Tobias Gailhofer, BSc.

Robust Real-Time Tracking for UWB based Smart Car Access Systems

Master Thesis

Betreuer
Assoc. Prof Dr. Klaus Witrisal
Dipl. Ing. Alexander Venus
Dr. Stefan Tertinek, NXP Semiconductors Austria GmbH

Institute for Signal Processing and Speech Communication
Graz University of Technology, Austria

in co-operation with
NXP Semiconductors Austria GmbH
Gratkorn, Austria

Graz, May 2023

Affidavit

I declare that I have authored this master's thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date

Signature

Acknowledgements

This thesis has been conducted at the Institute of Signal Processing and Speech Communication at Graz University of Technology.
First and foremost, I would like to thank my supervisor Alexander Venus for all the advice and help throughout this thesis, as well as my supervisors Klaus Witrisal and Stefan Tertinek together with NXP Semiconductors Austria Gmbh.
Furthermore, I want to thank my partner, family, and friends for their constant support and motivation.

Graz, May 2023 Tobias Gailhofer

Contents

Abstract

This thesis focuses on developing and testing an Ultra-Wideband (UWB) ranging and positioning measurement setup based on the NXP Ranger4 UWB chip, and the implementation and evaluation of tracking filters with the application in smart car access systems. A novel tracking filter is proposed, which combines a particle-based probabilistic data association (PDA) filter with a machine learning (ML)-fingerprinting approach. The measurement setup was built and developed based on a previous project and adapted to enable wireless, real-time measurements using the Ranger4 chips. In combination with an optical reference tracking system, five training datasets and eight test trajectories were captured in a car-based scenario, using different combinations of robot and human agents. A metal obstruction was used for parts of the measurements to simulate Non-Line-of-Sight (NLOS) situations and collect data for testing the tracking filter in suboptimal conditions. As baseline tracking filters, an Extended Kalman filter (EKF) and a Particle filter were implemented in Python. A particle-based PDA filter was also implemented, tracking not only the position but also the probability of detection for Line-of-Sight (LOS) components. This PDA filter was combined with a Gaussian process regression (GPR), to map and predict signal parameters from the measurement area based on the predicted agent position, and match these parameters to the observed signal parameters. Two variants of the two PDA-based filters were implemented respectively, one with a fixed probability of LOS detection, and the other with a tracked probability. Analysis of the filters on the test trajectories showed a performance gain for the filter using the ML augmentation without tracking the probability of LOS detection, while the same filter that was tracking the LOS detection probability showed showed poorer performance when compared to the other PDA filters. Performance gains for the ML augmentation were different, depending on the match of obstruction layout between training and test datasets.

Kurzfassung

In dieser Masterarbeit liegt der Fokus auf der Entwicklung und der Testung eines Ultrabreitband Entfernungs- und Positionierungsmessungssystems, das auf NXP Ranger4 UWB-Chips basiert, sowie auf der Umsetzung und Evaluierung von Trackingfiltern mit der Anwendung in Smart Car Access Systemen. Ein neuartiger Trackingfilter, der eine partikelbasierten Probablistic Data Association-Filter PDA-Filter mit dem ML-fingerprinting Zugang kombiniert, wird vorgeschlagen. Die Messanordnung wurde basierend auf einem vorangegangenen Projekt entwickelt und gebaut bzw. adaptiert, um wireless Echtzeit-Messungen mit dem Ranger4-Chip zu ermöglichen. In Kombination mit einem optischen Referenz-Trackingsystem, wurden fünf Trainingsdatensätze und acht Testtrajektorien in einem autobasierten Szenario aufgenommen, wobei unterschiedliche Kombinationen an Roboter- und menschlichen Agenten verwendet wurden. Ein metallisches Hindernis wurde für einen Teil der Messungen verwendet, um eine Non-Line-of-Sight (NLOS)-Situation zu simulieren und Daten für Tests der Tracking-Filter unter suboptimalen Umständen zu sammeln. Als Referenz-Trackingfilter wurde ein Extended Kalman Filter und ein Partikelfilter in Python implementiert. Ein partikelbasierter PDA-Filter, der nicht nur die Position sondern auch die Detektionswahrscheinlichkeit von LOS-Komponenten schätzt, wurde ebenso implementiert. Der PDA-Filter wurde mit der Gauß-Prozess Regression kombiniert, um Signalparameter im Messbereich basierend auf der vorhergesagten Agenten-Position zu vermessen und vorauszusagen und diese mit diesen Parametern mit den gemessenen Signalparametern übereinzustimmen. Zwei Varianten der zwei PDA-basierten Filter werden jeweils implementiert: einer mit einer fixen Wahrscheinlichkeit für die Detektion von LOS und ein anderer mit einer geschätzten Wahrscheinlichkeit. Die Analyse der Filter anhand der Testtrajektorien zeigte eine Verbesserung der Filter, die ML-Augmentation ohne dem Tracking der Wahrscheinlichkeit der LOS-Detektion nutzen, während der gleiche Filter mit dem Tracking der LOS-Dedektionswahrscheinlichkeit im Vergleich mit den PDA-Filtern schlechtere Leistungen zeigten. Niedrigere Fehler konnten für die ML-Augmentation beobachtet werden, dies war abhängig von der Übereinstimmung der Hindernisaufstellung zwischen Trainings- und Testdatensätzen.

List of Symbols

$\mathbf{x}$	Agent state vector
d	Distance
τ	Time of Flight
α	Complex channel gain
t	Time
n	Discrete time index
k	Multipath component index
j	Anchor index
$\mathcal{N}(...)$	Gaussian distribution
T_s	Sampling time
B	Bandwidth
$\hat{B}$	RMS bandwidth
β	Roll-off factor
σ	Standard deviation
$\mathbf{z}$	Measurement vector
$\mathbf{R}$	Measurement covariance matrix
$\mathbf{A}$	Transition matrix of the state space model
$\mathbf{Q}$	Process noise matrix
$\mathbf{P}$	Agent state covariance
$\mathbf{H}$	Jacobian matrix
i	Particle index
w_i	Particle weight
N	Nr. of particles
$\mathbf{B}$	Input matrix of the state space model
u_m	Signal to Noise Ratio
m	Measurement index
M	Nr. of measurements
a	Data association variable
q	Likelihood of LOS detection
$\mathcal{Q}$	Set of discrete detection likelihoods
$\mathbf{Q}$	Transition matrix of the Likelihood of LOS detection
$\mathbf{p}$	Agent position
$\mathbf{P}_{train}$	ML training input values
$\mathbf{Z}_{f,train}$	ML training output values
$\tilde{\mathbf{t}}_f$	ML prediction output values
$\tilde{\mathbf{\Sigma}}_f$	ML prediction output covariance

1 Introduction

Wireless positioning has, in recent years, become a topic of significant interest in both the scientific communities and in the industry. With the advances on the Internet of Things, location awareness has become increasingly important in many aspects of digitalisation. This work is focused on the specific use case of on-person agent tracking in an unknown area. The primary application which is used as the basis of the project is positioning a keyfob agent in relation to a car for smart car access features, such as automated locking and unlocking of doors or opening of the trunk. This work focuses on tracking and positioning an agent using the NXP Ranger4 Ultra-Wideband (UWB) system, developed for this and similar applications needing precise wireless ranging.

Ultra-Wideband technology enables measuring distances between an agent/anchor pair by estimating the round-trip Time of Flight (ToF) of the electromagnetic waves between the two antennas. Using at least three anchors with known positions $\mathbf{x_i} = [x_i, y_i]^T$, the position of the agent $\mathbf{x} = [x, y]^T$ can be estimated via trilateration. The technical advantage of using UWB signals compared to other ranging technologies is the low pulse duration of UWB pulses (pulse duration $T_s = 2\,\text{ns}$ for a bandwidth of $B = 500MHz$), and therefore their high Nyquist resolution. This leads to an accuracy level in the cm-range, with theoretical limits below 1 cm for Signal-to-Noise-Ratios (SNRs) $\gtrsim 25\,\text{dB}$. [1] Current research branches into several different aspects of tracking and positioning. The combination of tracking with Machine Learning (ML) based approaches has gained a lot of traction, either directly estimating positions as presented in [2], [3], or as ranging error mitigation [4], [5]. In this thesis, a new approach is proposed, combining a tracking filter with a fingerprinting ML concept using signal parameters.

1.1 Goals and Outline of this Thesis

The goal of this master thesis is to develop a wireless, real time data acquisition and tracking system based on the NXP Ranger4 UWB chip. Building on the hardware setup of a prior project [6], the measurement setup needs to be adapted to manage channel

impulse response (CIR) data transmissions to enable the use of more advanced tracking filters. A dataset is acquired using this updated hardware setup. The dataset consists of multiple scenarios with different impairments to the UWB ranging to evaluate algorithm performance in sub-optimal conditions.
Different advanced tracking filters are implemented and evaluated, using different techniques to overcome the ranging impairments, such as probabilistic data association (PDA), tracking of the likelihood of a Line-Of-Sight detection, and ML. A comparison of the different tracking algorithms on multiple scenarios concludes this thesis, as well as a measurement setup to be used for further works in the field of UWB positioning.

Starting with a brief introduction to UWB ranging concepts, the theoretical concepts of positioning and tracking are presented. The tracking algorithms are discussed, followed by a description of the hardware and software setup and the data acquisition scenarios. Lastly, the results of the filter analysis are presented for the various scenarios, showing filter performance in different situations.

2 Ultra-Wideband Ranging

Wireless ranging describes the estimation of distances between two wireless ranging nodes. Many different protocols exist to determine the distance between two antennas, for example Received Signal Strength (RSS), Time of Arrival (ToA), Time Difference of Arrival (TDoA), Angle of Arrival (AoA), etc. [1]
With the NXP Ranger4 chips, a ToF protocol is used, measuring the time difference between a transmission and the reception of the answer. The transmission structure for UWB is defined in [7]. A ranging preamble is defined, consisting of a sequence of pseudo-random pulses with a "perfect" autocorrelation function close to an isolated UWB pulse $s(t)$. This characteristic causes the correlation of the received signal with the known sequence to represent the channel impulse response. This correlation is approximated by the signal model, which is discussed in the next section. Ranging algorithms and LOS detection methods will also be shortly discussed, as well as the Cramer-Rao Lower Bound for distance estimation.

2.1 Signal Model and Error Sources

Under real-life conditions, many different effects affect the received signal. Main sources of errors are multipath propagation, shadowing or fading of the wireless channel, as well as noise. The noise is modelled as additive white complex Gaussian with double-sided power spectral density $N_0/2$. It cannot be mitigated, but needs to be incorporated into the models for tracking filters. The second challenge is the presence of multipath components (MPCs) in the received signal. Due to the high bandwidth of UWB, individual reflected components from the environment appear as resolvable components in the received signal. Geometrically, the first signal component, which travelled along the line-of-sight (LOS), arrives with a delay of $\tau_{n,0}^{(j)}$ and a complex gain $\alpha_{n,0}^{(j)}$, with n denoting the timestep and j the anchor index. Reflected components, which take a geometrically longer path, arrive later with delay $\tau_{n,k}^{(j)}$ and a complex gain $\alpha_{n,k}^{(j)}$. For an unobstructed LOS between transmitter and receiver, $\alpha_{n,0}^{(j)}$ will be significantly larger than $\alpha_{n,k}^{(j)}$ for $k > 0$. In situations where the

LOS path is obstructed, or constructive interference happens between two MPCs, $\alpha_{n,k}^{(j)}$ can also be larger than $\alpha_{n,0}^{(j)}$, or $\alpha_{n,0}^{(j)}$ can drop close to zero, corresponding to a non-line-of-sight (NLOS) measurement. In these situations, a more complex algorithm is necessary to correctly estimate $\tau_{n,0}^{(j)}$, as simply detecting the peak of the signal will not result in an accurate estimate. If the LOS component and an MPC arrive in close proximity, the distance estimate can also be degraded or biased due an interference of the two components.

Another source of errors with ranging, especially for the keyless car access scenario investigated in this thesis, is the human body [8]. UWB pulses are attenuated strongly by the human body and radio waves bend around the body, causing either NLOS measurements or biased detections, and hence degrading the results.
Additional error sources in UWB ranging are posed by timing errors and clock drift between the two nodes, introducing errors on the hardware level [1].

The complex baseband signal model incorporates these two error sources. It is modelled for timestep n and anchor j in (2.1), an example of a received signal is shown in Figure 2.1.

$$r_n^{(j)}(t) = \alpha_{n,0}^{(j)} s(t - \tau_{n,0}^{(j)}) + \sum_{k=1}^{K_n^{(j)}} \alpha_{n,k}^{(j)} s(t - \tau_{n,k}^{(j)}) + w_n^{(j)}(t) \tag{2.1}$$

with the variables defined as:

n ... timestep
j ... anchor index
k ... multipath component index
$s(t-\tau)$ ... transmitted pulse shape
τ ... delay
α ... complex gain
$w(t)$ ... additive white complex Gaussian noise

The pulse shape $s(t)$ used by the NXP Ranger4 chip is a Root-Raised Cosine pulse, shown in (2.2). The pulse has a bandwidth of $B = 500\,\text{MHz}$, resulting in a sampling rate $T_s = 1\,\text{ns}$, and a roll-off factor $\beta = 0.5$.

$$s(t) = \frac{cos(\pi\beta\frac{t}{T_s})}{1 - (2\beta\frac{t}{T_s})} \frac{sin(\pi\frac{t}{T_s})}{\pi\frac{t}{T_s}} \tag{2.2}$$

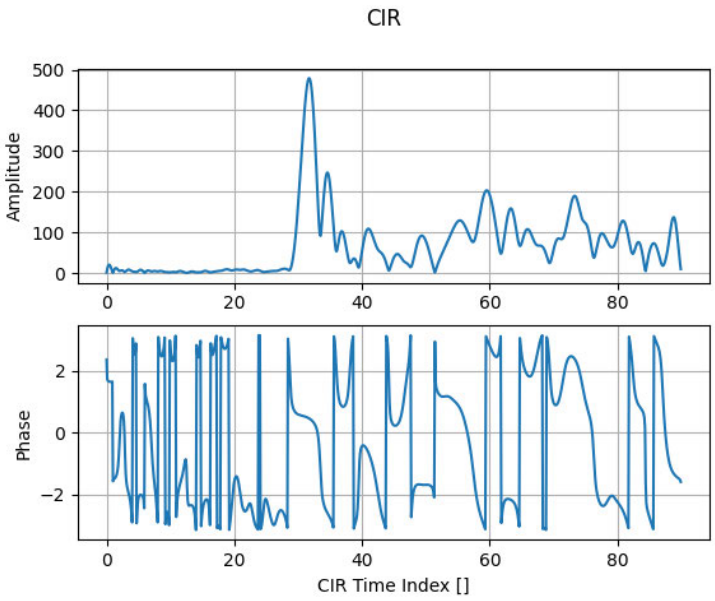


Figure 2.1: Supersampled plot of a CIR.

Due to hardware effects and filtering, the actual pulses differ from the analytic Root-Raised Cosine pulse. The received pulse was extracted by averaging and oversampling a large set of received CIRs. This derived pulse shape improves matching of signal components in the received signal. The two pulse shapes are compared in Figure 2.2.

2.2 Line-Of-Sight Detection and Estimation

To estimate the delays and the complex gains of the signal components from the received signal, various methods can be used. The NXP Ranger4 chip utilizes a threshold detector to detect the reception of a signal, and then uses a Search-Back Algorithm as presented in [9]. After estimating the CIR, the noise power and a detection threshold are determined. The strongest path is found as a maximum of the CIR, and going back from this index, the first index exceeding the detection threshold is found. The corresponding peak to the index is found, and then a timestamp can be accurately estimated by interpolation of the received signal and matching of the pulse shape.

Another option is to jointly estimate the parameters $\alpha_{n,k}^{(j)}$ and $\tau_{n,k}^{(j)}$. A channel estimation and detection algorithm, approximating a joint maximum likelihood estimator for these parameters, will be discussed in Chapter 3.

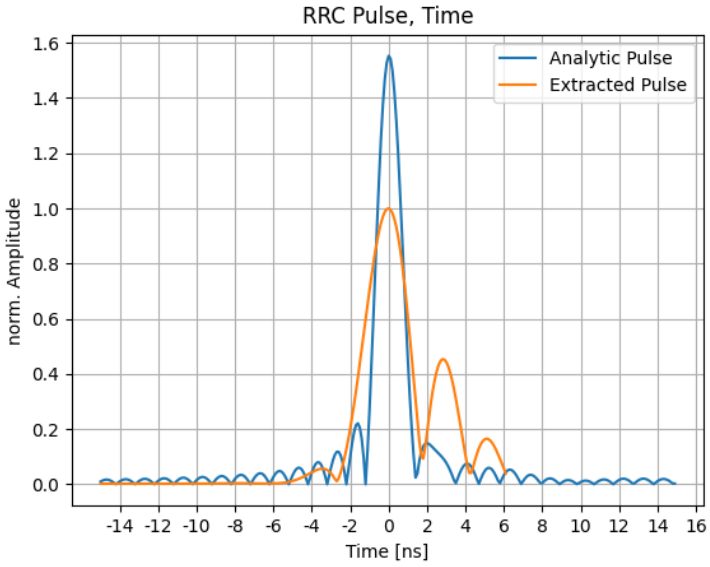


Figure 2.2: Analytic vs. actual Root Raised Cosine pulses.

2.3 Double-Sided Two-Way Ranging

To measure the ToF between two nodes, at least two transmissions are necessary to be able to measure a time difference. The NXP Ranger4 node uses a double-sided two-way ranging protocol to achieve a precise ranging estimate, illustrated in Figure 2.3. Node (a) sends a message to Node (b), which takes time D_b to reply. Node (a) measures the time R_a between transmitting and receiving the messages. Node (a) also answer after time D_a, with node (b) also measuring the time R_b between transmission and reception of the messages. The ToF τ can be calculated using (2.3).
While the ToF could be calculated from R_a and D_b only, the double-sided two-way ranging procedure cancels linear clock drift errors between the two nodes, reducing error introduced by mismatching clock frequencies. The conversion from time to distance is easily done by multiplying the speed of light c with the delay time τ. [10]

$$\tau = \frac{R_a R_b - D_a D_b}{R_a + R_b + D_a + D_b} \tag{2.3}$$

$$d = c\tau \tag{2.4}$$

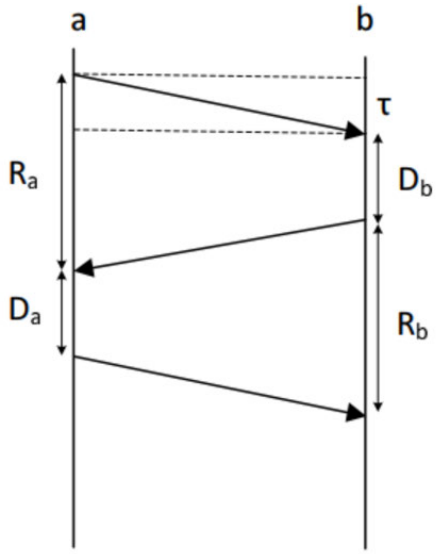


Figure 2.3: Double-sided two-way Ranging sequence. *NXP Semiconductors, NCJ29D5 Ultra-Wideband ICs for Automotive Applications, 2021.*

2.4 Distance Cramer-Rao Lower Bound

A theoretical limit for estimating distances from pulse transmissions in an AWGN environment is described by the Cramer-Rao Lower Bound (CRLB). This bound defines a lower limit to the mean-square error (MSE) achievable for a noisy estimation. It is based on the inverse Fisher Information Matrix, details can be found in [1]. Assuming only AWGN as error source, the lower bound for the distance estimation σ_d is defined by the Rott-Mean-Square (RMS) bandwidth $\hat{B}$ and the SNR, with bandwidth B and roll-off factor β:

$$\sigma_d \geq \frac{c}{2\sqrt{2}\pi\sqrt{SNR}\hat{B}} \tag{2.5}$$

$$\hat{B} = \sqrt{B^2(\frac{1}{12} + \frac{\pi^2 - 8}{4\pi^2}\beta^2)} \tag{2.6}$$

For 20 dB SNR, $B = 500\,\text{MHz}$ and a roll-off factor $\beta = 0.5$, the CRLB results to:

$$\sigma_d \geq \frac{3 \times 10^8\,\text{m}\,\text{s}^{-1}}{2\sqrt{2}\pi\sqrt{20\,\text{dB}} \cdot 154\,\text{MHz}} \tag{2.7}$$

$$\sigma_d \geq 0.022\,\text{m} \tag{2.8}$$

In real-world applications, as with the NXP Ranger4 chip, AWGN is not the main source of error, resulting in the CRLB being significantly lower compared to achieved errors. As

mentioned above, clock drift and multipath are potential error sources, neither of which is taken into account in the model for this CRLB estimate [1].

3 Positioning and Tracking Algorithms

Following ranging in Chapter 2, this chapter will discuss algorithms for positioning and tracking the agent. The simplest setup for positioning in a 2-D plane is comprised of a minimum of three anchor nodes and one agent node. Through the three distances between the agent and the anchors, and the anchor positions, the agent's position is unambiguously defined:

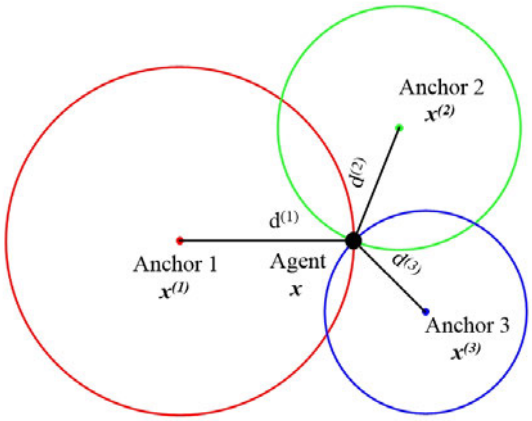


Figure 3.1: Trilateration

3.1 Positioning with a Single Signal Snapshot

The distance $d^{(j)}$ between the agent and the j-th anchor is defined as the norm of the vector between the points $\mathbf{x}$ and $\mathbf{x}^{(j)}$, with the measurement function h:

$$d^{(j)} = h^{(j)}(\mathbf{x}) = \|\mathbf{x}^{(j)} - \mathbf{x}\| = \sqrt{(x^{(j)} - x)^2 + (y^{(j)} - y)^2} \tag{3.1}$$

The measurement equation follows, defining $\hat{d}$ as the measured distance, taking into account the inaccuracies of the measurement system and various effects and modelling them as an additive Gaussian distance error, with $\mathcal{N}$ denoting a Gaussian distribution, as:

$$\hat{d}^{(j)} = h^{(j)}(\mathbf{x}) + w^{(j)}, \qquad w^{(j)} \sim \mathcal{N}(0, {\sigma^{(j)}}^2) \tag{3.2}$$

The noise components are assumed to be independent for each distance measurement. In vector notation, with measurement vector $\mathbf{z}$ and the measurement covariance matrix $\mathbf{R}$:

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{w} \qquad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{3.3}$$

$$\mathbf{R} = \begin{bmatrix} \sigma^{(1)2} & 0 & 0 \\ 0 & \sigma^{(2)2} & 0 \\ 0 & 0 & \sigma^{(3)2} \end{bmatrix} \tag{3.4}$$

The probability $p(\mathbf{z}|\mathbf{x})$ describes the probability density function (PDF) which arises from the measurement. Due to the noise, the distances $d^{(j)}$ are modelled as Gaussian random variables (RVs), with $\mu = d^{(j)}$ and ${\sigma^{(j)}}^2$. With the prior probability $p(\mathbf{x})$, which entails all knowledge about the possible values of $\mathbf{x}$, the posterior probability $p(\mathbf{x}|\mathbf{z})$ is calculated:

$$p(\mathbf{x}|\mathbf{z}) = p(\mathbf{z}|\mathbf{x})p(\mathbf{x}) \tag{3.5}$$

By maximizing this likelihood with regards to $\mathbf{x}$, the likeliest estimate $\hat{\mathbf{x}}$ will be found. The PDF of the posterior probability $p(\mathbf{x}|\mathbf{z})$ will not be Gaussian or independent, because of the nonlinear mapping between the measurement and position.

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{z}) \tag{3.6}$$

A simple approach to solve the maximization problem would be a grid search, calculating the distance vector $\tilde{\mathbf{d}}$ for all possible agent positions $\tilde{\mathbf{x}}$, and then minimizing the Euclidean distance between $\tilde{\mathbf{d}}$ and $\mathbf{z}$ to find the most likely estimate $\hat{\mathbf{x}}$. This approach is limited due to high computational cost and limited accuracy and range [1]

3.2 Tracking Model

To incorporate prior knowledge in form of a previous position and a physical model of the agent's movement, a recursive update of the position estimate can be used in form of a tracking filter. This physical model of the movement is given in form of a discrete-time state-space model, with the transition matrix $\mathbf{A}$, state vector $\mathbf{x_n}$ and sampling time T_s.

Generally, due to the lack of detailed information about the movement, a constant velocity state-space model is used for the tracking filter, as found in [11]:

$$\mathbf{x_n} = \mathbf{A}\mathbf{x_{n-1}} + \mathbf{v_{n-1}} \tag{3.7}$$

$$\mathbf{x_n} = \begin{bmatrix} x_n \\ y_n \\ \dot{x_n} \\ \dot{y_n} \end{bmatrix} \tag{3.8}$$

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & T_s & 0 \\ 0 & 1 & 0 & T_s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

$$\mathbf{v_{n-1}} = \mathcal{N}(0, \mathbf{Q_{n-1}}) \tag{3.10}$$

This state vector evolves as a Markov sequence, meaning that the conditional distribution depending on the previous steps $p(\mathbf{x_n}|\mathbf{x_{n-1}}, \mathbf{x_{n-2}}, ...\mathbf{x_1}, \mathbf{x_0})$ is entirely described by the distribution assuming only the $(n-1)-th$ timestep $p(\mathbf{x_n}|\mathbf{x_{n-1}})$. By combining this Markov sequence with the state-space model, a recursive update of the position estimate can be formulated. The state-space model incorporates the process noise $\mathbf{v_n}$. This noise describes the inaccuracy of the physical model compared to the actual movement of the agent.

At each timestep, the marginal prior distribution $p(\mathbf{x}_n|\mathbf{x}_{n-1})$ is calculated, which can be interpreted as a prediction of the state vector with regard to the previous state $\mathbf{x_{n-1}}$ and all previous measurements $\mathbf{z_{0:n-1}}$ by applying the Chapman-Kolmogorov equation. [12]

$$p(\mathbf{x_n}|\mathbf{z_{0:n-1}}) = \int p(\mathbf{x_n}|\mathbf{x_{n-1}})p(\mathbf{x_{n-1}}|\mathbf{z_{0:n-1}})d\mathbf{x_{n-1}} \tag{3.11}$$

To get the posterior distribution $p(\mathbf{x_n}|\mathbf{z_{0:n}})$, the prediction and the measurement are combined with Bayes law. The resulting probability is a weighted combination of the prediction and the measurement.

$$p(\mathbf{x_n}|\mathbf{z_{0:n}}) = \frac{p(\mathbf{z_n}|\mathbf{x_n})p(\mathbf{x_{n-1}}|\mathbf{z_{0:n-1}})}{p(\mathbf{z_n}|\mathbf{z_{0:n-1}})} \tag{3.12}$$

3.3 Extended Kalman Filter

This iterative prediction and update sequence is the basis for the Kalman filter. The derivation of the Kalman filter can be found in [11], [13]. Because of the Gaussian assumptions, the trilateration problem needs to be linearised around the operating point to allow the posterior distribution $p(\mathbf{x_n}|\mathbf{z_{0:n}})$ to be approximated by a Gaussian distribution. The operating point for the linearisation is the predicted state vector. The approximation are found in [12],(22)-(24), with the estimate covariance matrix $\mathbf{P}$, as:

$$\text{Prior:} \quad p(\mathbf{x}_n|\mathbf{z}_{0:n-1}) \approx \mathcal{N}(\hat{\mathbf{x}}_{n|n-1}, \mathbf{P}_{n|n-1}) \tag{3.13}$$

$$\text{Posterior:} \quad p(\mathbf{x}_n|\mathbf{z}_{0:n}) \approx \mathcal{N}(\hat{\mathbf{x}}_{n|n}, \mathbf{P}_{n|n}) \tag{3.14}$$

$$\text{LHF:} \quad p(\mathbf{z}_n|\mathbf{x}_n) \approx \mathcal{N}(\mathbf{z}_n, \mathbf{R}_n) \tag{3.15}$$

The linearisation is done using the Jacobian matrix, the derivative of the measurement function $\mathbf{h}(\mathbf{x})$ at the predicted state $\hat{\mathbf{x}}_{n|n-1}$. This linearised variant is called the Extended Kalman filter (EKF). The linearisation is formulated in (3.16), as found in [12] (30).

$$\mathbf{H}_n = \left.\frac{\partial \mathbf{h}(\mathbf{x})}{\partial x}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{n|n-1}} \tag{3.16}$$

Additionally, the process noise matrix $\mathbf{Q}_n$ is used, shown in (3.10), describing the randomness in the state evolution. Since the constant-velocity model does not depict the actual movement of the agent very well, the uncertainty of this mismatch needs to be accounted for. The process noise is an important parameter, essentially weighting the prediction against the measurement by widening or narrowing the predicted covariance. The second relevant matrix is $\mathbf{R}_n$, the measurement covariance matrix, shown in (3.4). Because the three measurements are assumed to be independently distributed, $\mathbf{R}_n$ is a diagonal matrix with the diagonal elements being the variance of the single measurements. The variance for the individual measurements can be fixed to a heuristically defined value, or adapted for each timestep based on measurement quality. Assuming the signal model in Section 2.1 is correct and the signal components can be resolved by the CEDA or the LOS detection algorithm, the elements of $\mathbf{R}_n$ can be calculated via the distance CRLB from measurement SNR, as in (2.6).

The filter equations are entirely based on matrix/vector algebra:

$$\text{Prediction:} \tag{3.17}$$

$$\hat{\mathbf{x}}_{n|n-1} = \mathbf{A}\hat{\mathbf{x}}_{n-1|n-1} \tag{3.18}$$

$$\mathbf{P}_{n|n-1} = \mathbf{A}\mathbf{P}_{n-1|n-1}\mathbf{A}^T + \mathbf{Q}_{n-1} \tag{3.19}$$

$$\text{Update:} \tag{3.20}$$

$$\mathbf{K}_n = \mathbf{P}_{n|n-1}\mathbf{H}^T(\mathbf{H}\mathbf{P}_{n|n-1}\mathbf{H}^T + \mathbf{R}_n) \tag{3.21}$$

$$\hat{\mathbf{x}}_{n|n} = \hat{\mathbf{x}}_{n|n-1} + \mathbf{K}_n(\mathbf{z}_n - \mathbf{h}(\hat{\mathbf{x}}_{n|n-1})) \tag{3.22}$$

$$\mathbf{P}_{n|n} = (\mathbf{I} - \mathbf{K}_n\mathbf{H}_n)\mathbf{P}_{n|n-1} \tag{3.23}$$

The signal flow for the EKF is shown in Figure 3.2. The EKF directly uses the distances estimates from the Ranger4 Chip, together with the SNR estimate.

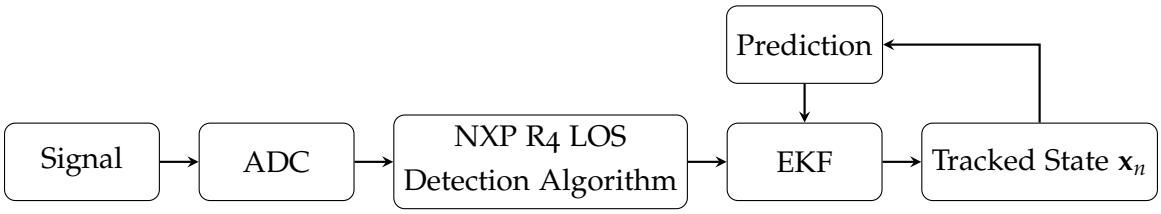


Figure 3.2: Factor Graph for the PDA filter.

3.3.1 Discussion

For linear Gaussian estimation problems, the Kalman filter is the optimal tracking solution [13]. In our case, with the resulting posterior probability not following a Gaussian distribution, it is not an optimal solution but only an approximation. Other tracking filters can estimate the resulting posterior probability more accurately. Because of the very low amount of operations necessary, this is by far the fastest filter presented here. However, the EKF is prone to NLOS measurements, instantly losing its track upon receiving wrong measurements.

3.4 Particle Filter

The particle filter, more specifically the Sampling Importance Resampling filter (SIR) is a sequential Monte Carlo approach. Instead of mean and covariance, N particles, each a sample of the state vector $\mathbf{x}_n$, are assigned a weight and used to represent the current

posterior probability. With large N, the characterization of the likelihood approaches an equivalent representation to a functional description of the likelihood. The implementation is based on [12]. The particles are given as $\{\mathbf{x}_n^i, w_n^i\}_{i=0}^N$, where the particle i represents a support point $\mathbf{x}_n^i$ with the weight w_n^i, to approximate the posterior PDF $p(\mathbf{x}_n|\mathbf{z}_{0:n})$ for time n. The weights are normalized to $\sum_i w_n^i = 1$. The posterior PDF can then be approximated by:

$$p(\mathbf{x}_n|\mathbf{z}_{0:n-1}) \approx \sum_{i=1}^{N} w_n^i \delta(\mathbf{x}_n - \mathbf{x}_n^i) \tag{3.24}$$

Again, the same sequential approach will be applied. Starting from a state $\{\mathbf{x}_{n-1}^i, w_{n-1}^1\}_{i=1}^N$, the particles are propagated through the state-space equations, with a sample of the process noise $\mathbf{v}_{\mathbf{n-1}}^{\mathbf{i}} \sim \mathcal{N}(0, \mathbf{Q}_n)$ being drawn and added to particle i. The process noise takes the dynamic model mismatch in form of changes of velocity into account by adding random acceleration to the particles. This sampling from the importance density $p(\mathbf{x}_n|\mathbf{x}_{n-1}^i)$ is similar to the prediction step of the Kalman filter. For particle i, the prediction works as follows:

$$\mathbf{x}_{n|n-1}^i = \mathbf{A}\mathbf{x}_{n-1|n-1}^i + \mathbf{B}\mathbf{v}_n^i \tag{3.25}$$

$$\mathbf{B} = \begin{bmatrix} \frac{T_s^2}{2} & 0 & T_s & 0 \\ 0 & \frac{T_s^2}{2} & 0 & T_s \end{bmatrix}$$

After the prediction step, the weights of the particles are updated:

$$w_n^i = p(\mathbf{z}_n|\mathbf{x}_{n|n-1}^i) \tag{3.26}$$

$$w_n^i = \frac{w_n^i}{\sum_{i=1}^N w_n^i} \tag{3.27}$$

While the measurement likelihood $p(\mathbf{z}_n|\mathbf{x}_n)$ is not Gaussian and therefore not trivial to evaluate, the measurement function can be applied to the particles, yielding $\hat{\mathbf{z}}_n^i = \mathbf{h}(\mathbf{x}_{n|n-1}^i)$. Using this particle measurement $\hat{\mathbf{z}}_n^i$, the distribution becomes Gaussian with the mean being the measurement $\mathbf{z}_n$, and the covariance being the measurement matrix $\mathbf{R}_n$. As with the EKF, the measurement matrix $\mathbf{R}_n$ can either be fixed, or updated based on SNR.

$$p(\mathbf{z}_n|\hat{\mathbf{z}}_n^i) = \mathcal{N}(\hat{\mathbf{z}}_n^i|\mathbf{z}_n, \mathbf{R}_n) \tag{3.28}$$

After the weight updates, the particles are resampled according to their weight to avoid the particle degeneracy effect, i.e. particles drifting apart. The resampling concentrates the particles by drawing particles with replacement from the probability mass function (PMF) represented by the weights $\{w_n^i\}_{i=1}^I$.

$$\mathbf{x}_{n|n}^i = resample(\mathbf{x}_{n|n-1}^i, w_n^i) \tag{3.29}$$

Algorithm 1 shows the resampling procedure used in the presented work, also found in [12]. "Systematic Resampling" is a fast and valid approximation of the optimum resampling procedure.

Algorithm 1 Systematic Resampling

1: Initialize CDF: $c_1 = 0$
2: **for** $i = 2 : N$ **do**
3: $\quad c_i = c_{i-1} + w_n^i$
4: **end for**
5: Start at bottom of CDF: $i = 1$
6: Draw a starting point: $u_1 \sim \mathcal{U}[0\ \frac{1}{N}]$
7: **for** $j = 1 : N$ **do**
8: $\quad$ Move along CDF: $u_j = u_1 + \frac{1}{N}(j-1)$
9: $\quad$ **while** $u_j > c_i$ **do**
10: $\quad\quad i = i + 1$
11: $\quad$ **end while**
12: $\quad$ Assign Sample: $x_n^j = x_n^i$
13: $\quad$ Assign weight: $w_n^j = \frac{1}{N}$
14: **end for**

After resampling, the final estimate $\hat{\mathbf{x}}_\mathbf{n}$ is calculated by finding the mean of the posterior PDF, which means taking the average of the particles. The signal flowchart shown in Figure 3.3 is the same as for the EKF, with the particle filter also using the Ranger4 distance estimate.

$$\hat{\mathbf{x}}_n = \frac{\sum_{j=1}^{N} \mathbf{x}_{n|n}^j}{N} \tag{3.30}$$

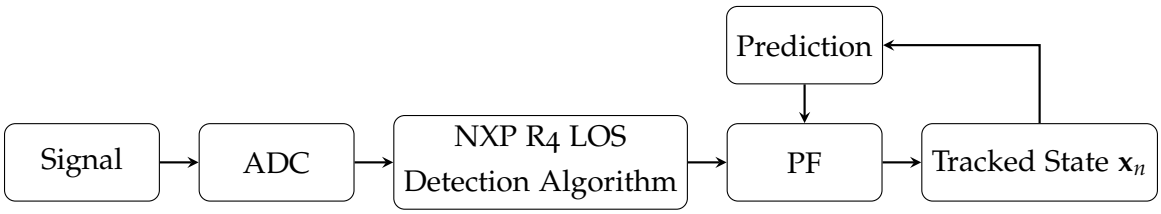


Figure 3.3: Factor Graph for the PDA filter.

3.4.1 Discussion

The particle filter requires propagation of all N particles through the state-space model, $2 * N$ samples drawn from the process noise distribution (independent samples from $\mathbf{v} = [v_x, v_y]^T$), and N evaluations of the measurement likelihood as well as resampling. It is, therefore, much more computationally costly, but approximates the posterior likelihood precisely (for $N \to \infty$ particles), as it represents a non-Gaussian distribution.

3.5 Probabilistic Data Association Filter

The Probabilistic Data Association (PDA) filter is used when multiple measurements per anchor are available, compared to the particle filter and the Extended Kalman filter, which both only use one measurement per anchor. The measurements used in the presented work originate from a channel estimation and detection algorithm (CEDA), as described in the following section. The detection probability (P_D) of the LOS component can either be fixed, or tracked to dynamically adjust to NLOS situations, as they do not occur randomly when there are obstructions. We use a special variant of the PF-based multi-sensor PDA filter, as proposed in [14], with the amplitude tracking neglected to reduce the feature space, and allow for a lower number of particles and therefore a lower filter runtime.

3.5.1 Channel Estimation and Detection Algorithm

The CEDA aims to isolate the individual signal components from the received signal and estimate the components' amplitudes $\alpha_{n,k}$ and delays $\tau_{n,k}$. The CEDA was implemented as in [14].

$$r_n^{(j)}(t) = \alpha_{n,0}^{(j)} s(t - \tau_{n,0}^{(j)}) + \sum_{k=1}^{K_n^{(j)}} \alpha_{n,k}^{(j)} s(t - \tau_{n,k}^{(j)}) + w_n^{(j)}(t) \tag{3.31}$$

After sampling the signal model in (3.31), a discrete-time specular signal vector can be defined as:

$$\mathbf{r}_n^{(j)} = \mathbf{S}(\boldsymbol{\tau}_n^{(j)})\boldsymbol{\alpha}_n^{(j)} + \mathbf{w}_n^{(j)} \tag{3.32}$$

To estimate the complex amplitude vector $\boldsymbol{\alpha}_n^{(j)} = [\alpha_{n,0}^{(j)} ... \alpha_{n,K_n^{(j)}}^{(j)}]$ and the delay vector $\boldsymbol{\tau}_n^{(j)} = [\tau_{n,0}^{(j)} ... \tau_{n,K_n^{(j)}}^{(j)}]$, a maximum likelihood approach can be formulated, resulting in the following set of equations. The indices are omitted for simplicity, and $\hat{\boldsymbol{\alpha}}, \hat{\tau}$ denote the estimations.

$$\hat{\boldsymbol{\alpha}}(\boldsymbol{\tau}) = (\mathbf{S}(\boldsymbol{\tau})^H \mathbf{S}(\boldsymbol{\tau}))^{-1} \mathbf{S}(\boldsymbol{\tau})^H \mathbf{r} \tag{3.33}$$

$$\hat{\tau}_k = \arg\max_{\tau_k} \frac{|\mathbf{r}_{res}^H \mathbf{s}(\tau_k)|^2}{||\mathbf{s}(\tau_k)||^2} \tag{3.34}$$

$$\mathbf{r}_{res} = \mathbf{r} - \mathbf{S}(\hat{\boldsymbol{\tau}}_{k-1}) \hat{\boldsymbol{\alpha}}(\hat{\boldsymbol{\tau}}_{k-1}) \tag{3.35}$$

$$\hat{\sigma}^2 = \frac{1}{N-1} ||\mathbf{r}_{res}||^2 \tag{3.36}$$

$$\hat{u} = \frac{|\hat{\alpha}|^2}{\hat{\sigma}^2} \tag{3.37}$$

These equations can be used in an iterative way to search and subtract the individual components from the signal, as shown in algorithm 2. It should be noted that this specific CEDA assumes the τ_m to be uncorrelated and estimates them independently, while the gain α_m is estimated jointly for each iteration. This can cause small errors for cases where two signal components arrive in close vicinity (i.e. $\tau_m - \tau_{m'}$ is small).

Algorithm 2 Snapshot-based CEDA

Initialization:
$m = 0$, $\hat{\boldsymbol{\tau}}_0 = []$, $\hat{\mathbf{u}}_0 = []$
while $\hat{u}_m > \gamma$ **do**
    $m++$
    if $m > 0$ **then**
        compute $\mathbf{r}_{res} = \mathbf{r} - \mathbf{S}(\hat{\boldsymbol{\tau}}_{m-1}) \hat{\boldsymbol{\alpha}}(\hat{\boldsymbol{\tau}}_{k-1})$
    end if
    add component $\hat{\tau}_m = \arg\max_{\tau_m} \frac{|\mathbf{r}_{res}^H \mathbf{s}(\tau_m)|^2}{||\mathbf{s}(\tau_m)||^2}$
    $\hat{\boldsymbol{\tau}}_m = [\hat{\boldsymbol{\tau}}_{m-1}; \hat{\tau}_m]$
    compute $\hat{\sigma}^2 = \frac{1}{N-1} ||\mathbf{r}_{res}||^2$
    compute $\hat{\boldsymbol{\alpha}}(\boldsymbol{\tau}) = (\mathbf{S}(\boldsymbol{\tau})^H \mathbf{S}(\boldsymbol{\tau}))^{-1} \mathbf{S}(\boldsymbol{\tau})^H \mathbf{r}$
    compute $\hat{u}_m = \frac{|\hat{\alpha}_m|^2}{\hat{\sigma}^2}$
    $\hat{\mathbf{u}}_m = [\hat{\mathbf{u}}_{m-1}; \hat{u}_m]$
end while

By using the complex gain $\hat{\alpha}_m$ and the estimated noise variance, the SNR $\hat{u}_m$ is calculated for each measurement. The distance standard deviation $\hat{\boldsymbol{\sigma}}_n^{(j)} = [\hat{\sigma}_{n,0}^{(j)}, ..., \hat{\sigma}_{n,M}^{(j)}]$ is calculated using (2.6). The vector $\hat{\boldsymbol{\tau}}_n^{(j)}$ can easily be transformed to the distance vector $\hat{\mathbf{d}}_n^{(j)} = [\hat{d}_{n,0}^{(j)}, ..., \hat{d}_{n,M}^{(j)}]$

using (2.4). The result is a combined measurement vector $\hat{\mathbf{z}}_n^{(j)} = [\hat{\mathbf{d}}_n^{(j)}, \hat{\sigma}_n^{(j)}]$ which is used by the PDA filter as input.

3.5.2 Data Association Model

For every timestep n and anchor j, the components of the measurement vector $\hat{\mathbf{z}}_n^{(j)}$ are subject to data association uncertainty. Any of the $M_n^{(j)}$ measurements could originate from the LOS component, an NLOS component or it is a clutter measurement without a physical source. It is also uncertain whether the LOS component is present for this instance. The model for this filter implementation only distinguishes between LOS and NLOS measurements. An association variable is defined:

$$a_n^{(j)} = \begin{cases} m \in \mathcal{M}_n^{(j)}, & \hat{z}_{n,m}^{(j)} \text{ is the LOS measurement in } \hat{\mathbf{z}}_n^{(j)} \\ 0, & \text{no LOS measurement in } \hat{\mathbf{z}}_n^{(j)} \end{cases} \tag{3.38}$$

Together with the probability of a LOS measurement $p_{D,n}^{(j)}(q_n^{(j)})$, which will be discussed shortly, the probability mass function of $a_n^{(j)}$ and $M_n^{(j)}$ is proportional to:

$$h_m(a_n^{(j)}, M_n^{(j)}; q_n^{(j)}) = \begin{cases} \frac{p_{D,n}^{(j)}(q_n^{(j)})}{M_n^{(j)}}, & a_n^{(j)} \in \mathcal{M}_n^{(j)} \\ 1 - p_{D,n}^{(j)}(q_n^{(j)}), & a_n^{(j)} = 0 \end{cases} \tag{3.39}$$

The following likelihoods and the filter do not distinguish the components per se, but weight each measurement into the final posterior likelihood depending on distance $\hat{d}_{n,m}^{(j)}$ and standard deviation $\hat{\sigma}_{n,m}^{(j)}$. That way, all possible information of the received measurements is used for the final estimation of $\mathbf{x}_n$.

3.5.3 Distance Likelihood

The distance likelihood function is also defined for the LOS event and the NLOS event. The LOS likelihood function for measurement $\hat{\mathbf{z}} = [\hat{d}_{n,m}^{(j)}, \hat{\sigma}_{n,m}^{(j)}]$ is defined as:

$$p_{LOS}(\hat{\mathbf{z}}_{n,m}^{(j)} | \mathbf{x}_n) = \mathcal{N}(\hat{d}_{n,m}^{(j)}; h^{(j)}(\mathbf{x}_n), \hat{\sigma}_{n,m}^{(j)}) \tag{3.40}$$

and the NLOS likelihood function as a uniform distribution $\mathcal{U}(0, d_{max})$, since all measurements except the LOS measurement are modelled as clutter.

$$p_{NLOS}(\hat{d}_{n,m}^{(j)}) = \mathcal{U}(0, d_{max}) \tag{3.41}$$

The overall distance likelihood function follows as:

$$p(\hat{z}_{n,m}^{(j)}|\mathbf{x}_n, a_n^{(j)}) = \begin{cases} p_{LOS}(\hat{\mathbf{z}}|\mathbf{x}_n), & a_n^{(j)} = m \\ p_{NLOS}(\hat{d}_{n,m}^{(j)}), & a_n^{(j)} \neq m \end{cases} \tag{3.42}$$

3.5.4 LOS Existence Model

The PMF $p_{D,n}^{(j)}(q_n^{(j)})$ describes the PMF of a LOS measurement for timestep n at anchor j. It is modelled as:

$$p_{D,n}^{(j)}(q_n^{(j)}) = q_n^{(j)} \tag{3.43}$$

For simplicity, it is assumed that a LOS component only exists if it is detectable by the CEDA. The random variable $q_n^{(j)}$, which is tracked for each anchor individually, describes the probability of a LOS detection. It is tracked as a discrete random variable in form of a first-order Markov process, with values from the set $\mathcal{Q} = \{v_1 ... v_Q\}, v_i \in (0, 1]$ and transition matrix $[\mathbf{Q}]_{i,k} = \Psi(q_n^{(j)} = v_i | q_{n-1}^{(j)} = v_k)$.

3.5.5 Joint Measurement Likelihood

The joint measurement likelihood can be written as:

$$p(\hat{\mathbf{z}}_n^{(j)}|\mathbf{x}_n, a_n^{(j)}) = \prod_{m=1}^{M_n^{(j)}} p(\hat{\mathbf{z}}_{n,m}^{(j)}|\mathbf{x}_n, a_n^{(j)}) \tag{3.44}$$

By neglecting constant terms, a pseudo likelihood function can be defined as:

$$g(\hat{\mathbf{z}}_n^{(j)}; \mathbf{x}_n, a_n^{(j)}) = \prod_{m=1}^{M_n^{(j)}} \times \begin{cases} p_{NLOS}(\hat{d}_{n,m}^{(j)}), & a_n^{(j)} = 0 \\ p_{LOS}(\hat{\mathbf{z}}), & a_n^{(j)} \in M_n^{(j)} \end{cases} \tag{3.45}$$

The joint posterior likelihood for all anchors $p(\mathbf{x}_{0...n}, \mathbf{a}_{0...n}, \mathbf{q}_{0...n}|\mathbf{z}_{0...n})$ (indices will be omitted for simplicity) can be derived up to a constant factor to be as follows. $\Upsilon(\mathbf{x}_n|x_{n-1})$ denotes the state transition PDF of the dynamic model for the agent, similar to Kalman and particle filter.

$$\begin{aligned} p(\mathbf{x}_{0...n}, \mathbf{a}_{0...n}, \mathbf{q}_{0...n}|\mathbf{z}) &\propto p(\mathbf{z}|\mathbf{x}, \mathbf{a}, \mathbf{q}) p(\mathbf{x}, \mathbf{a}, \mathbf{u}, \mathbf{q}) \\ &= p(\mathbf{z}|\mathbf{x}, \mathbf{a}, \mathbf{q}) p(\mathbf{a}|\mathbf{q}) p(\mathbf{x}) p(\mathbf{q}) \\ &\propto p(\mathbf{x}_0) \prod_{j=1}^{J} p(q_o^{(j)}) \prod_{n'=1}^{n} \Upsilon(\mathbf{x}_{n'}|x_{n'-1}) \Psi(q_{n'}^{(j)}|q_{n'-1}^{(j)}) ... \\ &\quad ... h_m(a_{n'}^{(j)}; q_{n'}^{(j)}) g(\hat{\mathbf{z}}_{n'}^{(j)}; \mathbf{x}_{n'}, a_{n'}^{(j)}) \end{aligned} \tag{3.46}$$

3.5.6 Posterior Likelihood

To derive an estimate of the agent position, the minimum mean square error (MMSE) estimate needs to be calculated for $\mathbf{x}_n$ [13].

$$\hat{\mathbf{x}}_n^{MMSE} = \int \hat{\mathbf{x}}_n p(\mathbf{x}_n|\mathbf{z}) d\mathbf{x}_n \tag{3.47}$$

These marginal posterior functions can easily be calculated by executing the Sum-Product algorithm, presented in [15], [16]. On the factor graph, messages are passed between parts of the filter algorithm, representing the joint posterior in (3.46). For this application, the graph passes messages only forward in time, resulting in exact results for the posterior distributions. The concepts and derivation of message-passing algorithms for tracking can be found in [15]; a similar filter is also presented in [16]. The graph for timesteps $(n-1)$, (n) and anchor (j), is shown in Figure 3.4.

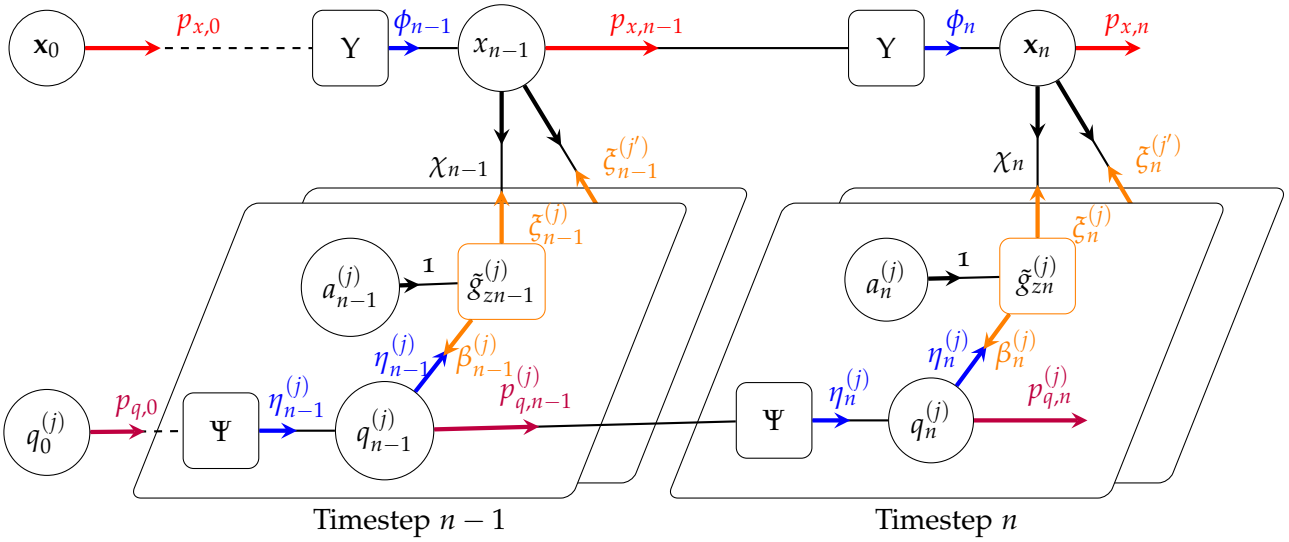


Figure 3.4: Factor Graph for the PDA filter.

As with Kalman and Particle filter, the PDA filter consists of a prediction step and an update step. For the PDA, the prediction step consists of the two prior likelihoods ϕ_n and $\eta_n^{(j)}$:

$$\phi_n(\mathbf{x}_n) = \int \Upsilon(\mathbf{x}_n|\mathbf{x}_{n-1}) p_{x,n-1}(\mathbf{x}_{n-1}) d\mathbf{x}_{n-1} \tag{3.48}$$

$$\eta_n^{(j)}(q_n^{(j)}) = \sum_{q_{n-1}^{(j)}=1}^{Q} \Psi(q_n^{(j)}|q_{n-1}^{(j)}) p_{q,n-1}^{(j)}(q_{n-1}^{(j)}) \tag{3.49}$$

The first equation, ϕ_n, is the same propagation through the state-space model as for the other filters. The prediction for $q_n^{(j)}$ is the multiplication with the transition matrix of the Markov sequence model $\mathbf{Q}$.

The measurement update messages $\eta_n^{(j)}$, $\xi_n^{(j)}$ and $\beta_n^{(j)}$ are based on the combined measurement function $\tilde{g}_{zn}^{(j)}(a_{n'}^{(j)}, q_{n'}^{(j)}, \hat{\mathbf{z}}_{n'}^{(j)}, \mathbf{x}_{n'}) = h_m(a_{n'}^{(j)}; q_{n'}^{(j)}) g(\hat{\mathbf{z}}_{n'}^{(j)}; \mathbf{x}_{n'}, a_{n'}^{(j)})$:

$$\eta_n^{(j)}(\mathbf{x}_n) = \sum_{q_n^{(j)}=1}^{Q} \eta_n^{(j)}(q_n^{(j)}) \sum_{a_n^{(j)}=1}^{M_n^{(j)}} \tilde{g}_{zn}^{(j)}(a_{n'}^{(j)}, q_{n'}^{(j)}, \hat{\mathbf{z}}_{n'}^{(j)}, \mathbf{x}_{n'}) \tag{3.50}$$

$$\xi_n^{(j)}(\mathbf{x}_n) = \phi_n(\mathbf{x}_n) \prod_{j'=1}^{J} \frac{\xi_{n-1}^{(j')}(\mathbf{x}_n)}{\xi_{n-1}^{(j)}(\mathbf{x}_n)} \tag{3.51}$$

$$\beta_n^{(j)}(q_n^{(j)}) = \int \chi_n^{(j)}(\mathbf{x}_n) \sum_{a_n^{(j)}=1}^{M_n^{(j)}} \tilde{g}_{zn}^{(j)}(a_{n'}^{(j)}, q_{n'}^{(j)}, \hat{\mathbf{z}}_{n'}^{(j)}, \mathbf{x}_{n'}) d\mathbf{x}_n \tag{3.52}$$

The final posterior likelihoods follow as:

$$p(\mathbf{x}_n|\mathbf{z}) \propto p_{xn} = \phi_n(\mathbf{x}_n) \prod_{j=1}^{J} \xi_{n-1}^{(j)}(\mathbf{x}_n) \tag{3.53}$$

$$p(q_n^{(j)}|\mathbf{z}) \propto p_{qn} = \eta_n^{(j)}(q_n^{(j)}) \times \beta_n^{(j)}(q_n^{(j)}) \tag{3.54}$$

As aforementioned, we use a particle-based implementation of the filter represented by (3.50)-(3.54), which means the posterior likelihood $p(\mathbf{x}_n|\mathbf{z})$ is approximated by particles and not analytically calculated. Details on the particle based implementations can be found in [16].

3.5.7 Filter Equations

Due to the particle approach, all integrals over $d\mathbf{x}_n$ become sums over the state $\{\mathbf{x}_n^i, w_n^i\}_{i=0}^N$. The filter can be implemented using the following set of numerical equations, starting

with particles $\{\mathbf{x}^i_{n-1|n-1}, w^i_{n-1}\}^N_{i=0}$, vector $\mathbf{q}^i_{n-1|n-1}$ as the anchor-wise weight vector for the discrete LOS detection probability and measurement vector $\hat{\mathbf{z}}^{(j)}_n = [\hat{\mathbf{d}}^{(j)}_n, \hat{\boldsymbol{\sigma}}^{(j)}_{n,m}]$:

Prediction:

$$\mathbf{x}^i_{n|n-1} = \mathbf{A}\mathbf{x}^i_{n-1|n-1} + \mathbf{B}\mathbf{v}^i_n \tag{3.55}$$

$$\mathbf{q}^{(j)}_{n|n-1} = \mathbf{Q}\mathbf{q}^{(j)}_{n-1|n-1} \tag{3.56}$$

Weight Update:

$$\tilde{w}^{m,i,(j)}_n = \mathcal{N}(\hat{d}^{(j)}_{n,m} | h^{(j)}(\mathbf{x}^i_{n|n-1}), \hat{\sigma}^{(j)}_{n,m}) \tag{3.57}$$

$$\tilde{w}^{q,i,(j)}_n = \sum_{m=1}^{M^{(j)}_n} \frac{d_{max}}{M^{(j)}_n} v_q \tilde{w}^{m,i,(j)}_{\mathbf{z}} + (1 - v_q) \tag{3.58}$$

$$w^i_n = \prod_{j=1}^{J} \sum_{q=0}^{Q} q^{v_q,(j)}_{n|n-1} \tilde{w}^{q,i,(j)}_n \tag{3.59}$$

Resampling, Estimate:

$$\mathbf{x}^i_{n|n} = resample(\mathbf{x}^i_{n|n-1}, w^i_n) \tag{3.60}$$

$$\hat{\mathbf{x}}_n = \frac{\sum_{j=1}^{N} \mathbf{x}^j_{n|n}}{N} \tag{3.61}$$

Update for q:

$$\mathbf{q}'^{(j)}_{n|n} = \mathbf{q}^{(j)}_{n|n-1} \sum_{m=1}^{M} \tilde{w}^{q,i,(j)}_n \tag{3.62}$$

$$\mathbf{q}^{(j)}_{n|n} = \frac{\mathbf{q}'^{(j)}_{n|n}}{\sum \mathbf{q}'^{(j)}_{n|n}} \tag{3.63}$$

A second filter variant with a fixed LOS detection probability P_D was also used as comparison to evaluate the benefit of the tracking. The fixed P_D variant is described in the following equations:

Prediction:

$$\mathbf{x}^i_{n|n-1} = \mathbf{A}\mathbf{x}^i_{n-1|n-1} + \mathbf{B}\mathbf{v}^i_n \tag{3.64}$$

Weight Update:

$$\tilde{w}^{m,i,(j)}_n = \mathcal{N}(\hat{d}^{(j)}_{n,m} | h^{(j)}(\mathbf{x}^i_{n|n-1}), \hat{\sigma}^{(j)}_{n,m}) \tag{3.65}$$

$$w^i_n = \prod_{j=1}^{J} \sum_{m=1}^{M^{(j)}_n} \frac{d_{max}}{M^{(j)}_n} P_D \tilde{w}^{m,i,(j)}_n + (1 - P_D) \tag{3.66}$$

Resampling, Estimate:

$$\mathbf{x}^i_{n|n} = resample(\mathbf{x}^i_{n|n-1}, w^i_n) \tag{3.67}$$

$$\hat{\mathbf{x}}_n = \frac{\sum_{j=1}^{N} \mathbf{x}^j_{n|n}}{N} \tag{3.68}$$

$$\hat{\mathbf{x}}_\mathbf{n} = \frac{\sum_{j=1}^{N} \mathbf{x}^j_{n|n}}{N} \tag{3.69}$$

The signal flowchart is shown in Figure 3.5. Instead of the Ranger4 distance estimate, the captured CIR is used as input for the CEDA.

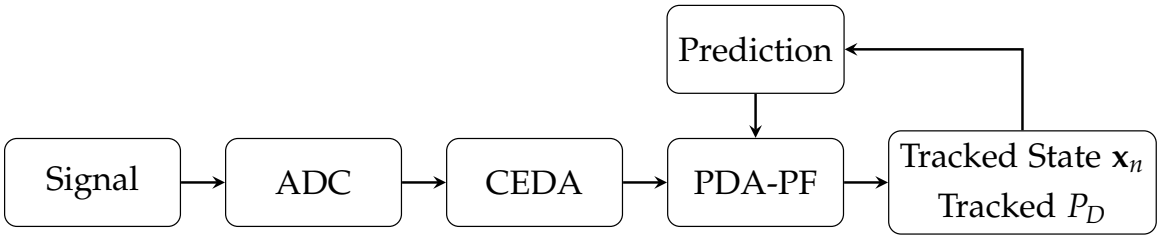


Figure 3.5: Factor Graph for the PDA filter.

3.5.8 Discussion

In terms of computational cost, the PDA filter is itself is similar to the particle filter, with the exception of needing $M_N * N$ measurement likelihood evaluations and some additional computations for the P_D tracking. The CEDA is quite complex with the iterative optimization process, taking about half the total execution time of a PDA filter iteration.

3.6 Machine Learning Augmented PDA Filter

To enhance the performance for the PDA filter, an augmentation with machine learning (ML-PDA) was proposed in this thesis and tested. As a novel approach, machine learning as an additional measurement likelihood was added to the filter, similar to what is proposed in [17]. To facilitate the implementation, a Gaussian Process, as found in [18], was chosen as inference model, because it estimates a PDF $p(\mathbf{z}_f|\tilde{\mathbf{t}}_f(\mathbf{p}_n); \mathbf{P}_{train}, \mathbf{Z}_{f,train})$, as a Gaussian distribution $\mathbf{z}_f \sim \mathcal{N}(\mu_z, \sigma_z)$, which is convenient for using it in the Bayesian filter framework. The features to be learned and predicted were chosen as a set of signal parameters derived from the CIR, which have already been used for ML-supported UWB positioning methods, e.g. in [4] and [5]. The input was chosen as the agent position $\mathbf{p}_n$.

3.6.1 Gaussian Process

Gaussian processes describe a machine learning approach using kernel functions. The used kernel function $k(x_n, x_m)$ is the main design parameter for a Gaussian process, largely influencing the mapping between target $\mathbf{t}_{f,n}$ and input values $\mathbf{p}_n$. Detailed explanations for Gaussian Processes can be found in [18]. The regression for input $\mathbf{p}_n$ and output $\tilde{\mathbf{t}}_{f,n}$ is done using the following equations, with $\mathbf{k}^T(\mathbf{p}_n)$ as the vector of kernel function evaluated for $\mathbf{k}(\mathbf{p}_n, \mathbf{P}_{train})$. The matrix $\mathbf{C}$ denotes the covariance of the N training data points $\mathbf{P}_{train}$, $\mathbf{Z}_{f,train}$ are the training target values, and $c = k(\mathbf{p}_n, \mathbf{p}_n) + \beta^{-1}$, β^{-1} being a hyperparameter describing the noise precision. [18]

$$\mu_t(\mathbf{p}_n) = \mathbf{k}^T(\mathbf{p}_n)\mathbf{C}^{-1}\mathbf{Z}_{f,train} \tag{3.70}$$

$$\sigma_t(\mathbf{p}_n) = c - \mathbf{k}^T(\mathbf{p}_n)\mathbf{C}^{-1}\mathbf{k} \tag{3.71}$$

The training of a Gaussian process is done by maximizing the likelihood $p(\mathbf{t_{train}}|\boldsymbol{\Theta})$, with Θ being a kernel specific set of hyperparameters, which can be done using gradient algorithms or similar approaches. In this work, the kernel function was chosen heuristically to be a combination of a Radial Basis function and a White Kernel, with the White Kernel modelling AWGN in the training data. The hyperparameter l defines the width of the Radial Basis functions and is tuned during training.

$$k(p_n, p_m) = exp(-\frac{||p_n - p_m||^2}{2l^2}) + \sigma_\delta \delta(n-m) \tag{3.72}$$

3.6.2 Signal Parameters

As outputs, a set of six signal parameters was chosen based on [4], [5]. The parameters are:

$$\text{Signal Energy: } E_r = \int_T |r(t)|^2 dt \tag{3.73}$$

$$\text{Max. Amplitude: } r_{max} = \arg\max_t |r(t)| \tag{3.74}$$

$$\text{Rise Time: } t_{rise} = t_H - t_L \tag{3.75}$$

$$t_L = min\{t : |r(t)| \geq \alpha\sigma_n\}, \qquad \alpha > 0 \tag{3.76}$$

$$t_H = min\{t : |r(t)| \geq \beta r_{max}\}, \qquad 0 < \beta \leq 1 \tag{3.77}$$

$$\text{Mean Excess Delay: } \tau_{MED} = \int_T t \frac{|r(t)|^2}{E_r} dt \tag{3.78}$$

$$\text{RMS Delay Spread: } \tau_{RMS} = \int_T (t - \tau_{MED})^2 \frac{|r(t)|^2}{E_r} dt \tag{3.79}$$

$$\text{Kurtosis: } \kappa = \frac{1}{\sigma_{|r|}^4 T} \int_T (|r(t)| - \mu_{|r|})^4 t \tag{3.80}$$

$$\mu_{|r|} = \frac{1}{T} \int_T |r(t)| dt \tag{3.81}$$

$$\sigma_{|r|} = \frac{1}{T} \int_T (|r(t)| - \mu_{|r|})^2 dt \tag{3.82}$$

The integrals of the parameters were approximated by sums. The parameter α was chosen to be 20 and β to be 0.8, based in heuristic testing. The details on training data will be shown in Section 5.2.1, together with the measurement scenarios. The parameters are transformed to the logarithmic domain for better scaling and normalized to reduce the difference in the numerical ranges of the individual outputs values to a minimum for the GP regression, similar to [17].

3.6.3 Integration into PDA Model

The Gaussian process is integrated into the filter as an additional measurement likelihood between the predicted parameter vector

$$\tilde{\mathbf{t}}_{f,n}^{(j)}(\mathbf{p}_n) = [\tilde{E}_r^{(j)}, \tilde{r}_{max}^{(j)}, \tilde{t}_{rise}^{(j)}, \tilde{\tau}_{MED}^{(j)}, \tilde{\tau}_{RMS}^{(j)}, \tilde{\kappa}^{(j)}], \quad \tilde{\mathbf{\Sigma}}_{f,n}^{(j)}(\mathbf{p}_n) = Diag.[\tilde{\sigma}_{E_r}, \tilde{\sigma}_{r_{max}}, \tilde{\sigma}_{t_{rise}}, \tilde{\sigma}_{\tau_{MED}}, \tilde{\sigma}_{\tau_{RMS}}, \tilde{\sigma}_\kappa]$$

and the parameter vector calculated from the current measurements CIR:

$$\mathbf{z}_{f,n}^{(j)} = [E_r^{(j)}, r_{max}^{(j)}, t_{rise}^{(j)}, \tau_{MED}^{(j)}, \tau_{RMS}^{(j)}, \kappa^{(j)}]$$

The parameters are assumed to be distributed independently, with the covariance between the parameters being zero. This assumption is done to simplify the filter, while the actual covariance will not be zero due to the parameters depending on each other, and originate from the same CIR.

Using the estimated PDF of the Gaussian process we can evaluate the measurement likelihood $p_{GP}(\mathbf{z}_{f,n}^{(j)}|\tilde{\mathbf{t}}_{f,n}^{(j)}(\mathbf{p}_n), \tilde{\mathbf{\Sigma}}_{f,n}^{(j)}(\mathbf{p}_n))$ as:

$$p_{GP}^{(j)}(\mathbf{z}_{f,n}^{(j)}|\tilde{\mathbf{t}}_{f,n}^{(j)}(\mathbf{p}_n), \tilde{\mathbf{\Sigma}}_{f,n}^{(j)}(\mathbf{p}_n)) = \mathcal{N}(\mathbf{z}_{f,n}^{(j)}|\tilde{\mathbf{t}}_{f,n}^{(j)}(\mathbf{p}_n), \tilde{\mathbf{\Sigma}}_{f,n}^{(j)}(\mathbf{p}_n)) \tag{3.83}$$

A measurement function $\lambda(\mathbf{z}_{f,n}; \tilde{\mathbf{t}}_{f,n}(\mathbf{p}_n))$ is defined as:

$$\lambda(\mathbf{z}_{f,n}; \tilde{\mathbf{t}}_{f,n}(\mathbf{p}_n)) = \begin{cases} 1, & a_n^{(j)} \in M_n^{(j)} \\ p_{GP}^{(j)}(\mathbf{z}_{f,n}^{(j)}|\tilde{\mathbf{t}}_{f,n}^{(j)}(\mathbf{p}_n), \tilde{\mathbf{\Sigma}}_{f,n}^{(j)}(\mathbf{p}_n)), & a_n^{(j)} = 0 \end{cases} \tag{3.84}$$

For the factor graph of the PDA filter, this means an additional input for the joint measurement equation $\tilde{g}_{zn,GP}$. We assume the features to contain only NLOS information, i.e. to be uncorrelated to the LOS measurements $\hat{\mathbf{z}}_{n'}^{(j)}$ and to be uninformative in LOS conditions ($a_n^{(j)} = 0$). This way, the algorithm only uses the GP likelihood only in NLOS condition.

$$\tilde{g}_{zn,GP}^{(j)}(...) = h_m(a_{n'}^{(j)}; q_{n'}^{(j)}) g(\hat{\mathbf{z}}_{n'}^{(j)}; \mathbf{x}_{n'}, a_{n'}^{(j)}) \lambda(\mathbf{z}_{f,n}; \tilde{\mathbf{t}}_{f,n}(\mathbf{p}_n)) \tag{3.85}$$

$$= \begin{cases} \frac{p_{D,n}^{(j)}(q_n^{(j)})}{M_n^{(j)}} p_{LOS}(\hat{z}_{n,m}^{(j)}), & a_n^{(j)} \in M_n^{(j)} \\ (1 - p_{D,n}^{(j)}(q_n^{(j)})) p_{NLOS}(\hat{d}_{n,m}^{(j)}) p_{GP}^{(j)}(\mathbf{z}_{f,n}^{(j)}|\tilde{\mathbf{t}}_{f,n}^{(j)}(\mathbf{p}_n), \tilde{\mathbf{\Sigma}}_{f,n}^{(j)}(\mathbf{p}_n)), & a_n^{(j)} = 0 \end{cases} \tag{3.86}$$

3.6.4 Filter Equations

The filter equations for the ML-PDA filter are summarized below. The likelihood $p_{GP}^{(j)}(\mathbf{t}_n^{(j)}|\tilde{\mathbf{t}}_n^{(j)}(\mathbf{p}_n)), \tilde{\mathbf{\Sigma}}_{t,n}^{(j)}(\mathbf{p}_n)))$ weighted with $(1 - v_q)$, getting more weight for a low probability of LOS detection.

Prediction:

$$\mathbf{x}^i_{n|n-1} = \mathbf{A}\mathbf{x}^i_{n-1|n-1} + \mathbf{B}\mathbf{v}^i_n \tag{3.87}$$

$$\mathbf{q}^{(j)}_{n|n-1} = \Psi\mathbf{q}^{(j)}_{n-1|n-1} \tag{3.88}$$

$$\tilde{\mathbf{t}}^{i,(j)}_n, \tilde{\mathbf{\Sigma}}^{i,(j)}_{t,n} = \text{GP Regression}(\mathbf{x}^i_{n|n-1}) \tag{3.89}$$

Weight Update:

$$\mathbf{t}^{(j)}_n = \text{Calc. Params}(r^{(j)}_n(t)) \tag{3.90}$$

$$\tilde{w}^{m,i,(j)}_n = \mathcal{N}(\hat{d}^{(j)}_{n.m}|h^{(j)}(\mathbf{x}^i_{n|n-1}), \hat{\sigma}^{(j)}_{n,m}) \tag{3.91}$$

$$\tilde{w}^{q,i,(j)}_{n,GP} = \sum_{m=1}^{M^{(j)}_n} \frac{d_{max}}{M^{(j)}_n} v_q \times + \\ + (1 - v_q)\mathcal{N}(\mathbf{z}^{(j)}_{f,n}|\tilde{\mathbf{t}}^{(j)}_{f,n}(\mathbf{p}_n), \tilde{\mathbf{\Sigma}}^{(j)}_{f,n}(\mathbf{p}_n)) \tag{3.92}$$

$$\tilde{w}^{q,i,(j)}_{n,Q} = \sum_{m=1}^{M^{(j)}_n} \frac{d_{max}}{M^{(j)}_n} v_q \tilde{w}^{m,i,(j)}_n + (1 - v_q) \tag{3.93}$$

$$w^i_n = \prod_{j=1}^{J} \sum_{q=0}^{Q} q^{v_q,(j)}_{n|n-1} \tilde{w}^{q,i,(j)}_{n,GP} \tag{3.94}$$

Resampling, Estimate:

$$\mathbf{x}^i_{n|n} = resample(\mathbf{x}^i_{n|n-1}, w^i_n) \tag{3.95}$$

$$\hat{\mathbf{x}}_n = \frac{\sum_{j=1}^{N} \mathbf{x}^j_{n|n}}{N} \tag{3.96}$$

Update for q:

$$\mathbf{q}'^{(j)}_{n|n} = \mathbf{q}^{(j)}_{n|n-1} \sum_{m=1}^{M} \tilde{w}^{q,i,(j)}_{n,Q} \tag{3.97}$$

$$\mathbf{q}^{(j)}_{n|n} = \frac{\mathbf{q}'^{(j)}_{n|n}}{\sum \mathbf{q}'^{(j)}_{n|n}} \tag{3.98}$$

A second filter variant with fixed P_D was also implemented:

Prediction:

$$\mathbf{x}^i_{n|n-1} = \mathbf{A}\mathbf{x}^i_{n-1|n-1} + \mathbf{B}\mathbf{v}^i_n \tag{3.99}$$

$$\tilde{\mathbf{t}}^{i,(j)}_n, \tilde{\mathbf{\Sigma}}^{i,(j)}_{t,n} = \text{GP Regression}(\mathbf{x}^i_{n|n-1}) \tag{3.100}$$

Weight Update:

$$\tilde{w}^{m,i,(j)}_n = \mathcal{N}(\hat{d}^{(j)}_{n,m} | h^{(j)}(\mathbf{x}^i_{n|n-1}), \hat{\sigma}^{(j)}_{n,m}) \tag{3.101}$$

$$w^i_n = \prod_{j=1}^{J} \sum_{m=1}^{M^{(j)}_n} \frac{d_{max}}{M^{(j)}_n} v_q \tilde{w}^{m,i,(j)}_n + \\ + (1 - v_q)\mathcal{N}(\mathbf{z}^{(j)}_{f,n} | \tilde{\mathbf{t}}^{(j)}_{f,n}(\mathbf{p}_n), \tilde{\mathbf{\Sigma}}^{(j)}_{f,n}(\mathbf{p}_n)) \tag{3.102}$$

Resampling, Estimate:

$$\mathbf{x}^i_{n|n} = resample(\mathbf{x}^i_{n|n-1}, w^i_n) \tag{3.103}$$

$$\hat{\mathbf{x}}_n = \frac{\sum_{j=1}^{N} \mathbf{x}^j_{n|n}}{N} \tag{3.104}$$

The signal flowchart, shown in Figure 3.6, visualizes how the captured CIR is used for the CEDA and the calculation of the signal parameters. The Gaussian Process Regression outputs an estimate based on the predicted position.

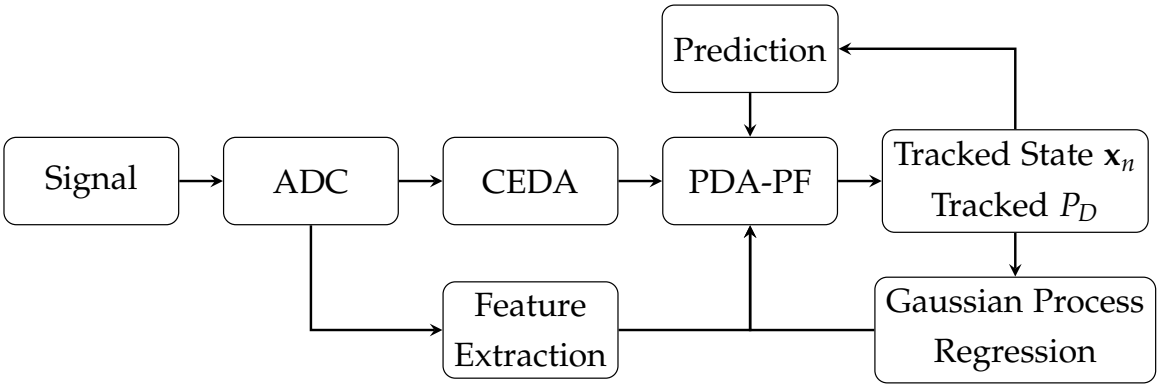


Figure 3.6: Factor Graph for the PDA filter.

3.6.5 Discussion

In addition to the CEDA and the multiple measurement likelihoods, the prediction of the Gaussian process is quite computationally costly, with a vector-matrix product between training data and its inverse covariance matrix, taking $O(N^2)$ operations, depending on the

number of training data points. For small data sets this will not add relevant amounts of execution time, but will quickly dominate the overall execution time for large datasets.

4 Measurement Setup

To enable wireless and real-time measurements, the NXP Ranger4 PCBs need to be connected to a power supply, and the control signals and data need to be transmitted to and from the nodes. A first iteration of the measurement setup was done in a previous project work, based on ESP8266 microcontrollers transmitting and receiving control data to the three anchors and the agent. Due to the limited memory on the ESP8266, transmission of the CIR was not possible, which is necessary for the more advanced tracking filters discussed in this work. Power for the nodes was provided by LiPO batteries with charging and voltage control electronics. The nodes were contained in 3D-printed housings with openings for a Micro-USB connector for charging, an On-Off switch and the UWB antenna. This chapter will present the updated measurement setup and the additional hardware used. The software setup for collecting data is shortly described, as well as the filter and ML software details.

4.1 The Ranger4 Chip

The NXP Ranger4 Evaluation Board is a PCB board featuring the NXP NCJ29D5 IC. This is an Ultra-Wideband IC that is used for ranging measurements in the automotive sector. The IC uses 500 MHz of bandwidth, with a carrier frequency between 6 and 8.5 GHz. Different radio modes are supported, offering IEEE 802.15.4 compliance as well as data transmission and secure ranging modes.

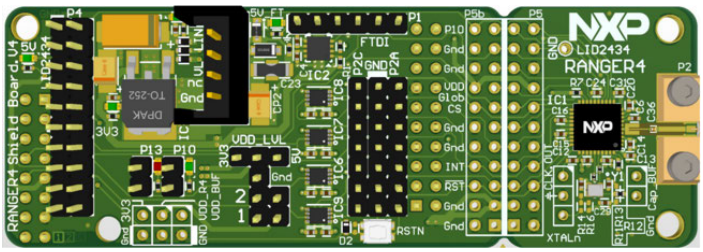


Figure 4.1: The NXP Ranger4 Evaluation Board. *NXP Semiconductors, NCJ29D5 Ultra-Wideband ICs for Automotive Applications, 2021.*

4.2 Hardware Setup

Communication with the chip is done via Universal Asynchronous Receive/Transmit (UART) interface with a baud rate of 921600 bit/s. Only the Ranger4 board acting as the agent needs to have a data connection to collect the ranging data and control ranging parameters. A Raspberry Pi Computer was mounted on the board to enable wireless communication. A custom adapter PCB was soldered, to enable mounting the Ranger4 board onto the Raspberry Pi, and a case was 3D-printed to protect the boards during measurements and to ensure ease of handling. A USB battery pack is used as a power source. [10]
The three anchor nodes were kept in their casings from the previous setup, only removing the ESP8266, but keeping the LiPo batteries and charging boards.

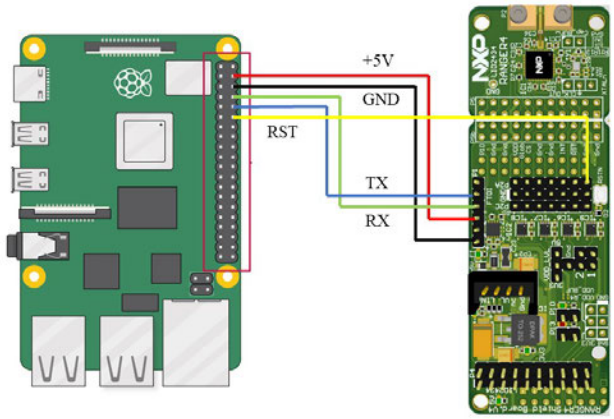


Figure 4.2: Wiring between the Raspberry Pi and the Ranger4 board.

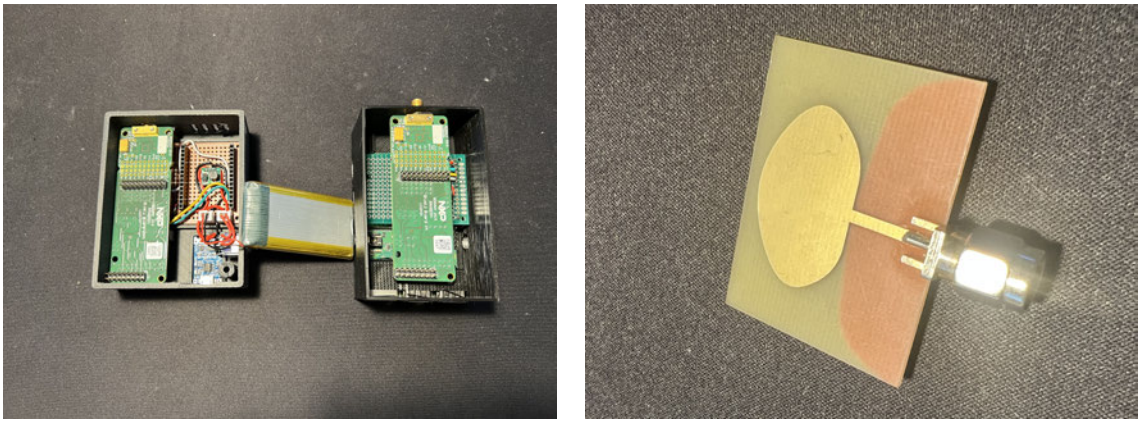
To enable dynamic measurements without human body interference, an iRobot Create 3 robot was set up and used to run trajectories with the agent mounted in a 3D printed stand on the robot. This robot was chosen for its simplicity in handling and programming. A web-based Python environment is available, offering basic movement functions to program trajectories. The trajectories can be programmed based on lengths and angles, which is automatically supervised by the on-board wheel encoders and optical odometry sensor, leading to high accuracy and repeatability. A charging dock is also available with an automatic homing function, as well as an on-board USB-C power connection which was used to run the agent and the Raspberry Pi for the measurement scenarios on the robot [19].

Figure 4.3: Robot with the agent mounted on top.

In addition to the connecting PCB, a casing for the nodes was designed and 3D-printed to protect the boards and hold the battery, shown in Figure 4.4a. A holding clamp was also printed for the casing, to mount the nodes on microphone stands. This reduces the issue of floor reflections and the height can be adjusted easily, matching the height of the agent in different setups to ensure the 2D assumption of the scenario.

For antennas, the Decawave WB002 UWB antenna was used for all nodes, mounted directly onto the boards with the SMA connector. The antenna design and parameters are described in [20]. The WB002 antenna shows good performance for UWB ranging, with $<$6dB variation of radiated power on the azimuth plane, and low variation of the group delay at $6.5GHz$. An angle-dependent variation of group delay directly translates to a ranging error, since the introduced delay is added to the estimated ToF.

To collect the data and run live filters as well as archive the data, the Raspberry Pi is connected to a laptop via a WiFi router. For referencing the nodes' positions, an Optitrack Motion Capture System was provided by NXP Semiconductors, Gratkorn, Austria. The



(a) Picture of the nodes in their casings.

(b) The Decawave WB002 antenna.

Figure 4.4: •

system consists of 4 cameras facing the measurement area from different angles, connected to a central laptop running the tracking software. This laptop was connected to the measurement laptop via a network connection, streaming time sync and position data to be collected and archived. A schematic overview of the entire setup is shown in Figure 4.5, with the two laptops, the three anchors and agent as well as the Optitrack cameras.

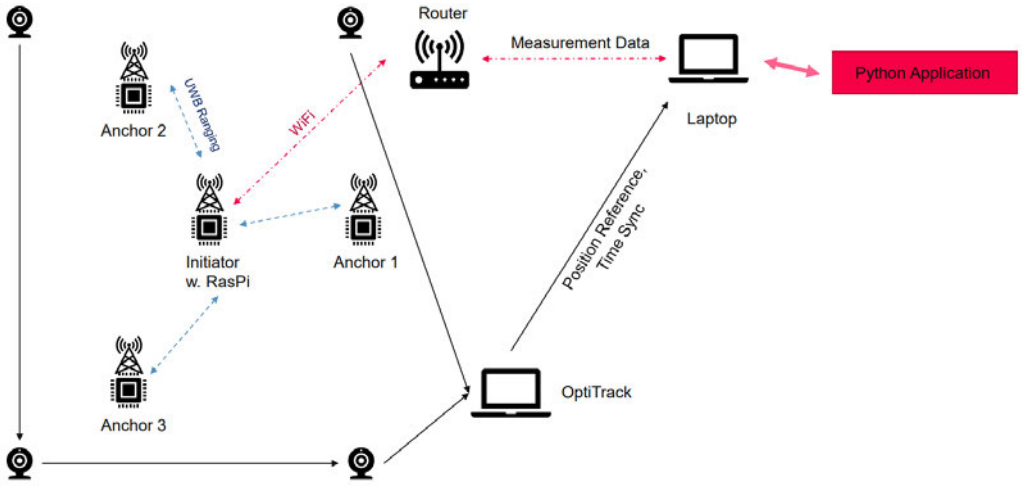


Figure 4.5: Overview of the setup and data flow.

The Optitrack system requires optical markers to be mounted onto any object that needs to be tracked. At least three markers in a geometrically fixed constellation need to be used per object, with the tracking point being in a fixed relation to the markers. An anchor with the markers is shown in Figure 4.6. This tracking point can be calibrated in relation to the markers, which was used to center the tracking points into the center of the antennas on

the nodes.

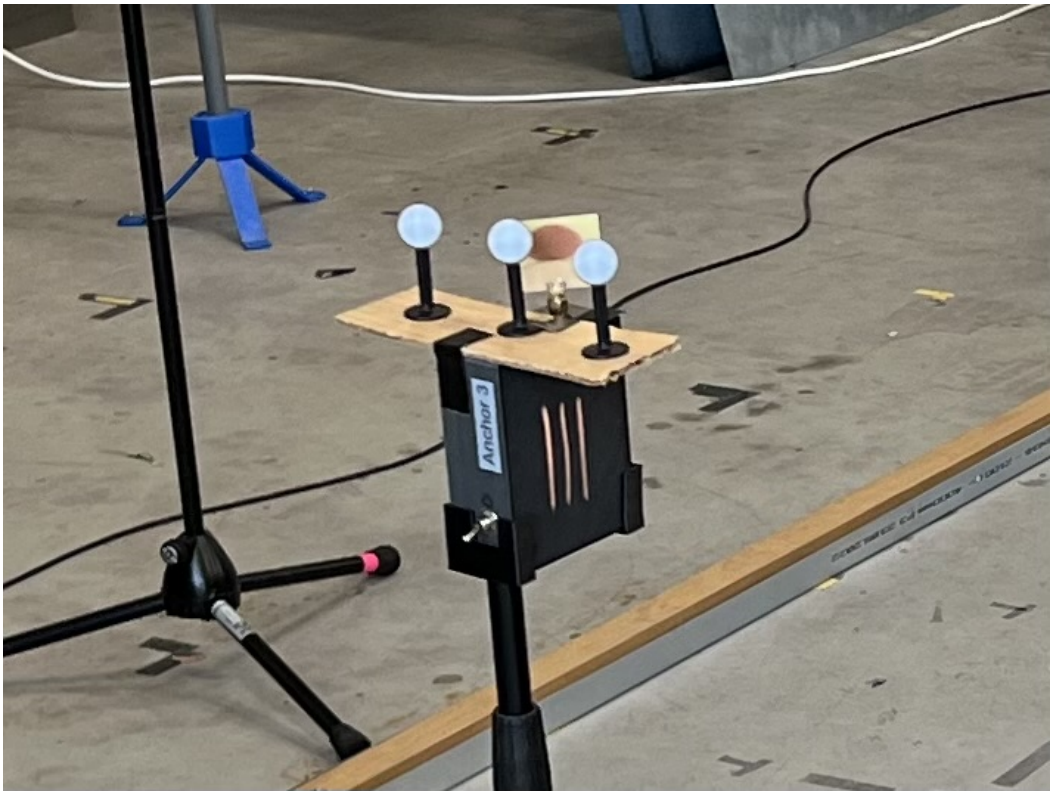


Figure 4.6: A node with the optical markers mounted.

4.3 Software Setup

To enable a single workflow across all needs of this project, Python was chosen as a programming language. It offers a wide variety of hardware- and mathematical libraries to work with, as well as good networking capabilities. Python is also well supported on the Raspberry Pi. NumPy was used as a math library, scikit-learn for some additional functionalities like the Gaussian Process and Matplotlib for all plots. The referenced files can be found in a supplementary git folder.

4.3.1 Measurement Software

The basis of the project is the control and measurement software between the PC and he Raspberry Pi. The Python file ”main.py” was written to run on the Raspberry Pi to set

up a web socket connection between the Raspberry and the PC, with the counter piece "Data_Receiver.py" running on the PC. This code is used to start and stop measurements, transmit measurement data and control the measurement parameters. The commands to the Ranger4 chip are passed via the UART interface as UTF-8 strings, offering the following options:

Key	Description
a X	Set nr. of anchors to X
q	Set to idle mode
p X	Set peak output power
C X X	Set radio mode (0...8) and carrier frequency in kHz
i X	start ranging and return distance only
c X	start ranging and return CIR and distance

The control string used for setting up and starting the measurements was as follows: *"a 3 C 0 6500 c"*. This string starts a measurement with three anchors, radio mode 0 at 6.5 GHz and returns the CIR, the distance estimate and the CIR information string for each anchor and each timestep. For the distance-only mode, the sampling time T_s is 20 ms, while for the CIR mode T_s is 300 ms. The slower sampling time is necessary due to the time the CIR transmission takes on the UART interface. The CIR is transmitted as 2048 integers in string format, with every other element of the array being an imaginary value, representing the complex baseband CIR. These CIR values are relative values, meaning they do not directly represent the physical signal amplitude. This scaled amplitude depends on the gain settings the RF frontend on the Ranger4 chip used to scale the signal to the proper dynamic range to be digitized. The additional information transmitted by the chip can be used to convert the relative amplitudes back to voltage values, although with limited accuracy due to the integer representation used for the transmission. This also effects the noise floor, potentially causing mismatches between the noise power given by the chip and a noise power calculated off of the transmitted CIR. A Python function was written to convert the relative amplitudes to voltage levels, which is necessary for comparing different CIR, which potentially do not have a matching scaling otherwise.

The additional information returned by the Ranger4 chip contains the following entries:

Index	Description
0	Counter
1	Initiator ID
2	Anchor ID
3	Single-Sided Ranging 1
4	Single-Sided Ranging 2
5	Double-Sided Ranging
6	Error Code
7	Ranging Algorithm
8	Anchor Group
9	Detection Threshold Power
10	Edge Index
11	First Path Detected
12	First Path Index
13	First Path Power
14	Max Tap Index
15	Max Tap Power
16	Noise Power
17	Overall Received Power
18	First Path Offset

All these return strings are converted to NumPy arrays, analysed in the receiving part of the script, and stored and archived at the end of the measurement.

The receiving script also has the option of running a set of filters for each received timestep, to analyse filters in real-time. A live plot option to show the CIR for three anchors and the position estimate for each timestep was also added.

4.3.2 Optitrack

The Optitrack system, which is installed in a measurement facility of NXP, was connected to the measurement laptop via a network cable. To time-synchronize the two separate measurement setups, a local Network-Time Protocol server was running on a Raspberry Pi. The Optitrack position data was transmitted to the measurement laptop, using the Message Queuing Telemetry Transport (MQTT) protocol. The MQTT messages were received by a proprietary C# interface and stored in a SQlite database for later use.

4.3.3 Filters

The filters presented in Chapter 3 were each implemented as a class, with an initialization method and a method for iterating the next timestep. The implementations can be found in the file "filters.py". Parameters for the filters can be set with a setup class containing relevant tuning variables like sampling time T_s, process noise σ_Q and the fixed detection probability P_D. The filters were implemented following the filter equations given in Chapter 3. The filter objects store the estimates, facilitating storage after analysis runs.

4.3.4 Gaussian Process

For Gaussian Process Regression, as used with the ML-augmented PDA filter described in Section 3.6, the scikit-learn GaussianProcessRegressor module was used. The GPR objects were trained per anchor, with the training data sets presented in Chapter 5. After training, the objects were stored in a file using the joblib library.

5 Measurement Campaign

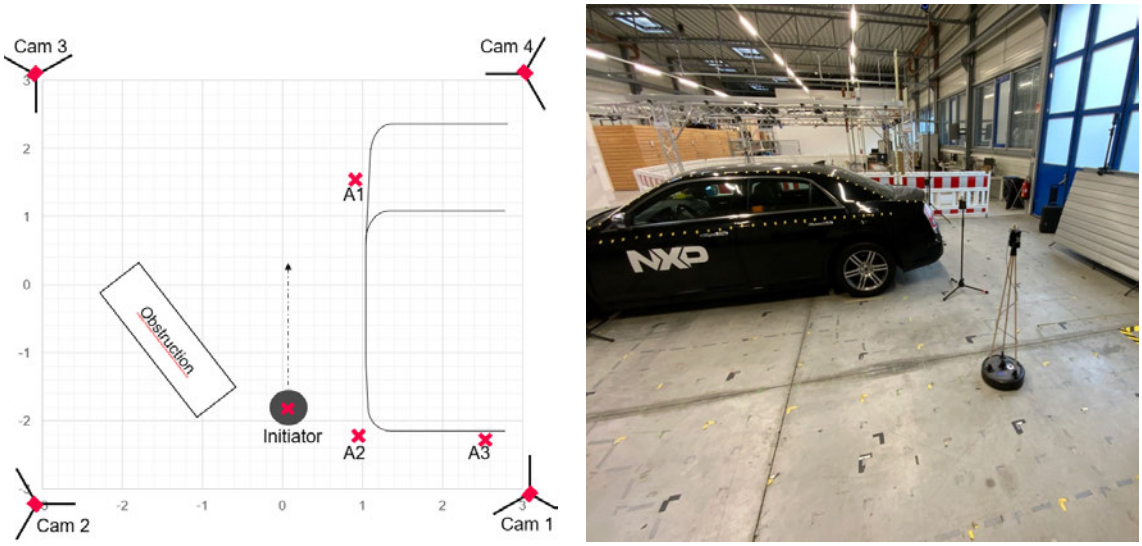
Following the implementation of the measurement system presented in Chapter 4, capturing a dataset was an important goal of this master thesis. The measurements were taken at a facility of NXP in Gratkorn, with the Optitrack system described in Section 4.3.2. The combination of wireless ranging setup and Optitrack system, as discussed in Section 4.2, enabled capturing dynamic trajectories with precise time and position references. The measurement campaign was, hence, focused on capturing different dynamic trajectories.

5.1 Design and Challenges

The goal of the measurement campaign was to obtain real-world data on which different tracking filters can be tested and compared. Impairments to the measurement were added to enable evaluation of the filters in non-optimal environments.
The individual scenarios were chosen to vary both in trajectory, impairments and agent type, with a human and a robot agent. The anchors were fixed in position for all scenarios, spread around a car similar to an application scenario. Besides the obstruction, no changes in geometry or environment were made. Figure 5.1a shows the schematic layout of the measurement area, Figure 5.1b shows a picture of the car with the anchor nodes and the robot agent, with the agent node mounted on top.

5.1.1 Obstruction

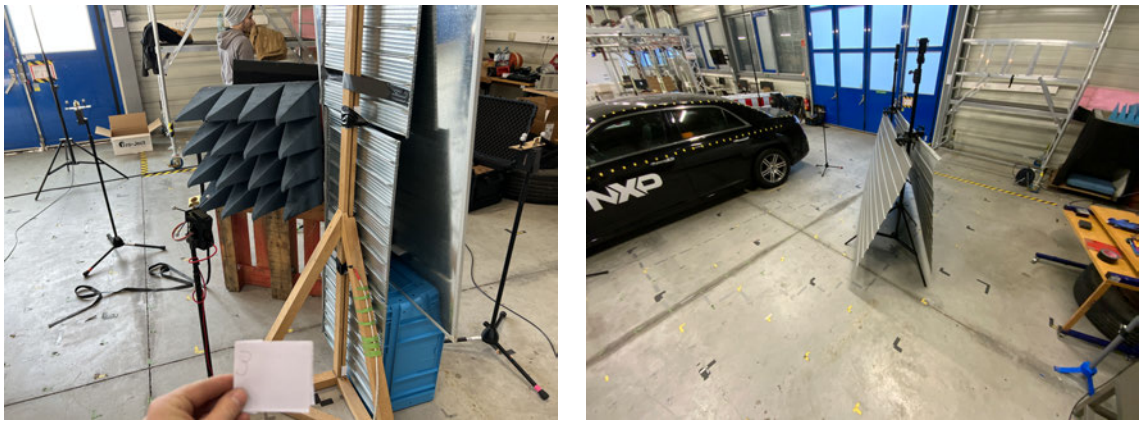
To simulate an NLOS situation between the respective anchors and the agent, an obstruction between the two nodes is necessary to block the LOS path. In initial heuristic trials, shown in Figure 5.2a, it proved difficult to reliably dampen or block the UWB signal on the direct path, with the Ranger4 chip's Search-Back algorithm still mostly detecting the LOS component correctly. RF absorbers as well as different combinations of metal plates were tested.



(a) Floorplan of the measurement scenarios.

(b) The anchors around the car.

Figure 5.1

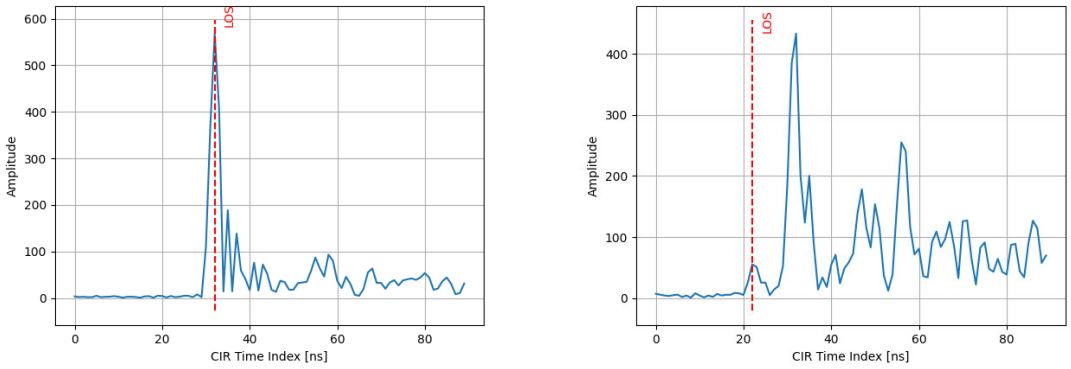


(a) Test of different obstructions.

(b) Final obstruction set up.

Figure 5.2

After testing and evaluating, two large metal plates mounted on stands were used as an obstruction, as shown in Figure 5.2b. The width of the obstruction was beneficial for a longer phase of obstructed ranging, posing a higher challenge for the filters. The double plate setup showed good shadowing for the LOS components, although no perfect NLOS scenario could be achieved. A comparison between a LOS CIR and the obstructed CIR is shown in Figure 5.3.



(a) CIR without an obstruction.

(b) CIR with an obstruction between nodes.

Figure 5.3

5.1.2 Human Body Interference

As mentioned in Chapter 2, the human body poses a big challenge to UWB ranging. As shown in [8], the human body shows strong shadowing and dispersion on the UWB signals, causing loss of the LOS component as well as additional MPCs. While these effects were not included in the models used for the filters, evaluation of the filters under these effects is important, considering the UWB keyfob usecase. Two on-body positions for the agent were used, one on the side of the body, one in front.

(a) Agent position in front of the body.

(b) Agent position on the side of the body.

Figure 5.4

The on-body positions of the agent were difficult to track with the Optitrack system, since the optical markers were often shadowed from the cameras by the body, causing a loss of track and incomplete position references. As a workaround, a wooden pole with optical

markers at the top end was fixed to the agent, to elevate the optical markers above the head. The tracking point was then calibrated to the antenna, resulting in a correctly tracked continuous reference position despite the on-body position.

Figure 5.5: Agent with the attached tracking extension.

5.1.3 Robot

To gather data without human-body interference, the iRobot Create 3 robot was used, as described in Section 4.2. The robot's deterministic trajectories enabled running a near-exact path with and without the obstruction in place, making it possible to directly compare the effect of the obstruction on filter behaviour and performance.
The robot also made it possible to systematically take measurements across the measurement area, to generate a training data set for ML applications. Otherwise, this would have been a very time-consuming task, because either the agent needs to be moved in small iterations across the area between measurements, or with a walking trajectory, human body effects would have distorted the training data.

5.2 Acquired Data

In the measurement campaign, a total of 13 datasets were captured and stored, consisting of both Ranger4 measurement data and the Optitrack position reference data. Of these 13 datasets, five are training datasets for the GP, and eight are trajectories with four of them using the robot, and four are on-body measurements. The datasets are listed below:

Dataset	Description
1	Training dataset 1
2	Training dataset 2
3	Training dataset 3
4	Training dataset 4
5	Training dataset 5
6	Trajectory 1, on-body, without obstruction, side
7	Trajectory 2, on-body, without obstruction, front
8	Trajectory 3, on-body, with obstruction, side
9	Trajectory 4, on-body, with obstruction, front
10	Trajectory 5, robot, with obstruction
11	Trajectory 6, robot, with obstruction, human obstruction
12	Trajectory 7, robot, with obstruction, dynamic human obstruction
13	Trajectory 8, robot, without obstruction

5.2.1 Training Datasets

The training datasets cover most of the measurement area, to be able to match a position to a set of parameters using the GPR. This process is called fingerprinting, comparable to what was shown in [2], [3]. Training datasets 2, 3 and 4 were taken with the obstruction in place, datasets 1 and 5 were taken without the obstruction. Figure 5.6 shows all training datasets, Figure 5.7 shows the datasets with the obstruction. The area occupied by the obstruction is clearly visible, since no datapoints can be taken there. This subset of the training datasets was used to train the GPR.

Taking a look at the CIR of the individual anchors for training dataset 4, shown in Figure 5.8, the drop-off in amplitude for the LOS component behind the obstruction can be observed. Also, stronger MPCs can be seen, following the LOS component with a bias between 5-8 meters. This can possibly be explained by a strong wall reflection, or the blue steel door as seen in the back of Figure 5.2b.

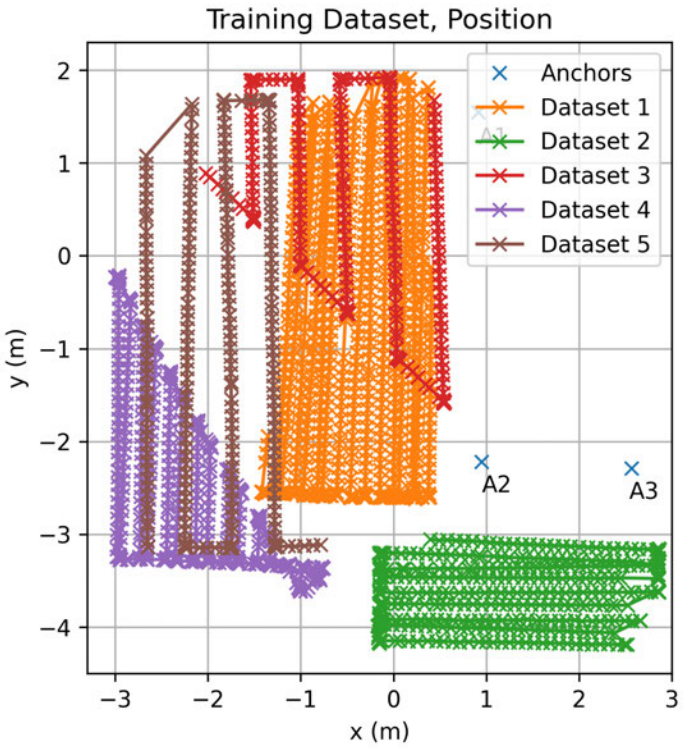


Figure 5.6: All 5 training datasets.

5.2.2 On-body Datasets

The first four trajectory datasets were done by carrying the agent on the body, as described above. Two measurements were done with the obstruction, and two without, both paths are shown in Figure 5.9. For both cases, one measurement was done with the agent in front of the body, and the second measurement with the agent on the side of the body.

As expected, Figure 5.10 shows strong fading of the LOS component, caused by the body obstructing the path between agent and anchor. Also visible are white lines, which represent timesteps were no data connection between the agent and the anchor was possible, and therefore no ranging happened.

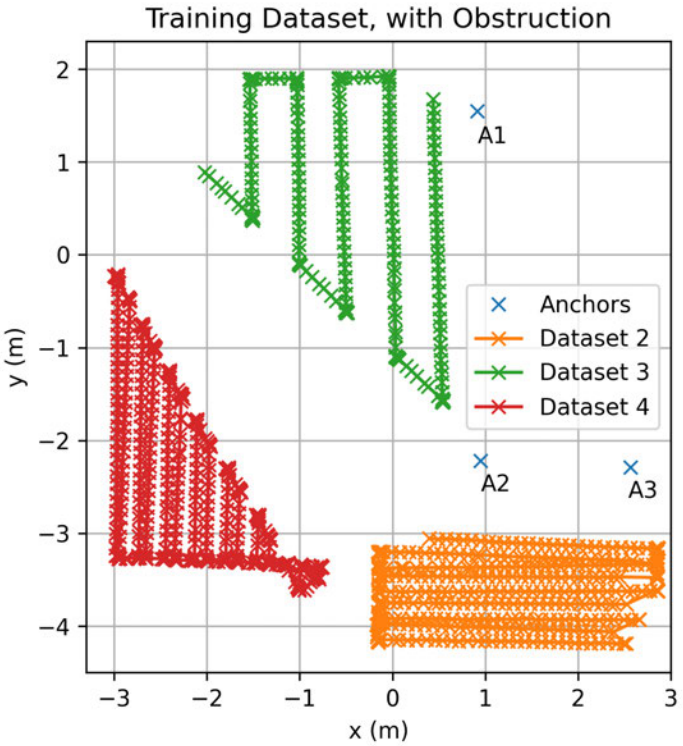


Figure 5.7: The 3 datasets with obstruction.

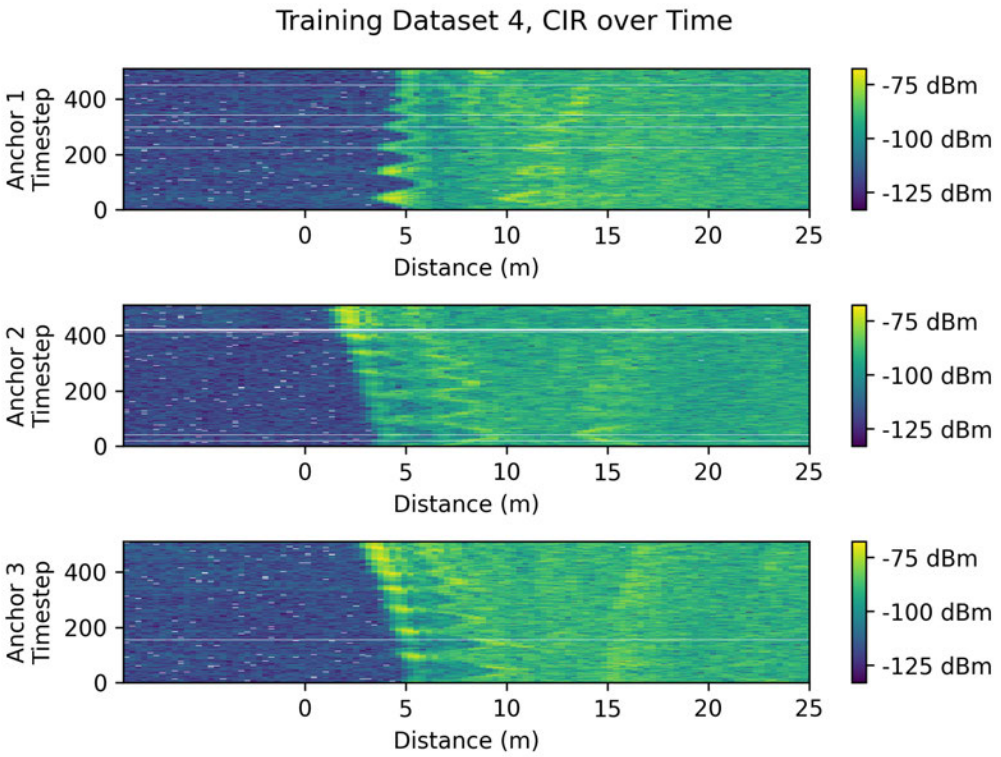


Figure 5.8: The CIRs of the three anchors.

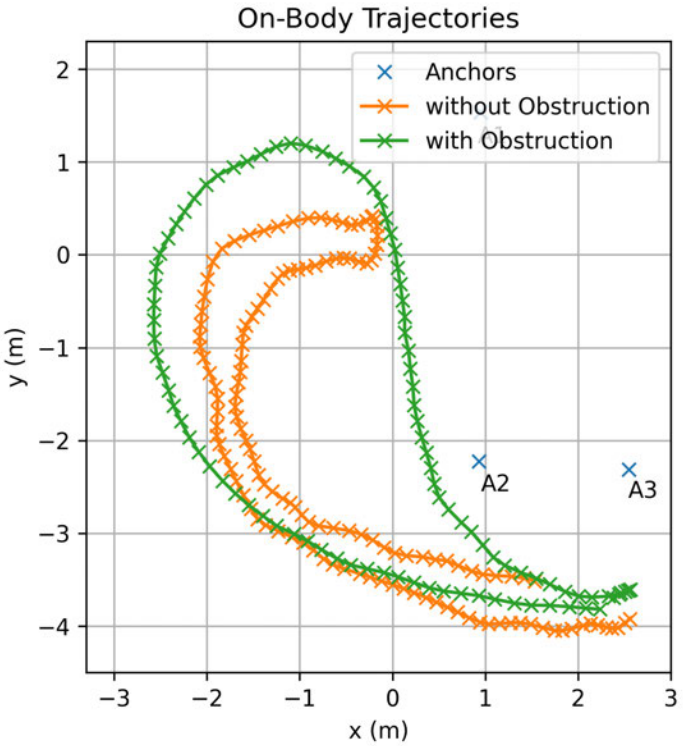


Figure 5.9: The two on-body trajectories.

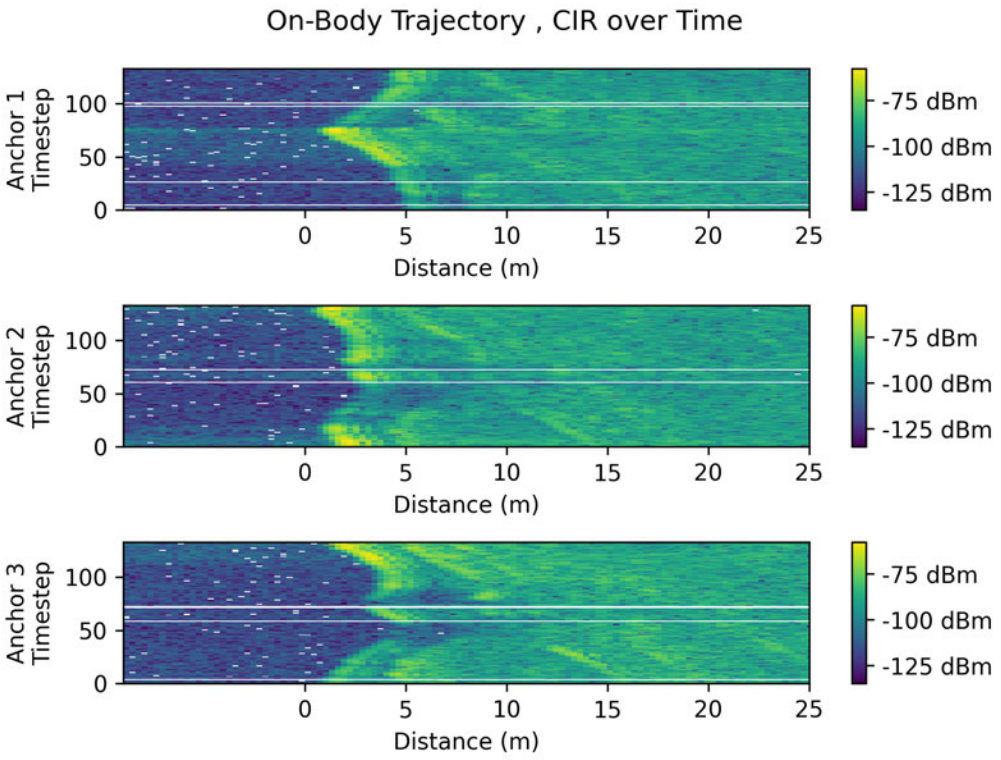


Figure 5.10: The CIRs of the three anchors.

5.2.3 Robot Datasets

For the four trajectories captured with the robot, the same path was used each time, shown in Figure 5.11. Trajectory 5 was taken with the obstruction in place, for trajectories 6 and 7 two people were added in front of the obstruction as additional dampening. Trajectory 8 was done entirely without any obstruction.

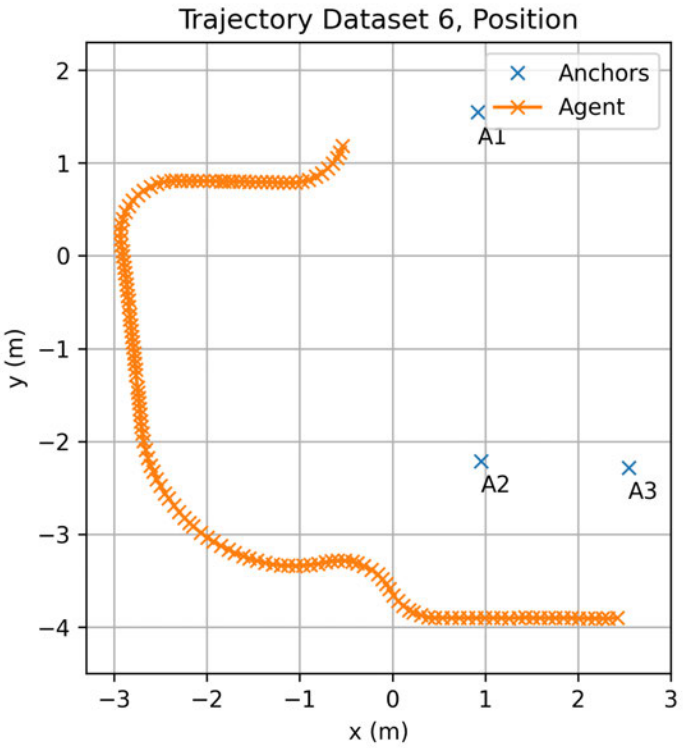


Figure 5.11: The robot trajectory.

Figure 5.12 shows a comparison between the CIRs of trajectory 5 (with obstruction) and trajectory 8 (without obstruction). The areas marked by the red rectangles show where the LOS component is shadowed in trajectory 5, compared to the continuous LOS components in trajectory 8.

5.3 Preparation of the Measurement Data

To be able to use the dataset for filter analysis, the position reference had to be imported into the python environment and matched to the samples of the ranging results. Due to

the much higher sampling rate of the Optitrack system (0.03 s vs 0.3 s for the ranging system), and different delays in data transmission and processing, the correct position reference samples had to be fitted to the ranging samples. This was done by calculating the Maximum Likelihood position estimates as described in the beginning of Chapter 3, and then jointly matching all estimated positions to a sub-sampled, shifted version of the reference points, finding an optimal time shift to minimize the difference between estimate and reference. The corresponding code is found in the file "MoCap-CombineData.py".
In addition, some parts of the reference trajectories were still missing sections due to track losses of the Optitrack system. These sections were interpolated with a cubic spline interpolation scheme, using the *scipy.interpolate.splprep* function.
Finally, all trajectories were saved in individual archives, containing the measurement samples with distance, CIR, additional ranging information vectors, and the matched reference positions.

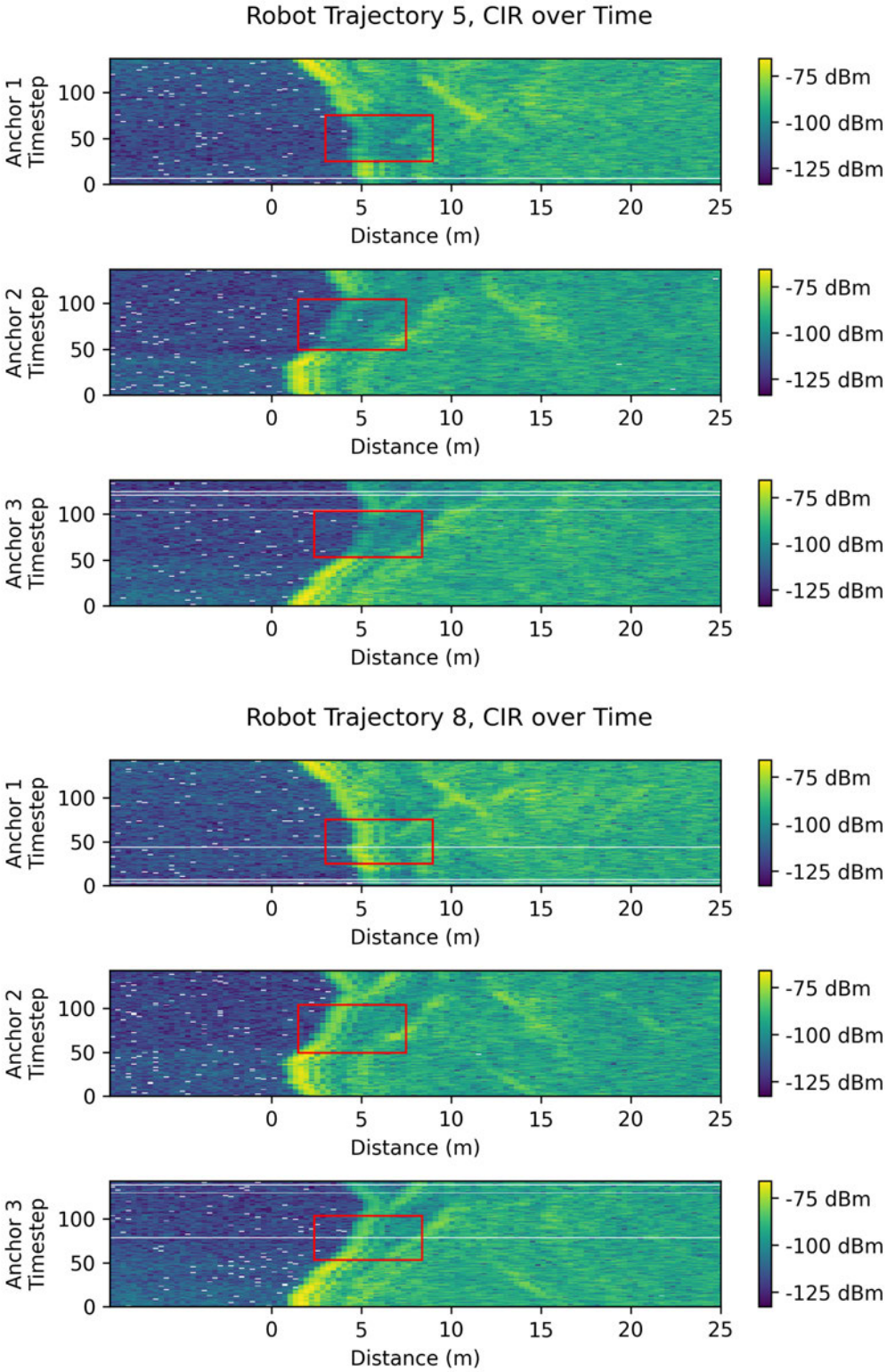


Figure 5.12: CIR comparison between trajectories with and without obstruction.

6 Evaluation of Algorithms

As a final step in this thesis, the different filters were tested and compared. Some tuning parameters and the analysis environment are briefly discussed, before the results of the analysis are presented.

6.1 Filter Variants

A total of six filter variants was compared in the final analysis. As a baseline comparison, the EKF filter was tested as well as the implementation of the SIR/ Particle filter. Next, the PDA filter was tested, one variant with the P_D fixed, and a second variant with P_D tracked. Lastly, two variants of the ML-augmented PDA filter were tested; again one with fixed P_D, and the other one with tracked P_D. The filters are listed below, with the abbreviations used in the plots:

1	EKF	Extended Kalman filter
2	Part	Particle filter
3	PDA, f. PD	PDA filter with fixed P_D
4	PDA, t. PD	PDA filter with tracked P_D
5	ML-PDA, f. PD	ML-augmented PDA filter with fixed P_D
6	ML-PDA, f. PD	ML-augmented PDA filter with tracked P_D

The tuning parameters used are shown in table 6.1.

The number of particles was chosen relatively low with $N = 1000$ due to the focus on real-time tracking in this project. Also, the fixed P_D was chosen relatively high, due to the fact that it proved difficult to generate long NLOS situations, with the NXP Ranger4 chips mostly still finding the LOS distance, even under strong shadowing. The measurement noise for the filters was calculated via the distance CRLB from the measurement SNR for each timestep.
The CEDA was set to finish after six signal components for each CIR to reduce runtime,

Parameter	Value
Sampling time T_s	0.3 s
Process noise σ_Q	250 m ($\sim 2,5\frac{m}{s^2}$)
Fixed P_D	0.9
Particles N	1000

Table 6.1: The tuning parameters.

and limit the complexity of the filter calculations, again with the goal to keep the design as close to real-time as possible. The start positions were initialized as the first reference position, with some noise added.

For the P_D tracking, a discrete Markov Chain with four states was used. The state vector $\mathcal{Q}$ was chosen to spread evenly between 0.01 and 1, depicting the entire possible range for the random variable $q_n^{(j)}$. The transition matrix $\mathbf{Q}$ was designed in a way to keep the detection of probability relatively high. The transition likelihood towards higher $q_n^{(j)}$, $\Psi(q_n^{(j)} = v_{k-1}|q_{n-1}^{(j)} = v_k)$, was chosen to be 0.4, while the transition likelihood toward lower $q_n^{(j)}$, $\Psi(q_n^{(j)} = v_{k+1}|q_n^{(j)} = v_k)$, was chosen to be 0.05. This choice was made by running test runs, which showed better performance for transition matrices with a strong trend for high $q_n^{(j)}$.

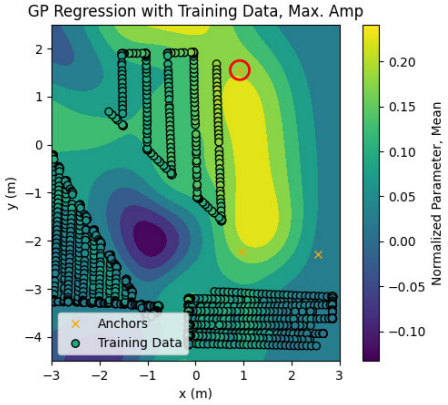
$$\mathcal{Q} = \begin{bmatrix} 0.01 \\ 0.333 \\ 0.666 \\ 1 \end{bmatrix} \tag{6.1}$$

$$\mathbf{Q} = \begin{bmatrix} 0.6 & 0.05 & 0 & 0 \\ 0.4 & 0.55 & 0.05 & 0 \\ 0 & 0.4 & 0.55 & 0.05 \\ 0 & 0 & 0.4 & 0.95 \end{bmatrix} \tag{6.2}$$

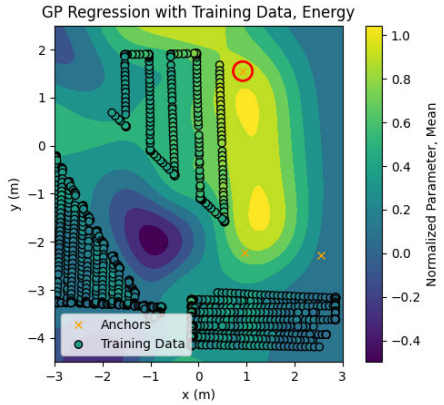
6.2 Training the Gaussian Process

To train the GP for the filter application, the reduced training dataset with the obstruction was used, as described in Section 5.2.1. Tests showed that using the full dataset led to badly matching predictions for both trajectories with and without obstruction, possibly due to

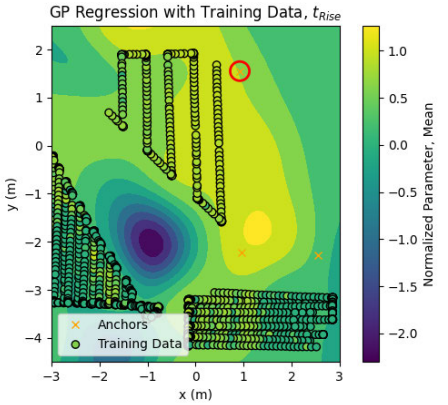
the fact that two conflicting datasets were used for training. With the obstructed trajectories being more interesting for the evaluation of the filters, the training data was limited to the training datasets with the obstruction. Based on this training dataset, the parameters discussed in Chapter 3.6 were calculated, normalized and transformed into logarithmic scale. These conditioned parameters were then used to train the GPR objects to be used in the ML-augmented PDA filter. A separate object was trained for each anchor. The training results show a good match between the model and the training data, with the predicted areas between the training data areas running smoothly. For visualization, the parameters were plotted together with a grid of the regression results for the entire used area. Figures 6.1a through 6.1f show the results for Anchor 1. Figure 6.2 shows the predicted standard deviation for the energy parameter. The standard deviation is close to 0 for areas where training data points are close and rises with higher distance to the training datapoints.



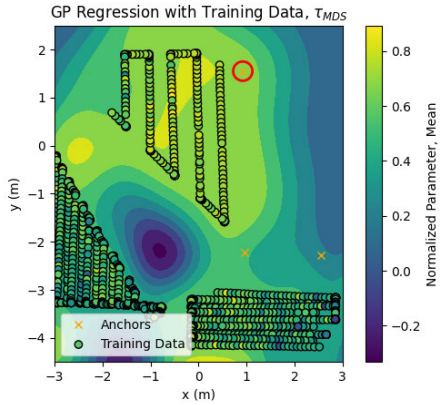
(a) Maximum amplitude parameter, training data vs. prediction.



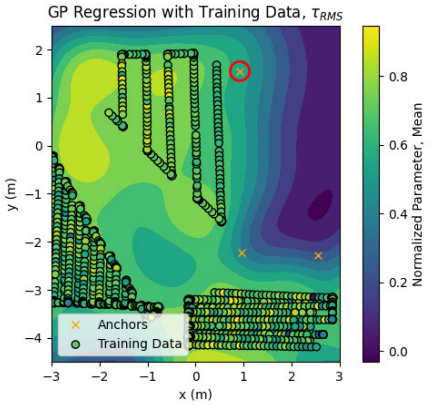
(b) Energy parameter, training data vs. prediction.



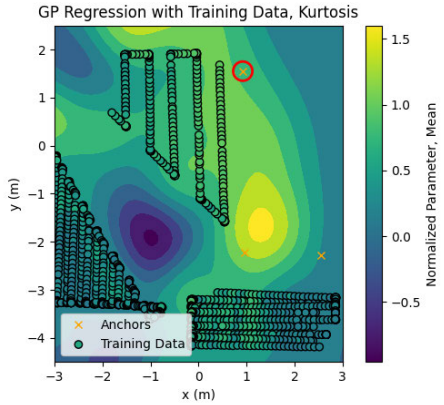
(c) t_{rise} parameter, training data vs. prediction.



(d) τ_{MDS} parameter, training data vs. prediction.



(e) τ_{RMS} parameter, training data vs. prediction.



(f) Kurtosis parameter, training data vs. prediction.

Figure 6.1

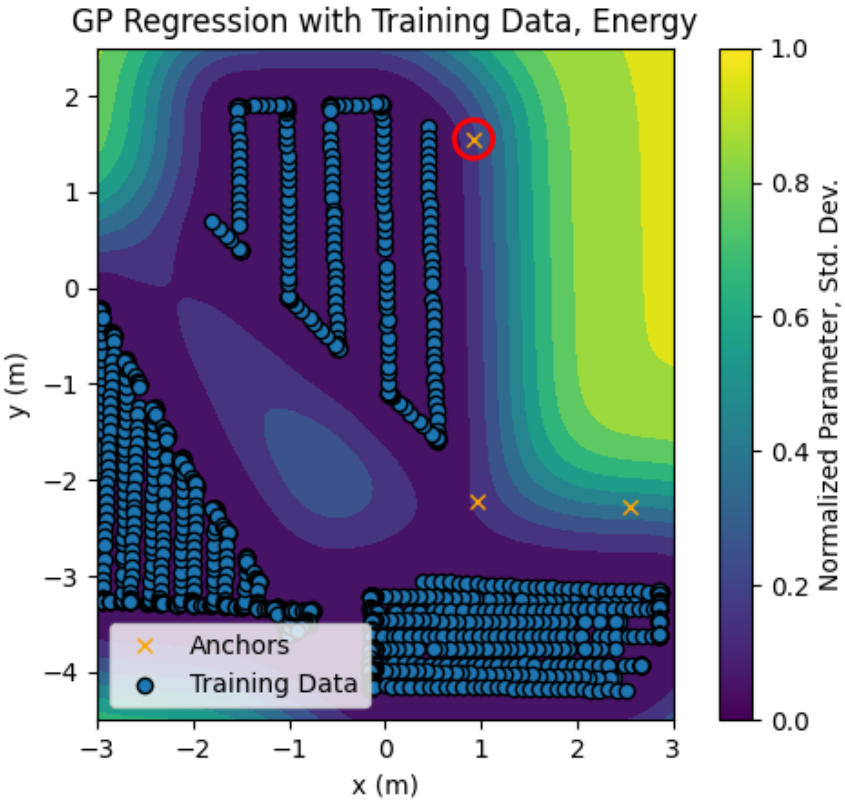


Figure 6.2: Predicted standard deviation for the energy parameter.

6.3 Analysis Setup

For the filter analysis, a testbed script was written to run the six different filters for the given trajectories with the prepared data described in Chapter 5.3, the corresponding code can be found in file "MoCap_Filter_Analyzer.py". To augment the analysis with long NLOS sections, the robot trajectories were used twice, with the second run having the CEDA results close to the ground truth LOS distance, obtained by the Optitrack system (see Section 4.3.2), removed from the measurement vectors. The distance estimate closest to the correct distance of the resulting measurement vector was then used as input for the EKF and particle filter. The resulting CEDA estimates for one anchor are shown in Figure 6.3, the unmodified results on the left, the one modified to have a NLOS section on the right.

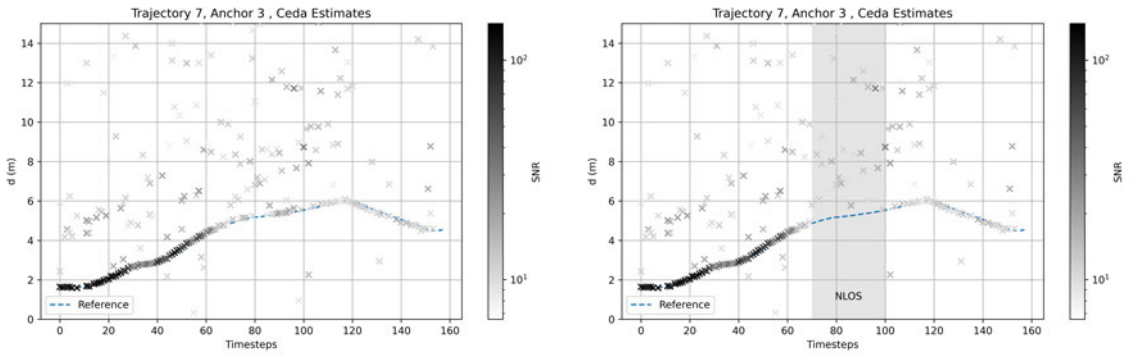


Figure 6.3: Comparison of the CEDA estimates for the modified trajectory.

To achieve a statistically meaningful analysis result, a Monte Carlo simulation was done, running all six filters 500 times on each trajectory. To create variation between the multiple runs on the trajectories, AWGN was added to the measured CIRs and the ranging estimates to achieve constant 20 dB SNR in relation to the maximum tap power of each individual CIR. The noise realizations were drawn individually for each run.

6.4 Position CRLB

To find a lower limit for position accuracy, the individual distance CRLBs can be combined, and, by using the Jacobian matrix as with the EKF, the position CRLB can be approximated, as found in [16], [21]. It results in an estimate for the optimal filter performance achievable for a given measurement SNR. Starting with the Fisher Information Matrix for the range estimations $\mathbf{F}_d$, which consists of the inverse of the measurement variances $\sigma^{(j)2}$, and the Jacobian $\mathbf{H}_n(\hat{\mathbf{x}}_n)$, the estimated position CRLB can be formulated as:

$$\mathbf{F}_{d,n} \leq \begin{bmatrix} \frac{1}{\sigma^{(1)2}} & 0 & 0 \\ 0 & \frac{1}{\sigma^{(2)2}} & 0 \\ 0 & 0 & \frac{1}{\sigma^{(3)2}} \end{bmatrix} \tag{6.3}$$

$$\mathbf{\Sigma}_{pos,n} \geq \mathbf{F}_{pos,n}^{-1} = (\mathbf{H}_n \mathbf{F}_d \mathbf{H}_n^T)^{-1} \tag{6.4}$$

As shown in [22], the position CRLB for nonlinear tracking filters can be formed as a posterior CRLB, combining the position CRLB of the prior, which is calculated from the state-space model, with the measurement CRLB at each timestep.

$$\mathbf{\Sigma}_{pos,pred,n} \geq \mathbf{A}\mathbf{\Sigma}_{pos,n-1}\mathbf{A}^T + \mathbf{Q} \tag{6.5}$$

$$\mathbf{\Sigma}_{pos,n} \geq ((\mathbf{\Sigma}_{pos,pred,n})^{-1} + \mathbf{F}_{pos,n})^{-1} \tag{6.6}$$

Based on the position CRLB, the lower bound for the position MSE can be calculated as:

$$MSE_n \geq \sigma_{x,n}^2 + \sigma_{y,n}^2 \tag{6.7}$$

This posterior MSE was plotted alongside the error plots from the analysis, to illustrate a theoretical lower bound for the position estimation. It should be noted that only the measurement SNR is considered with this variant of the CRLB, neglecting all other sources of error and hence estimating a comparably low bound for measurements with high multipath or shadowing effects. For ideal tracking cases, this lower bound matches the achieved errors quite well, as will be shown shortly.

6.5 Example Run

To visualize the behaviour of the filters on the real-world data, two trajectories are shown and discussed here. One example trajectory is chosen as on-body measurement, the other one is a robot trajectory with an NLOS section.

6.5.1 Trajectory 1, On-Body

This trajectory illustrates effects introduced by the human body quite well, with the agent being shadowed from the anchors by the body on the first half of the trajectory, and then facing them on the second half. No obstruction was used on this trajectory. The trajectory starts at the rear end of the car, walking around the car towards the driver's door, turning around and walking back behind the car.

Figures 6.4 shows the reference trajectory, with the numbers denoting the timesteps next to the track. Figure 6.5 shows the resulting filter estimates for the given ranging data on this trajectory. While the plot does not allow for a statistical analysis of the filter performance, a general idea of how the filters behave under the given conditions is presented. The EKF and Particle filter show quite a random behaviour, given the data, and both lose the track. The PDA filters manage to track the agent more reliably, with the filter variants using a

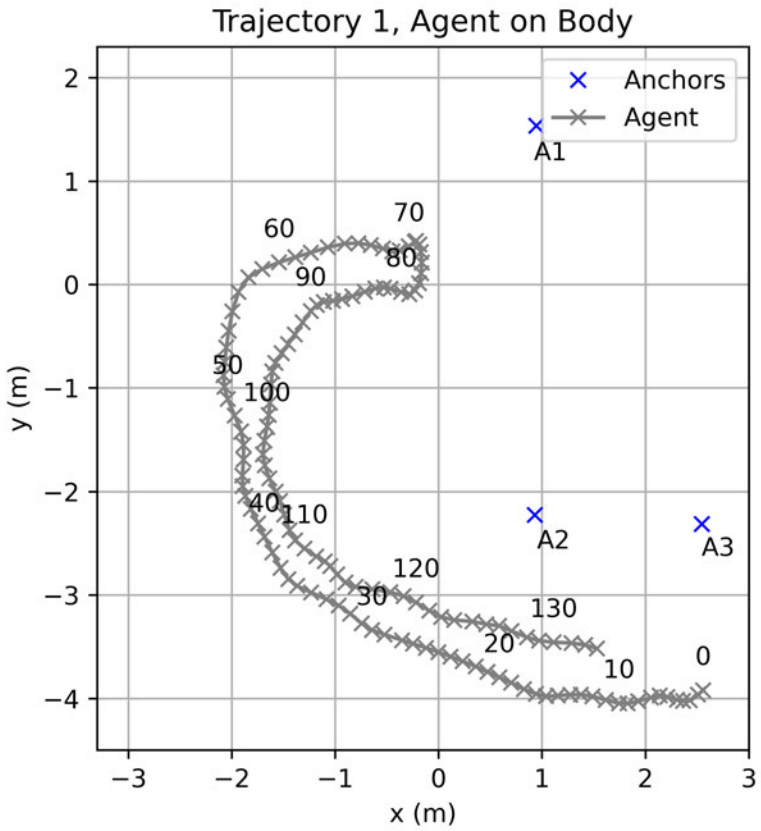


Figure 6.4: Reference path for Trajectory 1.

tracked P_D staying closest.

The behaviour of the tracked P_D is shown in Figure 6.6. The plots illustrate how the estimated likelihood of detection falls for sections of the trajectory were the strongest CEDA estimate do not match the reference distance, indicating partially (degraded, but present LOS component) or fully (no detectable LOS component) shadowed LOS paths between agent and anchor. The timesteps were the Ranger4 chip returned a distance estimate of 0 m indicate unsuccessful ranging sequences, meaning no communication between the two nodes.

Figures 6.7 and 6.8 show the results of the CEDA for Anchor 3. The deviating trajectories for the PDA filters can be seen where no LOS measurements were detected, for example between timesteps 40 to 55. In the bottom plot, the behaviour of the particles for the PDA filters are compared. While both filters diverge around timestep 40, different effects can be observed around timestep 70. The filter variant with the fixed P_D diverges faster due

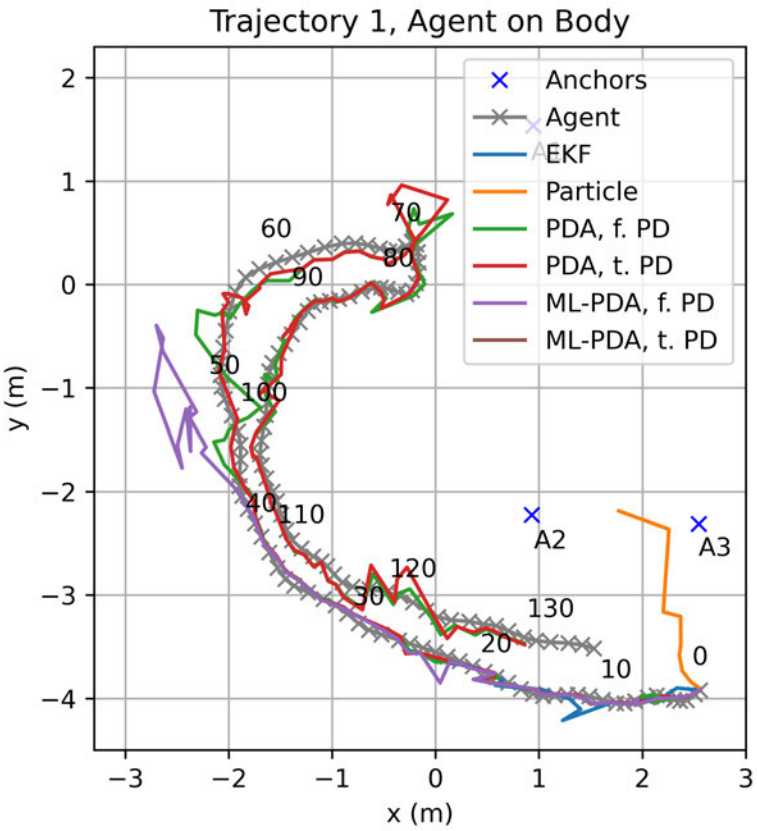


Figure 6.5: The filter trajectories for Trajectory 1.

to additional measurements giving weight to particles (around timestep 72), while the P_D tracking tunes the measurement likelihood down for the tracking model and follows the geometric proposition supplied by the state-space model. For both filters, the particles concentrate around the track again after the LOS measurements return.

Figure 6.9 shows the same plot as in Figure 6.8, only for the two ML-PDA variants. For the fixed-P_D variant, the additional likelihood evaluated from the GP predictions seems to constrain the particles, compared to the standard variant. The combination of the GP likelihood with the tracked P_D seems to worsen the effect of the NLOS section, causing the filter to lose the track quickly. The assumption is that due to the scaling of the likelihoods via the P_D as described in Chapter 3.6, the filter relies largely on the GP likelihood if the P_D drops. If the trajectory proposed by the GP estimates does not fit the CEDA estimates, this causes a downward spiral: the low P_D causes the measurements to be weighted less, which in turn causes the P_D to fall even more. While the filter recovers again around timestep 60 for the first NLOS section, the track is lost again after the second NLOS section. Due to the

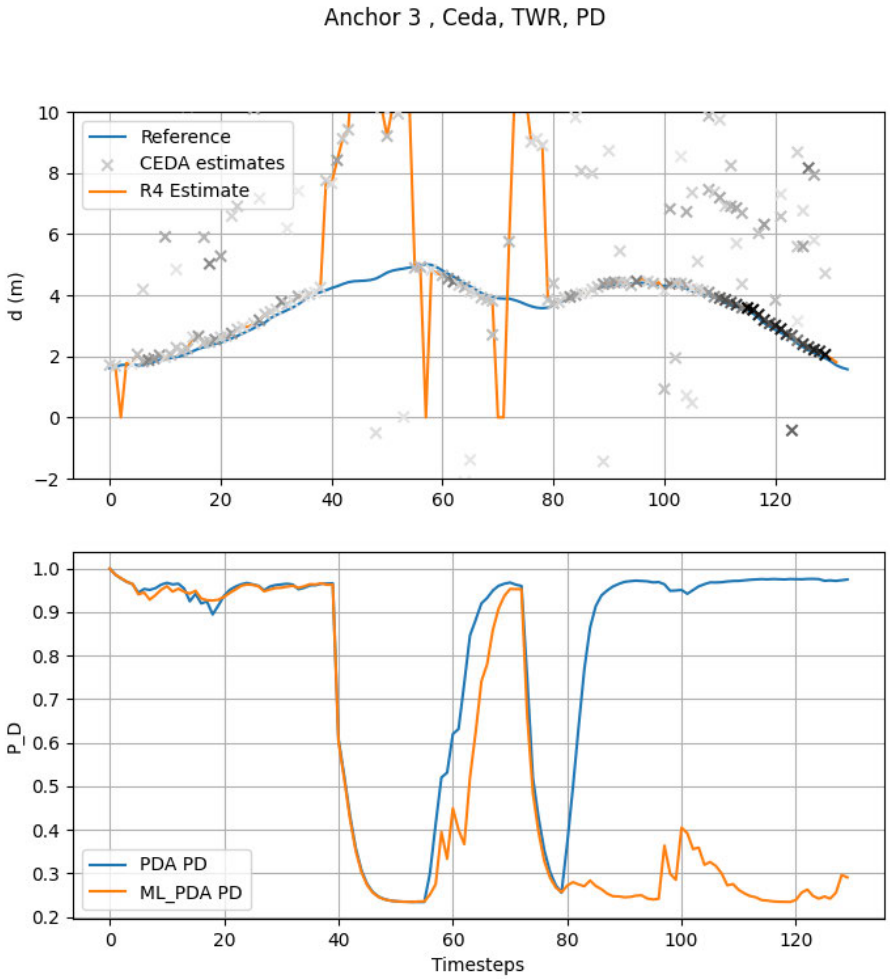


Figure 6.6: The ranging measurements with the reference distance and the P_D for Anchor 3.

weighting with the GP likelihood, the particles do not diverge wide enough to re-capture the track.

The final evaluation is done by calculating the position estimation error. The estimation errors, together with the position CRLB, are plotted in Figure 6.10. The EKF and the Particle filter perform, as expected, worst. The Particle filter loses the track almost immediately, the EKF deviates quite badly for the NLOS sections also. Taking a look at the PDA filter variants, the filters without the additional GP likelihood perform better. The ML-augmented filters were expected to perform less ideal for the on-body trajectories, since the training data was taken with the robot and not on a human body, making this a scenario with limited match to the training. The table below shows the RMSE for the six filters for this

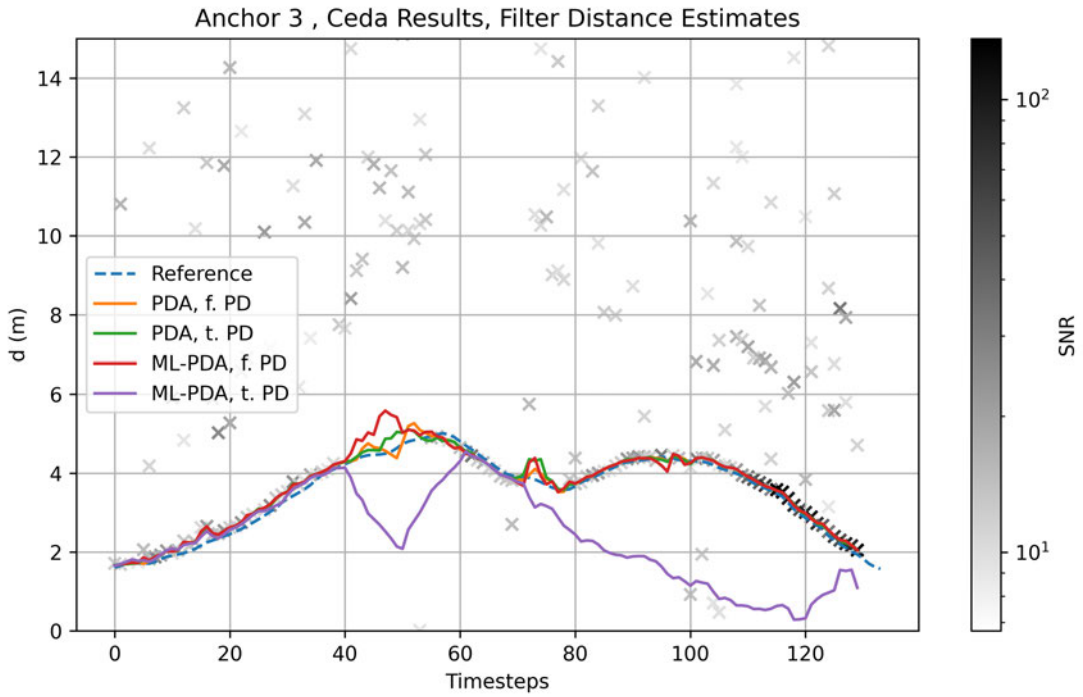


Figure 6.7: The CEDA estimates with the filters for Anchor 3.

trajectory:

Filter	RMSE (m)
EKF	1.725
Part	2.067
PDA, f. PD	0.181
PDA, t. PD	0.262
ML-PDA, f. PD	0.362
ML-PDA, f. PD	0.561

Table 6.2: RMSE results for Trajectory 1.

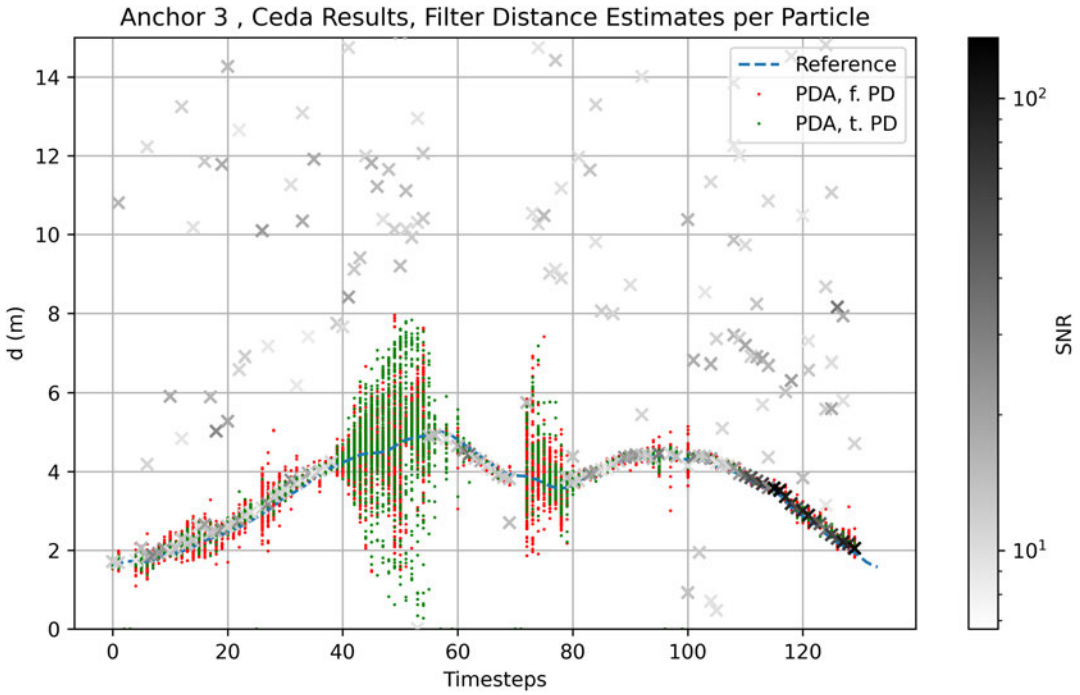


Figure 6.8: The CEDA estimates with the filters for Anchor 3.

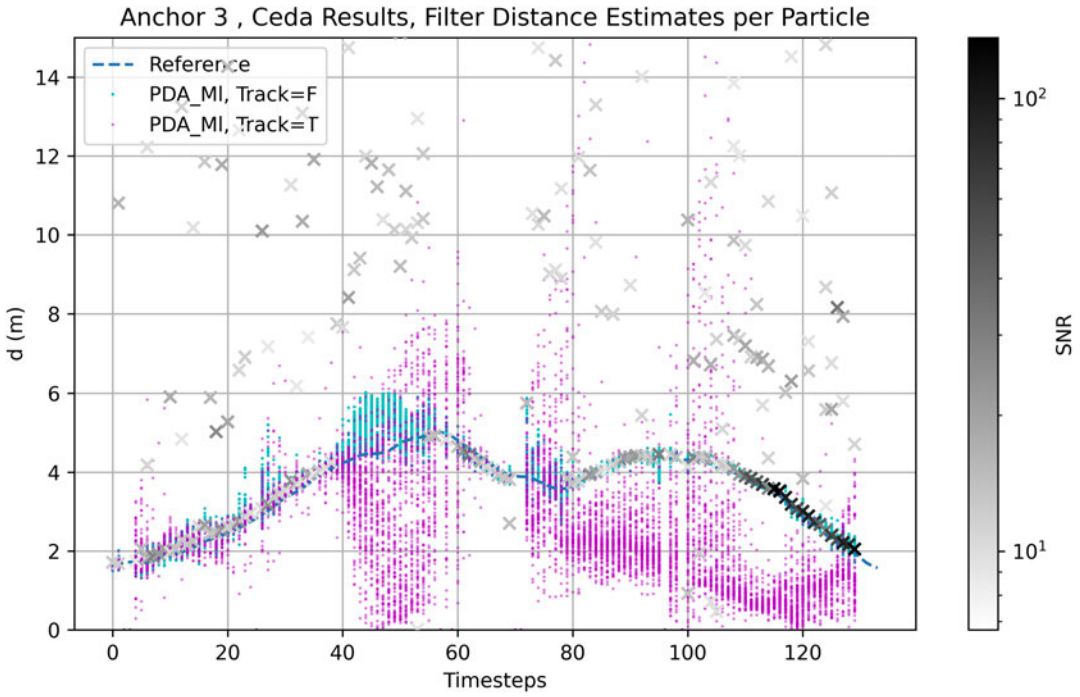


Figure 6.9: The CEDA estimates with the filters for Anchor 3.

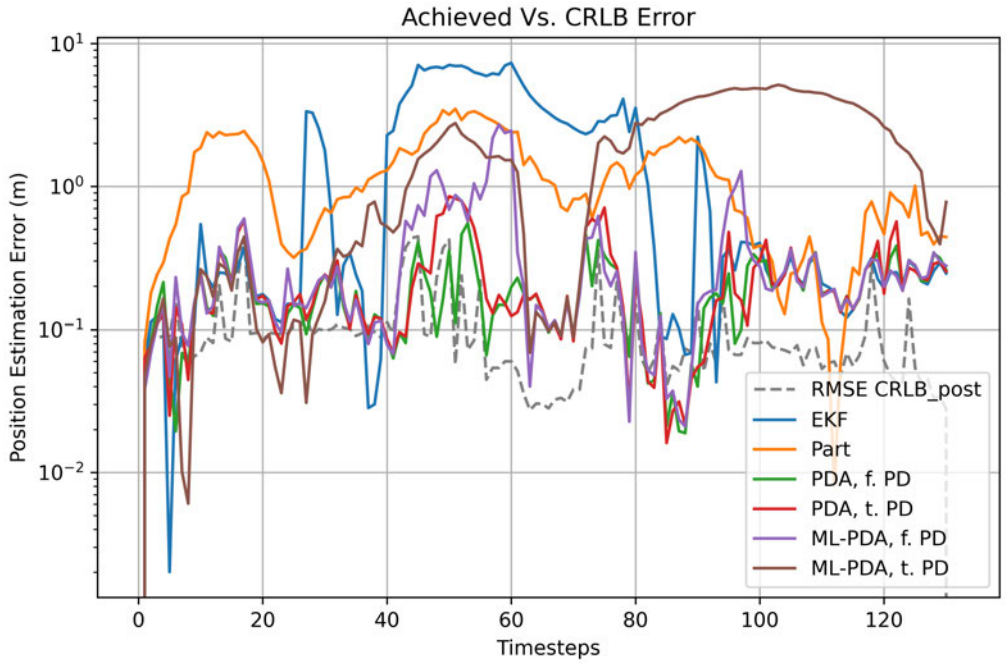


Figure 6.10: Error analysis of the six filters.

6.5.2 Trajectory 9, Robot with NLOS modification

The second example trajectory is the first robot trajectory, modified to a full NLOS situations from timestep 70 to timestep 100. This was chosen to show the effect of the artificially created NLOS section, with a complete loss of the LOS component for 30 timesteps.

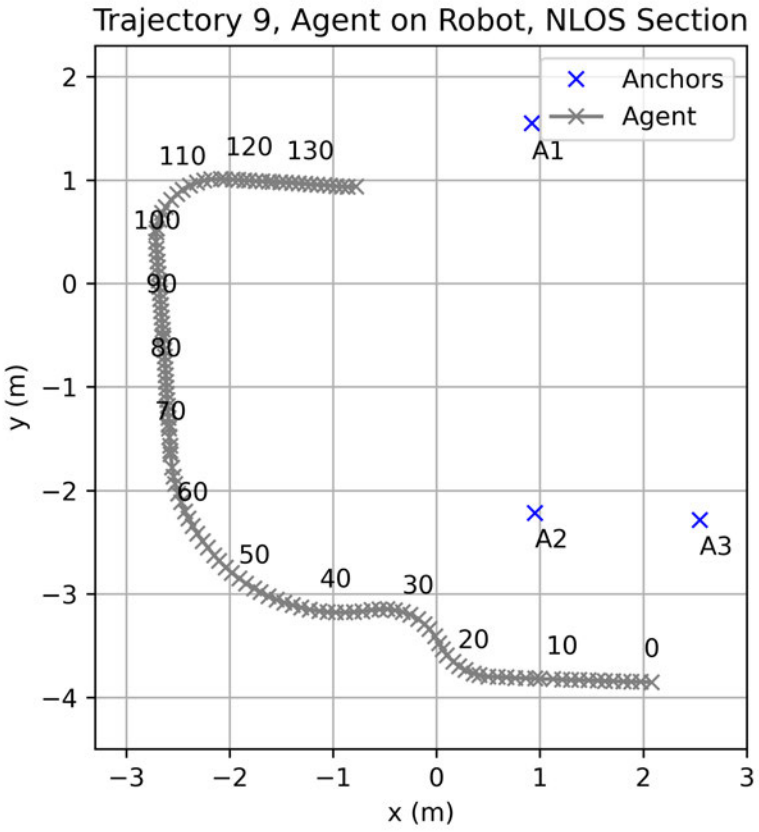


Figure 6.11: Reference path for Trajectory 9.

Figures 6.11 shows the reference trajectory, with the numbers denoting the timesteps next to the track. The NLOS section is located at the straight part of the trajectory, meaning the dynamic model should match the actual movement. Figure 6.12 shows the resulting filter estimates for the given ranging data on this trajectory. Again, no qualitative assessment can be done with the plot alone. Again, the Particle filter and EKF quickly lose the track.

The behaviour of the tracked P_D is shown in Figure 6.13. The P_D drops quickly after timestep 70, indicating a correct detection of the NLOS measurements. Again, the P_D of the ML-PDA does not recover from a section of suboptimal measurements, staying low for

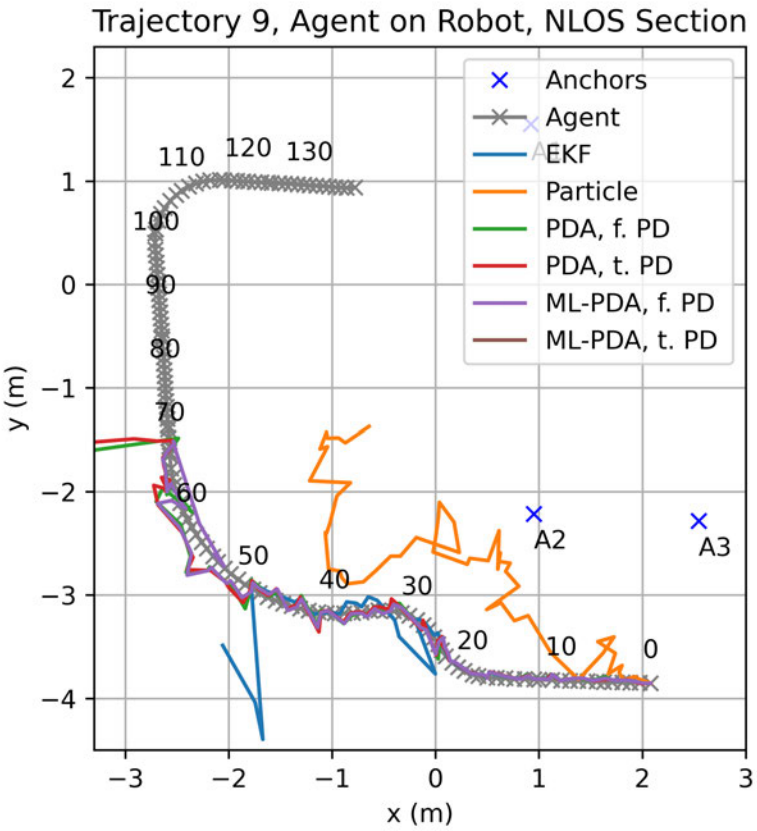


Figure 6.12: The filter trajectories for Trajectory 9.

the rest of the scenario, mainly following the GP likelihood instead of the measurement likelihood, although measurements would be available.

Figures 6.14 and 6.15 show the results of the CEDA for Anchor 3. The plots show how the PDA filter variants lose the track for the NLOS section, but can mostly recover the track after LOS measurements become available again. Figure 6.15 illustrates how both the tracked and the fixed P_D PDA filters diverge from the true track, with the particles spreading. The particles of the fixed P_D filter spread among the available CEDA estimates, while the other filter just spreads according to the dynamic model. Since close to no weighting is happening for the tracked filter, the particles reconverge much quicker when measurements become available, compared to the fixed P_D PDA filter.

The ML-PDA filters behave differently, as shown in Figure 6.15: The particles of the fixed-P_D filter stay much tighter throughout the NLOS section, quickly recovering the path again. Due to the low P_D of the tracked-P_D ML-PDA filter, the particles do not concentrate around the trajectory again, instead relying on the GP likelihood and the dynamic model.

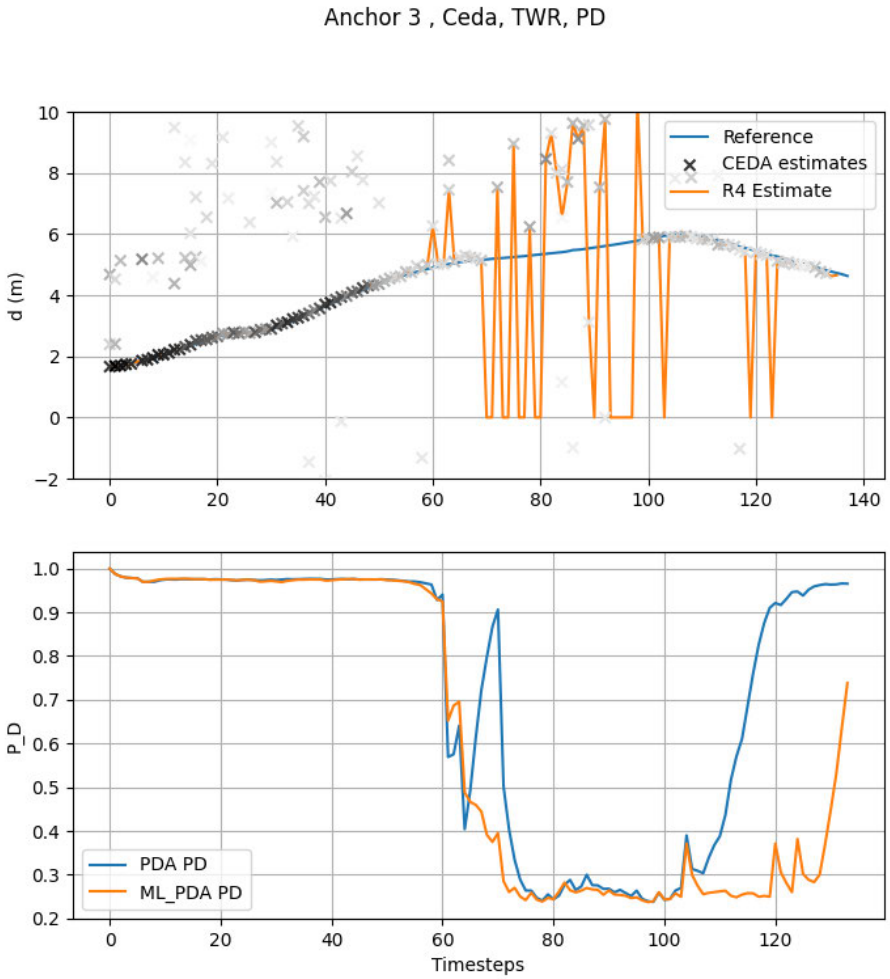


Figure 6.13: The ranging measurements with the reference distance and the P_D for all Anchor 3.

The numerical evaluation can be found in Figure 6.17. The EKF and the Particle filter again perform worst. The results are similar as before, although, as shown in the table below, a small performance gain was found for the fixed-P_D ML-PDA filter with this trajectory.

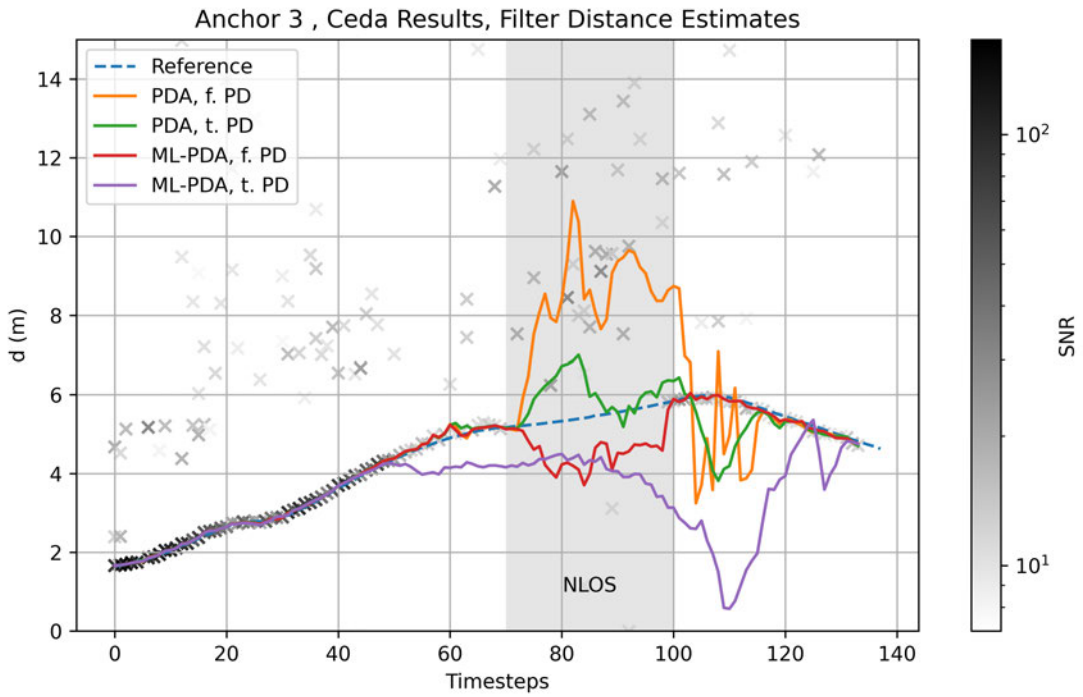


Figure 6.14: The CEDA estimates with the filters for Anchor 3.

Filter	RMSE (m)
EKF	1.055
Part	7.223
PDA, f. PD	2.682
PDA, t. PD	0.599
ML-PDA, f. PD	0.515
ML-PDA, f. PD	1.377

Table 6.3: RMSE results for Trajectory 9.

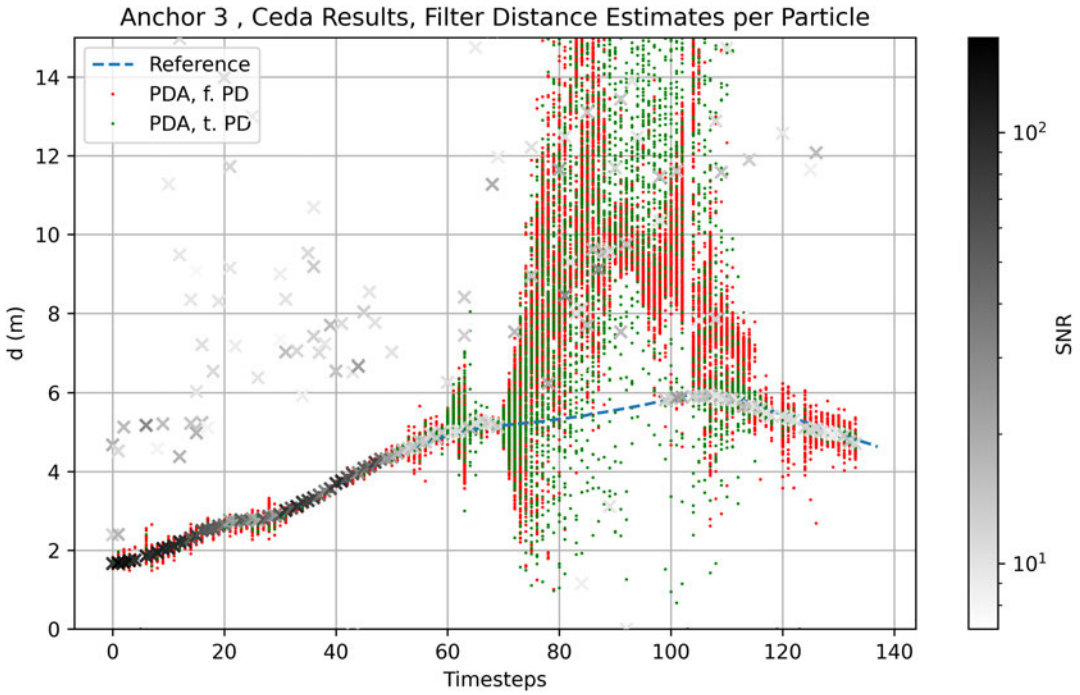


Figure 6.15: The CEDA estimates with the filters for Anchor 3.

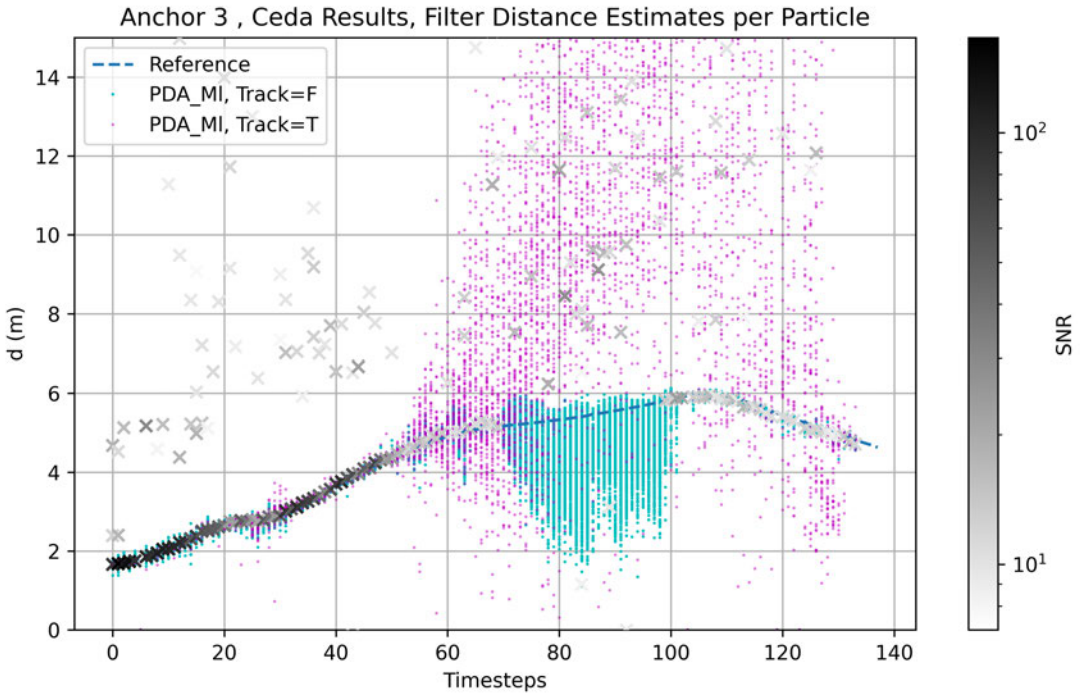


Figure 6.16: The CEDA estimates with the filters for Anchor 3.

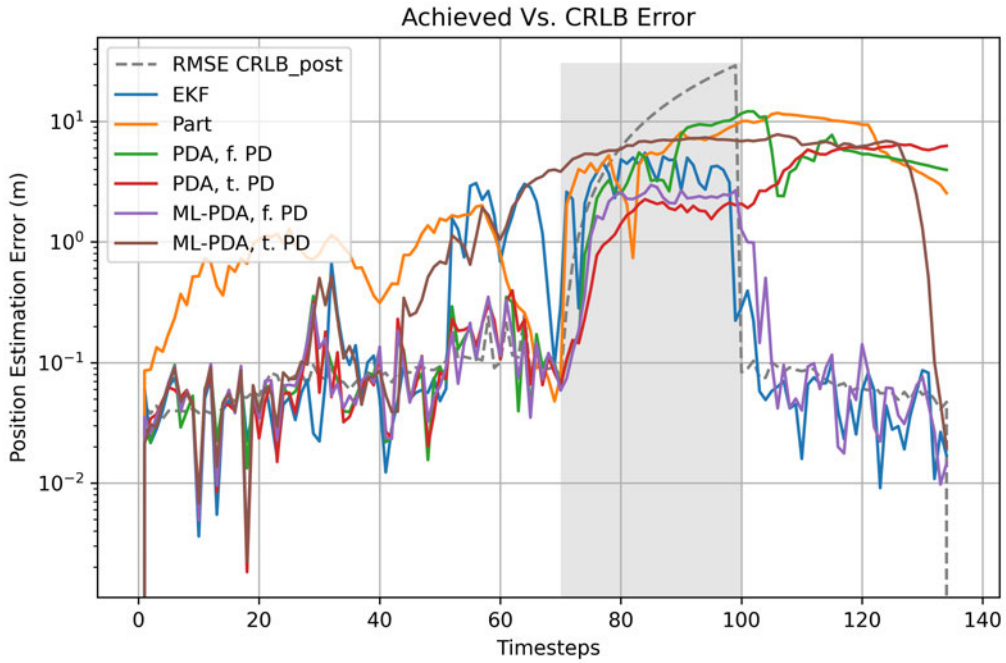


Figure 6.17: Error analysis of the six filters.

6.6 Results

6.6.1 Entire Dataset

The overall analysis of the six filters with the averaged RMSE values, as well as the three trajectory groups (On-Body, Robot, Robot with NLOS section) are summarized in table 6.4.

Filter RMSE (m)	Entire Dataset	On-Body	Robot	Robot, NLOS section
EKF	2.082	2.685	1.457	2.070
Part	4.019	1.387	2.554	8.756
PDA, f. PD	1.740	0.433	1.190	3.753
PDA, t. PD	0.883	0.3807	0.857	1.517
ML-PDA, f. PD	0.283	0.333	0.109	0.373
ML-PDA, t. PD	1.953	1.043	2.082	2.854

Table 6.4: Mean RMSE results for the analysis.

Across all trajectories, the ML-PDA with fixed P_D performs best, showing 0.5 m less averaged RMSE over the entire set of the test trajectories than the second-best filter, the PDA filter with tracked P_D. The ML-PDA filter with tracked P_D shows comparably bad performance, being outperformed by the PDA filter with fixed P_D for all types of trajectories except for the trajectories with artificial NLOS sections. While these result cannot be entirely explained, the interaction of the P_D tracking with the GP-likelihood does not seem to be beneficial to the overall tracking result. As explained above, a mismatch between GP-likelihood and CEDA estimates might lead to a degradation of the P_D, causing the loss of the track. Further work would likely lead to better understanding of this combined filter, potentially improving its performance beyond the fixed-P_D variant. An error in the implementation of the tracked-P_D ML-PDA filter can also not be ruled out, since the time invested into filter development was limited. The particle filter performs worst, with almost the RMSE being almost 2m higher than for the EKF. This indicates too few particles for the scenario, causing the filter to easily lose the track. A low amount of particles was chosen on purpose, to compare the (also particle-based) PDA filters with the Particle filters for applications with limited processing power.

Figure 6.18 shows a cumulative density function (CDF) plot for the RMSE errors. The distribution of error magnitude can be compared with this plot, with a quicker rise of

the curve meaning more estimations with a small error magnitude. It can be seen how the fixed-P_D ML-PDA filter reduces errors over 90% of the errors to below 1 m, while the fixed-P_D PDA filter only manages a 90% error rate of 6 m.

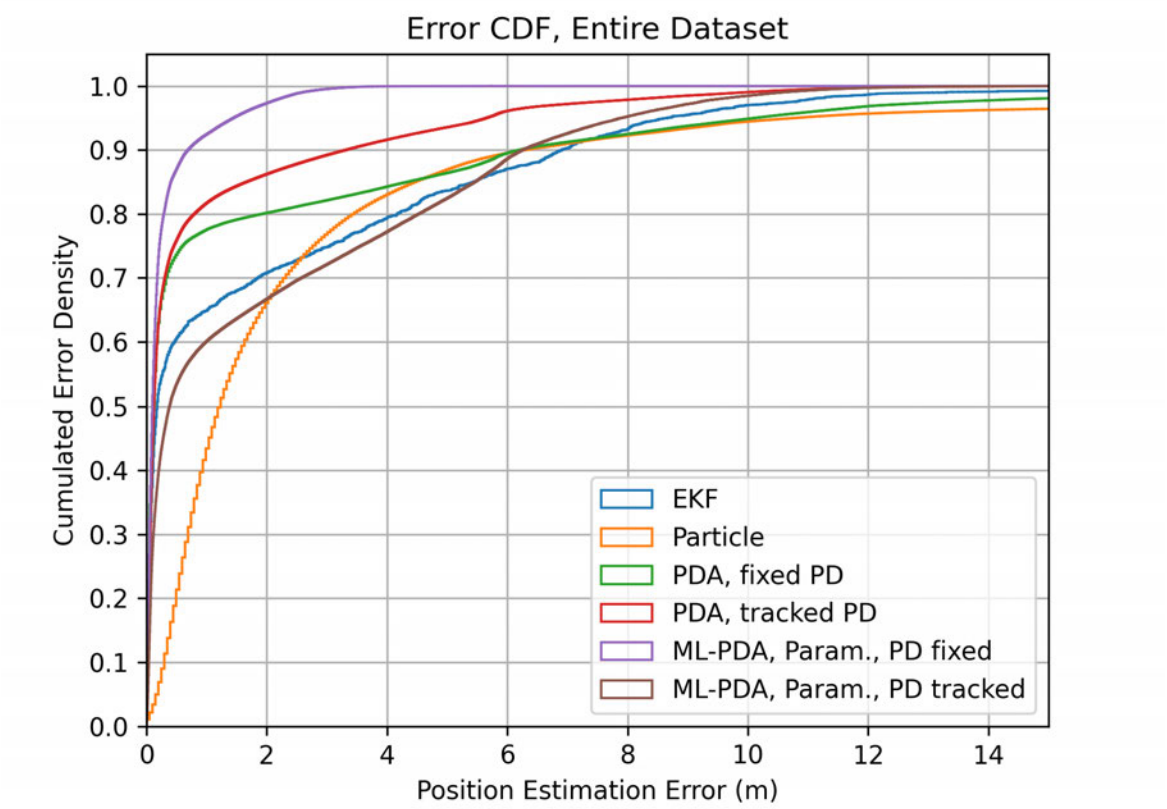


Figure 6.18: Error CDF plot for the six filters, all trajectories.

6.6.2 On-Body Trajectories

Figure 6.19 shows the RMSE CDF plot for the on-body trajectories. The tracked-P_D PDA filter and the fixed-P_D ML-PDA filter show very similar performance, with almost identical error distributions. The performance of the fixed-P_D ML-PDA filter is unexpected, because of the mismatch in scenario between the training data and the trajectory. The fixed-P_D PDA filter also performs well. The EKF shows worse performance compared to the entire dataset. Table 6.5 shows the RMSE results for the individual trajectories. Trajectories 1 and 3 show higher error rates than the other two trajectories. This is due to the stronger shadowing effects from carrying the agent at the side of the body.

Filter RMSE (m)	Traj. 1	Traj. 2	Traj. 3	Traj. 4
EKF	3.781	0.980	2.486	3.495
Part	1.370	1.535	1.535	1.492
PDA, f. PD	0.261	0.288	0.918	0.266
PDA, t. PD	0.356	0.289	0.563	0.315
ML-PDA, f. PD	0.436	0.349	0.278	0.269
ML-PDA, t. PD	0.698	1.098	1.493	0.887

Table 6.5: RMSE results for the on-body trajectories.

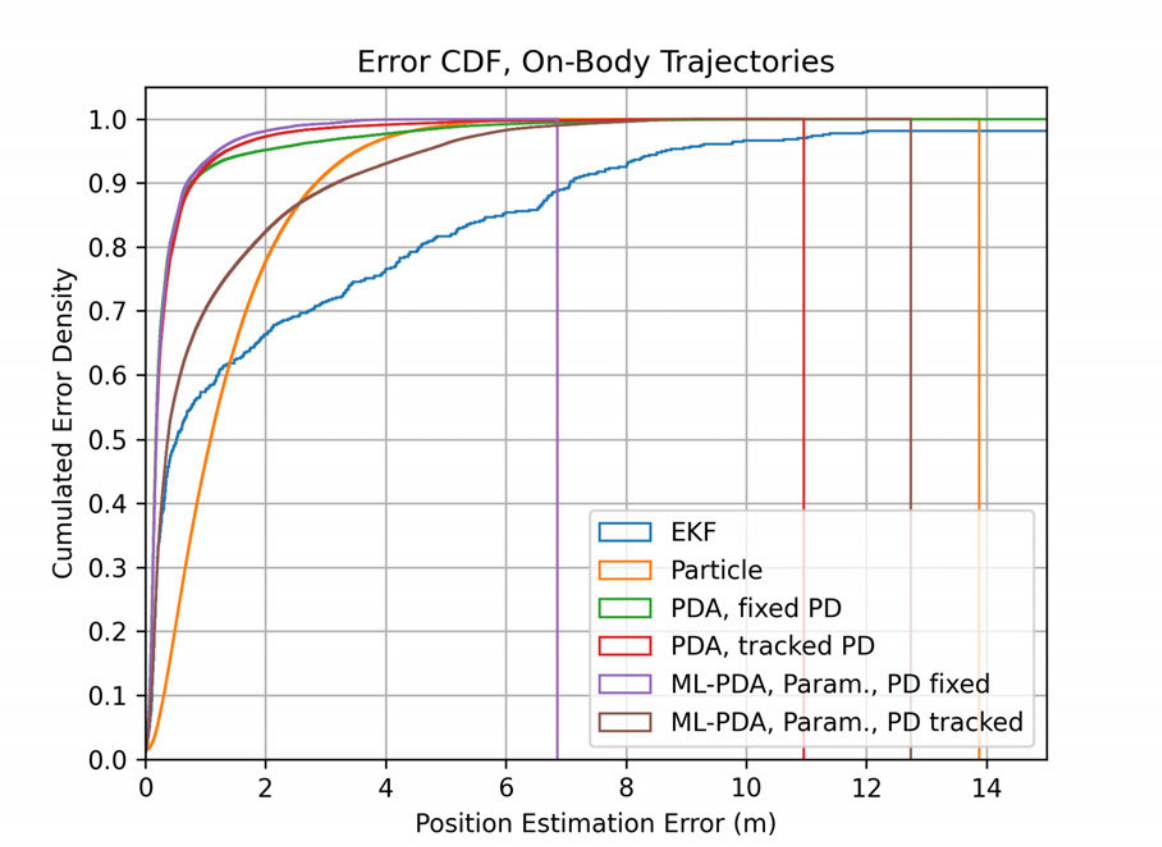


Figure 6.19: Error CDF plot for the six filters, on-body trajectories.

6.6.3 Robot Trajectories

In Figure 6.20, the RMSE CDF for the robot trajectories are shown. The fixed-P_D ML-PDA filter performs best, with no error exceeding 4m. The two PDA filters behave similarly, with

the tracked-P_D variant performing slightly better. The Particle filter shows many larger errors, as with all the other scenarios also. The EKF performs better than for the on-body trajectories. Table 6.6 shows that a larger part of the mean error shown above originates from Trajectory 7. This is the trajectory with the additional dynamic human obstruction between anchors and agent, meaning a person was actively shadowing the agent from the anchors. This is also cause of the overall worse performance compared to the on-body trajectories shown above.

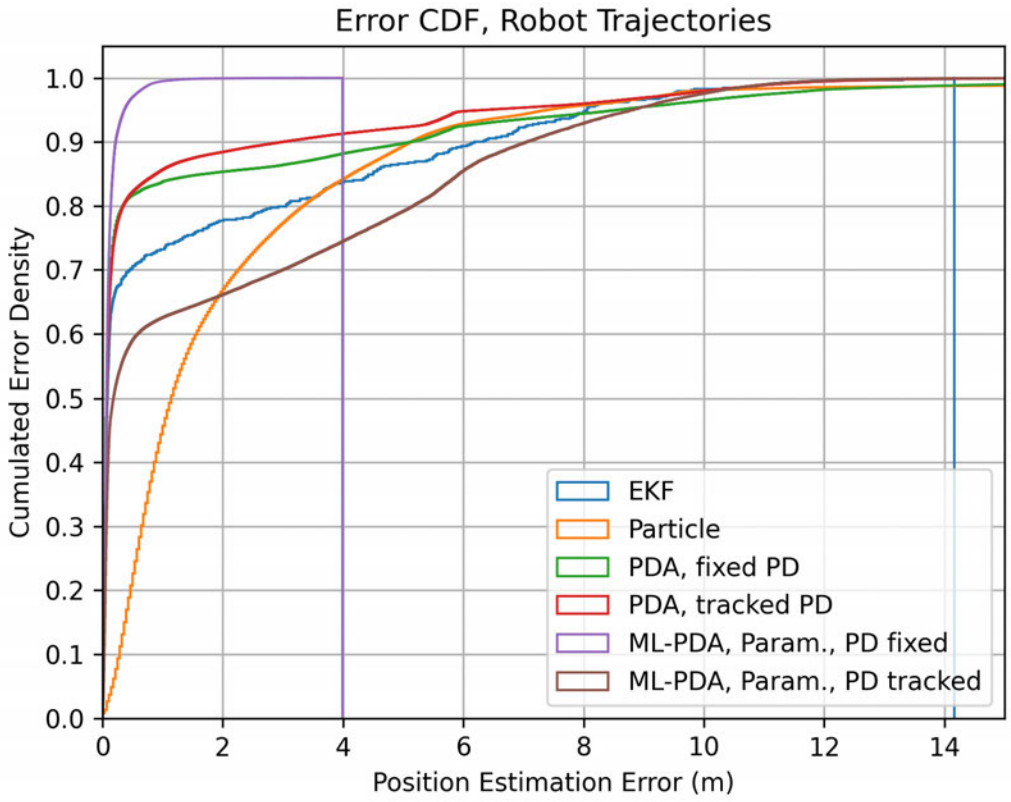


Figure 6.20: Error CDF plot for the six filters, robot trajectories.

6.6.4 Robot Trajectories with NLOS Section

The RMSE CDF of the robot trajectories with NLOS sections are shown in Figure 6.21. The NLOS section covers roughly 1/4 of the trajectories, causing high errors on this section and raising the amount of higher errors. Again, fixed-P_D ML-PDA filter performs best, followed by the tracked-P_D PDA filter. A larger gap between the fixed-P_D and tracked-P_D PDA filters

Filter RMSE (m)	Traj. 5	Traj. 6	Traj. 7	Traj. 8
EKF	0.997	1.148	3.631	0.054
Part	1.268	2.549	5.580	0.820
PDA, f. PD	0.095	0.740	3.869	0.058
PDA, t. PD	0.101	0.509	2.764	0.057
ML-PDA, f. PD	0.099	0.148	0.136	0.057
ML-PDA, t. PD	1.118	2.528	4.447	0.237

Table 6.6: RMSE results for the robot trajectories.

is observable, showing that the P_D tracking results in a performance gain especially under long NLOS situations. The EKF outperforms the fixed-P_D PDA filter, indicating a large amount of lost tracks for the PDA filter. Table 6.7 shows the mean RMSE for the individual trajectories.

Filter RMSE (m)	Traj. 9	Traj. 10	Traj. 11	Traj. 12
EKF	2.206	1.768	2.173	2.271
Part	6.197	7.7679	7.896	10.696
PDA, f. PD	3.132	3.899	3.845	3.515
PDA, t. PD	1.089	1.460	1.948	1.145
ML-PDA, f. PD	0.505	0.438	0.309	0.374
ML-PDA, t. PD	2.368	2.715	3.883	1.965

Table 6.7: RMSE results for the robot trajectories with NLOS section.

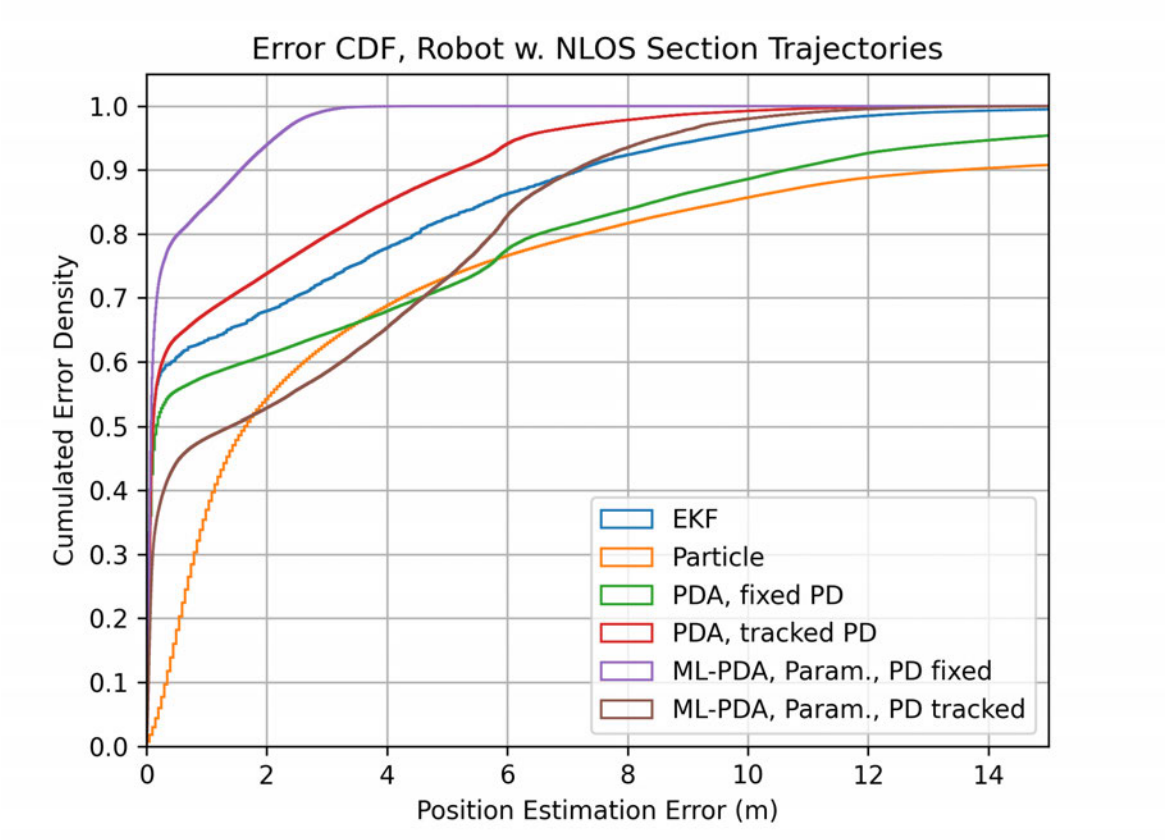


Figure 6.21: Error CDF plot for the six filters, robot trajectories with NLOS section.

7 Conclusion

The goals of this master thesis were to develop a real-time wireless data acquisition system, based on the NXP Ranger4 UWB chip, collect real-world test data and test and compare a set of tracking algorithms on the acquired datasets. The basics of tracking were briefly discussed, and the used algorithms were presented based on literature. Two reference filters, the EKF and the Particle filter, were discussed, as well as a message-passing based Probabilistic Data Association filter, also based on particles. The factor graph for the message-passing algorithm was discussed. A novel machine learning-based augmentation of the filter was presented, based on ML approaches found in [4], [5] and also [2], [3].
The data acquisition system was developed based on a previous project's hardware, upgraded and extensively tested. The measurement system was set up in combination with an Optitrack system provided by NXP, to generate reference position data to enable algorithm performance evaluation. A robot was added to capture trajectories without human body interference. A setup containing an obstruction and a car was designed for the measurements to be taken, loosely imitating a smart car access scenario. Multiple trajectories on the human body and on the robot were taken, as well as a training dataset covering most of the relevant area around the car.
To compare the different algorithms, six filter variants were run in a Monte Carlo simulation with added AWGN and 500 runs to evaluate the performance under varying real-world conditions. The analysis showed a performance gain achieved by the ML-probabilistic data association filter with fixed probability of LOS detection P_D, with all PDA variants outperforming the EKF and Particle filter in all scenarios. The performance of the tracked-P_D ML-PDA filter was found to be worse than a fixed-P_D PDA filter, with the poorer performance being accredited to a potential negative feedback loop between the tracking of the P_D and a mismatch in channel estimation and detection algorithm based likelihood and the GP-based likelihood.

The developed measurement hardware and software aims to facilitate future work in the area of UWB ranging and positioning, enabling quick and easy capturing of dynamic

trajectories. Further work on the ML-augmented PDA filters would potentially improve both the P_D-tracked as well as the fixed-P_D variants even further, with focus on the fusion of the information from the GP in NLOS situations. This was not developed further in this thesis due to time and resource constrains, but the combination of tracking filters and ML might hold potential as a novel approach.

Bibliography

[1] Z. Sahinoglu, S. Gezici, and I. Guvenc, *Ultra-wideband Positioning Systems*, First edition. Cambridge University Press, 2008.

[2] J. Vieira, E. Leitinger, M. Sarajlic, X. Li, and F. Tufvesson, "Deep convolutional neural networks for massive MIMO fingerprint-based positioning," Oct. 2017. DOI: 10.1109/PIMRC.2017.8292280.

[3] V. Savic and E. G. Larsson, "Fingerprinting-based positioning in distributed massive MIMO systems," in *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, 2015, pp. 1–5. DOI: 10.1109/VTCFall.2015.7390953.

[4] H. Wymeersch, S. Marano, W. M. Gifford, and M. Z. Win, "A machine learning approach to ranging error mitigation for UWB localization," in *IEEE Transactions on Communications*, vol. 60, 2012, pp. 1719–1728. DOI: 10.1109/TCOMM.2012.042712.110035.

[5] S. Maranò, W. M. Gifford, H. Wymeersch, and M. Z. Win, "Nlos identification and mitigation for localization based on UWB experimental data," in *IEEE Journal on Selected Areas in Communications*, vol. 28, 2010, pp. 1026–1035. DOI: 10.1109/JSAC.2010.100907.

[6] T. Gailhofer, "Measurement Setup with the NXP Ranger4 UWB Chips," SPSC, TU Graz, Master-Seminarprojekt, 2022.

[7] "IEEE standard for information technology– local and metropolitan area networks– specific requirements– part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs): Amendment 1: Add alternate PHYs," pp. 1–210, 2007. DOI: 10.1109/IEEESTD.2007.4299496.

[8] T. Wilding, E. Leitinger, U. Muehlmann, and K. Witrisal, "Modeling human body influence in UWB channels," in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, 2020, pp. 1–6. DOI: 10.1109/PIMRC48278.2020.9217057.

[9] K. Haneda, K.-i. Takizawa, J.-i. Takada, M. Dashti, and P. Vainikainen, "Performance evaluation of threshold-based UWB ranging methods - leading edge vs. search back -," in *2009 3rd European Conference on Antennas and Propagation*, 2009, pp. 3673–3677.

[10] NXP Semiconductors, *NCJ29D5 Ultra-Wideband ICs for Automotive Applications*, 2021.

[11] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation: Theory, Algorithms and Software*. Jan. 2004, ISBN: 047141655X. DOI: 10.1002/0471221279.ch11.

[12] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002. DOI: 10.1109/78.978374.

[13] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall, 1997.

[14] A. Venus, E. Leitinger, S. Tertinek, and K. Witrisal, "A message passing based adaptive PDA algorithm for robust radio-based localization and tracking," in *2021 IEEE Radar Conference (RadarConf21)*, 2021, pp. 1–6. DOI: 10.1109/RadarConf2147009.2021.9455311.

[15] F. Meyer, T. Kropfreiter, J. L. Williams, *et al.*, "Message passing algorithms for scalable multitarget tracking," *Proceedings of the IEEE*, vol. 106, no. 2, pp. 221–259, 2018. DOI: 10.1109/JPROC.2018.2789427.

[16] A. Venus, E. Leitinger, S. Tertinek, and K. Witrisal, "A graph-based algorithm for robust sequential localization exploiting multipath for obstructed-los-bias mitigation," in *Arxiv*, 2022. eprint: 2207.08646.

[17] Y. Zhao, F. Yin, F. Gunnarsson, M. Amirijoo, and G. Hendeby, "Gaussian process for propagation modeling and proximity reports based indoor positioning," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, 2016, pp. 1–5. DOI: 10.1109/VTCSpring.2016.7504255.

[18] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006, ISBN: 0387310738.

[19] iRobot Corporation. "iRobot® Create® 3 Educational Robot Documentation." (2023), [Online]. Available: https://iroboteducation.github.io/create3_docs/.

[20] Decawave, *Decawave WBoo2 Hardware Build Instruction*, 2014.

[21] Y. Shen and M. Z. Win, "Fundamental limits of wideband localization— part i: A general framework," vol. 56, no. 10, pp. 4956–4980, 2010. DOI: 10.1109/TIT.2010.2060110.

[22] P. Tichavsky, C. Muravchik, and A. Nehorai, "Posterior Cramer-Rao bounds for discrete-time nonlinear filtering," vol. 46, no. 5, pp. 1386–1396, 1998. DOI: 10.1109/78.668800.