Indoor positioning with AI/ML using simulated 5G data

Generalizability to changes in indoor environment

PHILIP SÖDERBOM & JONATHAN NILSSON
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY


Indoor positioning with AI/ML using simulated 5G data

Generalizability to changes in indoor environment

Philip Söderbom
ph4041so-s@student.lu.se

Jonathan Nilsson
jo1086ni-s@student.lu.se

Department of Electrical and Information Technology
Lund University

Master's thesis work carried out at Ericsson

Supervisors: Emma Wittenmark, emma.wittenmark@ericsson.com
David Sugirtharaj, david.sugirtharaj@ericsson.com
William Tärneberg, william.tarneberg@eit.lth.se
Aleksei Fedorov, aleksei.fedorov@eit.lth.se

Examiner: Michael Lentmaier, michael.lentmaier@eit.lth.se

June 17, 2024

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

With future technologies and Industry 4.0, the need for robust and accurate positioning to optimize productivity and industrial operations increases. The previous positioning methods using triangulation or angle-of-arrival are not good enough, especially not for clutter-dense factories with poor line-of-sight conditions.

For this thesis project, a convolutional neural network model was trained to better estimate time-of-arrival and classify line-of-sight for a clutter-sparse simulated factory. For the clutter-dense factory, a residual network fingerprinting model was trained to map channel impulse responses to a position. Both the clutter-sparse and clutter-dense factories were modified by moving and rotating machines as well as adding robots in order to investigate the robustness to changes in the factory environment.

The factories simulated production areas, assembly areas, beam structures and robots using the Ericsson state-of-the-art version of Nvidia Omniverse.

The fingerprinting model was also used in a real scenario from Mobile World Congress 2024, where a robot drove two routes in an open office environment.

The results show that both neural networks gave a positioning error below 1 m. For the modified scenarios, the convolutional neural network was more robust than the residual network fingerprinting model. The modifications mainly impacted positioning errors locally where changes were introduced.

Keywords: Indoor positioning, 5G, Industry 4.0, Simulations, AI, ML, Neural Networks

Acknowledgment

First and foremost we, would like to express our gratitude towards our supervisors at Ericsson, Emma Wittenmark and David Sugirtharaj, for their help and supportive guidance throughout the entire project. The same goes for line manager Anders Ericsson, for all the helpful discussions and positivity through more difficult times.

We would also like to thank our supervisors at LTH, William Tärneberg and Aleksei Fedorov for helping us define the scope of this thesis and the meetings to discuss progress and ideas. An additional thanks to Michael Lentmaier for taking on the role as examiner.

Several people at Ericsson have provided invaluable help with their expertise in different steps during the thesis. Therefore a special thank you to Atieh Rajabi Khamesi, Johannes Nygren, Rong Du A, Yves Teganya and Erik Gudmundson.

Sincerely,
Philip & Jonathan

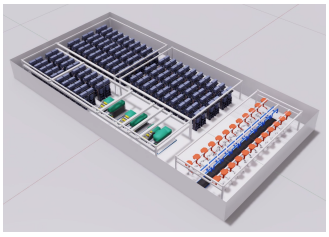
Popular Science Summary

Indoor positioning in simulated factory environments and generalization aspect with AI

Industrial processes are moving toward automation, in what is called Industry 4.0, the fourth industrial revolution. For factories and warehouses, the need for efficiency and precision is paramount. For instance, precise indoor positioning of autonomous vehicles or manufactured goods. The use of Artificial Intelligence (AI) could be a way to facilitate improved positioning accuracy and adaptibility for these needs.

In factory environments, machinery and equipment are often densely deployed with other obstructions like pillars and beam structures. For such scenarios with limited visibility, conventional positioning solutions face challenges in providing accurate positioning. Consequently, leveraging AI methods to address issues of conventional solutions is a growing interest.

This thesis has been conducted by creating two baseline factory environments, designed to have different line-of-sight conditions. The simulated baseline factories have been modified by introducing changes to the physical arrangement of machines and equipment.



Simulated 5G data for the factory environments was used to train two AI models, learning patterns in the data to do positioning predictions. Besides evaluating positioning for the two baselines, the models' capability to generalize to modified factory environments is investigated. The generalization aspect is of interest as it would provide robust and flexible solutions for autonomous industries, aligning with the Industry 4.0 transformation.

The positioning results are promising and lead to significant improvements compared to conventional solutions, especially for scenarios with poor line-of-sight conditions. Moreover, the generalization aspect is assessed to do quite well, depending on how drastic the changes are.

Towards the end, real 5G data measurements from an office environment is evaluated with one of the AI models.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AGV	Automated Guided Vehicle
AI/ML	Artificial Intelligence/Machine Learning
AMF	Access and Mobility Management Function
AoA	Angle-of-Arrival
AWGN	Additive White Gaussian Noise
CDF	Cumulative Distribution Function
CIR	Channel Impulse Response
CNN	Convolutional Neural Network
DNN	Deep Neural Network
gNB	gNodeB
JSI	Jaccard Similarity Index
LMF	Location Management Function
LoS	Line-of-Sight
MWC	Mobile World Congress
NLoS	Non-Line-of-Sight
NR	New Radio
RAN	Radio Access Network
ResNet	Residual Network
RMSE	Root Mean Squared Error
ROS	Robot Operating System
SRS	Sounding Reference Signals
ToA	Time-of-Arrival
TRP	Transmission Receiver Point
UE	User Equipment
3GPP	3rd Generation Partnership Project

Chapter 1
Introduction

1.1 Background

In an era of continuous advancements in technology and connectivity, innovative solutions allow for increased efficiency and precision across various domains. In the context of Industry 4.0, the fourth industrial revolution with ongoing shift to wireless factories, the need for precise indoor positioning solutions is emphasized. Accurate positioning is paramount for enabling efficient automation and increased productivity in the modern industrial landscape. For companies, positioning can be used for optimizing logistics with autonomous vehicles, enhancing safety, or accurately locating assets [1]. Asset localization could be of manufactured goods, robots, or personnel with user equipment (UE). It is already in use today, but with improved positioning accuracy, this can be better applied and new fields can be explored.

Technology advancement in localization indoors is important since global navigation satellite systems such as GPS can not be utilized indoors due to absence of clear line-of-sight (LoS) communications with satellites. Today, indoor positioning is a topic researched using several technologies such as WiFi, Bluetooth and 5G. Although, there is not one best solution when it comes to accuracy, cost and availability, this thesis will look deeper into the possibilities using 5G.

Nonetheless, most positioning solutions struggle to provide accurate positioning for environments with heavy non-line-of-sight (NLoS) conditions causing multipath propagation. Multipath propagation is a phenomena arising from radio signals interacting with surrounding obstacles in the environment, for instance by reflections. In [2] a method for mitigating the negative impact of NLoS on positioning accuracy is studied. Even though the effects of NLoS are dampened, the results therein are desired to be improved upon. Consequently, leveraging Artificial Intelligence (AI) and Machine Learning (ML) methods to address such issues is a growing interest.

1.2 Previous Work

The application of AI/ML in the field of indoor positioning has been explored with various models. A Convolutional Neural Network (CNN) model is presented in [3], stating that CNN in combination with multiple input multiple output (MIMO) datasets is better than other neural networks thus far due its accurate positioning. Another Deep Neural Network (DNN) model, namely a recurrent neural network, is studied with the conclusion that DNNs perform better than positioning approaches based on decision trees [4].

Over the last few years, fingerprinting models have gained prominence due to their applicability to scenarios with heavy NLoS conditions, taking advantage of multipath transmission to serve as a fingerprint, characterizing the communication channel for a transmitting device [5][6]. The fingerprinting approach using DNNs is proposed for indoor localization in [7] and [8].

Ericsson has conducted previous work regarding indoor positioning using AI/ML, and developed CNN and Residual Network (ResNet) models for this purpose, tailored for sparse (moderate LoS) and dense (heavy NLoS) scenarios respectively. These models have been trained with data from *statistical channel models* defined by the 3rd Generation Partnership Project (3GPP), which is a standardization organization developing specifications within telecommunication technologies [9][10].

1.3 Problem Formulation

This master thesis aims to expand the research about AI/ML for indoor positioning using 5G data. By evaluating the performance, this study seeks to enhance indoor positioning accuracy, and thus potentially pave the way for optimizing industrial operations and productivity, aligning with the ongoing shift to wireless factories described in Section 1.1.

For indoor environments with moderate LoS conditions, a CNN model is applied. This model aims at enhancing input data to a legacy time-of-arrival based solution. For environments with heavy NLoS, a ResNet fingerprinting model is applied. This approach does not rely on the legacy time-of-arrival solution and can therefore withstand these harsh conditions.

The generalization capabilities of the models in terms of physical changes to the indoor environment are of particular interest for this project. Specifically, simulated factory environments are studied from which site-specific *ray-tracing channel models* are used to obtain 5G data through a series of simulations. There are fundamental differences for the *statistical* and *ray-tracing* channel model approaches. Most prominently, a ray-tracing model requires highly detailed environment models, including material properties which is not needed for a statistical channel model [11]. Using a ray-tracing channel model also makes it possible to investigate how different physical changes to the environment affect the results.

In addition to simulated 5G data, real measurements from an office environment is used to see if a ResNet fingerprinting model is applicable not only to simulated data.

1.3.1 Research Questions

- ***RQ1: To what extent can AI/ML models improve positioning accuracy in industrial indoor environments compared to legacy time-of-arrival based solutions?***
- ***RQ2: How robust are these models in terms of generalizing to physical changes in the environments?***

1.3.2 Limitations

The positioning using the simulated 5G data will be limited to stationary objects, even though robots and people often move with a certain speed. This is because of the limited time doing this master thesis. Thus, tracking of moving objects is not in the scope of the project.

Also, the antenna patterns on the transmitter and receiver are implemented as omni-directional for the simulated 5G data. This is an ideal antenna and is not possible to replicate in reality. This is because other aspects, such as rotating the UE and using different antenna patterns would need to be applied in order to draw conclusions. Although this is an interesting aspect, it will not be covered in this thesis due to time limitations.

The AI/ML models used in this project is a CNN for moderate LoS scenarios and a ResNet fingerprinting model for heavy NLoS scenarios. This is because they have have shown promising results with data from statistical channel models and due limitation in time, only two AI/ML models are investigated.

1.4 Disposition

Chapter 2 of this thesis introduces the theory of 5G communication, channel modeling, and legacy positioning solutions with 5G. Furthermore, principles of machine learning are given to describe the two AI/ML models leveraged in this project for indoor positioning. Chapter 3 describes the methodology with steps taken to arrive at the results, including how factory environments were generated and how 5G data was collected through simulations and later used to train two AI/ML models. The last section of Chapter 3 describes how real 5G data has been collected through measurements in an office environment.

In Chapter 4, the results of the thesis are presented. First, positioning results with a legacy time-of-arrival triangulation method are presented, followed by the results produced by the two AI/ML models. The presented results are discussed in Chapter 5. Following the discussion, Chapter 6 presents the conclusions and answers to the research questions.

Chapter 2
Theory

To understand more about 5G and how it is currently used for positioning, this chapter introduces the theory of 5G communication, channel properties, and how legacy positioning solutions work. Then, an overview of machine learning is given, including deep learning and the core mechanisms of neural networks, leading to the introduction of two neural network architectures used to do positioning in this master thesis.

2.1 5G Communication

The first initiative that later on became 5G started in 2015 by 3GPP as a workshop to set the scope for New Radio (NR). The goal of NR was to exploit the possibilities to achieve even stricter requirements. Three years later in 2018, the requirements for commercial deployment were reached [12, p. 5].

As a difference from the previous generations of cellular network, 5G provided features allowing more applications in today's society. It has possibilities to support Enhanced Mobile Broadband (eMBB), allowing to transmit and receive data with a higher rate and increased reliability. This leads to better quality when streaming, opening more opportunities for internet of things and much more. Another feature was Ultra Reliable Low Latency Communications (URLLC). This puts a strict requirement on latency and reliability allowing precise operations when low latency is critical. A third feature is Massive Machine-Type Communications (mMTC). This provides support for more devices within a small area than the previous cellular network 4G, as long as data is sent intermittently.

The frequencies used in 5G can be divided into two ranges. Frequency range 1 covers frequencies from 0.41 GHz to 7.125 GHz and frequency range 2 and covers frequencies between 24.25 GHz and 71 GHz [13]. Frequency range 1 is often used for indoor positioning.

In the 5G architecture, the Radio Access Network (RAN) includes one or several gNodeB (gNB) units that control all radio-related functions in one or several cells by the baseband processing unit. This includes connection establishment, admission control and more. A logical setup of the gNB could be one or more Transmission Receiver Points (TRPs) transmitting and receiving radio signals. These radio signals are then processed by the baseband processing unit placed elsewhere.

When doing positioning with 5G, the location management function (LMF) has an important role in the architecture. This function, together with the access and mobility management function (AMF), is located in the core network and is the function that estimates the position of the UE. In order for the LMF function to get the information needed from the RAN, the information is transferred via the AMF as a link between the core network and RAN [14][15, p. 73-78]. Figure 2.1 shows an illustration of this architecture.

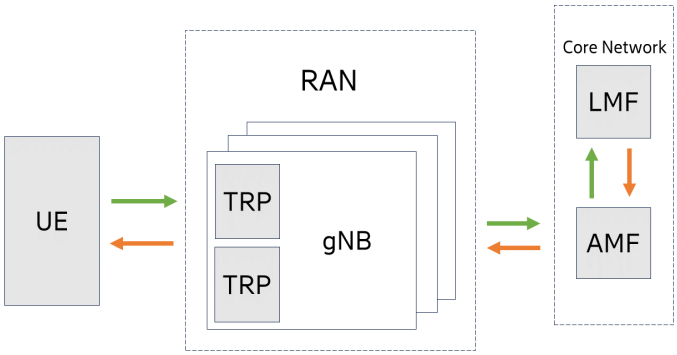


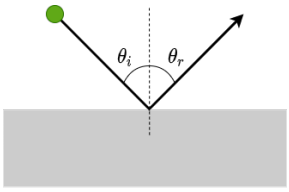
Figure 2.1: 5G architecture visualizing how the different parts communicate.

2.2 Multipath Propagation

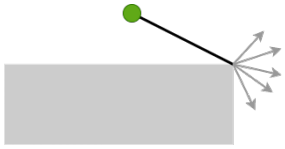
The propagation of radio signals describes how the electromagnetic waves travel and interact with the surrounding environment [16, p. 19-20]. Multipath propagation arise from interactions with surrounding obstacles in the environment. This means that the same radio signals gets received multiple times by the same receiver, but with different amplitude, delay and phase. A way to capture multipath propagation in simulations is by using a ray-tracer. This traces paths between one or several transmitters and receivers, using descriptions of the environment and properties of the antennas [17].

The different interactions that can occur to a radio signal propagating through space are described below, and depicted in Figure 2.2.

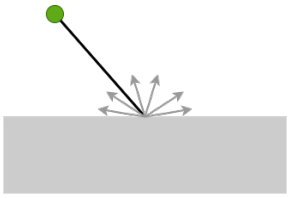
- **Specular reflection** refers to when a radio signal hits a smooth surface with angle of reflection, θ_r equal to the angle of incidence θ_i following Snell's law [12, p. 49-53].
- **Diffuse scattering** refers to when the radio signal is reflected from a rough surface. Diffuse scattering causes multipath components with spread angle of departures, smaller amplitude, different phase shifts, and different delays [12, p. 65-68].
- **Diffraction** can be explained using Huygens' principal, which states that each point in a wave-front can be considered as a source of a spherical wave [12, p. 55-64]. When a radio signal hits an edge, it spreads in all directions similar to diffuse scattering. This gives the effect of signals curving an edge.
- **Transmission** refers to when the radio signal penetrates into and through the interfering object. The angle of refraction, θ_r follows Snell's law both entering and leaving the interfering object [12, p. 49-53]. Assuming the interfering object has a homogeneous material, the angle of incidence, θ_i is equal to the angel of transmission θ_t.



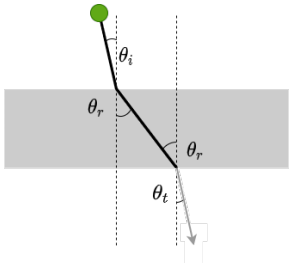
(a) Specular reflection.



(b) Diffraction.



(c) Diffuse scattering.



(d) Transmission.

Figure 2.2: Interactions causing multipath propagation.

2.3 Channel Impulse Response

The Channel Impulse Response (CIR) describes valuable characteristics of the wireless channel between a transmitter and receiver. As the name implies, an impulse is sent from the transmitter and interacts with its surroundings. When it is received by the receiver, the different multipath propagations' amplitude, delay and phase after the specular reflections, diffuse scatterings, transmissions and diffractions can be visualized.

The channel includes noise and is dependent on parameters such as angle of the transmitter and receiver. Since this is cumbersome to express in mathematical terms, a way to describe the CIR without these parameters can be seen in Equation 2.1.

$$h(\tau) = \sum_{i=1}^{M} c_i \delta(\tau - \tau_i) \tag{2.1}$$

In the equation above, c_i is the complex value characterizing the phase- and amplitude of multipath component i, $\delta(\tau - \tau_i)$ is the Dirac function delayed τ_i time instances and M is the number of multipaths included.

The time-of-arrival (ToA) is the time it takes for the impulse to be received by the receiver and can be seen in the CIR as the delay for the first multipath to be received. If the first multipath is affected by only transmission or no interaction at all, the receiver is deemed as having line-of-sight (LoS). On the contrary, if the receiver does not have LoS, it is said to be non-line-of-sight (NLoS). This classification is often difficult due the complexity of the CIR.

2.4 Positioning with 5G

When doing positioning using 5G, there are different methods that can be used. One method is to use angle-of-arrival that uses algorithms to determine the angle of the arriving radio signal in order to position the UE [18].

Another way to estimate position is to use the ToA followed by triangulation. This requires precise time synchronization between the TRPs which sometimes can be hard to acquire. This can be done using either downlink or uplink radio signals [14].

For both angle-of-arrival and ToA positioning methods, LoS between the transmitter and receiver is crucial for accurate positioning.

2.4.1 Uplink Time-of-Arrival Positioning

When estimating the positioning using uplink signals, the receiver evaluates at the Sounding Reference Signals (SRS) sent from the UE. These signals are used to estimate the channel and from that, the time of arrival can be calculated. The RAN decides where to schedule the SRS in the uplink and which SRS configuration to use. When the SRS later on is received, cross-correlation is done with the reference SRS to see how the channel has affected the SRS. This cross-correlation gives information about the channel and with some signal processing, a channel impulse response can be estimated.

As described in Section 2.3, the ToA is estimated as the delay to the first peak, corresponding to the first received multipath of the SRS. Since the LoS is difficult to classify accurately, a soft classification can be used. This means that a number is set depending on how likely the receiver is to have LoS, usually a number between 0 and 1. Otherwise, hard classification can be used. This classifies the link as either LoS or NLoS.

In order to do 2D triangulation, at least three ToA measurements are needed [19]. When this is fulfilled, these measurements are converted to distances by multiplying with the speed of light. With these distances, triangulation is achieved by solving the equation system 2.2

$$\begin{cases} d_1 = \sqrt{(x-x_1)^2 + (y-y_1)^2} \\ d_2 = \sqrt{(x-x_2)^2 + (y-y_2)^2} \\ d_3 = \sqrt{(x-x_3)^2 + (y-y_3)^2} \\ \vdots \\ d_i = \sqrt{(x-x_i)^2 + (y-y_i)^2} \end{cases} \tag{2.2}$$

where (x, y) is the position of the UE, and (x_i, y_i) is the position of the i:th TRP and d_i is the distance from the i:th TRP to the UE. When more than necessary ToA are involved in the triangulation, the system becomes overdetermined and several possible solutions are given. A visualization of four TRPs triangulating a UE can be seen in Figure 2.3.

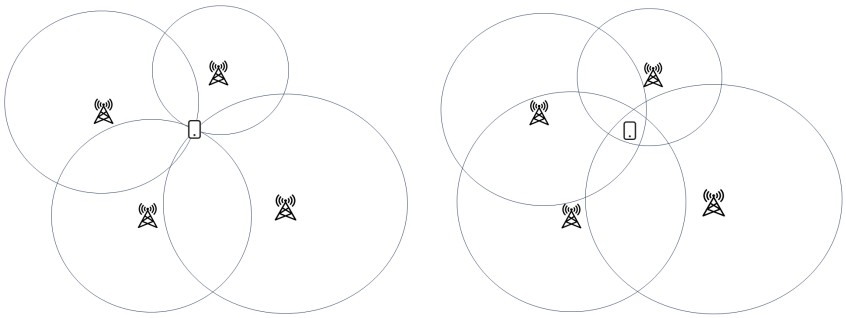


Figure 2.3: A visualization of triangulation with and without noise or synchronization errors.

In Figure 2.3a, the UE is positioned perfectly at the intersection of the four TRPs' estimated range. For Figure 2.3b, the triangulation is affected by noise and synchronization errors, leading to a less accurate positioning within an area. To get a single position estimation, a weighted least square method can be utilized [20]. This weighs the range from the receivers depending on the number from the LoS soft classification. All this computation, among other possible input parameters, are fed to the LMF that estimates a position.

2.5 Machine Learning

The term AI is prominently used and is a broad field. A subset of AI is Machine Learning (ML), which is the way of enabling machines or systems to learn from large amounts of data. ML applies algorithms to analyze and find patterns in the *training data*, rather than giving a computer explicit instructions. This training dataset contains samples of how an input x is related to an output variable y. In other words, learning the mapping $y = f(x)$. A training dataset containing n observed pairs of input-output samples can be denoted $\mathcal{T} = \{x_i, y_i\}_{i=1}^{n}$.

The purpose of an ML model is to fit a mathematical model to the training data such that it accurately predicts an output $\hat{y}$ and generalizes to previously unseen data, in which only the input variable x is known to the model. A too simple model is underfitting and fails to capture data complexities, which gives poor performance even on the training data. On the other hand, if the model is too complex it will be overfitting the training data which shows by large prediction errors due to a less general model.

2.5.1 Accuracy Metrics

Assuming having target values $\{y_i\}_{i=1}^{n}$, and predicted output values $\{\hat{y}_i\}_{i=1}^{n}$ from a trained model, the prediction errors $\{e_i\}_{i=1}^{n}$ can be formed to measure model accuracy using different metrics.

Metrics for numerical errors

The following metrics are often used for numerical errors:

- Root Mean Squared Error, RMSE = $\sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}$
- Cumulative Distribution Function, CDF: $F_X(x) = P(X \leq x)$

The RMSE metric penalizes large prediction errors, capturing outliers in the predictions [21, p. 214-215]. It is desired to be as small as possible, implying a model with better accuracy.

The CDF denoted $F_X(x)$, is a way to inspect the distribution of errors, X. The CDF evaluated at x is the probability that the error takes a value less than or equal to x [22, p. 76-77]. Different CDF percentiles are used to inspect the error distribution. If the positioning error is x meters at 90% of the CDF, then $F_X(x) = 0.90$, and consequently, a randomly selected sample has a positioning error below x meters with 90% probability.

Metrics for categorical errors

For a model performing binary classification, the following error metrics are commonly used:

- Accuracy = $\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$
- F1-score = $\frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$

Binary classification uses true negatives (TN), false negatives (FN), true positives (TP) and false positives (FP) to measure accuracy and F1-score [21, p. 216-217]. The accuracy metric shows the proportion of correct classifications, compared to the total number of predictions, while the F1-score focuses on TP, FP, and FN [23]. Both accuracy and F1-score are metrics within $[0, 1]$ and desired to be as close to 1 as possible, as it means the classification performance is better.

2.6 Deep Learning with Neural Networks

A neural network is a specific type of ML algorithm, consisting of layers of interconnected neurons that process input data and generate an output, learning from previous experience in the form of training data. Figure 2.4 shows an example of a fully connected neural network with five sequential layers. Fully connected means that every neuron in one layer are connected to all neurons in the adjacent layer through *weights*.

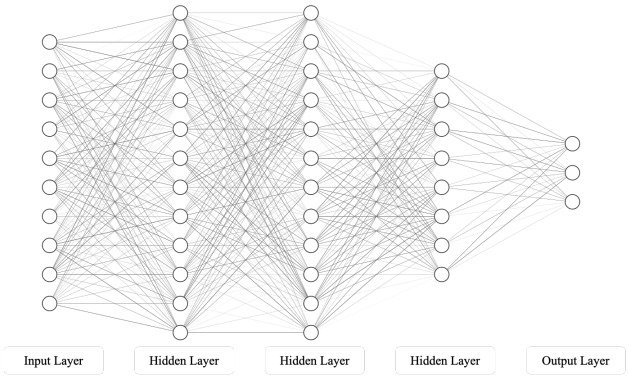


Figure 2.4: Fully connected neural network with five layers.

The output of a neural network is produced by feeding the network with training data which propagates through the layers from left to right. During this propagation, the input data to the network is transformed by weights and *activation functions* being applied, allowing the network to learn complex patterns by introducing non-linearity [21, p. 62-63]. Apart from weights, there is also a *bias* term b for each neuron in a layer allowing the activation function to be shifted by an offset b, giving better flexibility of fitting the training data [24, p. 4][25].

Networks of greater depth are referred to as DNNs or Deep Learning algorithms, which are capable to learn input-output mappings by building complex representations expressed in other simpler representations [26].

2.6.1 Training a Neural Network

Collected data is split into *train*, *validation* and *test* datasets, with training and validation data used for training the neural network. Prior to using collected data for training the model[1], data normalization is applied as a pre-processing technique to enhance model performance. Normalizing data ensures that the magnitude of different features in the data are of similar scale, which is beneficial for the model to learn and helps accelerating training [27]. After training, the neural network is given test data to see how it performs on previously unseen data.

During training *epochs*, the validation data is used to detect if the network is being under- or overfitted to the training data. The term epoch is a hyperparameter[2] that defines the number of times that the learning algorithm will go through the entire training dataset. Another hyperparameter is the *batch size*, which determines the number of training samples to use before updating the internal network parameters – the weights and biases. The batch size has shown to have impact on generalization ability [28].

The network learns by updating its network parameters, with the aim of approximating an output for all training inputs. A cost function $C(\mathbf{w}, \mathbf{b})$ is defined to measure the error between predicted and target output values in the training process [24, p. 17-20]. For instance, the RMSE metric could be used to measures how well the parameters are adjusted to achieve a minimal error for the training data. Moreover, gradient descent is an algorithm for minimizing $C(\mathbf{w}, \mathbf{b})$ using the gradient of the cost function, computed by a backpropagation algorithm [24, p. 39]. The gradient vector ∇C contains partial derivatives of C with regard to all depending variables, n weights and m biases.

$$\nabla C = \left(\frac{\partial C}{\partial w_1}, \ldots, \frac{\partial C}{\partial w_n}, \frac{\partial C}{\partial b_1}, \ldots, \frac{\partial C}{\partial b_m} \right) \tag{2.3}$$

Consequently, C changes according to $\Delta C \approx \nabla C \cdot \Delta \mathbf{w}$, where $\Delta \mathbf{w}$ is a transposed vector with the amount of change for each network parameter.

In each iteration of the algorithm, $\Delta \mathbf{w}$ should be chosen such that $\Delta C < 0$ to descend the cost function. In this fashion, the network parameters are updated until a minima of C is found. The cost function is sometimes also called loss function[3], which is the term used in this thesis. Readers interested in learning about the gradient descent algorithm in more detail are referred to [24, p. 16-22] by Michael Nielsen.

2.6.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) use an architecture making them well suited for multidimensional input and are commonly used for various image recognition tasks. A task could be to classify images of handwritten digits. CNNs take

[1] Model and network is used interchangeably in the context of neural networks.

[2] A hyperparameter specifies details for the training process, in contrast to parameters that define the model itself.

[3] Loss function is usually defined to measure the error on a single data point, while cost function is the average of loss functions across an entire dataset.

advantage of local receptive fields, pooling and parameter sharing [24, p. 170]. The architecture is comprised of stacked convolutional layers, pooling layers and fully connected layers [29], see Figure 2.5.

A key difference with CNNs compared to the fully connected network in Figure 2.4 is that there is no longer a one-to-one correspondence between layers. Instead, a small region called local receptive field of the input will be connected to a single neuron in the following hidden layer.

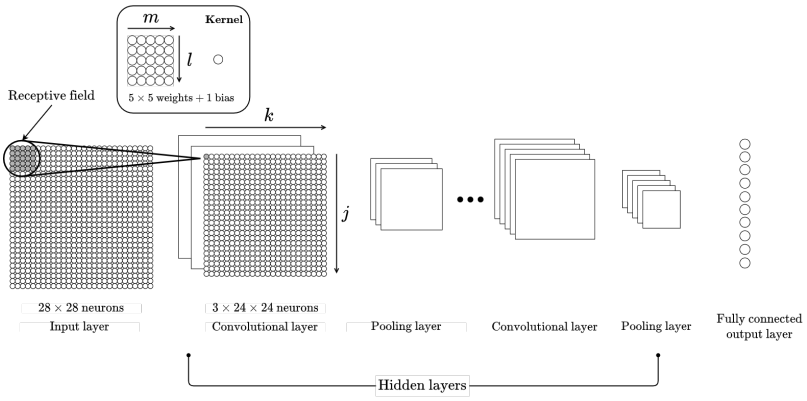


Figure 2.5: Illustrated CNN architecture with receptive field and kernel.

For receptive fields of size 5×5, as in Figure 2.5, each hidden neuron will be assigned a bias term and 25 weights connected to its local receptive field. This set of parameters will be the same for all hidden neurons, called parameter sharing and is often defined as a kernel [24, p. 148-150].

The receptive field is applied using a sliding window to iterate the entire input from left to right, top to bottom. For each position of the local receptive field, the output of the hidden neuron at j, k is determined by Equation 2.4 where σ is the activation function [24, p. 170-172][30, p. 150].

$$\sigma\left(b + \sum_{l=0}^{4}\sum_{m=0}^{4} w_{l,m} \cdot x_{j+l,k+m}\right) \tag{2.4}$$

Input to σ is the shared bias b, shared weights $w_{l,m}$ and $x_{j+l,k+m}$ denoting the input activation at position $j + l, k + m$.

A convolutional layer is usually followed by an immediate pooling layer that reduces the dimensionality of the convolutional output of hidden neurons. Max-pooling is a common pooling method that summarizes a small region of neurons by using the maximum value within the local region. Lastly, the pooled layer is flattened into a fully connected layer, represented by the vertically aligned neurons in Figure 2.5. For this example there are 10 neurons in the output layer, corresponding to the possible output classes for digit classification (0-9). The neuron with highest value is chosen as the predicted output class.

2.6.3 Residual Network

An issue with CNNs is the so-called vanishing gradient problem, arising during backpropagation while training deep networks, increasing with the network depth. What happens is that the gradient in Equation 2.3 gets too small and is gradually diminished [31]. To mitigate this problem, the Residual Network (ResNet) architecture was introduced in [32] as a special form of a CNN.

The ResNet architecture allows for building much deeper networks by using *residual skip connections*. These are used to bypass one or more layers, adding the input of a previous layer to the output of a later layer. A deeper network makes it feasible to learn more complex patterns in the training data.

Instead of learning a desired mapping $\mathcal{H}(\mathbf{x})$, the layers approximate a residual function $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$ such that the original mapping becomes $\mathcal{F}(\mathbf{x}) + \mathbf{x}$. Figure 2.6 illustrates a small part of a ResNet with a skip connection bypassing two convolutional layers. This neither adds extra network parameters nor computational complexity [32].

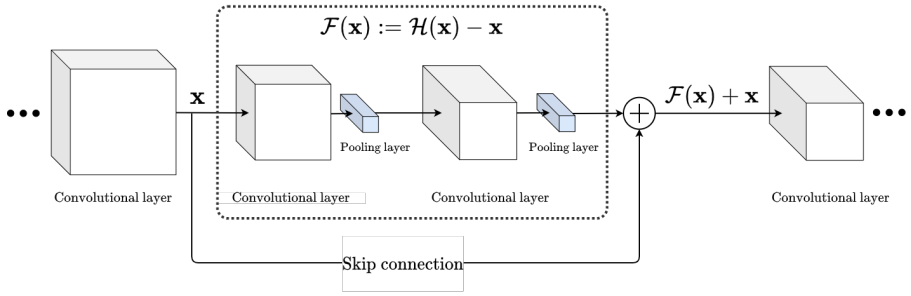


Figure 2.6: Illustration of a skip connection in ResNet.

Chapter 3
Methodology

The forthcoming sections in this chapter goes into further detail about the workflow, describing the different tools and their roles in the chain of work. Following the general description of the workflow steps, details on sparse and dense factory scenarios are given, respectively. In these sections, the used AI/ML models are described with high-level structures of how collected data is used. The final section of this chapter describes how real data measurements have been collected and processed for AI/ML training.

3.1 Workflow

The indoor environments that specifically are of interest for this project are factories conforming to 3GPP's specifications on indoor factory small halls [9, p. 85]. The factory environments are generated by a version of Nvidia Omniverse[1], used internally at Ericsson. Henceforth, this tool is referred to as factory generator. The studied factory environments are categorized as sparse and dense scenarios. Figure 3.1 shows a schematic overview of the workflow.

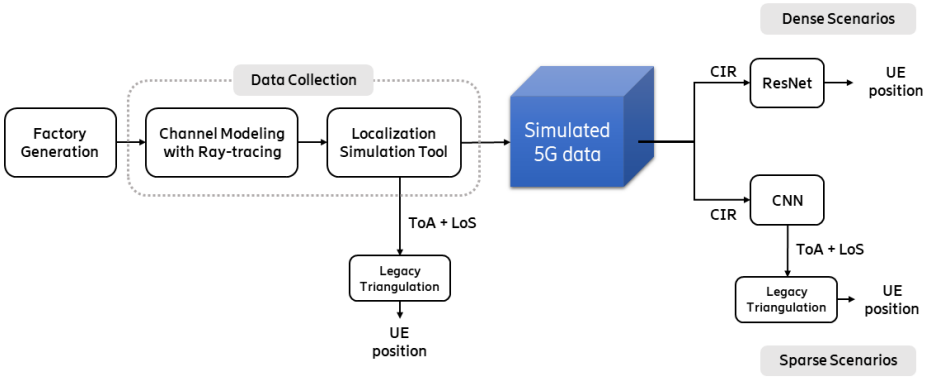


Figure 3.1: Overview of workflow.

[1]Nvidia Omniverse is a real-time 3D graphics tool used to build 3D environments etc.

Initially two baseline factories are generated using the factory generator, a sparse and dense scenario, respectively. Modified factory scenarios are generated by introducing different types of changes to the baseline factories. The purpose is to see if a model trained on a baseline factory can generalize to a modified factory scenario, as described in Section 1.3. For evaluating a model's generalization in terms of robustness to changes in the factory environment, the model is trained on data from a baseline factory and then evaluated by testing the model on test data collected from modified factory environments. The training of a model encompasses the usual training procedure for neural networks, as described in Section 2.6.1.

The collection of simulated 5G data starts by calculating the radio propagation paths using an Ericsson state-of-the-art 3D ray-tracing channel model. The ray-tracing is simulated for an individual factory scenario to create a site-specific channel model. Then, by running an internal localization simulation tool, data for positioning with 5G is obtained. This workflow is iterated to collect data for the baselines and modified factory scenarios.

The localization simulation tool is also used to calculate positioning errors using a legacy solution based on triangulation with ToA. The legacy positioning results are to be used as a performance baseline when comparing the performance of AI/ML based solutions.

The mentioned tools above existed prior to our thesis work. However, training AI/ML models for indoor positioning using simulated data collected with site-specific channel models through this chain of tools has not been extensively explored previously, as described in Sections 1.2 and 1.3.

3.2 Factory Generation

In order to generate representative 3D models of factories, a commercial simulator from Nvidia named Omniverse is used with an Ericsson extension [33]. All 3D objects are modeled with realistic surface properties depending on material, which is of importance when performing ray-tracing simulations as described in Section 3.3.1. Factory beams and pillars and machines are all metallic. The walls, ceiling and floor are of concrete. The generated 3D factory is later used as input to the ray-tracer, representing the factory environment. Ericsson's internal version of Omniverse makes it feasible to create factory environments resulting in site-specific channel models, rather than using statistical channel models from 3GPP.

When generating the factory environments, a set of parameters are tuned to change the outcome of the generation. The parameters in direct connection to the factory environment layout are set in the factory generator GUI. Setting parameters for the node deployment is done by editing the code base prior to deploying nodes via the GUI. The factory environments have dimensions $60 \times 120 \times 10$ meters and have realistic machines and equipment placed in them. The factory is partitioned into an *assembly* and *production* area, see Figure 3.2.

Furthermore, a parameter can be set to adjust the density of machines and equipment, also referred to as clutters, in the factory. Fewer machines and equipment yields a factory with a lower clutter density. There is also a possibility to add robots to the factory floor. These robots are automated guided vehicles (AGVs) and forklifts.

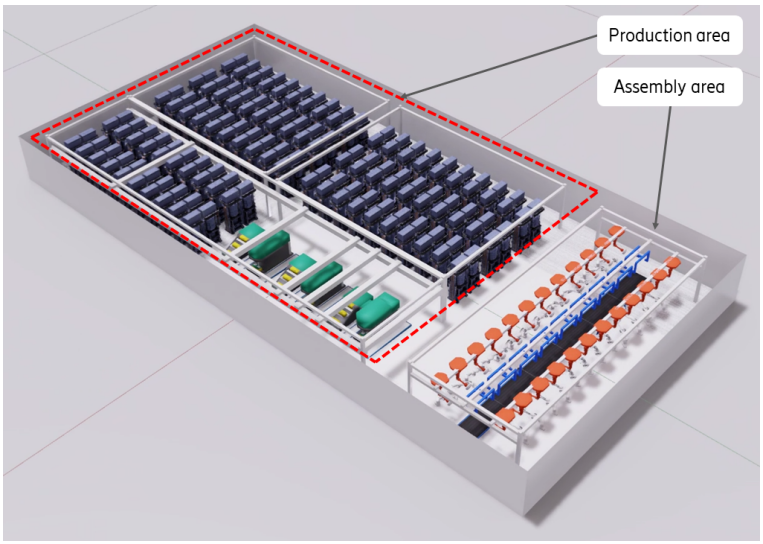


Figure 3.2: Generated factory example with marked factory areas.

3.2.1 Baseline Factories

To generate factory environments with different LoS conditions – sparse and dense scenarios – the parameter settings in the factory generator are adjusted appropriately. For the sparse scenario, a LoS probability around 50% is desired and 15% is desired for the dense scenario. The parameters for generating the baseline factories are stated below.

Sparse factory scenario

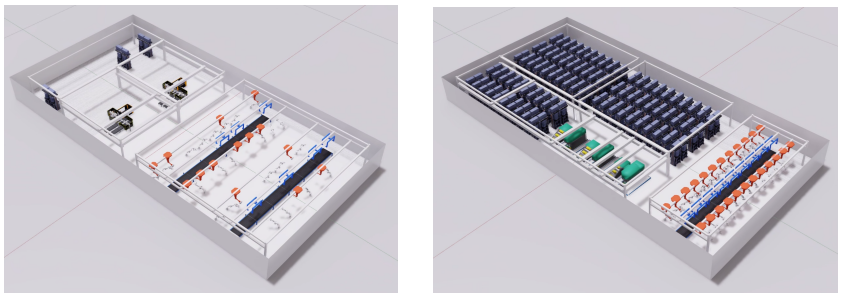
- Assembly area coverage: 50%
- Production area coverage: 50%
- Clutter density: 50%

Dense factory scenario

- Assembly area coverage: 20%
- Production area coverage: 80%
- Clutter density: 80%

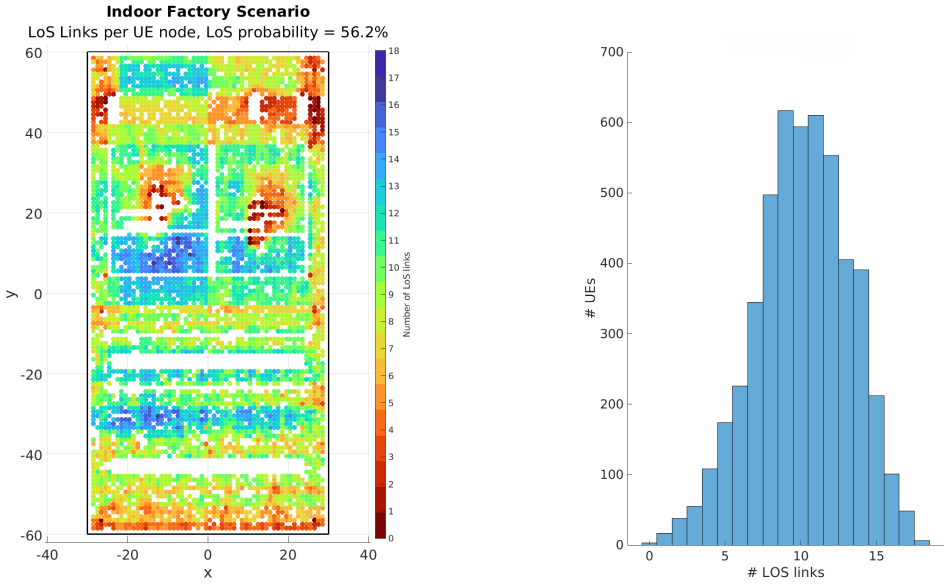
The listed parameters for sparse and dense factory scenarios result in the factories seen in Figure 3.3a and 3.3b, respectively. Figure 3.4 and 3.5 depict the corresponding LoS heatmaps and histograms, confirming that the LoS conditions are drastically different for the two scenarios. The LoS heatmaps display the scenarios' LoS probability and how LoS conditions vary depending on where in the factory a UE node is placed.

By increasing the clutter density to 80% and allocating a larger partition for the production area, the LoS property is successfully decreased for the dense scenario. The sparse scenario's LoS probability is 56%, which is decreased to 14% for the dense scenario. The calculation of a scenario's LoS probability is described in Section 3.3.2.



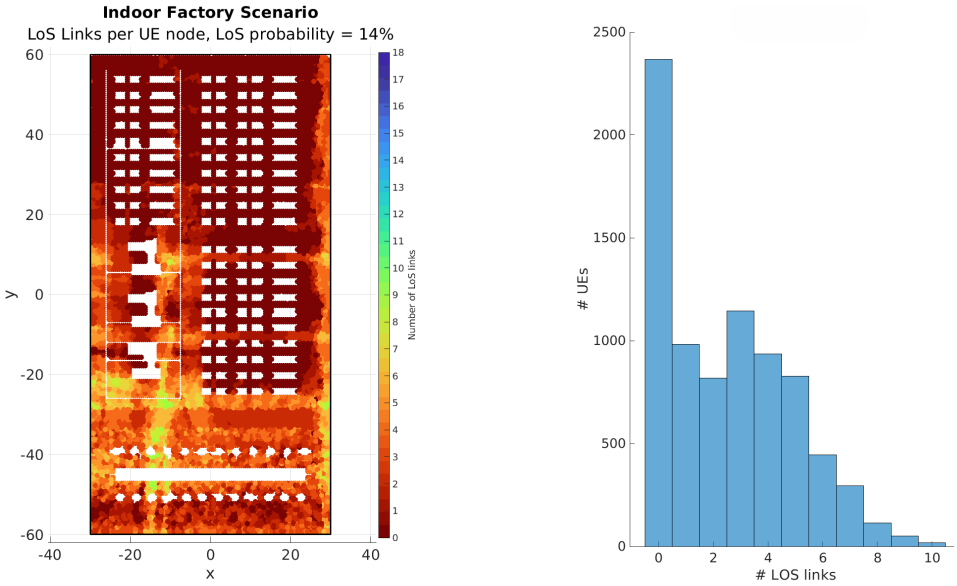
(a) Sparse baseline factory. **(b)** Dense baseline factory.

Figure 3.3: Baseline factories.



(a) LoS heatmap. **(b)** LoS histogram.

Figure 3.4: LoS overview for sparse baseline factory.



(a) LoS heatmap. **(b)** LoS histogram.

Figure 3.5: LoS overview for dense baseline factory.

3.2.2 Node Deployment in Factory

With a generated factory environment, UE and TRP nodes are deployed to simulate radio propagation from transmitter to receiver [33, p. 34]. For the scenarios studied, there are 18 TRPs attached to the ceiling 10 meters above the floor, see Figure 3.6. The number of deployed UE nodes vary depending on available floor space and the distance between them, d_{UE}. For sparse factory scenarios, $d_{\text{UE}} = 1$ meter while for dense scenarios is decreased to $d_{\text{UE}} = 0.75$ meter. This is done by editing the code base. The reason for decreasing d_{UE} is to deploy UEs tighter since the available floor space has been reduced due to higher clutter density in the factory. The nodes are uniformly placed across the entire factory floor at 1.5 meters height, except for locations occupied by already placed machines or robots.

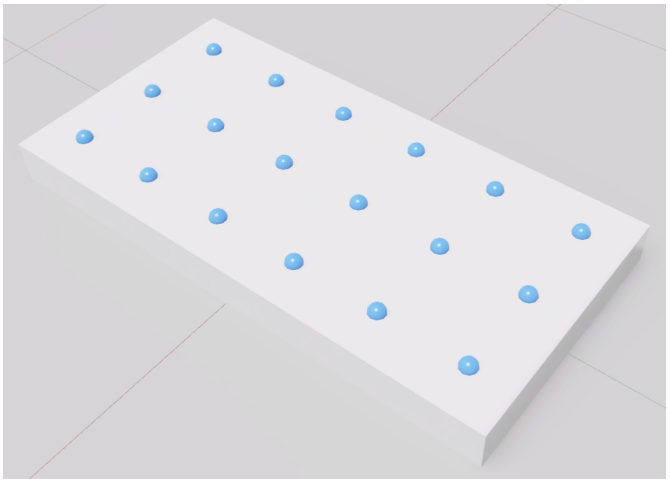


Figure 3.6: TRP node deployment.

In addition to the original UE deployment, with wall offset by one d_{UE} for both x and y, three other shifted versions are deployed. For instance, the shifted deployment in y assigns the nodes a wall offset by $\frac{3}{2}d_{\text{UE}}$ with regards to the y axis, see Figure 3.7. The wall offset is also edited in the code base. This method of shifting the UE node deployment allows for running more advanced ray-tracing calculations, enriching the resulting channel model without causing memory issues due to a too large number of deployed nodes in the same run. This might seem counter-intuitive for a state-of-the-art ray-tracer, however it is not designed to simulate such large number of nodes that these deployments encompasses.

The alternative would be setting a lower d_{UE}, yielding more UE nodes and correspondingly more training data, but limiting the simulations, see Section 3.3.1. To successfully run the ray-tracer with a large number of deployed nodes, the tracing parameters for multipath propagation modeling would require adjustment. Thus resulting in a less detailed channel model. See Table 3.1 for used tracing parameters. For this reason, the ray-tracing simulations are run for different shifted node deployments with fewer nodes in each simulation.

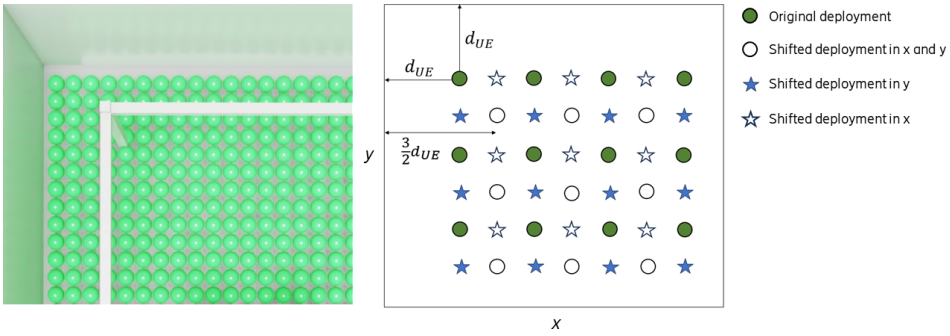


Figure 3.7: Original UE node deployment from factory generator with $d_{\text{UE}} = 1\text{m}$ and schematic deployment with shifted wall offsets.

3.2.3 Modifications to Baseline Factories

To test the ability of the models to generalize to changes in the factory environment, different types of changes are introduced to the baseline factory.

Descriptions of different types of factory modifications are given below, providing a sense of what has physically changed from the baseline factory. However, the descriptions lack a quantifiable metric to compare different changes to each other. In Section 3.2.4, a method for quantifying the amount of change is described. For an overview of the different modified scenarios, see Appendix A and B.

- **Added robots**: For this modification, AGVs and forklifts are added to the factory floor. This type of modification is the most subtle change since the factory layout is unchanged. Nonetheless, this type of change is likely to be the most frequent and common variation taking place in a real factory environment. Hence, it is of interest to see if the AI/ML models generalize to such changes.

  Moreover, varying setups of added robots have been introduced. Adding different number of robots (12 or 20), and adding robots exclusively in the assembly or production area, respectively.

- **Moved machines**: This type of modification physically moves objects in the factory. Individual or groups of machines are moved in the production area, whereas the assembly area is moved in its entirety. The moved distance is different depending on whether the scenario is sparse or dense since available floor space is limited in the dense scenarios.

- **Rotated machines**: In contrast to changing the location of machines, this modification rotates machines both in the production and assembly area. The machines are rotated more in the sparse scenario modifications than in the dense scenario modifications, since more room for rotation is possible in the sparse scenario.

- **Flipped factory layout**: The factory areas are rearranged such that the production and assembly area switches location.

- **Different factory layout**: This scenario does not originate from the baseline factory, instead a completely new factory is generated with 40% assembly area and 60% production area.

3.2.4 Quantifying Factory Changes

For quantifying the amount of change that a modified factory scenario has introduced compared to the baseline factory, a method using Voronoi diagrams as input images to compute the Jaccard Similarity Index (JSI) has been developed [34][35][36][37, p. 3-9].

The JSI is computed according to Equation 3.1. This is done using MATLAB's built-in `jaccard` function, which is used for image segmentation of two binary images, A and B. Taking the intersection divided by the union quantifies how many pixels in the images are similar. In [35] a similar method has been used to compare images of source code and in [36], the JSI is used for image matching. As the JSI value decreases, the modified factory scenario becomes less similar to the baseline.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \in [0,1] \tag{3.1}$$

The binary images, A and B, are generated by plotting Voronoi diagrams for the deployed UE nodes for the baseline and a modified factory, respectively. Such a Voronoi diagram will enclose all the UE positions in regions shaped as polygons with equidistant boundaries between two or more data points.

Consequently, by processing the Voronoi diagram, it is possible to set aside regions whose area exceeds a tuned threshold, and thus capture factory areas with no deployed UE nodes. The areas with no deployed UE nodes correspond to factory areas occupied by machines and other equipment. With these areas highlighted in black, changes in a modified factory are efficiently distinguishable. Figure 3.8 shows a processed Voronoi diagram for a baseline and modified factory scenario, with segmentation differences highlighted.

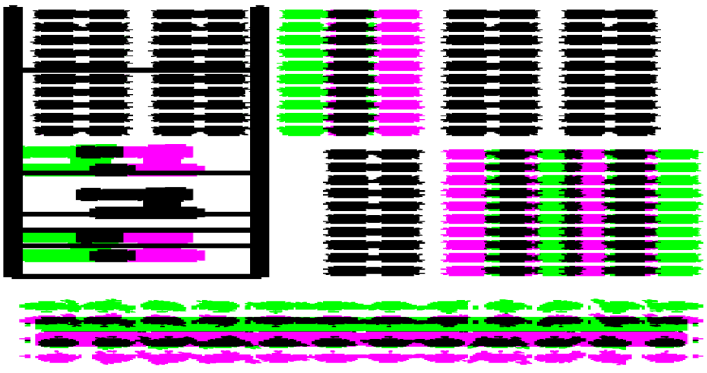


Figure 3.8: Processed Voronoi diagram of baseline factory and factory with moved machines, where segmentation differences are highlighted.

3.3 Data Collection Through Simulations

For a generated factory environment with deployed UE nodes, training data is collected for the deployed nodes by running ray-tracing and localization simulations, as the following sections describe.

3.3.1 Channel Modeling with Ray-tracing

The ray-tracing is done using Ericsson's 3D ray-tracing channel model, which takes a scenario environment and node deployment as input, coming from the factory generator. The ray-tracer traces the propagation paths between every UE and TRP antenna, and applies a propagation model to output link information and a channel model. The UEs are simulated to transmit with carrier frequency 3.5 GHz, within frequency range 1. This tool makes it possible to model the communication channels between UE and TRP in the generated factory environments, by simulating multipath propagation as described in Section 2.2.

The aim with the ray-tracing simulations is to retrieve site-specific channel characteristics of the radio propagation for the generated factory environments. The physical properties of 3D objects in the factory are taken into account to get correct behavior of reflections and scattering interactions.

How detailed the channel model can get depends on, as mentioned in Section 3.2.2, the number of deployed nodes, but also the scenario. With a more cluttered scenario, more interactions arise thus leading to a more computationally heavy simulation for the ray-tracer. Table 3.1 lists parameters used for ray-tracing simulations in factories with sparse and dense clutter. Max paths per link refers to the number of stored links between UE and TRP.

Table 3.1: Ray-tracing parameters for factories with sparse and dense clutter.

Parameter	Sparse clutter	Dense clutter
Carrier frequency	3.5 GHz	3.5 GHz
Max specular reflections	6	4
Max diffuse scattering	1	1
Max total interactions	7	5
Max paths per link	100	50

3.3.2 Localization Simulation Tool

In this step, the data which is to be used for training the two AI/ML models is obtained through simulations in a MATLAB based simulator. Obtained data from simulations includes CIR, ToA and LoS. The true UE positions are obtained during the factory generation and node deployment in Section 3.2.2. Furthermore, positioning errors using the legacy triangulation solution based on ToA, is computed in this simulator.

To get a more realistic channel model, additive white Gaussian noise (AWGN) is added to the simulated CIR measurements. The simulations are divided into a number of so-called *drops*. In each drop, $n = 1000$ non-overlapping UE nodes are randomly selected, meaning that no UE node is selected more than once. This is of importance when it comes to training and testing of the ML models, since all files in a dataset will contain diverse node deployments as opposed to just having nodes from a certain subarea of the factory floor.

Moreover, the scenario LoS probability is calculated and a script has been developed to plot a heatmap visualizing the number of LoS links per UE node in the factory, see Figures 3.4a and 3.5a. For the factory environments that this project encompasses, the scenario LoS probability is calculated based on Boolean matrices. Each row in such a matrix, M, corresponds to a UE node and the columns represents the 18 different TRPs. That is, UE node at row i has a value 0 or 1 in each column j depending on if it has LoS to the TRP at column j or not. Therefore, $M_{ij} = 1$ means that the UE node at row i has LoS to the j:th TRP. The Boolean flags in M are determined according to Equation 3.2,

$$M_{ij} = \begin{cases} 1 & \text{if } |\tau_{ij} - d_{ij}| \leq T \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

where τ_{ij} is the first path delay for a signal sent from UE i to TRP j, converted from time to distance domain with meters as unit. The threshold T is a parameter chosen in the simulator. The geometrical distance between UE i and TRP j is denoted d_{ij}. The LoS probability, P_{LoS}, is then calculated as in Equation 3.3.

$$P_{\text{LoS}} = \frac{1}{18n} \sum_{i=1}^{n} \sum_{j=1}^{18} M_{ij} \tag{3.3}$$

The LoS data used for training is determined by Equation 3.2 and is considered to be the ground truth since it depends on the true UE position. The ground truth ToA is determined as the geometrical distance converted to time, which is only valid for LoS links.

For the legacy triangulation solution, LoS is determined using hard classification based on if the first peak in the CIR also is the strongest peak. The ToA is estimated as the delay to the first peak in the CIR. It is also assumed that the UE was not positioned outside the factory and therefore, the x and y coordinates were bounded within the factory area.

3.3.3 Resulting Data

Retrieved data after localization simulation is processed by a script developed to extract the necessary attributes for AI/ML training. This is done on a per drop basis and the resulting files in MATLAB contain the data fields with shapes shown in Table 3.2. That is, every file contains $n = 1000$ samples corresponding to one drop.

Table 3.2: Resulting data from localization simulation tool with data shapes.

CIR	Observed ToA	LoS flags	UE position
(n, 18, 256, 2)	(n, 18)	(n, 18)	(n, 2)

Regarding the 4-dimensional CIR data, the first dimension represents the number of samples, which is equal to the number of UE nodes per drop (n transmitters). The second dimension corresponds to the number of TRPs (receivers). The third and fourth dimension represents the time instances and the communication channels for each TRP antenna, respectively. That is, each TRP has two receiving antenna ports, one for each polarization.

Figure 3.9 shows two samples of the collected CIR data. The two samples correspond to two UEs transmitting to a randomly chosen TRP. The transmitting UEs are chosen such that one has a LoS link (solid line) to the receiving TRP, while the other is a NLoS link (dashed line). Two markers are placed along the delay axis indicating when a transmitted signal would have arrived to the receiver if it propagated in a direct path, corresponding to a LoS link. Consequently, according to Equation 3.2, a NLoS link should have its marker (hollow) placed with an offset greater than T from its detected first peak. In Appendix C more sample CIR plots are shown.

Prior to using the CIR data for training the AI/ML models, it is normalized. Data normalization is a standard pre-processing technique to enhance model performance, with benefits described in Section 2.6.1.

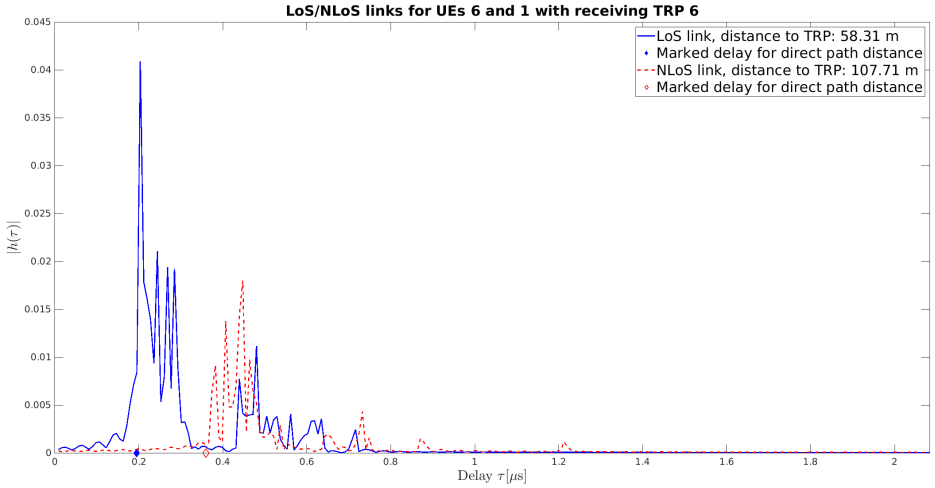


Figure 3.9: CIR data plotted as $|h(\tau)|$ for UEs 1 and 6 transmitting to TRP 6.

3.4 Sparse Factory Scenarios with CNN

Indoor positioning for sparse factory scenarios is studied with a CNN model. This model uses CIR data as input and outputs ToA delay estimation and LoS classification, see Table 3.3.

The intention of the CNN model is to achieve better ToA estimations and LoS classifications by learning patterns from the CIR data. This avoids further ToA and LoS calculations with varying degree of accuracy. Since the ground truth ToA is only valid for the LoS links, the NLoS links and corresponding ToA estimations are poorly estimated and therefore excluded in the triangulation. Based on this LoS classification and estimated ToA delay from the model output, a conventional triangulation algorithm is used to estimate UE position. The CNN model also had boundaries and only predicted the UE to be inside the factory area.

Table 3.3: Input and output data shapes for CNN model.

Input	Output	
CIR	ToA estimation	LoS classification
$(18n, 256, 2)$	$(18n, 1)$	$(18n, 1)$

The ToA training data includes a few undefined `NaN` values caused by total blockage between UE and TRP. To handle such cases, the ToA is set to 0 and corresponding link is classified as NLoS.

The input CIR data is of shape $(18n, 256, 2)$, with the number of transmitting UEs (n) and TRPs (18) flattened into a joint dimension. Hence, each row of the dataset represents the CIR for one instance of a transmitter to receiver link, each with two receiving TRP antenna ports. A visualization of this can be seen in

Figure 3.10, where each of the $18n$ rows is represented as an input image to the CNN with 256×2 pixels. Each pixel corresponds to a complex CIR value at some delay instance $1 \leq \tau \leq 256$ for one of two receiving antenna ports.

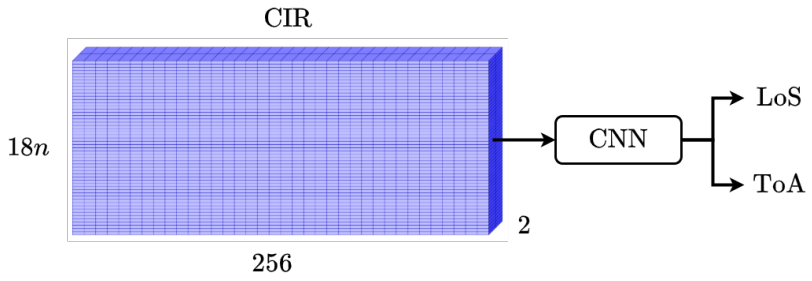


Figure 3.10: Visualization of input to CNN model.

3.4.1 Data Splitting and Training CNN

Training the CNN model for sparse factory scenarios is done by training on data collected from the sparse baseline factory in Figure 3.3a. Data collection is performed according to Section 3.3 and results in $324n$ samples, out of which $72n$, are assigned as *test* data for the baseline. For the remaining $252n$ samples, 10% is set aside as *validation* and 90% as *training* data.

In order to get the optimal solution, a series of different batch sizes and number of training epochs have been tested. This showed that a batch size of 500 and 300 epochs was the best choice as it gave the best prediction performance for validation data, and prevented the model from being overfitted. Extending the number of training epochs resulted in the validation ToA delay error abruptly increasing towards the training loss curve, indicating that the model was overfitted, as described in Section 2.6.1.

The resulting training plot is seen in Figure 3.11. This plot shows the training loss, LoS accuracy and delay for LoS links, evaluated with the validation dataset.

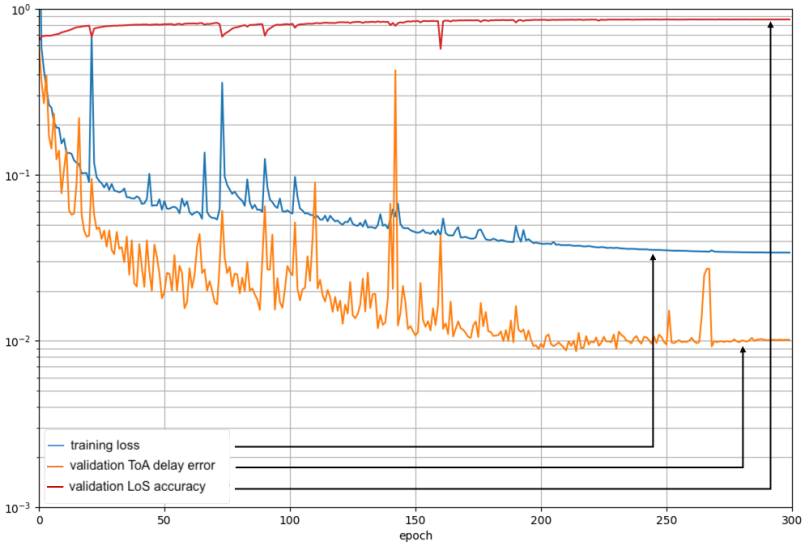


Figure 3.11: Training loss and validation errors for CNN model.

3.5 Dense Factory Scenarios with ResNet

For the dense factory scenarios with heavy NLoS conditions, a ResNet model is used with a fingerprinting approach, which takes CIR data as input and directly outputs predicted UE positions, see Table 3.4. The reason for using a fingerprinting approach is due to poor LoS conditions, where a conventional triangulation method is not feasible since at least three TRPs with LoS links are needed, as described in Section 2.4.1.

Table 3.4: Input and output data shapes for ResNet model.

Input	Output
CIR	UE position
$(n, 256, 36)$	$(n, 2)$

Input CIR data to the neural network is reshaped such that each of the n rows in Figure 3.12 corresponds to one 2-dimensional input of size $(256, 36)$. The number of receivers (18) and antennas per receiver (2) are flattened into a joint dimension to get 36 combinations of receiving TRP and antenna port. The input can thus be seen as an image consisting of 256×36 pixels. Each pixel holds a value corresponding to the complex CIR, $h(\tau)$, at some delay instance $1 \leq \tau \leq 256$, for a receiving TRP and antenna port.

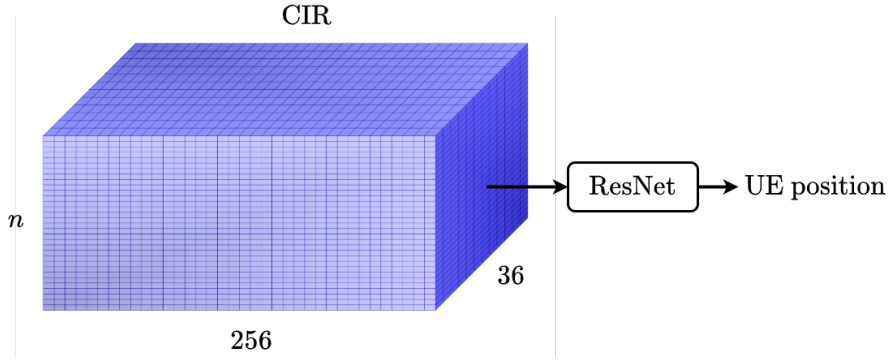


Figure 3.12: Visualization of input to ResNet model.

Figure 3.13 visualizes the norm of complex CIR values, $|h(\tau)|$, for one of n samples of size $(256, 36)$. The magnitude is represented with increasing pixel intensity. The model is trained to map such an input image – corresponding to 256 CIR values across 18 TRPs each with two antenna ports – to a specific UE position in the factory. This way of viewing the CIR data provides a deeper interpretation of how the model manages to learn patterns within the data, compared to inspecting a single CIR in isolation. More CIR plots in this view are found in Appendix C.

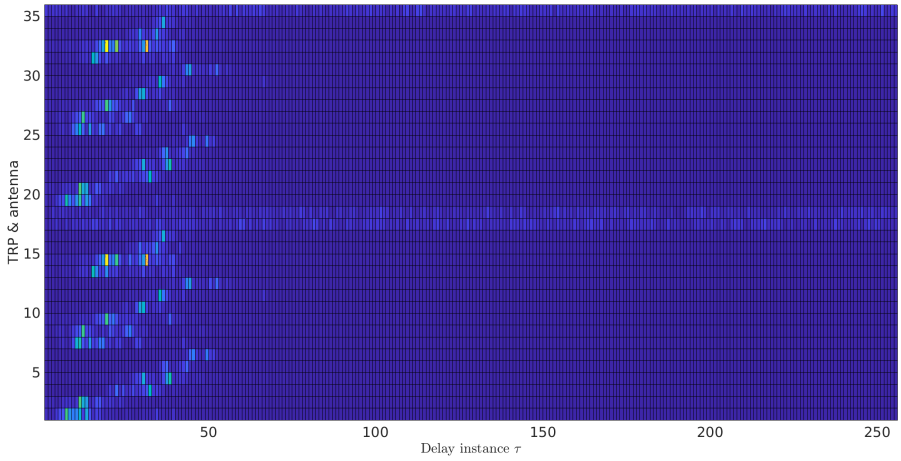


Figure 3.13: Visualized CIR input image of size 256×36.

3.5.1 Data Splitting and Training ResNet

The ResNet model was trained on data collected from the dense baseline factory, see Figure 3.3b. A total of $32n$ data samples were collected according to Section 3.3, with a 12.5% test split to leave out $4n$ samples as *test* data for the baseline. Furthermore, 6.25% is used as *validation* data. The remaining $26n$ samples are used as *training* data, denoted $\mathcal{T}$. Recall that $n = 1000$, as described in Section 3.3.2. Table 3.5 summarizes the splits.

Furthermore, data is collected from modified factory scenarios described in Section 3.2.3. Data from modified scenarios is mainly used as *test* datasets for a network trained on the baseline dataset $\mathcal{T}$. However, they are also used for mixed training, see Section 3.5.2.

Table 3.5: Summary of splits for training, validation and test datasets.

Training	Validation	Test
81.25%	6.25%	12.5%

To enhance model performance, tuning hyperparameters for the specific setup should be done. Therefore, a series of different batch sizes have been tested to fit the number of training samples employed for the model. Figure 3.14 shows the training loss for batch size 100 over 2000 epochs, which gave the best results.

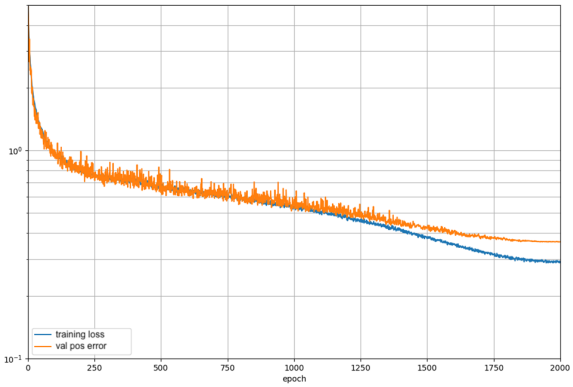


Figure 3.14: Training loss and validation positioning error with baseline dataset $\mathcal{T}$.

3.5.2 Mixed Training

Mixed training refers to that the model is trained on data from a mixed setup of factory scenarios, effectively increasing the number of training samples and forming a diverse training dataset. For this project, it was decided to include aggregated data collected from four scenarios with added robots deployed in the factory, in addition to data collected from the baseline scenario to form a mixed dataset $\mathcal{T}_{\text{mixed}}$. This increased number of training samples from $26n$ to $52n$. The inclusion of data from modified scenarios with added robots is due them being the most likely changes to a factory, which is a possible incentive to do this kind of mixed training.

Figure 3.15 depicts how data is aggregated and split for the mixed training. The horizontal dashed line separates data used for training and data used for testing the model. It might seem that the data from the modified factory scenarios with added robots, is used for both training and testing. However, the test sets used for testing has shifted the node deployment according to Section 3.2.2, indicated by the notes in Figure 3.15.

Figure 3.16 shows the training loss and positioning error for validation data, against the training epochs when training the ResNet model on dataset $\mathcal{T}_{\text{mixed}}$. Different batch sizes and number of epochs were tested, but remained unchanged from when training on dataset $\mathcal{T}$ in Section 3.5.1.

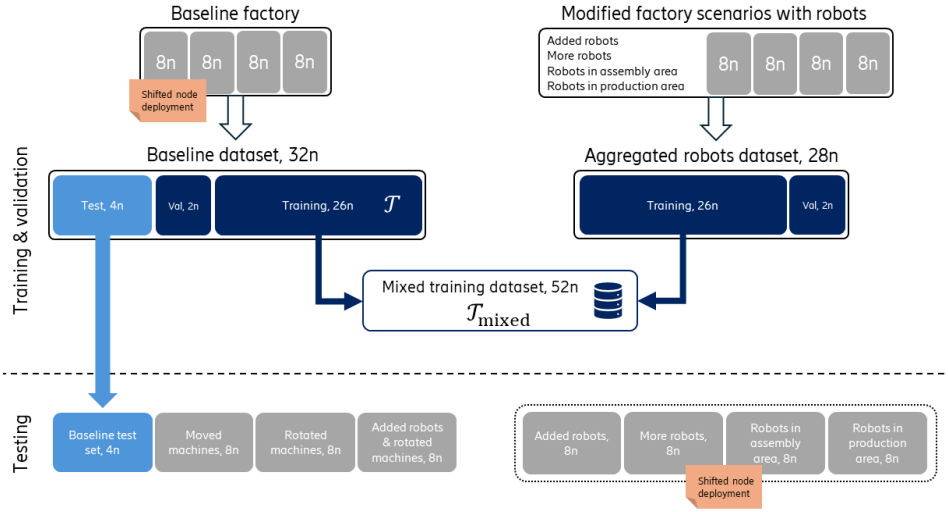


Figure 3.15: Illustration of data splitting for mixed training.

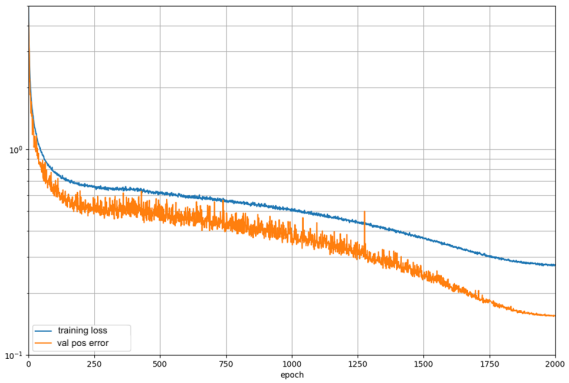


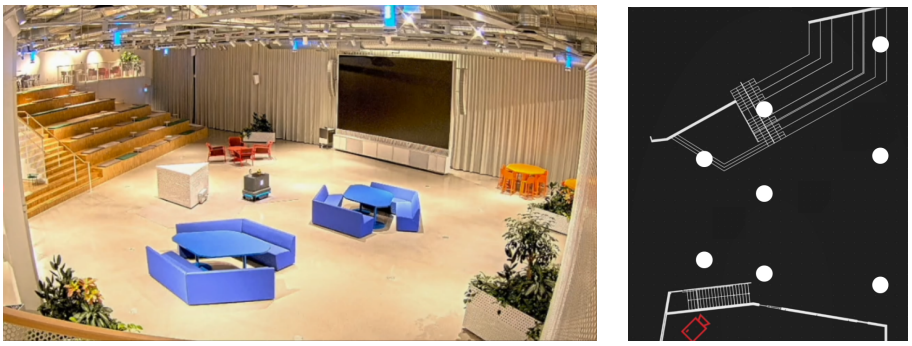
Figure 3.16: Training loss and validation positioning error using $\mathcal{T}_{\text{mixed}}$ dataset.

3.6 Real Data Measurements

For the factory scenarios in previous sections, 5G data has been collected *through simulations* with certain simplifications. Omni-directional antennas for both transmitter and receiver have been assumed, as well as limited multipath propagation and simulated noise. Therefore, one can ask how well the simulated environment and modeled channel also reflect the real world. To better understand this, real data measurements have been used to train and evaluate the ResNet fingerprinting model, and compared to the current legacy triangulation solution. The reason for not using the CNN model was because ground truth LoS and ToA are needed. Due time limitations, only the ResNet fingerprinting model was therefore used in this work.

A robot ran for three days during Mobile World Congress 2024 (MWC) with a maximum velocity of 0.2 m/s, following two different routes in an office environment with dimensions 28×35 meters, seen in Figure 3.17a. During MWC, the open-source software Robot Operating System (ROS) [38] was used to give navigational instructions to the robot, but also collect the true position of the robot.

To receive CIR measurements from the robot SRS transmissions, eight TRPs placed on the ceiling, see Figure 3.17b. This placement assured a high percentage of LoS between the robot and each TRP, thus leading to a sparse scenario.



(a) **(b)**

Figure 3.17: Office environment depicted in (a) with location of TRPs in (b).

3.6.1 Data Extraction

The real CIR data was received for 25 time instances over eight TRPs, each with four antenna ports. In comparison, simulated CIR data was measured for 256 time instances over 18 TRPs, each with two antenna ports as described in Section 3.3.3.

As ground truth positions for the ResNet fingerprinting model, the ROS software was used to collect the robot positions with a sampling rate of 10 samples per second. During periods of the day, the robot stood still. This led to the data being unevenly distributed and therefore, such measurements were omitted. To

achieve this, measurements were removed for periods when the robot was static for more than 20 consecutive samples, mitigating an uneven distribution.

In order to map the CIR captured by the LMF with the corresponding robot positions, the timestamps needed to be matched between the two systems. Since the timestamps were not synchronized, they were matched with seconds precision and trailing milliseconds were excluded from the timestamps. Consequently, all CIRs with the same timestamp were matched with the first robot position with that timestamp, leading to several CIRs for the same position. With CIRs and robot positions matched, the CIR data was parsed and reshaped conforming to ResNet model input.

For the legacy solution using triangulation, Ericsson's estimation of the positioning shown at MWC was used. This data had the estimated position of the robot timestamps. Since these timestamps were not synchronized with the timestamps of the robot positions, they were also matched the same way as for the CIR.

3.6.2 Training ResNet

The ResNet fingerprinting model used and described in Section 3.5 is also applied for the real data measurements. However, the input data shape is instead $(n, 25, 32)$ as seen in Figure 3.18. This is a consequence of the different measurement format described in Section 3.6.1.

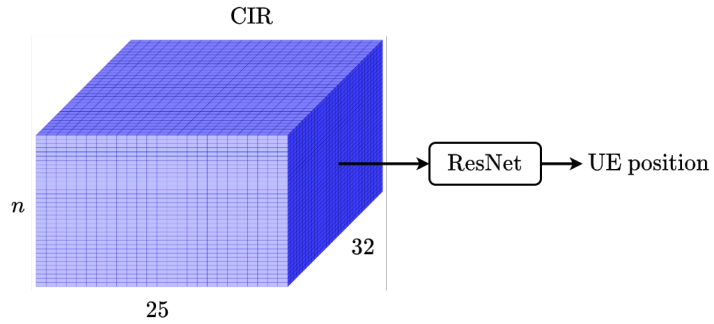


Figure 3.18: Visualization of real data input to ResNet model.

For training the ResNet fingerprinting model, $60n$ training samples were used. For validation, $8n$ samples were used and the model was tested on $4n$ samples. Different batch sizes were tested, showing that 400 gave the best result and was therefore chosen. The number of epochs was decreased from 2000 to 1250 to prevent the model from being overfitted. The training loss and the positioning error for the validation can be seen in Figure 3.19.

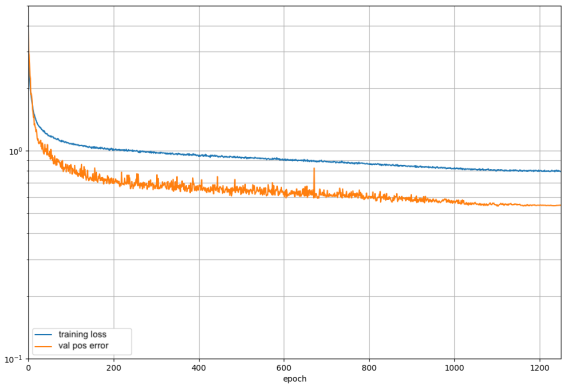


Figure 3.19: Training loss and validation errors for real data measurements.

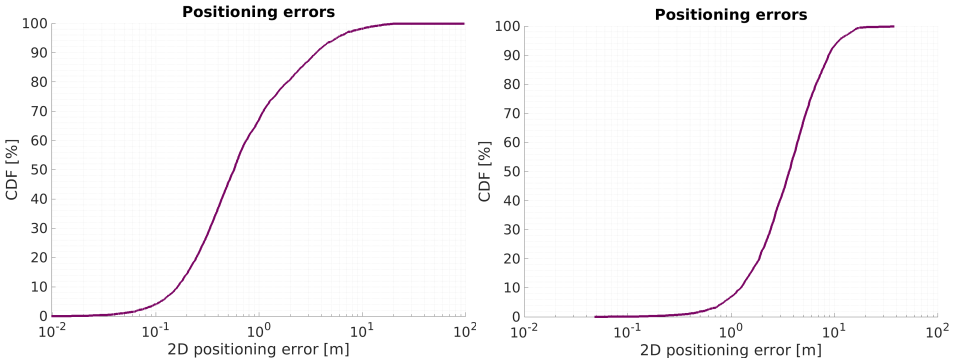
Chapter 4
Results

In this chapter, the positioning results are presented for the sparse and dense factory scenarios with simulated data as well as for an office environment using real data measurements. Initially, the results for the legacy solution using ToA based triangulation are presented, followed by the CNN and ResNet results, respectively.

The positioning errors are shown by the RMSE metric and four CDF percentiles, highlighting the 90% CDF positioning errors since reliability is needed. In Tables 4.3-4.5, the left-most column lists the test sets for different modified scenarios, described in Section 3.2.3. The second column lists the JSI value, indicating how similar the scenario is relative to the baseline factory. See Appendix A and B for an overview of the different modified scenarios.

4.1 Legacy Triangulation Solution

For the legacy solution, triangulation based on ToA is used to estimate UE position, as described in Section 2.4.1. Figure 4.1 shows the CDF positioning errors, for sparse and dense factory scenarios, respectively. In Table 4.1, a comprehensive view of the positioning errors is given.



(a) Sparse factory. **(b)** Dense factory.

Figure 4.1: CDF plots of legacy positioning results.

Table 4.1: Summarized test results for legacy solutions.

Scenario	CDF positioning error [m]				RMSE [m]
	50%	80%	90%	99%	
Sparse baseline	0.571	1.877	3.540	12.650	3.112
Dense baseline	3.620	6.610	8.747	15.967	5.718

4.2 Sparse Factory Scenarios with CNN

For the sparse scenarios, the CNN model produces LoS classifications and ToA estimations. The results can be seen in Table 4.2. Since triangulation only uses LoS links, the ToA estimations with NLoS are excluded in the triangulation algorithm. The ToA estimation error is converted from time to distance with meters as unit.

Table 4.2: Summarized LoS and ToA errors for CNN model with test sets from modified scenarios.

Scenario test set	LoS classification [%]						CDF ToA estimation error [m]			
	TP	FN	FP	TN	Accuracy	F1	50%	80%	90%	99%
Baseline	89.9	10.1	18.3	81.7	86.3	88.1	0.088	0.187	0.271	0.728
Rotated machines	90.1	9.9	19.2	80.8	85.9	87.5	0.088	0.185	0.268	0.721
Added robots (20) & moved machines	85.8	14.2	21.6	78.4	82.5	84.7	0.094	0.200	0.292	0.858
Flipped factory layout	83.2	16.8	23.0	77.0	80.9	84.4	0.099	0.221	0.343	10.689
Different factory layout	85.6	14.4	30.1	69.9	77.3	78.0	0.099	0.149	0.311	9.344

The positioning errors when using the LoS classifications and ToA estimations in Table 4.2 used for triangulation can be seen in Table 4.3. The CDF plot of the positioning errors can be seen in Figure 4.2. In Appendix A.1, the positioning errors are plotted in 3D space to see where errors occur in the sparse factory scenarios.

Table 4.3: Summarized positioning results for sparse scenarios.

Scenario test set	JSI	CDF positioning error [m]				RMSE [m]
		50%	80%	90%	99%	
Baseline	1.000	0.554	0.772	0.939	5.985	2.852
Rotated machines	0.954	0.574	0.807	1.000	5.069	1.444
Added robots (20) & moved machines	0.798	0.571	0.826	1.080	5.482	1.588
Flipped factory layout	0.761	0.563	0.779	0.964	6.335	2.081
Different factory layout	0.732	0.685	1.077	1.413	5.490	2.320

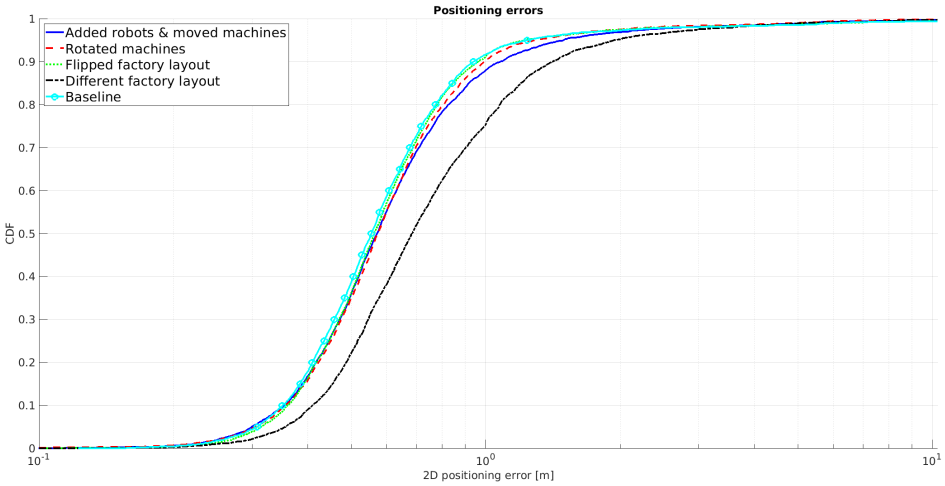


Figure 4.2: CDF plot of positioning errors for sparse factory scenarios.

4.3 Dense Factory Scenarios with ResNet

In this section, the positioning results for the dense factory scenarios are presented. These results are produced by the ResNet fingerprinting model. The results for different scenario test sets are summarized in Table 4.4, and Figure 4.3 shows the complete CDF plots. In Appendix B.1, the positioning errors are plotted in 3D space to see where errors occur in the dense factory scenarios.

Table 4.4: Summarized positioning results for ResNet model with test sets from modified scenarios.

Scenario test set	JSI	CDF positioning error [m]				RMSE [m]
		50%	80%	90%	99%	
Baseline	1.000	0.373	0.683	0.987	4.446	1.177
Added robots (12)	0.998	0.145	0.304	0.470	1.517	0.432
Robots in production area (12)	0.998	0.124	0.286	0.448	1.618	0.711
Robots in assembly area (12)	0.998	0.200	0.479	0.711	2.704	1.012
More robots (20)	0.997	0.235	0.528	0.771	2.931	0.858
Rotated machines	0.921	0.688	1.685	2.925	7.342	1.910
Added robots (12) & rotated machines	0.918	0.708	1.766	3.013	7.093	1.9334
Moved machines	0.857	1.967	3.952	5.521	11.221	3.483

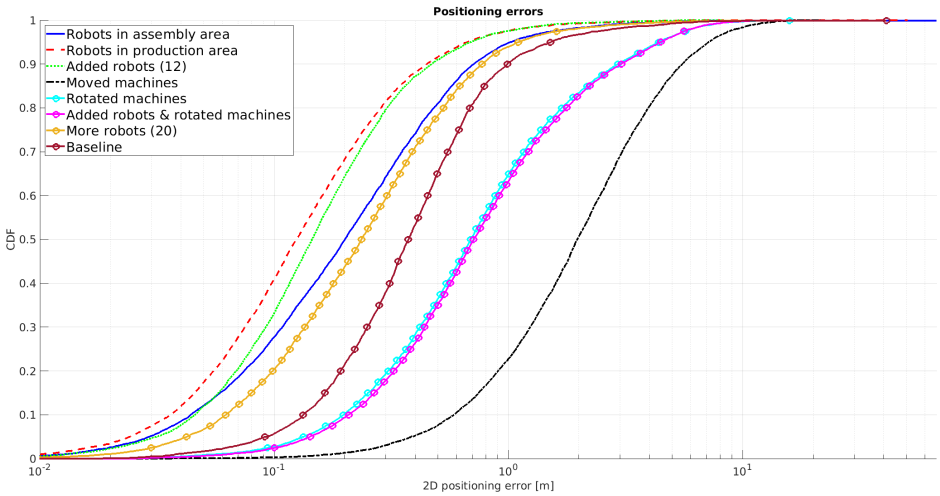


Figure 4.3: CDF plot of positioning errors for dense factory scenarios.

4.3.1 Mixed Training Results

The results using mixed training, as described in Section 3.5.2, are presented here. Table 4.5 summarizes the positioning results for different scenario test sets. Figure 4.4 shows the complete CDF plots for all the modified scenarios. In Appendix B.1.1, the 3D representations of the positioning errors are shown.

To clarify which scenarios the model has been trained on, a horizontal line is drawn separating these. The scenarios above this line are used for training the model and the scenarios below the line are used for testing. The JSI values for the aggregated scenarios in mixed training are left empty because the modified scenarios used for testing are only compared to the baseline.

Table 4.5: Summarized positioning results for ResNet model with mixed training.

Scenario test set	JSI	CDF positioning error [m]				RMSE [m]
		50%	80%	90%	99%	
Baseline	1.000	0.125	0.303	0.472	2.127	0.650
Added robots (12)	-	0.132	0.268	0.404	1.292	0.992
Robots in production area (12)	-	0.122	0.259	0.394	1.365	1.104
Robots in assembly area (12)	-	0.141	0.314	0.478	1.857	1.168
More robots (20)	-	0.170	0.358	0.525	1.890	0.936
Rotated machines	0.921	0.612	1.523	2.621	7.026	1.796
Added robots (12) & rotated machines	0.918	0.626	1.557	2.699	6.953	1.801
Moved machines	0.857	1.837	3.736	5.243	12.529	3.534

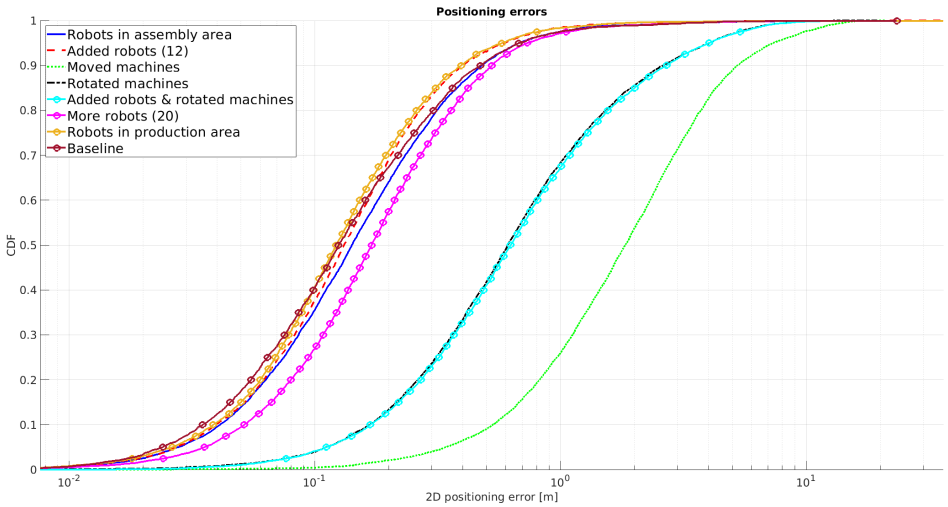


Figure 4.4: CDF plot of positioning errors for mixed training.

4.4 Real Data Measurements

The results from the real data measurements, described in Section 3.6, can be seen in Table 4.6 and Figure 4.5. The positioning errors are shown for a legacy solution and for the ResNet fingerprinting model.

Table 4.6: Summarized positioning results for real data measurements.

Positioning method	CDF positioning error [m]				RMSE [m]
	50%	80%	90%	99%	
Legacy	0.743	1.188	1.413	1.927	0.915
ResNet	0.324	0.695	0.969	4.462	1.032

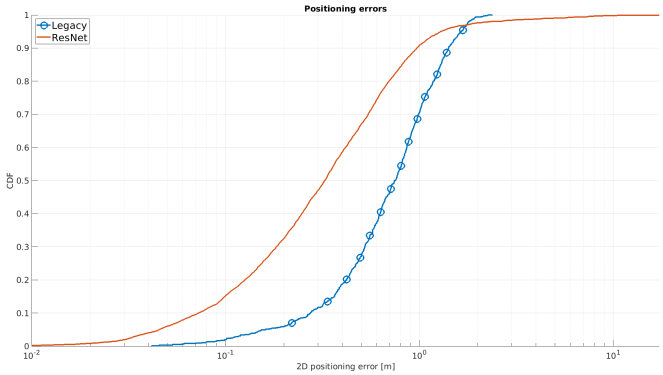
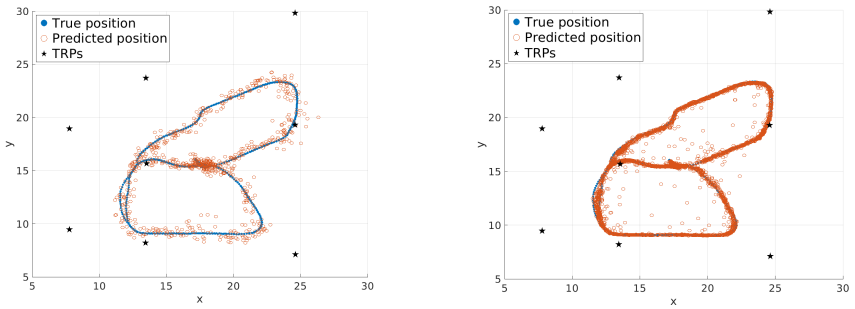


Figure 4.5: CDF plot of positioning errors for real data measurements.

In Figure 4.6, the true and predicted positions are shown for the test routes that the robot has traveled. The same test positions are used for the ResNet model and the legacy solution to get comparable results. The location of the TRPs can be seen in the same figure as stars.



(a) Legacy errors. **(b)** ResNet errors.

Figure 4.6: Positioning errors along test routes.

Chapter 5
Discussion

In this chapter the results presented in Chapter 4 are discussed with special consideration of the research questions stated in section 1.3.1, so that they can be answered in Chapter 6. The positioning errors discussed in the following sections refer to the error at 90% of the CDF, unless otherwise specified.

5.1 Sparse Factory Scenarios with CNN

The positioning estimates in the sparse scenario are affected by both the LoS classifications and ToA estimations presented in Table 4.2. An especially important part of the LoS classification is the FP, since it is the links with LoS that the triangulation is dependent on. If a link is falsely classified as LoS, that corresponding ToA estimation will have a large error.

When comparing the trained CNN model's positioning errors seen in Table 4.3 with the legacy positioning errors in Table 4.1 for the sparse baseline, the CNN model performs better. When comparing the CDF at 90%, the positioning accuracy is almost 73% better, which is a great improvement to the positioning.

Looking in Table 4.3, the RMSE for the baseline when using the CNN model gave a 2.85 m positioning error. This is a high value, considering the positioning error was below 1 m for the CDF at 90%. The reason for this is that RMSE magnifies larger positioning errors, as seen by the large peaks in Figure 5.1 visualizing the errors in a 3D plot. The majority of these peaks are located in the production area behind the tall machines. The heatmap seen in Figure 3.4a confirms that these are the locations where the UE has LoS to less than three TRPs, leading to inaccurate positioning using triangulation.

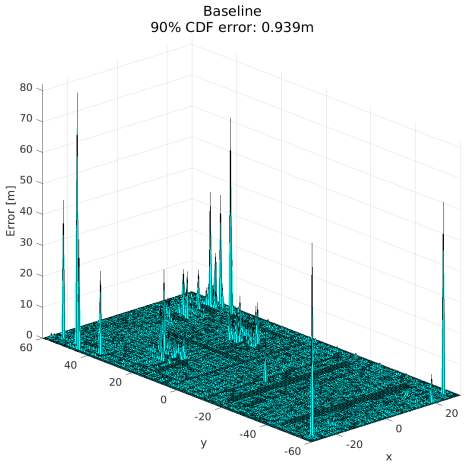


Figure 5.1: Positioning errors for baseline factory in 3D surface plot.

5.1.1 Generalization to Changes in the Factory

As seen in Table 4.3, the modifications *Rotated machines*, *Added robots & moved machines* and *Flipped scenario layout* gave a positioning error between 0.96 m and 1.08 m. This indicates that these changes to the environment do not affect the model significantly and that the positioning algorithm can be used without retraining the model.

When instead inspecting the RMSE for these modifications, these differs more than the positioning errors at 90% of the CDF. This is because of the positioning error peaks for the different modifications seen in Appendix A.1. These figures show that most of the factory area has low positioning errors, but some peaks can be seen locally where the modifications have taken place. Similar to the baseline scenario, the positioning error peaks can also be seen in areas where there are more NLoS links.

In the modification *Different factory layout*, a new factory was generated including a new machine and, a different layout of the production- and assembly area leading to a JSI of 0.732. The positioning error for this factory was 1.41 m, which is 60% worse compared to the baseline factory for the CDF at 90%. This is expected since different multipath propagation appears where a new machine is added and the scenario differs a lot compared to the baseline.

When looking at the 3D representation of the positioning errors in Figure A.4, one can see that the area with a new type of machine has higher positioning errors and more high peaks around the areas where the machines are located. Similar to the other scenarios, the positioning error peaks are located where a lot of changes have been made and where there are fewer LoS links. A difference from the other modifications is that it gives worse positioning in general, as seen in the CDF in Figure 4.2. This means that the CNN model struggles with accurate positioning even in open areas with more LoS. To solve this, the model likely needs to be retrained.

Even though the JSI for *Different factory layout* is relatively close to the JSI in scenario *Flipped factory layout*, the positioning error differs when looking at the CDF and RMSE. This indicates that only considering the JSI metric might not be sufficient for all cases. The JSI value is derived from an image analysis perspective and does not know what type of machines are used in the modification, in terms of height, shape, and materials. It just considers the occupied floor area. The addition of different types of machines alters the multipath propagation, resulting in a changed CIR, which might not be captured by the JSI value.

When inspecting the FP classifications in Table 4.2, it ranges from 18% to 30% for the different modifications and baseline. These are high values, considering the relatively good positioning results. A way to solve this could be to put a stricter condition for when a link should be classified as LoS. This leads to fewer links used in the triangulation, hence having too strict requirements could lead to difficulties in the triangulation algorithm too. The one with the worst FP classification was the *Different factory layout* scenario, which is expected since this scenario had the most changes. This modification also has the lowest F1-score and accuracy, which is reflected in the positioning errors.

5.2 Dense Factory Scenarios with ResNet

In Figure 4.1b, a CDF is plotted for the legacy positioning errors in a dense factory scenario with an error of approximately 8.75 m. In order to compare this to the ResNet results, we look in Table 4.3, where the positioning error for the baseline test set is 0.99 m with RMSE of 1.18 m. It is clear that the ResNet model achieves significantly improved positioning accuracy, going from 8.75 m to just below 1 m. This corresponds to a positioning error decreased by almost 89%.

Even looking at 99% of the CDF errors, the positioning error is around 4.4 m for the ResNet, compared to an error of nearly 16 m for the legacy solution. This corresponds to a 72% decrease. Figure 5.2 visualizes the positioning errors in a 3D plot, making it clear that most of the test positions give low errors. However, there are always outlier exceptions as seen by the large peak in this figure, which yields a larger RMSE metric.

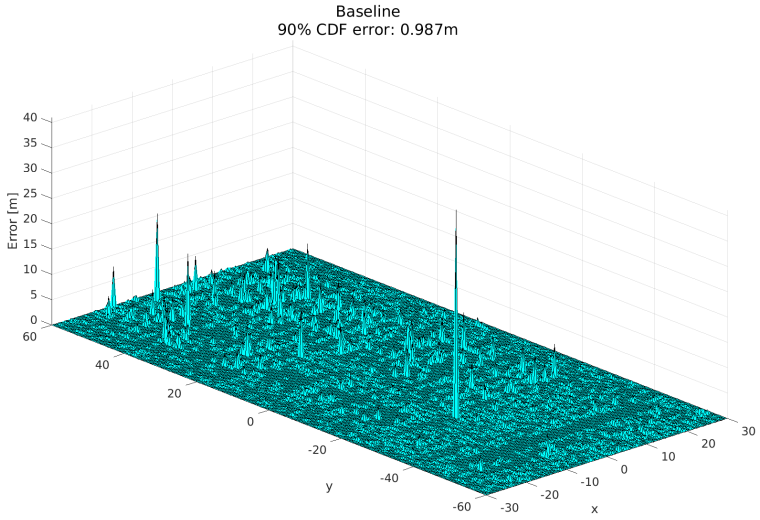


Figure 5.2: Positioning errors for baseline factory in 3D surface plot.

5.2.1 Generalization to Minor Changes in Factory

The type of changes are categorized as minor and major changes, depending on the JSI value. Since a fingerprinting model is used for the dense factory scenarios, it is expected to be sensitive to changes, and therefore modified scenarios with a JSI value below 0.95 will be denoted as major changes.

When introducing small changes such as adding robots to the factory floor with JSI values between 0.997 and 0.998, the positioning errors are all below 0.8 m. Moreover, the RMSE metrics are also improved compared to the baseline, indicating that there are no extreme outliers not being captured by the 90% CDF error. These minor changes are found in the first five rows of Table 4.3 and are

labeled as *Added robots (12)*, *More robots (20)*, *Robots in production area* and *Robots in assembly area*.

Surprisingly, the model performs better for test sets from these modified scenarios than for the baseline test set. That is, the model performs better for the unseen test data collected from the modified scenarios than for the baseline test set taken as a subset from the collected training data of the baseline factory. This is strange and unexpected, so we take a closer look.

The positioning error for the baseline test set is 0.99 m, while for twelve added robots, the error is just 0.47 m. With an additional eight robots added, the results go to 0.77 m. Somehow, positioning accuracy is enhanced by the added robots, but it does not improve further by adding more robots. This behavior could stem from the fact that these minor changes, being changed by only 0.2-0.3% compared to the baseline factory that the model was trained on, have variations that still lie within the range of learned patterns during training. The addition of robots in the modified scenario might have introduced alterations that in some way are beneficial for the model, making it ignore irrelevant features or better focus on relevant ones. For instance, it may have emphasized certain features especially useful for the model to interpret the fingerprinting patterns in the CIR data. Or vice versa, that previously misleading features now have been de-emphasized.

When these results were discovered we wanted to further test the effect of adding robots to the factory floor. To see how the performance would be affected depending on *where* in the factory the robots were placed, two other modifications were created. One places robots in the very dense production area, with heavy NLoS, and the other places robots exclusively in the more sparse assembly area. With *robots in the production area* we get an accuracy of 0.45 m, while for *robots in the assembly area* we get 0.71 m.

In this way, it shows that with robots exclusively placed in the confined production area, it does not affect the positioning accuracy compared to the test set with *12 added robots* spread across the entire factory (0.47 m). That is, the ray-tracing calculations seem to be unaffected by the robots when placed among the large machines of the production area. A reason for this could be that the signals transmitted by UEs in the production area, without any robots, interact with surrounding machines such that the multipath modeling is saturated during ray-tracing. The multipath modeling is saturated if the interactions exceed the maximum total number of allowed interactions, set according to Table 3.1. If the ray-tracing is saturated, any additional interactions that the robots might have caused are not taken into account, and consequently, the resulting CIR data from the localization simulation tool remains unchanged. If this is the case, then the test sets *12 added robots* and *robots in production area* would contain the same test CIR data, explaining the similar results.

5.2.2 Generalization to Major Changes in Factory

When introducing more prominent changes, by rotating or moving machines, the negative impact on positioning accuracy is noticed, which is more in line with what is expected. Looking in Table 4.3, these so-called major changes have JSI values ranging from 0.857 to 0.921 and are labeled as *Rotated machines*, *Moved machines*

and the combination *Added robots & rotated machines*. Compared to the minor changes discussed above, these changes deviate from the baseline factory between 8-15% and are thus less likely to lie within the patterns that the fingerprinting model has learned.

Beginning with inspecting the modification with rotated machines, we get a positioning accuracy of 2.93 m. Moreover, the largest errors also worsened, which is seen by the larger RMSE of 1.91 m and the 99% CDF error reaching 7.34 m. The same modification with added robots performs with similar accuracy, which solidifies that the addition of robots has minimal negative impact on the model's positioning performance.

When introducing changes with moved machines having a JSI value of 0.857 corresponding to the most prominent modification, the positioning error reaches 5.52 m with an RMSE of 3.48 m. This shows a trend that the positioning accuracy deteriorates as the JSI value decreases, corresponding to larger changes.

Not only does the positioning accuracy decrease, but the largest errors are worsened which is reflected by the 11.22 m positioning error at 99% of the CDF. At this point, the generalization capability of the model is questionable due to the loss of precision. It then becomes a question of whether an accuracy in excess of 5 m is sufficient for industrial needs, and perhaps the model requires retraining.

Interesting observations can be made by visually inspecting the 3D representation of the positioning errors in the factory environment. This allows us to investigate *where* the errors occur. Figure 5.3 shows where the positioning errors are found for the modified factory with moved machines. To the left, the baseline and modified factory scenarios are shown from above to see what has changed. To the right, the positioning errors are visualized in 3D.

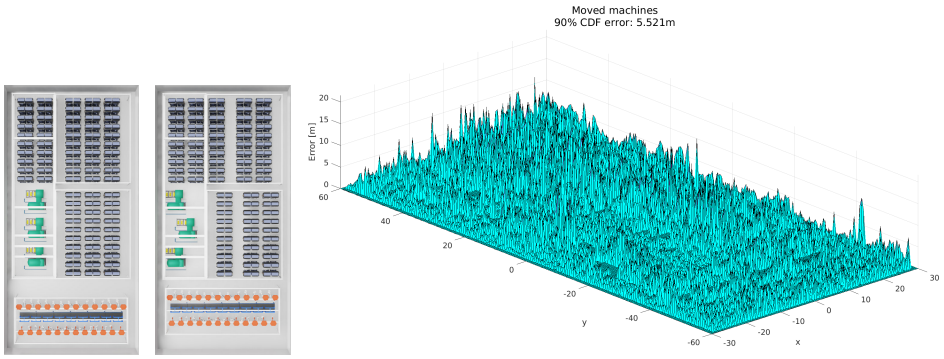


Figure 5.3: Positioning errors in baseline factory (a) when introducing modification with moved machines (b) plotted in 3D space (c).

Although the model struggles to generalize to the modification with moved machines, it is worthwhile noting that the largest errors are found in the specific local areas that has been subject to the changes. In Figure 5.3c, it is seen that the largest errors occur along the right part of the factory, which matches with where the factory has changed, compare Figures 5.3a and 5.3b.

For areas that has remained unchanged, a large extent of the measurements still give positioning errors closer to 1 m than 5 m. This insight could be useful, for instance if just a certain part of a factory has been redesigned, then it could be enough to retrain the model with data collected from just that specific area. This would be time saving, by not having to gather data from the entire factory again.

5.2.3 Mixed Training

Following the promising results of adding robots to the factory, mixed training was conducted in which also the scenarios with added robots were included as training data. This is described in Section 3.5.2.

Comparing the results of mixed training with the legacy solution in Table 4.1, the positioning accuracy is improved by almost 95% for the dense baseline scenario.

When performing tests for the model trained with the mixed dataset, $\mathcal{T}_{\text{mixed}}$, the positioning accuracy increased tremendously for the baseline test set, see Table 4.4 and 4.5. Indeed, the error is slightly more than halved when comparing with the baseline test result from using the original dataset $\mathcal{T}$. For scenarios that are part of the $\mathcal{T}_{\text{mixed}}$ dataset, the positioning accuracy improved by at least 12%, reaching 53% for the baseline test set specifically. For these scenarios, positioning errors around 0.5 m are achieved.

Positioning enhancement is achieved especially for scenarios that make up the mixed training dataset. Still, modified scenarios excluded from $\mathcal{T}_{\text{mixed}}$ experience an increase in positioning performance. Distinguishing test cases for which the scenario is not part of $\mathcal{T}_{\text{mixed}}$, the relative improvements range from 5-10%.

While the CDF positioning errors at 90% all decreased, it is a different story for the RMSE metric. The baseline test set gets a lowered RMSE, but the test sets with added robots and robots in production area give larger RMSE values for the mixed training setup. It is difficult to see any pattern for this behavior and there is no obvious explanation. However, it could be that some randomly selected test UE nodes are unfortunate in terms of placement – being located directly next to a nearby obstruction – causing an abnormal CIR which is difficult for the ResNet model to predict, and thus yielding large outlier errors.

5.3 Real Data Measurements

Even though the tracking errors in Figure 4.6b seem to be very precise for the AI/ML solution, it does not give the full picture. Almost all predictions are along the routes but the errors stem from the predictions being placed too far ahead or behind the true position. This behavior comes from that the trained model is biased to the given routes for training. This is because the measurements are not placed exhaustively in the office environment, as described in Section 3.6.

Also, when looking at the CDF positioning errors in Figure 4.5, the ResNet fingerprinting model is heavy-tailed. This is not the case when looking at the legacy triangulation solution in the same figure. It can be seen in Table 4.6 that the ResNet fingerprinting model gives higher accuracy up to 90% of the CDF, but after that, the legacy solution performs better. Meaning that the legacy solution performs better than the fingerprinting model concerning the worst 10% of the predictions.

Another thing to consider is the matching of the CIR and robot position timestamps. Since this matching is done with precision of seconds and the robot moved at a top speed of 0.2 m/s, this means that the CIR could be mapped to a position up to 0.2 m from its actual position. This also affects the legacy solution for the real data measurements.

5.4 Validity of Results

The indoor factories used in this thesis comply with the agreed scenario definition by 3GPP, commonly used in their work and in scientific papers. This assures that the indoor factory scenarios are relevant both today and for future work, with comparable results. Although this thesis used site-specific ray-tracing channel model, the positioning accuracy results can be comparable with 3GPP studies using statistical channel models.

Data obtained through simulations depend on Ericsson's state-of-the-art ray-tracing channel model, described in Section 3.3.1, which performs computationally heavy calculations describing realistic site-specific multipath propagation. Specular reflections and diffuse scattering interactions are modeled, as listed in Table 3.1. However, a deficiency is that further interactions arising from diffractions and transmissions, have not been modeled. The impact of not modeling transmissions, is negligible since objects in the studied factories are metallic, which would not make full penetration feasible for the signals. The lack of modeled diffractions could deteriorate the level of realism to the ray-tracing channel model, but has been omitted to make simulations feasible, since they are costly in terms of ray-tracing with added computational complexity. Nonetheless, it does not render the simulated ray-tracing calculations invalid. Actually, if inclusion of diffraction would be possible, it would enrich the channel modeling resulting in CIR data containing further patterns for an AI/ML model to learn from.

Since this thesis generates specific baselines for the different scenarios, the modifications need to be relevant. Although minor changes are more likely to be applied to the factories, major changes are just as important to stress the AI/ML models. The changes selected provide an overview of different kinds of

modifications that possibly could take place in a factory, yet there are endless other ways to change the environment.

Chapter 6
Conclusions

6.1 Research Question 1

To what extent can AI/ML models improve positioning accuracy in industrial indoor environments compared to legacy time-of-arrival based solutions?

The AI/ML positioning methods perform better than the legacy triangulation positioning solution. For both sparse and dense factory environments, the two AI/ML models give positioning errors below 1 m. Corresponding errors for the legacy solution are 3.5 m and 8.7 m for the sparse and dense baseline factories, respectively. Even though smaller positioning errors always are desired, the models massively improve positioning accuracy compared to legacy solutions, which is essential for being applicable in the modern industrial landscape.

In regards to the real data measurements taken from an open office environment, the positioning accuracy using AI/ML also improved compared to the legacy solution, going from 1.5 m to just below 1 m.

It is evident that the AI/ML models improve positioning accuracy for the studied environments compared to legacy solutions. This is especially the case for scenarios with heavy NLoS conditions, in which the legacy solution faces difficulties, and the positioning improvement is more prominent.

6.2 Research Question 2

How robust are these models in terms of generalizing to physical changes in the environments?

The CNN model applied to sparse factory scenarios, with relatively high LoS probability, generalizes well to physical changes in the environment and is considered to be robust.

The ResNet fingerprinting model applied for dense factory scenarios is not as robust, and it is concluded that the ResNet fingerprinting approach is more sensitive to changes in the environment compared to the CNN model. Nonetheless, it shows robustness to minor changes. From the results, it is seen that the ResNet model performs unexpectedly well, as discussed in Section 5.2.1, in terms of generalizing to scenarios with added robots. The presence of added robots across the factory floor is likely the most frequent type of change in a factory.

However, for larger changes, the positioning accuracy deteriorates and the ResNet model does not generalize well. Yet, the accuracy is mainly affected in the local area that changed, which could suppress the need to retrain the model with data collected from the entire factory.

6.3 Future Work

This project used simulated 5G data and since the time was limited, some assumptions had to be made. The antennas patterns for both the UE and TRP are omni-directional. Different antenna patterns could be applied using the simulations to see how much they affect the results. With omni-directional antennas, a LoS link is usually the strongest signal received by the antenna. If the antenna pattern have directional gain, this does not have to be the case anymore. This is because the signal strength is dependent on angle-of-arrival and angle-of-departure. Comparing these results would give a better understanding of how much the antenna pattern affects performance of the AI/ML models.

For the real 5G data, the robot only drove in two specific routes in an open office environment rather than the exhaustive node deployment as in the simulated factory environments. Therefore, another future work could be to collect data from the entire office room. The CNN model could also be used for this scenario, although requiring the ground truth ToA and LoS for every measurement.

Another future work is to explore other mixed training setups, with other factory scenarios constituting the mixed training dataset. This thesis only aggregated data from the baseline scenario and the scenarios with added robots, but endless varieties are possible. Since mixed training gave promising results, the possibility to further improve positioning accuracy exists.

References

[1] Farouq Halawa, Husam Dauod, In Gyu Lee, Yinglei Li, Sang Won Yoon, and Sung Hoon Chung. Introduction of a real time location system to enhance the warehouse safety and operational efficiency. *International Journal of Production Economics*, 224:107541, 2020.

[2] Jingrong Liu, Zhongliang Deng, and Enwen Hu. An nlos ranging error mitigation method for 5g positioning in indoor environments. *Applied Sciences*, 14(9), 2024.

[3] Wen-Long Chin, Cheng-Che Hsieh, David Shiung, and Tao Jiang. Intelligent indoor positioning based on artificial neural networks. *IEEE Network, Network, IEEE*, 34(6):164 – 170, 2020.

[4] Jianyuan Yu, Hussein Metwaly Saad, and R. Michael Buehrer. Centimeter-level indoor localization using channel state information with recurrent neural networks. *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS), Position, Location and Navigation Symposium (PLANS), 2020 IEEE/ION*, pages 1317 – 1323, 2020.

[5] Shuang Shang and Lixing Wang. Overview of wifi fingerprinting-based indoor positioning. *IET Communications*, 16(7):725–733, 2022.

[6] Mohammad Nabati and Seyed Ali Ghorashi. A real-time fingerprint-based indoor positioning using deep learning and preceding states. *Expert Systems with Applications*, 213:118889, 2023.

[7] Guan-Sian Wu and Po-Hsuan Tseng. A deep neural network-based indoor positioning method using channel state information. *2018 International Conference on Computing, Networking and Communications (ICNC), Computing, Networking and Communications (ICNC), 2018 International Conference on*, pages 290 – 294, 2018.

[8] Jaspreet Kaur, Mahmoud Shawky, Michael S. Mollel, Olaoluwa R Popoola, Muhammad Ali Imran, Qammer H. Abbasi, and Hasan T. Abbas. Ai-enabled csi fingerprinting for indoor localisation towards context-aware networking in 6g. In *2023 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–5, 2023.

[9] 3GPP. 5G; Study on channel model for frequencies from 0.5 to 100 GHz (3GPP TR 38.901 Version 18.0.0 Release 18). Technical Report 38.901, ETSI: Sophia Antopolis, France, March 2024. Available online: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3173 (Accessed May 16, 2024).

[10] 3GPP. Introducing 3gpp. https://www.3gpp.org/about-us/introducing-3gpp, 2024. Accessed April 18, 2024.

[11] Aswathi Vijayan, Michael Kuhn, Jobish John, Md. Noor-A-Rahim, Dirk Pesch, Billy O'Connor, Kevin Crean, and Eddie Armstrong. 5g wireless channel characterization in indoor factory environments: Simulation and validation. *2023 IEEE Wireless Communications and Networking Conference (WCNC), Wireless Communications and Networking Conference (WCNC), 2023 IEEE*, pages 1 – 6, 2023.

[12] Andreas F Molisch. *Wireless communications*, volume 34. John Wiley & Sons, 2012.

[13] 3GPP. NR; Base Station (BS) radio transmission and reception). Technical Report 38.104, ETSI: Sophia Antopolis, France, February 2024. Available online: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3202 (Accessed April 16, 2024).

[14] Florent Munier Satyam Dwivedi, Johannes Nygren and Fredrik Gunnarsson. 5g positioning: What you need to know. https://www.ericsson.com/en/blog/2020/12/5g-positioning--what-you-need-to-know, 2020. Accessed April 11, 2024.

[15] Erik Dahlman, Stefan Parkvall, and Johan Sköld. Chapter 6 - radio-interface architecture. In Erik Dahlman, Stefan Parkvall, and Johan Sköld, editors, *5G NR: the Next Generation Wireless Access Technology*, pages 73–102. Academic Press, 2018.

[16] Saleh Faruque. *Multipath Propagation.* SpringerBriefs in Electrical and Computer Engineering. Springer International Publishing, 2015.

[17] Simone Del Prete, Franco Fuschini, Marina Barbiroli, and Mohammad Hossein Zadeh. A study on propagation of frequency diverse array in multipath environments. *2023 IEEE-APS Topical Conference on Antennas and Propagation in Wireless Communications (APWC), Antennas and Propagation in Wireless Communications (APWC), 2023 IEEE-APS Topical Conference on*, pages 090 – 094, 2023.

[18] Alan Bensky. Chapter 3 - antennas and transmission lines. In Alan Bensky, editor, *Short-range Wireless Communication (Third Edition)*, pages 43–83. Newnes, third edition edition, 2019.

[19] Yimei Kang, Qingyang Wang, Jiawei Wang, and Renhai Chen. A High-Accuracy TOA-Based Localization Method Without Time Synchronization

in a Three-Dimensional Space. *IEEE Transactions on Industrial Informatics*, 15(1):173–182, January 2019.

[20] *Weighted Least-Squares Method*, pages 566–569. Springer New York, New York, NY, 2008.

[21] Ameet V. Joshi. *Machine learning and artificial intelligence.* Springer International Publishing, 2023.

[22] Kun Il Park and service) SpringerLink (Online. *Fundamentals of Probability and Stochastic Processes with Applications to Communications.* Springer International Publishing, 2018.

[23] Leon Derczynski. Complementarity, F-score, and NLP evaluation. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 261–266, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA).

[24] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA, 2015.

[25] Shengjie Wang, Tianyi Zhou, and Jeff Bilmes. Bias Also Matters: Bias Attribution for Deep Neural Network Explanation. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6659–6667. PMLR, May 2019. ISSN: 2640-3498.

[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[27] L. Huang, J. Qin, Y. Zhou, F. Zhu, L. Liu, and L. Shao. Normalization techniques in training dnns: Methodology, analysis and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Pattern Analysis and Machine Intelligence, IEEE Transactions on, IEEE Trans. Pattern Anal. Mach. Intell*, 45(8):10173 – 10196, 2023.

[28] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[29] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.

[30] Andreas Lindholm. *Machine learning. a first course for engineers and scientists.* Cambridge Textbooks - All. Cambridge University Press, 2022.

[31] Sunitha Basodi, Chunyan Ji, Haiping Zhang, and Yi Pan. Gradient amplification: An efficient way to train deep neural networks. *Big Data Mining and Analytics*, 3(3):196–207, 2020.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, December 2015. arXiv:1512.03385 [cs].

[33] Haodong Zhao. 3d modeling of factory scenarios for 5g evaluations. Master's thesis, Uppsala University, Department of Information Technology, 2023.

[34] Sven Kosub. A note on the triangle inequality for the jaccard distance. *CoRR*, abs/1612.02696, 2016.

[35] Chaiyong Ragkhitwetsagul, Jens Krinke, and Bruno Marnette. A picture is worth a thousand words: Code clone detection based on image similarity. In *2018 IEEE 12th International workshop on software clones (IWSC)*, pages 44–50. IEEE, 2018.

[36] Takaharu Kato, Ikuko Shimizu, Tomas Pajdla, and Organization MVA. Selecting image pairs for sfm by introducing jaccard similarity. pages 25 – 29, 2017.

[37] Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi diagrams and Delaunay triangulations*. World Scientific Publishing Company, 2013.

[38] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. Ros: an open-source robot operating system. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, May 2009.

Appendix A

Sparse Factory Scenarios

This Appendix gives an overview of the sparse factory scenarios, including all the different modifications and positioning errors plotted in 3D space to see where errors occur.

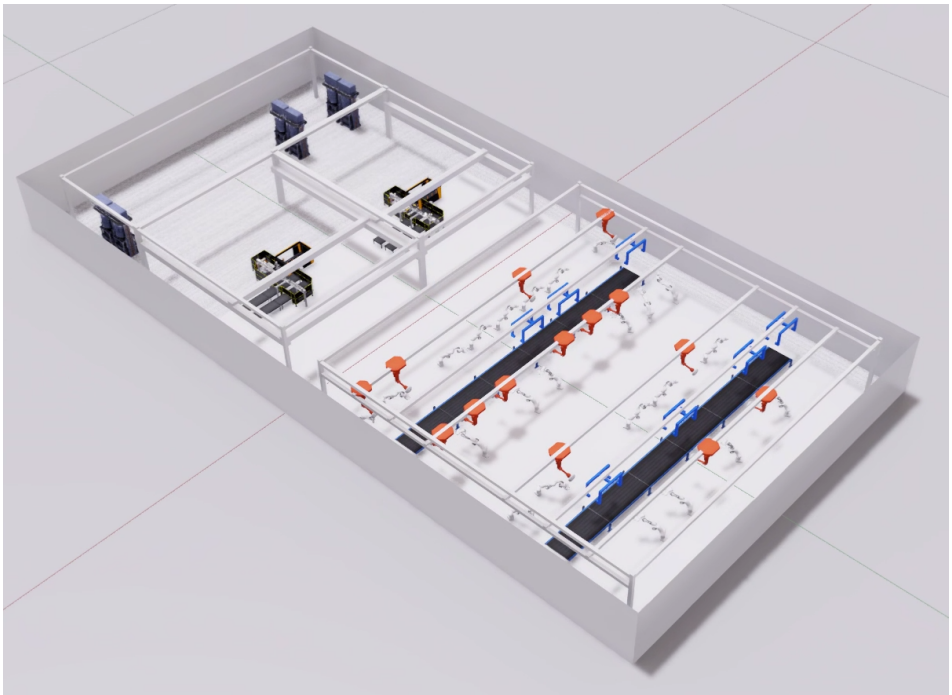
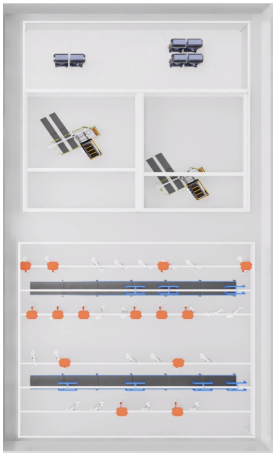


Figure A.1: Perspective view of sparse baseline factory.

(a) Baseline factory.

(b) Added robots (20) & moved machines.



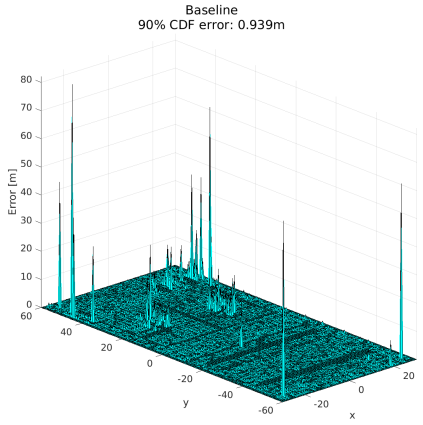
(c) Rotated machines.

(d) Flipped factory layout.

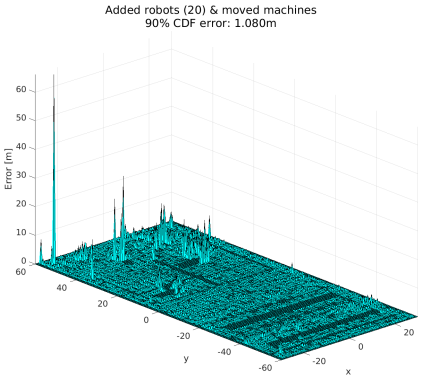
(e) Different factory layout.

Figure A.2: Sparse factory modifications.

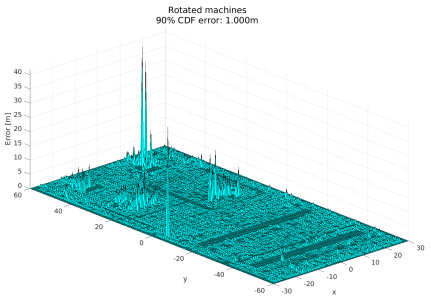
A.1 Positioning Errors



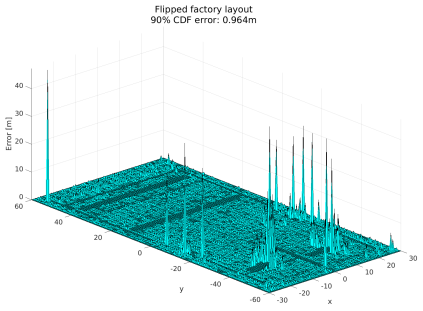
(a) Baseline.



(b) Added robots & moved machines.



(c) Rotated machines.



(d) Flipped factory layout.

Figure A.3: Positioning errors in sparse factory scenarios.

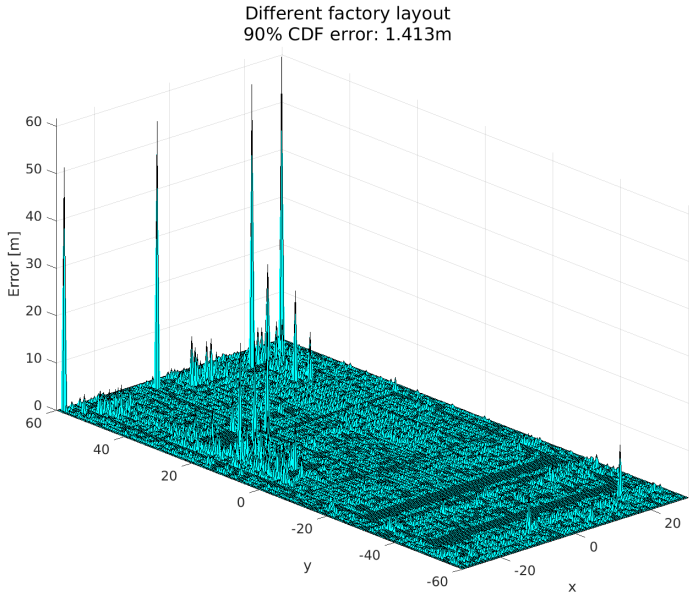


Figure A.4: Positioning errors in sparse factory scenarios (continued).

Appendix B

Dense Factory Scenarios

This Appendix gives an overview of the dense factory scenarios, including all the different modifications and positioning errors plotted in 3D space to see where errors occur.

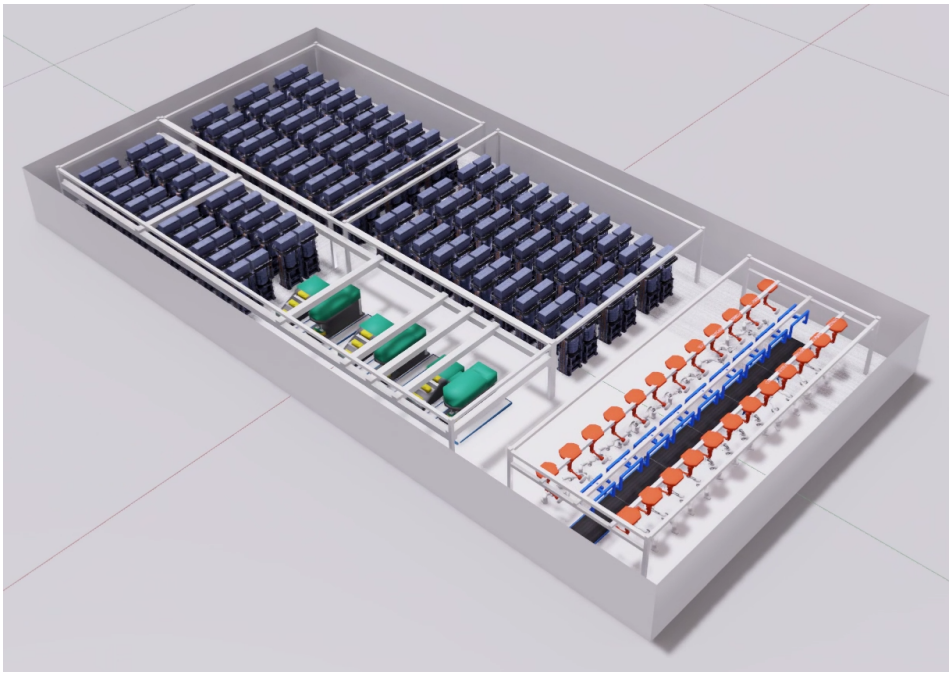
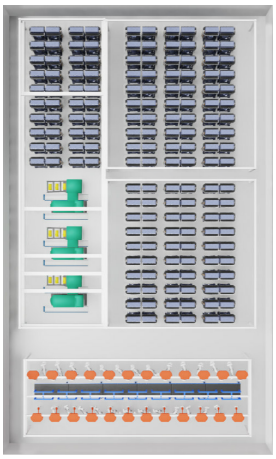
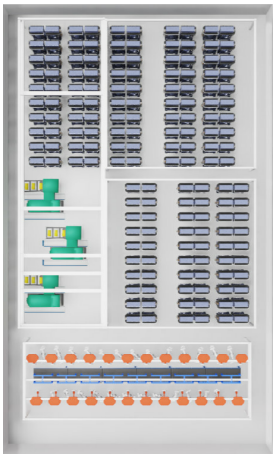


Figure B.1: Perspective view of dense baseline factory.



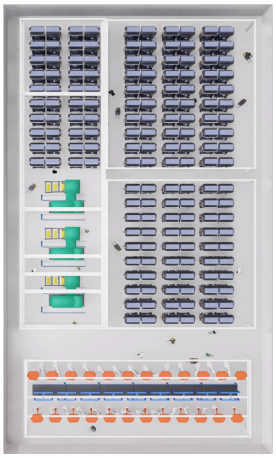
(a) Baseline factory.

(b) Rotated machines.

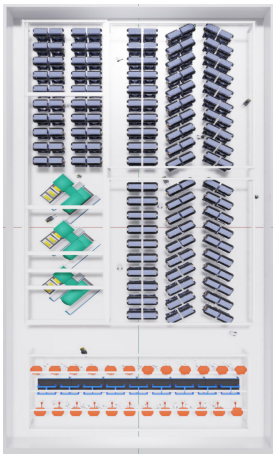


(c) Moved machines.

(d) Added robots (12).

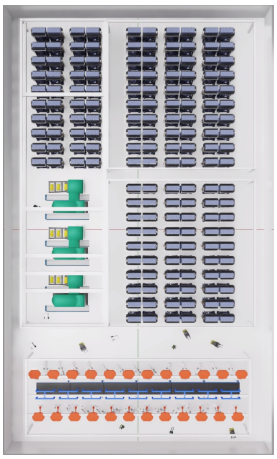


(e) More robots (20).



(f) Added robots (12) & rotated machines.

Figure B.2: Dense factory modifications.

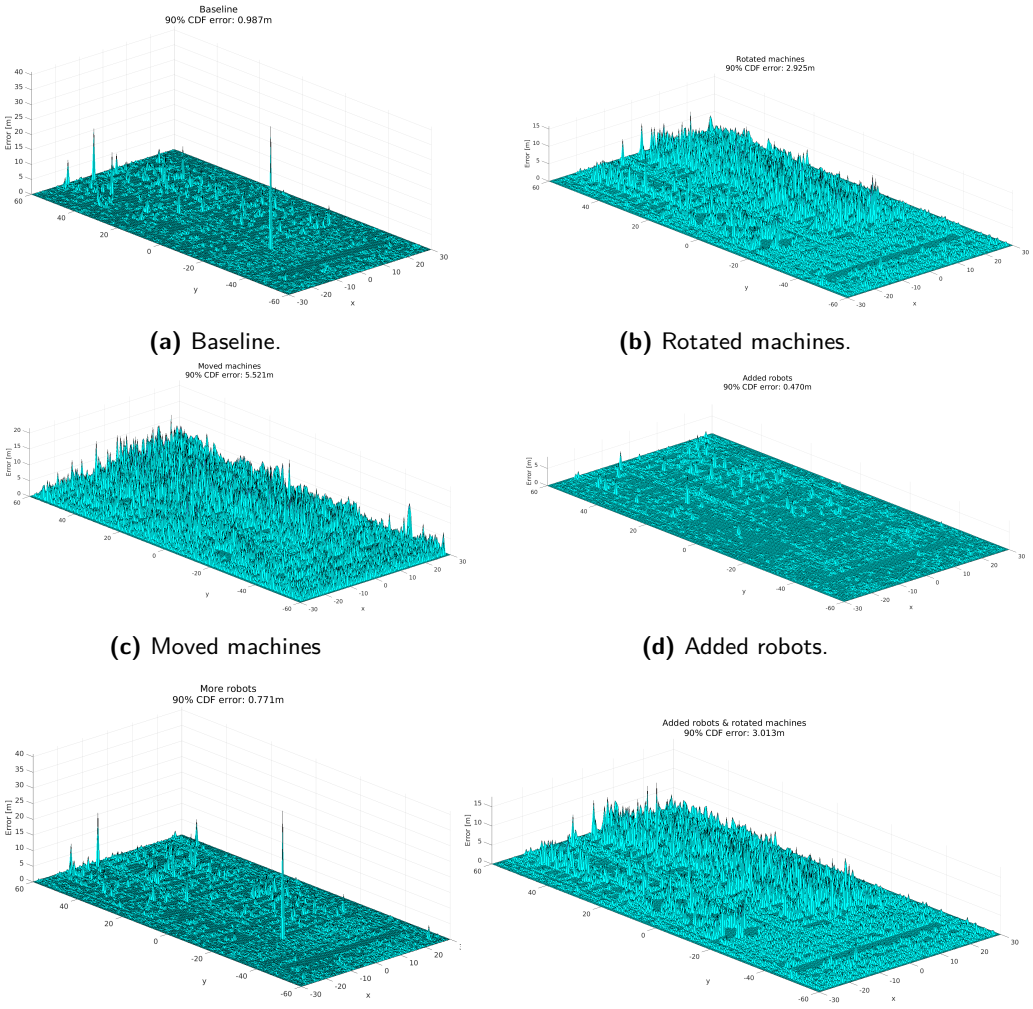


(a) Added robots in assembly area (12).

(b) Added robots in production area (12).

Figure B.3: Dense factory modifications (continued).

B.1 Positioning Errors

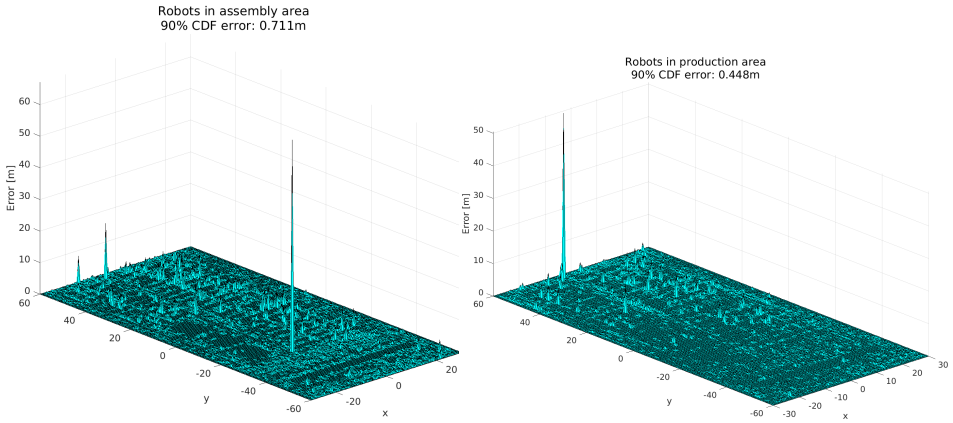


(a) Baseline. **(b)** Rotated machines.

(c) Moved machines **(d)** Added robots.

(e) More robots. **(f)** Added robots & rotated machines.

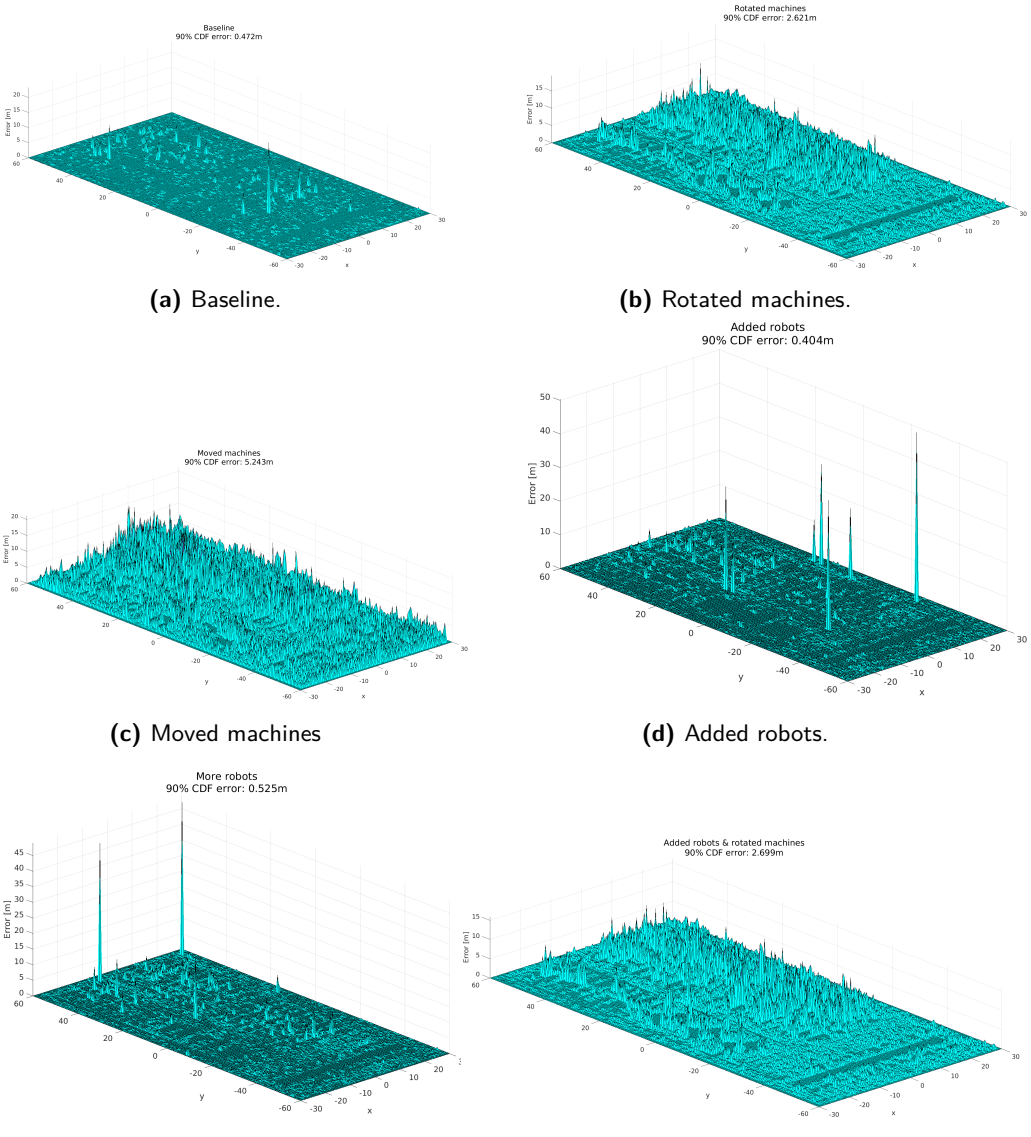
Figure B.4: Positioning errors in dense factory scenarios.



(a) Robots in assembly area. **(b)** Robots in production area.

Figure B.5: Positioning errors in dense factory scenarios (continued).

B.1.1 Positioning Errors for Mixed Training



(a) Baseline.

(b) Rotated machines.

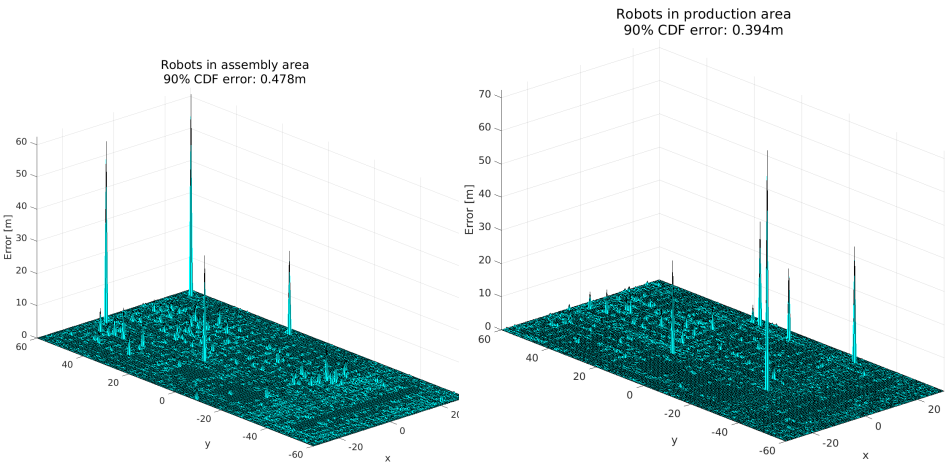
(c) Moved machines

(d) Added robots.

(e) More robots.

(f) Added robots & rotated machines.

Figure B.6: Positioning errors in dense factory scenarios for mixed training.



(a) Robots in assembly area. **(b)** Robots in production area.

Figure B.7: Positioning errors in dense factory scenarios for mixed training (continued).

Appendix C
Plots of CIR Data

This Appendix contains plots of CIR data to provide visualization of what data has been used in this thesis. In Figures C.1 and C.2, CIR data is plotted differentiating LoS and NLoS cases. Links having LoS are plotted in solid lines, while NLoS links are dashed.

Figures C.3 and C.4 visualize the CIR data in the form of 256×36 images, with pixel intensities corresponding to the CIR magnitudes.

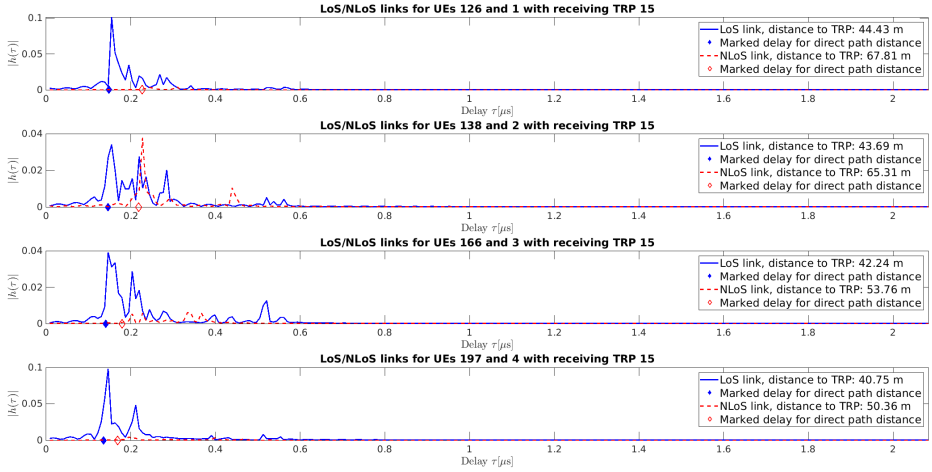


Figure C.1: CIR data for randomly selected UEs transmitting to TRP 15.

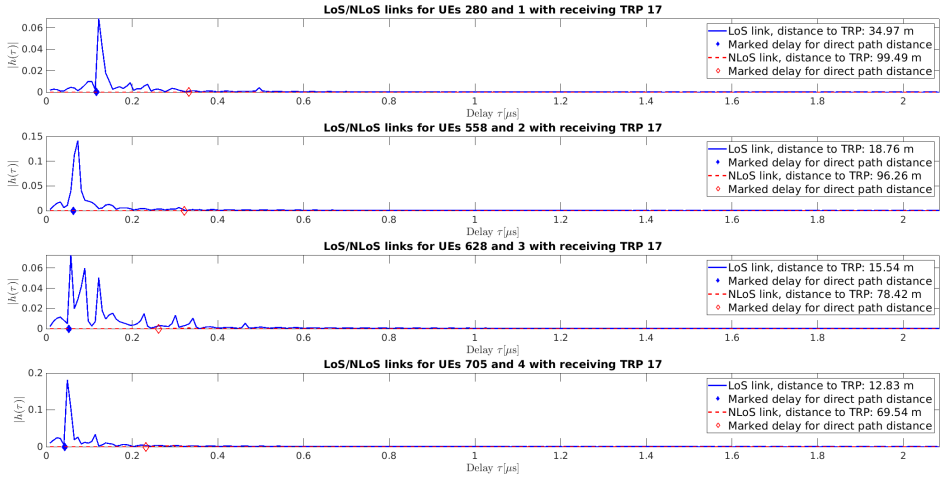


Figure C.2: CIR data for randomly selected UEs transmitting to TRP 17.

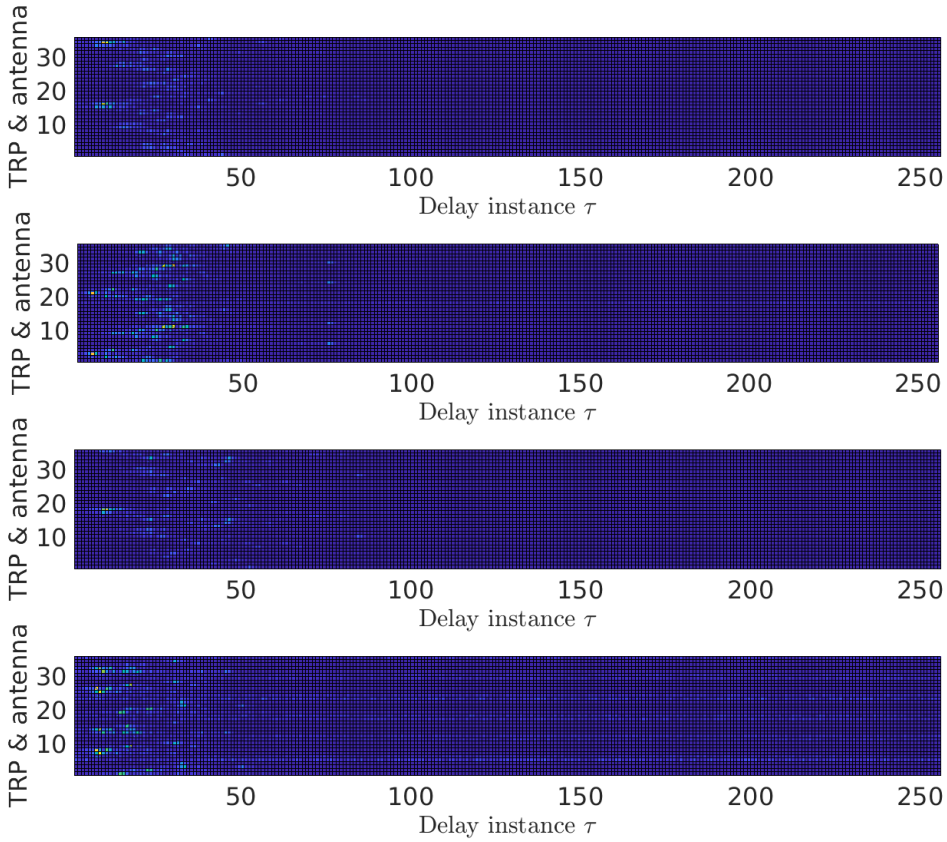


Figure C.3: Visualized CIR data in the form of 256×36 images.

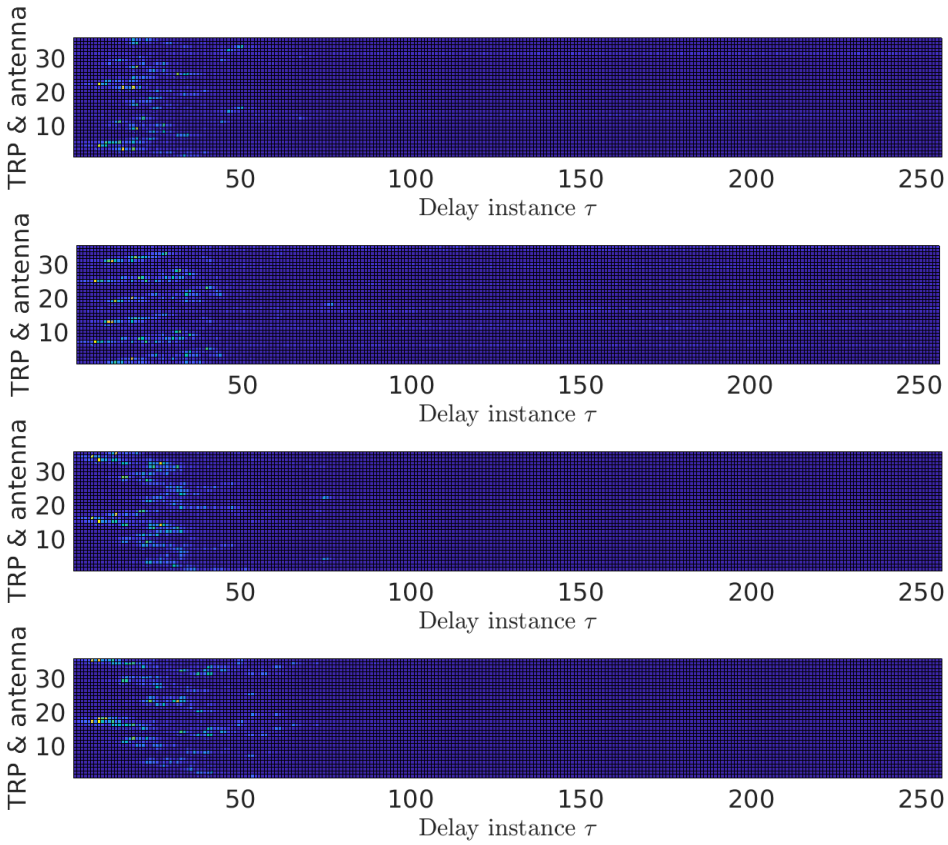


Figure C.4: Visualized CIR data in the form of 256×36 images.

Printed by Tryckeriet i E-huset, Lund 2024

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2024-1002
http://www.eit.lth.se