POLITECNICO DI TORINO

Department of Electronics and Telecommunications

Master's Degree in Mechatronic Engineering

Master's Thesis

Deep Learning Methodologies for UWB Ranging Error Compensation

Supervisor:
prof. Marcello Chiaberge

Candidate:
Simone Angarano

Academic Year 2019-2020

Printed October 13, 2020

Abstract

Ultra-Wideband is becoming extensively used in many kinds of robot and human positioning systems. From industrial robotic tasks to drones used for search and rescue operations, this high-accuracy technology allows to locate a target with an error of a few centimeters, outperforming other existing low-cost ranging methods like Bluetooth and Wi-Fi. This fact also lead Apple to equip the latest IPhone 11 with an UWB module specifically for precise localization applications. Unfortunately, this technology is very accurate only in Line-Of-Sight. Indeed, performances degrade significantly in Non-Line-Of-Sight scenarios, in which walls, furniture or people obstruct the direct path between the antennas. Moreover, reflections constitute another source of error, causing the receiver to detect multiple signals with different delays. The aim of this thesis is to compensate NLOS and multi-path errors and obtain a precise and reliable positioning system, that could allow to develop several service robotics applications that are now limited by unsatisfactory accuracies. Another fundamental goal is to guarantee a good scalability of the system to unseen scenarios, that is where even modern mitigation methods still fail. For this scope, a large dataset is built taking both LOS and NLOS measurements in different environments and experimenting on different types of obstacles. Then, modern deep learning methods are used to design a Convolutional Neural Network that estimates the error of the range estimates directly from raw Channel Impulse Response samples. Finally, a positioning test is conducted to verify the effectiveness of the method in a real scenario.

The rest of the work is organized as follows. Chapter 1 introduces the theoretical bases that are needed to understand the context and the development of the thesis: UWB definition and features, main ranging methods, positioning algorithms, machine learning and deep neural networks. In Chapter 2, an overview of the state of the art is addressed, highlighting strenghts and limits of each of the existing methods. Chapter 3 describes the experimental set adopted for the range measurements and the building of the dataset, also comprehending the characteristics of low-cost sensors and all the employed hardware and software. Chapter 4 addresses the design of the Deep Learning models used to perform the correction of the range estimate and the strategy adopted to optimally train and test different variants. Chapter 5 shows and comments the results reached after the training of the models, highlighting the effect of environment and obstacles on the prediction accuracy. It also presents the final positioning test conducted in real-time. Finally, Chapter

6 draws conclusions on the presented work, pointing at some improvable aspects of the method and suggesting future objectives to pursuit.

Contents

List of Tables

List of Figures

Acronyms

CIR Channel Impulse Response.

CNN Convolutional Neural Network.

FC Fully Connected.

GAP Global Average Pooling.

IC Integrated Circuit.

IR Impulse Radio.

KNN K-Nearest Neighbors.

LOS Line-of-Sight.

MAE Mean Absolute Error.

MSE Mean Squared Error.

NLOS Non-Line-of-Sight.

OS Operating System.

PSD Power Spectral Density.

ReLU Rectified Linear Unit.

RSL Received Signal Level.

TPU Tensor Processing Unit.

UWB Ultra-Wideband.

"The same thrill, the same awe and mystery, come again and again when we look at any problem deeply enough.
With more knowledge comes deeper, more wonderful mystery, luring one on to penetrate deeper still.
Never concerned that the answer may prove disappointment, but with pleasure and confidence we turn over each new stone to find unimagined strangeness leading on to more wonderful questions and mysteries - certainly a grand adventure!"

[R. Feynman]

Chapter 1

Introduction

The outstanding progress of robotics in the last two decades, aided by the increasing development of disciplines like control theory and machine learning, has led to the design of countless automated solutions for a wide variety of application fields. In particular, recent years have seen a growing interest in employing robots in everyday life contexts, pushed by the enormous innovation that this could bring in houses, offices and hospitals. Clearly, though, using moving robotic systems in uncontrolled and continuously changing environments gives rise to numerous issues. The presence of obstacles like people or objects, for example, requires the robot to have a very high confidence of the world around to travel from point to point in security. Dealing with these criticalities, the availability of a precise localization system is of paramount importance as the construction of a map is the first step for the unmanned vehicle to navigate in a complex environment. For this purpose, UWB is becoming extensively used in many kinds of applications for its high-accuracy technology, allowing to locate a target with an error of a few centimeters, and its low cost. This thesis aims at studying and compensating the errors in UWB positioning caused by NLOS conditions and reflections. In this first chapter, an overview of all the theoretical concepts at the base of the proposed work is presented. Firstly, the main notions about UWB are summarised, from its communications features to the localization applications. Then a brief introduction of machine learning and deep learning is addressed, focusing on both common shallow algorithms and recent deep methodologies.

1.1 Ultra-Wideband

1.1.1 Definition and Legislation

According to the definition given by the Federal Communication Commission (FCC), an UWB signal is "any signal which has a fractional bandwidth B_f larger than 0.20, or which occupies a bandwidth BW greater than 500 MHz" [1], i.e.,

$$B_f \geq 0.2 \quad or \quad BW > 500\,\text{MHz} \tag{1.1}$$

The fractional bandwidth is defined as the ratio between the bandwidth of the signal and its center frequency

$$Bf = \frac{BW}{f_c} = \frac{f_h - f_l}{\frac{f_h + f_l}{2}} \tag{1.2}$$

Being f_h and f_l respectively the highest and lowest transmitted frequencies at the -10 dB emission point and f_c the center frequency. To avoid interference with other existing communication systems, FCC and the European Commission's Radio Spectrum Committee defined specific spectral masks to regulate the use of UWB transmitters respectively in the USA and in the European Union. These masks (5.1) fix the maximum Power Spectral Density (PSD) transmittable for different frequency bands.

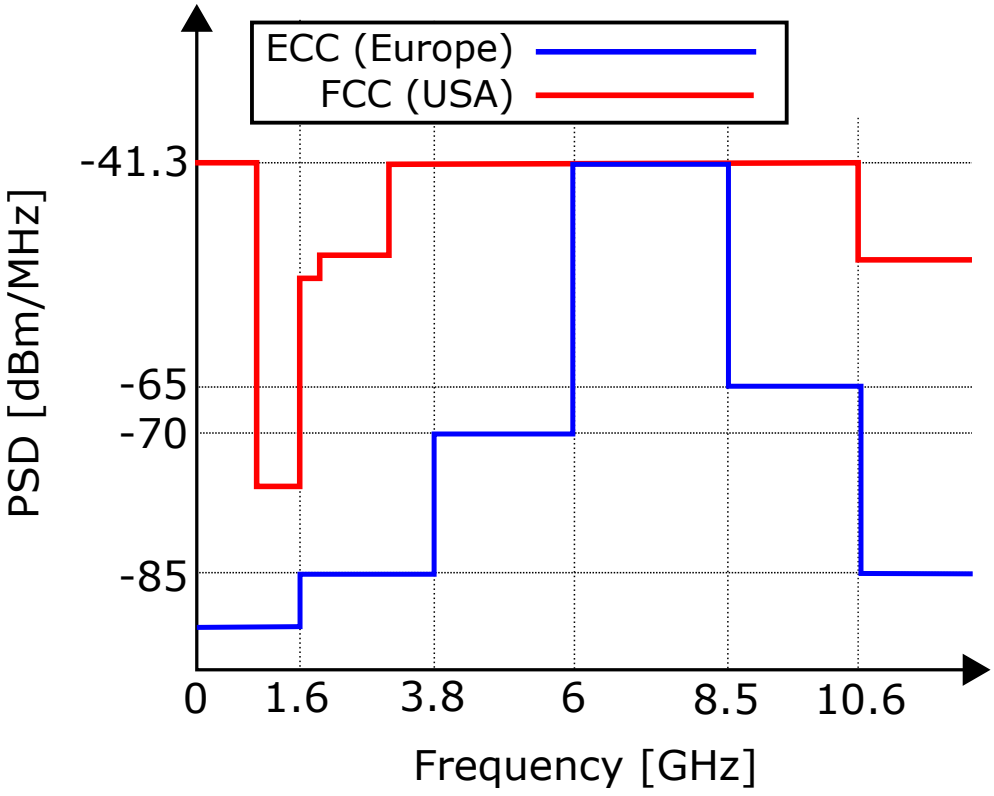


Figure 1.1. UWB spectral masks: FCC and EU.

1.1.2 Main Features

The main feature that makes UWB a unique and useful technology in many applications is that its huge bandwidth means very fine time resolution. This allows to have a better noise rejection with respect to other wireless communication systems and, hence, precise positioning.

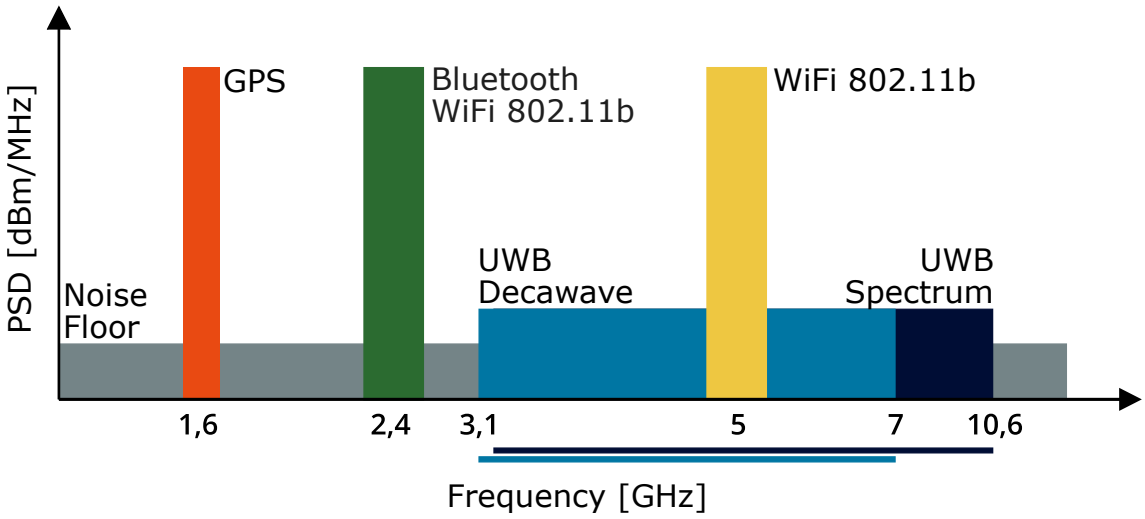


Figure 1.2. UWB bandwidth compared to Wi-Fi, Bluetooth and GPS.

To practically realize this narrow signal shape Impulse Radio (IR) is usually adopted, which is based on transmitting extremely short and low power pulses (1.3). As can be seen in 5.2, the center frequency of the signal is relatively low. This feature causes the UWB signal to suffer of small attenuations when passing through materials, contrary to systems at higher frequencies, and perform well for long range communication.

Finally, a low transmit power (in the order of μW) makes UWB the optimal low-cost technology that guarantees precise communication and positioning.

1.1.3 Transmission

As mentioned before, the most common transmission method for UWB is Impulse Radio (IR). It is based on the transmission of baseband pulses that code information through signal polaruty and the time position inside a transmission window. The pulses' length is typically in the order of sub-nanosecond and can have diverse shapes, usually starting from a Gaussian pulse (1.4).

$$x(t) = \frac{A}{\sqrt{2\pi\sigma^2}} e^{\frac{-t^2}{2\sigma^2}} \tag{1.3}$$

In 1.3, A is the pulse amplitude and sigma is the standard deviation.

IR is a carrier-less transmission, and this allows to have very low power consumption.

1.1.4 Modulation

In IR-UWB communication systems data can be modulated using different schemes (1.5):

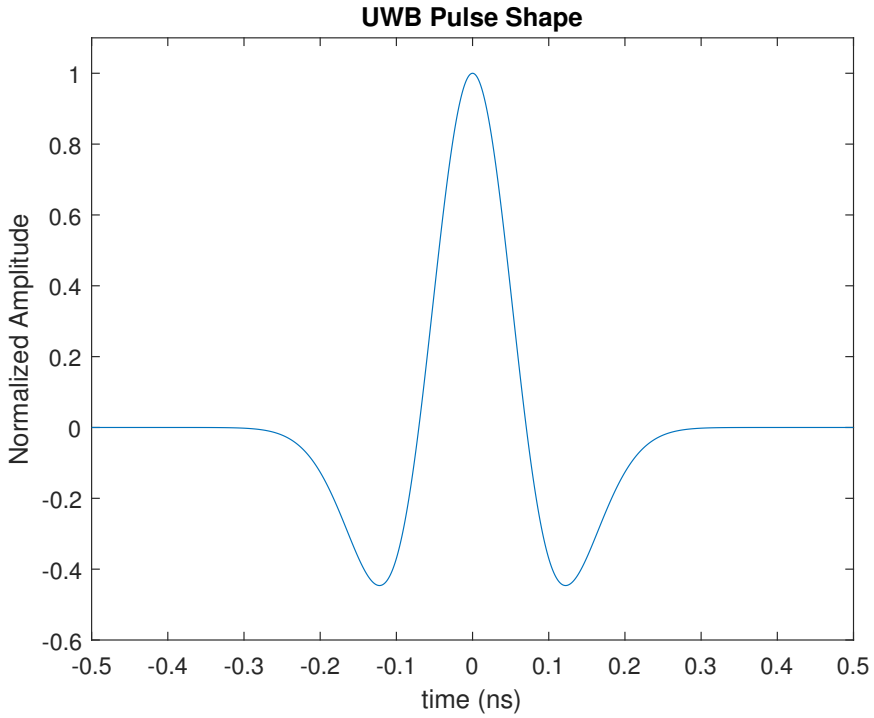


Figure 1.3. UWB pulse.

- Bi-Phase Modulation (BPM), in which the data is encoded in the polarity of the impulses,
- Pulse Amplitude Modulation (PAM), which is based on the amplitude of the pulses,
- On-Off Keying (OOK), in which the transmission of a pulse represents a bit "1" of data and its absence represents a "0",
- Pulse Position Modulation (PPM), in which the data is encoded by adding a shift to the impulse.

Among all the possibilities PPM has proven to be the most preferable one, because it smoothens the frequency spectrum of the signal. In fact, in UWB pulses are sent at a constant rate called Pulse Repetition Frequency (PRF), but this normally generates peaks in the spectrum that reduce the maximum transmitted power. Shifting the pulses their occurrence is not perfectly periodic and so the frequency spectrum appears smoothened (1.6).

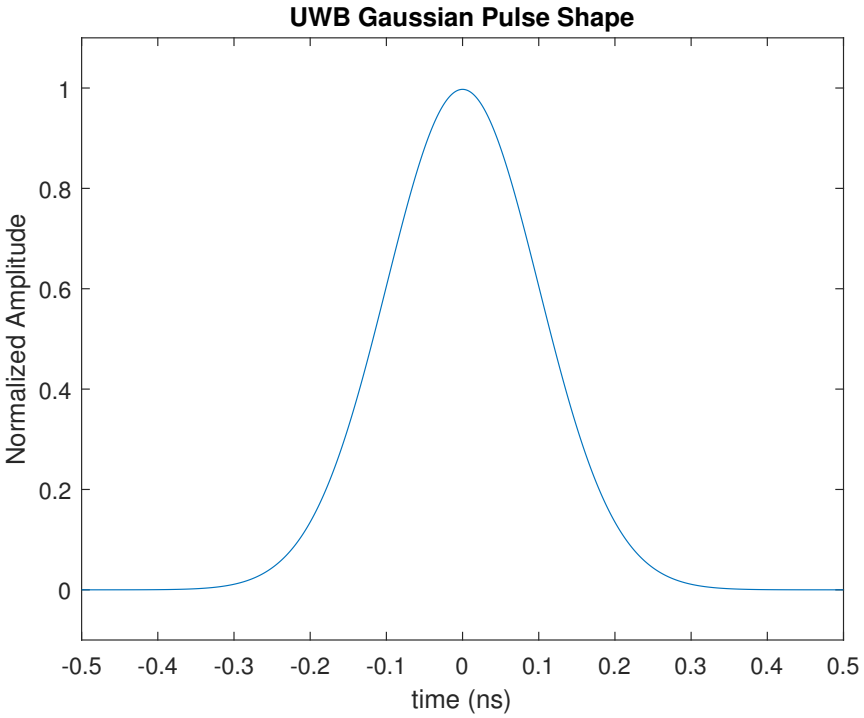


Figure 1.4. UWB Gaussian pulse.

1.1.5 Channel Model

The UWB wireless communication channel can be described with the aid of the channel impulse response (CIR) function $h(t)$, stochastically modelled as

$$h(t) = \sum_{n=1}^{N} a_n \delta(t - \tau_n)) \tag{1.4}$$

where N is the number of multipath components of amplitude a_n, delay τ_n and phase θ_n; δ is the dirac's delta function. In Non-Line-of-Sight cases, more complex models can be used, also accounting for phase shifts.

$$h(t) = \sum_{n=1}^{N} a_n \delta(t - \tau_n)\ e^{j}\theta_n \tag{1.5}$$

In 1.5, θ_n is the angle representing the phase shifting caused by the signal passing through an obstacle. The CIR is also equivalent to the inverse Fourier transform of the complex transfer function $H(j\omega)$. If the channel is modeled in such way, the response $y(t)$ of any transmitted signal $s(t)$ is found computing the convolution integral and adding a complex

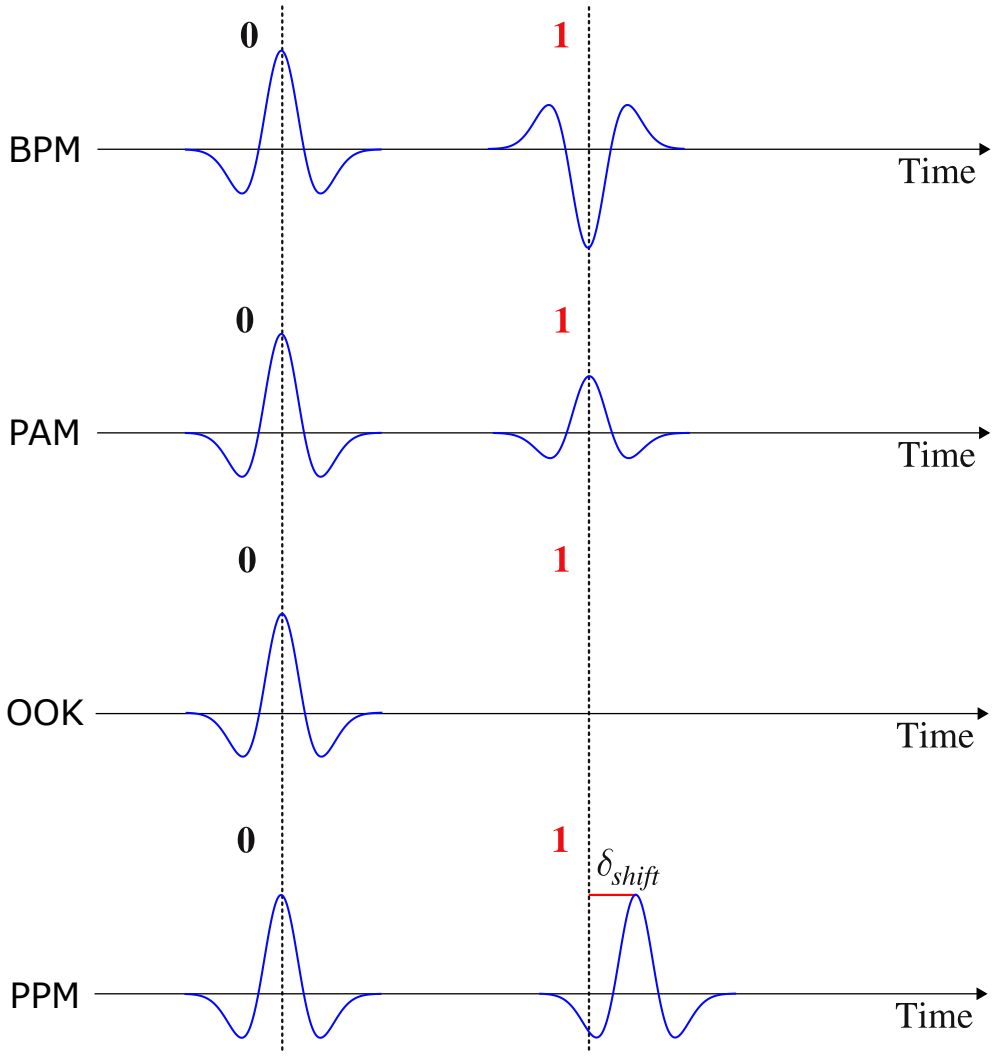


Figure 1.5. Modulation techniques: BPM, PAM, OOK and PPM.

Gaussian noise:

$$y(t) = \int_{-inf}^{inf} s(x)h(t-x)\,dx \; + n(t) \tag{1.6}$$

1.2 UWB for Localization

1.2.1 Ranging Setup and Terminology

To introduce the section about positioning methods, the typical experimental scenario will be presented, and the main specific terminology will be defined.

Referring to 1.7, the main elements are:

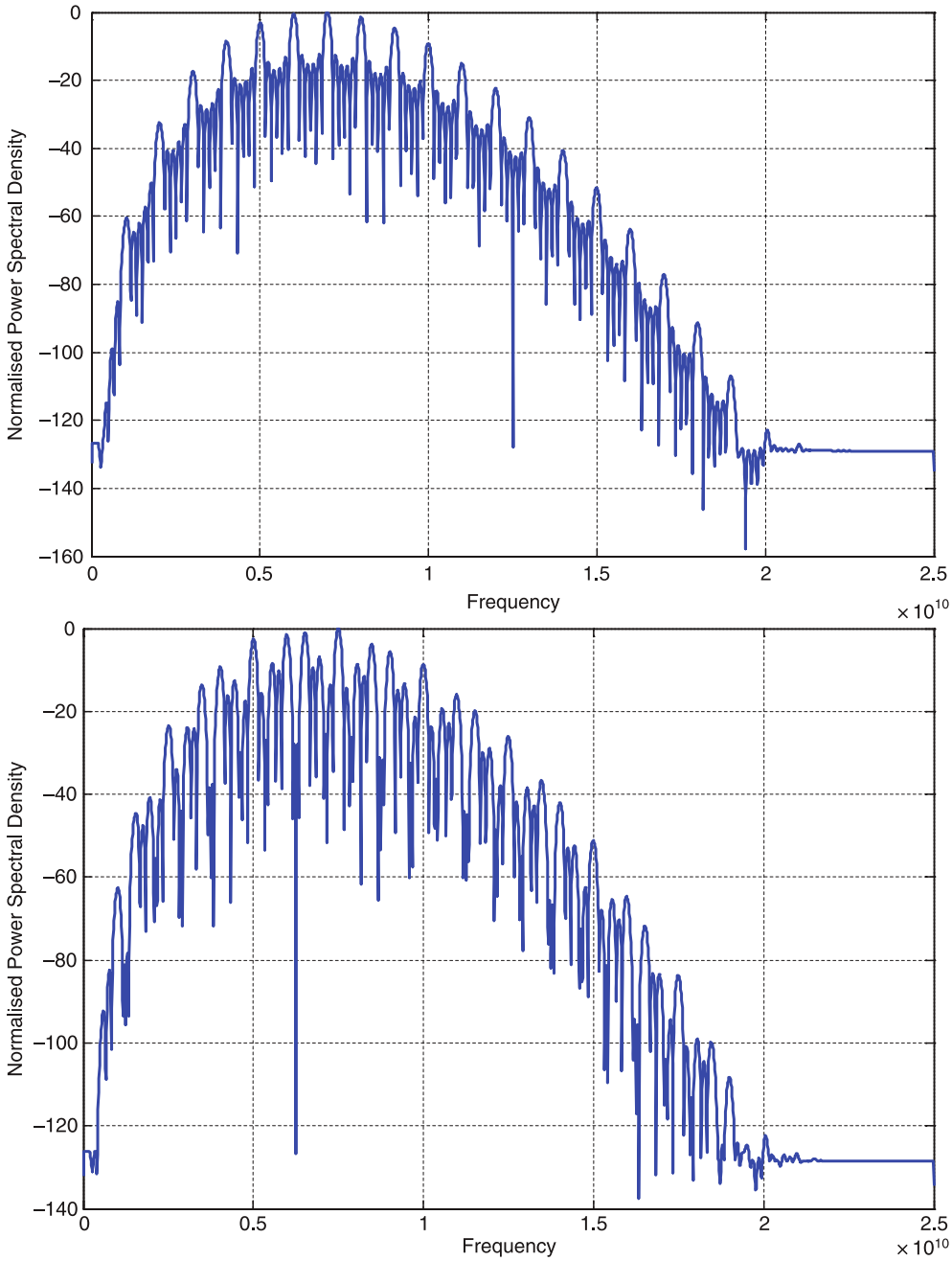


Figure 1.6. The spectrum generated by constant rate pulses (a) and the one generated by PPM (b).

- Anchor: fixed UWB transmitter/receiver, used as reference point for the positioning problem;

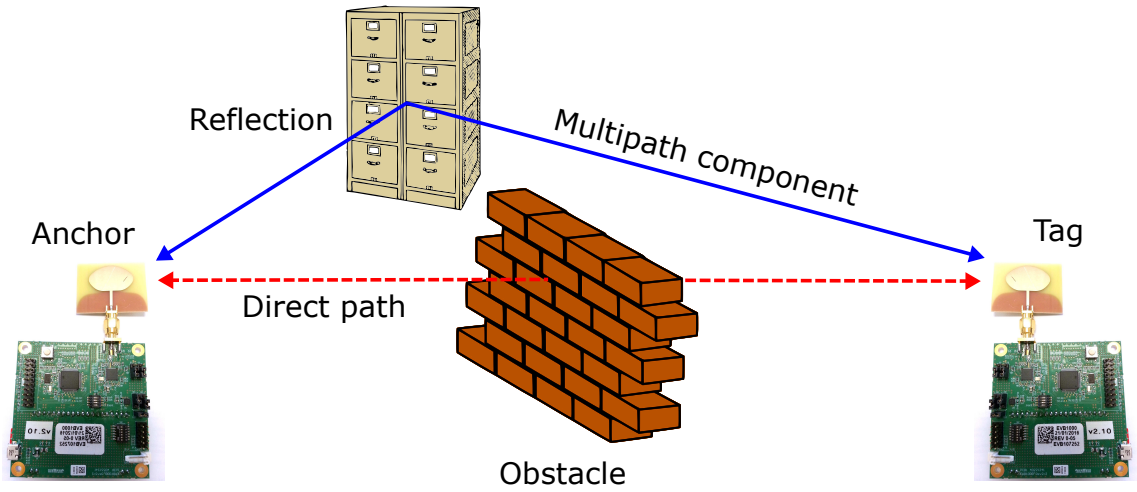


Figure 1.7. Ranging measurement setting.

- Tag: moving UWB node, attached to the point to locate and periodically communicating with all the anchors;
- Direct Path: straight line connecting an anchor to a tag;
- Obstacle: fixed or moving object obstructing the direct path between one or more anchor/tag couples;
- Multipath components: non-direct path signals (deriving from reflections, for example) arriving at the receiver.

1.2.2 Ranging Methods

There are four main methods to estimate the distance between two UWB sensors:

- Received Signal Strength (RSS), that estimates the distance from the amplitude of the received signal. An accurate knowledge of the path loss is vital for the effectiveness of the system;
- Angle of Arrival (AOA), that estimates the direction of the main signal path. Multipath components, as well as Non-Line-of-Sight, dramatically affect the result;
- Time of Arrival (TOA), that estimates the distance by measuring the propagation time of the signal. It requires synchronization between anchors and tags, and correctly identifying the first path is an important issue;
- Time Delay of Arrival (TDOA), which is a variant of TOA measuring time differences between the measurements and reducing synchronization errors.

Considering the very large bandwidth of an UWB system and its fine time resolution, TDOA and TOA methods are the most suitable and precise for this technique. In the following, the TOA method will be presented in detail and discussed.

Time of Arrival (TOA) Ranging

For the sake of simplicity, only the Line-of-Sight (LOS) condition will be discussed. In this case the phase delay of the signal is negligible, and the CIR can be modeled as

$$\sum_{n=1}^{\infty} a_n \delta(t - \tau_n) \tag{1.7}$$

The principle is intuitive: the distance d between two nodes can be estimated from the time interval s that it takes the signal to travel from one point to another. Knowing the wave speed is comparable to the speed of light c, the relation is

$$d = tc \tag{1.8}$$

t is usually computed as a difference of timestamps. Let's consider the transmission scheme in 1.8: T_{round} is the time interval between the transmission of the message from Device A and the reception of the answer, while T_{reply} is the time Device B takes to receive and send back the response. The difference between the intervals is the total travel time of the signal back and forth, so

$$T_{prop} = \frac{T_{round} - T_{reply}}{2} \tag{1.9}$$

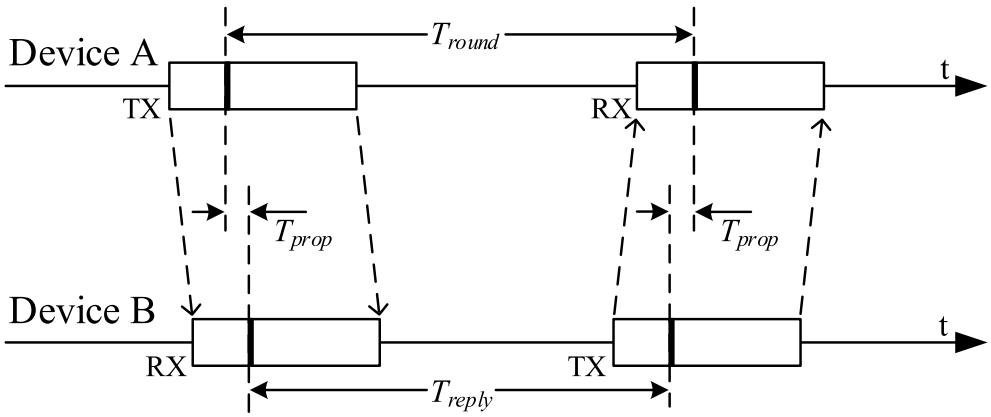


Figure 1.8. Single-Sided Two-Way Ranging (SS-TWR) transmission scheme.

This is the simplest transmission protocol and is called Single-Sided Two-Way Ranging (SS-TWR), as it needs only the transmission of two messages between Device A (the

anchor) and B (the tag). Moreover, global synchronization between the sensors is not necessary. Other more sophisticated protocols can be used to increase the accuracy of the measurements, but in most of the cases the drawback is the need to exchange more messages. An example is the Symmetrical Double-Sided Two-Way Ranging, in which one more message is exchanged. In this case, estimates are less affected by errors induced by the crystal that generates the clock.

1.2.3 Main Sources of Error

Now the principal phenomena causing errors in the range measurements will be briefly discussed [2]. The focus will be on the problem of NLOS, as it is the issue that this thesis aims at tackling.

Multipath propagation

In conventional TOA algorithms, the estimate of the time of arrival is the time shift of a template signal producing the maximum correlation with the received signal. However, in multipath environments this method is not optimal since many reflections of the transmitted signal interfere and lead to errors. Fortunately, the large bandwidth of UWB allows to distinguish multipath components from direct path without the use of complex algorithms, making it a minor issue for ranging estimations.

Multiple Access Interference

In environments in which multiple nodes are active, signals can interfere with each other and degrade the performance of the ranging method. Many techniques have been developed to tackle this problem, and most of them are based on a specific training code that helps the transmitter and the receiver to properly set-up.

High Time Resolution

The extremely large bandwidth of UWB signals enables very accurate estimations, but also creates some challenges in practical implementations. In fact, high time resolution means that phenomena like clock drift and jitter become very relevant to guarantee a certain accuracy. Moreover, it happens to be very impractical to sample the received signal at above the Nyquist rate, which is on the order of tens of GHz

Non-Line-of-Sight

It occurs in whatever case in which an obstacle blocks the direct path between tag and anchor. Considering the typical environment for service robotics applications, the obstacle could be static like a wall or a piece of furniture, or dynamic, caused by the presence of people. In this case two sub-scenarios are possible:

- Soft NLOS: the direct path of the signal arrives to the receiver with a lower amplitude than the reflections because of the attenuation given by the material;

- Hard NLOS: the direct path does not arrive with a sufficient amplitude to be distinguished from the noise, so the reflections are the first detected signals.

In this thesis both the situations will be tackled. The consequence is that the estimate of TOA, and thus the range, is generally greater than the true value as it represents the length of a non-direct path. This problem is considered as the main drawback of UWB, as a small error determination of the first path leads to a great bias on the measurements. Several techniques have been adopted to solve this problem, as it will be further discussed in Chapter 2. The most effective method consists in identifying the NLOS measures in real-time and exclude them from the positioning algorithm. Many techniques have proven to achieve very good results in the identification, starting from statistical methods to modern machine learning algorithms. The main drawback, although, is that a high number of anchors is required to obtain a univocal position even when some anchors are excluded from the computation; for this purpose, it is interesting to develop mitigation methods allowing to use all the anchors regardless the situation.

1.2.4 Trilateration

The estimation of the position of the tag from the range measurements can be performed using different algorithms, all based on a geometric problem of trilateration. Considering, for clarity, the two-dimensional case, the location (x, y) of the tag can be found at the intersection of the circles having center in the anchor position (x_i, y_i) and radius equal to the measured range d_i. This implies that, ideally, a minimum number of three anchors is needed to univocally determine the position (1.9).

In the ideal 3D case, it is sufficient to solve the non-linear system containing the equations of four circles:

$$\begin{cases} (x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2 = d_1^2 \\ (x-x_2)^2 + (y-y_2)^2 + (z-z_2)^2 = d_2^2 \\ (x-x_3)^2 + (y-y_3)^2 + (z-z_3)^2 = d_3^2 \\ (x-x_4)^2 + (y-y_4)^2 + (z-z_4)^2 = d_4^2 \end{cases} \tag{1.10}$$

However, in practical cases it is likely that the intersection is not found due to the error on the range estimates. It is also common to use five or more anchors to increase the accuracy of positioning. In both the cases the problem is no more univocally determined, as it is respectively underdetermined and overdetermined problem. This means that an optimality criterion should be found to determine the point. Typical approaches use numerical algorithms that minimize the distance between real and estimated positions.

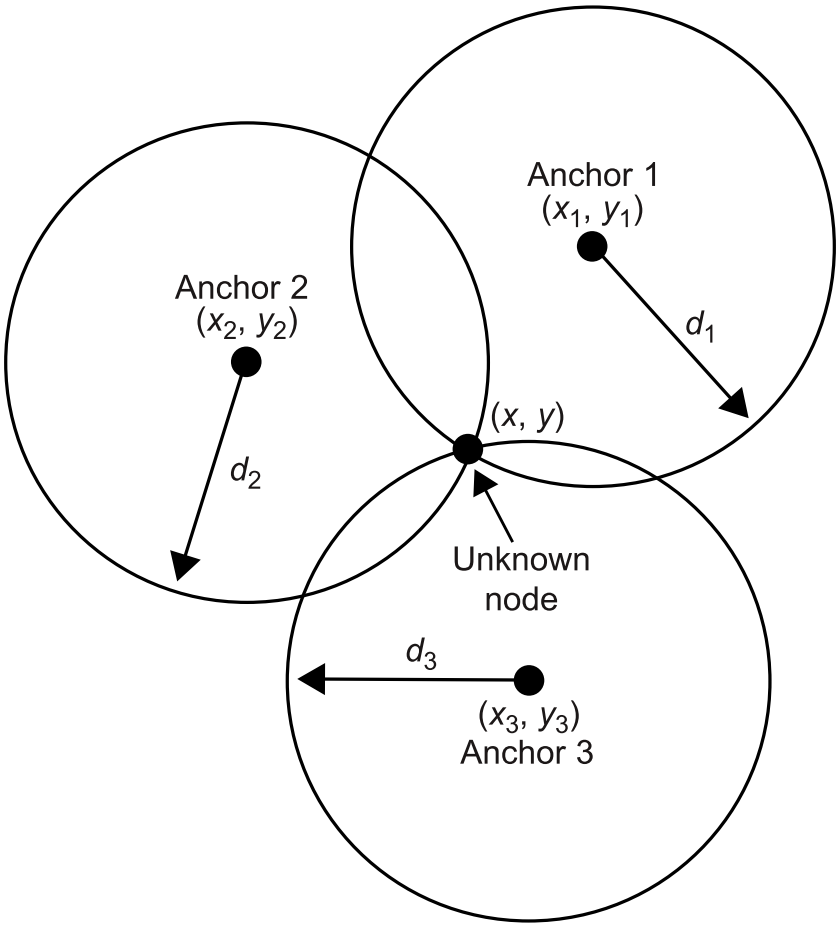


Figure 1.9. Two-dimensional positioning problem using three anchors.

Gauss-Newton

The Gauss-Newton method is an iterative process used to solve nonlinear problems of least-square approximation. The goal is to minimize the sum of the squares of the residuals

$$S = \sum_{i=1}^{m} r_i^2 \tag{1.11}$$

Where m is the number of anchors and r_i is the residual, defined as

$$r_i = d_i - \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} \tag{1.12}$$

d_i is the distance between anchor i and the tag, $\mathbf{x} = [x, y, z]$ is the position vector of the tag and $\mathbf{x}_i = [x_i, y_i, z_i]$ the position vector of the i^{th} anchor. At each step, starting from an initial estimate $\mathbf{x}_0$, the algorithm computes the value $\mathbf{x}_{k+1}$ as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{r}(\mathbf{x}_k) \tag{1.13}$$

Where $\mathbf{r}$ is the vector of the residuals and $\mathbf{J}$ is the Jacobian matrix of the system.

$$\mathbf{J} = \begin{bmatrix} \frac{-(x_k-x_1)}{\sqrt{(x-x_1)^2+(y-y_1)^2+(z-z_1)^2}} & \cdots & \frac{-(z_k-z_1)}{\sqrt{(x-x_1)^2+(y-y_1)^2+(z-z_1)^2}} \\ \vdots & \ddots & \vdots \\ \frac{-(x_k-x_m)}{\sqrt{(x-x_m)^2+(y-y_m)^2+(z-z_m)^2}} & \cdots & \frac{-(z_k-z_m)}{\sqrt{(x-x_m)^2+(y-y_m)^2+(z-z_m)^2}} \end{bmatrix} \tag{1.14}$$

The algorithm stops once a certain precision threshold is reached, or the number of iterations exceeds the imposed limit. As many other estimation algorithms, the choice of the initial estimate is crucial for the convergence.

Extended Kalman Filter

The Kalman filter (KF) is "an optimal Bayesian recursive estimator for linear dynamic systems from noisy measurements" [3]. The generic state equations for a dynamic system are

$$\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{v}_{k-1}), \mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k), \tag{1.15}$$

Where $\mathbf{x}$ is the system state, $\mathbf{u}$ is the input and $\mathbf{z}$ is the measured output. $\mathbf{f}$ and $\mathbf{h}$ are nonlinear functions and $\mathbf{v}$, $\mathbf{w}$ are noises. To apply standard KF, the following assumptions must be made:

- The disturbances $\mathbf{v}_k$ and $\mathbf{w}_k$ are gaussian density functions of known parameters and are additive

$$\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k),\ \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k) \tag{1.16}$$

- The function $\mathbf{f}_{k-1}$ is known and linear with respect to $(\mathbf{x}_{k-1}, \mathbf{u}_{k-1})$ and $\mathbf{v}_{k-1}$
- The function $\mathbf{h}_k$ is known and linear with respect to $\mathbf{x}_k$ and $\mathbf{w}_k$

In cases where the dependency between the state and the measurements is nonlinear, like positioning, more complex methods must be used. The Extended Kalman Filter aims at achieving optimal estimation linearizing the state equations through a Taylor expansion around a certain mean value and apply the linear Kalman Filter to this linearized model. The algorithm is a succession of prediction and update. At each time step:

1. Prediction:
   (a) Estimate the predicted state

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \cdot \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k, \mathbf{u}_k \tag{1.17}$$

Where $\mathbf{B}$ is the input matrix and $\mathbf{F}$ is the linearized state matrix defined as

$$\mathbf{F}_k = \nabla_{\mathbf{x}} \mathbf{f}_k(\mathbf{x}, \mathbf{u})|_{\mathbf{x}=\mathbf{x}'} \tag{1.18}$$

That is the Jacobian of $\mathbf{f}(\mathbf{x}, \mathbf{u})$ with respect on $\mathbf{x}$, taking $\mathbf{x}'$ as linearization point

(b) Estimate the predicted covariance matrix $\hat{\mathbf{P}}$ based on prior knowledge

$$\hat{\mathbf{P}}_{k|k-1} = \mathbf{F}_k \cdot \hat{\mathbf{P}}_{k-1|k-1} \cdot \mathbf{F}_k^T + \mathbf{Q}_k \tag{1.19}$$

2. Update:

(a) Compute the innovation as the difference between the measurement and its prediction

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}) \tag{1.20}$$

(b) Compute the Kalman gain matrix

$$\mathbf{K}_k = \hat{\mathbf{P}}_{k|k-1} \cdot \mathbf{H}_k^T \cdot (\mathbf{H}_k \cdot \hat{\mathbf{P}}_{k|k-1} \cdot \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \tag{1.21}$$

Where $\mathbf{R}$ represents the covariance matrix related to the observation vector, while $\mathbf{H}$ is the linearized observation matrix defined as

$$\mathbf{H}_k = \nabla_{\mathbf{x}} \mathbf{h}_k(\mathbf{x}) \tag{1.22}$$

(c) Compute the a posteriori state estimate

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \cdot \tilde{\mathbf{y}}_k \tag{1.23}$$

(d) Compute the a posteriori state covariance matrix

$$\hat{\mathbf{P}}_{k|k} = (\mathbf{I}_n - \mathbf{K}_k \cdot \mathbf{H}_k) \cdot \hat{\mathbf{P}}_{k|k-1} \tag{1.24}$$

Where $\mathbf{I}_n$ is the identity matrix

1.3 Deep Learning

1.3.1 Definition of Machine Learning

According to [4], the engineering-oriented definition of machine learning was originally created by computer scientist Tom Mitchell in 1997: "A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E". Machine learning is,

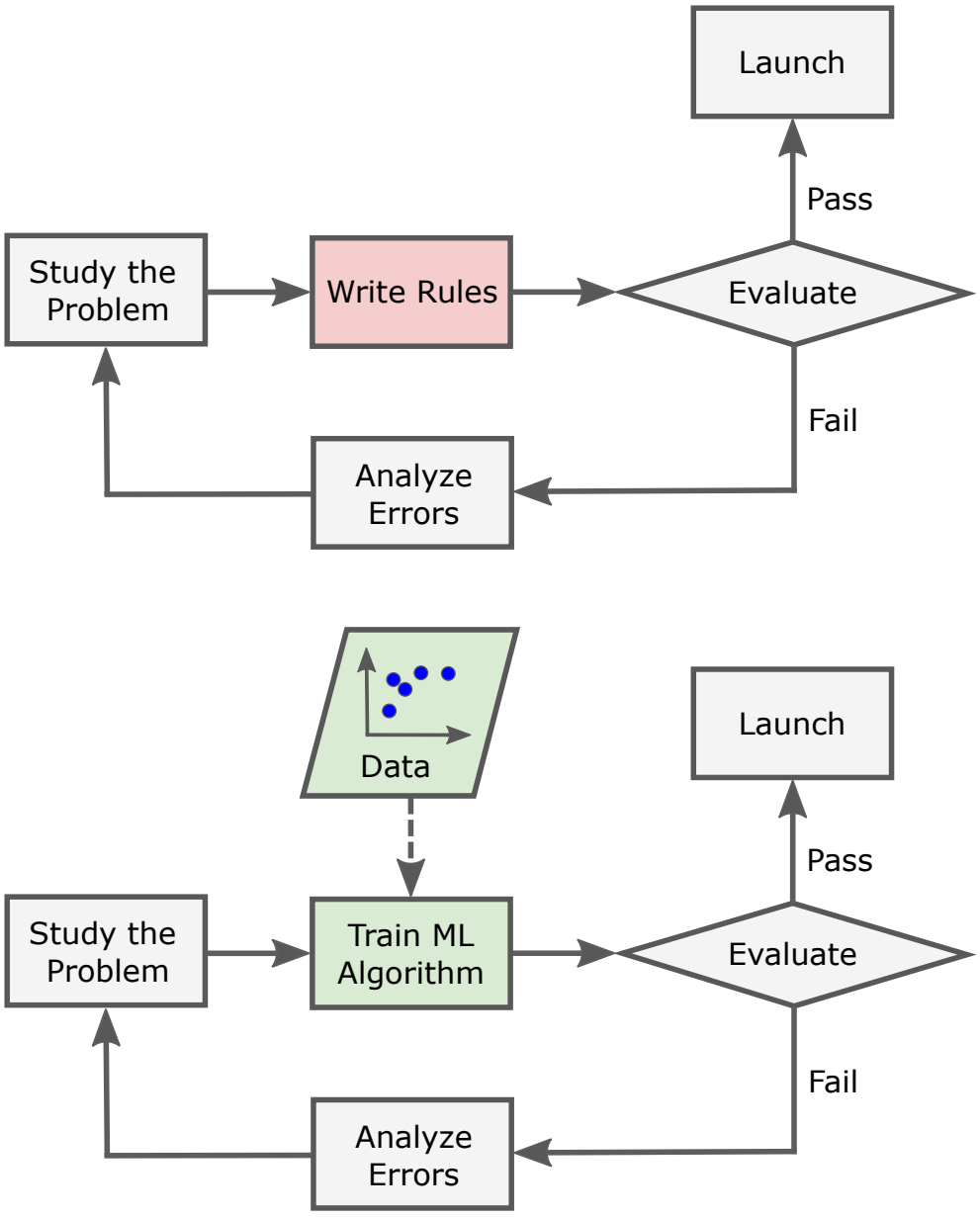


Figure 1.10. The different learning cycles for the classic approach and ML.

in practice, the ability of computers to learn without being explicitly programmed but extracting information from examples and deriving more general conclusions by induction.

As can be seen in 1.10, the main advantage with respect to classical programming and machine learning is that the optimization cycle, called training, is automatic, and this allows to push optimization to unpreceded levels without manually altering the model. Obviously, the more the problem is complex, the higher amount of data is required to

properly train the algorithm. A correct selection of the data and of the evaluation strategy is of paramount importance and should be assessed during the preliminary study of the problem. There are different types of machine learning systems, based on the following criteria:

- Supervised/Unsupervised, whether they are trained with the supervision of a human;
- Online/Batch learning, whether they learn incrementally on the run;
- Instance-based/Model-based, whether they work just comparing new data to known data or try to detect patterns and build a predictive model;
- Classification/Regression, whether the model predicts a discrete label for each data point or a continuous value.

1.3.2 Challenges in Machine Learning

As said before, the potentiality of machine learning is enormous and can span through many different contexts. However, some critical issues must be considered and correctly managed to achieve good results with ML methods.

Quantity, Quality and Representativeness of Data

First, machine learning has its foundation on data, so it is paramount importance that the learning process feeds the algorithm with a reasonable amount of good-quality data. In 2001, a research from Microsoft [5] showed that even simple algorithms can achieve very good performance if they are given enough data. This led the authors to reconsider the trade-off between algorithm and dataset development.
On the other hand, it is not always easy, or at least cheap, to build a dataset big enough to show such results. Small and medium datasets are very common, so it is often more feasible to work on an optimal learning model than collecting millions of data-points. Nevertheless, a large amount of data does not always guarantee a correct representation of the phenomenon under study. In fact, the dataset should also guarantee representativeness, i.e. a correct coverage and balance between all the possible cases. A model trained on a set that is plenty of examples extracted under the same conditions will be biased to work well only under those conditions and be bad on something it is not prepared for. The more an algorithm is expected to generalize the examples, the more it should be trained on data from different conditions.
Another important rule to follow is to select only the meaningful features from the data before feeding it into the algorithm and discard the futile ones. The process of feature engineering is crucial not to get the model stuck into trying to learn from irrelevant data and emphasize the important markers.

Overfitting and Underfitting

Overfitting is one of the most common problems in modern machine learning algorithms, in which huge quantities of data are used to train very complex models. When the algorithms try to fit the data with the best possible accuracy it models very fine trends in the samples. Doing so, wrong generalizations can occur, either caused by chance (like irrelevant features, for example) or noise in the data. This leads to critical errors in the testing phase where a simpler model could have performed better. An example can be observed in 1.11, where the goal is separate the blue data points from the red ones. The correctly fitting model is represented by the black line and leaves some outliers in the wrong half of the plane. The green line, however, puts every point in the right-side overfitting the data: it models something that is casual or irrelevant (in this case noise) leading to a loss of accuracy in testing.
Underfitting is the opposite problem, although it happens more rarely. It occurs when a too simple model is used to fit the data, resulting in a very bad performance also on the training set. In 1.11, underfitting is represented by the orange line.

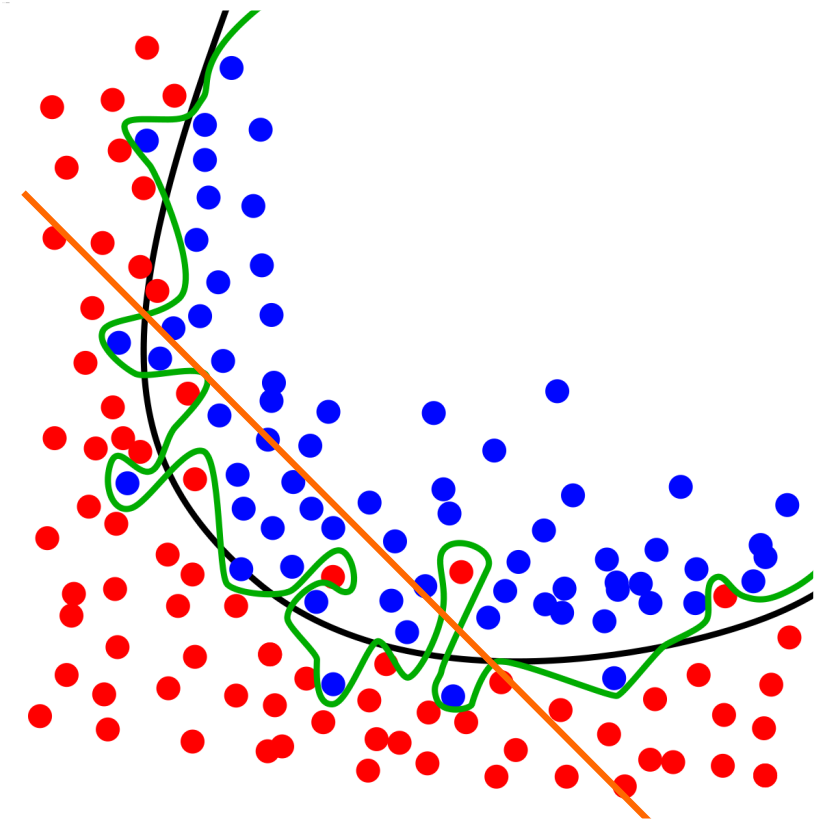


Figure 1.11. Example of correct fitting (black), overfitting (green) and underfitting (orange).

1.3.3 Common Learning Algorithms

In the following paragraphs, the most used machine learning algorithms will be briefly explained, starting from the simpler ones and going towards more recent methods. The explanation aims at giving some context about the main techniques that can be found in literature.

Linear Regressor

It is the simplest learning model possible since it is realized by a weighted sum of the input features. The weights are the tunable parameters of the model and an additional bias term could be present.

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{1.25}$$

In 1.25 $\hat{y}$ is the predicted value, n is the number of features, x_i is the i-th feature value and θ_i is the j-th model parameter (including the bias θ_0 and all the feature weights). The expression can also be written in vectorial form as follows:

$$\hat{y} = h_{\boldsymbol{\theta}(\boldsymbol{x})} = \boldsymbol{\theta} \cdot \mathbf{x} \tag{1.26}$$

To train this model properly we need to find the value of $\boldsymbol{\theta}$ that minimizes the error of the predictions. The most used metric for this purpose is the mean squared error (MSE):

$$\text{MSE}(\mathbf{X}, \boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2 \tag{1.27}$$

So, the optimization problem to solve is the following:

$$\hat{\boldsymbol{\theta}} = arg \min_{\boldsymbol{\theta}} MSE(\boldsymbol{\theta}) \tag{1.28}$$

Since the model is linear, the functional associated to this problem is convex. This means that the absolute minimum can be computed in closed form, and corresponds to the so-called least square (or normal equation) solution:

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T y \tag{1.29}$$

Where $\hat{\boldsymbol{\theta}}$ is the value of $\boldsymbol{\theta}$ that minimizes the cost function and y is the vector of target data-points. This method is very light and fast, and leads to good results in simple cases, where a linear model is able to describe the data under study. This is not always possible, so kernels and nonlinear models have been developed.

Support Vector Machine

A Support Vector Machine (SVM) is a powerful tool to perform linear or nonlinear classification and regression. Its base concept is described in 1.12a: the algorithm finds the largest possible strip of plane between the data points on the margin. This method is computationally very effective because it only uses the closest points (called support vectors) and ignores the others. When the complete separation of the classes is not possible, an approach called soft margin classification must be applied: the outliers are allowed to stay on the wrong side of the margin and the algorithm tries to minimize their number (1.12b). When the data is not linearly separable, a kernel trick must be used to project features in a new data space and obtain a nonlinear classifier.

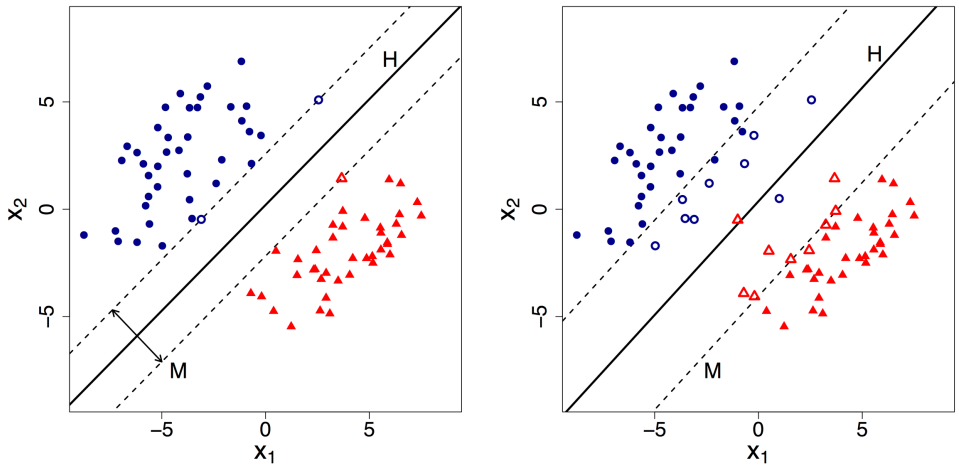


Figure 1.12. Example of SVM for classification: hard margin (a) and soft margin (b).

K-Nearest Neighbor

KNN is a non-parametric method that can be used for classification or regression. The input for the algorithm consists of the k closest data-points to the sample under test in the feature space. In case of classification, the point is assigned the most frequent label among the neighbors, while for regression the output is the average of the values of the neighbors. The algorithm relies on the concept of distance, so the choice of the metric is critical as well as data normalization. A peculiarity of this method is that it is affected by the local structure of the data in different ways varying the value of k 1.13.

1.3.4 Convolutional Neural Networks

Convolutional Neural Networks first came out in the 1980s from the study of the brain's visual cortex. In the last decade, thanks to the outstanding progress of technology lead to a

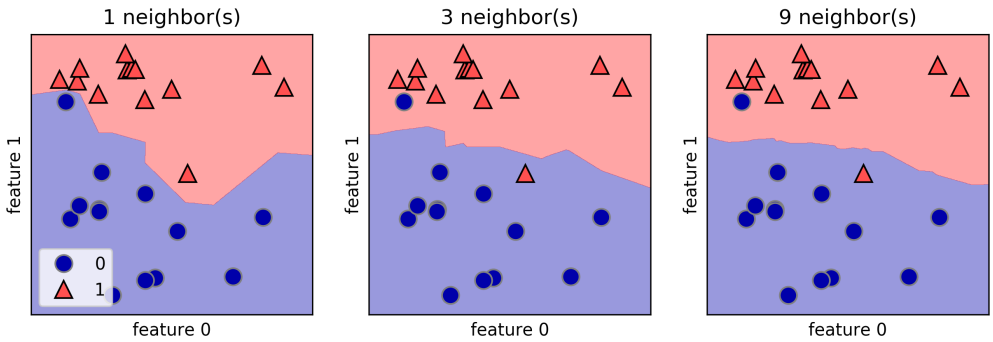


Figure 1.13. Example of KNN classification for different values of k.

great increase of the computational power of machines. This made possible to train CNN with great amounts of data reaching superhuman performance on complex visual tasks like classification, segmentation and identification. One of the most impressive results are the one reached by autonomous driving systems and natural language processing. In this section the basic building blocks and the typical project workflow will be presented.

Convolutional Layer

The most important block of a CNN is the convolutional layer (1.14). Its goal is to extract features from the input and optimization makes it possible to learn how to do it in the best way. Each pixel on the layer is connected to a specific receptive field, concentrating the input in small low-level features. Stacking convolutional layers more complex (or high level) features can be extracted.

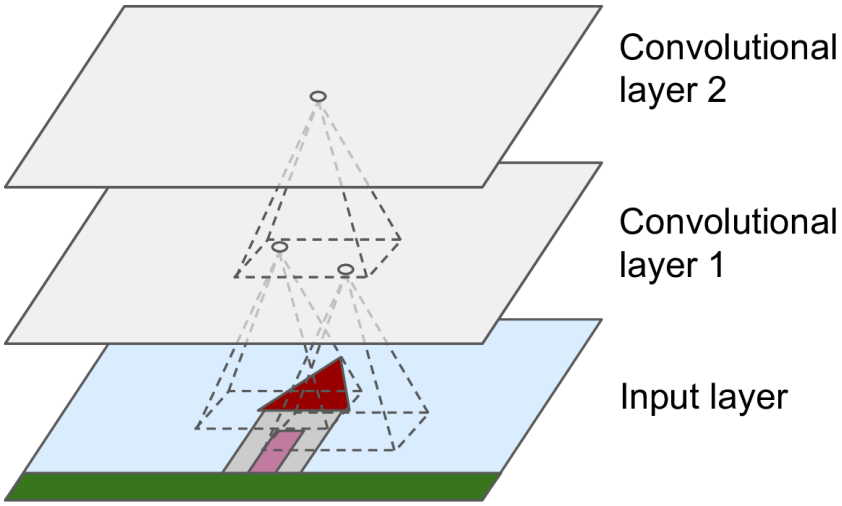


Figure 1.14. Convolutional layers with rectangular receptive fields [4].

This method leverages the fact that different levels of structure are present in an image: the direction of lines, the dimension of objects, their disposition, many types of patterns can be found and elaborated. Each cell weights its convolutions with a certain value that is learnt through the training phase thanks to a mechanism called backpropagation.

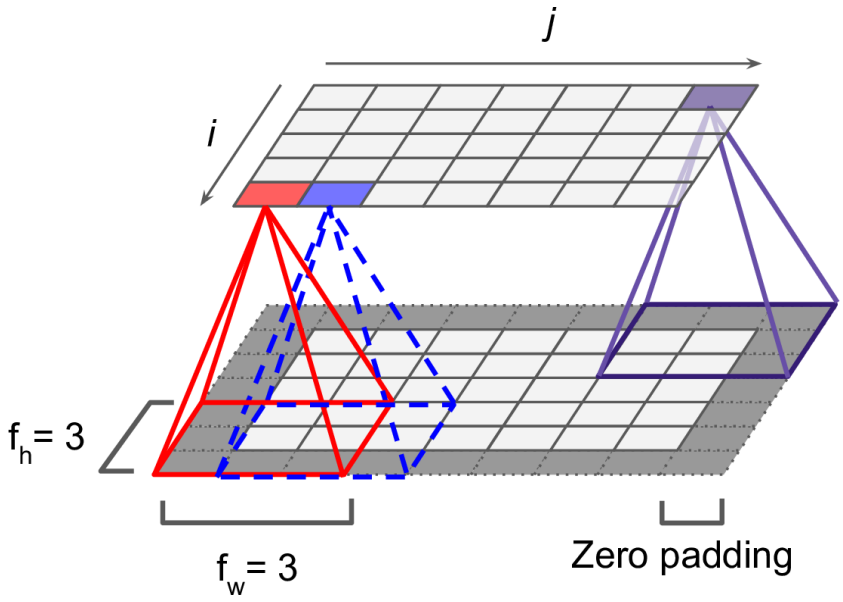


Figure 1.15. Connection between layers, filter dimensions and zero padding [4]

Referring to 1.15, f_h and f_w are the dimensions of the receptive field, i and j are the dimensions of the layer. The darker pixels represent the so called "zero padding": to feed each pixel with the correct number of inputs, a contour of zeros is added to the image. The distance between the receptive fields of adjacent pixels is called stride (in this case equal to 1): when a reduction of the number of parameters is needed, strides greater than one can be adopted, leading to a less complex model. Images are usually not made of a single layer: in fact, color images contain three levels, one for the red channel, one for blue and one for green. In the same way, convolutional layers can be designed to contain multiple filters, increasing the variety of extracted features and, of course, computational weight.

Pooling Layer

Another way to reduce the complexity of the model is pooling. It is usually performed after feature extraction (so after convolutional layers) and consists in keeping only the features showing the highest values or "activations". So, for each group of pixels the most significant value is considered, and all the others are deleted. This method is called Max Pooling and significantly reduces the dimensions of the net. Other pooling techniques are

possible: for example, Average Pooling considers the mean value among the considered space.

Fully Connected Layer

Fully connected (FC) layers connect every neuron from one layer to every neuron of the next (1.16). This creates 2^n connections, where n is the number of neurons of the layer. FC layers are used to process the features extracted by the previous layers, and generally end with a single neuron preparing the output to be generated. Every neuron holds a trainable weight.

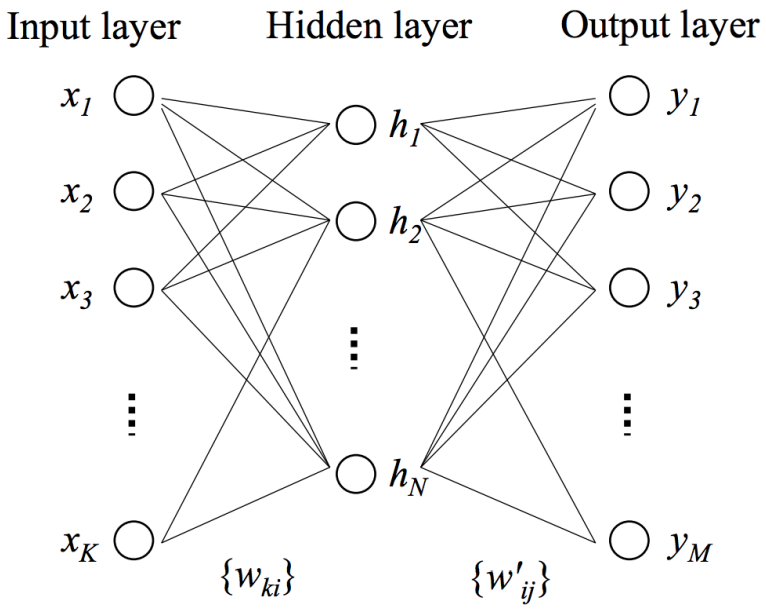


Figure 1.16. Fully connected layer with N neurons.

Activation Function

Activation functions determine the output of a certain layer, determining not only the overall output of the algorithm, but also its convergence speed. They are mathematical expressions that regulate neurons' activation, such that their output value is high if the input is relevant. Typically, the output is rescaled between 0 and 1 or between -1 and 1. Computational efficiency is of fundamental importance, as activation functions must be run for each neuron: for this reason, many different functions have been developed to achieve the best trade-off between performance and simplicity. Some examples can be found in 1.17

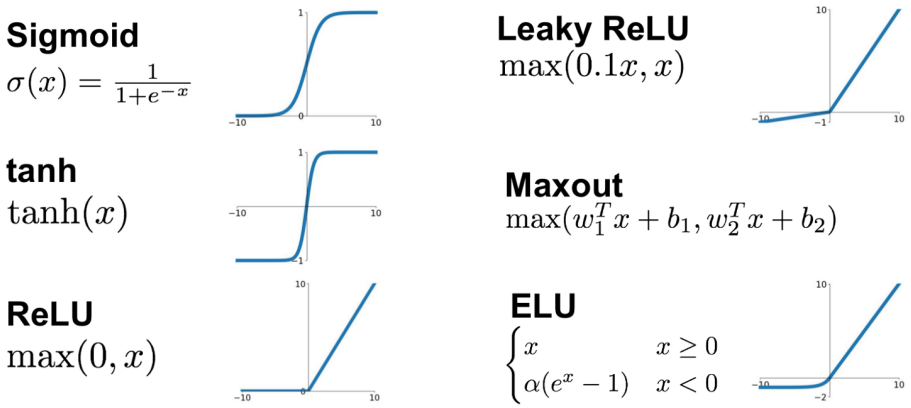


Figure 1.17. Some of the most used activation functions

Loss

To make the neural network work effectively, all the weights present in the structure must be optimized in the so called "training phase". In this phase, the net is fed with a huge number of examples and its ability of outputting the expected result is measured by some loss function. Like activation functions, losses must be fast to compute and precise in describing how the model is wrong with respect to the collected data. The most common loss functions are:

- MSE (Mean Squared Error)

$$\text{MSE}\ (y, f(x)) = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - y^{(i)})^2 \tag{1.30}$$

- MAE (Mean Absolute Error)

$$\text{MAE}\ (y, f(x)) = \frac{1}{m}\sum_{i=1}^{m}|x^{(i)} - y^{(i)}| \tag{1.31}$$

- Huber loss

$$H(y, f(x)) = \begin{cases} \frac{1}{2}[y - f(x)]^2 & \text{for } |y - f(x)| \leq \delta, \\ \delta\,(|y - f(x)| - \delta/2) & \text{otherwise.} \end{cases} \tag{1.32}$$

Optimizer

The optimizer has the responsibility of updating the weight of every neuron at each training step through backpropagation. As the goal is to reach the absolute minimum of the loss function

$$W = arg\min_{W} \text{MSE}(W) \tag{1.33}$$

the optimizer typically exploits the gradient of the function to descend towards lower and lower minima.

$$W_{t+1} = W_t - \eta \nabla W \tag{1.34}$$

where W is the parameter of the model, η is a weight applied to the correction and called Learning Rate (LR), and δW is the partial derivative of the loss function with respect to W. This approach is called Gradient Descent (GD) and is the base of many other optimizers, which add more sophisticated mechanisms to guarantee a good convergence in short time. The most used optimizer nowadays is Adam [6], which adds to the standard GD three main features:

- Stochasticity: instead of running the optimizer on all the data points at each step, a small set of them is considered. This dramatically reduces computational time and still guarantees fast convergence even if the computed gradient is just an approximation of the real one.
- Per-parameter Learning Rate: the learning rate is not the same for all the parameters, so the optimizer updates some of them faster than others. This is very effectful for computer vision problems, in which sparse gradients are frequent.
- Momentum: to prevent the optimizer from getting stuck in local minima, the step of each instant is influenced also on past gradient values.

1.3.5 Deep Learning

Now that the main blocks and strategies used in Convolutional Neural Networks have been presented, a short recap of how "deep" and "very deep" convolutional networks evolved in the last years evolved is addressed. As mentioned before, progress in computational power of machines lead to a growth in model complexity. For what concerns CNNs, this meant deepness: stacking more and more layers allows to extract more meaningful features from data and reach higher accuracy in many tasks. The first outstanding result in this direction was achieved by Alex Krizhevsky in 2012 with AlexNet [7]. The primary discover was that depth is essential for high performance models, and the second was that such deepness can be feasible using graphic processing units (GPUs) for training. As can be seen in 1.18, the net used multiple convolutional layers, max pooling and fully connected layers before the output.

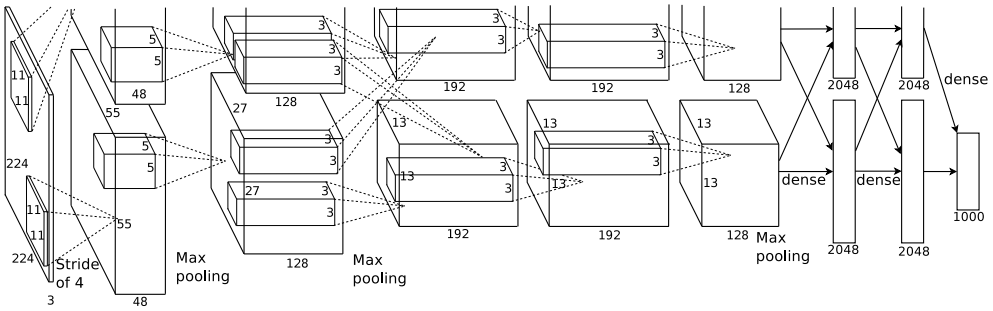


Figure 1.18. Structure of Alexnet

AlexNet outperformed every other image classification algorithm, drastically raising the bar for every machine learning competition. In the next few years an explosion in deep learning research lead to the development of several new methods and structures for neural networks, that will be mentioned in the next chapters.

Chapter 2

State of the Art

This chapter is dedicated to the presentation of the techniques that have been developed in the last years to tackle the problem of NLOS for positioning purposes. The papers that will be referenced are the theoretical bases on which this thesis is grounded, and the starting point for the development of a new and better mitigation method. As the two aspects are strongly correlated, both classification and mitigation techniques are presented, alongside some recent advance in neural networks that has been adopted for the design.

2.1 NLOS Classification

As mentioned before, one first way to compensate NLOS errors is to properly identify what range measurements are affected by error and exclude them from the computation of the position estimate. This technique proved to work fine in many scenarios but shows a strong drawback: it needs many more anchors than the minimum, and still does not provide a good solution for the limit cases in which the anchors to exclude are too much. In such a limit case the number of available range measurements could be insufficient for the determination of the position. Anyways, if the anchor position is chosen wisely that should be unlikely to happen, so classification can still be a good choice for some applications. Here some of the most relevant contributions present in literature are explained and commented, starting from classic statistical approaches, to shallow machine learning methods and recent deep learning experiments.

2.1.1 Statistical Approach

The very first methods that aimed at performing classification relied only on simple data like received signal strength (RSS) and total signal energy, trying to fix some simple rule to distinguish LOS from NLOS measurements without too much computation. It is the case of [8], in which a threshold on RSS does the job, obtaining good results. The value of the threshold, though, is strongly dependent on the measurement scenario and

the distances between the tag and the anchors.
Then, other statistical strategies have been developed: [9] applies Bayesian methods based on the estimation of the density function of the data, while [10] introduces a fingerprint-based algorithm using a vocabulary. For each measurement, sample signals are compared to the CIR to determine the channel type.

2.1.2 Shallow Learning

The explosion of machine learning in the last decade has led to the development of novel techniques to achieve better and better results. Shallow learning algorithms do not require huge amounts of data, so tools like KNN and others have become largely used. Among the most relevant ones, SVMs have been adopted by [11] and [12] and used in combination with a least squares (LS) algorithm by [13] and [14]. For [15], instead, the AdaBoost (Adaptive Boost) technique has been used [16]. It consists in a meta-algorithm that transforms several low-accuracy methods, called "weak learners", in a more performing "strong learner". To do so, it corrects the errors in the learners and keeps their strength. In [17] a Multi-Layer Perceptron (MLP) starts introducing some deepness with a very small neural network, although a Binary Decision Tree (BDT) still performs better with an accuracy of 87%.
The main flaw of shallow algorithms, although they perform very well for classification, is that the features need to be extracted manually from the CIR signal. The choice of the features is crucial for the effectiveness of the method and can depend on the scenario: the most used are:

- Received Signal Power Level

$$\mathrm{RSL} = 10\, log_{10}\left(\frac{C \cdot 2^{17}}{N^2}\right) - A_{\mathrm{dBm}} \tag{2.1}$$

- Received Signal Power to First Path Power Level

$$\mathrm{RFPR} = \mathrm{RSL} - \mathrm{FSL} \tag{2.2}$$

  where FSL is the estimated First path Signal power Level, computed as

$$\mathrm{FSL} = 10\, log_{10}\left(\frac{F_1^2 + F_2^2 + F_3^2}{N^2}\right) - A_{\mathrm{dBm}} \tag{2.3}$$

- Signal Energy

$$\epsilon_r = \int_T |r(t)|^2 dt \tag{2.4}$$

- Mean Excess Delay Spread

$$\tau_{\text{MED}} = \int_T t \frac{|r(t)|^2}{\epsilon_r} dt \tag{2.5}$$

- Variance

$$\sigma^2_{|r|} = \frac{1}{T} \int_T \left[|r(t)| - \mu_{|r|} \right]^2 dt \tag{2.6}$$

- Mean Value

$$\mu_r = \frac{1}{T} \int_T |r(t)| dt \tag{2.7}$$

- Kurtosis

$$\kappa = \frac{1}{\sigma^4_{|r|} T} \int_T \left[|r(t)| - \mu_{|r|} \right]^4 dt \tag{2.8}$$

- Amplitude

$$A = \max \left(|r(t)| \right) \tag{2.9}$$

2.1.3 Deep Learning

Since shallow algorithms reach very high accuracy values, not many deep learning has been used for classification if not for the previous issue regarding features. Nevertheless, neural networks demonstrated to allow even better performances in many scenarios. In [18] a CNN is used and compared with SVMs. The result is a 20% to 45% improvement in overall classification (2.1). A Recurrent Neural Network (RNN) is applied for classification in [19], yielding very high accuracy. An RNN is a neural network in which the outputs of upper layers are used as inputs for lower ones, transforming a layer in a memory cell. The presence of feedback connections builds the so-called Long Short-Term Memory (LSTM) structure.

2.2 Error Compensation

Trying to compensate the error caused by NLOS condition is a way harder task for any algorithm. When simply identifying the problem can be tackled efficiently from simple models, a regressor that traces it back to the measurement error has to take into account multiple factors: different materials give different delays to the signal, and the walls that

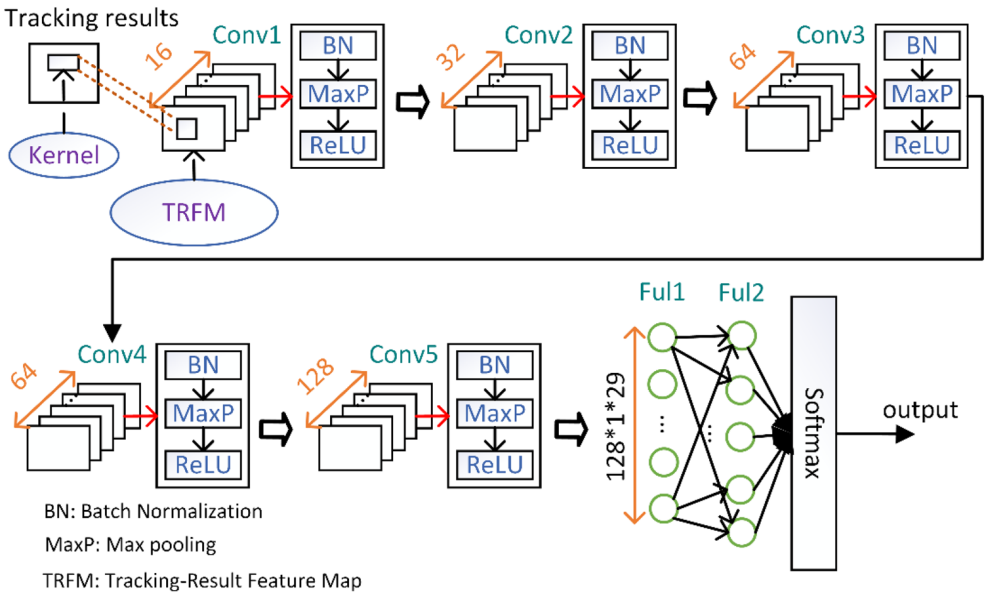


Figure 2.1. Structure of the CNN used for classification in [18]

waves find on their way generate different reflections depending on their position and orientation. This leads to a wide variety of situations that should be covered by a method that aims at generalizing the phenomenon, but still this goal does not seem to be achieved from modern techniques, although good results have been reached for the single scenario.

2.2.1 Statistical Approach

Kalman Filter (KF), in many variants, is one of the most used statistical methods adopted for mitigation. In [20] KF is aided by a machine learning classification method reaching good results. Unscented Kalman Filters (UKF) are adopted in [21]: it is a nonlinear extension of KF based on unscented transformations. In [22] a Schmidt-Kalman filter (SKF) is used to reduce the complexity that a standard KF would carry to the computation and still yield acceptable results. Finally, a Biased Kalman Filter (BKF) is adopted in [23], demonstrating the effectiveness of the method. Moreover, other methods have proven to improve the precision of ranging, like fuzzy theory [24], non-parametric error modelling [25] and expectation maximization [26].

2.2.2 Shallow Learning

Like for classification, machine learning allows to reach much higher performance than classic models and has been largely experimented. In [27] the idea of a kernel-based regressor is applied guaranteeing robustness to fingerprinting approaches. A Relevance Vector Machine (RVM) is used in [28], while a least square version of SVM is developed

in [13]. A complete comparison between SVMs and Gaussian Processes (GP) is conducted in [29] and [30], showing that such non-parametric techniques have the potential to significantly improve localization. In [31] all the main shallow methods are considered and compared: Binary Decision Tree, SVM, KNN, Gaussian Process and Generalized Linear Models.
This paper proved that unless some improvement can be obtained, mitigation is a much more challenge with respect to classification, and that results strongly depend on the measurement scenario. This discourages from using shallow methods because the undue specificity could be derived from the fact that the features are extracted manually: some higher-level patterns in the CIR may hide the key to generalize the phenomenon.

2.2.3 Deep Learning

This last consideration led many researchers to start big measurement campaigns and experiment on deeper algorithms. In [32] an artificial neural network is used: it consists in a deep stack of fully connected layers intercut with batch normalization layers and ReLU activation functions (2.2).

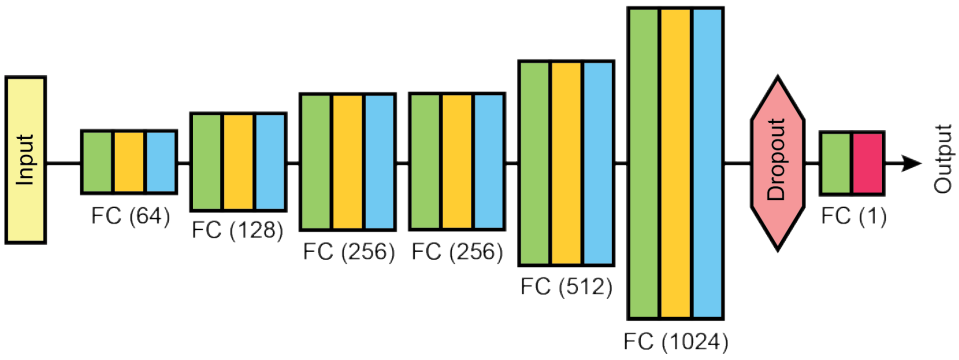


Figure 2.2. Structure of the ANN used for mitigation in [32]

Batch normalization is a method developed in 2015 to make NNs work faster: it scales and center the output of each layer mitigating the problem of internal covariate shift [33]. Convolutional Neural Networks are adopted in some recent papers: in [34] it is showed that mitigation is not very effective in environments different from the one in which training takes place, [35] applies specifically to the error given by the orientation of the antenna, showing good results, and [36] proposes a direct position estimation for enhanced precision, but not focusing on the NLOS issue.
Many methods try to combine Neural Networks with other techniques. In [37], for example, a NN is combined to fingerprinting using raytracing, but this implies to have a detailed description of the environment and, hence, is not easily generalizable to other situations. Same consideration can be made for [38], that combines fingerprinting with

a Deep Belief Network (DBN), and [39], that uses a Supervised Autoencoder (SAE) to model the environment in which the positioning takes place. An autoencoder is a neural network that learns to copy its input to the output, encoding and then decoding data. The inner layer holds the feature representation of the input and carries the meaningful information. Teaching the SAE to model and reproduce CIR signals, it learns intrinsic properties about the transmission channels, but they are still too specific to be translated to a different context. In [40], instead, a CNN is applied jointly with a Weighted Least Squares positioning algorithm.

Although this method achieves good results in improving position accuracy, the authors state that the tests have been conducted in the same environment in which the dataset was built, not guaranteeing generality.

Chapter 3

Experimental Set

As seen in Chapter 2, all the NLOS mitigation techniques suffer from the fact that results are strongly dependent on the measurement scenario and difficult to generalize. For this reason, the goal of this thesis is to develop a novel mitigation method that goes deeper and extracts higher level features: in this way, it is possible to achieve a better scalability of the model to different environments. As deeper learning techniques require larger amounts of data, the first step is collecting a sufficiently big set of ranging measurements in the first experimental scenario further described in this chapter. Then, the meaningful information must be extracted and prepared to be used from the mitigation algorithm. After the algorithm has been properly trained and tuned, the method is tested "online", that is in a real scenario in which the position of a tag has to be estimated in real time. This scenario is described and commented further on. Finally, some issues regarding the sensors are reported, pointing at the critical aspects and possible workarounds.

3.1 Hardware and Software

3.1.1 Decawave TREK1000

TREK1000 is an UWB positioning evaluation kit from DecaWave (3.1). As demonstrated in [41], this system has the best performance compared with other systems available on the market. Each TREK100 kit contains four EVB1000 units that can operate as anchors or tags. Each board is equipped with DecaWave's DW1000 IEEE802.15.4-2011 UWB compliant wireless transceiver Integrated Circuit (IC), STM32F105 ARM Cortex M3 processor, micro-USB interface, LCD display and off-board antenna. The system has a minimum accuracy of ±10 cm using two-way ranging time-of-flight technique. The minimum positioning accuracy estimated by the manufacturer is ±30 cm for a moving tag and the nominal maximum measurable distance in LOS condition is about 300 m. The EVB1000 boards have different working modes that can be configured using dip-switches. It is possible to choose between two channels (2 and 5), which have two different central

frequencies (3.99 GHz and 6.48 GHz), and two data rates (110 kbps or 6.8 Mbps). Table 3.1.1 contains all the details about the board: the default setting (3.99 GHz and 110 kbps) is suggested for long distance ranging and it guarantees an update rate of 3.5 Hz confirmed by experimental tests. The TREK1000 kit is also supplied with the software DecaRanging that allows to save the acquired ranging data, while 3D positioning needs to be performed with custom methods. The structure and functioning of DecaRanging is presented in Section 3.1.3.

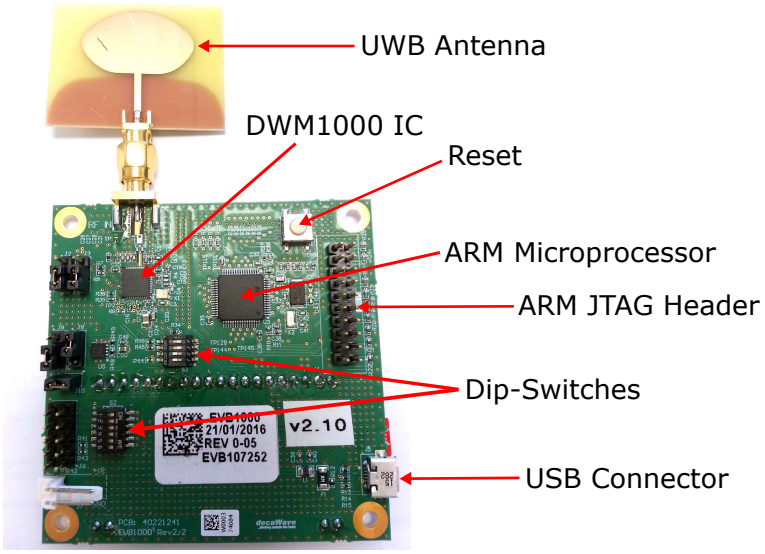


Figure 3.1. Decawave EVB1000 device.

3.1.2 Leica AT403

The absolute laser tracker AT403 is used to obtain a precise ground truth for both range and position measurements. The tracker is able to accurately follow the movement of a reflector and measure the absolute position at regular intervals in time or in space. This feature is particularly interesting for the scope of the thesis, as the required number of samples is high and taking range estimates singularly would require a huge amount of time. The list of measurements taken in each session is saved, thanks to the software Leica Tracker Pilot, into a log file reporting the 3D coordinates of the reflector with respect to the tracker reference frame and the timestamp of the result. In 3.1.2 the main features of the tracker are presented [42].

Board Dimensions	120x70 mm (including the antenna)
Weight	39 g
Operating Band	3.5-6.5 GHz 6 Channels (500 MHz width) U.S. FCC Compliant
Center Frequency	2 Channels: Ch.2: 3993.6 MHz Ch.5: 6489.6 MHz
Max Power Spectral Density	-41.3 dBm/MHz
Antenna	WB002 Omni-directional Planar
Ranging Techniques	Pulsed TWR (Two Way Ranging)
Max Range	290 m (LOS)
Ranging Precision	10 cm
Localization Technique	RTL software provided by Decawave
Network Protocols	TDMA
Max Positioning Update Rate (3 anchors, 1 tag)	3.57 Hz / 10 Hz (Depending on the operating mode)

Table 3.1. TREK1000 technical datasheet.

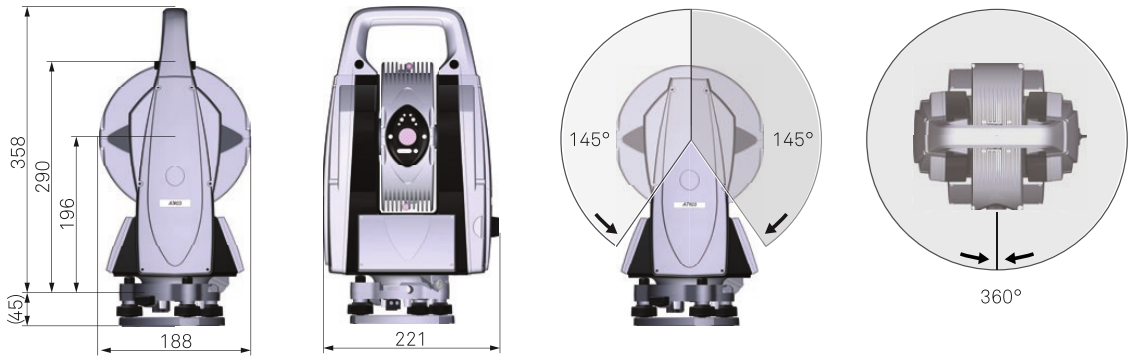


Figure 3.2. Leica AT403.

Figure 3.3. Leica AT403 on its tripod.

3.1.3 DecaRanging

Decawave's DecaRanging is a demonstration application that drives DW1000 integrated circuit, to prove the accurate measurements that can be made between a pair of units using two-way ranging. The DecaRanging PC application offers an alternative to the ARM embedded DecaRanging application allowing for additional configuration and diagnostic display possibilities. A DW1000 controlled by the DecaRanging PC application can perform two-way ranging to another DW1000 controlled by either the DecaRanging PC application or the DecaRanging ARM application.

Figure 3.4. Leica Red Ring Reflector 0.5”.

Sensor Unit Dimensions and Weight	290x221x188 mm / 7.3 kg
Controller Dimensions and Weight	250x112x63 mm / 0.8 kg
Measurement Angle	Horizontal ±360°, Vertical ±145°
Dynamic Measuring Speed	Maximum 10 Hz
Range	320 m
Accuracy	±15 µm + 6 µm/m
Laser Class	2
Laser Type	635 nm, < 1 mW
Operating Temperature	-15°C to +45°C
Relative Humidity	< 95%
Reflector diameter	0.5”

Table 3.2. AT403 technical datasheet [42].

There are two ways for the DecaRanging PC application to connect and control the DW1000 IC on the DW1000 evaluation boards. That is either via the USB interface or via the SPI interface header (and employing a Cheetah USB-to-SPI convertor). In either case the DecaRanging PC application essentially has control of the DW1000 which it drives to

exchange messages between a pair of devices, calculate the time-of-flight of those messages and display the resultant distance between the two units. Ranging operations can be performed in both LOS and NLOS conditions in all supported modes and configurations.

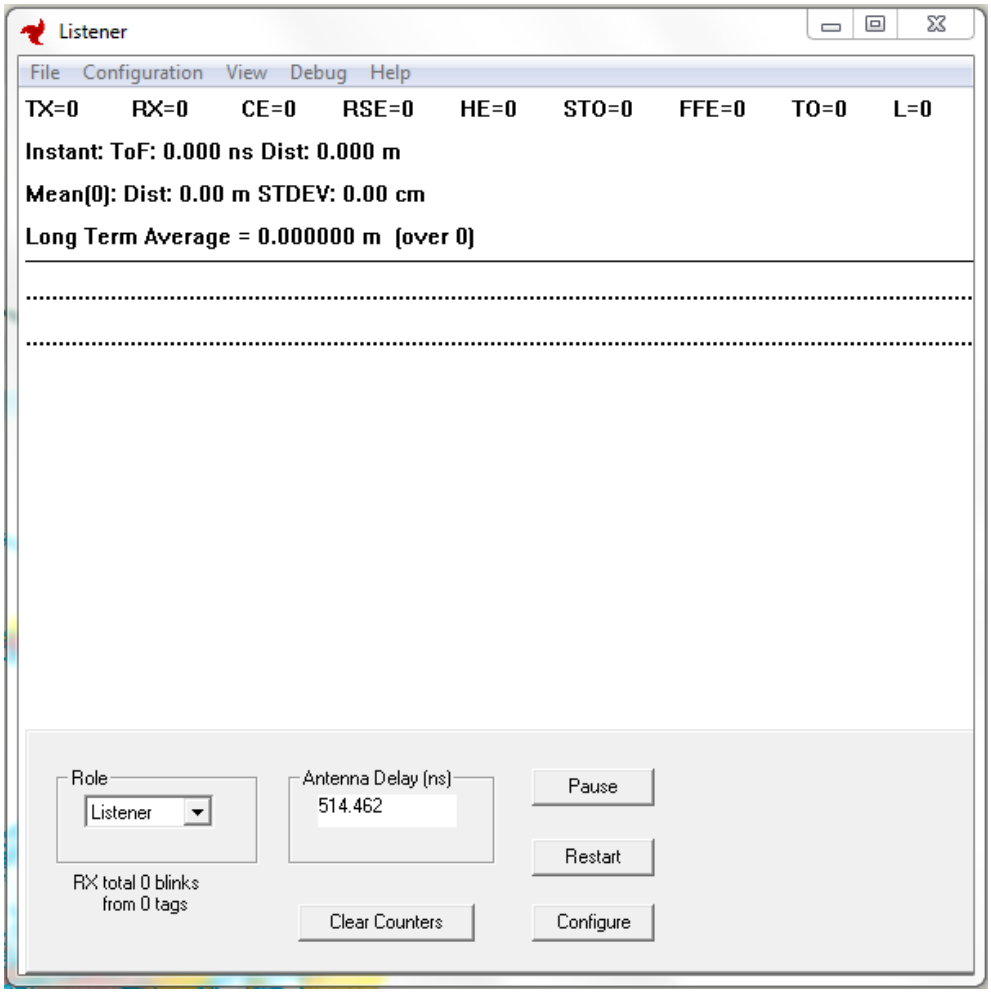


Figure 3.5. Decaranging window.

Looking at 3.5, the main elements of the DecaRanging window are visible. In the upper part, counters keep trace of the number of sent and received messages and of different kind of errors. A bit below, the data of the current measurement are present alongside statistic reports like mean and standard deviation. The commands for the configuration of the pair of boards is in the lower part of the screen, allowing to change the settings, modify antenna delay for calibration and assign roles to the sensors to pair them.

Through the View menu it is possible to view the CIR graph in real time, as shown in 3.6. As CIR is an array of complex values, the red line is the plot of the real values, the green line is the imaginary signal, and the blue line is the computed magnitude. The time

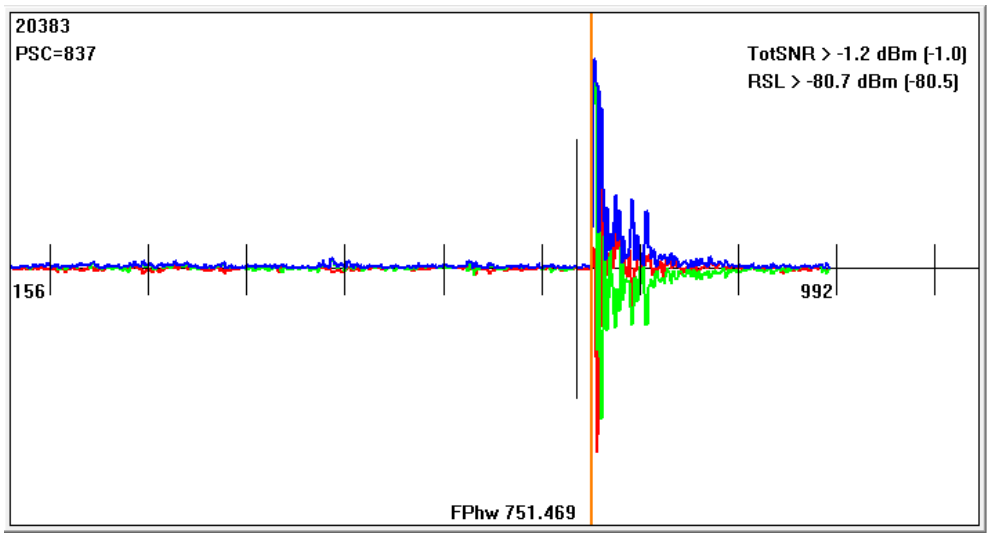


Figure 3.6. Decaranging CIR plot.

index at which the first path is detected is signed with an orange vertical line and the value in the lower part of the graph. In the upper-left part the maximum amplitude is reported, while information about noise and received power are in the upper-right one. From the Debug menu it is possible to toggle CIR logging, that saves all the information about the measures in a file. As the software is not customizable and only allows to manage one tag and one anchor, the Contiki-UWB system is preferred. DecaRanging is mainly used for the first tests and to precisely calibrate the sensors.

3.1.4 Contiki-UWB

Contiki is an open source operating system that runs on constrained embedded systems and provides standardized low-power wireless communication. This OS was ported in 2011 for the Decawave EVB1000 and DWM1001 platforms as described in [43], and includes the implementation of two communication protocols with UWB ranging primitives. To correctly test the performance of a mitigation algorithm on 3D positioning, at least four anchors and a tag are required, and DecaRanging only allows to manage two boards. For this reason, the Contiki-UWB source code (written in C and available at [43]) is modified to have full management of any number of anchors. In particular, the customized firmware allows to set the desired configuration for all the nodes, perform ranging between the tag and each anchor sequentially and output all the important information needed to run the algorithm: anchor label, CIR vector, range estimate and first path index. In the next sections it will be better discussed how the data are exported and processed to obtain the final position estimate.

3.2 Range Measurement Campaign

Now that the main hardware and software tools have been presented, the experimental procedure to collect the ranging estimates and build the dataset is presented. The relevant data are transmitted by the devices via serial port and saved in memory, while a simple Python script is used to extract the meaningful information from the log files, process it and prepare it for the deep learning algorithm.

3.2.1 Configuration and Calibration

First, the DWM1000 boards must be configured and prepared for the measurement session. The adopted configuration, described in 4.1, guarantees long distance ranging and an update frequency of at least 3.5 Hz. According with the user manuals [44] and [45], the calibration procedure is performed with a distance of 9 m between the nodes by tuning the antenna delay such that the measured range is correct. This parameter represents the propagation delay through the DW1000 devices from the points at which the transmitter timestamps are applied to the points at which the receiver timestamps are captured. The antenna delay must be the same for all the nodes, a value of 515.462 ns is assigned to both of them.

Channel	2 - Centre 4.0 GHz (500 MHz width)
Preamble Length	1024
SFD	Non Standard
Preamble Code	9
Pulse Repetition Frequency	64 MHz
Data Rate	110 kbit/s
Antenna Delay	515.462 ns
Tag Blink Period	1000 ms
Tag Poll Period	1000 ms
Anchor Response Delay	150 ms
Tag Response Delay	200 ms

Table 3.3. TREK1000 configuration for the measurement campaign.

3.2.2 Measurement Scenario

The measurement campaign is conducted in four different environments, trying to cover a wide variety of LOS and NLOS scenarios. In the outdoor environment the only source of error is the presence of obstacles and the consequent signal delay. In the three rooms used for indoor measurements, also the effect of multipath components strongly influences the CIR signal. Taking measurements in different conditions allows also to perform training, validation and testing on completely different datasets, and hence avoid overfitting and encouraging the deep learning model to learn domain-independent features.
As shown in 3.7, the measurements are taken by using the laser tracker as ground truth. First, the reflector is placed on the anchor to precisely measure its position with the laser tracker and have a landmark. Then, AT403 follows the reflector placed on the moving tag estimating its position ten times per second. Meanwhile, tag and anchor perform two-way ranging. The tag follows a path that spans around the selected environment, that is filled with obstacles to give rise to both LOS and NLOS measurements. After a satisfying amount of samples have been obtained from an environment, the configuration is changed by changing the position of the anchor or the type and position of the obstacles. When a sufficient number of alternatives has been tested, the environment is changed. Three rooms with different dimensions are chosen: one small, one medium and one big 3.4. The shape of the rooms, although, is similar in order to consider only the effect of dimension scaling. To consider also the effect of walls, some measurements are taken across two rooms (the small and the medium one). For each set, the room and the obstacles are noted down to be inserted in the dataset. The complete list of obstacles is reported in 3.5 and shown in 3.8.

Room	**Dimensions**
Large	10m x 5m
Medium	5m x 5m
Small	5m x 3.50m

Table 3.4. Dimensions of the rooms.

3.2.3 Data Extraction and Processing

The Contiki firmware saves its log files with a specific syntax, so to recover only the useful information from them in an automated way a parser is needed. So, a simple Python script has been written to read from the files, filter the information and perform all the needed operations. The data to be extracted for each range measurement are:

- Range estimate;

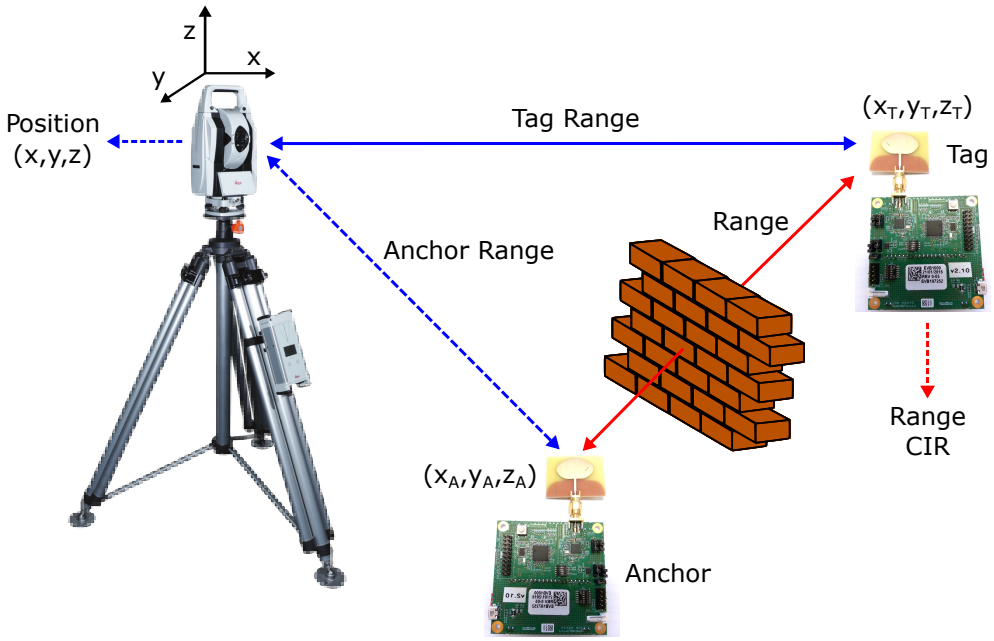


Figure 3.7. Range measurement setting.

Figure 3.8. Objects used as obstacles.

- CIR vector;
- First path index.

The range estimate only needs to be read, as it is already in the log file as a float number. The first path index is also a float in the file, so it must be converted to integer to be used: indeed, it is needed to cut the CIR signal and consider only the meaningful part, that is the

Obstacle	Dimensions
Polystyrene Plate	1m x 0,5m x 0,09m
Plastic Bins	0,6m x 0,3m x 0,3m
Wooden Plate	1m x 0,4m x 0,03m
Cardboard Box	0,6m x 0,4m x 0,2m
TV	0,5m x 0,3m x 0,04m
Glass Plate	1,50m x 0,5m x 0,01m
Aluminum Plate	1,50m x 0,4m x 0,001m
Wooden Door	2m x 0,5m x 0,05m
Metal Window	1m x 1m x 0,06m

Table 3.5. Dimensions of the obstacles.

portion between the first path arrival and the last path. According to [40], 152 samples should be sufficient to capture it all (3.9). To be sure to consider all the information carried by the signal, 5 additional samples are taken before the first path index: indeed, device's built-in first path detection algorithm could be less reliable in critical NLOS cases.

For what concerns CIR, the log file provides real and imaginary parts for each of the 1016 samples. As the proposed method uses the amplitude of the vector, the script computes it as

$$|\mathrm{CIR}| = \sqrt{\mathrm{CIR}_R^2 + \mathrm{CIR}_I^2} \tag{3.1}$$

Then, the UWB sample is matched with the ground-truth range measured by the laser tracker by comparing the time-stamps of the measures. The actual range is computed by difference knowing the fixed position of the anchor and the coordinates of the moving tag:

$$r_0 = \sqrt{(x_T - x_A)^2 + (y_T - y_A)^2 + (z_T - z_A)^2} \tag{3.2}$$

Finally, the range estimate is compared to the ground truth distance and the error is computed as

$$r_{\mathrm{err}} = r_m - r_0 \tag{3.3}$$

The value r_{err} is what the CNN will learn to predict being based only on the CNN samples. The room in which the measurement has taken place is included 3.6 as well as the

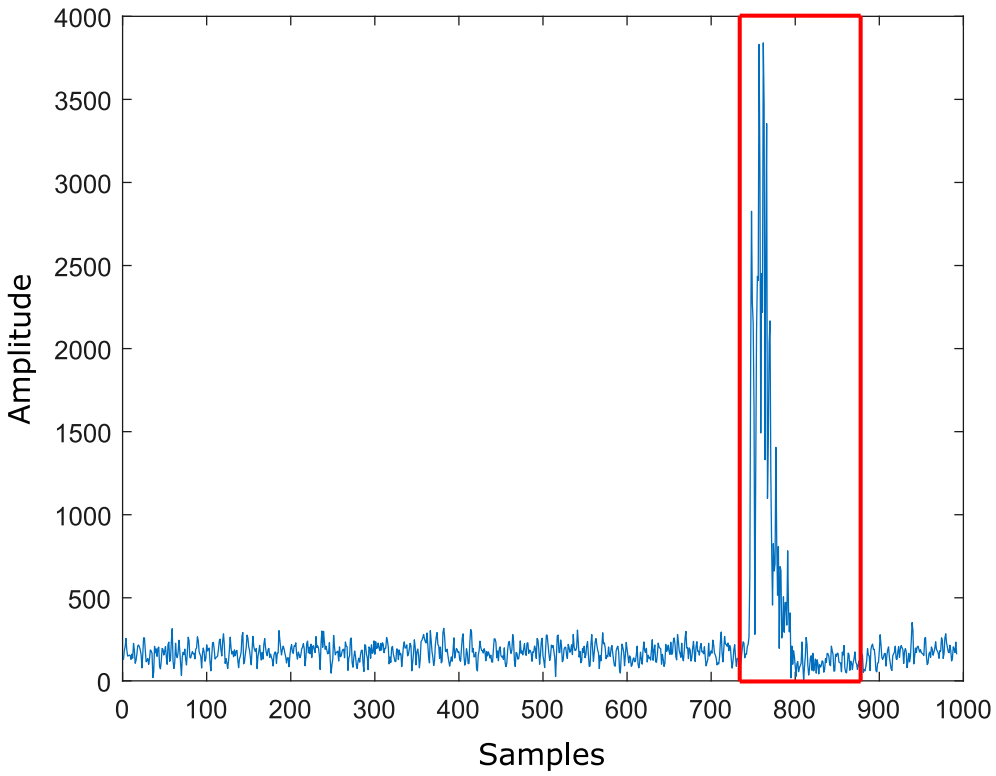


Figure 3.9. CIR signal with cropping window.

Environment	**Code**
Cross-Room	0
Large Room	1
Medium Room	2
Small Room	3
Outdoor	4

Table 3.6. Environment encoding.

vector of the adopted obstacles using one-hot coding 3.7. For the sake of completeness, also the UWB estimate of the range and the ground-truth distances are present.

So, in summary, each data-point in the dataset carries the information shown in 3.10. The whole dataset is packed in a single structure using the Pandas library and Python object serialization tool Pickle, and is publicly released to be useful for future works on this subject. The data file is completed with a file of instructions that explains how to

Obstacle	Code
Wall	1000000000
Polystyrene	0100000000
Plastic	0010000000
Wooden Plate	0001000000
Cardboard Box	0000100000
TV	0000010000
Glass Plate	0000001000
Aluminum Plate	0000000100
Wooden Door	0000000010
Metal Window	0000000001

Table 3.7. Obstacle 1-hot encoding.

properly read and interpret the dataset and the coding used for all the features [46]. The way the samples are used by the algorithm is described in the next chapter.

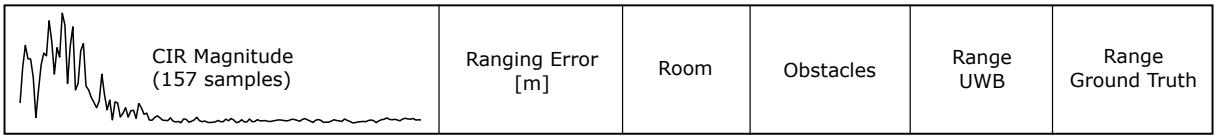


Figure 3.10. Structure of the samples.

3.3 Position Estimation Test

This section describes the experimental procedure to finally test the proposed method in a real scenario for a positioning task. The methodologies are very similar to those adopted for the ranging measurement campaign, with the difference that in this case five UWB sensors are used at the same time to obtain an estimate of the 3D position of the tag. Two different positioning algorithms are implemented to assess their precision and take the best. Finally, the position estimates are compared to the ones obtained without range mitigation to evaluate the effect of the method on the final accuracy.

3.3.1 Measurement Scenario

The experimental setting is similar to the one set-up for the creation of the dataset, but in this case the tag collects the information of each range estimate performed with all the anchors and sends it to a PC via a serial port. Differently from the ranging measurement scenario, here the tag is put in a fixed position, as the presence of a person moving it around the room would inevitably affect the position estimate. So, the measurements are all static and the AT403 is no more used to track the tag but only to obtain a precise ground-truth on the fixed positions of the sensors. Two rooms are chosen to conduct the test: one is the big room in which part of the dataset has been collected, while the other is a completely new room of smaller dimension. In both cases, the four anchors are placed near the corners of the room to surround the tag and to have the possibility of placing obstacles in between. For each room, a LOS and two NLOS scenarios are tested. For each configuration around a hundred ranges for each anchor are collected and written to a log file.

3.3.2 Data Extraction and Processing

While the log file is being written, a Python script reads it and extracts the necessary information exactly as done for the ranging measurement campaign. Then, the range samples are passed to the ML model for the mitigation after they are normalized with the same mean and standard deviation computed from the training set. For this test, the multi-layer perceptron is adopted as it is lighter and faster than the convolutional neural network. After the mitigation, both raw and corrected ranges feed the positioning algorithm to obtain two estimates of the final position. Both Gauss-Newton and the Extended Kalman Filter are used to discover which guarantees the highest accuracy for this application. Finally, the outputs are compared to prove which one is estimating the actual position at best.

3.4 Board Issues

Some problems have been reported during the measurement campaign regarding the DWM1000 boards. The scope of this section is to report the main issues, trying to identify a possible cause and describe, when possible, how these have been solved.

3.4.1 Bias Correction

Decawave's user manual [44] specifies the distance to execute the calibration of the boards to compensate bias errors in range measurements. As described in [47], there is a bias which varies with received signal level (RSL) that leads to a bias in the calculated time of flight. This is illustrated in 3.11, where the red line, labelled "Ideal", indicates the ideal result (constant) and the blue line, labelled "Actual", indicates the actual measured

result (which varies with RSL). For most applications this bias can be ignored, however higher precision ranging applications must correct for this effect. Knowing the calibration distance, the firmware can compensate this bias following a table of adjustment. As in some cases this method seems too coarse and imprecise, it is auspicable that the deep learning algorithm also tackles at correcting this bias. So, the proposed method will also compensate bias error alongside NLOS conditions.

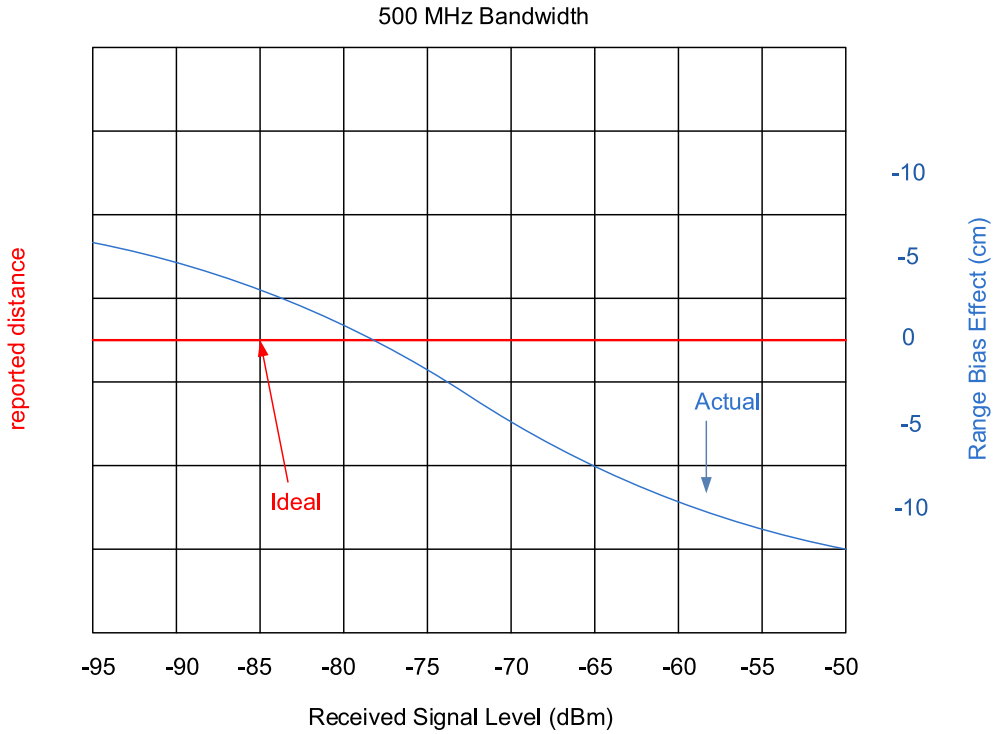


Figure 3.11. Bias effect caused by RSL.

3.4.2 Measurement Variance and Loss of Calibration

During long sessions of measurements, some strange behaviors are observed: series of range estimates show very high variance due to oscillations in the values, while the mean estimated distance deviates from the actual one with a sort of drift. This phenomenon cannot be caused by obstacles as it occurs even in LOS conditions, and it disappears after some minutes. According to [47] it is not a clock issue, as its drift in time is way more negligible. Another possible cause could be temperature: indeed, the technical notes report an antenna delay drift due to over-heating. Considering the long working time during the measurement campaign, this is the most plausible origin for the bad behavior of the boards. Moreover, the oscillation in range estimates could have the same cause as the increment of instrumentation noise generates wrong identifications of the first received

path and, hence, wrong time of arrival estimates. To overcome this problem, that also caused NLOS measurements to absurdly underestimate ranges, some precautions against over-heating are taken: the boards are used without their plastic case and as far as possible from any object to facilitate heat exchange, and prolonged use is avoided. With these tricks, a significant improvement in performance has been noticed and nearly no more strange behaviors are reported.

Chapter 4

Mitigation

In this chapter, the deep learning method adopted to enhance positioning is described. All the design phases are reported and commented, from the structure of the model to the optimization strategy.

4.1 Development Environment

The development environment in which the algorithm is built is Jupyter, a web-based interactive computational environment for creating notebook documents coded in many different languages [48]. It is an open-source service for cloud computing using one or more remote GPUs. It allows to create virtual environments for the developing of deep learning applications using popular libraries such as Keras, TensorFlow, PyTorch and Pandas. For this thesis, Jupyter is used to connect to the Synapses computational system at PIC4SeR. This system provides an NVIDIA RTX 2080 GPU with 8 GB of dedicated RAM and an NVIDIA RTX 2080 with 11 GB 3.1.1, that is suitable for medium-sized projects like this work of thesis, considerably speeding up the training process. The libraries Tensorflow and Keras are used to implement the deep learning model and all the tools used to train and evaluate it: together, they allow to realize complex network structures, to experiment a wide variety of activations and losses and to collect and visualize results in a simple way. For this reason, they constitute a complete and powerful tool for the purpose of this thesis.

4.2 Data Preprocessing

The first step to make a model learn is to feed it with an appropriate and well-built dataset, and to prepare the one created during the measurement campaign some simple preprocessing operations are needed:

GPU Model	RTX 2080	RTX 2080 Ti
Architecture Type	Turing	Turing
RTX-OPS	60T	78T
Boost Clock	1800 MHz (OC)	1635 MHz (OC)
Frame Buffer	8 GB GDDR6	11GB GDDR6
Memory Speed	14 Gbps	14 Gbps

Table 4.1. GPU Specifications.

4.2.1 Normalization

As pointed out by many papers and reported in [4], feature scaling is the most important transformations to apply to data, as machine learning models perform better if attributes have all the same scale. The most common way, that is applied here, is standardization. The mean value and standard deviation are computed among the whole dataset as

$$\overline{x} = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} \tag{4.1}$$

$$\sigma_x = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{m} (x_{ij} - \overline{x})^2}{n \cdot m}} \tag{4.2}$$

Where n is the number of data-points in the dataset and m is the number of samples in each CIR vector. In this case m is equal to 157. Then, each feature is normalized subtracting $\overline{x}$ and dividing by σ_x: the result is a distribution with zero mean and ranging between 0 and 1.

$$x_{ij}^{\text{norm}} = \frac{x_{ij} - \overline{x}}{\sigma_x} \tag{4.3}$$

This technique is minimally affected by outliers, differently from other scaling methods like min-max. Moreover, to be sure to properly normalize the data, a Batch Normalization layer is put on the input of the Deep Learning models. In this way, the network autonomously learn the optimal scaling to apply.

4.2.2 Shuffling

Since the data have been collected in blocks and need to be splitted in groups, the model should not be influenced by the order in which samples arrive. To avoid any bias or deviation in the learning phase, the whole dataset is shuffled before the use preserving the stochasticity of the process.

4.2.3 Splitting

Finally, the dataset is divided in three splits to optimize the model. As it is common practice to do in deep learning, nearly 64% of the samples are used to directly train the network, while 20% of them form the final test set. The remaining 16% is employed for validation: it is a way to control, during the training phase, how the model performs on data it has not been trained on. This concept is better explained in Section 4.4. To highlight the effect of obstacles and rooms on mitigation accuracy, several experiments are conducted choosing different splits for training and testing, as better explained in Chapter 5. This also alows to evaluate how general is the representation that the net builds of the physical phenomenon, and determines whether this method can be applied in scenarios that differ from the ones considered in the dataset.

4.3 Models

In this section the focus is on the designed Deep Neural Network models. First, the structure of the configurations is presented, commenting the main aspects and the crucial project choices. During the tuning of the net many different configurations have been experimented: the next chapter shows comparison between the results of alternative structures. Nevertheless, all the methods adopted and tested are listed and explained in the following sections, or alternatively in Section 1.3.

4.3.1 Structure

Two main structures are tested to evaluate the validity of the proposed approach and estimate the effect of different scenarios on the mitigation task. The first is a simple Multi Layer Perceptron (MLP), consisting of a series of five Fully Connected layers interleaved with batch normalization modules. At the end a dropout layer regularizes the model and the last one-neuron layer gives the output through a linear activation function (4.1). The second model is a CNN similar to common state-of-the-art examples for classification or regression tasks. Following an AlexNet-like structure, a series of convolutional layers is responsible of extracting the best features from the data. Obviously, as the CIR vector is one-dimensional, the adopted convolutional layers are 1D, too. To extract higher and higher-level features, the number of stacked filters increases while their dimension remains constant at 3x1. The activation function for the layers is ReLU, while a Batch Normalization layer is interposed between convolutionals.
Then, the series is interrupted by a Global Average Pooling layer and a Dropout layer, the functioning of which is explained in the next section. These blocks introduce to the last part of the network, where the features are used to predict the value of the ranging error using a fully connected layer. The last layer collapses in a single neuron that estimates the ranging error and hence needs a linear activation function (4.2). In general, relatively simple models are preferred to guarantee the possibility of running mitigation in real time

and, eventually, the deployment on a Tensor Processing Unit (TPU).

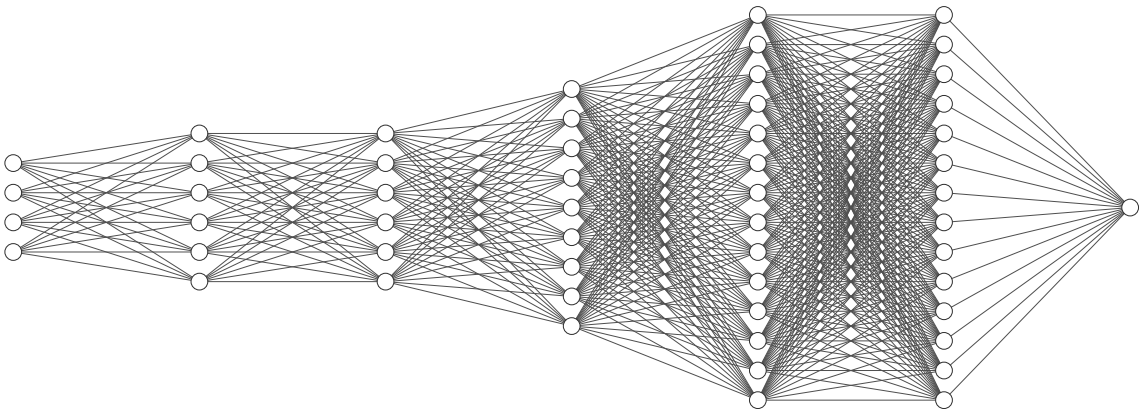


Figure 4.1. Structure of the proposed MLP: the circles represent the neurons of each layer.

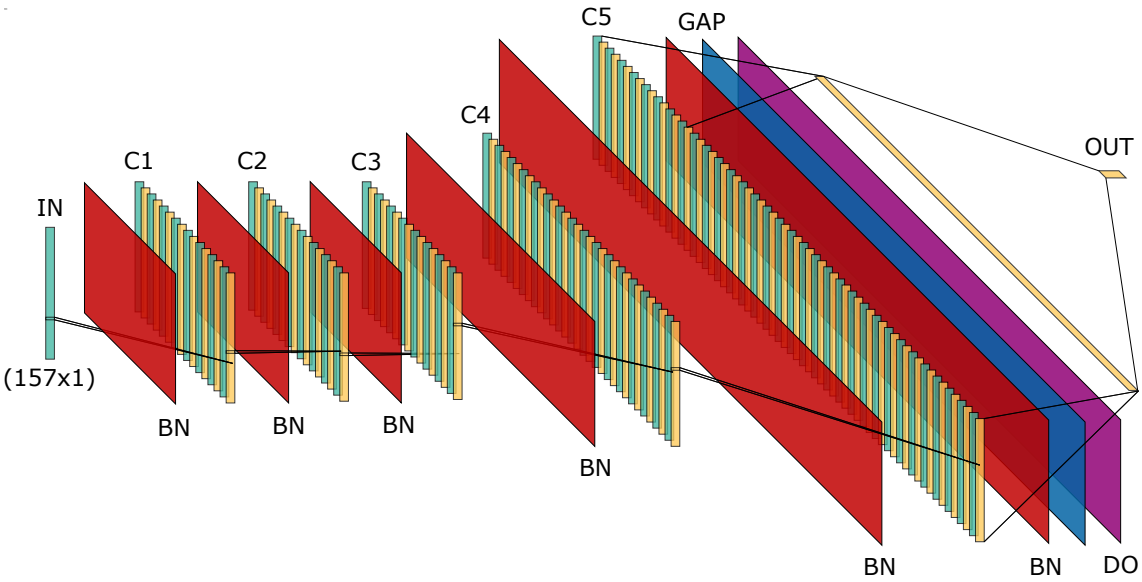


Figure 4.2. Structure of the proposed CNN: Convolutional layers (C), Batch Normalization (BN), Global Average Pooling (GAP), Dropout (D) and Fully Connected layers (FC).

4.3.2 Layers

Now that the main structure has been presented, a focus on the adopted techniques is addressed to explain all the design choices done in the experimentation of different solutions. Some basic concepts are already introduced in 1.3, so they are not repeated: in this context more recently developed techniques are described.

Global Average Pooling

As explained in the previous chapters, pooling consists in a down-sampling of the input: it is a very effective way of reducing the size of the features extracted from the convolutional layers, and hence decrease the complexity of the model. Moreover, it provides invariance to shifts in position and other minor transformations of an object, thus improving generalization performance. However, recent studies questioned the benefits of pooling, and specifically max pooling, as it does not seem to provide invariance in many modern scenarios where very deep networks are used. What happens to be a better choice is to apply a single Global Average Pooling (GAP) layer between the convolutional series and fully connected layers. GAP takes the average value for each of the feature maps and group all in a one-dimensional vector (4.3).

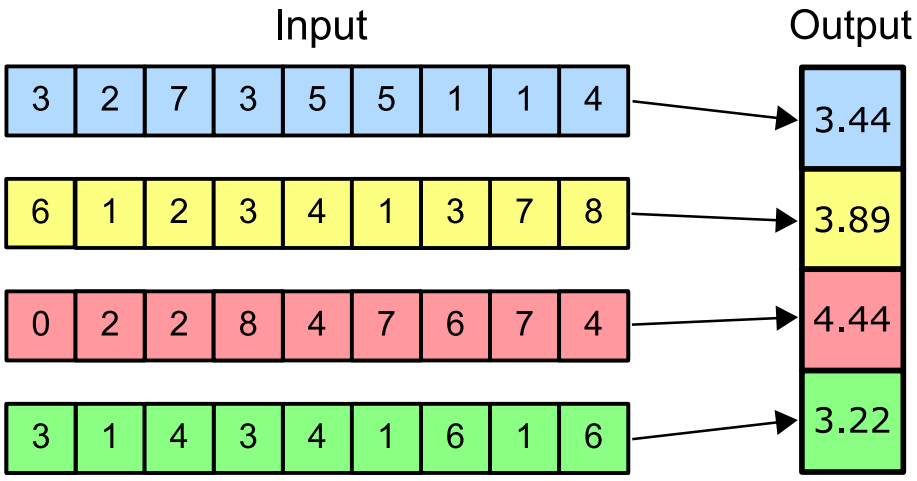


Figure 4.3. Example of the functioning of Global Average pooling.

This computation is way more efficient than flattening, that was usually done to obtain a 1D vector to feed FC layers: flattening just forms a long vector concatenating all the feature maps, causing an enormous quantity of parameters. In synthesis, GAP can replace a fully connected layer and a flattening operator, recognizing in a more native way the meaningful features as it does not need to train its parameter. Moreover, it reduces the complexity of the GAP part of the network and, hence, the risk of overfitting the training data.

Monte Carlo Dropout

Speaking of overfitting, Dropout is an efficient regularization strategy to encourage the net to generalize its representation without sticking too much on training data. It consists in casually turning off part of the neurons of a fully connected layer during the training phase, with a probability that can be tuned properly (4.4). Dropout generates a non-deterministic training, compensating fitting biases. With normal dropout, dropout is not applied during test time; instead, all connections are present, but the weights are adjusted accordingly multiplying the dropout ratio. Such a model during test time can be understood as an average of an ensemble of different neural networks. This means determinism: without other source of randomness, given one test data point, the model will always predict the same label or value. In Monte Carlo dropout, the dropout is applied at both training and test time. At test time, the prediction is no longer deterministic, but depending on which links you randomly choose to keep. Therefore, given the same datapoint, the model could predict different values each time. So, the primary goal of Monte Carlo dropout is to generate random predictions and interpret them as samples from a probabilistic distribution. In the authors' words, they call it Bayesian interpretation [49].

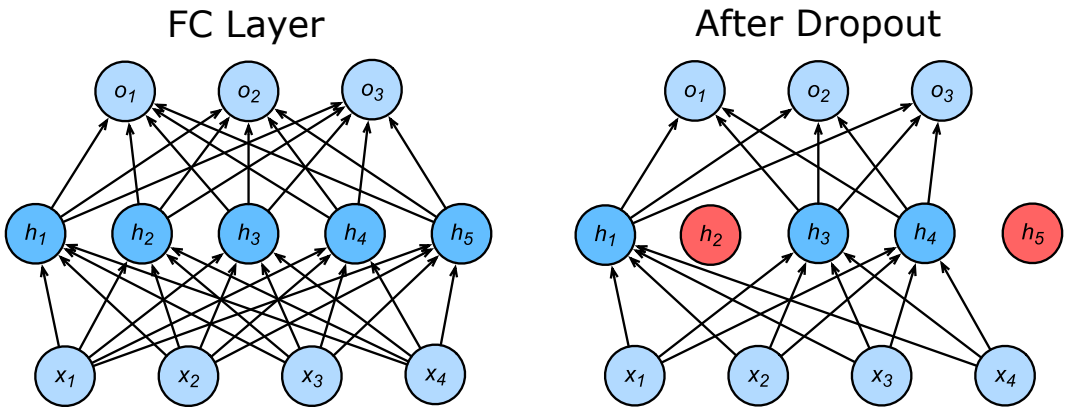


Figure 4.4. Example of the functioning of Dropout.

Activation Functions

As said in previous chapters, the role of activation functions is to determine whether a neuron should be activated or not, based on whether its input is relevant for the model's prediction. They also help normalizing the output of each neuron to a range between 1 and 0 or between -1 and 1. Four different activation functions are tested in this work (4.5):

- ReLU: it is the most used function in deep learning, for its optimal performance and easy computation.

$$f(x) = \max(0, x) \tag{4.4}$$

- Mish: as reported in [50], Mish is a very recent nonlinear activation function that showed better results than other common alternatives. In this work, it is compared to ReLU to find an optimal configuration for the net.

$$f(x) = x \tanh(\log(1 + e^x)) \tag{4.5}$$

- Sigmoid: this function is used exclusively on top of the binary classifier, as it must output values close to +1 or -1 depending on the most likely class.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.6}$$

- Linear: this function is used exclusively for the regressor's output, as the prediction must be a measure of the error.

$$f(x) = cx \tag{4.7}$$

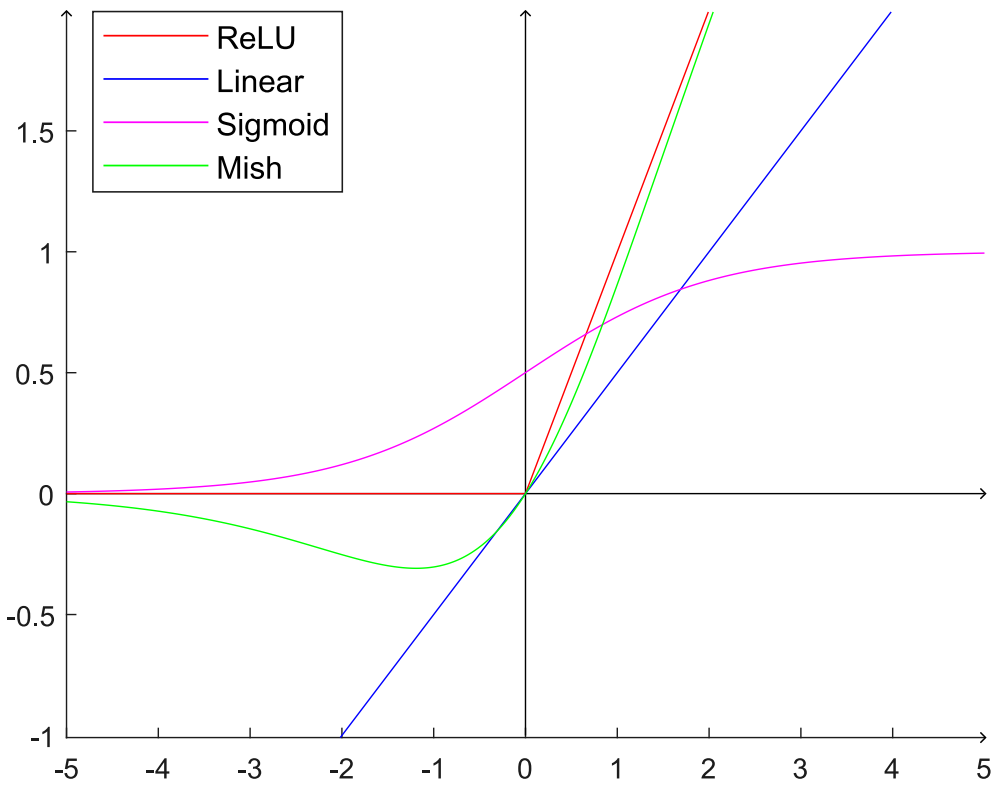


Figure 4.5. Comparison between the activation functions used for the CNN.

4.4 Learning Strategy

In this section, the procedure followed to properly train the CNN is described, commenting the adopted tools and the values chosen for the main hyper-parameters.

4.4.1 Loss Functions and Metrics

The loss function is responsible to quantify the error of the predictions, allowing the model to improve itself. The functions adopted for the regressor are all described in detail in Section 1.3 (MSE, MAE, Huber). Nevertheless, the case under study needs to properly choose some metrics that quantify the effectiveness of the mitigation. The Mean Absolute Error is selected to evaluate how close the predicted errors are to the real ones, while two metrics are adopted to describe the ranging errors after the mitigation. The two statistical distributions of the error are compared computing the ratio between the means and the standard deviations, obtaining the percentage of improvement given by the method.

4.4.2 Optimizer

The optimizer is the agent that allows the network to learn. It computes the gradients with respect to each parameter through backpropagation and then updates all the values. The procedure is repeated for each sample (or group of samples) that enters the CNN, aiming at the set of parameters that minimizes the loss function. As in this case two tasks are executed at once, the optimizer minimizes the sum of the loss functions. Three different alternatives are tested:

- SGD: Stochastic Gradient Descent, as explained in Section 1.3
- RMSprop: The central idea of RMSprop is keep the moving average of the squared gradients for each weight. Then, the gradient is divided by the square root of the mean square. Therefore, it is called RMS (Root Mean Square) propagation. The update rule is

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta)\left(\frac{\delta C}{\delta w}\right)^2 \tag{4.8}$$

$$w_t = w_{t-1} - \frac{\eta}{\sqrt{E[g^2]_t}}\frac{\delta C}{\delta w} \tag{4.9}$$

- Adam: a stochastic gradient descent algorithm including momentum and specific learning rate for each parameter, as explained in Section 1.3. This is the most recent and most used of the considered optimizers.

The effectiveness of an optimizer also strongly depends on the choice of its parameters, so they must be tuned properly in the model evaluation phase. For example, Adam has four main parameters:

- Learning Rate: the step size in weight updates. Larger values result in faster learning, while smaller ones mean small initial learning steps. Adam automatically changes the learning rate for each parameter, but always uses the initial learning rate as an upper limit.
- β_1: the exponential decay rate for the first moment estimates.
- β_2: the exponential decay rate for the second-moment estimates.
- ε: a very small number that preserves stability preventing any division by zero in the implementation.

In addition, a Learning Rate scheduler is used to enhance the work of the optimizer. Its scope is to reduce the learning rate with the passing of epochs, and consequently force the net to take smaller and smaller steps. This makes the optimization more stable and the convergence smoother. The used scheduler updates the learning rate every twenty epochs following the formula:

$$\mathrm{LR}_i = \mathrm{LR}_0 \cdot s^{\left\lfloor \frac{1+i}{e} \right\rfloor} \tag{4.10}$$

where LR_0 is the initial learning rate, LR_i is the value at epoch i. s is the step and e is the number of epochs between two updates.

4.4.3 Validation

As already mentioned in the preprocessing section, the dataset is split in three parts, each with a specific task. The training set is what the network learns from and the test set is the final benchmark assessing how the model performs on instances it has never seen before. The advantage of using an additional portion, called validation split, is that it gives a feedback of the model accuracy during the training phase. This allows to see in real time and quantify the generalization error, that is the error rate on new cases. In this way the net can be modified and tuned to reduce generalization error by avoiding overfitting. For the case of this project, validation is run at the end of each epoch, and the results are saved and plotted at the end of training phase to be compared with the accuracy reported on the training dataset.

4.4.4 Early Stopping

Early stopping is a form of regularization used to avoid overfitting in long training iterations. Indeed, when the number of epochs is high, the learner continues learning at the expense of increasing generalization error. Early stopping aims at detecting the moment

when this deviation begins and stop the training before the net is compromised. The implementation is simple, as looking at validation loss it easy to spot overfitting: when it starts to increase contrary to training loss, the net has stopped learning.

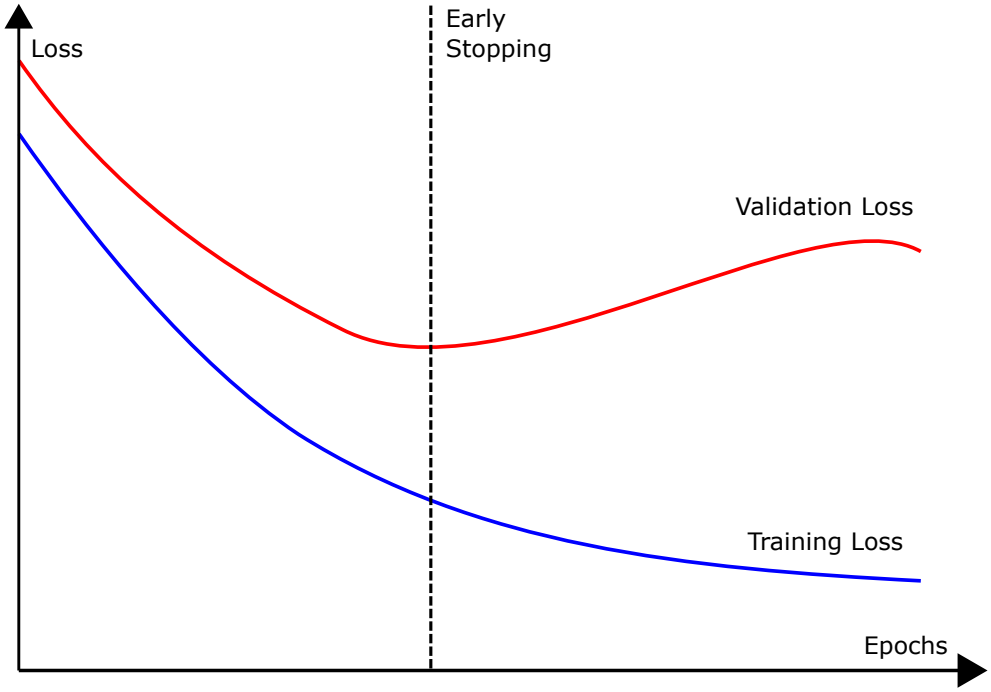


Figure 4.6. Importance of Early Stopping in training phase

Chapter 5

Experimental Tests and Results

In this chapter, the main experimental results are resumed, analyzed and commented. First, a look is given to the ranging measurements dataset, reporting the relevant features and statistics of the different sets of samples. Then, the results of the network training are presented, comparing the different tested combinations and highlighting some important insights. Finally, the outcomes of the positioning test conducted in a real scenario are shown and commented, to determine whether a significant improvement has been reached or not.

5.1 Dataset Analysis

The measurement campaign has led to the creation of a dataset with over 55.000 samples containing all the relevant information for the application of a mitigation task: CIR, ranging error, room label, obstacle labels, estimated and ground-truth range. In this section, a brief analysis of the collected data is conducted, trying to visualize and recognize patterns that can be seen to the naked eye. This preliminary step is of fundamental importance when dealing with deep algorithms, as it allows to make some rough hypotheses on the strong correlations between the data and to know what to expect from the model to learn. For this reason, 5.1 and 5.1 show some important characteristics for each subset, separated by room and by obstacle. The rooms are simply split considering the three rooms singularly, the measurements taken between adjacent rooms and finally outdoor samples. As the number of different obstacles is much higher, the objects are clustered based on the type of material: plastic, cardboard and polystyrene are grouped as light materials, metal objects like the aluminum plate, the TV, the glass plate and the window are classified as heavy, wooden obstacles are considered together. The last two groups are represented respectively by walls and measurements without any obstacle, to have an idea of the difference between LOS and NLOS samples. The MAE, the mean CIR signal and statistical distribution of the error are reported, in order to know what the starting point of the mitigation process is. For the sake of completeness, in the Appendix (6) the

complete tables including also other interesting metrics are present.

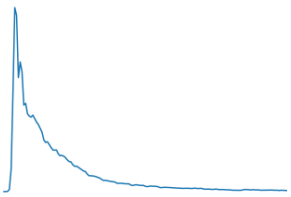
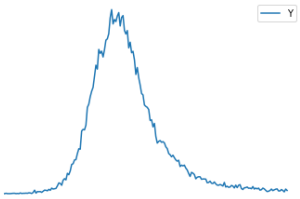
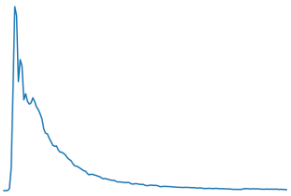
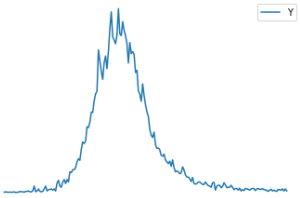
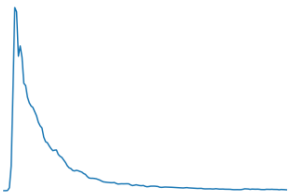
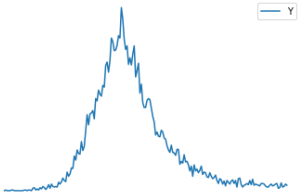
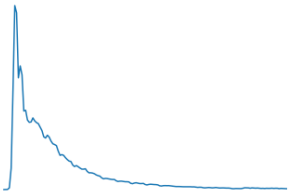
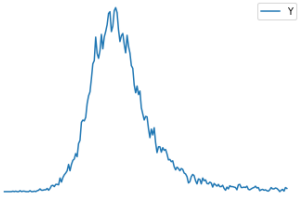
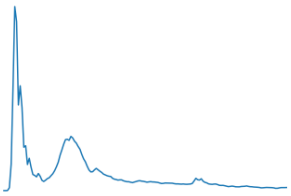
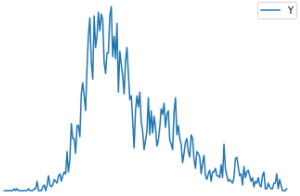
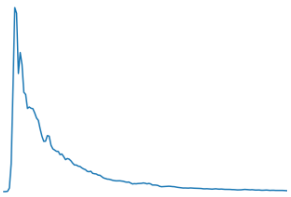
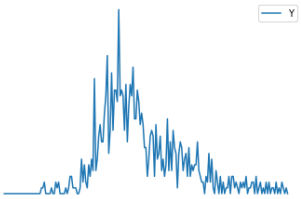
Subset	**MAE [m]**	**Mean CIR**	**Error Distribution**
All	0,12423939		
Big room	0,12527636		
Medium room	0,13911064		
Small room	0,10685123		
Outdoor	0,13446487		
Cross-room	0,1658175		

Table 5.1. Dataset metrics and graphs for different environments.

A quick look is sufficient to notice important differences between the splits: the overall

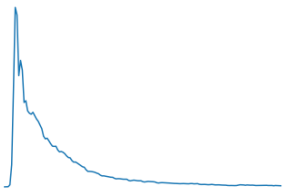
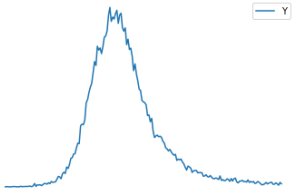
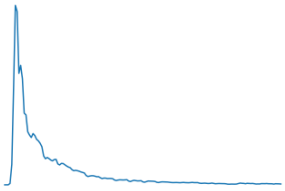
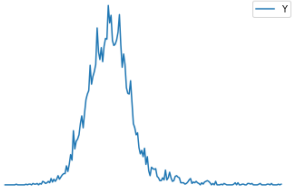
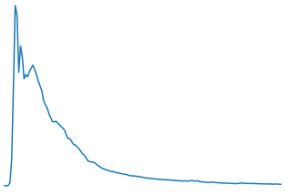
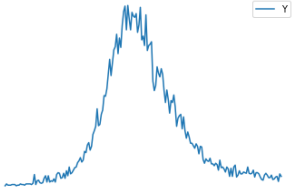
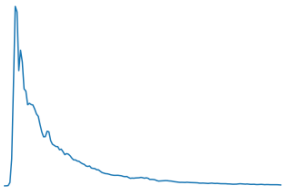
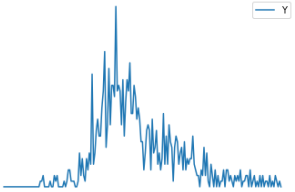
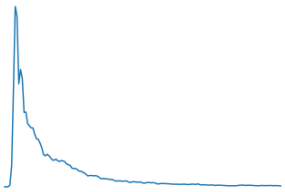
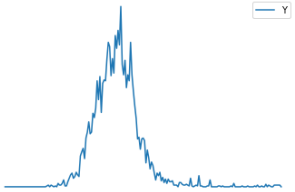
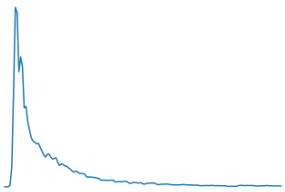
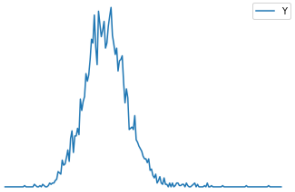
Subset	MAE [m]	Mean CIR	Error Distribution
All	0,12423939		
Light materials	0,0733789		
Heavy materials	0,19614786		
Wall	0,1658175		
Wood	0,07835419		
LOS	0,05942106		

Table 5.2. Dataset metrics and graphs for different materials.

error distribution is approximating a Gaussian, as it is for all the three rooms. Outdoor and cross-room measurements, instead, present a more irregular and unbalanced histogram, while the CIR signal shows a different tail effect. This is probably caused by the absence

of multipath components and reflections. For what concerns materials, there is a deeper difference in terms of MAE as values go from 6 cm to 20 cm. Excluding the wall subset, the shapes of the histograms are more similar, but there is a big difference in terms of mean and standard deviation: for example, heavy materials present a larger distribution than light materials and wood. Also CIR signals reflect this phenomenon, as very diverse shapes can be observed.

To visualize the distribution of the different clusters in the dataspace, Principal Component Analysis (PCA) is used to project the 157 dimensions of each sample into a 3D space conserving most of the information. For this scope, the Tensorboard Projector provided by Tensorflow is used to plot the samples labelled by room and by material. In the first case, the three rooms are included as well as the outdoor set, as shown in 5.1: an evident prevalence of samples from the big room can be found in the lower central part of the plot, while the medium room samples are more present in in the left side of the distribution and the CIR signals measured in the small room are on the right. Finally, the outdoor set is the more recognizable in the upper part of the plot, highly concentrated.

For what concerns materials, an analogue procedure is carried out considering considering three objects for clearness: the aluminum plate, the plastic bins and the wooden door (5.2). Again, a remarkable separation can be seen by eye, as the metal samples occupy all the upper part of the graph and light objects like plastic and wood take the lower zone. Moreover, also the spatial distribution of wood occupies specific zones showing different features from plastic. In conclusion, this qualitative analysis allows to have a first prove of the representativity of data and draw some conclusion on how the model will perform: for example, from the second group of images it is evident that a model trained on measures taken in presence of plastic will more easily mitigate the error caused by wood and will predict less accurate corrections for metal samples.

5.2 Range Correction

In this section the results of the proposed deep learning algorithm are presented. The first test is conducted by taking the whole dataset and splitting it in train, validation and test set. The aim of this first benchmark is to evaluate the overall effectiveness of the method, without focusing too much on reaching the best mitigation performances. More performant models can surely be designed and optimized, but as simple algorithms are being used, it is sufficient to see a significant improvement to confirm the validity of the approach. In 5.3, the results of the test are summaried: MSE and MAE are reported, as well as the ratio between them and the raw benchmarks, expressed in percentage. The metrics show a considerable mitigation effect: the resulting MAE is about 6.75 cm, which means a reduction of 46%. This improvement is reflected also by the other metrics, and in particular by the error distribution parameters, which are fully reported in the Appendix (6). It is evident that the model shifts the distribution to zero with great precision, and simultaneously reduces the standard deviation tapering the error distribution, as shown

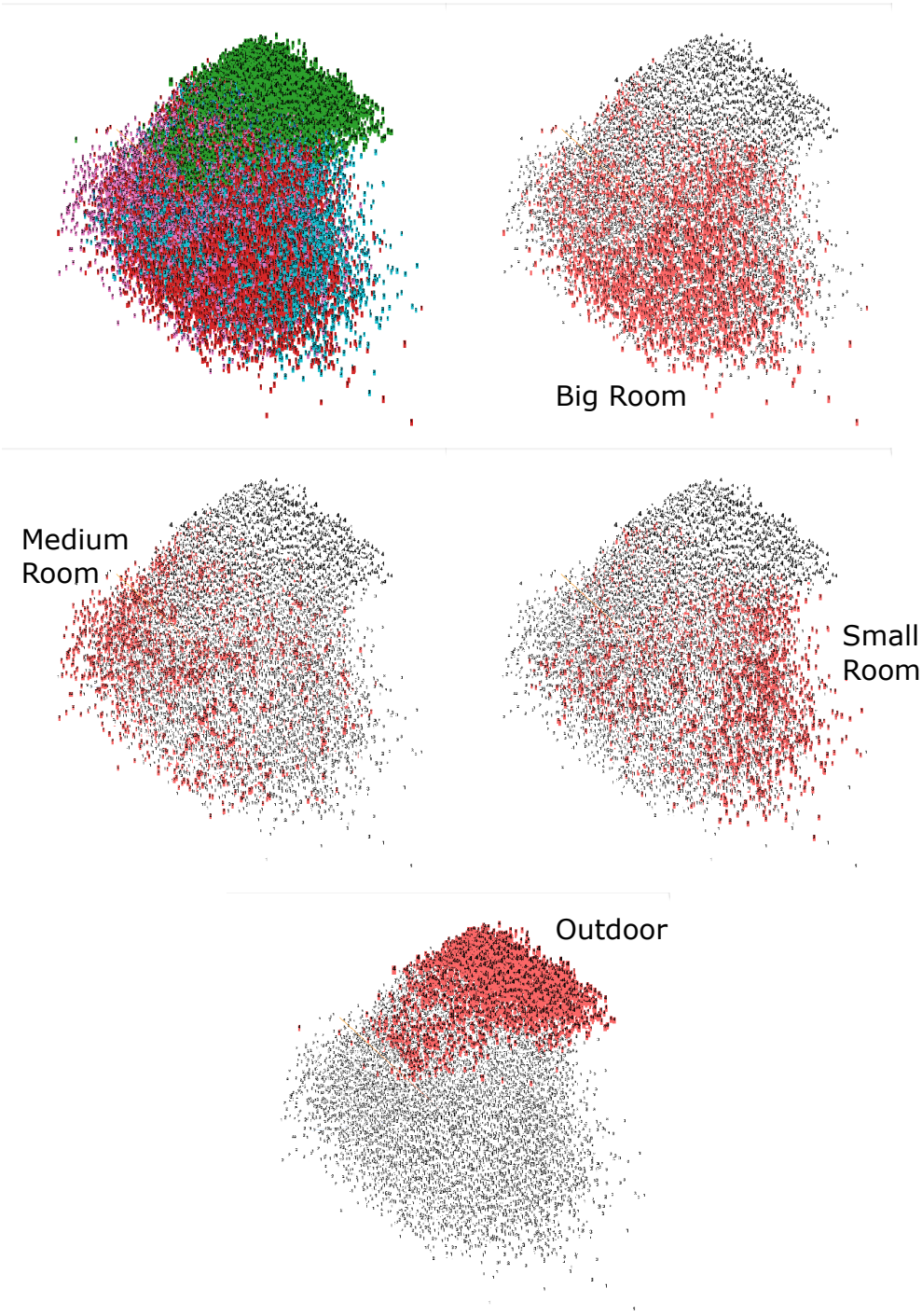


Figure 5.1. Principal component analysis on the dataset separed by room.

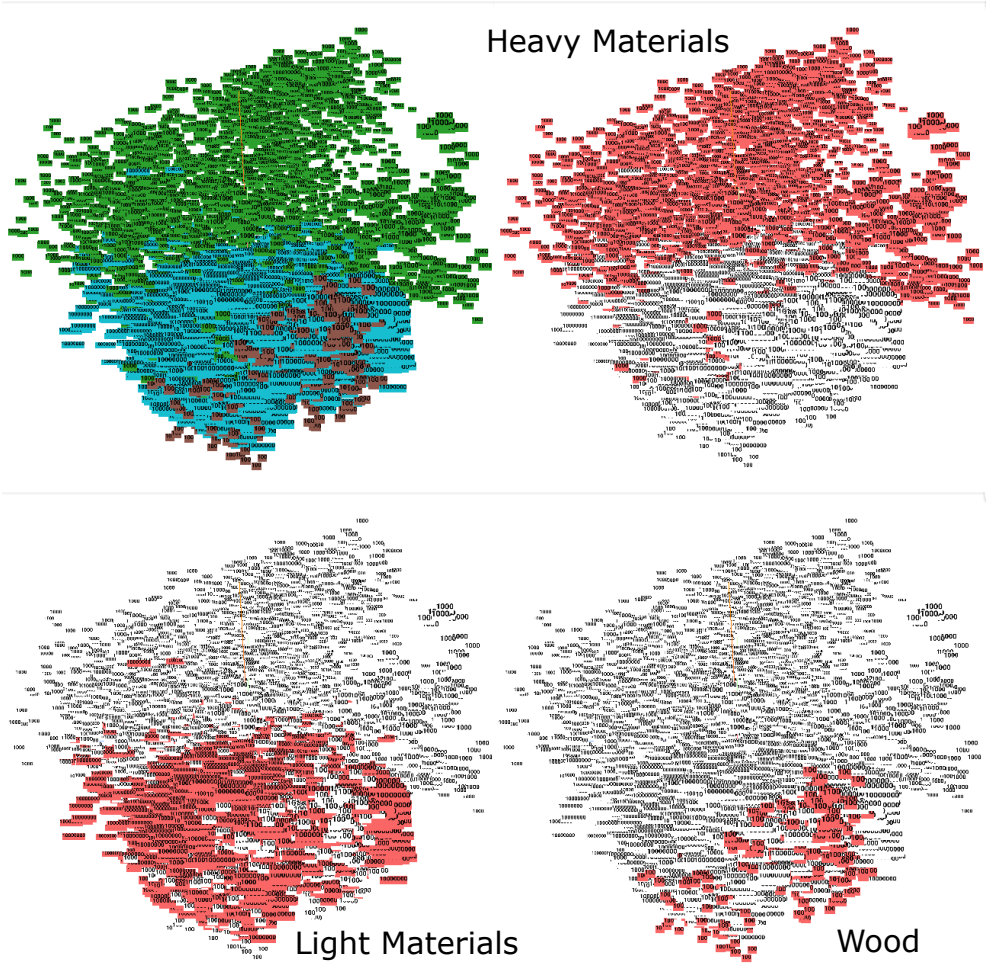


Figure 5.2. Principal component analysis on the dataset separed by material.

in 5.3. Only the results of the MLP are reported in the following tables and graphs, as the accuracies achieved with the CNN are very similar. Again, all the results are consultable inthe Appendix (6).

Train Set	**Test Set**	**MSE**	**MAE [m]**	**MSE %**	**MAE %**
All	All	0,01411715	0,06753542	36,237585	54,3591065

Table 5.3. Results obtained from the initial tests on the whole dataset.

To have an additional and more restrictive benchmark of the accuracy of the model, a test is conducted training the models on all the dataset excluding a single environment

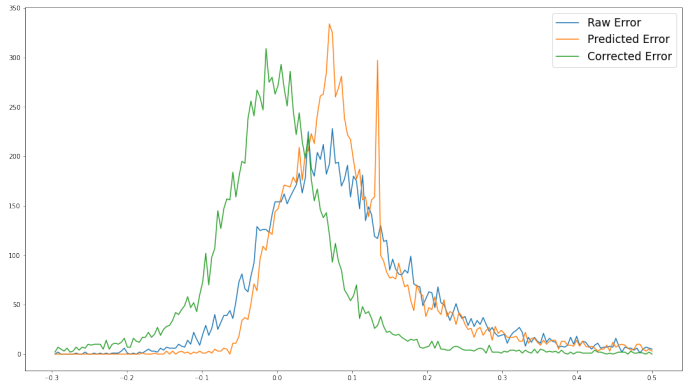


Figure 5.3. Statistical distribution of the error for the first test: raw values, model predictions and corrected output.

and then evaluating it on that environment. In this way it is impossible for the net to take advantage of overfitting as the scenario is completely unseen. The results, summed up in 5.4, show a good result for what concerns the rooms: an MAE of about 7-8 cm is achieved, proving that the net learns general features with respect to the room shape and dimension. Outdoor samples obtain an MAE of 10 cm, still meaning a 23% improvement on the raw, which is good if we consider the radical difference between indoor and outdoor scenarios. 5.4 shows how the model prediction changes the statistical distribution of the error, both shifting the mean close to zero and reducing the standard deviation.

Train Set	**Test Set**	**MSE**	**MAE [m]**	**MSE %**	**MAE %**
All - Big room	Big room	0,02666005	0,08040226	58,6798374	64,179912
All - Medium room	Medium room	0,01539912	0,0797291	37,4236211	57,3134432

Table 5.4. Results obtained from the restrictive benchmark computed excluding a single room from the training dataset and testing on that room.

After demonstrating the effectiveness of the proposed method, numerous tests are conducted to highlight and evaluate the effect of changing the room on the mitigation task.

5.2.1 Environmental Influence

Changing room modifies the geometry of the walls and hence affects the multipath components arriving to the UWB receiver, that outdoor measurements totally lack. To do this test, the net is trained on the data coming from a single room subset and tested on the same room with different data. The only exception is the subset containing the cross-room measurements, that is used only for testing as it contains less than a thousand samples and

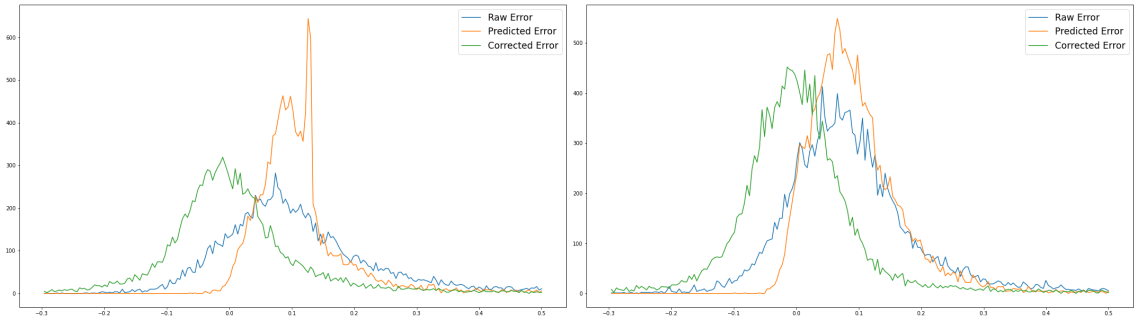


Figure 5.4. Statistical distribution of the error for the restrictive tests on big room (on the left) and medium room (on the right).

it is not enough to properly train a model. Then, the same model is evaluated on the data coming from all the other rooms: the result shows how much the performance degrades in a different environment. The complete table with the results for all the metrics can be found in the Appendix (6), while here some observations are presented alongside MSE and its percentage reduction as a summary score (5.5).

As expected, the best correction is found testing the model in the same environment it has been trained for, with a little difference between the two models. However, the metrics show that transfer learning between the three rooms leads to very small losses (less than 2 cm) compared to those caused by cross-room and outdoor measurements. The samples taken in open space show the worst results, because they are taken in a completely different scenario and the models struggle to adapt to that situation. The cross room also lead to a considerable loss as the walls are very different obstacles from other objects used inside rooms. In conclusion, both the nets seem to learn features that allow generalization with respect to the environment. In fact, the effect of the geometry of the room is strongly reduced and a sensible improvement is shown also for very different scenarios like outdoor and cross-room. The only pejorative results are found when training on outdoor samples and trying to model multipath components derived from reflections, as in this case the model finds itself in a completely unseen situation. This explanation is confirmed by the fact that the loss is more pronounced in the small room, in which reflections are certainly more dominant. Finally, the only significant difference between the two models is that the CNN shows less variance between the results. This means that the net eventually learns more general features from the data.

5.2.2 Obstacle Influence

The second most important factor determining the ranging error is the way obstacles attenuate and delay the received signal, so a series of tests is conducted to measure the effect of different materials on the proposed method. Similarly to what done with rooms, the models are trained on a certain category of obstacles and tested on all the other materials.

	Big Room	Medium Room	Small Room	Outdoor	Cross-room
Big Room			MAE [m]		
	0,07787607	0,08550771	0,07873359	0,10559664	0,09618016
			MAE / RAW %		
	62,1634183	61,46741	73,68524538	78,5310221	58,00362399
Medium Room			MAE [m]		
	0,08834743	0,07738681	0,08228663	0,10208596	0,09822902
			MAE / RAW %		
	70,5220261	55,6296827	77,0104632	75,92016925	59,23923542
Small Room			MAE [m]		
	0,08452915	0,08284867	0,06609059	0,12467928	0,09493756
			MAE / RAW %		
	67,4741407	59,5559531	61,852906	92,72256478	57,25424591
Outdoor			MAE [m]		
	0,12747252	0,14105202	0,11191515	0,05970919	0,16872023
			MAE / RAW %		
	101,753049	101,395562	104,7392252	44,4050466	101,7505562

Table 5.5. Summary of the results obtained for the test on environmental influence.

The obstacles are arranged in the following custers: light materials (plastic, polystyrene and cardboard), heavy materials (TV, aluminum and glass), wall and wood. An additional split is created with all the LOS samples, to evaluate whether a net trained on NLOS data can improve the precision of such measurements. The table with the results for all the metrics is reported in the Appendix (6), while here a summary of the most important insights is presented 5.6.

As previously thought, there is a sensible decrease in model absolute accuracy transferring the knowledge of the net to different obstacles. However, in nearly all the tests there still is an improvement with respect to the raw estimates, meaning that part of what the algorithms learn does not relate to the kind of obstacle. Only in two cases the situation is worsened by mitigation: the first is the one obtained when the models are tested on the LOS split, while the second is the specific case in which the models are trained on heavy materials and tested on light ones. In both cases, the cause is a marked difference between training and testing samples, highlighting the fact that obstacles have a higher

	Light Materials	**Heavy Materials**	**Wood**	**LOS**	**Wall**
Light Materials			MAE [m]		
	0,06050171	0,14163822	0,06247131	0,07362957	0,1165821
			MAE / RAW %		
	82,451099	72,2099226	79,72938314	123,911563	70,30747601
Heavy Materials			MAE [m]		
	0,08750228	0,0955946	0,07483216	0,07795361	0,09275154
			MAE / RAW %		
	119,247194	48,7359885	95,50499191	131,188511	55,93591704
Wood			MAE [m]		
	0,06828006	0,13794212	0,04708451	0,06202191	0,1058047
			MAE / RAW %		
	93,0513533	70,3255788	60,09188759	104,3769753	63,80792083
LOS			MAE [m]		
	0,06650509	0,160725	0,05933153	0,04647238	0,13225417
			MAE / RAW %		
	90,6324427	81,9407347	75,72222013	78,20859529	79,7588728

Table 5.6. Summary of the results obtained for the test on obstacle influence.

impact on the effectiveness of the method than rooms. Speaking in absolute terms, heavy and wall subsets still keep the highest values, but they both improve strongly compared to the raw values, especially the wall split that obtains an MAE of about 10 cm. The best result reached on the wall subset (9 cm MAE) is the one trained on heavy materials, confirming the similarity between the two in screening the UWB signal. For what concerns the other groups, a strong correspondence is found between light materials and wood measurements, deriving in a small loss of accuracy when switching from a subset to the other. In conclusion, the tests conducted on the effect of materials show that a complete and effective dataset needs the presence of a sufficient number of different obstacles more than it needs different environments. In particular, metallic objects and walls cause the strongest effect on ranging estimates for their ability to screen radio signals. On the contrary, also a considerable number of LOS samples are needed to keep the model from

worsening raw measurements. As noticed for the tests on the rooms, the two models perform in a very similar way, with the same difference: again, the convolutional network seems to lead to lower losses when changing test set.

5.3 Position Correction

This section presents the main results of the last test conducted to verify the validity of the proposed method in a real 3D positioning scenario. As explained in 3.3, both LOS and NLOS test are conducted in two different rooms, one of which completely new. As description for the results of these tests, the graphs of the position estimates are reported as well as a comparison between the distance between the final outcome of the algorithms and the actual tag position. For clarity, only the results of the Extended Kalman Filter are reported, since it gives more accurate estimates than Gauss-Newton. As shown in 5.7 and in 5.5, mitigation is always improving the estimation accuracy in the big room. This demonstrates that the method is, at least, effective in the same scenario it has been trained for even if the position of the nodes and the configuration of the obstacles are different, and that correction on range values involves an improvement in position accuracy. In the new room even better results are obtained as the mitigated estimate is always closer than the raw one to the ground-truth reaching very high accuracy (about 8 cm in the LOS case and 7 cm in NLOS).

These results demonstrate that the proposed method provides generality with respect to room shape and dimensions, and hence can be a very useful tool in any indoor environment. Unfortunately, the test involves objects that are very similar to the ones used to build the dataset, so no further conclusion can be drawn on the effect of using different obstacles. However, the set of materials included in the dataset appears to be a good representation of what can be found in a typical office-like or domestic indoor environment, and further experimentations including new objects, and eventually people, can be conducted in future works.

Room	Condition	Mitigation	Error	ON/OFF %
Big Room	LOS	OFF	0,33751641	60,7952471
		ON	0,20519394	
	NLOS	OFF	0,12168457	86,7879935
		ON	0,1056076	
		OFF	0,37282038	61,8176997
		ON	0,23046899	
New Room	LOS	OFF	0,15784129	53,2814495
		ON	0,08410013	
	NLOS	OFF	0,47329691	41,0136498
		ON	0,19411634	
		OFF	0,21085436	34,0450994
		ON	0,07178558	

Table 5.7. Results of the positioning tests.

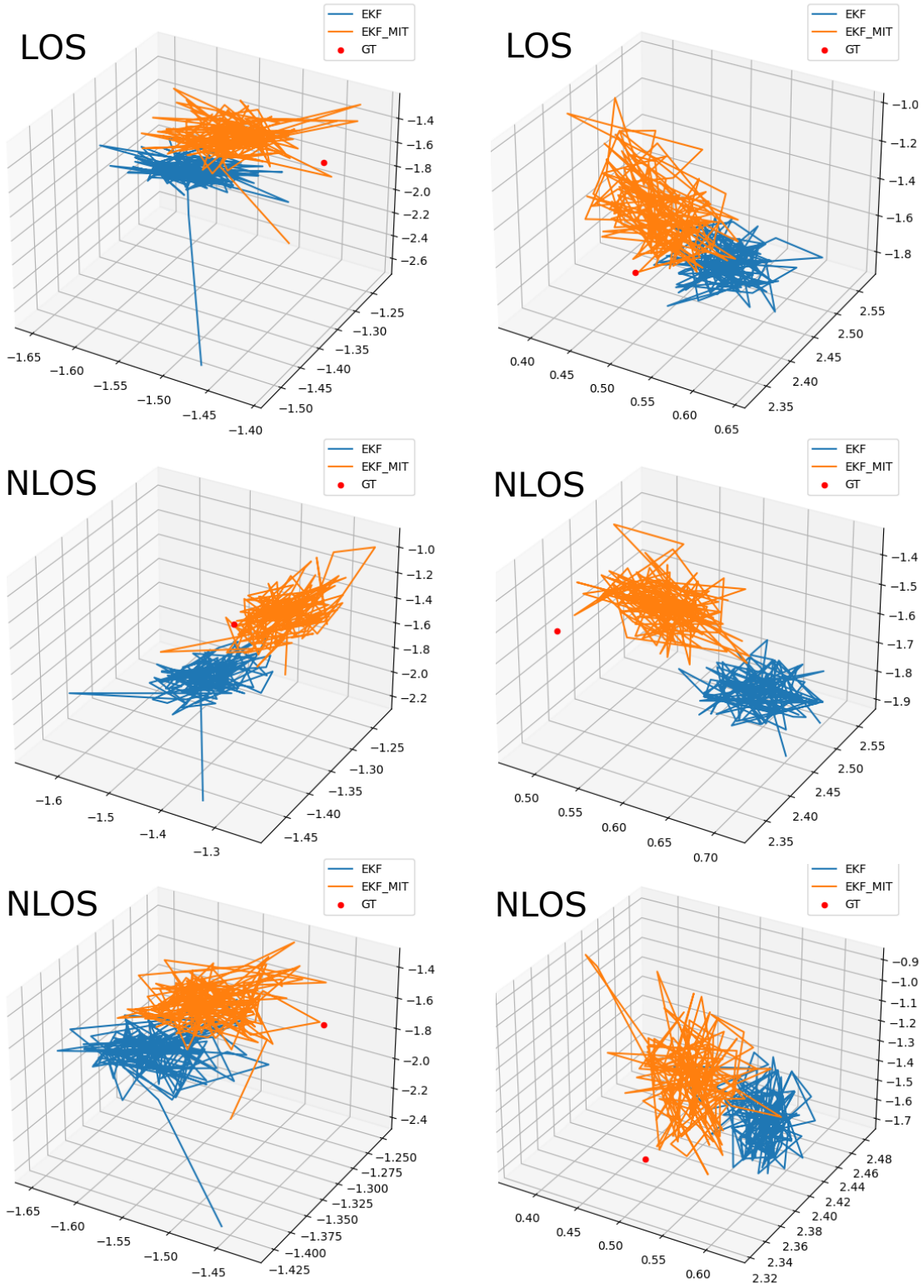


Figure 5.5. 3D representation of the position estimates: big room on the left column, new room on the right one.

Chapter 6

Conclusions

The principal aim of this thesis was to compensate NLOS and multipath errors in UWB positioning using deep learning models that automatically extract features from raw CIR samples in order to predict the error of the single range measurements. The interest in developing such a method lies in the great potentiality of deep algorithms, which are able to recognize complex underlying correlations in data and exploiting to learn patterns that generalize to a wide variety of situations. Hence, the goal was to obtain a compensation system scalable to unseen and unexpected scenarios. Finally, the work aimed at verifying the thesis that a correction on range estimates actually improves the accuracy of simple positioning algorithms and, if so, how much.

To guarantee a proper training for the deep learning models and to reach the maximum generalization, a dataset of over 55 thousand range samples has been built. The measurements covered a wide variety of LOS and NLOS scenarios, as five different environments and ten different obstacles have been included. A multi-layer perceptron and a CNN were chosen to be trained and tested on the dataset, and both proved the validity of the proposed idea. The results show that using the whole dataset to feed the models led to a reduction of the mean absolute error of over 45%, reaching the value of 6.7 cm. Such an outcome reached with these simple models is impressive considering that the MAE of the LOS ranges of the dataset is 5.9 cm. To have a more restrictive benchmark, another test has been conducted by excluding a room from the training set and using it only for testing. Although there is a further loss of about 1 cm, the result is still important as it proves that the influence of the environment can be learnt from deep models. Clearly, transfer knowledge from a room to an outdoor scenario causes a greater loss because of the completely different propagation of the signals and, in particular, because of the absence of multipath components. Still, an improvement of 27% for the MAE was detected. After the validity of the method had been verified, a series of cross tests were conducted to study the effect of different environments and obstacles on the performance of the neural networks. In summary, the main cause of the error given by the change of environment has resulted to be the reflections of the signal on walls and objects, and this is confirmed by the fact that the two most different settings were the outdoor and the small room, the latter being

largely influenced by multipath components. Between the rooms, instead, the difference in terms of performance was minimum. As far as obstacles are concerned, a more marked distinction was noticed: heavy objects (like metal and walls) have a very different impact on UWB signals than wood, plastic or cardboard. Therefore, a net trained for the former performs badly for the latter. The major problem, indeed, is found when a net trained on NLOS samples tries to correct LOS measurements. Nevertheless, an improvement of the raw MAE was detected in almost all cases: this means that the models are able to learn a way to compensate part of the error independently from the type of obstacle and that a dataset with a sufficient number of examples for a wide variety of materials can lead to excellent results in many different scenarios.

Finally, a positioning test was conducted in a typical real-time scenario, using Gauss-Newton and an Extended Kalman Filter for 3D localization. The estimation was performed simultaneously both on raw range measurements and corrected ones and the results have been compared. The test took place in two different rooms, one already used to build the dataset and one completely new. All the results of both rooms showed an improvement in the estimation accuracy, ranging between 13% and 65%, with a mean of 44%. It is interesting to notice that a significant enhancement can be achieved even for LOS scenarios, and that the results observed in the new room are better than those reported in the known one. This confirms that the shape and dimension of the environment affects the proposed method only marginally and that, even with very simple positioning algorithms, the compensation of range errors leads to an important improvement on the position estimation.

However, some critical aspects of the work and a few imprecisions must be reported and commented. Firstly, to a certain extent the dataset lacks in completeness, as pointed out by some of the results from the cross tests. As highlighted by the study on the distribution of the error in different subsets, the UWB sensors have rarely been challenged by critical situations. This caused the data splits to be quite similar with an MAE never over 20 cm. Including more screening materials in the measurement campaign and focusing more on walls and cross-room situations could have led to an additional strength for the dataset, improving its robustness. Secondly, another missing but useful aspect is the study of how human obstacles affect ranging precision, as the situation is very common in environments frequented by people. Thirdly, an additional weakness of the method is that little time has been dedicated to the optimization of the deep learning models in order to seek better and better results. Actually, the main scope of this work was to confirm the validity of the overall approach and to evaluate the influence of different measurement conditions on its accuracy, and for this reason the fine tuning of the learning methodologies has been left to future developments. As far as the localization tests are concerned, a criticality of the method as it is now is the precision of positioning algorithms. Indeed, two very simple techniques have been used, but their accuracy leaves room for improvement even in LOS conditions. That is why investing more time in multiple tests to tune the Extended Kalman Filter, for example, could have led to a better convergence and to smaller errors. Finally, the final tests suffer from the lack of completely new obstacles,

as very similar materials have been employed to create NLOS conditions. Evaluating the effectiveness of the proposed method on a set of different objects would have added scientific value to the whole work.

Since the validity of the deep learning approach has been demonstrated and a study of the effects of different factors on its accuracy has been conducted, a path for future research is open and a good number of goals of improvements can be identified. First of all, the weak points and limits of the present work, highlighted in the previous paragraphs, should be tackled, from the inclusion of uninvestigated measurement scenarios in the dataset to the developing of more precise localization algorithms that take advantage of ranging error compensation. Moreover, great importance resides on the possibility of implementing the compensation model and the positioning algorithm directly on the UWB sensors to perform the estimation refinement on board. For this reason, a deep analysis of the trade-off between model complexity and achieved accuracy should be conducted with particular attention to power consumption and physical dimensions. The results of this thesis, showing how very good accuracies can be achieved with simple models, pave the way to the design of an all-in-one compensated UWB positioning system.

Bibliography

[1] G. Breed, "A summary of FCC rules for ultra wideband communications," *High Frequency Electronics*, vol. 4, no. 1, pp. 42–44, 2005.

[2] S. A. Zekavat and M. Buehrer, *Handbook of Position Location: Theory, Practice, and Advances*. Piscataway, NJ, USA: Wiley - IEEE, 2011.

[3] D. Dardari, E. Falletti, and M. Luise, *Satellite and Terrestrial Radio Positioning Techniques: A Signal Processing Perspective*. Oxford, UK: Elsevier, 2012, p. 432.

[4] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: concept, tools, and techniques to build intelligent systems*. Newton, MA, USA: O'Reilly Media, 2017.

[5] M. Banko and E. Brill, "Scaling to very very large corpora for natural language disambiguation," Toulouse, France, 2001, pp. 26–33.

[6] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *in 3rd Int. Conf. on Learning Representations (ICLR 2015)*, pp. 1–15, 2015.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

[8] K. Gururaj, A. K. Rajendra, Y. Song, C. L. Law, and G. Cai, "Real-time identification of NLOS range measurements for enhanced UWB localization," *in Int. Conf. Indoor Positioning and Indoor Navigation (IPIN 2017)*, pp. 1–7, 2017.

[9] S. Al-Jazzar and J. Caffery, "ML & Bayesian TOA location estimators for NLOS environments," *in IEEE Vehicular Technology Conf.*, vol. 56, no. 2, pp. 1178–1181, 2002.

[10] Y. Jin, W. S. Soh, and W. C. Wong, "Indoor localization with channel impulse response based fingerprint and nonparametric regression," *IEEE Transactions on Wireless Communications*, vol. 9, no. 3, pp. 1120–1127, 2010.

[11] M. Kolakowski and J. Modelski, "Detection of direct path component absence in NLOS UWB channel," *in 22nd Int. Microwave and Radar Conf. (MIKON 2018)*, pp. 247–250, 2018.

[12] C. Wu, H. Hou, W. Wang, Q. Huang, and X. Gao, "TDOA Based Indoor Positioning with NLOS Identification by Machine Learning," *in 10th Int. Conf. Wireless Communications and Signal Processing (WCSP 2018)*, no. Ml, pp. 1–6, 2018.

[13] S. Maranò, W. M. Gifford, H. Wymeersch, and M. Z. Win, "NLOS identification and mitigation for localization based on UWB experimental data," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 7, pp. 1026–1035, 2010.

[14] W. Li, T. Zhang, and Q. Zhang, "Experimental researches on an UWB NLOS identification method based on machine learning," *in Int. Conf. on Communication Technology Proceedings (ICCT 2013)*, pp. 473–477, 2013.

[15] Y. Zhu, W. Xia, F. Yan, and L. Shen, "NLOS Identification via AdaBoost for Wireless Network Localization," *IEEE Communications Letters*, vol. 23, no. 12, pp. 2234–2237, 2019.

[16] Y. Freund, R. E. Schapire, P. Avenue, and F. Park, *A Short Introduction to Boosting*, Florham Park, NJ, USA, 1999.

[17] S. Krishnan, R. Xenia Mendoza Santos, E. Ranier Yap, and M. Thu Zin, "Improving UWB Based Indoor Positioning in Industrial Environments Through Machine Learning," in *in 15th Int. Conf. on Control, Automation, Robotics and Vision (ICARCV 2018)*, Singapore, Singapore: IEEE, 2018, pp. 1484–1488.

[18] Q. Liu, Z. Huang, and J. Wang, "Indoor non-line-of-sight and multipath detection using deep learning approach," *GPS Solutions*, vol. 23, no. 3, pp. 1–14, 2019.

[19] J. S. Choi, W. H. Lee, J. H. Lee, J. H. Lee, and S. C. Kim, "Deep Learning Based NLOS Identification with Commodity WLAN Devices," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 4, pp. 3295–3303, 2018.

[20] J. He, Y. Geng, F. Liu, and C. Xu, "CC-KF: Enhanced TOA performance in multipath and NLOS indoor extreme environment," *IEEE Sensors Journal*, vol. 14, no. 11, pp. 3766–3774, 2014.

[21] T. Nowak and A. Eidloth, "Dynamic multipath mitigation applying unscented Kalman filters in local positioning systems," *International Journal of Microwave and Wireless Technologies*, vol. 3, no. 3, pp. 365–372, 2011.

[22] J. Zhu and S. S. Kia, "Bias Compensation for UWB Ranging for Pedestrian Geolocation Applications," *IEEE Sensors Letters*, vol. 3, no. 9, pp. 24–27, 2019.

[23] C. D. Wann and C. S. Hsueh, "NLOS mitigation with biased Kalman filters for range estimation in UWB systems," Penang, Malaysia: IEEE, 2007, pp. 1–4.

[24] K. Wen, K. Yu, and Y. Li, "NLOS identification and compensation for UWB ranging based on obstruction classification," in *in 25th European Signal Processing Conf. (EUSIPCO 2017)*, Kos, Greece: IEEE, 2017, pp. 2704–2708.

[25] A. Haggenmiller, M. Krogius, and E. Olson, "Non-parametric error modeling for ultra-wideband localization networks," vol. 2019-May, Monréal, Canada: IEEE, 2019, pp. 2568–2574.

[26] A. Prorok, L. Gonon, and A. Martinoli, "Online model estimation of ultra-wideband TDOA measurements for mobile robot localization," in *in IEEE Int. Conf. on Robotics and Automation (ICRA 2012)*, St. Paul, MN, USA: IEEE, 2012, pp. 807–814.

[27] J. Li, I. T. Lu, J. S. Lu, and L. Zhang, "Robust kernel-based machine learning localization using NLOS TOAs or TDOAs," *in 2017 IEEE Long Island Systems, Applications and Technology Conf. (LISAT 2017)*, pp. 1–6, 2017.

[28] T. Van Nguyen, Y. Jeong, H. Shin, and M. Z. Win, "Machine Learning for Wideband Localization," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 7, pp. 1357–1380, 2015.

[29] Z. Xiao, H. Wen, A. Markham, N. Trigoni, P. Blunsom, and J. Frolik, "Non-Line-of-Sight Identification and Mitigation Using Received Signal Strength," *IEEE Transactions on Wireless Communications*, vol. 14, no. 3, pp. 1689–1702, 2015.

[30] H. Wymeersch, S. Maranò, W. M. Gifford, and M. Z. Win, "A machine learning approach to ranging error mitigation for UWB localization," *IEEE Transactions on Communications*, vol. 60, no. 6, pp. 1719–1728, 2012.

[31] V. Barral, C. J. Escudero, J. A. García-Naya, and R. Maneiro-Catoira, "NLOS identification and mitigation using low-cost UWB devices," *Sensors (Switzerland)*, vol. 19, no. 16, 2019.

[32] L. Schmid, D. Salido-Monzú, and A. Wieser, "Accuracy Assessment and Learned Error Mitigation of UWB ToF Ranging," in *in Int. Conf. on Indoor Positioning and Indoor Navigation (IPIN 2019)*, Pisa, Italy: IEEE, 2019, pp. 1–8.

[33] S. Ioffe and C. Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, 2015.

[34] V. Barral, C. J. Escudero, J. A. García-Naya, and P. Suárez-Casal, "Environmental cross-validation of NLOS machine learning classification/mitigation with low-cost uwb positioning systems," *Sensors*, vol. 19, no. 24, 2019.

[35] J. Tiemann, J. Pillmann, and C. Wietfeld, "Ultra-Wideband Antenna-Induced Error Prediction Using Deep Learning on Channel Response Data," in *in IEEE 85th Vehicular Technology Conf. (VTC Spring 2017)*, Sydney, Australia, 2017, pp. 1–5.

[36] A. Niitsoo, T. Edelhäußer, E. Eberlein, N. Hadaschik, and C. Mutschler, "A deep learning approach to position estimation from channel impulse responses," *Sensors*, vol. 19, no. 5, pp. 1–23, 2019.

[37] M. N. de Sousa and R. S. Thomä, "Enhancement of localization systems in nlos urban scenario with multipath ray tracing fingerprints and machine learning," *Sensors*, vol. 18, no. 11, 2018.

[38] J. Luo and H. Gao, "Deep Belief Networks for Fingerprinting Indoor Localization Using Ultrawideband Technology," *International Journal of Distributed Sensor Networks*, vol. 12, no. 1, 2016.

[39] P. Yazdanian and V. Pourahmadi, "DeepPos: Deep Supervised Autoencoder Network for CSI Based Indoor Localization," pp. 2–11, 2018. [Online]. Available: http://arxiv.org/abs/1811.12182.

[40] K. Bregar and M. Mohorcic, "Improving Indoor Localization Using Convolutional Neural Networks on Computationally Restricted Devices," *IEEE Access*, vol. 6, pp. 17 429–17 441, 2018.

[41] A. R. Jiménez Ruiz and F. Seco Granja, "Comparing Ubisense, BeSpoon, and DecaWave UWB Location Systems: Indoor Performance Analysis," *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 8, pp. 2106–2117, 2017.

[42] Leica, "Leica absolute tracker AT403 brochure," 2018.

[43] P. Corbalán, T. Istomin, and G. P. Picco, "Poster: Enabling Contiki on Ultra-wideband Radios," ser. in International Conference on Embedded Wireless Systems and Networks (EWSN 2018), 2018.

[44] *EVK1000 User Manual: How To Use, Configure and Interface To the DW1000*, 2013.

[45] *Application Note : Antenna Delay in DW1000-Based Products*, 2014.

[46] S. Angarano, *DeepUWB Dataset*. [Online]. Available: https://github.com/eimonfo/DeepUWB-Dataset.

[47] *Application Note : Sources of Error in DW1000 Based Two-Way Ranging (TWR) Schemes*, 2014. [Online]. Available: https://www.decawave.com/wp-content/uploads/2018/10/APS011_Sources-of-Error-in-Two-Way-Ranging-Schemes_v1.1.pdf.

[48] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter Notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt, Eds., IOS Press, 2016, pp. 87–90.

[49] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," *in 33rd Int. Conf. on Machine Learning (ICML 2016)*, vol. 3, pp. 1651–1660, 2016.

[50] D. Misra, "Mish: A Self Regularized Non-Monotonic Neural Activation Function," 2019. [Online]. Available: https://arxiv.org/pdf/1908.08681.pdf.

Appendix

Subset	MSE	MAE	Mean	Std. Deviation
All	0,0389572	0,12423939	0,10607736	0,16644909
Big room	0,04543307	0,12527636	0,10575347	0,18507061
Medium room	0,04114813	0,13911064	0,12435696	0,16026671
Small room	0,03049724	0,10685123	0,08813608	0,15076658
Outdoor	0,03635124	0,13446487	0,11236741	0,1540441
Cross-room	0,05323144	0,1658175	0,15745018	0,16873271

Table 6.1. Raw dataset metrics and graphs for different environments.

Subset	MSE	MAE	Mean	Std. Deviation
All	0,0389572	0,12423939	0,10607736	0,16644909
Light materials	0,01566282	0,0733789	0,04974112	0,11484878
Heavy materials	0,0775372	0,19614786	0,18050533	0,21203252
Wall	0,05323144	0,1658175	0,15745018	0,16873271
Wood	0,01165025	0,07835419	0,06781103	0,08398317
LOS	0,00597477	0,05942106	0,03003114	0,07123189

Table 6.2. Raw dataset metrics for different materials.

Model	MSE	MAE	MSE %	MAE %	Mean %	Std. Deviation %
MLP	0,01411715	0,06753542	36,237585	54,3591065	3,658232598	70,5067287
CNN	0,01549805	0,06748596	39,7822439	54,3192962	0,81462759	72,4951442

Table 6.3. Results obtained from the initial tests on the whole dataset.

Train Set	Test Set	MSE	MAE	MSE %	MAE %	Mean %	Std. Deviation %
All - Room 1	Room 1	0,02666005	0,08040226	58,6798374	64,179912	7,38334447	88,1267432
All - Room 2	Room 2	0,01539912	0,0797291	37,4236211	57,3134432	14,2513598	76,6383726
All - Room 3	Room 3	0,01562999	0,07319096	51,2505097	68,4980096	7,54634576	82,8077845
All - Outdoor	Outdoor	0,02261999	0,10383502	62,2261771	77,2209253	33,4999017	94,5360045
All - Cross-room	Cross-room	0,01608017	0,08803432	30,2080286	53,0910907	21,9676362	72,341225

Table 6.4. Results obtained from the restrictive benchmark computed excluding a single room from the training dataset and testing on that room (with the MLP).

Room	Big Room		Medium Room		Small Room		Outdoor		Cross-room	
Big Room	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
	0,02467988	0,07787607	0,0161449	0,08550771	0,01646831	0,07873359	0,02134021	0,10559664	0,01862305	0,09618016
	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %
	54,321404	62,1634183	39,2360486	61,46741	53,99934948	73,68524538	58,7055824	78,5310221	34,98505474	58,00362399
	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %
	4,01798708	83,9266147	11,2294558	78,8046762	5,93640802	85,0492603	32,8329907	91,76694462	18,0056615	79,15423904
Medium Room	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
	0,02736029	0,08834743	0,01405944	0,07738681	0,018028569	0,08228663	0,01941089	0,10208596	0,01922299	0,09822902
	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %
	60,2210937	70,5220261	34,1678716	55,6296827	59,11541741	77,0104632	53,3981438	75,92016925	36,11209535	59,23923542
	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %
	9,906148841	89,1993915	22,4620665	68,7190741	3,609721626	89,0360708	3,9469242	90,40682772	23,7169094	79,17468705
Small Room	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
	0,026333	0,08452915	0,01567311	0,08284867	0,01359128	0,06609059	0,03565791	0,12467928	0,01812906	0,09493756
	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %
	57,9599873	67,4741407	38,0894837	59,5559531	44,56560986	61,852906	98,0926792	92,72256478	34,05705061	57,25424591
	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %
	6,26009362	87,6118449	10,6473349	77,6797807	4,336044147	79,695335	56,6666499	115,4159095	19,3308738	77,7725764
Outdoor	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
	0,04722738	0,12747252	0,04153212	0,14105202	0,032127373	0,11191515	0,00702039	0,05970919	0,05371977	0,16872023
	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %
	103,949354	101,753049	100,933191	101,395562	105,3451891	104,7392252	19,31265361	44,4050466	100,9173628	101,7505562
	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %
	98,95735456	102,915679	99,7306318	100,905428	101,4142107	103,052646	13,8234327	53,96402088	99,6639222	101,1444023

Table 6.5. Results obtained by the MLP in the environmental influence test.

Room	Big Room		Medium Room		Small Room		Outdoor		Cross-room	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
	0,02430015	0,07523015	0,01309175	0,0740025	0,01437539	0,06646964	0,01670577	0,08992542	0,01573104	0,08676665
	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %
Big Room	53,485603	60,0513518	31,81614868	53,19686386	47,1366896	62,2076511	45,9565279	66,8765137	29,5521569	52,3265936
	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %
	15,110193	81,9384443	12,5815811	70,7250223	17,1183779	78,89529079	25,39284535	81,8432215	19,19075614	72,18115906
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
	0,02354875	0,0774286	0,01094638	0,06630033	0,01503517	0,070367329	0,0131093	0,08692176	0,01038965	0,07260749
	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %
Medium Room	51,8317415	61,806232	26,60237582	47,6601416	49,3000983	65,8554232	36,062864	64,6427259	19,517881	43,7875914
	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %
	0,13801662	82,9197794	3,36966249	64,5605494	8,0473112	81,19582276	8,086138577	74,09969927	7,830113333	59,99689508
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
	0,02497546	0,09225181	0,01449335	0,08785396	0,015953	0,0782483	0,02017894	0,10785439	0,01393093	0,09242873
	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %
Small Room	54,9719874	73,6386396	35,22237886	63,15401709	52,3096555	73,2310718	55,5110013	80,2100851	26,1704903	55,7412392
	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %
	39,360392	82,3794213	38,111729	69,0541622	43,7128376	75,41238417	56,54131915	82,48638708	30,36263488	63,98935403
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
	0,0394199	0,13824739	0,02412871	0,11858151	0,02865731	0,123739503	0,0114687	0,07872491	0,02172962	0,11046717
	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %
Outdoor	86,7647781	110,353929	58,63865601	85,24258564	93,9669173	115,80541	31,5496761	58,5468216	40,8210226	66,619729
	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %
	75,6578964	98,1862776	51,1485774	88,4272579	91,5040587	98,72462388	10,4019234	66,00072511	19,27673939	85,53581282

Table 6.6. Results obtained by the CNN in the environmental influence test.

Obstacle	Light Materials		Heavy Materials		Wood		LOS		Wall	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Light Materials	0,01045379	0,06050171	0,05175483	0,14163822	0,00784166	0,06247131	0,00829849	0,07362957	0,02828939	0,1165821
	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %
	66,7427234	82,451099	66,7483855	72,2099226	67,30892259	79,72938314	138,8922121	123,911563	53,14413362	70,30747601
	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %
	67,51573966	89,2785766	45,4984634	100,061087	34,30785473	101,7467	186,015122	101,0293567	28,0881473	96,22395323
Heavy Materials	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
	0,01826684	0,08750228	0,03009475	0,0955946	34,30785473	101,7467	0,0098499	0,07795361	0,01659004	0,09275154
	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %
	116,625516	119,247194	38,8133045	48,7359885	84,597302	95,50499191	164,8582333	131,188511	31,16586475	55,93591704
	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %
	73,46752963	113,304334	6,94601505	79,0664258	49,01959492	111,396557	161,549637	121,5602825	4,84908724	76,24086328
Wood	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
	0,01405245	0,06828006	0,04891188	0,13794212	0,00386083	0,04708451	0,00632522	0,06202191	0,02532679	0,1058047
	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %
	89,7185407	93,0513533	63,0818229	70,3255788	33,13945103	60,09188759	105,8655006	104,3769753	47,57862619	63,80792083
	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %
	27,96187221	102,509971	42,9101256	97,7019605	6,069968276	80,4054303	122,09439	99,08805464	29,643663	90,21698562
LOS	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
	0,01397264	0,06650509	0,06122711	0,160725	0,00721748	0,05933153	0,00351147	0,04647238	0,03645505	0,13225417
	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %
	89,2089899	90,6324427	78,9648182	81,9407347	61,95127085	75,72222013	58,77163631	78,20859529	68,48405174	79,7588728
	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %
	35,49649416	101,774624	73,2458591	98,6471616	48,13130182	93,4032991	1,69536716	81,74716599	69,5577252	92,73909205

Table 6.7. Results obtained by the MLP in the obstacle influence test.

Obstacle	Light Materials		Heavy Materials		Wood		LOS		Wall	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Light Materials	0,00921336	0,05569007	0,05589486	0,14504856	0,0052337	0,04642639	0,00480115	0,05482112	0,02667527	0,10518509
	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %
	58,8231386	75,8938463	72,08779667	73,94858033	44,9234866	59,251958	80,3570703	92,2587306	50,1118657	63,434251
	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %
	26,9165842	90,5483023	61,1202964	98,6203849	9,84021641	85,78185466	106,1012313	86,3883175	42,5779291	88,31193996
Heavy Materials	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
	0,02191028	0,09977915	0,00921336	0,05569007	0,01363681	0,08753617	0,01111186	0,08222251	0,0133337	0,09071007
	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %
	139,887233	135,977984	11,88250265	28,3918821	117,051618	111,718561	185,979716	138,372664	25,0485406	54,7047623
	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %
	95,6308065	122,054724	26,9165842	90,5483023	76,6906387	124,5095234	178,3709499	127,4676858	27,44373552	63,4958627
Wood	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
	0,01184395	0,05508218	0,0518572	0,1458458	0,00317958	0,04099512	0,00335696	0,04511035	0,02262894	0,10455675
	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %
	75,6182665	75,0654201	66,88041243	74,3550288	27,2919387	52,3202672	56,1855953	75,9164283	42,5104752	63,0553163
	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %
	4,80179883	94,7422583	61,5831062	93,7372196	27,5766114	63,85572782	50,44688146	78,51753265	49,18174786	76,47282215
LOS	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
	0,01267488	0,05851574	0,06250627	0,16230033	0,00720045	0,05803423	0,00300621	0,04175818	0,03675226	0,12892334
	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %	MSE %	MAE %
	80,923379	79,7446399	80,6145553	82,74386861	61,8050938	74,0665332	50,3150762	70,2750451	69,0423871	77,7501403
	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %	MEAN %	STD %
	58,0768372	94,7507312	78,7293078	97,0145179	70,2089218	83,64446292	42,69144567	72,30024418	73,63160511	90,53492619

Table 6.8. Results obtained by the CNN in the obstacle influence test.