. This is well within the constraints of real-time indoor positioning applications. For reference, without online learning our model inference takes 6.4 ms per sample. This means adding our online learning

has basically no extra cost associated with it when it comes to speed. The accuracy over time can be seen in Figure 8.

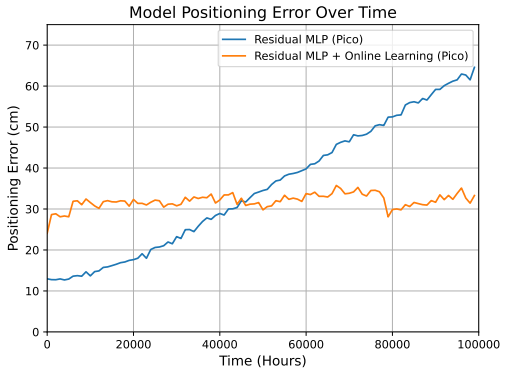


Figure 8: Positioning error over time on the Raspberry Pi Pico, comparing the baseline model with the model using online learning. Note that the error remains stable over time when online learning is applied.

6.3 Conclusion on Feasibility

The successful deployment of our quantized model on a resource-constrained microcontroller demonstrates the feasibility of running adaptive VLP systems entirely on-device. Performing online learning and localization at the edge enables scalable, low-cost, and robust indoor deployments. However, quantization significantly impacts the ability of the online learning to correctly adapt, making the trade-off worthwhile only for longer deployment durations.

7 Responsible Research

This research was conducted with a strong emphasis on transparency, ethical use of tools, and environmental sustainability. The following subsections detail our efforts to ensure reproducibility, responsible use of language models, and the ecological impact of our work.

7.1 Reproducibility

All experiments, models, and datasets used in this research are fully documented and made publicly available to ensure transparency and reproducibility. We provide detailed instructions and code repositories to allow others to replicate our results and build upon our work. As mentioned in the experimental setup, the code base is made publicly available[2].

7.2 Usage of LLMs

Large language models (LLMs) were used solely to assist with rewriting and improving the clarity of the manuscript, with all content being carefully reviewed. All technical content, experimental design, and analysis were developed independently to maintain the integrity of the research.

[2]The full source code, along with instructions, is available at: https://github.com/einstein8612/VLP

7.3 Environmental Responsibility

Our research promotes environmental sustainability by focusing on extending the lifespan of LED-based positioning systems through adaptive online learning. By enabling longer-term use and reducing the need for frequent hardware replacement, we contribute to minimizing electronic waste.

8 Discussion

In the following subsections, we briefly discuss our results and their broader significance. We highlight how each component, including data preprocessing, model design, and online learning, contributes to building systems for Visible Light Positioning that are scalable, accurate, and efficient.

8.1 Data Preprocessing

The results demonstrate the effectiveness of our proposed improvements across various aspects of the system. The enhanced data cleaning steps, particularly early LED position estimation combined with inverse distance weighted interpolation, significantly reduce positioning errors by over 30%, underscoring the critical role of robust preprocessing for reliable Visible Light Positioning (VLP). Conversely, the modified scoring function proved less effective, suggesting that careful tuning or alternative methods may be required to avoid degrading model performance.

8.2 Model Changes

Our lightweight residual MLP model achieves comparable accuracy to the baseline while drastically reducing the number of parameters by over 97%. This reduction was achieved while still having a lower mean error than the baseline, in some runs as low as 7.33 mm. These results indicate the potential for even smaller models to maintain or improve positioning accuracy, opening avenues for further model compression and optimization tailored for resource-constrained deployments.

8.3 Online Learning

Finally, the online learning framework exhibits remarkable resilience to LED degradation over time, maintaining near-constant positioning accuracy despite significant hardware aging. While introducing some initial warm-up overhead, it ultimately delivers a sustained improvement of above 90% in accuracy compared to the non-adaptive baselines. Hyperparameter tuning further reveals how to balance accuracy, stability, and memory constraints for edge deployment. Together, these results suggest that adaptive VLP systems can offer robust, long-term localization without frequent manual recalibration or hardware replacement, reducing e-waste and maintenance costs.

Looking ahead, the increasing availability of TinyML hardware with floating-point support (e.g., float16/float32) may also help relax current quantization constraints. This could enable more accurate online adaptation at the edge without the degradation introduced by 8-bit quantization.

9 Conclusions and Future Work

In this research paper, we proposed an improved preprocessing pipeline, including a method for estimating LED positions from noisy data. These improvements, along with our improved model, have enabled us to gain an average reduction of more than 35% in positioning error. Furthermore, our online learning approach to data degradation has provided us with a constant error rate regardless of LED degradation. This corresponds to an error reduction of more than 90% after sufficient LED degradation. Lastly, we have shown that this method is able to be deployed on TinyML, without much trouble. There is some performance degradation, but the general trend still stays the same even with quantization and access to augmented reference data rather than real reference data.

While our approach has demonstrated strong performance and robustness in the face of LED degradation, there remain several promising directions for future research. These include developing predictive models to estimate when LED degradation begins to impact positioning accuracy, enabling timely recalibration instead of continuous adaptation. While this adds some targeted fingerprinting, it could improve long-term reliability when online learning alone is insufficient. Another direction is adding uncertainty estimation into the positioning pipeline using techniques such as Bayesian networks or Monte Carlo dropout, allowing the system to prioritize high-confidence samples. Collaborative learning is also a promising direction, where multiple edge devices adapt locally and periodically share updates, improving robustness and adaptation quality, especially when combined with uncertainty-based sample selection. Finally, although this work focused on a dense LED layout, future research should further explore alternative configurations to better understand their effect on accuracy and robustness under degradation.

A Experiment Reproduction Parameters

This appendix presents comprehensive tables outlining the reproducibility parameters for each individual experiment discussed in Section 5. For clarity and consistency, each table details the relevant settings, model architectures, training configurations, and simulation parameters used.

Table 5: Reproducibility Parameters for `Baseline`

Parameter	Value
Task	MLP-TINY
Data Source	LAMBERTIAN-IDW
Simulation Parameters	
Standard Deviation	0.005
Total Time	100000.0
Samples per Timestep	100
Timestep Duration	1000
Flickering Probability	0.001
Device	cuda
Seed	42
Model	MLP
dim_in	36
dim_out	2
hidden_layers	[256, 512, 1024, 512, 256]
lr	0.001
loss	mse
optimizer	adam
batch size	64
epochs	25
n_timesteps	100

Table 6: Reproducibility Parameters for `Residual MLP`

Parameter	Value
Task	MLP-PICO
Data Source	LAMBERTIAN-IDW
Simulation Parameters	
Standard Deviation	0.005
Total Time	100000.0
Samples per Timestep	100
Timestep Duration	1000
Flickering Probability	0.001
Device	cuda
Seed	42
Model	MLPPico
dim_in	36
dim_out	2
hidden_layers	[256, 25, 256, 25, 256]
lr	0.001
loss	mse
optimizer	adam
batch size	64
epochs	25
n_timesteps	100

Table 7: Reproducibility Parameters for `Residual MLP (Pico)`

Parameter	Value
Task	MLP-TINY
Data Source	LAMBERTIAN-IDW
Simulation Parameters	
Standard Deviation	0.005
Total Time	100000.0
Samples per Timestep	100
Timestep Duration	1000
Flickering Probability	0.001
Device	cuda
Seed	42
Model	MLP
dim_in	36
dim_out	2
hidden_layers	[256, 64, 256, 64, 256]
lr	0.001
loss	mse
optimizer	adam
batch size	64
epochs	25
n_timesteps	100

B Full model specifications

This section provides a detailed layer-wise breakdown of the `MLPResNet` architectures used in our experiments. For each model variant, the tables show the input dimensions, layer operations, output feature sizes, and nonlinear activation functions employed.

Table 8: Layer-wise Architecture of `MLPResNet`. NL denotes the type of nonlinearity used. HS means h-swish.

Input Size	Operator	#out	NL
(batch, 36)	NormalizeInput	36	-
(batch, 36)	Linear	256	ReLU
(batch, 256)	Linear	64	-
(batch, 64)	HS	64	HS
(batch, 64)	Linear	256	-
(batch, 256)	Add (Residual + HS)	256	HS
(batch, 256)	Linear	64	-
(batch, 64)	ReLU	64	HS
(batch, 64)	Linear	256	-
(batch, 256)	Add (Residual + HS)	256	HS
(batch, 256)	Linear	2	-

Table 9: Layer-wise Architecture of `MLPResNet (Pico)`. NL denotes the type of nonlinearity used. HS means h-swish.

Input Size	**Operator**	**#out**	**NL**
(batch, 36)	NormalizeInput	36	-
(batch, 36)	Linear	256	ReLU
(batch, 256)	Linear	64	-
(batch, 64)	ReLU	64	ReLU
(batch, 64)	Linear	256	-
(batch, 256)	Add (Residual + ReLU)	256	ReLU
(batch, 256)	Linear	64	-
(batch, 64)	ReLU	64	ReLU
(batch, 64)	Linear	256	-
(batch, 256)	Add (Residual + ReLU)	256	ReLU
(batch, 256)	Linear	2	-

References

[1] Jona Beysens, Ander Galisteo, Qing Wang, Diego Juara, Domenico Giustiniano, and Sofie Pollin. Densevlc: a cell-free massive mimo system with distributed leds. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '18, page 320–332, New York, NY, USA, 2018. Association for Computing Machinery.

[2] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Shlomi Regev, Rocky Rhodes, Tiezhen Wang, and Pete Warden. Tensorflow lite micro: Embedded machine learning on tinyml systems, 2021.

[3] Trong-Hop Do and Myungsik Yoo. An in-depth survey of visible light communication based positioning systems. *Sensors*, 16(5), 2016.

[4] Pengfei Du, Sheng Zhang, Chen Chen, Arokiaswami Alphones, and Wen-De Zhong. Demonstration of a low-complexity indoor visible light positioning system using an enhanced tdoa scheme. *IEEE Photonics Journal*, 10(4):1–10, 2018.

[5] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[6] TM-21 Working Group et al. Projecting long term lumen maintenance of led light sources (ies tm-21-11). *New York, NY, USA: Illuminating Engineering Society of Norh America (IESNA)*, 2008.

[7] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3, 2019.

[8] I. Kåsa. A circle fitting procedure and its error analysis. *IEEE Transactions on Instrumentation and Measurement*, IM-25(1):8–14, 1976.

[9] Junhai Luo, Liying Fan, and Husheng Li. Indoor positioning systems based on visible light communication: State of the art. *IEEE Communications Surveys and Tutorials*, 19(4):2871–2893, 2017.

[10] Sheng Zhang, Pengfei Du, Chen Chen, and Wen-De Zhong. 3d indoor visible light positioning system using rss ratio with neural network. In *2018 23rd Opto-Electronics and Communications Conference (OECC)*, pages 1–2. IEEE, 2018.

[11] Ran Zhu, Maxim Van den Abeele, Jona Beysens, Jie Yang, and Qing Wang. Centimeter-level indoor visible light positioning. *IEEE Communications Magazine*, 62(3):48–53, 2024.

[12] Zhiyu Zhu, Yang Yang, Mingzhe Chen, Caili Guo, Julian Cheng, and Shuguang Cui. A survey on indoor visible light positioning systems: Fundamentals, applications, and challenges. *IEEE Communications Surveys and Tutorials*, pages 1–1, 2024.

[13] Yuan Zhuang, Luchi Hua, Longning Qi, Jun Yang, Pan Cao, Yue Cao, Yongpeng Wu, John Thompson, and Harald Haas. A survey of positioning systems using visible led lights. *IEEE Communications Surveys and Tutorials*, 20(3):1963–1988, 2018.