TECHNICAL UNIVERSITY OF CRETE
ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT
TELECOMMUNICATIONS DIVISION

# RFID Localization with Multistatic Interrogation and Neural Networks

by

Georgios Papadopoulos

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DIPLOMA OF

ELECTRICAL AND COMPUTER ENGINEERING

March 2024

THESIS COMMITTEE

Professor Vasilis Digalakis, *Thesis Supervisor*
Professor Aggelos Bletsas
Professor Michail G. Lagoudakis

# Abstract

The use of low-cost, batteryless radio frequency identification (RFID) tags for efficient communication and localization via signal backscattering, has been recently established in research and industry scenarios. This work draws inspiration from a recently proposed multistatic localization technique, that exploits the form of ellipses in the plane, using phase information and potentially, (elliptical) direction-of-arrival.

A 2D localization scheme is introduced, that exploits neural networks (NNs) for improved accuracy. Specifically, a Deep Feed-Forward Neural Network is developed that performs regression for tag location estimation from phase-based measurements as input, offered from a custom multistatic RFID interrogation setup.

In the first part, traditional techniques are contrasted with neural networks trained with $10^6$ samples and, through simulation comparisons across an 8x4 $m^2$ area, a pronounced advantage for the latter is demonstrated for random tag positions. Then, a substantially improved setup is presented which, by utilizing only one extra antenna, yields a remarkable 76% reduction in mean absolute error (MAE), in the order of 2.48 cm, while median absolute error is below 1 cm. Contrary to bibliography, it is proven that this outcome is not due to phase ambiguity, but the determining factor is the increased difference in the values of the phase-based inputs within the tested area, thus giving the neural network more diverse information to be trained upon.

In the second part, experimental data are used to evaluate the efficiency of the neural networks in two real-world scenarios, with tags being positioned up to 1.2 meters away from the antennas and 44 samples taken in total. It is remarked that the system is trained with simulated data, while the experimental data mentioned above are only used for testing. This time, the NNs perform in a comparable fashion to the traditional methods, in both experimental scenarios, with MAE ranging from 19 cm (in the first) to 27 cm (in the second scenario) and the median within a centimeter of each other, at approximately 8.5 cm.

Two major conclusions can be drawn from this thesis; firstly that NNs trained exclusively with simulated data can perform reasonably well with experimental (real-world) measurements, comparably to state-of-the-art methods; secondly, there is plenty room for improvement, by using experimental data for training, perhaps with a tag in a moving robotic platform, and carefully selected reader antenna placements, as already shown in simulations.

# Acknowledgements

First and foremost I would like to thank *Prof. Vasilis Digalakis* and *Prof. Michail G. Lagoudakis* of the committee for their cooperation, given the unique situation created due to *Prof. Aggelos Bletsas*, my original supervisor, being on leave the last two months. As he was not eligible to serve as the official supervisor, a change had to be made at the last moment and I am grateful that it could all happen in time.

To *Prof. Aggelos Bletsas*, I would like to express him my gratitude, for all the lessons and the guidance he gave me throughout this journey and for pushing me to achieve what I have.

To all the *lab colleagues* I met, I thank them for helping me when I struggled and for making me feel comfortable. A huge thank you to *Dimitris Aggelou*, who went out of his way to take all the experimental measurements for this thesis. I really owe you one my friend.

To the *Entasi Radio* university team, you taught me a lot of lessons, made me get out of my comfort zone much more than I could ever imagine myself do and showed me a completely new world that I adored.

To all the wonderful people I met through these years, I consider myself immensely lucky to have met you and to call you *friends.* And to my partner *Mary*, thank you for being there through the ugly and the beautiful.

Finally, all these would not have been possible without the support of my *family.* Thank you for supporting me and for believing in me all these years.

# Table of Contents

# List of Figures

# Chapter 1

# Introduction

The ever-increasing need to automate procedures in many aspects of everyday life, has led to a tremendous increase in the usage of radio frequency identification (RFID) tags. These tags can either be active, semi-passive (battery-assisted) or passive (batteryless), with the latter being used in this work. Passive tags are characterized by their low cost, ease of installation, and batteryless operation, finding applications in item detection and data collection in the supply chain, tickets for public transportation, credit cards, etc., robotic systems, livestock and agriculture, as well as in event management, in open or closed spaces, such as festivals and museums, respectively.

An essential component for RFID tag interrogation is the RFID reader, responsible for transmitting and receiving signals to and from the tag. Two main architectures, monostatic and multistatic, are employed for this purpose. In monostatic setups, a single device handles both transmission and

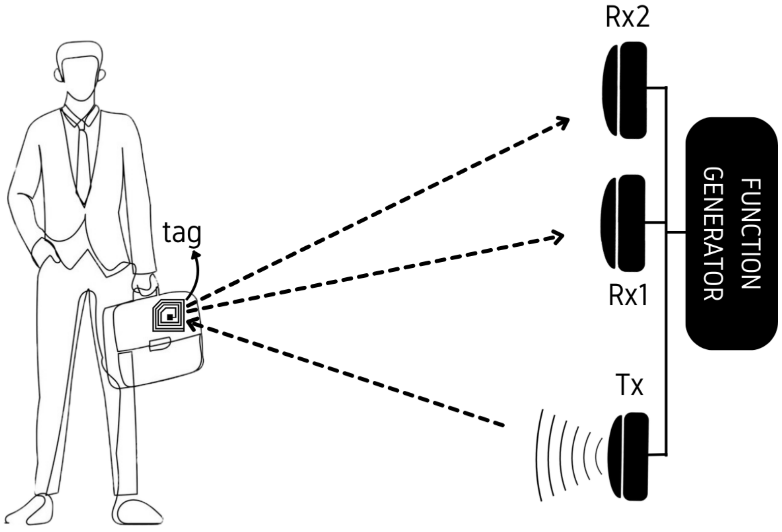


Figure 1.1: Multistatic interrogation of an RFID tag.

reception, while multistatic setups use separate devices, as seen in Fig. 1.1. Although they offer wider coverage and greater accuracy, challenges such as carrier frequency offset and timing constraints are introduced in the system, which demand special techniques to be solved. In this work, both scenarios are analyzed, while a multistatic setups will be used for further experimentation.

In addition to the successful interrogation of tags for information retrieval, many applications also require accurate estimation of the tag's location, to track items or predict their trajectories. Some combination of Received Signal Strength Indicator (RSSI), phase measurements and reference tags, are mainly used to tackle this problem, in both monostatic and multistatic setups, with the latter obviously being the preferred, as more information can be extracted.

In recent years, the surge in the use of neural networks, has sparked new interest in the localization field as well, as more complex systems can be utilized and even more information can be extracted from the existing ones. This work relies on the multistatic localization technique introduced at [1], which exclusively uses phase measurements, and aims to improve its accuracy, by introducing neural networks and experimenting with the antenna setup.

## 1.1 Gen2 Protocol

The Gen 2 Protocol, as defined in [2], governs the communication and operation between RFID readers and tags, which is based on backscatter communication. Each RFID tag is equipped with an antenna that can toggle between two states, $Z_0$ and $Z_1$. In the low state ($Z_0$), the antenna operates as an open-circuit, maximizing its reflection coefficient to backscatter the signal to the receiver. In the high state ($Z_1$), "perfect" matching occurs, setting the reflection coefficient to zero and maximizing power transfer for RF energy harvesting, where the tag absorbs energy from the signal.

The Gen 2 Protocol also defines crucial time constraints that must be satisfied for successful tag interrogation, employing the framed slotted Aloha (FSA) algorithm. Firstly, the transmission of a continuous wave (CW) by the reader, activates nearby tags. Subsequently, the QUERY command is transmitted to the tag to initialize various parameters, including tag rate, line code, and number of FSA slots. Assuming a single tag is present and a single FSA slot is available, a random 16-bit sequence (RN16) is generated and backscattered from the tag, triggering the transmission of an acknowledgment (ACK) message. The successful matching of RN16 and ACK sequences indicates a successful handshake, leading the tag to backscatter its 128-bit payload, which includes the electronic product code (EPC) – a 96-bit identification information. Finally, after the tag backscatters a message, a known preamble sequence is transmitted, providing valuable information for channel estimation.

## 1.2 Miller Line Coding

Line coding is used in Gen2 tags, in order to ensure the reliable function of the tags and consequently the reliable communication. As noted before, Gen2 tags are batteryless and harvest energy when they operate on state $Z_1$. Line codes guarantee a perfect 50/50% distribution of the duty cycle between states $Z_0$ and $Z_1$, so that regardless of the original bit sequence, enough energy will always be absorbed, while of course keeping the information intact. Another benefit of using line codes is the introduction of memory to the signal, which helps to distinguish noise signal from real tag response, enhancing data detection.

In Miller line codes, there are four possible waveforms for each encoded bit; two for bit-1, denoted as 1-High and 1-Low ($\mathbf{S}_{m1}, \mathbf{S}_{m2}$) and two for bit-0, denoted as 0-High and 0-Low ($\mathbf{S}_{m3}, \mathbf{S}_{m4}$)[1]. The transition from one symbol to the next always changes the line level (High → Low, Low → High), except from when we are transitioning from bit-0 to bit-1, where the line level remains the same. Furthermore bit-0 waveforms must change level in the middle of the bit, whereas bit-1 waveforms will have constant level at the middle of the bit.

---

[1] Keywords "High" and "Low" refer to the line level at the start of the waveform.

Focusing on Miller 2 line encoding ($m = 2$), the four possible waveforms—based on the above rules—are as follows:

$$\mathbb{S}_2 = \{\mathbf{S}_{21}, \mathbf{S}_{22}, \mathbf{S}_{23}, \mathbf{S}_{24}\} = \{10\,01,\ 01\,10,\ 10\,10,\ 01\,01\},$$

where the first two correspond to bit 1 and the last two to bit 0. So in order to encode one bit we need a sequence of four zeros and ones. In order to make the distinction between the bit and this smaller quantity, the notation of the chip is established, indicating each ”low” or ”high” line level of the bit, the duration of which is considered to be constant. Hence, each bit on Miller 2 will consist of four chips:

$$T_b = 4\,T_{chip} \tag{1.1}$$

Accordingly, for Miller 4 (8) line coding there will be 2× (4×) more chips per bit, thus the bit duration will increase.

Returning now on the $\mathbb{S}_2$ waveforms of Miller 2, it can be seen that they can be produced in turn by two, worm-like, sub-waveforms denoted as:

$$S_0 = \begin{cases} 1, & \text{if } 0 \leq t \leq \frac{T_w}{2} \\ 0, & \text{if } \frac{T_w}{2} \leq t \leq T_w \end{cases},\ S_1 = \begin{cases} 0, & \text{if } 0 \leq t \leq \frac{T_w}{2} \\ 1, & \text{if } \frac{T_w}{2} \leq t \leq T_w \end{cases},$$

where $T_w = \frac{1}{BLF}$ is the duration of the sub-waveform and $BLF$ is the backscatter link frequency. So the time needed to transmit one whole bit, will be equal to $T_b = 2\,T_w = \frac{2}{BLF}$ [2]. Finally, the oversampling factor, denoting the number of samples per bit, will be equal to:

$$L = \frac{T_b}{T_s} = \frac{2}{T_s \times BLF} \tag{1.2}$$

## 1.3 Thesis Organization & Contributions

In Chapter 2, monostatic and multistatic system models, with and without synchronized software-defined radios (SDRs), are analyzed in detail, with the individual parts of the signal, the noise model and the derived symbol-to-noise ratio being thoroughly explained. Methods to compensate for the problems that non-synchronized SDRs introduce and DC and channel estimation techniques, are also offered.

In Chapter 3, the EllDoA method [1] for multistatic 2D localization, which exploits the form of ellipses to estimate the tag position, is analyzed. The concept of ambiguity in the phase measurements is introduced, while an additional localization method which only takes advantage of the DoA values is also mentioned.

In Chapter 4, a localization technique that uses neural networks is offered, while simultaneously a brief introduction to the structure of a neural network is given, with the various parameters that compose it being explained. The choices that were made during the construction procedure are elucidated and more complex setups, that have additional antennas and create ambiguous data, are presented.

In Chapter 5, experiments with simulated data are being conducted, for both the neural networks and the traditional methods, in multiple scenarios. The results are then compared, in order to assess the efficiency benefits of the neural networks. Different topologies tested, show that increasing the distance between antenna pairs has a huge benefit in the efficiency, while on the most part, NNs outperformed the traditional methods, reaching as low as 2.5 cm mean absolute error (MAE), for random tag positions in an 8x4 $m^2$ area. Experimental evaluation is also provided for tags in a smaller area (up to 1.2 m away from the antennas), with data from two real-world scenarios. NNs again prove their robustness performing comparably to the traditional methods in both testing areas, while being trained exclusively on simulated data.

In Chapter 6, the summary of the thesis is presented, with the two main findings being the viability of the NNs both in simulations and real-world experiments and the remarkable effect that the increased distance between the receiving antenna pairs had on the simulation results. Ideas and directions for future research, upon this work, are also provided.

## 1.4 Notation

Symbols $\mathbb{N}$, $\mathbb{R}$, $\mathbb{C}$ represent the set of natural, real and complex numbers, respectively. $\mathcal{CN}(\mathbf{m}, \mathbf{C})$ refers to the proper complex Gaussian distribution with mean vector $\mathbf{m}$ and covariance matrix $\mathbf{C}$. $\Re(w)$ and $\Im(w)$ refer to the real and imaginary part of a complex scalar $w$, respectively, while $\angle w$ refers to the phase. Expectation of function $g(\cdot)$ of continuous random variable $x$ with probability density function (PDF) $f_x(\cdot)$ is denoted as $\mathbb{E}[g(x)] = \int_x g(x) f_x(x) dx$.

# Chapter 2

# System Model & Housekeeping Tasks

## 2.1 System Model

For the entirety of this work a multistatic setup of a single transmitter (Tx) antenna and $M > 1$ receiver (Rx) antennas is assumed, except for a brief mention in the monostatic equivalent.

### 2.1.1 Channel Model

*Large-scale* channel path-loss is modeled by the following equation, as per [3]:

$$L_q = \left(\frac{\lambda}{4\pi d_0}\right)^2 \left(\frac{d_0}{d_q}\right)^{v_q}, \tag{2.1}$$

where $q \in \{\mathrm{CR}_i,\ \mathrm{CT},\ \mathrm{TR}_i\}$ denotes the carrier-to-$i$-th receiver, carrier-to-tag and tag-to-$i$-th receiver links, respectively, $\lambda$ is the carrier wavelength, $d_q$ is the distance between the components of each link, $d_0$ is a reference distance and $v_q$ is the path-loss exponent for link q. When $d_0 = 1m$ and $v_q = 2$, we get the free space channel path-loss, which assumes no multipath effect:

$$L_q = \left(\frac{\lambda}{4\pi d_q}\right)^2 \tag{2.2}$$

Due to strong line-of-sight (LoS) signals present in this problem, *small-*

*scale* Rice flat fading channel model [3] is adopted:

$$h_q \sim \mathcal{CN}\left(\sqrt{\frac{k_q}{k_q+1}\sigma_q^2},\ \frac{\sigma_q^2}{k_q+1}\right), \tag{2.3}$$

where $\sigma_q^2 = \mathbb{E}\left[|h_q|^2\right]$ is the average power of the scattering components and $k_q > 0$ is the power ratio between the deterministic LoS component and the scattering components. For $k_q = 0$ Rayleigh fading is obtained.

The end-2-end complex channel gain for path $i$ is denoted as:

$$h_i = h_{\text{CT}}\, h_{\text{TR},i} = |h_{\text{CT}}\, h_{\text{TR},i}|\, e^{-j\phi_i} \in \mathbb{C} \tag{2.4}$$

where $\phi_i = \phi_{\text{CT}} + \phi_{\text{TR},i} \in [0, 2\pi)$, $h_{\text{CT}}$ and $h_{\text{TR},i}$ denotes the baseband complex channel coefficients for the Tx (carrier emitter) antenna-to-tag and tag to-$i$-th Rx (receiver) antenna link, respectively.

### 2.1.2 Signal Model with Unsynchronized SDRs

As the RFID readers are not synchronized, the system will suffer from induced frequency and phase terms, denoted as carrier frequency offset (CFO) and carrier phase offset (CPO). Therefore, the carrier-emitter will transmit a continuous wave (CW) of frequency $F_c$, whose baseband equivalent is given by:

$$c(t) = \sqrt{2P_c}\, e^{-j(2\pi\Delta f t + \Delta\phi_R)}, \tag{2.5}$$

where $\Delta f$ and $\Delta\phi_R$ are the CFO & CPO, respectively, and $P_c$ is the power of the carrier wave. The scattered waveform from the tag, as it receives the continuous wave passed through the carrier-to-tag link (CT), is given by [4]:

$$u(t) = \sqrt{\eta L_{\text{CT}}}\, h_{\text{CT}} \left[(A_s - \Gamma_0) + (\Gamma_0 - \Gamma_1)\, x_{tag}(t)\right] c(t), \tag{2.6}$$

where $x_{tag}(t) \in \{0, 1\},\ \eta \in \{0, 1\}$ models tag power scattering efficiency, $A_s \in \mathbb{C}$ symbolizes tag's antenna load-independent structural mode parameter [5] and $\Gamma_0,\ \Gamma_1$ stands for the reflection coefficients for bit "0" and bit "1", respectively.

The received signal at the $i$-th Rx antenna will be the superposition of the CW in Eq. (2.5) passed directly to the reader, through the CR$_i$ link and the backscattered signal from the tag in Eq. (2.6), that will pass through the TR$_i$ link:

$$\begin{aligned} y_i(t) =& \sqrt{L_{\mathrm{CR},i}}\, h_{\mathrm{CR},i}\, c(t) + \sqrt{L_{\mathrm{TR},i}}\, h_{\mathrm{TR},i}\, u(t) + n_i(t) \\ =& \Big[ \sqrt{2P_c} \Big( \sqrt{L_{\mathrm{CR},i}}\, h_{\mathrm{CR},i} + \sqrt{\eta L_{\mathrm{CT}}\, L_{\mathrm{TR},i}}\, (A_s - \Gamma_0)\, h_{\mathrm{CT}}\, h_{\mathrm{TR},i} \Big) \\ & + \sqrt{2P_c\, \eta L_{\mathrm{CT}}\, L_{\mathrm{TR},i}}\, (\Gamma_0 - \Gamma_1)\, h_{\mathrm{CT}}\, h_{\mathrm{TR},i}\, x_{tag}(t) \Big] \\ & \cdot\, e^{-j(2\pi \Delta f t + \Delta\phi_R)} + n_i(t) \end{aligned} \tag{2.7}$$

The term inside the brackets can be divided into a time-independent DC term and the time dependent term, which includes the tag bit:

$$m_{dc,i} = \sqrt{2P_c} \left( \sqrt{L_{\mathrm{CR},i}}\, h_{\mathrm{CR},i} + \sqrt{\eta L_{\mathrm{CT}}\, L_{\mathrm{TR},i}}\, (A_s - \Gamma_0)\, h_{\mathrm{CT}}\, h_{\mathrm{TR},i} \right) \tag{2.8}$$

$$m_{tag,i} = \sqrt{2P_c\, \eta L_{\mathrm{CT}}\, L_{\mathrm{TR},i}}\, (\Gamma_0 - \Gamma_1)\, h_{\mathrm{CT}}\, h_{\mathrm{TR},i} \tag{2.9}$$

The discrete time equivalent, gives us the following:

$$y_i[k] = (m_{dc,i} + m_{tag,i}\, x_{tag}[k])\, e^{-j(2\pi \Delta f k T_s + \Delta\phi_R)} + n_i[k], \tag{2.10}$$

where $T_s$ is the receiver's sampling period. Thermal noise $n_i[k]$ follows the circularly-symmetric complex Gaussian distribution and is modeled by $n_i[k] \sim \mathcal{CN}\left(0, \sigma_{n_i}^2\right)$, with different noise samples being independent. The power spectral density of the noise follows:

$$S_{nn}(F) = \begin{cases} \dfrac{N_0}{2} & , \quad |F| \leq W \\ 0 & , \quad \text{otherwise} \end{cases}, \tag{2.11}$$

where $W$ stands for receiver's bandwidth and $N_0 = k_b T_\theta$, with $k_b$ and $T_\theta$ denoting the Boltzmann constant and receiver temperature, respectively. The noise, being complex, has a real and imaginary part, which follow normal distribution, with zero mean and equal variance $\sigma^2_{\Re(n_i[k])} = \sigma^2_{\Im(n_i[k])} \equiv \sigma^2_{part,i}$. In order for that to be true we solve the Pythagorean theorem:

$$\sigma^2_{n_i} = \sigma^2_{\Re(n_i[k])} + \sigma^2_{\Im(n_i[k])} = 2\,\sigma^2_{part,i}$$

$$\Rightarrow \sigma^2_{part,i} = \frac{\sigma^2_{n_i}}{2}$$

So to produce the noise samples, we will multiply two normally distributed random numbers (one for each plane) with $\sqrt{\frac{\sigma^2_{n_i}}{2}} = \frac{\sqrt{2}\sigma_{n_i}}{2}$.

### 2.1.3 Signal Model with Synchronized SDRs

Assuming synchronized SDRs, using an external function generator, eliminates CFO and CPO among the different Tx and Rx antennas, thus simplifying Eq. (2.10) to the following:

$$y_i[k] = (m_{dc,i} + m_{tag,i}\, x_{tag}[k]) + n_i[k] \tag{2.12}$$

In order to calculate the symbol-to-noise ratio, we must first remove/block the DC term using a DC-blocking filter, as it does not contribute any information on the transmitted data.

$$\tilde{y}_i[k] = m_{tag,i}\, x_{tag}[k] + n_i[k] \tag{2.13}$$

Symbol-to-noise ratio definition follows:

$$\begin{aligned}\mathbf{SNR} &= \frac{\mathbb{E}\left[|m_{tag,i}\, x_{tag}[k]|^2\right]}{\mathbb{E}\left[|n_i|^2\right]} \\ &= \frac{2P_c\,\eta\, L_{\mathrm{CT}}L_{\mathrm{TR},i}\,(\Gamma_0-\Gamma_1)^2\,\mathbb{E}\left[|h_{\mathrm{CT}}\, h_{\mathrm{TR},i}|^2\right]\;\mathbb{E}\left[|x_{\mathrm{tag}}|^2\right]}{2N_0WT_s}.\end{aligned}$$

Due to Miller line coding it stands true that $\mathbb{E}[|x_{\mathrm{tag}}|^2] = \int_0^T |x_{\mathrm{tag}}|^2 dt = \frac{T}{2}$ and assuming $\mathbb{E}[|h_q|^2] = 1,\ q \in \{\mathrm{CR}, i, \mathrm{CT},\ \mathrm{TR}, i\}$, we finally get:

$$\mathbf{SNR} = \frac{P_c\,\eta\, L_{\mathrm{CT}}L_{\mathrm{TR},i}\,(\Gamma_0-\Gamma_1)^2\,T}{2N_0WT_s} \tag{2.14}$$

Carrier-to-noise ratio is denoted as:

$$\mathbf{CNR} = \frac{\mathbb{E}\left[|m_{dc,i}|^2\right]}{\mathbb{E}\left[|n_i|^2\right]} \tag{2.15}$$

### 2.1.4 Monostatic Setup

The monostatic setup, naturally, simplifies the signal model, as there is no need for synchronization between the different SDRs. Thus $\Delta f = \Delta\phi_R = 0$ and furthermore the *small* and *large-scale* models are simplified too, with $L_{\mathrm{CT}} = L_{\mathrm{TR}} = L_i$, $h_{\mathrm{CT}} = h_{\mathrm{TR}}$, giving us $h_i = h_{\mathrm{CT}}^2$ and $h_{\mathrm{CR}}$ now referring to the leakage in the CR link. Applying the above in Eq. (2.7) gives us for the $i$-th antenna:

$$\begin{aligned} y_i(t) =& \Bigg[\sqrt{2P_c}\Big(\sqrt{L_{\mathrm{CR},i}}\, h_{\mathrm{CR},i} + \sqrt{\eta}\, L_i(A_s - \Gamma_0)\, h_i\Big) \\ &+ \sqrt{2P_c\,\eta}\, L_i(\Gamma_0 - \Gamma_1)\, h_i\, x_{tag}(t)\Bigg] + n_i(t) \end{aligned} \tag{2.16}$$

The corresponding SNR value follows:

$$\mathbf{SNR} = \frac{P_c\,\eta\, L_i^2\,(\Gamma_0 - \Gamma_1)^2\, T}{2N_0 W T_s} \tag{2.17}$$

## 2.2 Housekeeping Tasks

### 2.2.1 CFO & CPO Compensation for unsynchronized SDRs

In systems utilizing unsynchronized software-defined radios (SDRs), as seen in subsection 2.1.2, addressing carrier frequency offset (CFO) and carrier phase offset (CPO) is essential for reliable communication. The Gen2 protocol imposes strict time constraints, necessitating swift execution of synchronization and offset compensation algorithms to prevent communication timeouts. More specifically, the reader must start the transmission of the ACK signal, within $T_2$ seconds from the last symbol of the RN16, where $T_2$ takes values within $3\,T_w < T_2 < 20\,T_w$ [2]. Given that $T_w = \frac{1}{BLF}$, as seen in Section 1.2, setting BLF to its lowest possible value, i.e., $BLF = 40$kHz, we get $75\mu s < T_2 < 500\mu s$.

One approach, described in [6], employs a digital phase-locked loop (PLL) to compensate for CFO and CPO, following the methodology outlined in [7, App. C]. CFO tracking focused on sampling during time intervals where only the continuous wave (CW) signal is present, as no additional information from the tag is added. Typically, the latter 4/5 of the CW duration is only used, in order to avoid measurements during the time the PLL oscillates, until it settles in its output value. Tuning PLL parameters optimally is crucial for effective performance tailored to specific applications.

An alternative method explored, involved the use of a periodogram, which offered potentially higher resolution than PLLs but with increased computational overhead, making it impractical for real-time applications. Optimized PLL implementations stroke a better balance between accuracy and runtime efficiency, making them preferable.

In both methods, CPO compensation, which is crucial for tag localization, was not possible. To understand this, perfect CFO and DC compensation is assumed for the system in Eq. 2.10, with the resulting signal following:

$$\begin{aligned} y_{i,step}[k] &= \left(m_{tag,i}\, x_{tag}[k]\right) e^{-j\Delta\phi_R} + n_i[k] \\ &= \left(|m_{tag,i}|\, x_{tag}[k]\right) e^{-j(\phi_i+\Delta\phi_R)} + n_i[k], \end{aligned} \tag{2.18}$$

To estimate the tag location accurately, it was necessary to separate the phase offset $\Delta\phi_R$ from the phase induced by the channel $\phi_i$. One approach would be to assume constant channel for a specific time interval, known as coherence time, and then compute the overall induced phase for each transmitted packet. Each of these estimation would inherently involve the constant value of the channel as well.

### 2.2.2 Channel Estimation for Synchronized SDRs

DC offset removal is common practice in bibliography, in order for synchronization, data detection and channel estimation to be possible. A simple mean over the CW samples is enough for DC estimation. Bellow we present an even simpler way of channel estimation, based on the preamble bits, which does not require estimation of the constant DC term.

Based on Eq. (2.12), we get:

$$y_{pr1,i}[k] = \left(|m_{dc,i}|e^{j\phi_{dc,i}} - |m_{tag,i}|e^{j\phi_i}\right) + n_{pr1,i}[k] \tag{2.19}$$

$$y_{pr0,i}[k] = |m_{dc,i}|\, e^{j\phi_{dc,i}} + n_{pr0,i}[k] \tag{2.20}$$

Subtracting Eq. (2.19) from Eq. (2.20) we get:

$$y_{diff,i}[k] = |m_{tag,i}|e^{j\phi_i} + n_{diff,i}[k] \tag{2.21}$$

By calculating the inverse tangent of the imaginary part divided by the real part of Eq. 2.21, the induced phase of the propagation channel can therefore be estimated:

$$\hat{\phi}_i = \tan^{-1}\left(\frac{\Im(y_{diff,i}[k])}{\Re(y_{diff,i}[k])}\right) \tag{2.22}$$

# Chapter 3

# Multistatic Localization Using Phase-Based Techniques

Work in [1] introduces a state-of-the-art direction-of-arrival (DoA) estimation technique, which is comparable in efficiency with MUSIC algorithm; it is then utilized for multistatic 2D and 3D localization. In this Chapter, a summary of this work is given, in order to lay the foundation going forwards.

## 3.1 Topology

For practicality and without loss of generality, we assume that all antennas are located upon the x-axis, thus simplifying many of the equations that will be presented bellow, as the $y$ coordinates for the antennas are considered zero. The co-linear Tx and Rx antennas and the RFID tag define a plane, in which the RFID tag location is denoted as $\boldsymbol{x}_T = [x_{tag} \quad y_{tag}]^T$, while similarly the Tx and Rx antenna locations in that plane are denoted as $\boldsymbol{x}_{\mathrm{Tx}} = [x_{\mathrm{Tx}} \quad 0]^T$ and $\boldsymbol{x}_{\mathrm{Rx},i} = [x_{\mathrm{Rx},i} \quad 0]^T,\ i \in \{1, 2, \ldots, M\}$ for each one of the M receiving antennas, respectively. All antennas are facing towards the same direction, thus illuminating the same half-plane and they are synchronized using an external function generator.

The induced phase of the propagation channel $h_i$ that was defined in Eq. (2.4) and proven to be calculable for systems with synchronized SDRs in Section 2.2.2, can be written as follows:

$$\phi_{\mathrm{prop},i} \equiv \phi_i = \angle h_i = \frac{2\pi}{\lambda}(d_{\mathrm{CT}} + d_{\mathrm{TR},i}) + \phi_i^{mult}, \tag{3.1}$$

where $d_{\mathrm{CT}} = \|\boldsymbol{x}_{\mathrm{Tx}} - \boldsymbol{x}_{\mathrm{T}}\|_2$ is the Euclidean distance between the Tx antenna and the RFID tag, $d_{\mathrm{TR},i} = \|\boldsymbol{x}_{\mathrm{Rx},i} - \boldsymbol{x}_{\mathrm{T}}\|_2$ the Euclidean distance between the RFID tag and the $i$-th Rx antenna, $\lambda$ stands for the carrier wavelength and $\phi_i^{mult}$ is the phase due to multipath, during transmission of Tx and reception of Rx antenna i. The measured phase at the reader is also affected by various reasons, such as the tag itself, cabling and phase noise, with all these being accumulated into $\phi_i^n$. Thus, the overall phase model at the $i$-th Rx antenna follows:

$$\phi_{\mathrm{out},i} = \frac{2\pi}{\lambda}(d_{\mathrm{CT}} + d_{\mathrm{TR},i}) + \phi_i^n \tag{3.2}$$

From the above equation, it can be observed that any distance $d_{\mathrm{CT}} + d_{\mathrm{TR},i}$ with

$$d_{\mathrm{CT}} + d_{\mathrm{TR},i} = \lambda\frac{\phi}{2\pi} + k_i\lambda, \quad k_i \in \mathbb{N} \tag{3.3}$$

outputs the same wrapped phase measurement $\in [0, 2\pi)$, for a given value of noise. Thus, phase measurements introduce distance ambiguity that must be explicitly addressed.

In order to narrow down phase values to $[0, 2\pi)$, the measured phase from the $i$-th RX follows:

$$\begin{aligned}\phi_{\mathrm{Rx},i} &= \phi_{\mathrm{out},i} \bmod 2\pi \\ &\overset{(*)}{=} \left[\frac{2\pi}{\lambda}(d_{\mathrm{CT}} + d_{\mathrm{TR},i}) \bmod 2\pi + \phi_i^n \bmod 2\pi\right] \bmod 2\pi \\ &= \left[\frac{2\pi}{\lambda}(d_{\mathrm{CT}} + d_{\mathrm{TR},i}) \bmod 2\pi + \tilde{\phi}_i^n\right] \bmod 2\pi,\end{aligned} \tag{3.4}$$

where at $\overset{(*)}{=}$, the property $(\alpha + \beta) \bmod \gamma = [(\alpha \bmod \gamma) + (\beta \bmod \gamma)] \bmod \gamma$ was utilized.

## 3.2 Elliptical Direction-of-Arrival (EllDoA)

Given Eq. (3.3), we can convert a set of $K$ phase values into a single distance value $d_{\mathrm{CT}} + d_{\mathrm{TR},i}$, for $k_i \in \{0, 1, \dots, K\}$. So for each $i \in \{1, 2, \dots, M\}$ of the receiving antennas, a set of $K$ concentric ellipses is defined, where the foci of these ellipses are the locations of the Tx and the $i$-th Rx antenna and the

$K$ ellipses designate the possible coordinates of the tag.

To find the DoA of the tag's signal, the intersection point of two set of ellipses is needed, i.e. two receiving antennas are needed, contrary to MUSIC algorithm, which requires $N_{ant} \geq 2$ receiving antennas. The equations for two sets of ellipses will be equal to:

$$\frac{(x - x_i)^2}{a_i^2[k_i]} + \frac{y^2}{b_i^2[k_i]} = 1, \tag{3.5}$$

where $(x_i, 0)$ is the center for the $i$-th set of ellipses and $a_i$, $b_i$ are the semi-major and semi-minor axes, respectively. By definition, the distance of a point in the ellipse from the two foci is fixed and equal to $2a_i$. Thus, based on Eq. (3.3), we get:

$$2a_i[k_i] = d_{\mathrm{CT}} + d_{\mathrm{TR},i} = \lambda\frac{\phi_{\mathrm{Rx},i}}{2\pi} + k_i\lambda, \tag{3.6}$$

The linear eccentricity $c_i$ is the distance between the center and either of its two foci, or half the distance of the two foci, which is independent of $k_i$, as the foci for a set of $K$ concentric ellipses remain the same.

$$c_i = \|\boldsymbol{x}_{\mathrm{Tx}} - \boldsymbol{x}_{\mathrm{Rx},i}\|_2/2 \tag{3.7}$$

Semi-minor axis $b_i[k_i]$ can be easily calculated from the other two:

$$b_i[k_i] = \sqrt{a_i^2[k_i] - c_i^2} \tag{3.8}$$

The theory above demands for one intersection point to be found, between the two sets of ellipses, in order to calculate the DoA of the tag's signal. However, if we visualize the problem (Fig. 3.1), multiple intersection points will be found, which is logical, given that we have $K$ "pairs" of ellipses. Work in [1], proved that these intersection points indicate the possible tag locations, which in the case of Fig. 3.1a, lie in a single line, introducing distance ambiguity, while in the case of Fig. 3.1b, lie in two separate lines, therefore adding direction ambiguity too. Fortunately, the following theorem insures that, if the distance between the two receiving antennas is smaller than $\lambda/2$, direction ambiguity disappears, ensuring there is a unique DoA of the tag's signal.

***Theorem* 1.** *[1]Assuming one Tx and two Rx antennas located at the same line, if the distance $\delta x$ between the two receiving antennas is $\delta x < \lambda/2$, then a unique direction-of-arrival (DoA) $\theta$ of the tag signal can be found, irrespectively of the Tx antenna location.*

A close form to calculate the DoA, when $\delta x < \lambda/2$, is given in the following lemma:

***Lemma* 1.** *[1]Assume one Tx and two Rx antennas located at the same line, $\phi_A \equiv \phi_{Rx,1}$, $\phi_B \equiv \phi_{Rx,2}$ denote the phase at the receiving antennas, respectively and $\alpha = \|\boldsymbol{x}_T - \boldsymbol{x}_{Rx,1}\|_2 \equiv d_{TR,1}$, $\beta = \|\boldsymbol{x}_T - \boldsymbol{x}_{Rx,2}\|_2 \equiv d_{TR,2}$.*

*If the distance $\delta x$ between two receiving antennas satisfies $\delta x < \lambda/2$, then direction of arrival (DoA) $\theta \in (0, \pi/2)$ of the tag signal is given by:*

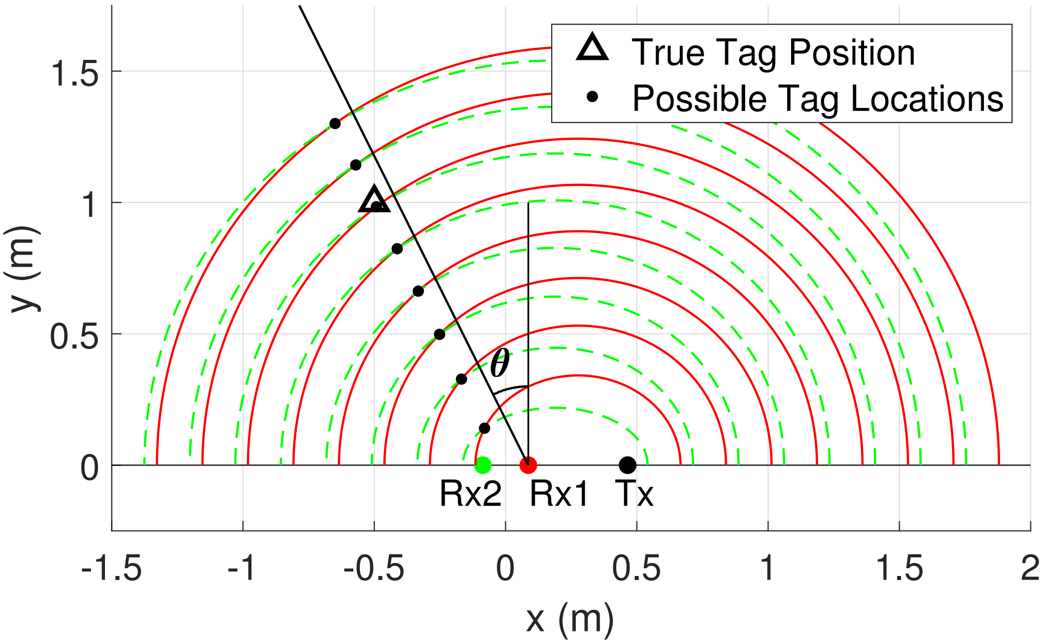
*For $\alpha > \beta$:*

$$\theta \approx \begin{cases} \sin^{-1}\left(\dfrac{\lambda(\phi_A - \phi_B)}{2\pi\,\delta x}\right) & , \quad \textit{if } \phi_A > \phi_B \\ \sin^{-1}\left(-\dfrac{\lambda(\phi_A - \phi_B + 2\pi)}{2\pi\,\delta x}\right) & , \quad \textit{if } \phi_A < \phi_B \end{cases}$$

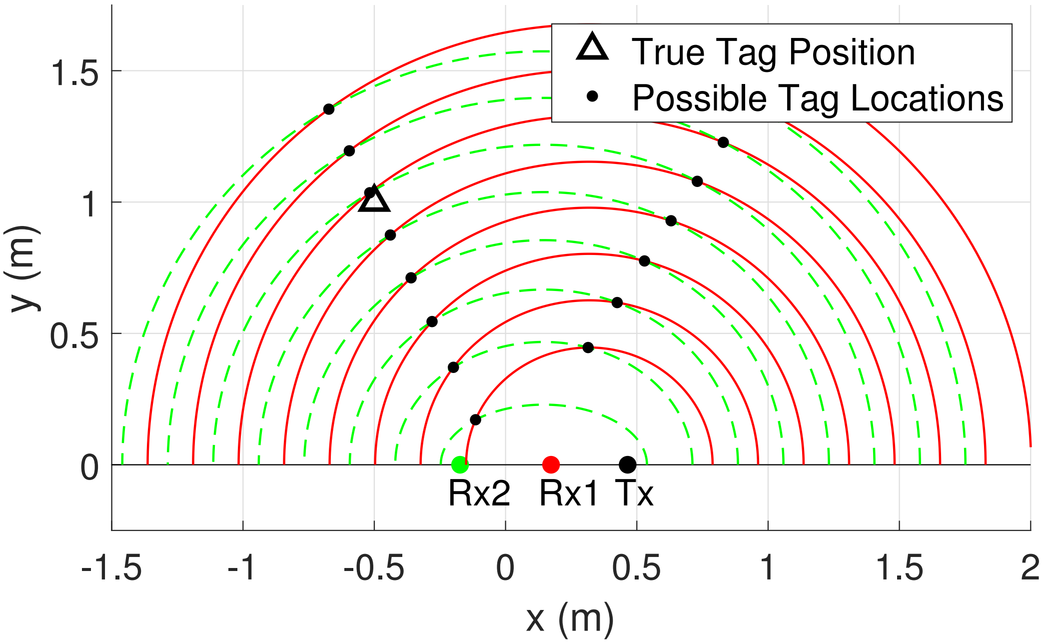
*and for $\alpha < \beta$:*

$$\theta \approx \begin{cases} \sin^{-1}\left(-\dfrac{\lambda(\phi_A - \phi_B - 2\pi)}{2\pi\,\delta x}\right) & , \quad \textit{if } \phi_A > \phi_B \\ \sin^{-1}\left(-\dfrac{\lambda(\phi_A - \phi_B)}{2\pi\,\delta x}\right) & , \quad \textit{if } \phi_A < \phi_B \end{cases}$$

In order for both Theorem 1 and Lemma 1 to stand, far-field propagation is assumed, i.e., tag must be located at least $10\lambda$ away from the antennas. Furthermore, prior knowledge of the tag position, relative to the antennas, is required to select the correct type. Fig. 3.1a also includes a graphic representation of the estimated DoA angle, as calculated above, for a simulated system. Finally, as we are only interested in the half plane that lies in front



(a) $\delta x < \lambda/2$



(b) $\delta x > \lambda/2$

Figure 3.1: EllDoA without (above) and with (below) ambiguity.

of the antennas ($y_{tag} > 0$), any intersection point found behind the antennas is discarded.

Let's now assume an intersection point $\boldsymbol{p} = [p[1]\ p[2]]$ of the two set of ellipses[1], that lies in front of the receiver antennas. Then, the DoA estimate will be:

$$\hat{\theta}_0 = 90^\circ - \tan^{-1}\left(\frac{p[2]}{p[1] - x_s}\right) \cdot \frac{180^\circ}{\pi}, \tag{3.9}$$

where $x_s = (x_{\mathrm{Rx},1} + x_{\mathrm{Rx},2})/2$ is the x-axis coordinate of the midpoint of the 2 Rx antennas. Moreover, given that the intersection points are collinear, a more robust DoA estimate can be obtained by averaging all the intersection points:

$$\hat{\theta}_0 = 90^\circ - \frac{1}{K}\sum_{k=0}^{K-1} \tan^{-1}\left(\frac{p_k[2]}{p_k[1] - x_s}\right) \cdot \frac{180^\circ}{\pi}, \tag{3.10}$$

## 3.3 Multistatic 2D Localization

As seen in Fig. 3.1a, 2 Rx antennas are only capable of estimating the direction in which the tag is located. To identify the tag's distance from the antennas and therefore its true location, a third Rx antenna is needed in order to create a third set of ellipses and afterwards search for the point where all three sets of ellipses intersect. In Fig. 3.2 the 2D localization scheme is displayed, with all the necessary information to understand the system topology and the two estimation methods implemented, which will be explained below. For better readability, only a portion of the ellipses have been plotted. In a real system, enough ellipses, to fill the area of interest, must be computed. However, as the area grows larger, so will the computational time, which may lead to implications, concerning the real-time tracking of the tag itself. A viable alternative would be to use the previous estimation of the tag position and "center" the ellipses around the tag, thus computing only a handful of sets, though this would require constantly robust measurements.

---

[1] The intersection points were calculated using `InterX` function created for MATLAB: https://www.mathworks.com/matlabcentral/fileexchange/22441-curve-intersections

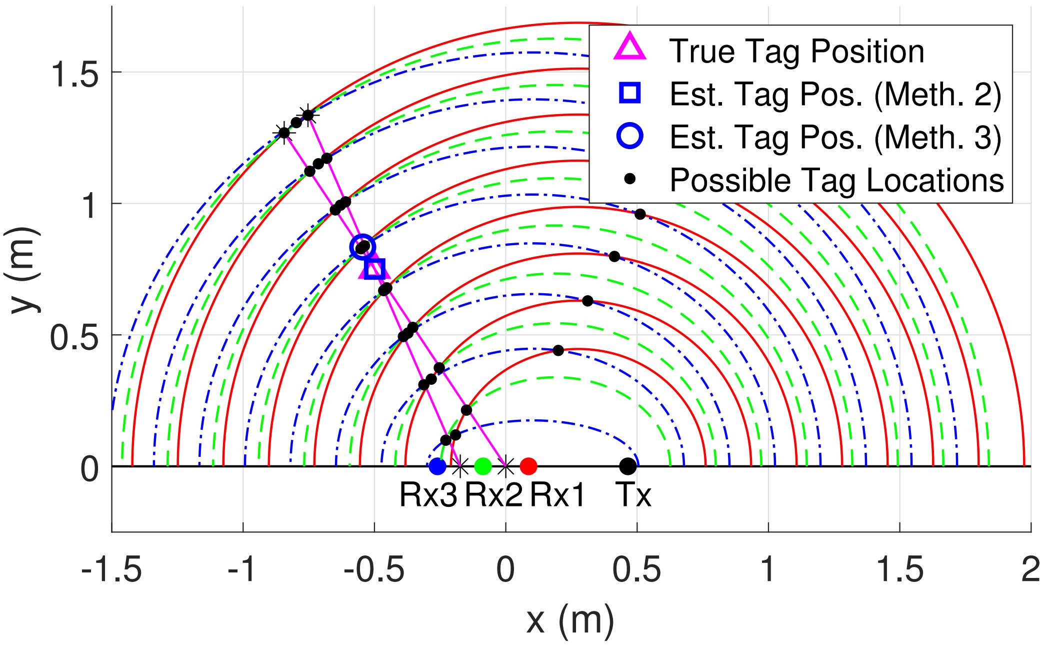


Figure 3.2: Multistatic 2D localization of an RFID tag.

There are 3 ways to find out the intersection points: 1) the intersection method of two conic sections, as per [8], which is not implemented bellow, 2) calculating 2 DoA estimates from pairs Rx1 - Rx2 and Rx2 - Rx3, drawing the lines passing from the medians of these two antenna pairs and finding the intersection points of the two lines, 3) by equating the left-hand sides of Eq. (3.5) in pairs of $i = \{1, 2, 3\}$ and finding the common intersection point of all 3 pairs (the obtained equations are simple, due to the fact that all ellipses foci are in the same line).

## Method 2 Implementation

Firstly, we take the two midpoints between the pair of antennas Rx1 - Rx2 and Rx2 - Rx3.

$$Rx_{midp1} = \frac{x_{Rx,1} + x_{Rx,2}}{2}, \quad Rx_{midp2} = \frac{x_{Rx,2} + x_{Rx,3}}{2}$$

We draw the two lines that start from the aforementioned points and end at the outer most intersection points that can be found for the pairs, say $\boldsymbol{p}_{12}$

and $\boldsymbol{p}_{23}$. The intersection point of the two lines gives us the estimated tag coordinates.

## Method 2.5 Implementation

An almost identical, but computationally less expensive way to find the intersection point, is to calculate the two DoA angles based on Th. 1, draw the two lines representing DoA and calculate their intersection point. This way, the ellipses and their intersection points are not computed, making the algorithm considerably faster. However, as it was stated previously, DoA measurements require partial knowledge of the tag position, i.e., whether $d_{\mathrm{TR},i}$ or $d_{\mathrm{TR},j}$ is greater. One approach to address this challenge would be to compute DoA for both types and select the value that has the smaller difference compared to the last estimation. This way continuity can be achieved, even when the tag passes through the perpendicular bisector of the two Rx antennas in question. Alternatively, a more effective and robust way to eradicate the issue, would be to place the antenna pairs on the edges of the area of interest, guaranteeing that only one condition will always be true. This estimation method is illustrated in Fig. 5.2, where the reader can also visualize how the ideal setup would be, in order to have no need for prior knowledge of the tag position.

## Method 3 Implementation

In the third method, we search for the three intersection points (for the 3 pairs of ellipses) that overlap the best. In order to achieve this, Eq. (3.5) is used, transferring all the terms on the left and declaring a new variable:

$$dps = \frac{(x - x_i)^2}{a_i^2[k_i]} + \frac{y^2}{b_i^2[k_i]} - 1$$

Replacing the coordinates of each intersection point to all the ellipse equations, a three dimensional vector is created, showing how well each intersection point fit the ellipses[2]. Then, for each Rx antenna, we find which one of the K ellipses does each intersection point best satisfy. We do this by selecting the minimum absolute value of dps out of the K. We save, for each intersection point, both the value of *dps* and the index $k \in \{0, 1, \ldots, K\}$ in a new matrix $dps2$. Finally, we add the 3 values computed for each intersection point (one for each Rx antenna) and the one with the minimum overall value is selected as the approximate tag location. Given the complexity of the algorithm, the respective MATLAB code is given below:

---

[2] Given that each intersection point is always part of two ellipses, it will have two zero values.

```
% Variables explanation:
% p: intersection points
% K: sets of ellipes, Nant: number of antennas
% a, b, midp: ellipse variables
dps = zeros(length(p),K,Nant);
for i=1:length(p)
    for k=1:K
        for j=1:Nant
            dps(i,k,j) = (p(1,i) - midp(j,1)).^2/a(j,k)^2 + p(2,i).^2/b(j,k)^2 - 1;
        end
    end
end

dps2 = zeros(length(p),2,Nant);
for i=1:length(p)
    [v1,idx1] = min(abs(dps(i,:,1)));
    [v2,idx2] = min(abs(dps(i,:,2)));
    [v3,idx3] = min(abs(dps(i,:,3)));
    dps2(i,1,1) = v1;
    dps2(i,2,1) = idx1;
    dps2(i,1,2) = v2;
    dps2(i,2,2) = idx2;
    dps2(i,1,3) = v3;
    dps2(i,2,3) = idx3;
end

sdsp2 = sum(dps2(:,1,:),3);

[~,idx] = min(sdsp2);
est_tag_coord = p(:,idx);
```

# Chapter 4

# Multistatic Localization Using Neural Networks

Prior art in [9] and [10] proved how neural networks (NNs) can improve the speed and accuracy of RFID tag localization, in monostatic systems, based mainly on phase measurements, without knowledge of the tag's initial position. Moreover [10], proved the inclusion of ambiguous data (wrapped phase measurements) to be advantageous for the performance of the NNs. In this Chapter, an even better performing and more efficient method of multistatic 2D tag localization, with the help of a Deep Feed Forward Neural Network (DFFNN), will be introduced, based on the theory of Chapter 3.

## 4.1 Introduction To Neural Networks

Neural Networks, inspired by the intricate architecture of the human brain, have demonstrated exceptional prowess in learning complex and non-linear relations between input data and desired output, through the processing of vast amounts of data. In situations where traditional methods reach their inherent limits or encounter challenges, NNs possess the capability to learn patterns and discover relations, that no human-designed algorithms can produce.

The structure of a typical NN usually includes an input layer, one or more hidden layers and an output layer, each containing multiple—interconnected between the layers—nodes. During the training process, input data are fed into the network and information is propagated through the hidden layers, where it is transformed through the use of biases, weighted connections and activation functions. These functions introduce non-linearities, allowing the network to capture complex relationships within the data. The received signal on a random node can be denoted as:

$$z = \sum_{i=1}^{n} x_i w_i + b, \tag{4.1}$$

where $x_i$ is the information received from the $i$-th neuron that the node receives, $w_i$ is the weight of this path and $b$ is the bias that is specific for the whole node. The signal will then be passed through the activation function (see 4.3.2 subsection) and sent to the next layer:

$$y = f(z) \tag{4.2}$$

The process explained above, is displayed in the Figure bellow:

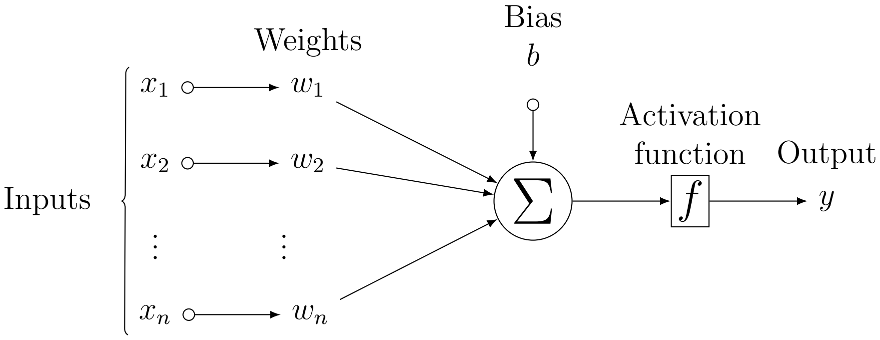


Figure 4.1: Artificial Neuron.

The output layer produces the final result, i.e., the prediction of the output. During the training phase, the network tries to adjust its parameters, in order to minimize the difference between its predictions and the actual outcomes, guided by a loss function. Besides of converging to the target output, the process of iteratively updating weights and biases, also known as Back-Propagation, is also responsible for enhancing the network's ability to generalize from the training data. Hence, a validation dataset is needed, to check after each iteration whether the network is becoming too closely related to the training set, failing to correctly predict new data[1]. Last but

---

[1] This issue is also known as overfitting.

not least, the test dataset is used to evaluate the overall ability of the model. Validation and test datasets ought to provide an unbiased evaluation of the model and therefore must never be used to train the neural network.

## 4.2 Variable Analysis & Pattern Recognition

To begin, we will find which variables should be used to feed the NN. In Fig. 4.2 we observe how three different measurements behave in the span of 100 iterations, where the location of the tag is fixed at a random place. We see that, due to the added phase noise (fully correlated with $\phi^n \sim \mathcal{N}(0, 4\pi^2)$), the phase measurements received at the 1st Rx antenna are completely random (as it will be at any of the three Rx antennas), while the phase difference between the 1st and the 2nd Rx antennas and the DoA angle are constant. Note that $\phi_{\mathrm{Rx},i}$ will take values in $(0, 2\pi)$ and as a result $\Delta\phi_{ij} \in (-2\pi, 2\pi)$, but to simplify the model we take the modulo $2\pi$, so it will also report in $(0, 2\pi)$. Finally, $\theta_{ij} \in (0, \pi)$.

The manners in which the phase difference and the DoA angle fluctuate in the 2-dimensional space are presented in Fig. 4.4 and Fig. 4.3. Overall, both values seem fitting for feature inputs; the only observations worth noting are the phase discontinuity, which is apparent in Fig. 4.3, and the constant

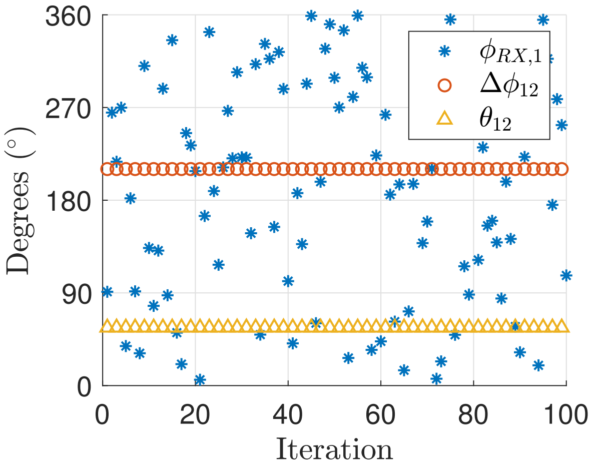


Figure 4.2: Phase based Measurements for fixed tag location.

zero value on the—perpendicular to the $x$-axis—line that passes through the midpoint of the two antennas.

## 4.3 NN Construction

A Regression Multi-Layer Perceptron (MLP) with 6 hidden layers was implemented, as per [9], in order to estimate the coordinates of the tag. Classification would not be efficient for this kind of work, as an extremely dense grid—consisting of thousands of classes—would be needed, in order to have good accuracy. The "pool" of feature inputs included all the phase-based measurements available, i.e., $\Delta\phi_{12}, \Delta\phi_{23}, \theta_{12}, \theta_{23}$, from which we either selected all four, the former two or the latter two, in order to compare the importance of each metric for the algorithm (see Section 5.1). The output of the system are the estimated tag coordinates, denoted as $\hat{x}_{tag}$ and $\hat{y}_{tag}$, calculated in centimeters[2]. A graphic representation of the neural network structure is presented in Fig 4.5, where the number of nodes for each layers is also displayed.

Custom loss functions were created, using the Euclidean distance to compute the error:

$$\boldsymbol{e} = ||\boldsymbol{x}_{tag} - \hat{\boldsymbol{x}}_{tag}||_2 = \sqrt{(x_{tag} - \hat{x}_{tag})^2 + (y_{tag} - \hat{y}_{tag})^2} \tag{4.3}$$

Mean absolute error ($\mathbb{E}[|\boldsymbol{e}|]$) and root mean squared error[3] ($\sqrt{\mathbb{E}[|\boldsymbol{e}|^2]}$) were both evaluated in separate NNs, in order to achieve the best efficiency for each loss function.[4] However, the differences between results were insignificant, thus, all results were extracted using the NNs that had MAE as their loss function. MAE and RMSE equations are presented in Eqs. 4.4 & 4.5,

---

[2] It was observed that converting the outputs to centimeters yielded slightly better results, compared to meters.

[3] Technically the mean squared error was used as the loss function and the root of the value was calculated afterwards.

[4] As clarified in Section 4.1 the loss function plays an active role in the training of the NN

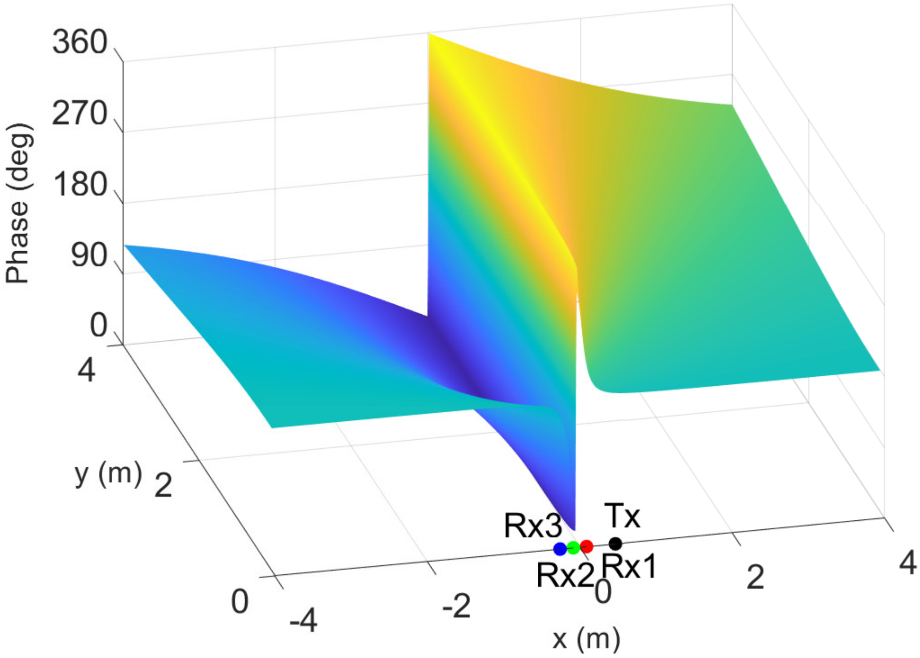


Figure 4.3: Phase Difference $\Delta\phi_{12}$ distribution.

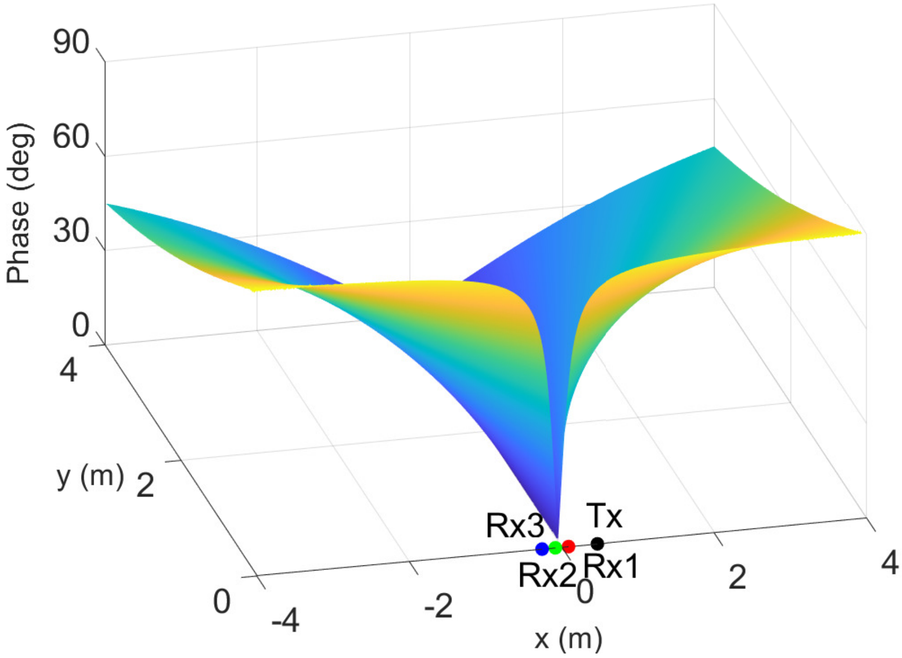


Figure 4.4: DoA angle $\theta_{12}$ distribution.

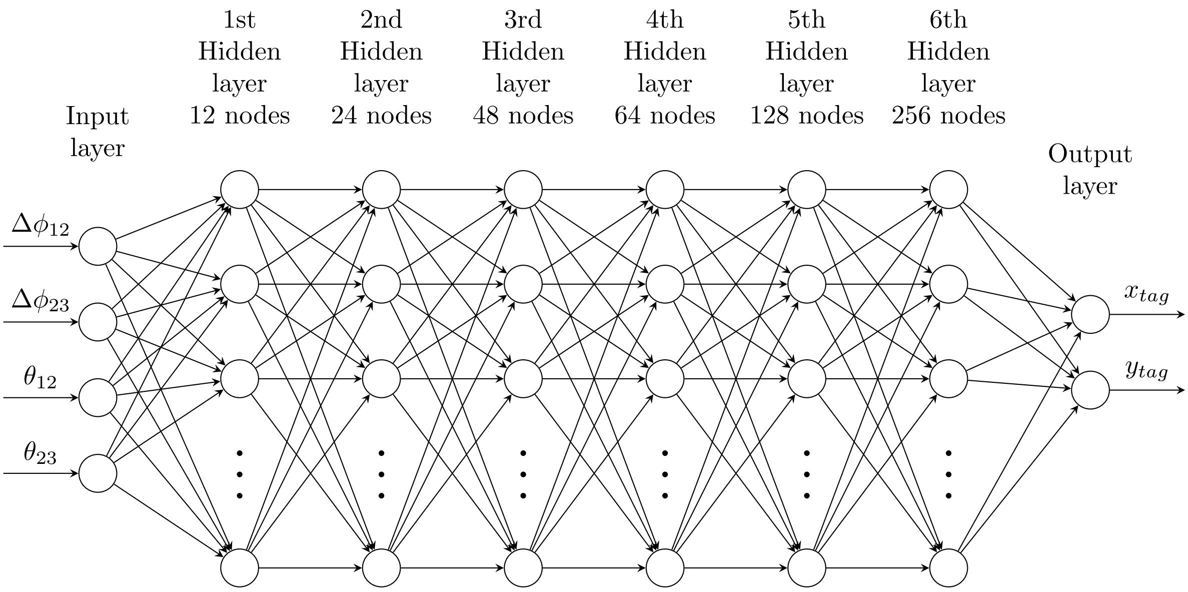


Figure 4.5: MLP Neural Network Structure.

respectively:

$$\mathbb{E}[|\boldsymbol{e}|] = \frac{1}{n}\sum_{i=1}^{n} ||\boldsymbol{x}_{tag,i} - \hat{\boldsymbol{x}}_{tag,i}||_2, \tag{4.4}$$

$$\sqrt{\mathbb{E}[|\boldsymbol{e}|^2]} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} ||\boldsymbol{x}_{tag,i} - \hat{\boldsymbol{x}}_{tag,i}||_2^2}, \tag{4.5}$$

where $n$ denotes the number of samples.

Finally, the trained NN can be mathematically expressed as:

$$f_{NN}(\Delta\phi_{12}, \Delta\phi_{23}, \theta_{12}, \theta_{23}) = \hat{\boldsymbol{x}}_{tag} \tag{4.6}$$

### 4.3.1 Data Scaling

Before feeding the network with the data, it is common practice to pre-process the input data first[5]. There are plenty of options here, which are divided in

---

[5] In regression problems, such as this, usually only the input data require pre-processing. Output data pre-processing is recommended only in multi-output cases, with different variations on the outputs and in classification problems.

two main categories; Normalization and Standardization. In normalization (e.g. `MinMaxScaler, MaxAbsScaler`), feature values are usually scaled down to $[0, 1]$ or $[-1, 1]$, while preserving the relationship between the minimum and maximum values of each feature. `MinMaxScaler` always scales down the inputs to fit $[0, 1]$, while `MaxAbsScaler` will scale down values to $[0, 1]$, if only positive values are present, $[-1, 0]$ if only negative are present and $[-1, 1]$ when both are present.[6]

In Standardization (e.g. `StandardScaler, RobustScaler`) the values after processing will have properties of a standard normal distribution with zero mean and variance equal to unit, ensuring faster convergence and algorithmic stability. In cases where all the inputs are positive (e.g. normalization to fit in $[0, 1]$), a zigzagging effect may be observed, due to the fact that weight updates will all have the same sign in each iteration, i.e., they will either all increase, or all decrease. As a result, the network will converge in a very slow manner, as seen in Figure 4.6, or it will be completely unable to converge [11]. Furthermore, for cases with many outliers, there is `RobustScaler`, which ignores extreme values.

It is of utmost importance to first divide the data and then create the transformation function, which performs the scaling, based only on the train subset. This way, no statistical information is passed to the training process

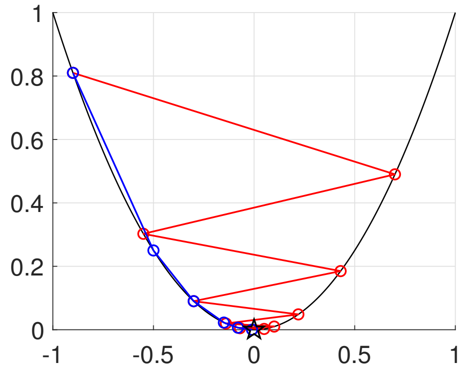


Figure 4.6: Zigzagging Effect on a Loss Function.

---

[6] For this specific problem, where the inputs are all positive (see Fig. 4.2), there will be no difference between `MinMaxScaler` and `MaxAbsScaler`.

about the validation and test datasets.

In the pursuit of refining the performance of the model, two further considerations come to the forefront. Firstly, when the inputs can be separated by their importance, it is common practice to scale down the less significant ones, thus focusing on features with more substantial contributions and managing the network's resources more efficiently. Secondly, it is desirable to have uncorrelated input variables, when possible. This condition minimizes multicollinearity, meaning that a single parameter change during backpropagation, will not affect the rest of the parameters, thus improving the stability of the network. In our system such a constraint cannot be met, as all of our inputs are products of the phase measurements and the position of the antennas. Finally, the preprocessing that is selected sometimes also determines which activation functions are preferable, as it will be seen in Chapter 5.

### 4.3.2 Activation Function

A big role in the efficiency of the network plays the activation function, which determines the extent in which a neuron is activated/passed in the next node. The three most commonly used activation functions for the hidden layers, which are all non-linear, are the rectified linear unit (`relu`) [12], defined as:

$$relu(x) = \begin{cases} 0 & , \quad x < 0 \\ x & , \quad x \geq 0 \end{cases} \tag{4.7}$$

and the closely related standard logistic function (`sigmoid`) and hyperbolic tangent (`tanh`), which both have a characteristic "S"-shaped curve and their definitions follow:

$$sigmoid(x) = \frac{1}{(1 + e^{-x})} \tag{4.8}$$

and

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{4.9}$$

respectively. However, when it comes to the output layer, the linear $f(x) = x$ function is used for regressors, as the output values are not bounded, nor normalized.

In Fig. 4.7 we can see that tanh is centered around the origin, whereas sigmoid is centered around 0.5 and, along with ReLU, will have only positive values. This distinction makes the hyperbolic tangent preferable, for the same reasons presented at the previous subsection [11]. Another reason to select tanh over sigmoid is its faster rate of change, as seen in Fig. 4.8, which in turn means faster converging. A common phenomenon, when using sigmoid and tanh activation functions, is the vanishing gradient problem, which occurs when your input is on either extreme of the scale, where these functions are relatively flat and their gradient is near zero. During backpropagation, this will cause the weight updates to have extremely small values, slowing down or bringing to a halt the learning process. ReLU on the other hand, has a step function for gradient, ensuring that there will be no vanishing gradient effect, therefore accelerating the convergence. On top of that, it is computationally cheaper, as no exponentials need to be calculated. However, dead neuron units occur frequently on ReLU, because the derivative of the function for negative inputs is zero, meaning that the weights will not be updated.

Given that the problems regarding ReLU concern the part where $x < 0$, there are many variations that try to introduce a non-zero gradient for these values [13]. Leaky ReLU (LReLU) [14] and parametric ReLU (PReLU) [15], both have linear equations for the right half-plane, with $lrelu(x) = 0.01x, \;\; x < 0$ and PReLU being the generalized form, with $prelu(x) =$

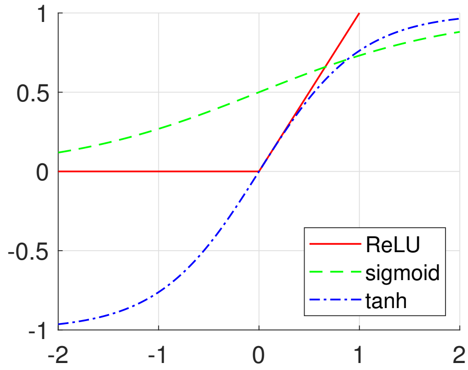


Figure 4.7: Activation Functions.

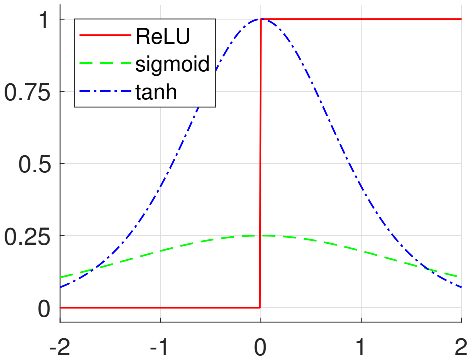


Figure 4.8: Derivatives.

$ax,\ x < 0$. Finally there is exponential ReLU (ELU) [16], with $elu(x) = \alpha(e^x - 1),\ x < 0$. Results in bibliography vary, regarding these methods, with many reporting the original ReLU as the best performing [13], concluding, once again, that every selection is problem based and that sometimes the simplest methods are the most efficient. In Chapter 5 experiments will be conducted to compare these functions.

### 4.3.3 Optimizer

An optimization algorithm is a method designed to minimize a loss function by updating the weights and biases of the model in response to the loss function output. In an N-dimensional space, the objective is to find the values of $x_i$, where $i \in \{0, 1, \ldots, N-1\}$, that minimize the value of $\mathcal{L}(x_1, x_2, \ldots, x_{N-1})$[7]. It is essential to note that in the imaginary N-D space, there may exist local minima, flat regions (plateaus), and other features that contribute to the complexity of the optimization problem. The two-dimensional equivalent of the problem is displayed in Fig. 4.6, while for the sake of completeness a three-dimensional model is also displayed in Fig. 4.9.

The two main optimizers used in the literature are RMSprop and Adam. To comprehend these optimizers, a brief overview of **Gradient Descent** and some of its variations is necessary. Similar to the converging line depicted in Fig. 4.6, gradient descent makes larger steps when the slope is steeper (indicating a larger first derivative). As we approach the base of the slope, the gradient gradually diminishes until reaching the minimum. Weight update is given by:

$$w_{t+1} = w_t - \eta \nabla \mathcal{L}_t(w_t), \tag{4.10}$$

where $\eta$ is the learning rate, and $\nabla \mathcal{L}_t(w_t) = \frac{\partial \mathcal{L}_t}{\partial w_t}$ is the partial derivative of the loss function $\mathcal{L}$ with respect to a weight $w$ at a particular round of the algorithm, denoted as $t$. Relying solely on the first derivative implies that this optimizer may struggle to overcome local minima and might become trapped in flat regions.

---

[7] The admission has been made that an N-dimensional space will have (N-1) variables, so that it can be displayed in N dimensions.

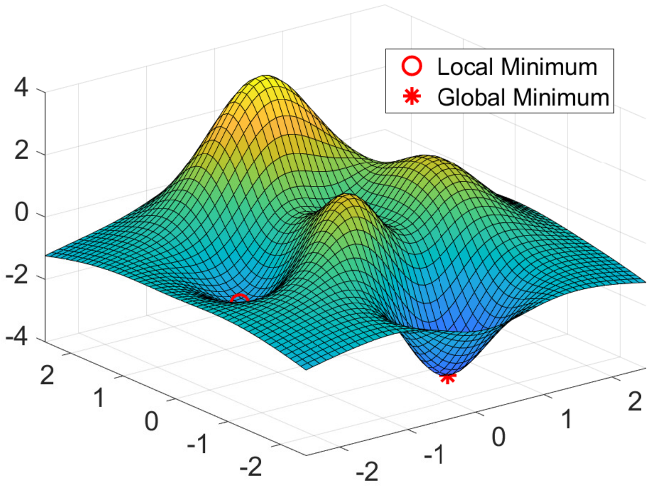


Figure 4.9: Loss Function for two variables.

The **Momentum** optimizer [17] builds upon Gradient Descent and incorporates memory into the algorithm in the form of momentum, effectively keeping track of past gradients. Drawing an analogy to a ball rolling inside a bowl, accelerating and decelerating, momentum applies this principle:

$$v_t = \mu v_{t-1} - \eta \nabla \mathcal{L}_t(w_t) \tag{4.11}$$

$$w_{t+1} = w_t + v_t, \tag{4.12}$$

where $\mu \in [0, 1]$ represents the momentum coefficient. This coefficient influences the impact of past gradients on the current update. The Momentum optimizer enables the algorithm to overcome local minima and navigate small plateaus, ultimately reducing convergence time.

The **Ada**ptive **Grad**ient Algorithm (**Adagrad**) [18] also incorporates information from past gradients, but it utilizes them in a distinct manner. Adagrad aims to mitigate overfitting in specific parameters by dynamically adjusting the learning rate. It executes smaller updates for parameters associated with frequently occurring features and larger updates for those linked to infrequently occurring features. The mathematical representation of Adagrad is given by:

$$\gamma_t = \sum_{i=1}^{t} \nabla^2 \mathcal{L}_t(w_t) \tag{4.13}$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\gamma_t}} \nabla \mathcal{L}_t(w_t) \tag{4.14}$$

Unlike optimizers with momentum, Adagrad lacks a momentum component, and due to the accumulation of squared gradients in the denominator, it tends to be relatively slow.

Moving on to **R**oot **M**ean **S**quare **Prop**agation (**RMSProp**), firstly proposed by Geoff Hinton, this optimizer maintains a moving average of the squared gradients for each weight and adjusts the weight updates accordingly. This can be compared with Adagrad, with the inclusion of a decay rate $\mu$, which helps the algorithm to "forget" past gradients, thereby enhancing the learning rate.

$$\gamma_t = \mu\, \gamma_{t-1} + (1-\mu) \nabla^2 \mathcal{L}_t(w_t), \tag{4.15}$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\gamma_t}} \nabla \mathcal{L}_t(w_t), \tag{4.16}$$

where $\mu \in [0, 1]$ is the moving average parameter, typically set to the default value of 0.9.

**Ada**ptive **M**oment **Estimation** (**Adam**) [19] combines the strengths of RMSprop and Momentum, leveraging Momentum's speed and RMSprop's adaptability to all parameters. The mathematical representation is as follows:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) \nabla \mathcal{L}_t(w_t), \tag{4.17}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) (\nabla \mathcal{L}_t(w_t))^2, \tag{4.18}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \tag{4.19}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}, \tag{4.20}$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t, \tag{4.21}$$

where $\beta_1$ and $\beta_2$ are the momentum and scaling parameters, and $\epsilon$ is a small constant to prevent division by zero. Adam consistently performs as well as, or even better than, RMSprop, all while being faster.

## 4.4 Introducing Ambiguity in the Phase Measurements

Work in [10] proved that the combination of a typical antenna setup, which produces unambiguous phase measurements, with a setup where the distance between antennas surpasses the $\lambda/2$ limit, thus introducing ambiguity, yields better resulting NNs. Received signal strength indicator (RSSI) measurements were also taken from N reference tags, giving the neural network more robust information to train upon.

Building on this insight, a novel setup was devised, featuring 4 Rx antennas arranged in pairs of two, to capture both ambiguous and non-ambiguous data. This configuration yields a total of 8 feature inputs: four related to neighboring antennas without ambiguity ($\Delta\phi_{12}, \Delta\phi_{34}, \theta_{12}, \theta_{34}$), and four related to non-neighboring antennas with ambiguity ($\Delta\phi_{13}, \Delta\phi_{14}, \Delta\phi_{23}, \Delta\phi_{24}$). The topology, along with the corresponding ellipses, is depicted in Fig. 4.10. The intersection of the two lines representing the Direction of Arrival (DoA) of the tag's signal, as elaborated in Section 3.3, is also illustrated, providing a reliable estimate of the tag's location.

In this configuration, the phase difference between two antennas from different pairs ($\delta x > \lambda/2$) will be wrapped, i.e., the same phase value may correspond to multiple points in the area. Fig. 4.11 visually represents the phase difference from the Rx2-Rx3 pair, clearly demonstrating that the phase is "folded" five times. If the outermost antennas were chosen (Rx1-Rx4 pair), even more folds would be evident, while further increasing the distance between antennas would result in additional wrapping. Therefore, in setups where pairs are positioned farther from each other, an increased number of folds is expected.

In broad terms, wrapping certain phase difference measurements could

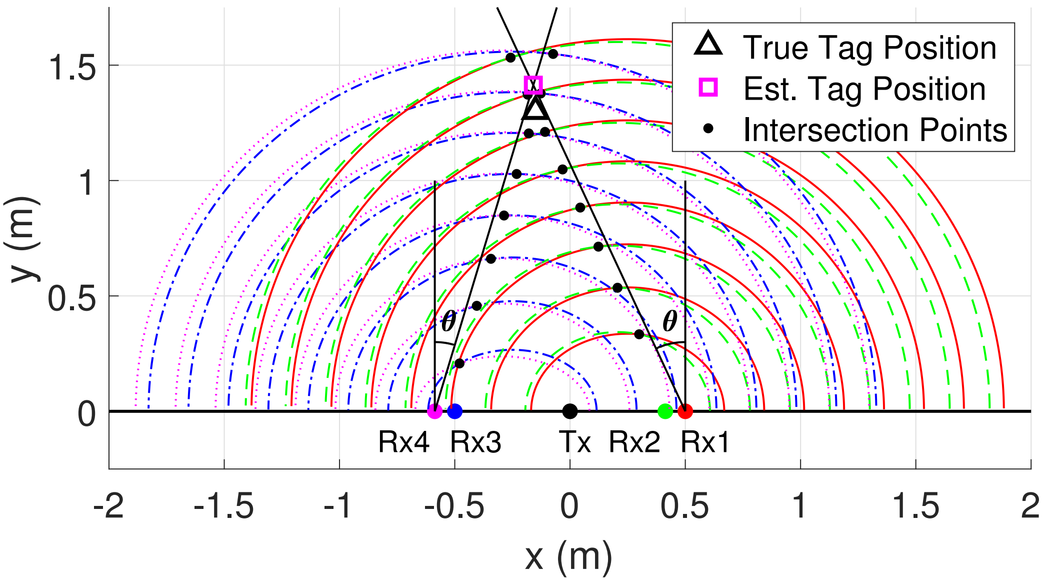


Figure 4.10: Multistatic 2D localization of an RFID tag, with ambiguity.

prove beneficial, as it amplifies the rate of change of the values, in relation to distance. This augmentation enables the NN to extract more information, since the difference of these values for neighboring points will be greater, while the non-ambiguous data provide a solid ground. The primary question in this context revolves around determining the optimal separation between the two pairs to maximize useful information, without disorienting the NN with excessive ambiguity. The answer, as all things in NNs, depends heavily on the specific problem that is being solved. Given that this work utilizes simulated data to train the NN, with samples for random tag positions within the area of interest, increasing the distance can be done with a lower risk of confusing the NN. As it will be seen in Section 5.1, simulation results of a setup with more than 30 folds in the $\Delta\phi_{23}$ measurements performed better than the aforementioned one, but not for the reason mentioned above. On the contrary, it will be proven that ambiguous data do not further reinforce the NNs efficiency and that the amplified differences in the non-ambiguous inputs, accompanied by the increased distance between the pairs, is the difference maker. Furthermore, in an experimental environment, where data samples may be more unreliable and with many outliers, a high level of ambiguity could prove unfavorable.

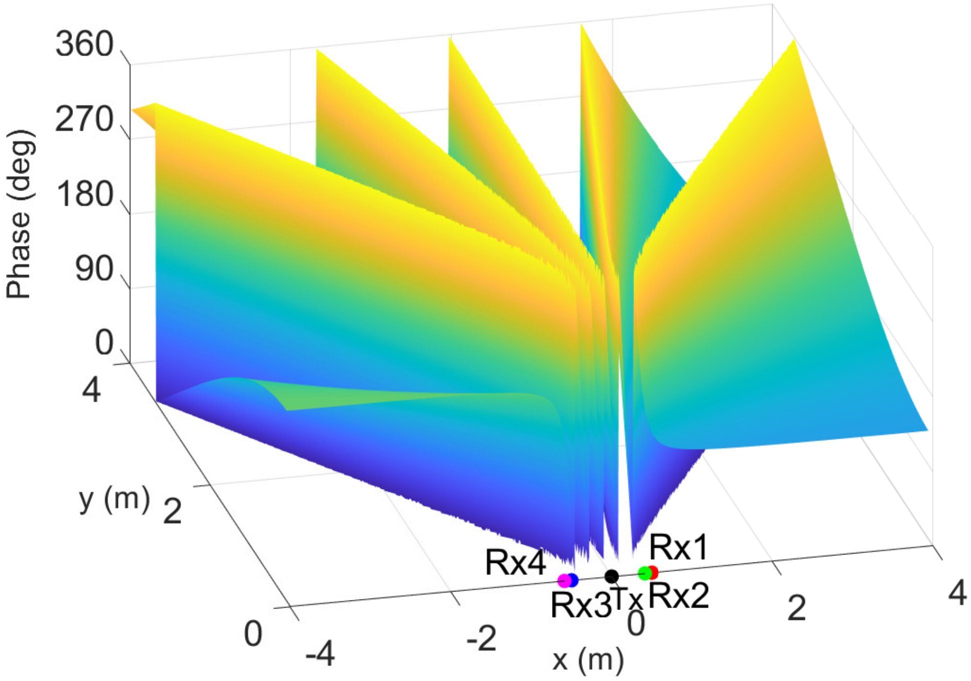


Figure 4.11: Phase Difference $\Delta\phi_{23}$ distribution, for non-neighboring antennas.

# Chapter 5

# Numerical Results

In this Chapter, MATLAB simulations were employed to gather measurements from various multistatic setups, which were then utilized to train neural networks (NNs) and compare their performance with the traditional EllDoA method. The Google Colaboratory (Colab) platform was utilized for building and training the NNs, providing three separate sessions with access to cloud computing resources at the time of writing. The implementation was carried out in Python, using TensorFlow (v2.15.0) and the Keras libraries.

For all the neural networks trained, learning rate was set to $10^{-4}$, batch size to 64, training epochs to 50, the database was split 85/15% train/validation and the custom MAE function, seen in Eq. 4.4, was used as the loss function. In order to ensure comparable results between different executions, the `tf.keras.utils.set_random_seed` utility was also used, which creates a fully deterministic program by setting all the random seeds of the program at the same value.[1]. Moreover, the `tf.keras.backend.set_floatx` option was evaluated, while trying to find the best compromise between training time and efficiency. By setting the float data type to be 64 bit long, an improvement of—at best— some millimeters was noted, in the expense of doubled training time. Thus, the more common 32 bit size was selected.

All experiments were conducted using the multistatic setup with synchronized Software-Defined Radios (SDRs) as described in Section 2.1.3. Thus, the phase noise ($\phi_i^n$) was considered fully correlated Gaussian and was modeled as such, with $\phi_i^n \equiv \phi_j^n \, \forall \, i, j \in 1, 2, \ldots, N_{ant}$. Rician fading was assumed with scattering components modeled by $k_{\mathrm{CT}} = 15$, $k_{\mathrm{CR},i} = 15$, $k_{\mathrm{TR},i} = 10$, and $\sigma_q^2 = \mathbb{E}\left[|h_q|^2\right] = 1$ for all paths. For the *large-scale* channel path-loss,

---

[1] A random seed is an initial input for a pseudorandom number generator, determining the sequence of generated numbers, thus enabling reproducibility.

reference distance $d_0 = 1m$ and path-loss exponents $\upsilon_{\mathrm{CT}} = \upsilon_{\mathrm{CR},i} = \upsilon_{\mathrm{TR},i} = 3$ were considered. Power of the carrier wave was $P_c = 25\mathrm{dBm}\,(0.316\mathrm{W})$, carrier frequency was set at $f_c = 865\mathrm{MHz}$, load-independent structural mode parameter equal to $A_s = 0.6047 + 1j * 0.5042$ [5] and the tag's scattering efficiency was $\eta = 0.1$. These parameter values were chosen to represent a realistic system with considerable losses throughout communication. The SNR, in relation to the distance of the tag from the Tx antenna, is presented in Fig. 5.1.

Three different topologies were created, as illustrated in Fig. 3.2, Fig. 4.10, and Fig. 5.2. In each case, the antennas were positioned along the x-axis. For Fig. 3.2 it stands that $x_{\mathrm{Tx}} = 0.465$, $x_{\mathrm{Rx},1} = d/2$, $x_{\mathrm{Rx},2} = -d/2$ and $x_{\mathrm{Rx},3} = -3d/2$, where $d = \lambda/2$ and $\lambda$ is the carrier wavelength. For Fig. 4.10: $x_{\mathrm{Tx}} = 0$, $x_{\mathrm{Rx},1} = 0.5$, $x_{\mathrm{Rx},2} = 0.5 - d/2$, $x_{\mathrm{Rx},3} = -0.5$ and $x_{\mathrm{Rx},4} = -0.5 - d/2$. Finally for Fig. 5.2: $x_{\mathrm{Tx}} = 0.465$, $x_{\mathrm{Rx},1} = 3$, $x_{\mathrm{Rx},2} = 3 - d/2$, $x_{\mathrm{Rx},3} = -3$ and $x_{\mathrm{Rx},4} = -3 - d/2$. An 8x4 m$^2$ area was assumed where the tag could be located, with $x_{tag} \in [-4, 4]$ and $y_{tag} \in [0, 4]$.

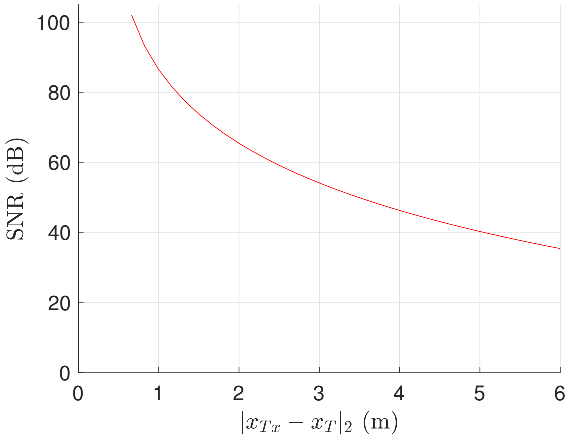


Figure 5.1: SNR relative to the distance.

## 5.1 Towards The Optimal Neural Network

In Chapter 4, the construction of the neural network was thoroughly explained and analyzed. While some variable choices were straightforward, others were narrowed down based on existing bibliography. This Section evaluates, compares and explains various setups. The areas where the preferred choice was apparent were the data scaling and the selection of optimizer. In the former, `RobustScaler` is always employed, to ensure an equal distribution of negative and positive inputs, while in the latter, `Adam` is undeniably the preferred choice. Although experiments were conducted to reach these conclusions, they were deemed less critical for inclusion.

For all comparisons, a pair of visuals will be provided: a Figure displaying the loss during the training process and a Table illustrating the test results, both using mean absolute error measurements. The mean over three separate NNs is calculated by training the network with different random states, indicating different splits of the train/validation data. This is done in order to ensure that the results will be as reliable as possible. Finally, in order to compare ambiguous with non-ambiguous data, a total of three databases were created, for the three systems in Fig. 3.2, Fig. 5.2 and Fig. 4.10. Each database consisted of almost[2] 2 million random samples, where 850,000 samples were used for training, 150,000 for validation and the rest for testing. Throughout the following subsections, the three setups will be compared between them.

### 5.1.1 Activation Functions Comparison

Starting with the activation functions, the two most common in `relu` & `tanh` were compared, as well as two variations of them in `elu`[3] & `sigmoid`, respectively. As demonstrated in Fig. 5.3 and Table 5.1, hyperbolic tangent, prevailed in all the experiments as the favorable choice, exhibiting similar convergence time to `relu`, while `elu` converged faster and was slightly better

---

[2] Some of the samples reported complex DoA values due to the noise in the system and, thus, were not used.

[3] Exponential relu was selected as the best performing alternation of relu.

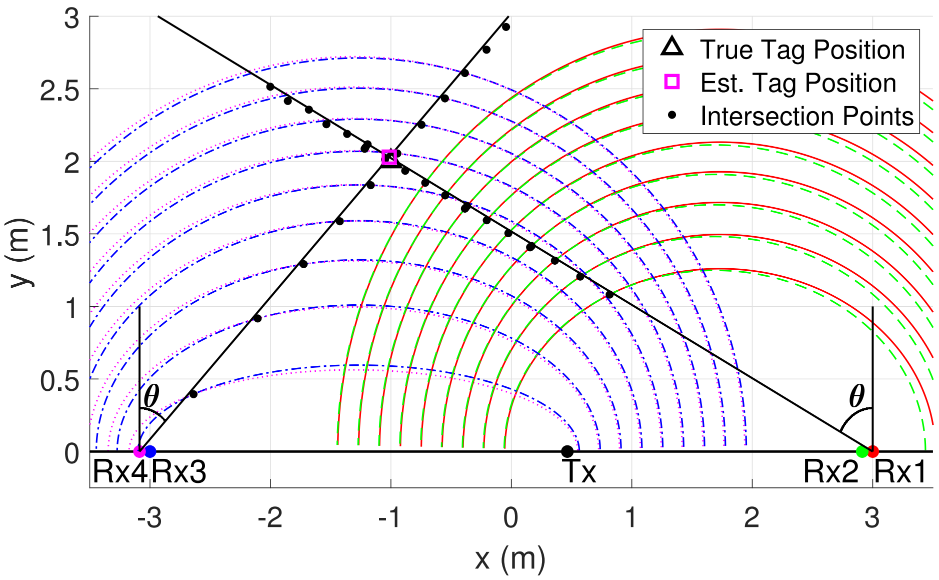


Figure 5.2: Multistatic 2D localization of an RFID tag, with increased wrapping in the phase measurements.

performing compared to `relu`. However, in the process of experimenting and refining the NNs `relu` frequently outperformed its variations, thus no safe conclusion can be drawn. The error increase from `tanh` to `elu` and `relu` was 14.5% and 29%, respectively. Finally, `sigmoid` could not ”keep up” with the rest, noting a significantly increased convergence time, caused partially due to a period of decreased learning rate, which indicates that the network tried to converge on a local minimum.

### 5.1.2 Feature Inputs Comparison

As discussed in preceding sections, correlated inputs can pose a challenge during the training of neural networks. However, in this problem, having completely uncorrelated inputs is not feasible as phase differences originate from the same $\phi_{ij}$ values. Moreover, DoA measurements, $\theta_{ij}$, are directly derived from the phase difference measurements $\Delta\phi_{ij}$—as indicated in Lemma 1—, resulting in even stronger correlation.

This raises the question of whether incorporating DoA values as feature inputs is beneficial or detrimental to the network's training process. To

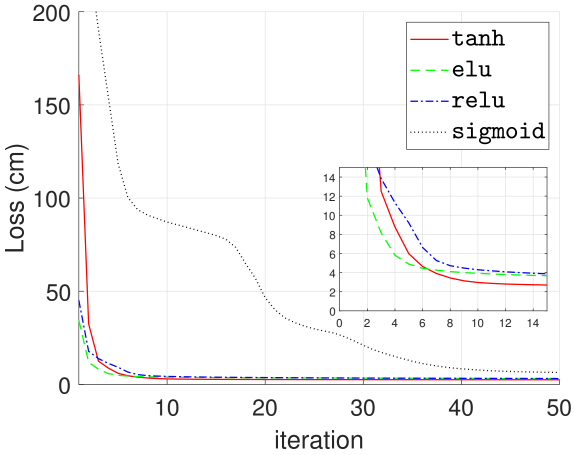


Figure 5.3: Training loss per iteration for various activation functions (Fig. 5.2 system was used).

Table 5.1: Test Loss (cm).

| tanh | elu | relu | sigmoid |
|---|---|---|---|
| 2.48 | 2.84 | 3.20 | 6.45 |

address this, experiments were conducted by varying only the feature inputs of the neural networks. For the original system depicted in Fig. 3.2, three cases were considered: including all four metrics as feature inputs, including only the phase difference measurements, and including only the direction-of-arrival measurements.

The mean losses during the training process are displayed in Fig. 5.4, while the mean test losses are displayed in Table 5.2. Notably, the model using exclusively $\Delta\phi_{ij}$ measurements deviated by only 1 cm from the optimal solution (12% error increase), but exhibited a significantly prolonged convergence time, while in two instances it showed signs of early converging, at around 7 and 17 epochs. Conversely, the utilization of only $\theta_{ij}$ measurements resulted in a fast-converging network that deviated by 4 cm from the target (36% error increase). In conclusion, DoA measurements, while insufficient on their own, introduce additional information that aids the neural network in achieving a more reliable convergence.

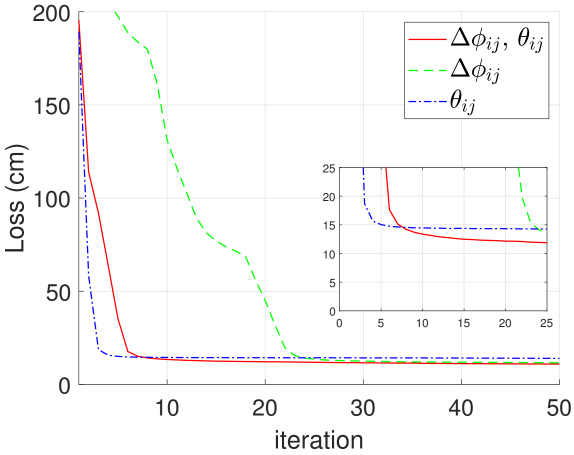


Figure 5.4: Training loss per iteration for various feature inputs (Fig. 3.2 setup was used).

Table 5.2: Test Loss (cm).

| $\Delta\phi_{ij}$, $\theta_{ij}$ | $\Delta\phi_{ij}$ | $\theta_{ij}$ |
|---|---|---|
| 10.55 | 11.79 | 14.37 |

### 5.1.3 Ambiguous vs. Non-Ambiguous Data

In Section 4.4, the impact of the distance between Rx antenna pairs on the phase wrapping phenomenon was analyzed, by comparing the two setups that produced ambiguous data. Displaying their errors in Fig. 5.5 and Table 5.3, reveal that the setup with increased distance between the pairs and, thus, increased number of wraps, is better performing, though it needs slightly more iterations to converge.

In an effort to comprehend the reason behind the increased efficiency of the best-performing model, experiments were conducted, where only ambiguous and only non-ambiguous data were used. The results, showcased in Fig. 5.6 and Table 5.4, revealed a remarkable dependency of the efficiency on the non-ambiguous data, with the inclusion of ambiguous data, surprisingly, having almost no effect on the loss. Despite maintaining four feature inputs in the non-ambiguous case, the increased distance between antenna pairs augments the differences in measurement values across the area of interest. This

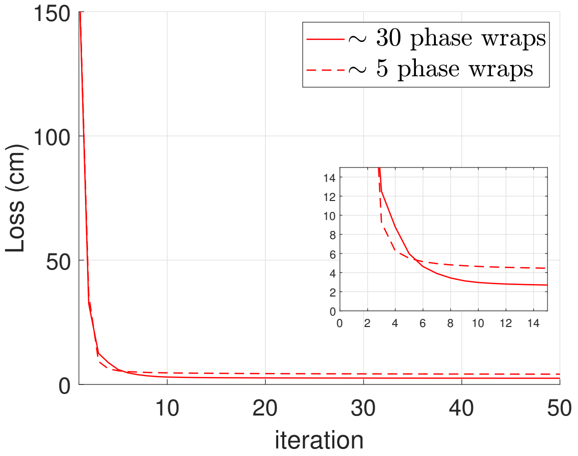


Figure 5.5: Training loss per iteration for the two ambiguous systems.

Table 5.3: Test Loss (cm).

| $\sim$ 30 phase wraps | $\sim$ 5 phase wraps |
|---|---|
| 2.48 | 4.32 |

observation is visualized in Fig. 5.7 and Fig. 5.8, where the non-ambiguous $\Delta\phi_{ij}$ distributions of the setup in Fig. 5.2 are presented. It can clearly be seen that the line of phase discontinuity follows the perpendicular bisector of the line segment formed by the antenna pair. Having two separate pairs, means that they can be positioned apart from each other, thus enabling the neural network to extract more valuable information. In contrast, the original setup with three Rx antennas, could not afford such flexibility, resulting in a similar distribution in the values of $\Delta\phi_{ij}$ measurements and, thus, noticeably lower efficiency compared to systems with four Rx antennas.

To conclude, some intriguing results are presented in Fig. 5.9 and Table 5.5. Various neural networks with different feature inputs were trained, for the configuration illustrated in Fig. 5.2. Notably, adding the ambiguous $\Delta\phi_{ij}$ measurements to the non-ambiguous, reduces the performance of the network, increasing the loss by 9.5%. Finally, similarly to the setup on Fig. 3.2, the NN that included both $\Delta\phi_{ij}$ and $\theta_{ij}$ measurements, performed better than its counterpart that only used phase differences, with the former having a MAE of 2.51 cm and the latter 3.18 cm (21% improvement). These

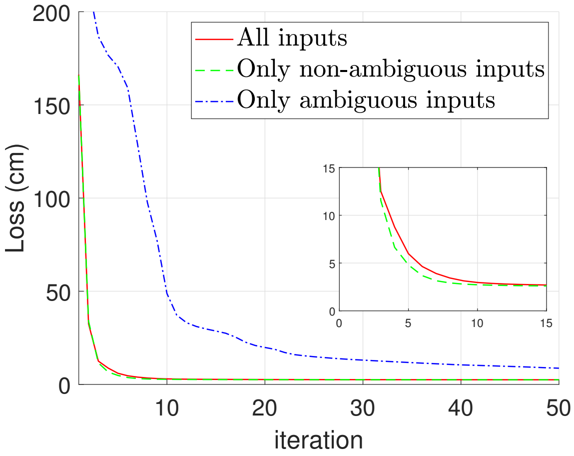


Figure 5.6: Training loss per iteration for various feature inputs (Fig. 5.2 setup was used).

Table 5.4: Test Loss (cm).

| All inputs | Only non-ambiguous | Only ambiguous |
|---|---|---|
| 2.48 | 2.51 | 8.17 |

observations further support the earlier assumption that DoA-trained neural networks do not perform well, but their inclusion as feature inputs create better performing and more robust neural networks. It only remains to be confirmed with experimental data.

### 5.1.4 Optimal NN

To summarize, the optimal neural network configuration, with a MAE of 2.48 cm, was based on the setup displayed in Fig. 5.2 and used `RobustScaler` for input scaling, `Adam` optimizer, `tanh` activation function[4] and included all eight metrics as feature inputs. For the setup in Fig. 3.2, the optimal NN reported a MAE of 10.55 cm and had the same characteristics, apart from the feature inputs, where the total number was four. Avoiding DoA inputs, to create a more fair comparison of the NNs with the traditional methods, the

---

[4] As it will be seen in Section 5.5, `relu` was more robust when tested with experimental data.

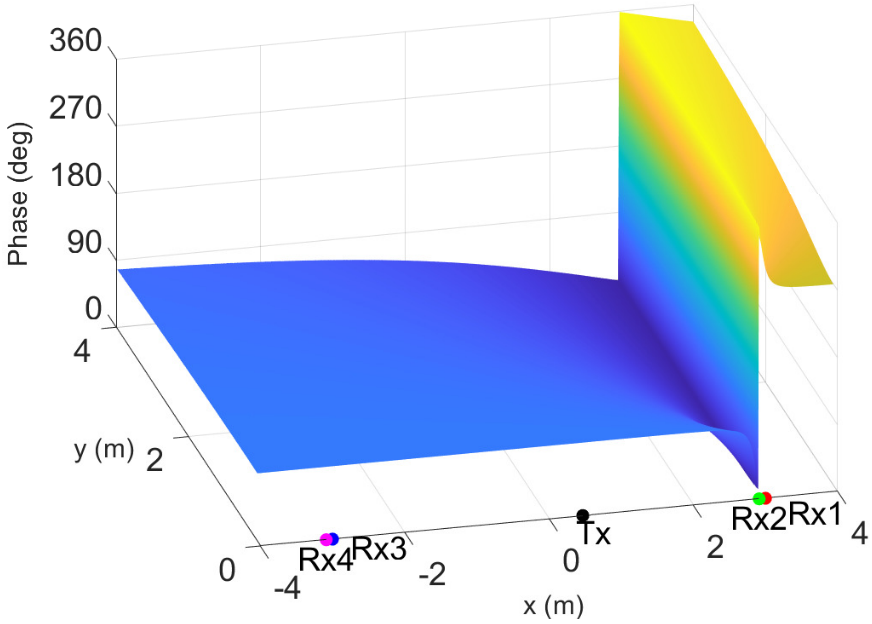


Figure 5.7: Phase Difference $\Delta\phi_{12}$ distribution.

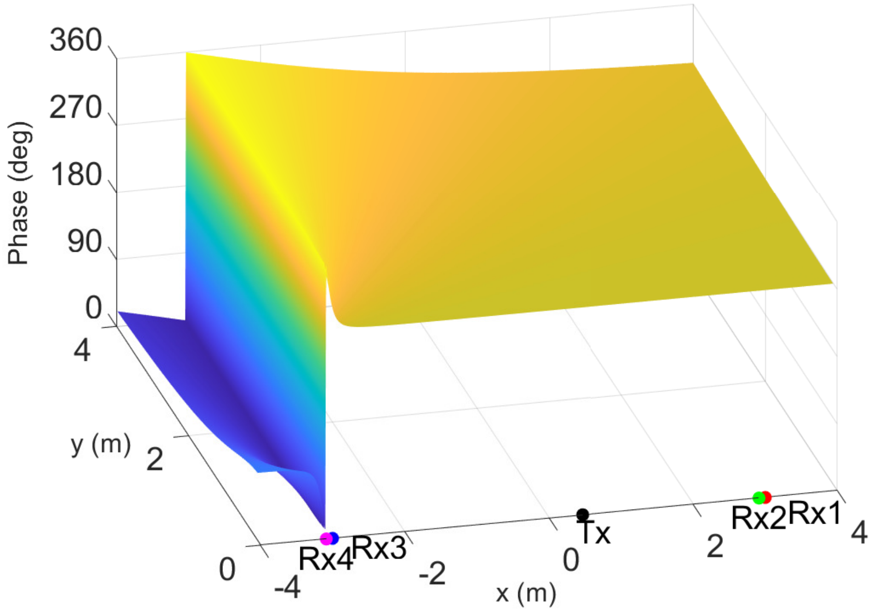


Figure 5.8: Phase Difference $\Delta\phi_{34}$ distribution.

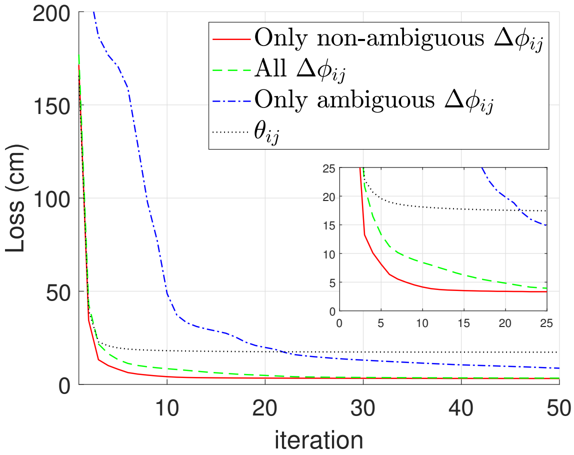


Figure 5.9: Training loss per iteration for various feature inputs (Fig. 5.2 setup was used).

Table 5.5: Test Loss (cm).

| Non-amb. $\Delta\phi_{ij}$ | All $\Delta\phi_{ij}$ | Amb. $\Delta\phi_{ij}$ | $\theta_{ij}$ |
|---|---|---|---|
| 3.18 | 3.48 | 8.17 | 17.19 |

MAE reported was 11.79 and 3.18 cm, for Fig. 3.2 and Fig. 5.2 respectively.

## 5.2 Model Mismatch

A reliable method to evaluate if a neural network has overfitted, is to test it using a dataset with slightly different parameters. This way the fluctuations in the parameters that a real-world setup would have, can be somewhat replicated, while the generalization capacity of the network can be determined too. Datasets of $10^5$ samples were created for the setups in Fig. 3.2 and Fig. 5.2, where only the scattering components were changed. For the decreased path losses scenario $k_{\mathrm{CT}} = 20$, $k_{\mathrm{CR},i} = 20$ and $k_{\mathrm{TR},i} = 15$ were selected, while for increased path losses the values were $k_{\mathrm{CT}} = 10$, $k_{\mathrm{CR},i} = 10$ and $k_{\mathrm{TR},i} = 7$. The corresponding losses are displayed in Table 5.6, where a 2.6% and 2.8% decrease in loss, for Fig. 3.2 and Fig. 5.2, respectively, was noted in the first case. In the second case a 1.9% and 0.8% increase in error was reported.

Table 5.6: Test Loss for different values of scattering components (cm).

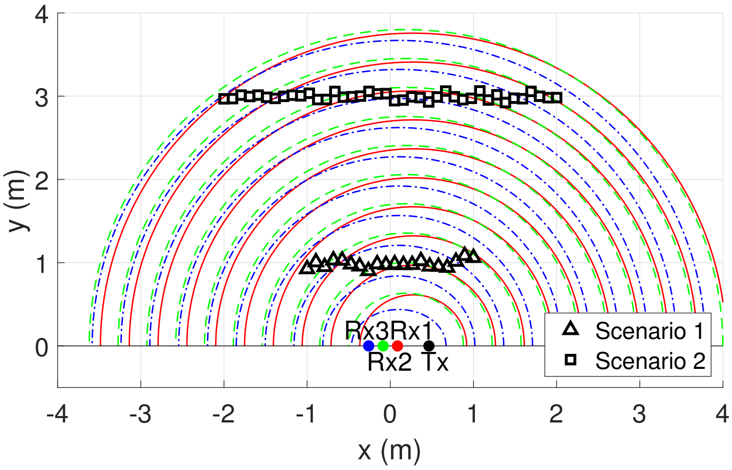
| cm | Opt. NN | Better $k_i$ | Worse $k_i$ |
|---|---|---|---|
| Fig. 3.2 | 10.55 | 10.28 | 10.75 |
| Fig. 5.2 | 2.48 | 2.41 | 2.50 |

## 5.3 Neural Networks vs. Traditional Phase-Based Techniques

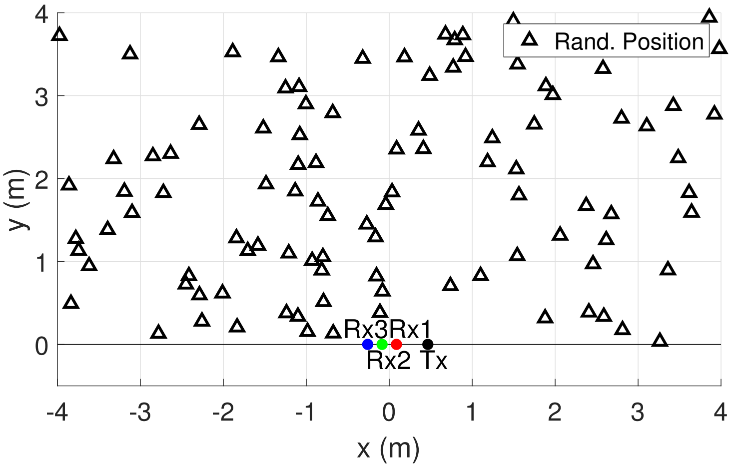
To evaluate the phase-based techniques, as analyzed in Chapter 3, simulations were conducted, using the setups depicted in Fig. 3.2 and Fig. 5.2. Specifically, the mean absolute error ($\mathbb{E}[|\boldsymbol{e}|]$) and the mean squared error ($\sqrt{\mathbb{E}[|\boldsymbol{e}|^2]}$), for various given paths of the tag and then for random tag locations, were calculated. The median absolute error ($|\tilde{\boldsymbol{e}}|$) was also computed to provide a more comprehensive understanding, as some Methods reported significantly increased error for specific positions (usually above the receiving antennas) and a noteworthy difference between mean and median was apparent.

The two paths displayed in Fig. 5.10a were considered, consisting of 20 and 40 tag locations respectively. For the first, 100 iterations were conducted, while for the second 50, thus each amassing to 2000 samples. For the random location scenario, depicted in Fig. 5.10b, 5000 samples were taken, in order to assure adequate coverage of the area. In all three cases, a total of 20 set of ellipses, centered approximately around the tag, were computed. After some experiments, it was selected as a fair balance between running time and coverage. As already mentioned, to compare the path-based results with the optimal NN, the mean error over three NNs is computed, for three different random states, i.e., three different splits of the data.

The error for the setup in Fig. 5.2 was evaluated, only comparing Method 2.5, which does not use ellipses to estimate the tag location, with the optimal NN for this situation. The evaluation of this setup using the ellipse-based techniques, was deemed extremely expensive computationally and thus, not applicable in a real world scenario. The results for all scenarios are shown in Table 5.7 & Table 5.8.



(a) Two path scenarios



(b) Random tag positions.

Figure 5.10: Movement Scenarios of the tag.

In the case of the three Rx antenna setup depicted in Fig. 3.2, Method 2 demonstrates superior performance compared to the rest, including the optimal NN, across both path scenarios. However, the NN exhibits a clear advantage for randomly selected tag positions, while Method 2 simultaneously presents the highest mean and lowest median absolute errors among traditional methods. Moreover, a significant disparity between mean and median absolute errors is observed for the neural network across all scenarios, attributed partly to the tag's passage in front of the "invisible lines",

Table 5.7: Simulation results of multistatic 2D localization in Fig. 3.2, considering different localization techniques and movement scenarios ($\phi^n \sim \mathcal{N}(0, 4\pi^2)$).

| cm | Movement Scenario | $E[\lvert\boldsymbol{e}\rvert]$ | $\lvert\tilde{\boldsymbol{e}}\rvert$ | $\sqrt{\mathbb{E}[\lvert\boldsymbol{e}\rvert^2]}$ |
|---|---|---|---|---|
| Method 2 | Scenario 1 | **0.51** | **0.34** | **0.71** |
| | Scenario 2 | **2.41** | **1.94** | **3.21** |
| | Rand. Position | 39.86 | 7.11 | 70.56 |
| Method 2.5 | Scenario 1 | 17.97 | 8.76 | 33.02 |
| | Scenario 2 | 23.42 | 8.71 | 67.80 |
| | Rand. Position | 21.47 | 8.91 | 56.26 |
| Method 3 | Scenario 1 | 4.84 | 4.52 | 5.65 |
| | Scenario 2 | 9.68 | 8.27 | 11.77 |
| | Rand. Position | 30.91 | 13.35 | 56.19 |
| Opt. NN | Scenario 1 | 2.13 | 0.81 | 6.04 |
| | Scenario 2 | 3.09 | 2.24 | 4.59 |
| | Rand. Position | **10.55** | **2.56** | **34.15** |

Table 5.8: Simulation results of multistatic 2D localization in Fig. 5.2, considering different localization techniques and movement scenarios ($\phi^n \sim \mathcal{N}(0, 4\pi^2)$).

| cm | Movement Scenario | $E[\lvert\boldsymbol{e}\rvert]$ | $\lvert\tilde{\boldsymbol{e}}\rvert$ | $\sqrt{\mathbb{E}[\lvert\boldsymbol{e}\rvert^2]}$ |
|---|---|---|---|---|
| Method 2.5 | Scenario 1 | 1.70 | 1.66 | 1.72 |
| | Scenario 2 | 4.68 | 4.61 | 4.70 |
| | Rand. Position | 5.60 | 4.44 | 14.50 |
| Opt. NN | Scenario 1 | **0.43** | **0.39** | **0.49** |
| | Scenario 2 | **0.60** | **0.54** | **0.70** |
| | Rand. Position | **2.48** | **0.83** | **10.41** |

where the values of $\Delta\phi_{ij}$ and $\theta_{ij}$ become zero, as depicted in Figures 4.3 & 4.4. The network's ability to estimate the $y$ coordinate in that area is heavily compromised, leading to degraded estimations. Additionally, in the two path scenarios both geometric methods outperform Method 2.5, which utilizes DoA intersection, but fall short for random tag positions. Exceptions include the remarkably low median absolute error of Method 2 and the

RMSE of Method 3, which is comparable to that of Method 2.5. Overall, the NN demonstrates adequate efficiency across all scenarios, exhibiting at least half the error of the best-performing traditional method in all metrics for random tag positions. However, all methods have their weak areas, as illustrated by the differences between mean and median values.

Transitioning to the setup of Fig. 5.2, the results clearly confirm that it is superior compared to the previous, with the optimal NN achieving sub-1 cm error in most tag positions. A notable 76.5% drop in mean absolute error is observed for random tag positions, with similar improvements reported across all cases. The gap between mean and median values vanishes in all cases, except for the random tag position of the optimal NN, where it is mitigated, but not completely eliminated. The increased error values that create this discrepancy are once again reported for tags placed above the antenna pairs, but as it has already been stated, the increased distance between the pairs heavily benefits the NN.

## 5.4 RDPF: A monostatic alternative

A viable alternative, in cases where the tag remains static for extended periods, involves employing a monostatic setup with a moving reader. Recent work [20] introduced a method of utilizing phase-based particle filtering to estimate tag locations. This method involves mounting the antenna on a robotic platform, allowing for multiple measurements as the antenna moves to evaluate the tag's position reflecting the signal.

Following experimentation with various parameters, the results presented in Table 5.9 were obtained. Notably, with 500 measurements per tag (i.e., 500 antenna positions), the Mean Absolute Error (MAE) closely matched that of the optimal Neural Network (NN) for the setup depicted in Fig. 5.2. Interestingly, the RDPF1 method exhibited a significantly higher median and lower Root Mean Square Error (RMSE), indicating that it may be more robust. Further evaluation based on two path scenarios[5] corroborated this

---

[5] The evaluation of RDPF in a moving tag scenario is not realistic and it was only performed for comparison purposes.

Table 5.9: Simulation results of monostatic 2D localization using RDPF1 [20], with $\phi^n \sim \mathcal{N}(0, 5°)$, $M = 10^5$ particles in an 8x4 $m^2$ area (cm).

| Movement Scenario | $E[\|\boldsymbol{e}\|]$ | $\|\tilde{\boldsymbol{e}}\|$ | $\sqrt{\mathbb{E}[\|\boldsymbol{e}\|^2]}$ |
|---|---|---|---|
| Scenario 1 | 2.12 | 1.49 | 2.53 |
| Scenario 2 | 2.86 | 2.62 | 3.56 |
| Rand. Position | 2.61 | 2.14 | 3.25 |

observation; significantly reduced errors for the NN imply that it suffers from outliers in certain regions, which, as already said, holds true for the areas above the Rx antennas.

Finally, an attempt was made to estimate the tag position using only four measurements, but as anticipated, it failed, consistently placing the tag at the center of the particle initialization area. RDPF relies on an increased amount of measurements which mimics the presence of numerous antennas, to overcome limitations presented by the monostatic setup. Consequently, for scenarios involving moving tags or when the antennas cannot be moved and a small amount of antennas are at disposal, this method proves impractical.

## 5.5 Experimental Evaluation

Two equivalent experimental setups were created, in order to evaluate the results yielded via simulations; one in the multipath-rich lab environment and one in a wider space, with higher ceiling, outside the amphitheater of the ECE building. The hardware used, consisted of a Gen2 UHF RFID tag, three USRP N200 Software Defined Radio (SDR) devices, with FLEX900 daughterboards. The external function generator used for the synchronization of the SDRs was the RIGOL DG2042 and the antennas used were two MT-242032/NRH 7dBic (1 Tx, 1 Rx) and two FlexiRay SF2 5.5dBic (2 Rx)[6]. Tx power and carrier frequency were set identical to the simulation parameters: $P_c$ = 25dBm and $f_c$ = 865MHz. Unfortunately, training with experimental data was not feasible due to the significant time required to

---

[6] Both types of antennas are Right-Hand Circularly Polarized (RHPC)

create sufficiently large datasets and, thus, the experimental data were used only for testing.

In regard to the topology, a near identical setup to that of Fig. 3.2 was constructed, with sub-1 cm error in the true position measurements of the antennas and the tag, while the whole antenna array was shifted 0.465m to the left, intending to place the Tx antenna at the origin. All antennas and tags were set at a height of 1.05m[7]. A custom movable steel rack housed all the devices, apart from the antennas and the tag.

### 5.5.1 Scenario 1: Lab (Rich Multipath)

For the first scenario, the lab environment was selected, known for its many reflective surfaces and multipath rich characteristics. Fig. 5.11, displays the setup constructed for the experimental evaluation of this scenario. A database of 20 samples was created, with the tag positions being displayed in Fig. 5.12. Two of them (10% of the total) were deemed unusable, as at least one of the Direction of Arrival (DoA) values was complex and could not be used. The error results for all methods are presented in Table 5.10. Given the problems accompanying this setup, the notable increase in the error measurements, compared to the simulations, was justifiable. However, the analogies between the various methods and the NNs broadly held true, with the best-performing NNs being within 1 cm of the Methods 2 & 3 in all three metrics, indicating that Method 2 performed slightly worse than expected and Method 3 slightly better. Method 2.5 suffered, exhibiting twice the error values of the rest, which was also justifiable considering the proximity of the tags to the antennas and the poor performance of this method in simulation measurements for the first path scenario.

Furthermore, a slight disturbance in the results was caused by the presence of some outliers (specifically three) in one of the three, theoretically optimal, NNs[8], resulting in somewhat skewed results. Due to the small amount

---

[7] Even though the methods in question perform 2D localization, i.e., the height error is not computed, it is important to denote the height at which the measurements were taken, as it affects the reflections.

[8] As it was clarified in Section 5.1, the mean over three separate NNs is calculated,

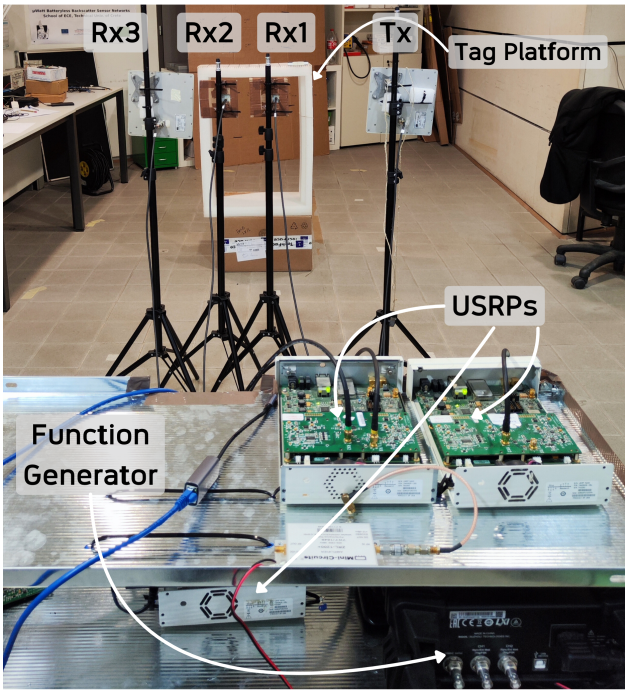


Figure 5.11: Experimental setup for multistatic 2D localization, inside the lab.

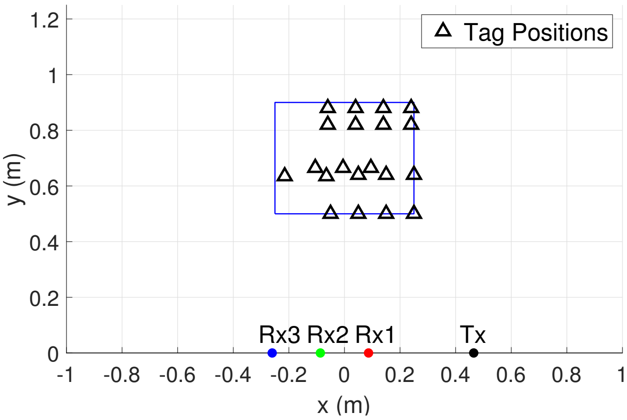


Figure 5.12: Tag Positions for the first experimental scenario.

of samples, all three metrics were affected from the problematic NN, reporting twice the mean absolute error of the other two (18 cm difference), while even the median was approximately 5 cm higher. Consequently, the averages

---

each using different random state, to ensure reliability.

Table 5.10: Experimental results of multistatic 2D localization in multipath rich environment, for various feature inputs and activation functions. Tags and antenna placement shown in Fig. 5.12 were used.

| cm | $E[\lvert\boldsymbol{e}\rvert]$ | $\lvert\tilde{\boldsymbol{e}}\rvert$ | $\sqrt{\mathbb{E}[\lvert\boldsymbol{e}\rvert^2]}$ |
|---|---|---|---|
| Method 2 | **18.66** | **8.09** | **30.56** |
| Method 2.5 | 41.40 | 21.78 | 61.12 |
| Method 3 | **18.41** | **8.07** | **30.98** |
| $\Delta\phi_{12}, \Delta\phi_{23}, \theta_{12}, \theta_{23}$ - `tanh` | 24.54 | 9.86 | 41.15 |
| $\Delta\phi_{12}, \Delta\phi_{23}$ - `tanh` | 37.76 | 14.03 | 63.65 |
| $\Delta\phi_{12}, \Delta\phi_{23}, \theta_{12}, \theta_{23}$ - `relu` | **18.97** | **8.25** | **30.57** |
| $\Delta\phi_{12}, \Delta\phi_{23}, \hat{\theta}_{12}, \hat{\theta}_{23}$ - `relu` | **18.93** | **8.21** | **30.70** |
| $\Delta\phi_{12}, \Delta\phi_{23}$ - `relu` | **18.97** | **8.51** | **30.51** |

of the three NNs were worse than the ellipses-based methods.

For this reason, the results of the NNs trained with `relu` as the activation function are also displayed, which did not suffer from outliers and were comparable with Methods 2 & 3. Moreover, the results of the NN trained without DoA and with estimated DoA (given by Eq. 3.9), from the second method are also presented[9]. For the estimated DoA, it must be noted that any advantage in the time division that the NN could have, vanishes and the comparable results of the NN with the traditional methods, suggest that the only reason to use such a NN is for cross validation.

The outliers issue noted above, even though minor, created uncertainty over which really is the "optimal NN". Therefore, this term will not be used in the current Section, as it creates confusion. A first evaluation of the situation, suggests that the `relu`-trained NN is more reliable than the `tanh`-trained. This statement is reinforced by the findings regarding the NN that exclusively used $\Delta\phi_{ij}$ measurements as inputs, where the number of outliers for the `tanh`-trained one heavily increased, deeming it unusable, while the `relu`-trained alternative was on par with the best performing methods

---

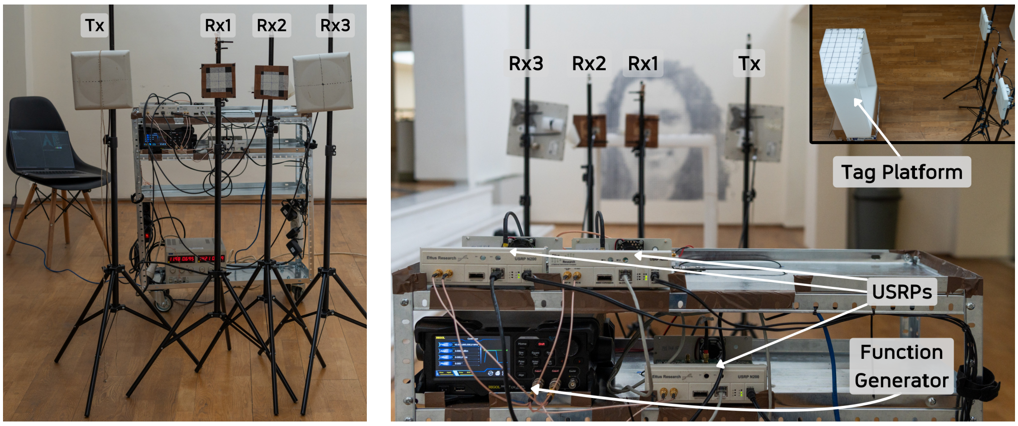
[9] DoA trained NNs have an "unfair advantage" over traditional methods, as the quadrant where the tag is located is considered known (see Lemma 1). Thus, they cannot be the only comparison present.

(fact that also refutes the significance of DoA measurements). Given the small sample size, a safer conclusion will be drawn, after the analysis of the forthcoming results in the next experimental scenario.

### 5.5.2 Scenario 2: Amphitheater Entrance

For the second scenario, an unknown—in terms of reflections—location was selected, seen in Fig 5.13, which was believed to suffer less from reflections. A wider range of samples was taken, with the tags being on average further away from the array of antennas, compared to the previous dataset, surpassing the one meter "boundary". Results shown in Table 5.11, reveal a setup that had more outliers compared to the first, i.e., samples with extremely high error, reporting an almost 10 cm increase in the mean absolute error, but being within a centimeter when using the median for evaluation. RMSE had a comparable increase to MAE measurements (40% and 60% for MAE and RMSE, respectively).

Given the results, the assumption about the better characteristics of the location cannot be confirmed, but it is important to note that the mean distance of the tag from the antennas was higher, therefore increasing the noise factor. Moreover, the superiority of `relu` activation function, over `tanh`



(a) Front view. (b) Back view.

Figure 5.13: Experimental setup for multistatic 2D localization, outside the ECE amphitheater.

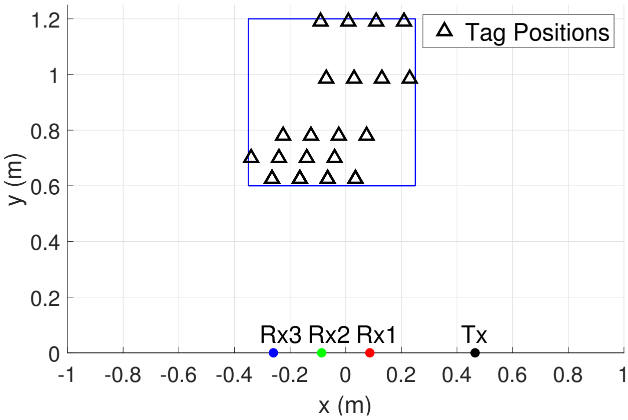


Figure 5.14: Tag Positions for the second experimental scenario.

Table 5.11: Experimental results of multistatic 2D localization (unknown multipath reflections). Tags and antenna placement shown in Fig. 5.14 were used.

| cm | $E[\|\boldsymbol{e}\|]$ | $\|\tilde{\boldsymbol{e}}\|$ | $\sqrt{\mathbb{E}[\|\boldsymbol{e}\|^2]}$ |
|---|---|---|---|
| Method 2 | **26.46** | **8.78** | **50.20** |
| Method 2.5 | 49.66 | 50.47 | 64.68 |
| Method 3 | **27.01** | **8.24** | **51.97** |
| $\Delta\phi_{12}, \Delta\phi_{23}, \theta_{12}, \theta_{23}$ - `tanh` | 37.30 | 13.10 | 64.25 |
| $\Delta\phi_{12}, \Delta\phi_{23}$ - `tanh` | 63.97 | 49.49 | 90.06 |
| $\Delta\phi_{12}, \Delta\phi_{23}, \theta_{12}, \theta_{23}$ - `relu` | **26.36** | **8.43** | **49.25** |
| $\Delta\phi_{12}, \Delta\phi_{23}, \hat{\theta}_{12}, \hat{\theta}_{23}$ - `relu` | **26.89** | **8.78** | **49.57** |
| $\Delta\phi_{12}, \Delta\phi_{23}$ - `relu` | **26.95** | 10.27 | **50.00** |

is established, while the significance of the DoA measurements in the training process, is yet again disputed.

# Chapter 6

# Conclusions

## 6.1 Conclusion

There are two main takeaways from this thesis. Firstly, it demonstrates the viability of neural networks (NNs) as a competitive alternative to traditional methods in real-world scenarios. NNs produce comparable results, whether or not Direction-of-Arrival (DoA) measurements are utilized, while exhibiting robustness against various noise and multipath profiles and operating at high speeds. Remarkably, these NNs achieve such performance despite being trained on a single comprehensive dataset, with simplistic modeling of multipath effects and constant channel variables. This underscores their incredible potential for implementation in applications where time and cost are critical factors, with minimal disruption to installation sites. Moreover, when trained on equally large datasets generated from experimental setups, NNs have the potential to outperform traditional techniques, by capturing the intricate dependencies between phase measurements and tag positions. In essence, these results serve as a proof-of-concept that NNs can surpass their traditional counterparts.

The second takeaway revolves around the concepts of increased antenna distances and ambiguity in the phase measurements. It has been proven in simulations that increased distance between antenna pairs are vastly beneficiary for neural networks, as well as for traditional methods (DoA intersection), reducing the error values to an impressive one quarter of what they previously were. Adjusting for transmission power, coverage from the antennas and other variables on an experimental setup, it is undeniable that a 5 antenna setup, similar to those depicted in Fig. 4.10 and Fig. 5.2, has the potential to outperform the 4 antenna one, created for the experimental evaluation of this work. Furthermore, ambiguity in the data itself did not help in the localization process, contrary to previous assumptions regarding monostatic setups [10]. However, for a neural network trained with experimental data, where outliers are inevitable and convergence may be challenging, the deliberate introduction of ambiguous data—with a carefully selected amount of phase wrapping—may prove beneficial. Ultimately, as it was already proven, what is beneficiary or optimal in a simulated environment does not always translate to the real world.

And with that in mind, some discrepancies between simulations and experiments must be addressed. First and foremost the activation function selected. Even though the findings in bibliography [9] suggested that `relu` was the preferred choice, after the simulation results, the deliberate choice was made to declare `tanh` as the optimal option and thus differentiate. The truth is that there were no indications to believe otherwise, given that both functions performed in a similar fashion, as seen in Fig. 5.3, with `tanh` emerging as the superior choice in most cases, but not by much. However, as it was made clear in Section 5.5, the correlation between simulations and experiments was not that strong, with `relu` activation function undeniably being the preferred choice, especially when DoA values were not used. Speaking of DoA measurements, it was also perceived (partly due to the findings concerning activation function) that they did not have a substantial impact on the efficiency of the NNs, especially given the fact that they required partial knowledge of the tag location.

## 6.2 Future Work

Arriving to the last Section, where ideas for future continuation of this work are given, it must be said that a bittersweet feeling emerges, as some of the following were not beyond reach. First and foremost, it would be very interesting to extract experimental results from the systems with two pairs of receiving antennas. Besides of the increased information in the distribution of the two phase differences, as seen in Fig. 5.7 and Fig. 5.8, such configurations offer the flexibility to strategically position antennas at the edges of the area of interest. This change would reposition ”blind spots” near the edges[1], where tag positioning may be less probable. Moreover, this setup unlocks the potential for leveraging Direction of Arrival (DoA) measurements, as the tag’s position, relative to the antenna pair, remains consistently known, i.e., either always on the right or on the left of the antenna pair. Let’s not forget that ambiguous data will be available in such a setup and despite the findings of this thesis, it would be intriguing to experiment further and test them with experimental data.

This setup, can also be altered either to strategically position the weak spots in areas of less significance, or to adjust phase wrapping for performance optimization. Additionally, scaling up this system, by employing multiple pairs of receiving antennas at varying distances, could prove advantageous. This approach enables the user to divide the area to subdivisions where the two neighboring pairs are used to estimate the tag location, or to unify all the information to a single NN. However the use of DoA will once again require partial knowledge of the tag position. Finally, in all setups, probability theory could also be used, by creating a grid and equating the relative likelihood that the tag is positioned in a each grid slot in the area and then using that information to decide where the antennas should be positioned. However, this requires additional time spent and disturbance at the installation site.

As the viability of the neural networks has already been proven, the next step would be to evaluate their performance when trained with experimental data. To achieve that, the only realistic solution would be to automate the procedure of data gathering, for example with the use of a moving robotic platform upon which the tag is placed. A necessary condition is for the true position of the tag to be calculated with no error.

Finally, addressing the challenges posed by unsynchronized Software Defined Radios (SDRs) opens new possibilities. It would be intriguing to assess the neural network’s performance when residual carrier and phase offsets (CFO & CPO) are present.

---

[1] As already established, regions above the antennas pairs have reduced performance due to the feature inputs being near zero, along the perpendicular bisector.

# Bibliography

[1] K. Skyvalakis, E. Giannelos, E. Andrianakis, and A. Bletsas, “Elliptical DoA Estimation & Localization,” in *2021 IEEE International Conference on RFID Technology and Applications (RFID-TA)*, New Delhi, India, Oct. 2021.

[2] *EPC Radio-Frequency Identity Protocols, Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHZ-960 MHZ, version 1.2.0.* EPC Global, 2008.

[3] A. Goldsmith, *Wireless Communications.* New York, NY, USA: Cambridge University Press, 2005.

[4] J. Kimionis, A. Bletsas, and J. N. Sahalos, “Increased range bistatic scatter radio,” *IEEE Transactions on Communications*, vol. 62, no. 3, pp. 1091–1104, Mar. 2014.

[5] A. Bletsas, A. G. Dimitriou, and J. N. Sahalos, “Improving backscatter radio tag efficiency,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 58, no. 6, pp. 1502–1509, May 2010.

[6] M. Ouroutzoglou, G. Vougioukas, G. N. Karystinos, and A. Bletsas, “Multistatic noncoherent linear complexity Miller sequence detection for Gen2 RFID/IoT,” *IEEE Transactions on Wireless Communications*, vol. 20, no. 12, pp. 8067–8080, Dec. 2021.

[7] M. Rice, *Digital Communications: A Discrete-time Approach.* Pearson/Prentice Hall, 2009.

[8] J. Richter-Gebert, *Perspectives on Projective Geometry: A Guided Tour Through Real and Complex Geometry.* Springer Berlin, Heidelberg, 2005.

[9] S. Megalou, A. R. Chatzistefanou, A. Tzitzis, T. V. Yioultsis, and A. G. Dimitriou, "Passive UHF-RFID hyperbolic positioning of moving tags by exploiting neural networks," *IEEE Journal of Radio Frequency Identification*, vol. 6, pp. 402–412, Mar. 2022.

[10] L. Mo, Y. Zhu, and D. Zhang, "UHF RFID indoor localization algorithm based on BP-SVR," *IEEE Journal of Radio Frequency Identification*, vol. 6, pp. 385–393, Jan. 2022.

[11] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, *Efficient BackProp.* Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 9–50.

[12] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, Madison, WI, USA, Jun. 2010.

[13] B. Ding, H. Qian, and J. Zhou, "Activation functions and their characteristics in deep neural networks," in *2018 Chinese Control And Decision Conference (CCDC)*, Jun. 2018.

[14] A. L. Maas, A. Y. Hannun, A. Y. Ng *et al.*, "Rectifier nonlinearities improve neural network acoustic models," 2013.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, Dec. 2015.

[16] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[17] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, Georgia, USA, 17–19 Jun. 2013.

[18] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, 2011.

[19] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[20] E. Giannelos, E. Andrianakis, K. Skyvalakis, A. G. Dimitriou, and A. Bletsas, “Robust RFID localization in multipath with phase-based particle filtering and a mobile robot,” *IEEE Journal of Radio Frequency Identification*, vol. 5, no. 3, pp. 302–310, Sept. 2021.