ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI INGEGNERIA E ARCHITETTURA

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA E TELECOMUNICAZIONI PER L'ENERGIA

MULTIPATH TRACKING TECHNIQUES FOR MILLIMETER WAVE COMMUNICATIONS

Tesi in
Sistemi di Telecomunicazioni LM

Relatore
Chiar.mo Prof. Ing.
DAVIDE DARDARI

Correlatori
Dott. Ing.
FRANCESCO GUIDI
Dott. Ing.
ANNA GUERRA

Presentata da
ANDREA NICOLINI

Terzo periodo, appello II

Anno Accademico 2017-2018

Keywords

Millimeter wave

Multipath tracking

Sensors data fusion

Kalman filter

MATLAB

"Wherever you will go,
there's always a way".
For you, mum.

Acronyms

5G Fifth Generation

LTE Long Term Evolution

mmWave Millimeter Wave

EHF Extremely High Frequency

BS Base Station

MU Mobile User

MS Mobile Station

MIMO Multiple Input Multiple Output

AoA Angle of Arrival

LOS Line Of Sight

NLOS Non Line Of Sight

KF Kalman Filter

RMSE Root Mean Square Error

QoS Quality of Service

EKF Extended Kalman Filter

3D Three Dimensions

AoD Angle of Departure

PDF Probability Density Function

MAP Maximum A-Posteriori

MMSE Minimum Mean Square Error

RX Receiver

TX Transmitter

MEMS Micro Electro Mechanical System

IMU Inertial Measurement Unit

DoA Distance of Arrival

ToA Time of Arrival

AWGN Additive White Gaussian Noise

CSI Channel State Information

SOA State Of the Art

List of Figures

List of Tables

Contents

Abstract

L'obiettivo di questo elaborato è studiare il problema del tracciamento efficiente e continuo dell'angolo di arrivo dei cammini multipli dominanti in un canale radio ad onde millimetriche. In particolare, viene considerato uno scenario di riferimento in cui devono essere tracciati il cammino diretto da una stazione base e due cammini riflessi da ostacoli in diverse condizioni operative e di movimento dell'utente mobile. Si è assunto che l'utente mobile può effettuare delle misure rumorose di angolo di arrivo dei tre cammini, uno in linea di vista e gli altri due non in linea di vista, ed eventualmente delle misure di distanza tra esso e le tre "sorgenti" (ad esempio ricavandole da misure di potenza ricevuta). Utilizzando un modello "spazio degli stati", sono stati investigati due diversi approcci: il primo utilizza un filtraggio di Kalman direttamente sulle misure di angolo di arrivo, mentre il secondo adotta un metodo a due passi in cui lo stato è rappresentato dalle posizioni della stazione base e dei due ostacoli, dalle quali vengono valutate le stime degli angoli di arrivo. In entrambi i casi è stato investigato l'impatto che ha sulla stima la fusione dei dati ottenuti dai sensori inerziali integrati nel dispositivo, ovvero velocità angolare ed accelerazione del mobile, con le misure di angolo di arrivo. Successivamente ad una fase di modellazione matematica dei due approcci, essi sono stati implementati e testati in MATLAB, sviluppando un simulatore in cui l'utente possa scegliere il valore di vari parametri a seconda dello scenario desiderato. Le analisi effettuate hanno mostrato la robustezza delle strategie proposte in diverse condizioni operative.

Introduction

Nowadays the humankind need to exchange bigger data and information at an ever high speed, with a lower energy consumption and more reliability. The modern *Fifth Generation* (*5G*) cellular systems will require, for example, larger throughput (1Gb/s or higher) and lower latency (less than 1ms) compared to the conventional *Long Term Evolution* (*LTE*) based systems [6].

Millimeter wave (*mmWave*) communications can provide a solution for that issues, because they offer a larger and continuous bandwidth compared to the conventional micro-wave communications: in fact, the latter work typically at sub-6GHz frequencies (almost congested) with a bandwidth less than 20MHz, while the former can use *EHF* bands (e.g. 28, 38, 60 and 70GHz) with a width from 500MHz to 2GHz [1].

Despite these advantages, mmWave communications introduce some issues. First of all, due to the path loss increase with frequency, that systems need apparently an higher transmission power or a greater number of *base stations* (*BSs*) installed (due to the lower propagation range) compared to the micro-wave systems [7]. Secondly, also the penetration loss increases (due to higher reflection coefficients), so an hypothetical outdoor propagation cannot spread through buildings [7]. As before, this means the need of more base stations, so the apparently growth of the costs of the technology. Moreover, diffraction represents the weakest propagation component impairment in mmWave mobile systems due to the relatively small wavelengths involved [7].

These problems are closely related and to overcome them a typical *Multiple-Input Multiple-Output* (*MIMO*) mmWave communication system, i.e., base station and *mobile user* (*MU*), exploits the multipath behavior of the channel through adaptive beamforming using antennas arrays. Conventional MIMO systems, in fact, permit both the base station and mobile user (or *mobile station*, *MS*) to use narrow beams and steer them, in order to provide the best beam alignment and counteract the path loss with a beamforming gain [7]. However, because of the dynamics of the channel, due to the movement of the mobile user or the presence of obstacles, the propagation parameters

evolve in space and time, therefore the best beam alignment must be track. Moreover, the propagation could be charactered by multipaths that could be resolved and exploited to increase the communication capacity (MIMO) as well as its reliability.

This work aims at investigating this issue, that is how to efficiently and continuously track the *Angle-of-Arrival* (*AoA*) of the dominant multipath components using an antenna array. In particular, a reference scenario in which the direct path from the base station and up to two paths reflected by scatterers have to tracked is considered under mobile station movement and different operating conditions. Is assumed that the mobile user can do a noisy measurement of the AoAs of the three paths, one *Line-Of-Sight* (*LOS*) and the others *Non-Line-Of-Sight* (*NLOS*), and possibly of the distance from itself to the scatterers (e.g., inferred from the received power).

Using a state-space modeling, two different approaches are investigated: the first one considers a *Kalman filtering* (*KF*) directly applied to the AoA measurements, the second instead, adopts a two-state method in which the state is represented by the position of the base station and scatterers from which the AoA estimates are evaluated. In both cases the impact of fusing the angular speed and position acceleration of the mobile station obtained from an inertial device with the AoA measurements is investigated.

The following work is structured in five chapters:

- The first chapter is a short overview on the typical strategies adopted for the beam tracking. It is explained how this work takes inspiration from the existing approaches, in particular from the reference article [2], and how data fusion was integrated in them.

- In the second chapter the first approach is introduced. In particular, the reference scenario, the mobility and observation models are introduced, then the implementation of the Kalman filtering for AoA tracking is described.

- In the third chapter the second approach is presented, i.e., using the noisy measurements of the three AoAs and the distance between the mobile user and the scatterers to estimate the real values of the scatterers relative positions, and from them extrapolate the AoAs values. As before, the reference scenario, the mobility and observation models are introduced, then the implementation of filter for position tracking is described.

- The fourth chapter shows the high-level flowchart of the algorithms and gives an explanation of the MATLAB implementation.

- In the last chapter the MATLAB simulations obtained in different conditions of movement, device rotation, noise values and filter parameters are explained. Then the results in terms of *Root Mean Square Error* (*RMSE*) and outlier rate, for AoA and positions, are shown. The performance of the two approaches are compared.

Chapter 1

Overview on typical approaches for beam tracking

Fist of all, it must be said that beam tracking in mobile communications is the method that follows the initial access phase. In fact, a mobile user that enters a cell for the first time must perform a scan of the space around itself to find a base station from which to be served, like in Figure 1.1. To ensure the best possible *Quality of Service* (*QoS*), it must find the most accurate alignment and keep it despite the dynamics of the channel and user.

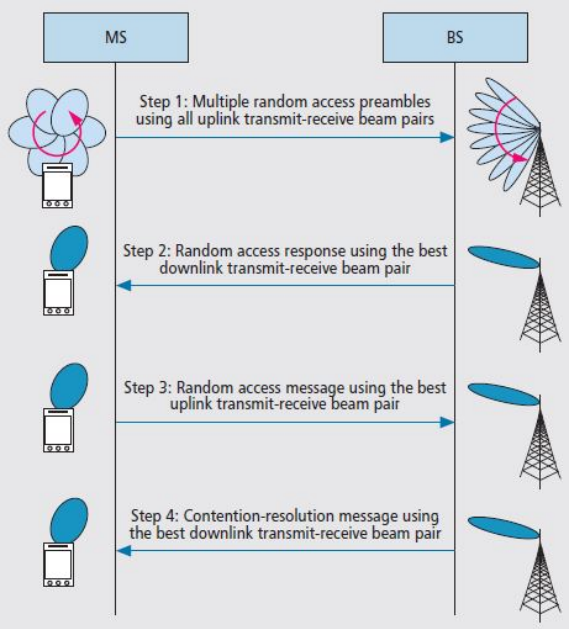

Figure 1.1: The random access procedure in mmWave beamforming cellular networks [1]

In order to ensure that alignment, it should be possible to change the direction of the beams of the antenna arrays used by the base station and possibly the mobile station. In multi-antenna systems the way to do that is choosing appropriately the coefficients (weights) of the array elements [6] [1]. Since the high path loss at mmWave, narrow beams are required to obtain an high beamforming gain and overcome this issue.

From the MIMO point of view (Figure 1.2), the multipath behavior of the channel is an important resource, because a multi-antenna system can typically exploit fading to increase capacity, range and radio link quality as a result of diversity, array and multiplexing gains, as well as interference reduction, using advanced signal processing.

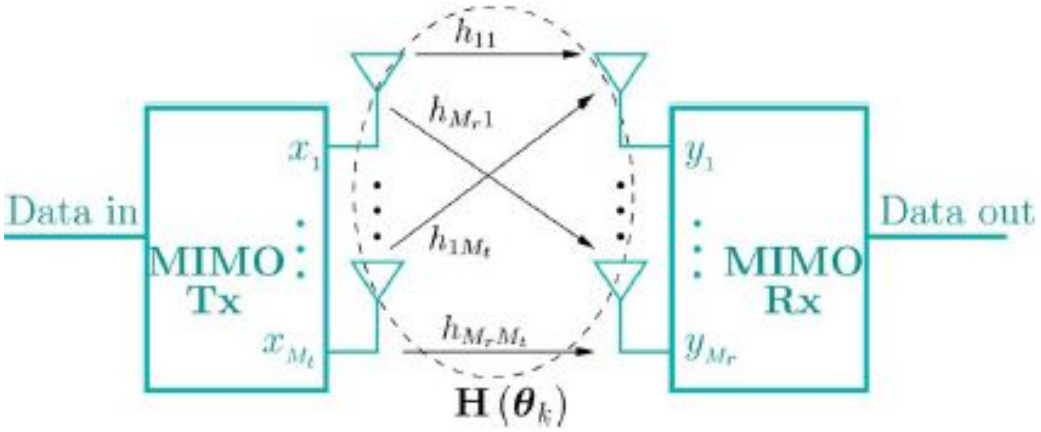

Figure 1.2: Description of the multi-antenna communication, i.e., MIMO principle, where $\boldsymbol{H}$ is the time varying channel matrix and $\boldsymbol{\theta_k}$ the parameters to estimate [2]

In other words, since at mmWave the propagation contribution is given by the LOS path (if exists) and few reflective paths, track them become fundamental for achieve the advantages cited before.

1.1 Conventional techniques for tracking of MIMO propagation path parameters

The approach from which the proposed work takes inspiration [2] is not referred to mmWave propagation, but to micro-wave propagation, for which there is an high multipath richness, especially in dense urban scenarios. Since the method proposed in [2] is targeted to a different propagation scenario,

in this thesis will be taken out only the parts that can be applied to the scenario under consideration and that represent a solid starting point.

The reference paper proposes a state-space approach for tracking the dynamic radio propagation path parameters over time, i.e., time-delays $\boldsymbol{\tau}$ and angles $\boldsymbol{\vartheta}$ and $\boldsymbol{\varphi}$ (Figure 1.3), as well as the coefficients of the array elements. A state-space model is derived and an *Extended Kalman filtering* (*EKF*) is applied for the parameter estimation of the non-linear measurement model. In general, there are several advantages in direct sequential estimate (tracking), as follows:

- Information about how parameters evolve over time, including the rate of change, can be obtained.
- Due to filtering the estimate error is reduced.
- Sequential estimate reduces the computational complexity, since the previous estimate can be updated using the new information captured by the current measurement.

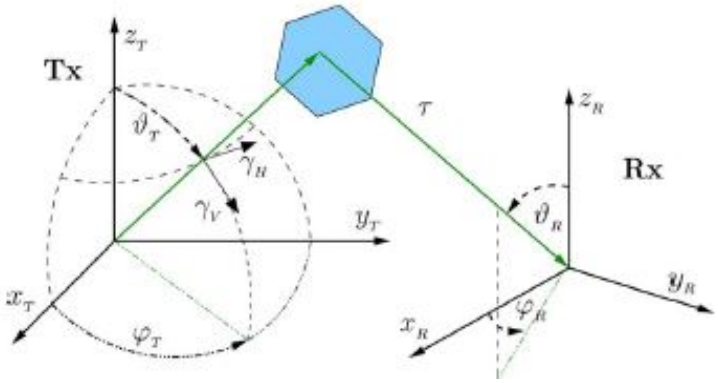

Figure 1.3: Double directional radio channel model parameters for a single propagation path [1]

The EKF is formulated in a specific, computationally attractive form, enabling scalability for large state and measurement dimensions. Realistic models for state-noise covariance and propagation are developed in order to capture the underlying physical phenomena, to keep the computational complexity reasonable, and to obtain an estimator with good statistical performance in terms of estimate error [2].

Another approach for the tracking of the dynamic space, time, and frequency dependent MIMO radio channel was proposed in [8], based on Particle Filter concept.

In the following sections the model and the estimate process will be examined.

1.1.1 State-space model

To describe the state-space model of the propagation paths, the paper [2] proposes the following structure for realistic and identifiable modeling of the dynamics. The state vector (parameters to track) at every time step k is given by:

$$\boldsymbol{\theta}_k \;=\; \left[\; \boldsymbol{\mu}^T \;\boldsymbol{\Delta\mu}^T \;\boldsymbol{\alpha}^T \;\boldsymbol{\phi}^T \;\boldsymbol{\Delta\phi}^T \;\right]^T \tag{1.1}$$

where $\boldsymbol{\phi}$ and $\boldsymbol{\Delta\phi}$ are respectively the phases of the paths and their rate of change, while $\boldsymbol{\alpha}$ are the polarimetric path coefficients. These parameters will not be taken into account for the rest of the paragraph. $\boldsymbol{\mu}$ and $\boldsymbol{\Delta\mu}$ are the vectors of parameters describing the propagation paths and their rates of change, that is:

$$\boldsymbol{\mu} \;=\; \left[\; \boldsymbol{\tau}^T \;\boldsymbol{\varphi}_T^T \;\boldsymbol{\vartheta}_T^T \;\boldsymbol{\varphi}_R^T \;\boldsymbol{\vartheta}_R^T \;\right]^T \tag{1.2}$$

where $\boldsymbol{\vartheta}$ are the elevation angles of departure/arrival and $\boldsymbol{\varphi}$ the azimuth angles of departure/arrival respectively (Figure 1.3). For simplicity and without losing in generality, the *three dimensional* (*3D*) approach from here on is replaced by a 2D approach, so elevation is not taken into account. Also the time delays of arrival $\boldsymbol{\tau}$ are not considered.

Summing up, the state vector is now given by the AoAs and *Angles-of-Departure* (*AoDs*) and their rates of change. Considering that for AoAs and AoDs the treatment is equivalent, we can consider only one of them, that is AoAs. Therefore:

$$\boldsymbol{\theta}_k \;=\; \left[\; \boldsymbol{\mu}^T \;\boldsymbol{\Delta\mu}^T \;\right]^T \;=\; \left[\; \boldsymbol{\varphi}_R^T \;\;\boldsymbol{\Delta\varphi}_R^T \;\right]^T . \tag{1.3}$$

In this way it is possible to neglect some steps followed by the reference paper, like the parametrization of the weights and the effects of the Doppler shift in channel sounding [2].

The continuous-time dynamic equation for a single path can be expressed as:

$$\frac{d\boldsymbol{\theta}(t)}{dt} \;=\; \boldsymbol{F}\;\boldsymbol{\theta}(t) \;+\; \boldsymbol{L}\;\boldsymbol{w}(t) \;=\; \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \left[\; \boldsymbol{\mu}^T(t) \;\boldsymbol{\Delta\mu}^T(t) \;\right]^T \;+\; \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \;\boldsymbol{w}(t) \tag{1.4}$$

where the rate of change of the structural parameters is perturbed by a white Normal distributed process $\boldsymbol{w}$(t) with a white power spectral density $\boldsymbol{W}$.

The choice for the process noise model is motivated by the assumption that the noise dynamics of the structural parameters are governed by the (macro-scale) motion of terminals and/or scatterers in the system. The values of $\boldsymbol{W}$ are essentially filter design parameters, so the selection and fine tuning of these should be done according to the expected kinematics in the system, as well as balancing between smoothness of the filtered estimates versus faster tracking. The former leads to losing tracking of paths, whereas the latter gives higher variance but enables robust performance and more reliable tracking. In addition, the state noise term captures all the modeling uncertainties [2].

State-space modeling of radio channel propagation parameters is based on the observation that the parameters, if properly chosen, evolve slowly with respect to the measurement interval, i.e., they are correlated in time. The process can be described using a Gauss–Markov model, i.e., denoting the state of the system at time k as in equation (1.1), with the *Probability Density Function* (*PDF*) of the next state dependent only on the current state and Normal distributed. Therefore, the state-space model used in the reference paper consists of a linear state equation, describing the dynamic behavior of the paths propagation parameters, and a non-linear measurement equation, mapping the double-directional model parameters to the channel sounder output data.

In channel sounding the observation interval is usually constant. Thus, the time between observations can be normalized to one and the discrete-time (linear and time invariant) state transition and state-noise covariance matrices can be solved in closed form using the following matrix fraction decomposition [2]:

$$\begin{aligned}\boldsymbol{Q} \;&=\; \int_0^{\Delta t} \boldsymbol{\Phi}(\Delta t-\tau)\boldsymbol{L}\boldsymbol{W}\boldsymbol{L}^T\boldsymbol{\Phi}^T(\Delta t-\tau)d\tau \;\approx \\ &\approx\; \boldsymbol{W}\left[\boldsymbol{L}\boldsymbol{L}^T\Delta t+\frac{1}{2}\Delta t^2\left(\boldsymbol{F}\boldsymbol{L}\boldsymbol{L}^T+\boldsymbol{L}\boldsymbol{L}^T\boldsymbol{F}^T\right)+\frac{1}{3}\Delta t^3\left(\boldsymbol{F}\boldsymbol{L}\boldsymbol{L}^T\boldsymbol{F}^T\right)\right] \quad (1.5)\end{aligned}$$

where Δt is the discretization interval.

The discrete-time state transition equation can be expressed as:

$$\boldsymbol{\theta}_k \;=\boldsymbol{\Phi}_k\;\boldsymbol{\theta}_{k-1}\;+\;\boldsymbol{v}_k \quad (1.6)$$

where $\boldsymbol{v}_k \sim N(0, \boldsymbol{Q})$ is the state noise with covariance matrix $\boldsymbol{Q}$ determined as in equation (1.5) and $\boldsymbol{\Phi}_k$ the state-transition matrix obtained as:

$$\boldsymbol{\Phi}_k \;=\; exp(\boldsymbol{F}\Delta t) \tag{1.7}$$

that is equal, for small Δt, to multiply every member of the matrix $\boldsymbol{F}$ by Δt and add the result to an identity matrix of the same dimension [2].

The non-linear measurement equation of the state-space model is given by:

$$\boldsymbol{y}_k \;=\; f\left(\boldsymbol{\theta}_k\right) \;+\; \boldsymbol{r}_k \tag{1.8}$$

where $f\left(\boldsymbol{\theta}_k\right)$ is a function of the propagation paths parameters which depends on the type of measurement, and the measurement noise is assumed to be $\boldsymbol{r}_k \sim N(0, \boldsymbol{R})$ with covariance matrix $\boldsymbol{R}$.

The core of proposed parameter estimation procedure (Figure 1.4), i.e., tracking the propagation path parameters over time, is based on the EKF.

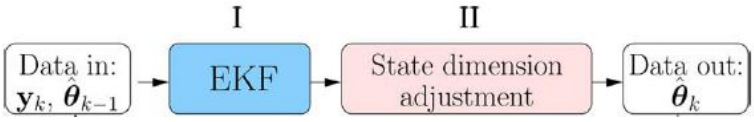

Figure 1.4: Estimation procedure principle [1]. Second step is not taking into account

The EKF uses Taylor series expansion to linearize the non-linear measurement model about the current estimates and it is a low complexity implementation of the Bayesian filtering.

The Bayesian filtering is a recursive algorithm to evaluate the marginal a-posteriori distribution $p\left(\boldsymbol{\theta}_k \mid \boldsymbol{y}_{1:k-1}\right)$ (belief) of the current state, depending on all past measurements, by exploiting the mobility model $p\left(\boldsymbol{\theta}_k \mid \boldsymbol{\theta}_{1:k-1}\right)$ and the measurement model $p\left(\boldsymbol{y}_k \mid \boldsymbol{\theta}_k\right)$, i.e., the likelihood for the measurements accounting for the observation noise, where $\boldsymbol{y}_k$ are the new incoming measurements.

The algorithm consists of three phases:

- Initialization, that is the first phase performed only once when the algorithm starts. In this step the a-priori marginal distribution $p\left(\boldsymbol{\theta}_{0:k}\right)$ is defined.

- Prediction, in which the algorithm evaluate the marginal a-posteriori distribution $p\left(\boldsymbol{\theta}_k \mid \boldsymbol{y}_{1:k-1}\right)$.

- Update, where the prediction is refined exploiting the measurement model, therefore $p\left(\boldsymbol{\theta}_k \mid \boldsymbol{y}_{1:k}\right)$ is evaluated. This last step is repeated recursively alternating with the prediction phase, until the algorithm stops.

Figure 1.5 shows the Bayesian filtering flowchart:

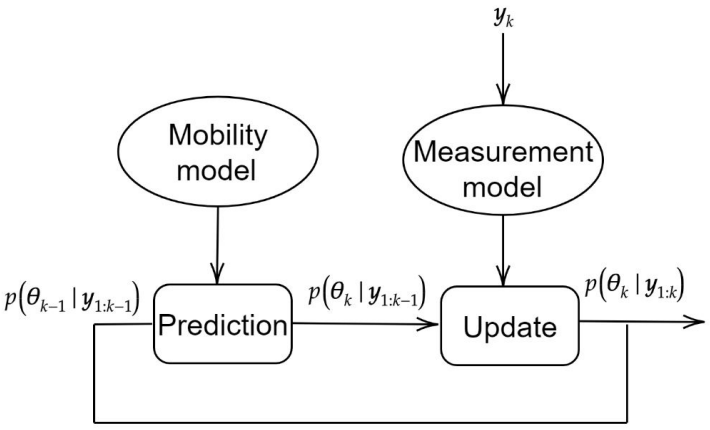

Figure 1.5: Bayesian filtering flowchart

where the following equations show the results of the prediction and update phases:

$$p\left(\boldsymbol{\theta}_k \mid \boldsymbol{y}_{1:k-1}\right) = \int p\left(\boldsymbol{\theta}_k \mid \boldsymbol{\theta}_{k-1}\right)\ p\left(\boldsymbol{\theta}_{k-1} \mid \boldsymbol{y}_{1:k-1}\right)\ d\boldsymbol{\theta}_{k-1}\ ; \tag{1.9}$$

$$p\left(\boldsymbol{\theta}_k \mid \boldsymbol{y}_{1:k}\right) = \frac{p\left(\boldsymbol{y}_k \mid \boldsymbol{\theta}_k\right)\ p\left(\boldsymbol{\theta}_k \mid \boldsymbol{y}_{1:k-1}\right)}{\int p\left(\boldsymbol{y}_k \mid \boldsymbol{\theta}_k\right)\ p\left(\boldsymbol{\theta}_k \mid \boldsymbol{y}_{1:k-1}\right)\ d\boldsymbol{\theta}_{k-1}}\ . \tag{1.10}$$

Relatively to the Kalman filtering, as it will be shown in the following chapters, the algorithm provides respectively the average value $\boldsymbol{m}_k$ and the covariance $\boldsymbol{P}_k$ of the marginal a-posteriori distribution, that is Gaussian.

In case of *Maximum A-Posteriori* (*MAP*, or *Minimum Mean Square Error*, *MMSE*) estimator, that average coincides with the estimate $\hat{\boldsymbol{\theta}}_k$ at time step k.

Therefore, under Markovian hypothesis, a Bayesian modeling of system dynamics and measurements to account for uncertainties gives an optimal, real-time and recursive solution, with low complexity despite the increase in the number of time steps [2]. To improve the filtering is possible to modify

the state model, and so the algorithm itself, introducing a term identifying an external input, that contains, for the proposed work, data from the embedded device sensors, as described in the next section.

1.2 Data fusion approach

A perfect beam alignment between the *transmitter* (TX) and the *receiver* (RX) needs fine tuning and it is vulnerable to different factors such as mobile device movement and rotation (Figure 1.6), i.e., a small misalignment between TX and RX beams may cause a significant loss in the received power, especially for systems with narrow beams.

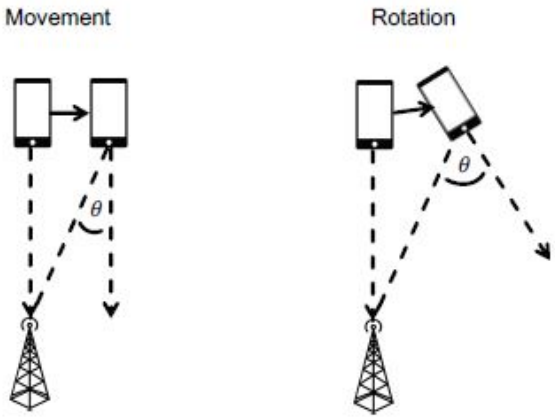

Figure 1.6: Motion misalignment in beamforming [3]

The rapid development of low-cost *Micro Electro-Mechanical Systems* ($MEMS$) provides an inexpensive solution that could improve the performance of the beam alignment. In most smart devices, the *Inertial Measurement Unit* (IMU) contains three key components: accelerometer, gyroscope and magnetometer:

- Accelerometer measures the acceleration of each axis. The problem of the accelerometer is mainly caused by additional force placed on the handset and gravity that needs to be subtracted from the model [4]. The accelerometer suffers from drifting issues due to the double integration of measurement over time.
- Gyroscope measures the angular speed in the sensor frame. The attitude is acquired by the integration of measurement over time. The

gyroscope is reasonably accurate over a short time, while the drifting issue makes it inaccurate due to the integration over time. This problem could be mitigated by the technique of sensor fusion [3].

- The magnetometer detects and measures the magnetic field. It offers a great option to measure the absolute position with respect to the Earth north pole. Yet the disadvantage is that the magnetic north pole does not align with the geographic one. Moreover, the direction varies based on nearby materials which could substantially affect the magnetic field [3].

The combination of multiple streams of sensor data for a more accurate result is called sensor (data) fusion. As shown in the literature [9] [4] [5], the accuracy of MEMS measurements suffers from the drifting and measurement noise in the long run. However, the problem is eased in the beamforming scenario, where the integration time is relatively short: within the short period, the drift accumulation is negligible [3].

1.2.1 Innovative algorithms for 3D mmWave beam tracking

An IMU includes a three-axis gyroscope, a three-axis accelerometer and a three-axis magnetometer, which are embedded in the MEMS inside the smart phones. In a 3D space, quaternion q algebra is widely used for defining the orientation of a rigid body coordinated frame [5]. Moreover, Euler angles are frequently used to specify the angular orientation of one coordinate system relative to another [9].

It is appropriate to introduce these two concepts because the two ends of the communication link only recognize the optimal direction in their own coordinate, where the coordinate transformation is unclear unless detailed geographic position and accurately environment model are given. Therefore, it is typical to define a global coordinate system where the origin is located at the base station [3] [9]. The optimal beam direction is given as mutual directions within this frame provided the positions of the mobile user and base station. Besides the global frame, there is a local coordinate system, which is attached to the mobile device rigidly and moves and rotates with the device itself. All the sensors in the mobile devices are working in this frame, while the antenna array is also placed in a certain plane in the local frame. Figure 1.7 shows how it is possible to compensate the misalignment between the global and local frame [10] .

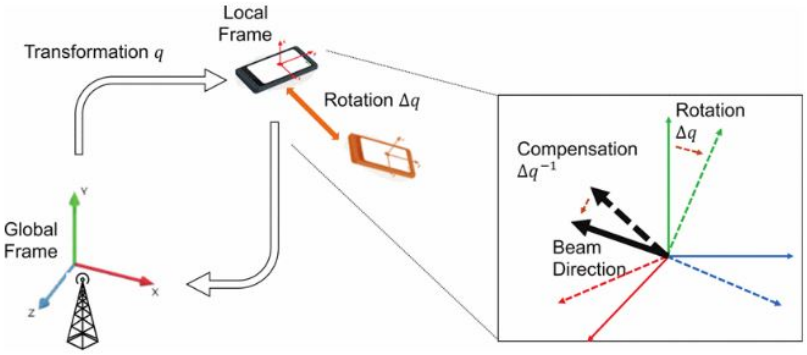

Figure 1.7: Beamforming misalignment compensation [3].

For tracking behavioral changes (self-rotation) of the mobile user, [4] [5] use the IMU of the smart phones to give self-measurements of the moving device. Although the gyroscope itself can measure the three rotation angles (yaw, pitch, roll) accurately in short time, the drift error caused by integration could cause significant errors in rotation tracking. For this reason, the data obtained from these sensors are incorporated by EKF [4] or by gradient descent algorithm [5] for tracking behavioral changes of the mobile user, where the gyroscope gives state update, while accelerometer and magnetometer give measurement update to correct the gyroscope drift error [4].

Therefore equation (1.6) changes to:

$$\boldsymbol{\theta}_k \;=\; g\left(\boldsymbol{\theta}_{k-1}\right) \;+\; \boldsymbol{v}_k \tag{1.11}$$

where the state vector $\boldsymbol{\theta}_k$ consists now of the mobile user location and speed at time step k, function $g(.)$ of the previous estimate and of the gyroscope data [4] [5].

Measurements from *Distance-of-Arrival* (*DoA*) and *Time-of-Arrival* (*ToA*) are gathered by the network to yield a mobile station position estimate, so the shape of the measurement equation (1.8) does not change, but the function $f(\boldsymbol{\theta}_k)$ and the noise associated now also depends on that values and on the accelerometer and magnetometer data [4] [5].

1.3 Proposed idea

Starting from the idea given by the reference paper [2], that aims to estimate and track the AoA at mobile user, and the innovative approaches that take in

consideration data of the embedded sensors, this work proposes two different approaches for tracking the dynamic mmWave propagation path parameters over time.

For both of them the flowchart is the same:

- Definition of the scenario, hypothesis, state to estimate and type of measurements allowed;
- Modeling of the system dynamic and its time-discrete equivalent, i.e., definition of the mobility model;
- Formulation of the measurement model;
- KF or EKF (depending on the linearity or non-linearity of the measurement model) for the state estimate;
- Performance evaluation in term of RMSE.

The scenario and starting hypothesis are the same for both. The first approach proposed is similar to [2], because the state is the AoA (and its rate of change) of the multiple paths received from the mobile user, whereas in the second approach the state is composed of the positions (and speeds) of the base station and scatterers where the paths originate. For both the solutions proposed, it is supposed that mobile user can get a noisy measurement of the AoA (e.g., given by the antenna array) and possibly, for the second approach, a noisy measurement of the distance between the sources of the paths and the mobile user. Both suppose the possible use of data fusion to update the mobility model.

In the following chapters a detailed explanation is given.

Chapter 2

First approach: beam tracking using AoA and its rate of change as state to estimate

The first approach proposed in this work is similar to the reference paper [2], due to the fact that it aims to track directly AoA of the multiple paths received from the mobile station.

In the following sections it is shown how the approach is structured, considering the starting hypothesis and the scenario, the mobility and measurement models, concluding with the estimation algorithm.

2.1 Scenario

Without losing in generality compared to a 3D approach, in this work a 2D (outdoor) scenario has been studied (Figure 2.1). In particular, it is hypothesized that a single mobile user is moving in a $200 \times 200m$ area, a typical size for systems using mmWave communications, in which the the cell radius will be between few meters (e.g., ultra dense scenarios) and few hundreds of meters (e.g., rural environments) [6]. The movement is supposed random walk like, with a speed typical of a mobile user walking or moving with a vehicle, depending on the testing conditions. The mobile user can also rotate, but the center of the (relative) reference system is supposed to be always the position of the mobile user itself.

Due to the consideration done in the introduction and in the first chapter about the mmWave channel behavior, it can be reasonably assumed that only few paths arrives at the mobile user with different AoAs. In particular, three paths are taking in consideration, one due to the direct link between

the only base station in the cell and the mobile user (LOS), and the others two due to the reflections by scatterers, seen as equivalent sources. These objects, for simplicity, are assumed to be points of reflection in the space. The positions of the three sources are fixed and known a-priori only with a large uncertainty.

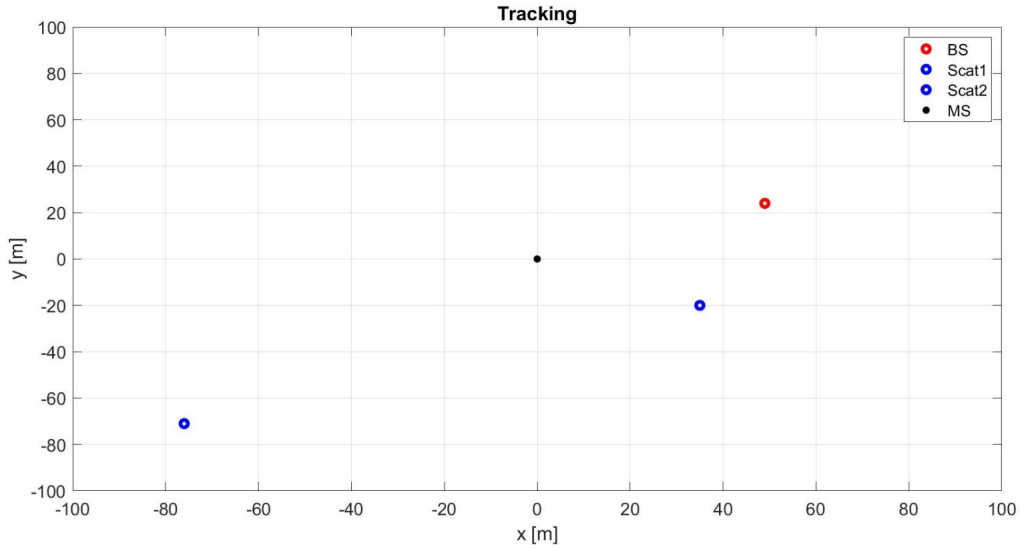

Figure 2.1: 2D scenario

This type of physical systems can often be modeled with differential equations, i.e., it is possible to define a state-space model, as shown in Chapter 1.

2.2 Mobility model

The discrete mobility, or state-space, model describes the evolution of the state over the time steps. In the proposed approach, the state $\boldsymbol{\theta}_k$ at time step k are the AoAs $\boldsymbol{\varphi}_k$ of the three paths received from the mobile user and their rate of change $\dot{\boldsymbol{\varphi}}_k$, as follow:

$$\boldsymbol{\theta}_k = [\ \varphi_{1,k}\ \varphi_{2,k}\ \varphi_{3,k}\ \dot{\varphi}_{1,k}\ \dot{\varphi}_{2,k}\ \dot{\varphi}_{3,k}\ ]^T \quad . \tag{2.1}$$

The dynamics of the system is modeled by this time-continue equation:

$$\frac{d\boldsymbol{\theta}(t)}{dt} \;=\; \boldsymbol{F}\;\boldsymbol{\theta}(t) \;+\boldsymbol{G}\;\boldsymbol{u}(t)+\;\boldsymbol{L}\;\boldsymbol{w}(t) \;=$$

$$= \begin{bmatrix} \mathbf{0}_{3x3} & \boldsymbol{I}_3 \\ \mathbf{0}_{3x3} & \mathbf{0}_{3x3} \end{bmatrix} \begin{bmatrix} \varphi_1(t) \\ \varphi_2(t) \\ \varphi_3(t) \\ \dot{\varphi}_1(t) \\ \dot{\varphi}_2(t) \\ \dot{\varphi}_3(t) \end{bmatrix} + \boldsymbol{I}_6\;\boldsymbol{u}(t) + \begin{bmatrix} \mathbf{0}_{3x3} & \mathbf{0}_{3x3} \\ \boldsymbol{I}_3 & \mathbf{0}_{3x3} \end{bmatrix}\;\boldsymbol{w}(t) \quad (2.2)$$

where the rate of change of the state is perturbed by an *Additive White Gaussian Noise* (*AWGN*) process $\boldsymbol{w}$(t) with an angle acceleration spectral density $\boldsymbol{W_e}$. It could be considered also as an index about the ability of the system to react to changes, i.e., system inertia.

The syntax $\mathbf{0}_{ixj}$ means a zeros matrix with i rows and j columns, while $\boldsymbol{I}_i$ is the identity matrix with i rows and i columns. The control signal $\boldsymbol{u}$(t) will be investigated in subsection (2.2.1) and contains noisy data from the gyroscope, in order to provide a better estimate.

As cited above, the system is sampled with a time step Δt, therefore the state evolves in a discrete manner between two consecutive time steps:

$$\begin{cases} \varphi_{i,k} = \varphi_{i,k-1} + \Delta t\;\dot{\varphi}_{i,k-1} + \dot{\varphi}_{g_{i,k}} + v_{\varphi_{i,k}} \\ \dot{\varphi}_{i,k} = \dot{\varphi}_{i,k-1} + v_{\dot{\varphi}_{i,k}} \end{cases} \qquad i = 1,2,3 \;\; . \quad (2.3)$$

The system above can be written in a matrix form:

$$\underbrace{\begin{bmatrix} \varphi_{1,k} \\ \varphi_{2,k} \\ \varphi_{3,k} \\ \dot{\varphi}_{1,k} \\ \dot{\varphi}_{2,k} \\ \dot{\varphi}_{3,k} \end{bmatrix}}_{\boldsymbol{\theta}_k} = \underbrace{\begin{bmatrix} \boldsymbol{I}_3 & diag_3(\Delta t) \\ \mathbf{0}_{3x3} & \boldsymbol{I}_3 \end{bmatrix}}_{\boldsymbol{\Phi}_k} \underbrace{\begin{bmatrix} \varphi_{1,k-1} \\ \varphi_{2,k-1} \\ \varphi_{3,k-1} \\ \dot{\varphi}_{1,k-1} \\ \dot{\varphi}_{2,k-1} \\ \dot{\varphi}_{3,k-1} \end{bmatrix}}_{\boldsymbol{\theta}_{k-1}} + \underbrace{diag_6(\Delta t)}_{\boldsymbol{B}\;=\;\boldsymbol{G}\;\Delta t} \underbrace{\begin{bmatrix} \dot{\varphi}_{g_{1,k}} \\ \dot{\varphi}_{g_{2,k}} \\ \dot{\varphi}_{g_{3,k}} \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{\boldsymbol{u}_k} + \underbrace{\begin{bmatrix} v_{\varphi_{1,k}} \\ v_{\varphi_{2,k}} \\ v_{\varphi_{3,k}} \\ v_{\dot{\varphi}_{1,k}} \\ v_{\dot{\varphi}_{2,k}} \\ v_{\dot{\varphi}_{3,k}} \end{bmatrix}}_{\boldsymbol{v}_k} \quad (2.4)$$

where the syntax $diag_i(\Delta t)$ means a diagonal matrix with i rows and i columns and the element Δt on the principal diagonal.

The matrix $\boldsymbol{\Phi}_k$ is called state-transition matrix and describes how the state evolves from the time step *k-1* to the time step *k*. The reference paper [2] gives a mathematical formula to evaluate this matrix, as follow:

$$\boldsymbol{\Phi}_k \;=\; exp(\boldsymbol{F}\Delta t) \;\xrightarrow{\Delta t \to 0}\; \boldsymbol{I}_6 + \boldsymbol{F}\Delta t \;\; . \quad (2.5)$$

The vector $\boldsymbol{v}_k \sim N(0, \boldsymbol{Q_e})$ represents the state noise, Normal distributed, with zero mean and covariance matrix $\boldsymbol{Q_e}$ determined as in equation (1.5).

2.2.1 Data fusion

Data from the gyroscope can give an additional information that could provide a more precise evaluation of the state evolution over the time steps. Therefore, this work follows the idea proposed in [4] to include these data in the control vector $\boldsymbol{u}_k$ in order to act directly on the mobility model.

As mentioned about the gyroscope, this sensor is afflicted by drifting issues due to the integration over time. However, since the sampling time is very short, its data could be reasonably considered accurate. Therefore the angular speed of the device could be considered known for every time step and is evaluated as:

$$\dot{\varphi}_{g_{i,k}} = -\frac{(rot_k - rot_{k-1})}{\Delta t} \quad i = 1, 2, 3 \tag{2.6}$$

where rot_k is the self-rotation of the device at the time step k refers to the non-rotated reference system (Figure 2.2). From the implementation point of view, since the equation above considers two consecutive time steps, the normalization for the sampling step Δt will be implicit. Moreover, it is a negative contribute because, as it is evident in Figure 2.2, the counterclockwise device rotation ($rot_k > 0$) causes a reduction of the AoA between instants *k-1* and *k*.

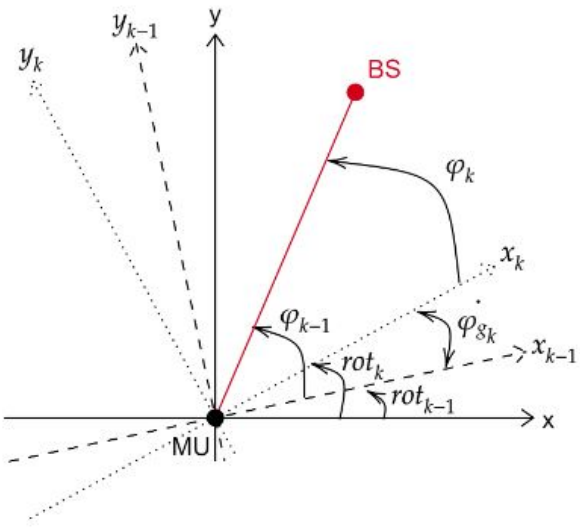

Figure 2.2: Effect of the device self rotation on AoA and gyroscope data

It is obvious that without this extra information the estimate of the next state is less accurate, so this value contributes to refine the predicted AoAs given by the product $\boldsymbol{\Phi}_k \boldsymbol{\theta}_{k-1}$ and acts in equal manner for every received path (although Figure 2.2 is referred only to the base station).

In reality, it is necessary to give a clarification. With reference to the Figure 2.2, the AoAs coincide with the values φ_k and φ_{k-1} only for a perfect estimate, so is more correct to use the measured angles instead of the expected values. Moreover, the gyroscope provides a noisy "measurement", that in this work it is supposed to be Normal distributed and included directly on the angular speed $\dot{\varphi}_{g_{i,k}}$ value.

In the general situation the mobile station is also moving, so:

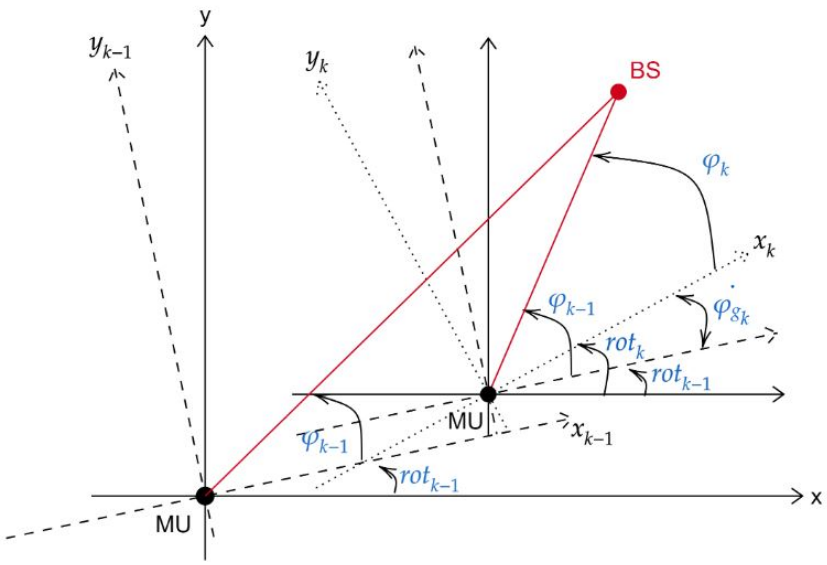

Figure 2.3: Effect of the device self rotation and movement on AoA and gyroscope data

2.3 Measurement model

At every time step the device can obtain and elaborate some measurements from the antennas array. In particular, following the ideas proposed in the reference papers [4] [5], this work assumes that is possible to have a noisy information about the effective AoAs between the received signal direction and the RX beam direction, with the device self rotation embedded in them, i.e., included in the value measured by antennas.

Because of the noise that affects them and due to the Bayesian approach adopted, measurements are modeled as stochastic process, as follows:

$$\boldsymbol{z}_k = \begin{bmatrix} \varphi_{1,k} \\ \varphi_{2,k} \\ \varphi_{3,k} \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{I}_3 & \boldsymbol{0}_{3x3} \end{bmatrix}}_{\boldsymbol{H}} \boldsymbol{\theta}_k + \boldsymbol{r}_k \tag{2.7}$$

where $\boldsymbol{r}_k \sim N(0, \boldsymbol{R})$ is the measurement noise, with covariance matrix $\boldsymbol{R} = diag_3(\sigma^2)$ and σ its standard deviation, assumed to be constant for hypothesis and depending on the antennas.

Due to the consistence between the type of measurements and the state, the observation matrix $\boldsymbol{H}$ is linear and time-independent; moreover, the AoA rates of change state components does not affect the measurements themselves.

2.4 KF estimation algorithm

The filtering phase aims to estimate the current state value given past estimates, current observations and the mobility model.

The Bayesian filter is optimum (time-varying filter) when the mobility model is linear with additive Gaussian noise (sequential MMSE estimator). In this case is called Kalman Filter and its equations can be solved in closed form, with resulting distributions Normal distributed.

The KF is an algorithm to compute recursively $\boldsymbol{m}_k$, i.e., averages vector of the a-posteriori Gaussian distribution of the state at time step k (for MAP/MMSE estimators coincides with the point estimates $\hat{\boldsymbol{\theta}}_k$), and $\boldsymbol{P}_k$, i.e., a-posteriori error covariance matrix, that give an idea about the accuracy of the estimate.

The algorithm consists in three main phases:

- Initialization, that is the first phase performed only once when the algorithm starts. In this step the a-priori averages vector $\boldsymbol{m}_0$ and a-priori error covariance matrix $\boldsymbol{P}_0$ of the Gaussian distributions representing the state estimate are defined.

- Prediction, in which the algorithm tries to give a first estimate of the state, predicting, at time step k, the averages vector $\boldsymbol{m}_k$ and its uncertainty $\boldsymbol{P}_k$ relying exclusively on the mobility model. Therefore the Gaussian distribution moves in the "space of the estimate" and gets larger, i.e., its standard deviation increases.

- Update, where the uncertainty is reduced thanks to the measurements that refine the estimate. This last step is repeated recursively alternating with the prediction phase, until the algorithm stops.

To better understand how the distribution evolves towards this steps, the Figure 2.4 explains schematically what has just been said.

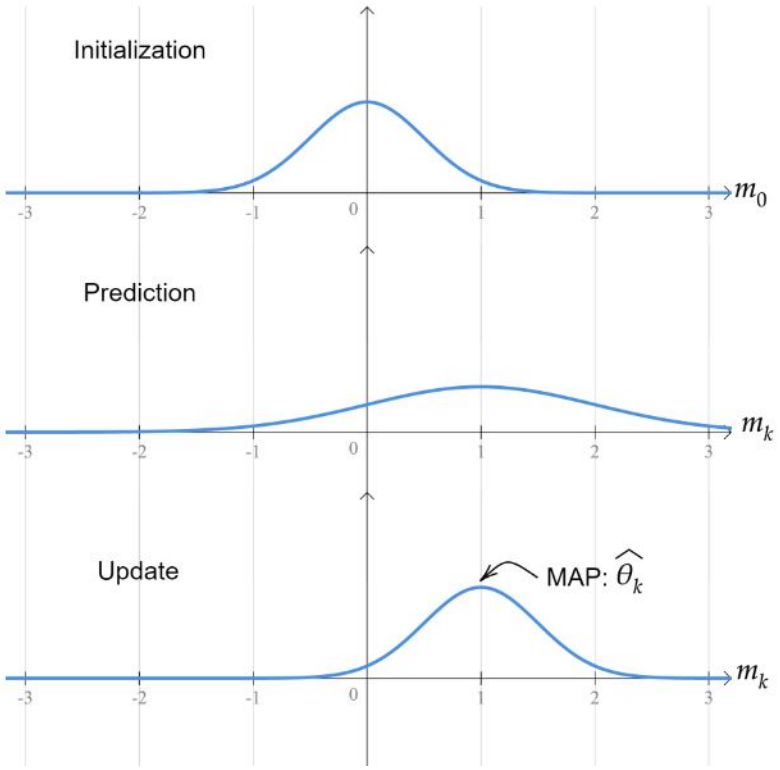

Figure 2.4: Phases of the KF and evolution of the estimate distribution

The mathematical implementation is reported below:

$$Init: \begin{cases} \boldsymbol{m}_0 = \underbrace{[\, m_{1,0}\ m_{2,0}\ m_{3,0}}_{AoAs}\ \underbrace{\dot{m}_{1,0}\ \dot{m}_{2,0}\ \dot{m}_{3,0}\,]^T}_{Rates\ of\ change} \\ \boldsymbol{P}_0 = \begin{bmatrix} diag_3\left(\sigma^2_{Kal}\right) & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} & diag_3\left(\sigma^2_{Kal1}\right) \end{bmatrix} \end{cases} \tag{2.8}$$

$$Prediction: \begin{cases} \boldsymbol{m}_k = \boldsymbol{\Phi}\boldsymbol{m}_{k-1} + \boldsymbol{B}\boldsymbol{u}_k \\ \boldsymbol{P}_k = \boldsymbol{\Phi}\boldsymbol{P}_{k-1}\boldsymbol{\Phi}^T + \boldsymbol{Q_e} \end{cases} \tag{2.9}$$

$$Update: \begin{cases} \begin{cases} \boldsymbol{v}_k = \boldsymbol{z}_k - \boldsymbol{H}\boldsymbol{m}_k \\ \boldsymbol{S}_k = \boldsymbol{H}\boldsymbol{P}_k\boldsymbol{H}^T + \boldsymbol{R} \\ \boldsymbol{K}_k = \boldsymbol{P}_k\boldsymbol{H}^T\boldsymbol{S}_k^{-1} \end{cases} \\ \begin{cases} \boldsymbol{m}_k = \boldsymbol{m}_k + \boldsymbol{K}_k\boldsymbol{v}_k \\ \boldsymbol{P}_k = \boldsymbol{P}_k - \boldsymbol{K}_k\boldsymbol{S}_k\boldsymbol{K}_k^T \end{cases} \end{cases} \tag{2.10}$$

where σ^2_{Kal} and σ^2_{Kal1} are respectively the initial variances that afflict the AoA values and their rates of change, which the former is typically bigger than the variance σ^2 of the measurement noise covariance matrix $\boldsymbol{R}$. Moreover, $\boldsymbol{v}_k$ is the innovation term while $\boldsymbol{S}_k$ its covariance, $\boldsymbol{H}$ is the (linear) measurement model, $\boldsymbol{K}_k$ the Kalman gain, $\boldsymbol{m}_k$ and $\boldsymbol{P}_k$ in the left members of (2.10) are respectively the averages vector and the covariance matrix of a Normal distribution evolving through the time steps, at time step k.

Because of the linearity of the mobility model and the Gaussian behavior of the noise, the algorithm above returns an optimum MMSE estimate, where $\hat{\boldsymbol{\theta}}_k$ coincides with $\boldsymbol{m}_k$ at every time step k. This means that the KF is a MAP estimator.

A similar method that performs the EKF is valid for the second approach. Therefore, because of the different type of state, some changes must be made, that will be explained in the next chapter.

Chapter 3

Second approach: beam tracking using scatterers positions and speeds as state to estimate

The flowchart of the second approach is very similar to the one of the first approach, with the main difference that it aims to track the positions of the scatterers and base station in order to obtain a kind of knowledge of the environment that could provide further *Channel State Information* (*CSI*). For example, knowing the relative position of the base station and scatterers compared to the mobile station permits to know the distance between the mobile station and these objects. This allows for both TX and RX to set an appropriate power to provide the communication, in order to waste the minimum possible energy.

In the following sections it is shown how the approach is structured, considering the starting hypothesis and the scenario, the mobility and measurement models, concluding with the estimation algorithm.

3.1 Scenario

In order to provide uniformity, the scenario is the same of the first approach (Figure 2.1). Summing up the hypothesis considered are:

- 2D outdoor scenario, $200 \times 200m$ area;
- One mobile user moving random walk like, possibly with self-rotation;

- Center of the reference system coincident with the position of the mobile user itself, for every time step;
- Three paths (one LOS from base station and two from scatterers);
- Fixed reflection points known a-priori with large uncertainties;
- Markovian hypothesis and Bayesian modeling of system dynamics and measurements.

3.2 Mobility model

The evolution of the state over the time steps is given by the discrete mobility, or state-space, model. In the proposed approach, the state $\boldsymbol{\theta}_k$ at time step k is composed of the positions $\boldsymbol{P}_k$ of the base station and scatterers, i.e., the source and reflection points, and their speeds $\dot{\boldsymbol{P}}_k$ relative to the mobile user reference coordinate system, as follows:

$$\boldsymbol{\theta}_k \;=\; \left[\; \boldsymbol{P}_{1,k}\; \boldsymbol{P}_{2,k}\; \boldsymbol{P}_{3,k}\; \dot{\boldsymbol{P}}_{1,k}\; \dot{\boldsymbol{P}}_{2,k}\; \dot{\boldsymbol{P}}_{3,k} \;\right]^T \tag{3.1}$$

with:

$$\begin{cases} \boldsymbol{P}_{i,k} = [\; x_{i,k}\; y_{i,k} \;]^T \\ \dot{\boldsymbol{P}}_{i,k} = [\; \dot{x}_{i,k}\; \dot{y}_{i,k} \;]^T \end{cases} \qquad i = 1, 2, 3 \quad . \tag{3.2}$$

The sentence above does not mean that the base station and scatterers are moving, but the meaning is that since the mobile user is moving, and so the reference system is moving accordingly, the relative positions of the three objects compared to the mobile user change at every time step, consistently with the mobile user movement (Figure 2.3). In other words, in a global reference system the base station and scatterers are fixed, but from the mobile user point of view (center of the local reference system) the objects appear in movement, so their positions evolve in time and must be tracked.

It is clear that tracking the positions permits also to obtain an estimate $\boldsymbol{\varphi}_k$ of the AoAs of the three paths received from the mobile station, simply applying the mathematical definition of arctangent:

$$\varphi_{i,k} = \; arctg\left(\frac{x_{i,k}}{y_{i,k}}\right) \qquad i = 1, 2, 3 \quad . \tag{3.3}$$

The dynamics of the system is modeled by this time-continue equation:

$$\frac{d\boldsymbol{\theta}(t)}{dt} = \boldsymbol{F}\ \boldsymbol{\theta}(t) + \boldsymbol{G}\ \boldsymbol{u}(t) + \boldsymbol{L}\ \boldsymbol{w}(t) =$$

$$= \begin{bmatrix} \mathbf{0}_{6x6} & \boldsymbol{I}_6 \\ \mathbf{0}_{6x6} & \mathbf{0}_{6x6} \end{bmatrix} \begin{bmatrix} x_1(t) \\ y_1(t) \\ x_2(t) \\ y_2(t) \\ x_3(t) \\ y_3(t) \\ \dot{x}_1(t) \\ \dot{y}_1(t) \\ \dot{x}_2(t) \\ \dot{y}_2(t) \\ \dot{x}_3(t) \\ \dot{y}_3(t) \end{bmatrix} + \boldsymbol{I}_{12}\ \boldsymbol{u}(t) + \begin{bmatrix} \mathbf{0}_{6x6} & \mathbf{0}_{6x6} \\ \boldsymbol{I}_6 & \mathbf{0}_{6x6} \end{bmatrix}\ \boldsymbol{w}(t) \quad (3.4)$$

where the speeds are perturbed by an AWGN process $\boldsymbol{w}(t)$ with an acceleration spectral density $\boldsymbol{W_e}$. Like in the first approach, $\boldsymbol{W_e}$ is an index about the system inertia, but in this case is the same as the one of the random walk process $\boldsymbol{W_u}$, that is the mobile user acceleration (process) noise spectral density.

The control signal $\boldsymbol{u}(t)$ will be investigated in subsection (3.2.1) and contains noisy data from the gyroscope and accelerometer, in order to provide a better estimate (data fusion).

The state evolves in a discrete manner between two consecutive time steps due to the sampling with time step Δt:

$$\begin{cases} \boldsymbol{P}_{i,k} = \boldsymbol{I}_2 \boldsymbol{P}_{i,k-1} + diag_2(\Delta t)\ \dot{\boldsymbol{P}}_{i,k-1} + diag_2(\Delta t)\dot{\boldsymbol{P}}_{g_{i,k}} + v_{\boldsymbol{P}_{i,k}} \\ \dot{\boldsymbol{P}}_{i,k} = \boldsymbol{I}_2 \dot{\boldsymbol{P}}_{i,k-1} + diag_2(\Delta t)\dot{\boldsymbol{P}}_{a_{i,k}} + v_{\dot{\boldsymbol{P}}_{i,k}} \end{cases} \quad (3.5)$$

and the system above can be written in a matrix form:

$$\underbrace{\begin{bmatrix} \boldsymbol{P}_{1,k} \\ \boldsymbol{P}_{2,k} \\ \boldsymbol{P}_{3,k} \\ \dot{\boldsymbol{P}}_{1,k} \\ \dot{\boldsymbol{P}}_{2,k} \\ \dot{\boldsymbol{P}}_{3,k} \end{bmatrix}}_{\boldsymbol{\theta}_k} = \underbrace{\begin{bmatrix} \boldsymbol{I}_6 & diag_6(\Delta t) \\ \mathbf{0}_{6x6} & \boldsymbol{I}_6 \end{bmatrix}}_{\boldsymbol{\Phi}_k = \boldsymbol{I}_{12} + \boldsymbol{F}\Delta t} \underbrace{\begin{bmatrix} \boldsymbol{P}_{1,k-1} \\ \boldsymbol{P}_{2,k-1} \\ \boldsymbol{P}_{3,k-1} \\ \dot{\boldsymbol{P}}_{1,k-1} \\ \dot{\boldsymbol{P}}_{2,k-1} \\ \dot{\boldsymbol{P}}_{3,k-1} \end{bmatrix}}_{\boldsymbol{\theta}_{k-1}} + \underbrace{diag_{12}(\Delta t)}_{\boldsymbol{B}\ =\ \boldsymbol{G}\ \Delta t} \underbrace{\begin{bmatrix} \dot{\boldsymbol{P}}_{g_{1,k}} \\ \dot{\boldsymbol{P}}_{g_{2,k}} \\ \dot{\boldsymbol{P}}_{g_{3,k}} \\ \ddot{\boldsymbol{P}}_{a_{1,k}} \\ \ddot{\boldsymbol{P}}_{a_{2,k}} \\ \ddot{\boldsymbol{P}}_{a_{3,k}} \end{bmatrix}}_{\boldsymbol{u}_k} + \underbrace{\begin{bmatrix} v_{\boldsymbol{P}_{1,k}} \\ v_{\boldsymbol{P}_{2,k}} \\ v_{\boldsymbol{P}_{3,k}} \\ v_{\dot{\boldsymbol{P}}_{1,k}} \\ v_{\dot{\boldsymbol{P}}_{2,k}} \\ v_{\dot{\boldsymbol{P}}_{3,k}} \end{bmatrix}}_{\boldsymbol{v}_k} \quad (3.6)$$

where the vector $\boldsymbol{u}_k$ of the gyroscope and the accelerometer data will be investigated in the next subsection,$\boldsymbol{\Phi}_k$ is the state-transition matrix [2] and the vector $\boldsymbol{v}_k \sim N(0, \boldsymbol{Q_e})$ represents the state noise, Normal distributed, with zero mean and covariance matrix $\boldsymbol{Q_e}$ determined as in equation (1.5). In particular:

$$\begin{cases} v_{\boldsymbol{P}_{i,k}} = [\ v_{x_{i,k}}\ v_{y_{i,k}}\ ]^T \\ v_{\dot{\boldsymbol{P}}_{i,k}} = [\ v_{\dot{x}_{i,k}}\ v_{\dot{y}_{i,k}}\ ]^T \end{cases} \qquad i = 1, 2, 3 \quad . \tag{3.7}$$

3.2.1 Data fusion

Data from the gyroscope and the accelerometer embedded in the device can give an additional term that could provide a more precise evaluation of the state evolution over the time steps. In particular, gyroscope provides an information about the device rotation in term of its angular speed, while the accelerometer provides the device spatial acceleration.

Therefore, this work follows the idea proposed in [4] to include these data in the control vector $\boldsymbol{u}_k$ in order to act directly on the mobility model, as I did in the first approach. So $\boldsymbol{u}_k$ is composed of:

$$\begin{cases} \dot{P}_{g_{i,k}} = [\ \dot{x}_{g_{i,k}}\ \dot{y}_{g_{i,k}}\ ]^T & \textit{Gyroscope} \\ \ddot{P}_{a_{i,k}} = [\ \ddot{x}_{a_{i,k}}\ \ddot{y}_{a_{i,k}}\ ]^T & \textit{Accelerometer} \end{cases} \qquad i = 1, 2, 3 \quad . \tag{3.8}$$

The first term of equation (3.8) does not represent the raw measurements obtained by the gyroscope, as it provides angular velocity, whereas in equation (3.8) the velocity is expressed in Cartesian coordinates. This means that the values in equation (3.8) must be computed starting from the measured angular velocity $\dot{\varphi}_{g_k}$ through a proper transformation. In particular it is:

$$\begin{cases} \dot{x}_{g_{i,k}} = \frac{\Delta x_{g_k}}{\Delta t} \\ \dot{y}_{g_{i,k}} = \frac{\Delta y_{g_k}}{\Delta t} \end{cases} \qquad i = 1, 2, 3 \tag{3.9}$$

with:

$$\begin{cases} \Delta x_{g_k} = d_{k-1} cos\left(\varphi_{k-1} - \dot{\varphi}_{g_k}\right) - d_{k-1} cos(\varphi_{k-1}) cos\left(\dot{\varphi}_{g_k}\right) \\ \Delta y_{g_k} = d_{k-1} sin\left(\varphi_{k-1} - \dot{\varphi}_{g_k}\right) - d_{k-1} sin(\varphi_{k-1}) cos\left(\dot{\varphi}_{g_k}\right) \end{cases} \tag{3.10}$$

where d_{k-1} and φ_{k-1} are respectively the measurements, at the previous time step, of the distances between mobile user and base station (or scatterers) and the measurements of the AoAs, as it will be shown in the next section. Indeed, since the vector $\boldsymbol{u}_k$ acts to improve the state prediction, the

measurements of distance and AoA at the time step k are not available yet. Moreover, because of the normalization in equation (3.9), in this case the value of the gyroscope data is given by:

$$\dot{\varphi}_{g_{i,k}} = -(rot_k - rot_{k-1}) \quad i = 1, 2, 3 \tag{3.11}$$

as explained in Figure 2.2, and acts equally on the three AoAs of the received signals. Therefore, the value $\varphi_{k-1} - \dot{\varphi}_{g_k}$ is an AoA prediction.

The mapping Δx_{g_k} of the angular speed at time step k given by the gyroscope on the x axis could be obtained as the difference between the two terms. The former is the abscissa of the base station (or scatterer) in the rotated reference system at the time step k, while the latter is the projection of the abscissa of the base station (or scatterer) in the rotated reference system at the time step *k-1* on the x-axis of the rotated system at time step k. A similar reasoning could be done for the Δy_{g_k}.

The graphic explanation of the equation (3.10) is given in Figure 3.1.

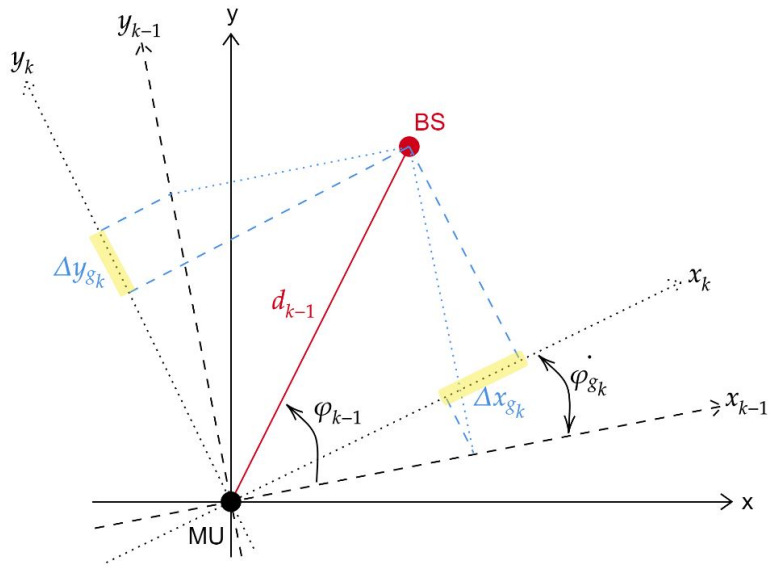

Figure 3.1: Mapping of the device angular speed to the Cartesian coordinates

As said previously, the accelerometer data act directly on the speed terms of the mobility model, because it provides an information about the acceleration of the mobile station in the global reference system, i.e., the acceleration of the base station and scatterers in the local reference system where the mobile station is the center. Therefore, the terms $\ddot{x}_{a_{i,k}}$ and $\ddot{y}_{a_{i,k}}$ of the control vector $\boldsymbol{u}_k$ are given by:

$$\begin{cases} \ddot{x}_{a_{i,k}} = \frac{\dot{x}_{u_{i,k}} - \dot{x}_{u_{i,k-1}}}{\Delta t} \\ \ddot{y}_{a_{i,k}} = \frac{\dot{y}_{u_{i,k}} - \dot{y}_{u_{i,k-1}}}{\Delta t} \end{cases} \qquad i = 1, 2, 3 \tag{3.12}$$

that are the speed variations compared to the x and y directions.

3.3 Measurement model

Compared to the first approach, the measurements that the device can obtain from the antennas array and elaborate at every time step are slightly different. Although the mobile user can still receive information about the AoAs between the received signal direction and the RX beam direction, extra information about the distance between the mobile user and base station (or scatterers) are supposed to be available. As before, the device self rotation is already included in the value measured by antennas and all of these measurements are noisy and modeled as stochastic process, as follow:

$$\boldsymbol{z}_k = \begin{bmatrix} \varphi_{1,k} \\ \varphi_{2,k} \\ \varphi_{3,k} \\ d_{1,k} \\ d_{2,k} \\ d_{3,k} \end{bmatrix} = h\left(\boldsymbol{\theta}_k\right) + \boldsymbol{r}_k \tag{3.13}$$

where $\boldsymbol{r}_k \sim N(0, \boldsymbol{R})$ is the measurement noise, with covariance matrix $\boldsymbol{R}$ defined as:

$$\boldsymbol{R} = \begin{bmatrix} diag_3\left(\sigma^2\right) & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} & diag_3\left(\sigma_1^2\right) \end{bmatrix} \tag{3.14}$$

with σ^2 and σ_1^2 are the variances of the noise that affects the AoAs and distances respectively, assumed to be constant for hypothesis and depending on the antennas.

It is evident from equation (3.13) that in this approach the relationship between the measurement vector and state vector is non-linear. In particular, the function $h(.)$ links the position estimate $\boldsymbol{P}_{i,k}$ in the local reference system to the estimate $\varphi_{i,k}$ of the AoA and to the distance $d_{i,k}$ estimate between the mobile user and the base station (and scatterers), as follows:

$$\begin{cases} \varphi_{i,k} = arctg\left(\frac{m_{x_{i,k}}}{m_{y_{i,k}}}\right) \\ d_{i,k} = \sqrt{m_{x_{i,k}}^2 + m_{y_{i,k}}^2} \end{cases} \qquad i = 1, 2, 3 \quad . \tag{3.15}$$

Since the model is non-linear, the KF method cannot be applied. In the following section we will use the EKF method, which is based on model linearization of function $h(.)$. The way to do that is to compute, at every time step k, the first-order partial derivatives compared to the parameters of the mobility model, i.e., the Jacobian matrix:

$$H_k = \frac{\delta h(\theta_k)}{\delta \theta_k} =$$

$$= \begin{bmatrix} \frac{\delta\varphi_{1,k}}{\delta x_{1,k}} & \frac{\delta\varphi_{1,k}}{\delta y_{1,k}} & \frac{\delta\varphi_{1,k}}{\delta x_{2,k}} & \frac{\delta\varphi_{1,k}}{\delta y_{2,k}} & \frac{\delta\varphi_{1,k}}{\delta x_{3,k}} & \frac{\delta\varphi_{1,k}}{\delta y_{3,k}} & \frac{\delta\varphi_{1,k}}{\delta \dot{x}_{1,k}} & \frac{\delta\varphi_{1,k}}{\delta \dot{y}_{1,k}} & \frac{\delta\varphi_{1,k}}{\delta \dot{x}_{2,k}} & \frac{\delta\varphi_{1,k}}{\delta \dot{y}_{2,k}} & \frac{\delta\varphi_{1,k}}{\delta \dot{x}_{3,k}} & \frac{\delta\varphi_{1,k}}{\delta \dot{y}_{3,k}} \\ \frac{\delta\varphi_{2,k}}{\delta x_{1,k}} & \frac{\delta\varphi_{2,k}}{\delta y_{1,k}} & \frac{\delta\varphi_{2,k}}{\delta x_{2,k}} & \frac{\delta\varphi_{2,k}}{\delta y_{2,k}} & \frac{\delta\varphi_{2,k}}{\delta x_{3,k}} & \frac{\delta\varphi_{2,k}}{\delta y_{3,k}} & \frac{\delta\varphi_{2,k}}{\delta \dot{x}_{1,k}} & \frac{\delta\varphi_{2,k}}{\delta \dot{y}_{1,k}} & \frac{\delta\varphi_{2,k}}{\delta \dot{x}_{2,k}} & \frac{\delta\varphi_{2,k}}{\delta \dot{y}_{2,k}} & \frac{\delta\varphi_{2,k}}{\delta \dot{x}_{3,k}} & \frac{\delta\varphi_{2,k}}{\delta \dot{y}_{3,k}} \\ \frac{\delta\varphi_{3,k}}{\delta x_{1,k}} & \frac{\delta\varphi_{3,k}}{\delta y_{1,k}} & \frac{\delta\varphi_{3,k}}{\delta x_{2,k}} & \frac{\delta\varphi_{3,k}}{\delta y_{2,k}} & \frac{\delta\varphi_{3,k}}{\delta x_{3,k}} & \frac{\delta\varphi_{3,k}}{\delta y_{3,k}} & \frac{\delta\varphi_{3,k}}{\delta \dot{x}_{1,k}} & \frac{\delta\varphi_{3,k}}{\delta \dot{y}_{1,k}} & \frac{\delta\varphi_{3,k}}{\delta \dot{x}_{2,k}} & \frac{\delta\varphi_{3,k}}{\delta \dot{y}_{2,k}} & \frac{\delta\varphi_{3,k}}{\delta \dot{x}_{3,k}} & \frac{\delta\varphi_{3,k}}{\delta \dot{y}_{3,k}} \\ \frac{\delta d_{1,k}}{\delta x_{1,k}} & \frac{\delta d_{1,k}}{\delta y_{1,k}} & \frac{\delta d_{1,k}}{\delta x_{2,k}} & \frac{\delta d_{1,k}}{\delta y_{2,k}} & \frac{\delta d_{1,k}}{\delta x_{3,k}} & \frac{\delta d_{1,k}}{\delta y_{3,k}} & \frac{\delta d_{1,k}}{\delta \dot{x}_{1,k}} & \frac{\delta d_{1,k}}{\delta \dot{y}_{1,k}} & \frac{\delta d_{1,k}}{\delta \dot{x}_{2,k}} & \frac{\delta d_{1,k}}{\delta \dot{y}_{2,k}} & \frac{\delta d_{1,k}}{\delta \dot{x}_{3,k}} & \frac{\delta d_{1,k}}{\delta \dot{y}_{3,k}} \\ \frac{\delta d_{2,k}}{\delta x_{1,k}} & \frac{\delta d_{2,k}}{\delta y_{1,k}} & \frac{\delta d_{2,k}}{\delta x_{2,k}} & \frac{\delta d_{2,k}}{\delta y_{2,k}} & \frac{\delta d_{2,k}}{\delta x_{3,k}} & \frac{\delta d_{2,k}}{\delta y_{3,k}} & \frac{\delta d_{2,k}}{\delta \dot{x}_{1,k}} & \frac{\delta d_{2,k}}{\delta \dot{y}_{1,k}} & \frac{\delta d_{2,k}}{\delta \dot{x}_{2,k}} & \frac{\delta d_{2,k}}{\delta \dot{y}_{2,k}} & \frac{\delta d_{2,k}}{\delta \dot{x}_{3,k}} & \frac{\delta d_{2,k}}{\delta \dot{y}_{3,k}} \\ \frac{\delta d_{3,k}}{\delta x_{1,k}} & \frac{\delta d_{3,k}}{\delta y_{1,k}} & \frac{\delta d_{3,k}}{\delta x_{2,k}} & \frac{\delta d_{3,k}}{\delta y_{2,k}} & \frac{\delta d_{3,k}}{\delta x_{3,k}} & \frac{\delta d_{3,k}}{\delta y_{3,k}} & \frac{\delta d_{3,k}}{\delta \dot{x}_{1,k}} & \frac{\delta d_{3,k}}{\delta \dot{y}_{1,k}} & \frac{\delta d_{3,k}}{\delta \dot{x}_{2,k}} & \frac{\delta d_{3,k}}{\delta \dot{y}_{2,k}} & \frac{\delta d_{3,k}}{\delta \dot{x}_{3,k}} & \frac{\delta d_{3,k}}{\delta \dot{y}_{3,k}} \end{bmatrix}$$

where:

$$\begin{cases} \frac{\delta\varphi_{i,k}}{\delta x_{i,k}} = \frac{m_{y_{i,k}}}{m^2_{x_{i,k}} + m^2_{y_{i,k}}} \\ \frac{\delta\varphi_{i,k}}{\delta y_{i,k}} = -\frac{m_{x_{1,k}}}{m^2_{x_{i,k}} + m^2_{y_{i,k}}} \end{cases} \qquad i = 1, 2, 3 \tag{3.16}$$

and:

$$\begin{cases} \frac{\delta d_{i,k}}{\delta x_{i,k}} = \frac{m_{x_{1,k}}}{\sqrt{m^2_{x_{1,k}} + m^2_{y_{1,k}}}} \\ \frac{\delta d_{i,k}}{\delta y_{i,k}} = \frac{m_{y_{1,k}}}{\sqrt{m^2_{x_{1,k}} + m^2_{y_{1,k}}}} \end{cases} \qquad i = 1, 2, 3 \tag{3.17}$$

and all the other terms are zero.

3.4 EKF estimation algorithm

The filtering phase aims as before to estimate the current state value given past estimates, current observations and the mobility model. Due to the non-linearity of the measurement model, the KF is implemented in its extended

version, that takes into account the linearization $\boldsymbol{H}_k$ of the function $h(.)$ for every time step k, as seen in the previous section.

Like in the first approach, the algorithm consists in the three main phases (Figure 2.4) of Initialization, Prediction and Update, and its mathematical implementation is reported below:

$$Init: \begin{cases} \boldsymbol{m}_0 = \underbrace{\begin{bmatrix} \boldsymbol{m}_{1,0} & \boldsymbol{m}_{2,0} & \boldsymbol{m}_{3,0} \end{bmatrix}}_{Positions} \underbrace{\begin{bmatrix} \dot{\boldsymbol{m}}_{1,0} & \dot{\boldsymbol{m}}_{2,0} & \dot{\boldsymbol{m}}_{3,0} \end{bmatrix}^T}_{Speeds} \\ \boldsymbol{P}_0 = \begin{bmatrix} diag_6\left(\sigma^2_{Kal}\right) & \boldsymbol{0}_{6x6} \\ \boldsymbol{0}_{6x6} & diag_6\left(\sigma^2_{Kal1}\right) \end{bmatrix} \end{cases} \tag{3.18}$$

with:

$$\begin{cases} \boldsymbol{m}_{i,k} = \begin{bmatrix} m_{x_{i,k}} & m_{y_{i,k}} \end{bmatrix}^T \\ \dot{\boldsymbol{m}}_{i,k} = \begin{bmatrix} \dot{m}_{x_{i,k}} & \dot{m}_{y_{i,k}} \end{bmatrix}^T \end{cases} \qquad i = 1, 2, 3 \tag{3.19}$$

$$Prediction: \begin{cases} \boldsymbol{m}_k = \boldsymbol{\Phi}\boldsymbol{m}_{k-1} + \boldsymbol{B}\boldsymbol{u}_k \\ \boldsymbol{P}_k = \boldsymbol{\Phi}\boldsymbol{P}_{k-1}\boldsymbol{\Phi}^T + \boldsymbol{Q_e} \end{cases} \tag{3.20}$$

$$Update: \begin{cases} \begin{cases} \boldsymbol{v}_k = \boldsymbol{z}_k - h(\boldsymbol{m}_k) \\ \boldsymbol{S}_k = \boldsymbol{H}_k\boldsymbol{P}_k\boldsymbol{H}_k^T + \boldsymbol{R} \\ \boldsymbol{K}_k = \boldsymbol{P}_k\boldsymbol{H}_k^T\boldsymbol{S}_k^{-1} \end{cases} \\ \begin{cases} \boldsymbol{m}_k = \boldsymbol{m}_k + \boldsymbol{K}_k\boldsymbol{v}_k \\ \boldsymbol{P}_k = (\boldsymbol{I}_{12} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_k \end{cases} \end{cases} \tag{3.21}$$

where σ^2_{Kal} and σ^2_{Kal1} are respectively the initial variances that afflict the positions and speed values, which "nature" is different compared to the variances σ^2 and ${\sigma^2}_1$ of the measurement noise covariance matrix $\boldsymbol{R}$. Moreover, the meaning of the parameters $\boldsymbol{v}_k$, $\boldsymbol{S}_k$ and $\boldsymbol{K}_k$ is the same cited in section (2.4).

In this case, due to the approximation performed (linaerization), the EKF is not the optimum estimator. However, it is a MAP estimator yet, so the estimates vector $\hat{\boldsymbol{\theta}}_k$ coincide with the averages vector $\boldsymbol{m}_k$ at every time step k.

When the algorithms described are implemented in simulators like MATLAB, there are some precautions and considerations that must be done to provide the correct operation. In the following chapter this implementation will be shown.

Chapter 4

Algorithms implementation

The two proposed approaches have been implemented in MATLAB, to simulate how they actually work in a typical mmWave communications scenario. As said in the end of the last chapter, there are some issues that emerge and some precautions and considerations that must be taken to provide the correct operation.

To check the algorithms behavior two different simulators were built, both based on a Monte Carlo approach. This means that they were run cyclically for a large number of times, with a randomness given by the measurements noise and by the different trajectories of the mobile user, and setting equivalent seeds between the two approaches to provide a fair comparison in terms of AoAs RMSE and outlier rate, i.e., the probability that the RMSE evaluated is higher than a fixed threshold.

4.1 First approach

The flowchart of the algorithm is shown in Figure 4.1 and explained step-by-step in the following subsections.

Simulation parameters initialization

In this first step the user can set the key parameters of the Monte Carlo simulation:

- Number of scatterers and measurements, with the corresponding value of noise standard deviation σ.
- Area of the scenario, simulation time and simulation step Δt.

- Number of Monte Carlo iterations.
- Data fusion enabler flag and gyroscope noise standard deviation.
- Device rotation speed, acceleration spectral density $\boldsymbol{W_u}$ for the mobile user and angle acceleration spectral density $\boldsymbol{W_e}$ for the mobility model.
- Standard deviations of the Kalman filter initialization phase that affect the a-priori distributions.
- Percentage of values (simulation) on which evaluate the RMSE and threshold value for the outliers.

Every different combination of these values leads to a different result (in the last chapter will be shown the values of interest).

Base station and scatterers positions

These values are randomly chosen inside of the defined area. For every simulation there is a different set of positions.

Mobile user behavior

Here are defined the mobile station mobility model and the state-noise covariance matrix $\boldsymbol{Q_e}$, the random walk process whereby the mobile station moves and the device rotation angle for every time step (compared to the non-rotated reference system).

The random walk of the mobile user is governed by the following equations (written in matrix form), that represent the evolution of the device position over the time steps:

$$\begin{bmatrix} x_{u,k} \\ y_{u,k} \\ \dot{x}_{u,k} \\ \dot{y}_{u,k} \end{bmatrix} = \Phi_u \begin{bmatrix} x_{u,k-1} \\ y_{u,k-1} \\ \dot{x}_{u,k-1} \\ \dot{y}_{u,k-1} \end{bmatrix} + \boldsymbol{v}_{u,k} \tag{4.1}$$

where $x_{u,k}$ and $y_{u,k}$ are the coordinates of the mobile user at time step k, $\dot{x}_{u,k}$ and $\dot{y}_{u,k}$ are the speeds of the device in the two axes, $\boldsymbol{v}_{u,k} \sim N(0, \boldsymbol{Q_u})$ represents the Normal distributed noise, with zero mean and covariance matrix $\boldsymbol{Q_u}$ determined as in equation (1.5), while $\boldsymbol{\Phi}_u$ is the transition matrix and describes how the position evolve from the time step *k-1* to the time step k, that is:

$$\Phi_u = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

Base station and scatterers relative positions

The base station and scatterers positions are evaluated and compared to the local reference system, in which the mobile station is the center, i.e., how these objects move compared to the mobile station.

AoA noiseless measurement

Due to the relative positions calculated above, noiseless AoA measurements without the rotation term are given by an arctangent relation between the x and y coordinates. These values are known only to the simulator, i.e., are unknown to the estimation algorithm.

Phase correction

This is a fundamental step in the implementation of the simulator, indeed a first issue emerges with the use of the MATLAB function *atan2* to perform the arctangent. Since this function assumes values in the interval $[-\pi, \pi]$, a correction of the AoAs to avoid any possible phase jump is required. Only after this step is possible to add the rotation term to the noiseless measurements and again correct this eventual issue.

AoA noisy measurement and gyroscope measurement

Now, at every time step, the AWGN noise is added to the measurements and to the the gyroscope data, in different intensity due to the different value of standard deviation σ. The gyroscope data is evaluated in agreement with equation (2.6), with the only difference that the normalization for Δt is implicit.

Measurement and mobility models

This implicit normalization affects also the matrix $\boldsymbol{B}$ multiplied by the control vector $\boldsymbol{u}_k$, therefore $\boldsymbol{B} = \boldsymbol{G}$. The mobility model is defined in agreement with equation (2.4) and the process noise covariance matrix $\boldsymbol{Q_e}$ follows the

equation (1.5). Due to the linearity of the measurement noise, the mobility model is defined in this step too, and so its respective noise covariance matrix $\boldsymbol{R}$, in agreement with equation (2.7).

Kalman Filter

The estimation algorithm is the core of the simulator. It follows the steps listed in the section (2.4), starting with an initialization phase in which the a-priori AoA averages vector $\boldsymbol{m}_0$ (using the first noisy measurement of AoAs) and the corresponding error covariance matrix $\boldsymbol{P}_0$ are defined. Due to the higher value of the filter standard deviations compared to the one of the measurements, this first Gaussian distribution is very large, i.e., the a-priori estimate is affected by huge uncertainty. If the algorithm starts with a bad estimate of the state or with too high values of σ_{Kal} and σ_{Kal1}, it could not converge.

Then the real iteration starts with the phases of prediction and update. While the former agrees with equation (2.9), the latter is implemented in an equivalent but more stable version shown in the Appendix 2. Moreover, in the update phase, the innovation term $\boldsymbol{v}_k$ must be mapped in the interval $[-\pi, \pi]$ in order to provide the correct operation.

At every time step the angle estimate error is finally evaluated as the difference between the noiseless AoA measurements and the actual estimates $\hat{\boldsymbol{\theta}}_k$ (because of the MAP estimator is the averages vector $\boldsymbol{m}_k$), by comparing this value with the difference between the noiseless and noisy AoA measurements, i.e., without the effect of the filtering.

RMSE and outlier rate evaluation

In this last phase the three AoAs RMSE and the outlier rates, i.e., the probability that the algorithm does not converge thus providing an extremely large error, are evaluated. The outlier event is detected by comparing the RMSE with a fixed threshold and removed from the evaluation of the RMSE itself.

4.2 Second approach

The flowchart of the algorithm is shown in Figure 4.2 and explained step-by-step in the following subsections.

Simulation parameters initialization

In this first step the user can set the key parameters of the Monte Carlo simulation:

- Number of scatterers and measurements (and distance measurements enabler flag), with the corresponding value of noise standard deviations σ and σ_1.
- Area of the scenario, simulation time and simulation step Δt.
- Number of Monte Carlo iterations.
- Data fusion enabler flag and sensors noise standard deviations.
- Device rotation speed, acceleration spectral density $\boldsymbol{W_u}$ for the mobile user and acceleration spectral density $\boldsymbol{W_e}$ for the mobility model.
- Standard deviations of the Extended Kalman filter initialization phase that affect the a-priori distributions.
- Percentage of values (simulation) on which evaluate the RMSE and threshold values for the outliers (AoAs and positions).

Base station and scatterers positions

These values are randomly chosen inside of the defined area. For every simulation there is a different set of positions.

Mobile user behavior

Here are defined the mobile user mobility model and the state-noise covariance matrix $\boldsymbol{Q_e}$, the random walk process whereby the mobile user moves and the device rotation angle for every time step (compared to the non-rotated reference system).

The random walk of the mobile user, due to the same seed value set in the MATLAB implementation, is the same of the first approach and is governed by equations (4.1) and (4.2).

Base station and scatterers relative positions

Then the base station and scatterers position are evaluated and compared to the local reference system, in which the mobile station is the center, i.e., how these objects move compared to the mobile station.

AoA and distance noiseless measurement

Due to the relative positions calculated above, noiseless AoA measurements without the rotation term are given by an arctangent relation between the x and y coordinates. Moreover, with the corresponding flag active, also noiseless measurements of the distances between the mobile user and the three "sources" are exploited and given by a Pitagora theorem relation between the x and y coordinates. These values are known only to the simulator, i.e., are unknown to the estimation algorithm.

Phase correction

This is a fundamental step in the implementation of the simulator, indeed a first issue emerges with the use of the MATLAB function *atan2* to perform the arctangent. Since this function assumes values in the interval $[-\pi, \pi]$, a correction of the AoAs to avoid any possible phase jump is required. Only after this step it is possible to add the rotation term to the noiseless measurements and again correct this eventual issue.

AoA noisy measurement and sensors measurement

Now, at each time step, the AWGN noise is added to the measurements and to the the gyroscope and accelerometer data, with different intensity due to the different values of standard deviation σ. The gyroscope and accelerometer data are evaluated in agreement with equations (2.6) and (3.12) respectively, with the only difference that the normalization for Δt is implicit.

Gyroscope data mapping in terms of position displacement

Since the control vector $\boldsymbol{u}_k$ acts directly on the mobility model, in this approach a mapping of the angular speed to a space displacement in the Cartesian coordinates is required. This conversion follows the equation (3.10), in which d_{k-1} and φ_{k-1} are the distance and AoA noiseless measurement at the previous time step, while $\dot{\varphi}_{g_k}$ is the noisy gyroscope data at the actual time step. Therefore the randomness is given only by the sensor.

Measurement and mobility models

The implicit normalization cited above affects also the matrix $\boldsymbol{B}$ multiplied by the control vector $\boldsymbol{u}_k$, therefore $\boldsymbol{B} = \boldsymbol{G}$. The mobility model is defined in agreement with equation (3.6) and the process noise covariance matrix $\boldsymbol{Q_e}$ follows the equation (1.5). Due to the non-linearity of the measurement

noise, the mobility model is not defined in this step, but only its relative noise covariance matrix $\boldsymbol{R}$, in agreement with equation (3.14). The linearization $\boldsymbol{H}_k$ of $h(.)$ must be done inside of the filtering algorithm.

Extended Kalman Filter

The estimation algorithm starts with an initialization phase in which the a-priori AoA averages vector $\boldsymbol{m}_0$ (using the first noisy measurement of AoAs and distances) and the corresponding error covariance matrix $\boldsymbol{P}_0$ are defined. Due to the higher value of the filter standard deviations compared to those of the measurements, this initial Gaussian distribution is very large, i.e., the a-priori estimate is affected by huge uncertainty. If the algorithm starts with a bad estimate of the state or with too high values of σ_{Kal} and σ_{Kal1}, it could not converge.

Then the real iteration starts with the phases of prediction and update. While the former agrees with equation (3.20), the latter is implemented in an equivalent but more stable version shown in the Appendix 2. Moreover, in the update phase, the value of the function $h(.)$ is evaluated in agreement with equation (3.15) and the linearization $\boldsymbol{H}_k$ of $h(.)$ is performed, as shown in section (3.3). Due to the use of the function *atan2* to evaluate h, a phase correction is required to avoid any possible 2π-jump that could cause the not convergence of the algorithm.

For every time step the angle estimate error is finally evaluated as the difference between the noiseless AoA measurements and the actual estimates $\hat{\boldsymbol{\theta}}_k$ (because of the MAP estimator is the averages vector $\boldsymbol{m}_k$), by comparing this value with the difference between the noiseless and noisy AoA measurements, i.e., without the effect of the filtering. Note that the estimate of the AoAs are performed with the *atan2* giving the positions estimate, therefore it is necessary again a phase correction before the evaluation of the angle estimate error.

A similar procedure is followed to evaluate the position estimate error. In this case the comparison is between the noiseless distances measurements and the actual distances estimates (given by a Pitagora theorem relation between the x and y positions estimates), and without the effect of the filtering. It is important to note that to ensure a fair comparison, the values of the estimates must be mapped, for every time step, on the non-rotated local reference system, as shown in the Appendix 2.

RMSE and outlier rate evaluation

In this last phase the three AoAs and distances RMSE and the outlier rates are evaluated. The outlier event is detected by comparing the RMSE with a fixed threshold and removed from the evaluation of the RMSE itself.

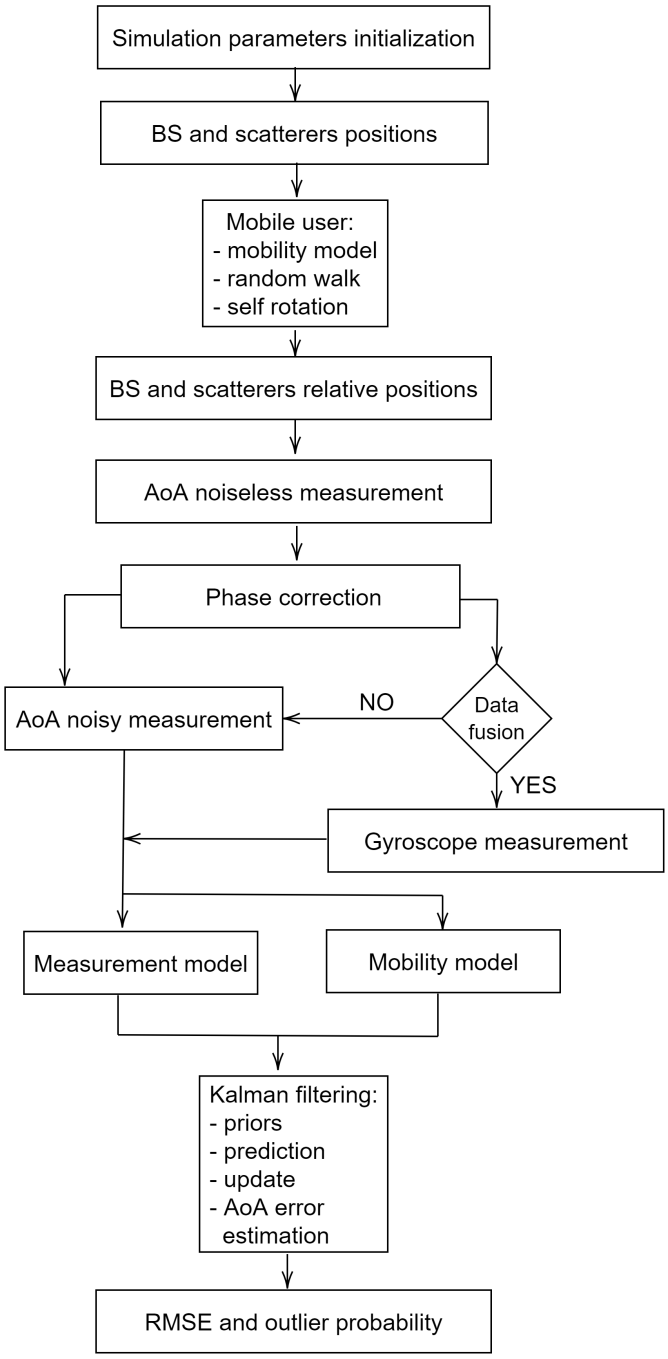

Figure 4.1: Flowchart of the first approach algoritm

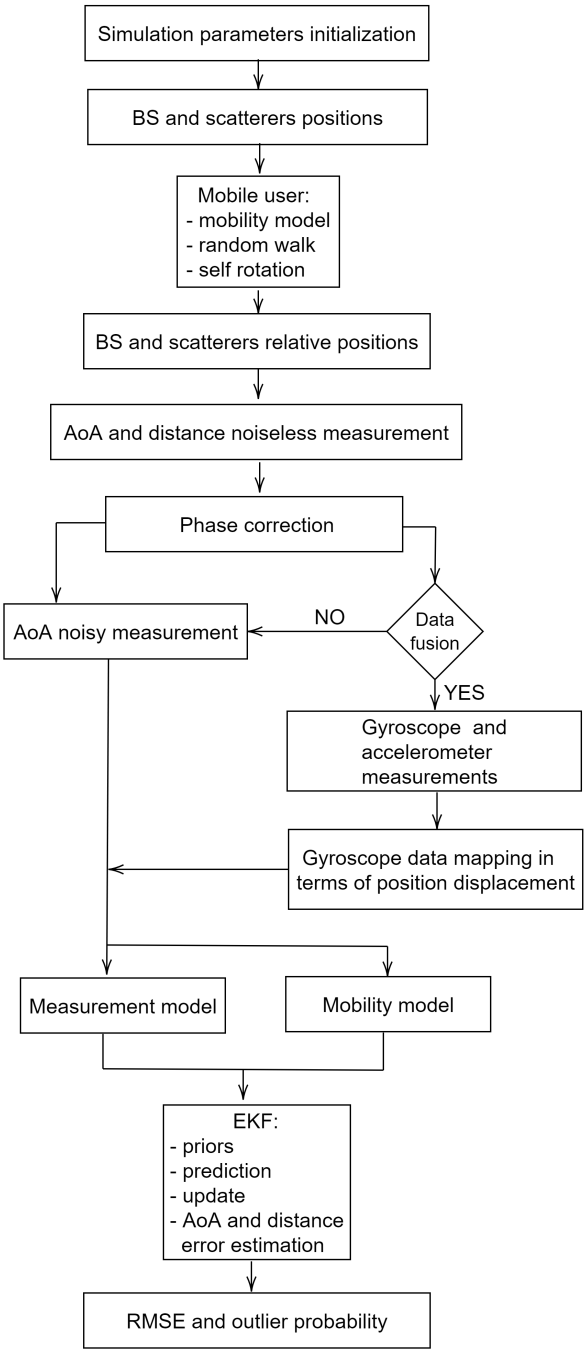

Figure 4.2: Flowchart of the second approach algoritm

Chapter 5

Simulations and results

In this last chapter the simulations performed to validate the approaches proposed and verify their goodness are shown. In these tests some parameters are fixed and others variables, in order to assess the algorithms in various operative situations, from the considered default condition to more critical ones.

In particular, the fixed parameters are the following:

- Number of base stations: 1.
- Number of scatterers: 2.
- Number of measurements: 3 for the first approach (AoAs) and 6 for the second approach (AoAs and distances between the mobile user and the three objects).
- Number of Monte Carlo iterations: 1000.
- Simulation step Δt: 10ms.

while the variable parameters are:

- Area of the scenario and simulation time.
- Device rotation speed, acceleration spectral density $\boldsymbol{W_u}$ for the mobile user and spectral density $\boldsymbol{W_e}$ for the mobility model.
- Measurements noise standard deviation σ and σ_1, i.e., the uncertainties that afflict the AoA and distance measurements (depending on the antennas), and distance measurements enabler flag for the second approach.

- Data fusion enabler flag and sensors noise standard deviations σ_g and σ_a, i.e., the uncertainties that afflict the angular speed and acceleration measurements of the embedded sensors.

- Standard deviations σ_{Kal} and σ_{Kal1} of the Kalman filter initialization phase that affect the a-priori distributions, i.e., the uncertainties of the a-priori state estimate.

- Threshold values for the outliers (AoAs and positions) and percentage of the simulation time considered in the RMSE evaluation, in order to verify how the transient behavior affects the RMSE value.

To give a complete view of the proposed approaches, i.e., of their advantages, issues and performances, the tests have been performed in the following operative conditions:

- Default settings with and without data fusion. With reference to these settings, the behavior of the approaches changing the duration of the simulations, the RMSE thresholds and the percentage of simulation in which evaluate them has been also shown.

- Changing the area of the scenario.

- Changing the rotation speed of the mobile user and its rotation direction.

- Changing the noise that affect the measurements and the sensors data.

- Changing the process noise spectral density of the mobile user, i.e., its movement in the space.

- Changing the process noise spectral density associated to the mobility model.

- Verification that the approaches do not diverge in critical conditions.

- Comparison, in term of RMSE, with the *State-Of-the-Art* (*SOA*) approaches from which the work takes inspiration.

In the following section will be shown only the relevant figures and results. For a more exhaustive treatment see the Appendix 1.

5.1 Default scenario

The default scenario is based on the typical human random walk, with an average speed of $3m/s$ (corresponding approximately to $\boldsymbol{W_u} = 1$) in a rectangular area of $200 \times 200m$, usual in non-dense urban scenarios. Moreover:

- Data fusion is enabled (and then disabled) and also the distance measurements for the second approach could be present or absent.
- The mobile user rotates 30 degrees per second counter-clockwise and the simulation time is set to $6s$ in order to appreciate a total rotation of 180 degrees and a movement of about ten/twenty meters.
- $\boldsymbol{W_e}$ of the mobility model is equal to the one of the mobile user $\boldsymbol{W_u}$ for the second approach, in order to avoid model mismatch, while for the first approach is fixed empirically to 0.001, in order to provide a good ratio between the process noise and the system inertia.
- Measurements standard deviations are set empirically to $\sigma = \pi/8$ (radiants) and $\sigma_1 = 10$ (meters).
- Sensors noise standard deviations are set empirically to $\sigma_g = 0.1$ [5] and $\sigma_a = 10$.
- Kalman standard deviations are set empirically to $\sigma_{Kal} = \pi/6$ (radiants) for the first approach and $\sigma_{Kal} = 20$ (meters) for the second approach, and $\sigma_{Kal1} = 0.01$ for both.
- Outliers threshold for AoA is set to 10 degrees, while for position is set to 5 meters. In the RMSE evaluation only the last 10% of the simulation time is considered in order not to include the transient behavior.

In the following sections the simulation results for the two approaches are shown in term of:

- AoA and position displacements between measurements and estimates.
- AoA and position estimates error with and without filtering (i.e., raw AoA measurements).

Finally a comparison in term of AoAs RMSE and outlier rates between the two approaches is carried out, changing some parameters from the default setting, like the distance measurement enabler for the second approach, the duration of the simulation, the percentage of the simulation interval on which evaluating the RMSE and the values of the thresholds for the AoA and distance outlier.

5.1.1 First approach

In this first simulation of the first approach behavior, data fusion is enabled.

The Figure 5.1 shows the movement of the mobile user in the area where the base station and the scatterers are located. For the default settings, the mobile user moves randomly, approximately for 20 meters in 6 seconds, i.e., at a typical human walk speed of $3m/s$. For this reason the objects remain quite far from the mobile user.

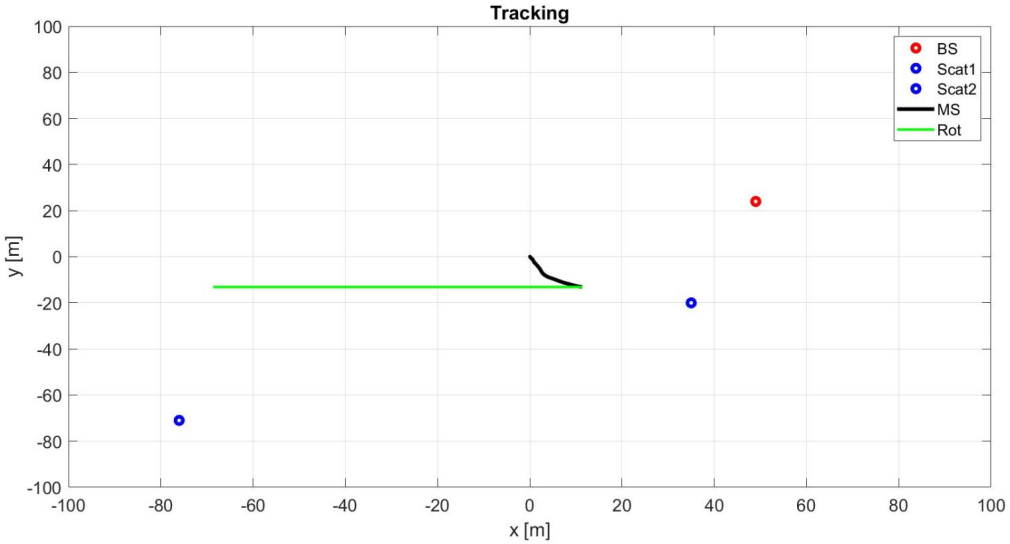

Figure 5.1: Default scenario, first approach. MU random walk and rotation in the 2D space. The green line evolves from zero to 180 degrees as index of the device rotation

In Figure (5.2) is reported the trend of the AoA measured with and without the measurement noise and the trend of the estimate. It is clear that, despite the noise that afflicts the measurement, the filtering manages to track the value of the true AoA. The descent trend is due to the counterclockwise device rotation, that is more relevant compared to the movement.

From it is also possible to deduce the angle correction cited in Chapter 4. It is clear in the trend of the AoA for scatterer1, where near time step 100 the value exceeds -180 degrees despite the function *atan2* provides results from $-\pi$ to π.

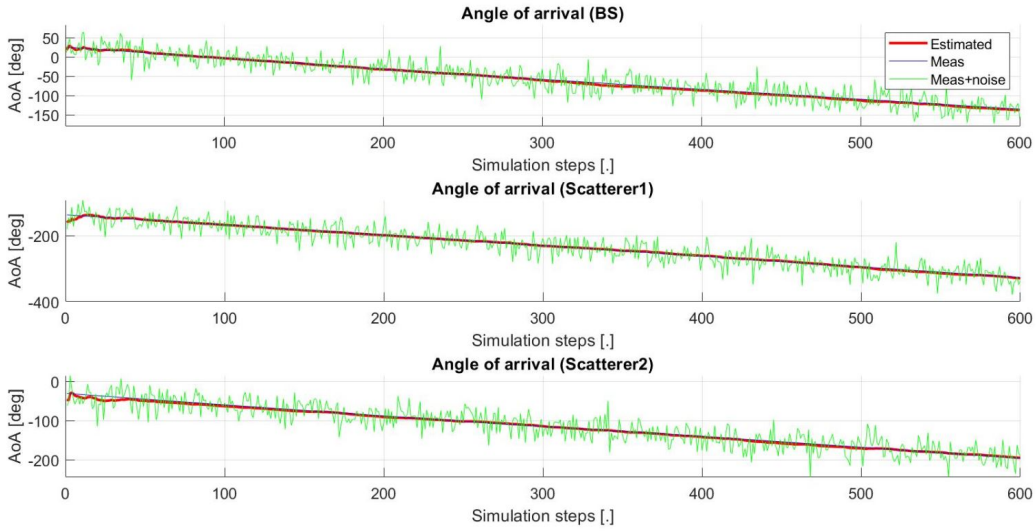

Figure 5.2: Default scenario, first approach. AoA measurements and estimates

In Figure 5.3 it is shown how the Kalman filter manages to reduce the angle estimate error.

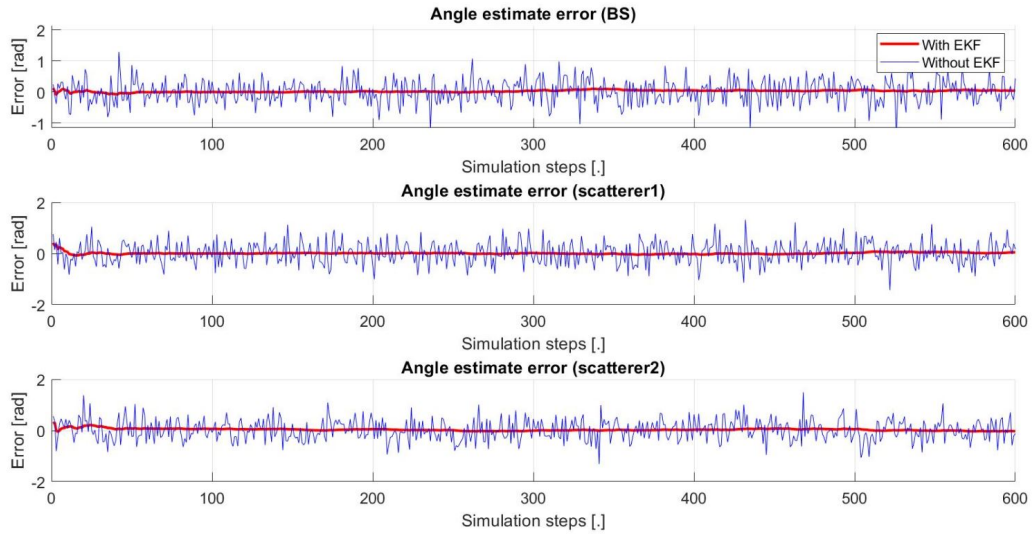

Figure 5.3: Default scenario, first approach. AoA estimate error comparison with and without estimation algorithm

Without data fusion

As it will be shown later, the data fusion disabling does not affect the results of the first approach in term of RMSE and outlier rate.

5.1.2 Second approach

It is simulated the behavior of the second approach with the use of data fusion. In this case, from the scenario representation is also possible to verify how the tracking acts. Indeed, for the base station and scatterers, it is clear how the positions estimates evolve from far locations, due to the high initial uncertainties, to locations close to the real ones. The movement and rotation behavior is according to the first approach.

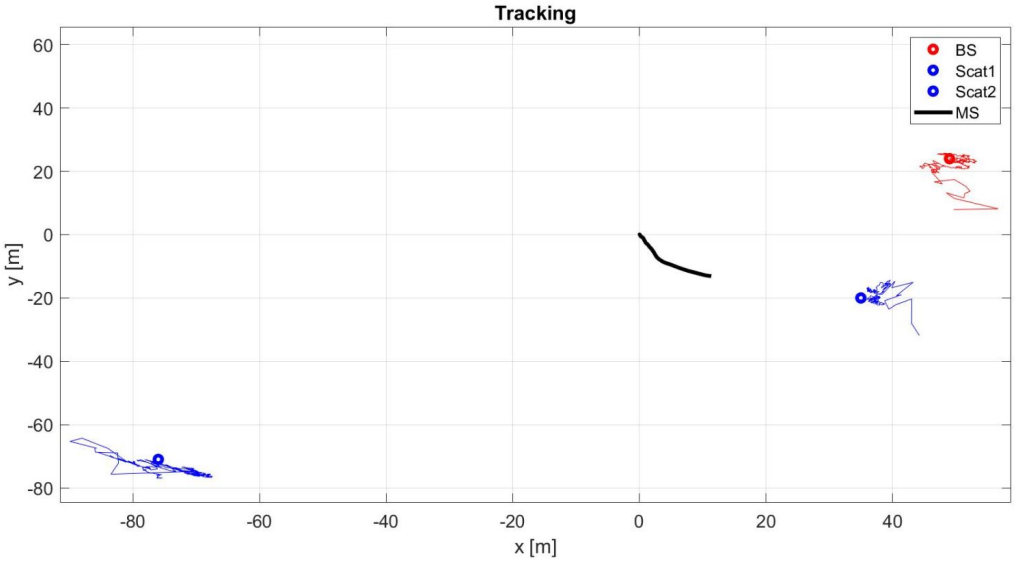

Figure 5.4: Default scenario, second approach. MU random walk and rotation in the 2D space, with position estimates

Therefore it is possible to appreciate the positions estimates evolution through the simulation steps. What have been said before is even more evident in Figure 5.5. The base station and the scatterers are fixed for all the duration of the simulation, but in the mobile user local reference system they appear to move. For this reason in Figure 5.5 also true positions evolve.

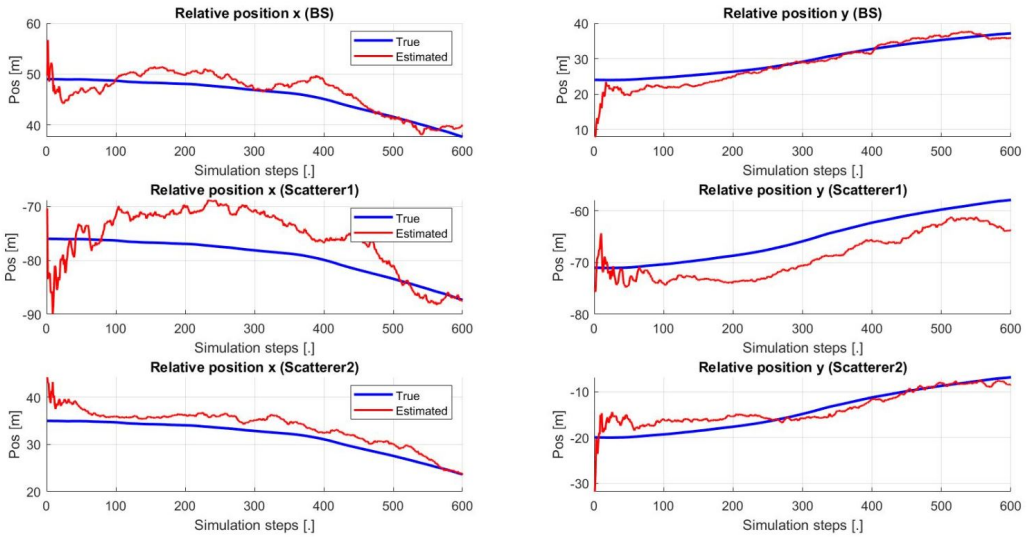

Figure 5.5: Default scenario, second approach. Positions measurements and estimates in the local reference system

The trends of the measured and estimate AoAs are the same compared to the ones of the first approach (Figure 5.6). Moreover, from Figures 5.7 and 5.8 it is possible to verify that both the position and angle estimation errors are strongly reduced thanks to the filtering.

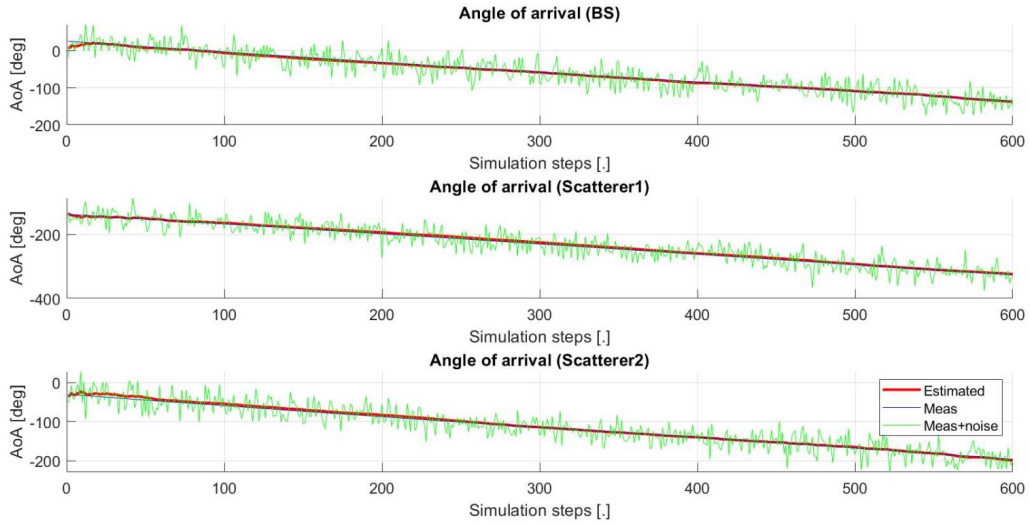

Figure 5.6: Default scenario, second approach. AoA measurements and estimates

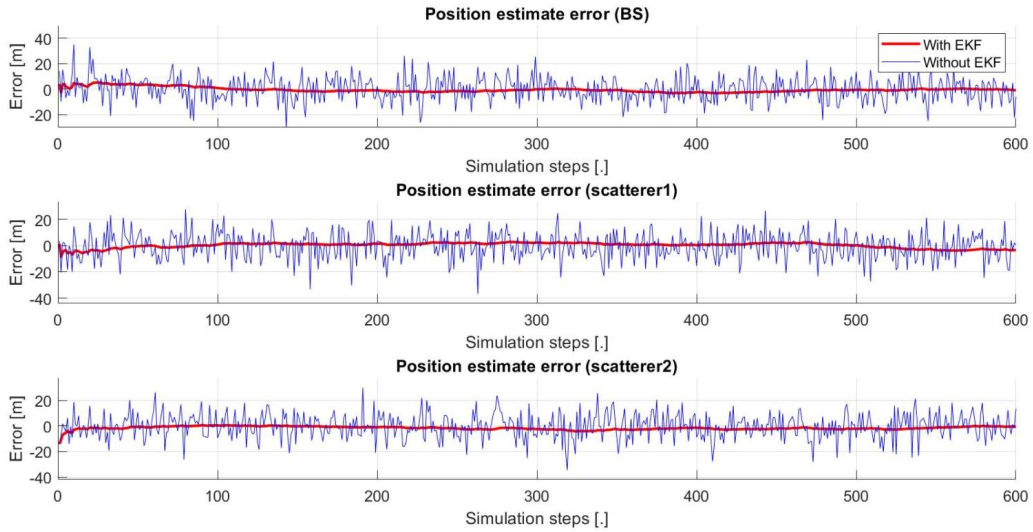

Figure 5.7: Default scenario, second approach. Positions estimate error comparison with and without estimation algorithm

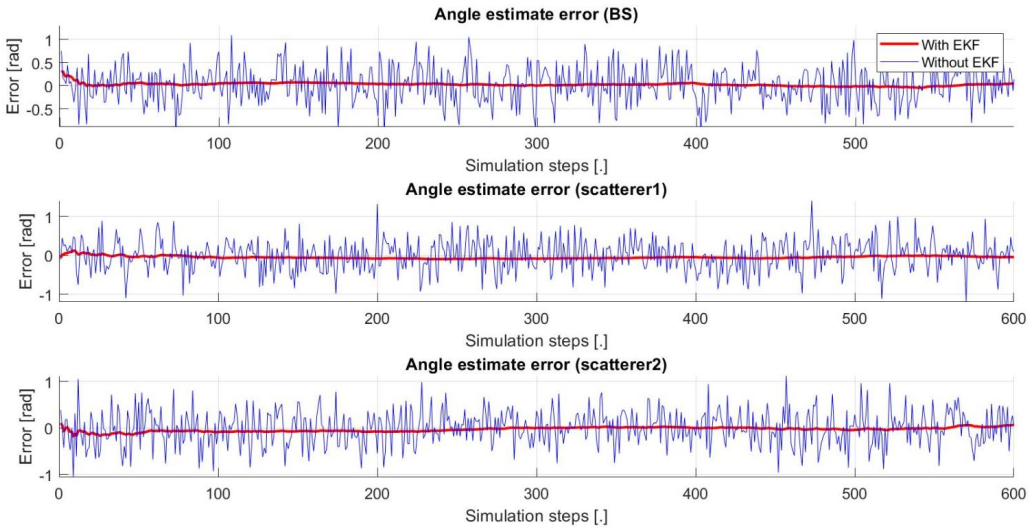

Figure 5.8: Default scenario, second approach. AoA estimate error comparison with and without estimation algorithm

Without data fusion

Unlike the first approach, in this case the data fusion disabling has a strong effect on the tracking performance. Just looking at the evolution of the estimate trajectory in the area (Figure 5.9) it is clear that the position is wrongly estimated, i.e., the algorithm drifts.

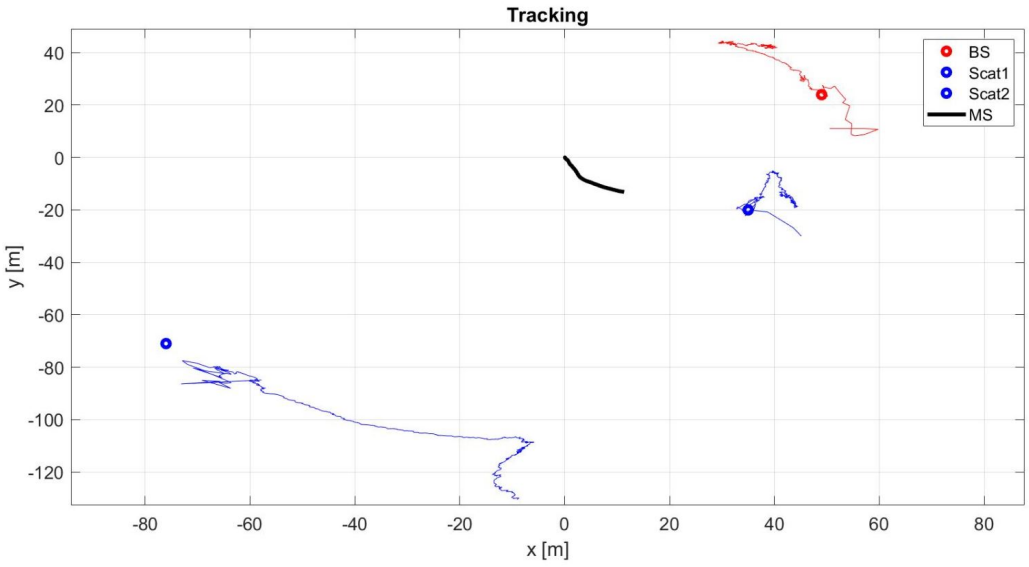

Figure 5.9: Default scenario, second approach, no data fusion. MU random walk and rotation in the 2D space, with position estimates

What has been said is confirmed by the estimate evolution of the base station and scatterers x and y coordinates in the local reference system of the mobile user (Figure 5.10) and by the trends of the AoA estimates compared to the ones of the measurements, which confirm than also the angle is wrongly estimated.

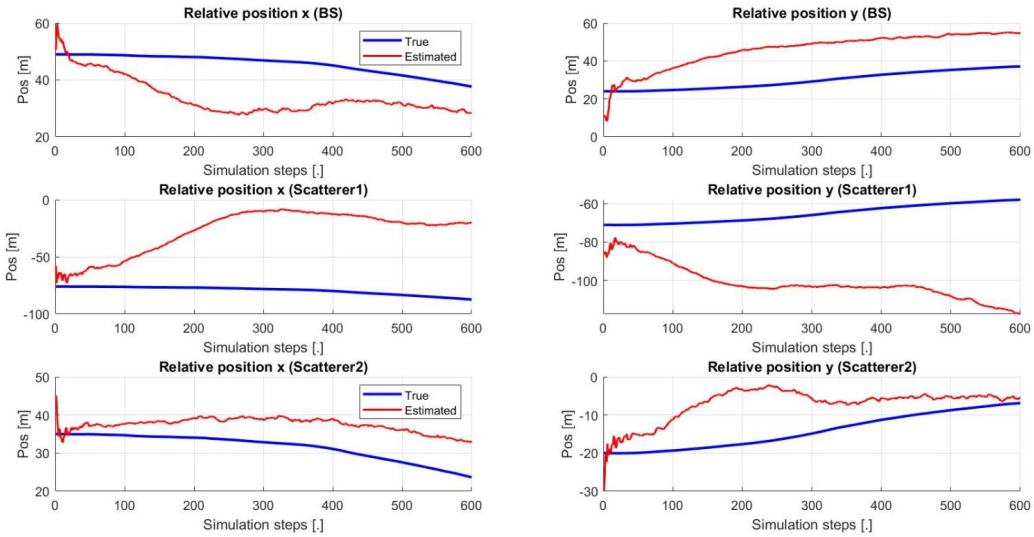

Figure 5.10: Default scenario, second approach, no data fusion. Positions measurements and estimates in the local reference system

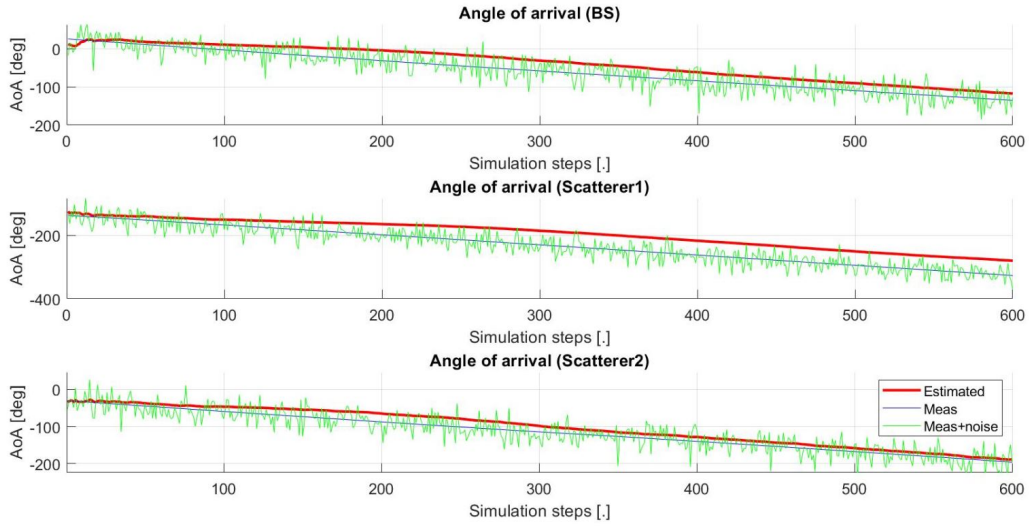

Figure 5.11: Default scenario, second approach, no data fusion. AoA measurements and estimates

A further confirm that estimate error grows compared to the one using data fusion, is given by Figures 5.12 and 5.13. Indeed, despite the smooth trends of the estimates, they drifts from the zero value. As will be shown in subsequent analysis, this problem is solvable with an appropriate increase of the value of the mobility model spectral density.

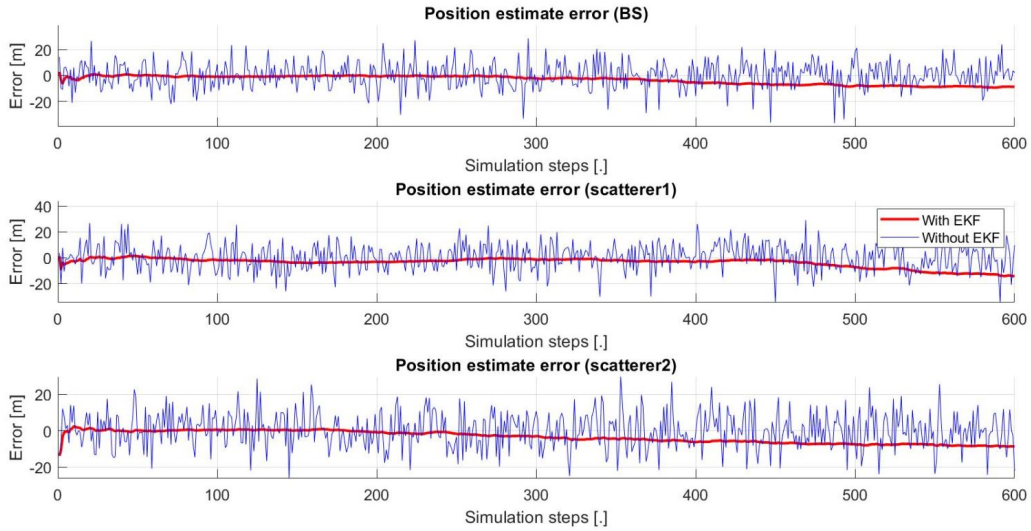

Figure 5.12: Default scenario, second approach, no data fusion. Positions estimate error comparison with and without estimation algorithm

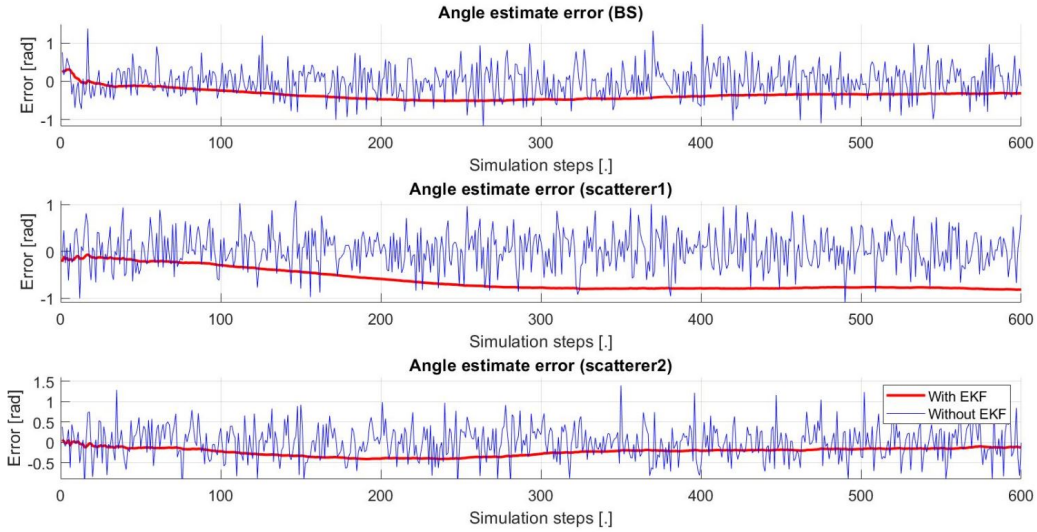

Figure 5.13: Default scenario, second approach, no data fusion. AoA estimate error comparison with and without estimation algorithm

5.1.3 Comparisons

In the following subsections some data comparisons about the simulations done in the two approaches will be shown, starting with the considerations

about data fusion enabling and disabling and then varying some of the parameters compared to the default setting. The performance is evaluated in term of AoA and distance RMSE conditioned to the probability that the estimates do not exceed fixed thresholds.

Data fusion

The Monte Carlo simulations provide the following values, reported in Table 5.1. For both the approaches the evaluated AoA RMSE is always lower than 3 degrees, that is a typical beamwidth for narrow beams [7], with a low probability that the evaluated values overcome the fixed threshold of 10 degrees. Moreover, for the second approach the distance estimate is available with a precision close to the meter despite a measurement noise standard deviation of 10 meters.

DEFAULT	First approach		Second approach			
	AoA RMSE (degrees)	AoA outlier rate	AoA RMSE (degrees)	AoA outlier rate	Pos. RMSE (meters)	Pos. outlier rate
BS	2.33	0.04	2.48	0.02	1.23	0
Scatterer1	2.08	0.01	2.14	0.02	1.22	0.02
Scatterer2	2.07	0.01	2.10	0.01	1.25	0.01

Table 5.1: Default scenario with data fusion. Comparison between the two approaches in term of RMSE and outlier rate

DEFAULT (no DF)	First approach		Second approach			
	AoA RMSE (degrees)	AoA outlier rate	AoA RMSE (degrees)	AoA outlier rate	Pos. RMSE (meters)	Pos. outlier rate
BS	2.47	0.04	6.05	0.74	2.54	0.71
Scatterer1	2.33	0.01	6.51	0.92	3	0.91
Scatterer2	2.24	0.01	6.20	0.91	2.97	0.91

Table 5.2: Default scenario, no data fusion. Comparison between the two approaches in term of RMSE and outlier rate

As said before, the data fusion disabling has a strong impact only on the performance of the second approach (Tables 5.2 and 5.3), making it useless due to the high outlier rate.

DF	First approach		Second approach			
	AoA RMSE (degrees)	AoA outlier rate	AoA RMSE (degrees)	AoA outlier rate	Pos. RMSE (meters)	Pos. outlier rate
YES	2.33	0.04	2.48	0.02	1.23	0
NO	2.47	0.04	6.05	0.74	2.54	0.71

Table 5.3: Default scenario. Comparison between the two approaches with and without data fusion in term of RMSE and outlier rate (relatively to the base station only)

Distance measurement disabled

As it can be appreciated in Table 5.4, disabling the distance measurement in the second approach leads to a worsening in the corresponding RMSE and mostly in the corresponding outlier rate. This means that the approach is still usable, but only for the AoA estimate. The little difference between the values of AoA RMSE and outlier rates with and without this extra information is due to the not perfect accuracy of Monte Carlo simulations.

RANGING	First approach		Second approach			
	AoA RMSE (degrees)	AoA outlier rate	AoA RMSE (degrees)	AoA outlier rate	Pos. RMSE (meters)	Pos. outlier rate
YES	-	-	2.48	0.02	1.23	0
NO	-	-	2.86	0	2.58	0.44

Table 5.4: Default scenario with data fusion, second approach. Effect of the disabling of the distance measurement (relatively to the base station only)

Duration of the simulation

The proposed approaches are stable and accurate also for higher simulation duration, as it can be noticed in Table 5.5.

TIME (s)	First approach		Second approach			
	AoA RMSE (degrees)	AoA outlier rate	AoA RMSE (degrees)	AoA outlier rate	Pos. RMSE (meters)	Pos. outlier rate
6 (default)	2.33	0.04	2.48	0.02	1.23	0
20	2.31	0.02	3.21	0.01	1.96	0.01

Table 5.5: Default scenario with data fusion. No drift of the approaches trough the time (relatively to the base station only)

Percentage of observation interval

The RMSE has been evaluated by considering an observation interval corresponding to the last 10% of the simulation time so that the initial transient is not accounted for the computation. In Table 5.6 it can be observed how enlarging the percentage of considered time steps causes a worsening in the AoA and distance RMSE for both the approaches.

% STEPS	First approach		Second approach			
	AoA RMSE (degrees)	AoA outlier rate	AoA RMSE (degrees)	AoA outlier rate	Pos. RMSE (meters)	Pos. outlier rate
10 (default)	2.33	0.04	2.48	0.02	1.23	0
50	2.58	0.04	2.82	0.02	1.37	0
100	3.10	0.04	3.40	0.02	1.62	0

Table 5.6: Default scenario with data fusion. RMSE variations increasing the observation interval (in percentage with respect to the simulation time) taken into account (relatively to the base station only)

AoA and distance thresholds

The RMSE and outlier rates are conditioned also by the threshold values chosen. In particular, for both the approaches and for both the threshold type, i.e., AoA and distance, an increase value provides a worsening of the corresponding RMSE but an improvement in term of outlier rate. This behavior is evident in Tables (5.7) and (5.8).

THRESHOLD (AoA)		First approach			Second approach			
Data fusion	**Threshold (degrees)**	AoA RMSE (degrees)	AoA outlier rate	**Threshold (degrees)**	AoA RMSE (degrees)	AoA outlier rate	Pos. RMSE (meters)	Pos. outlier rate
YES	**3**	1.59	0.28	**3**	1.58	0.32	1.23	0
	5	2.04	0.10	**5**	2.12	0.10	1.23	0
	10	2.33	0.04	**10**	2.48	0.02	1.23	0
	20	2.62	0.01	**20**	2.51	0.02	1.23	0
NO	**3**	1.64	0.32	**5**	3.13	0.91	2.54	0.71
	5	2.15	0.11	**10**	6.05	0.74	2.54	0.71
	10	2.47	0.04	**20**	10.84	0.43	2.54	0.71
	20	2.80	0.01	**30**	15.14	0.18	2.54	0.71

Table 5.7: Default scenario. Effect of the AoA threshold on the RMSE and outlier rate (relatively to the base station only)

THRESHOLD (position)		First approach		Second approach			
Data fusion	**Threshold (meters)**	AoA RMSE (degrees)	AoA outlier rate	AoA RMSE (degrees)	AoA outlier rate	Pos. RMSE (meters)	Pos. outlier rate
YES	**1**	-	-	2.48	0.02	0.62	0.53
	2	-	-	2.48	0.02	0.99	0.16
	3	-	-	2.48	0.02	1.16	0.03
	5	-	-	2.48	0.02	1.23	0
NO	**2**	-	-	6.05	0.74	1.06	0.89
	5	-	-	6.05	0.74	2.54	0.71
	10	-	-	6.05	0.74	5.82	0.25
	20	-	-	6.05	0.74	7.18	0

Table 5.8: Default scenario, second approach. Effect of the distance threshold on the RMSE and outlier rate (relatively to the base station only)

In the following section the behavior of the algorithms for displacement from the default setting is investigated, in particular changing the area of the scenario, the device speed rotation, the measurements and sensors data noise, the process noise spectral density of the mobile user and mobility model.

5.2 Area reduction effects

The simulation results show that area decreasing provides a worsening in term of AoA RMSE and outlier rate for both the approaches. In particular the second one reacts better to this issue and manages to maintain acceptable values of the parameters. Moreover, for the second approach, area reduction provides also a little improvement in term of distance RMSE, which becomes a large improvement considering the absence of data fusion.

AREA		**First approach**		**Second approach**			
Data fusion	**Area (meters^2)**	AoA RMSE (degrees)	AoA outlier rate	AoA RMSE (degrees)	AoA outlier rate	Pos. RMSE (meters)	Pos. outlier rate
YES	**25x25**	4.04	0.25	3.61	0.13	1.18	0
	50x50	3.47	0.15	3.34	0.08	1.18	0
	100x100	2.82	0.08	2.95	0.03	1.21	0
	200x200	2.33	0.04	2.48	0.02	1.23	0
NO	**25x25**	4.04	0.25	5.52	0.36	1.46	0
	50x50	3.53	0.15	5.53	0.37	2.08	0.07
	100x100	3.01	0.08	5.96	0.50	2.50	0.42
	200x200	2.47	0.04	6.05	0.74	2.54	0.71

Table 5.9: Effects of the scenario area reduction on the RMSE and outlier rates (relatively to the base station only)

In Appendix 1 details on position estimates evolution as a function of the area are reported.

5.3 Device rotation effects

In real scenarios it is expected that user rotates him/herself or the mobile device. Therefore it is important to investigate the impact of device rotation on the performance of the two algorithms proposed. As Table 5.10 explains, both of the algorithms proposed react well to that type of displacement, in particular with data fusion, providing a good performance.

Without data fusion instead, the outlier rates of the second approach drop with at higher rotation speed. The reason why the performances of the first approach remain stable is because this approach provide good estimates despite higher values of process noise $\boldsymbol{W}_e$, therefore the algorithm has inherently less inertia respect to the second one and can react faster in critical situations (like the increase rotation speed).

ROT		First approach		Second approach			
Data fusion	**Rotation (degrees/s)**	AoA RMSE (degrees)	AoA outlier rate	AoA RMSE (degrees)	AoA outlier rate	Pos. RMSE (meters)	Pos. outlier rate
YES	**0**	2.33	0.04	2.86	0.02	1.43	0
	10	2.33	0.04	2.88	0.02	1.33	0
	30	2.33	0.04	2.48	0.02	1.23	0
	90	2.33	0.04	2.85	0.02	1.62	0.01
NO	**0**	2.33	0.04	2.23	0.01	1.04	0
	10	2.34	0.04	3.34	0.02	1.54	0
	30	2.47	0.04	6.05	0.74	2.54	0.71
	90	3.61	0.05	7.56	0.98	0.40	0.88

Table 5.10: Effects of the variation of the device rotation speed on the RMSE and outlier rates (relatively to the base station only)

A further verification of the correct algorithm implementation is done by inverting the rotation direction of the device. As result, the trend of the AoA is inverted compared to the default one, i.e., increases with time as visible in Figure 5.14.

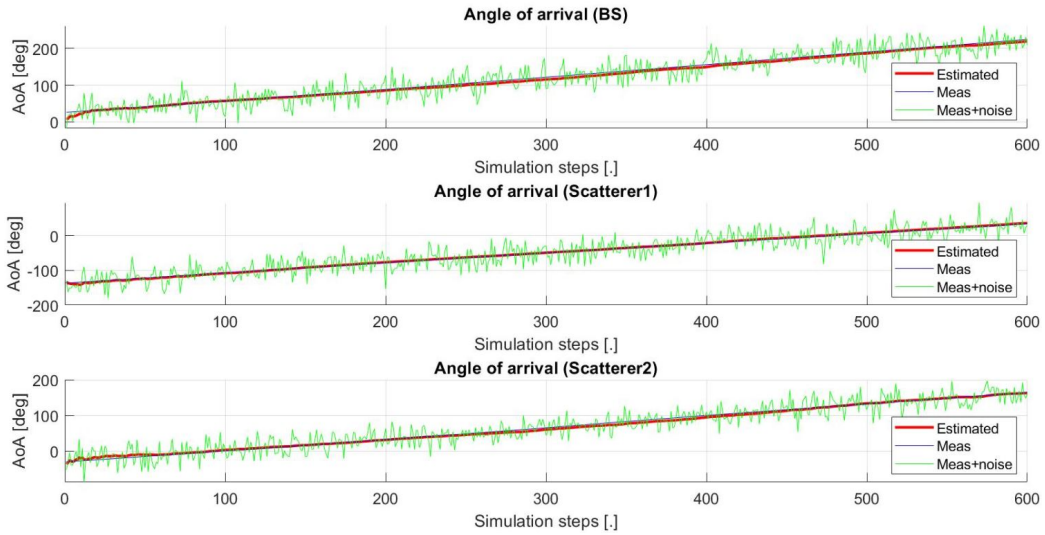

Figure 5.14: Default scenario. AoA measurements and estimates in condition of opposite device rotation direction

ROT	First approach		Second approach			
	AoA RMSE (degrees)	AoA outlier rate	AoA RMSE (degrees)	AoA outlier rate	Pos. RMSE (meters)	Pos. outlier rate
Counter-clockwise	2.33	0.04	2.48	0.02	1.23	0
Clockwise	2.33	0.04	2.62	0.02	1.20	0

Table 5.11: Default scenario. Verification of correct behavior also for clockwise rotation (relatively to the base station only)

5.4 Measurements and sensor data noise effects

The estimate processes are noise sensitive, because the increase of measurements and sensors noise, i.e., of their standard deviations σ, σ_1, σ_g and σ_a, affects the algorithms performances. The following simulations show that the setting of Kalman filter standard deviations does not always permit to reduce the noise effect. In particular, the outlier rates in Table 5.12 are still too high for a practical adoption of the algorithm.

NOISE $\sigma = \frac{3}{8}\pi$; $\sigma_1 = 30$ $\sigma_g = 0.3$; $\sigma_a = 30$	First approach			Second approach			
σ_{Kal} σ_{Kal1}	AoA RMSE (degrees)	AoA outlier rate	σ_{Kal} σ_{Kal1}	AoA RMSE (degrees)	AoA outlier rate	Pos. RMSE (meters)	Pos. outlier rate
$\pi/12$ **0.001**	4.53	0.14	**2 0.001**	5.34	0.53	2.62	0.40
$\pi/6$ **0.01**	5.55	0.14	**20 0.01**	5.07	0.34	2.60	0.32
$\pi/3$ **0.1**	4.63	0.17	**200 0.1**	5.12	0.39	2.58	0.32

Table 5.12: Effects of the measurements noise and sensors data noise increase. What happen to the RMSE and outlier rates (relatively to the base station only) with the variation of Kalman filter a-priori variances

However, it is important to specify that the noise values set, in particular $\sigma = 3\pi/8$, is quite high compared to typical conditions and it could be considered as a worst case scenario.

5.5 Process noise spectral density of the mobile user effects

The process noise spectral density of the mobile user is strictly related to the device movement, i.e., an increase value of $\boldsymbol{W_u}$ provides a faster movement in the 2D scenario.

The algorithms suffer from that displacement between the process noise spectral density of the mobile user and the one of the mobility model. In practical scenarios, this displacement corresponds to a model mismatch, i.e., the inertia of the filter does not match that actually experienced by the mobile user. The consequence is again a worsening of the performance in term of AoA and distance RMSE and outlier rates. To overcome this issue it is possible, as show in Table 5.13 and in Appendix 1, to increase the process noise spectral density of the mobility model.

W_u (MU)		First approach			Second approach			
Data fusion	**W_u**	AoA RMSE (degrees)	AoA outlier rate	**W_u**	AoA RMSE (degrees)	AoA outlier rate	Pos. RMSE (meters)	Pos. outlier rate
YES	**0.001**	1.86	0	**0.001**	2.21	0.02	1.06	0
	1	2.33	0.04	**1**	2.48	0.02	1.23	0
	10	3.32	0.12	**10**	3.64	0.07	1.98	0.04
	100	4.24	0.29	**100**	4.68	0.28	2.75	0.43
	100 (W_e=0.1)	3.77	0.03	**100 (W_e=100)**	3.33	0.02	1.90	0
NO	**0.001**	2.08	0	**0.001**	5.82	0.70	4.68	0.28
	1	2.47	0.04	**1**	6.05	0.74	2.54	0.71
	10	3.32	0.13	**10**	5.57	0.83	2.68	0.63
	100	4.24	0.29	**100**	5.32	0.92	2.67	0.56
	100 (W_e=0.1)	3.71	0.03	**100 (W_e=100)**	5.63	0.28	2.36	0.04

Table 5.13: Effects of the variation of the process noise spectral density of the mobile user on the RMSE and outlier rates (relatively to the base station only)

5.6 Process noise spectral density of the mobility model effects

As said above, the process noise spectral density of the mobility model plays a very important role in the estimation algorithms. In particular, the system inertia depends on it. In other words, an higher $\boldsymbol{W_e}$ provides a better reactivity of the process, that could cope with conditions and displacements, reducing the outlier rate. On the other side an higher $\boldsymbol{W_e}$ might cause an higher RMSE.

What has just been said is clear from Table 5.14 and Appendix 1.

W_e ($W_u = 10$)		First approach			Second approach			
Data fusion	W_e	AoA RMSE (degrees)	AoA outlier rate	W_e	AoA RMSE (degrees)	AoA outlier rate	Pos. RMSE (meters)	Pos. outlier rate
YES	**0.0001**	3.86	0.21	**0.1**	4.46	0.13	2.12	0.15
	0.001	3.32	0.12	**1**	3.64	0.07	1.98	0.04
	0.01	3.02	0.05	**10**	3.01	0.02	1.49	0
	0.1	3.68	0.01	**100**	3.27	0.02	1.73	0
NO	**0.0001**	6.48	0.80	**0.1**	5.59	0.96	2.54	0.52
	0.001	3.32	0.13	**1**	5.57	0.83	2.68	0.63
	0.01	2.96	0.05	**10**	5.38	0.34	2.52	0.23
	0.1	3.66	0.01	**100**	3.98	0.02	2.12	0.07

Table 5.14: Effects of the variation of the process noise spectral density of the mobility model on the RMSE and outlier rates (relatively to the base station only)

5.7 Stability in critical conditions

To demonstrate the reliability of the proposed algorithms, they are tested also under critical conditions, i.e., for a reduced area and faster rotation and movement compared to the default settings. In particular:

- Area: $100 \times 100m$;
- Rotation speed: $90deg/s$.
- $\boldsymbol{W_u} = 10$. For the second approach also $\boldsymbol{W_e}$ of the mobility model was set to this value, while for the first approach it is set to 0.005 in order to appreciate the performance variations.

Simulation results in Table 5.15 confirm that both of the algorithms provide acceptable performance.

CRITICAL	First approach		Second approach			
	AoA RMSE (degrees)	AoA outlier rate	AoA RMSE (degrees)	AoA outlier rate	Pos. RMSE (meters)	Pos. outlier rate
BS	3.43	0.10	4.03	0.07	2.30	0.08
Scatterer1	3.18	0.08	3.97	0.09	2.07	0.05
Scatterer2	3.25	0.07	4.10	0.08	2.11	0.06

Table 5.15: Behavior of the approaches in critical conditions of mobile user speed and rotation

5.8 Comparison with the state-of-the-art approaches

The last results concern the comparison between the proposed approaches and those presented in reference papers [4] and [5].

Two different comparisons have been done and, to provide a fair comparison, the following parameters have been set in order to reproduce the same scenarios as the reference papers ones:

- The first one between one of the simulations performed in [4] and the second approach, where the mobile user is supposed to be fixed and rotating at $10deg/s$ for 10 seconds ($\boldsymbol{W_u} = 0.0001$), in a $25 \times 25m$ area.

- The second one between one of the simulations performed in [5] and the second approach, where the mobile user is supposed to move for 5 meters at constant speed for 10 seconds ($\boldsymbol{W_u} = 0.02$), in a $25 \times 25m$ area.

To ensure these situations the following parameters were also set: $\sigma_1 = 3$; $\sigma_g = 0.1$; $\sigma_a = 1.3$ [4] [5].

The results show that, under these conditions, the proposed approach provides better performance in terms of RMSE position compared to the ones of the reference papers.

COMPARISON SOA	Second approach				
	Pos. RMSE (meters)	Pos. outlier rate		Pos. RMSE (meters)	Pos. outlier rate
Proposed	0.38	0	**Proposed**	0.39	0
SOA 2016	0.56	-	**SOA 2017**	0.96	-

Table 5.16: Comparison between the second approach and the reference papers [4] [5] in term of distance RMSE

Conclusion

This work takes inspiration from existing techniques for multipath tracking, in order to design innovative algorithms that aim to improve the performance also by integrating measurements from inertial devices through data fusion approaches.

In particular, it starts with the idea proposed in [2], that is a state-space approach for detection and tracking of MIMO propagation path parameters. In order to provide an improved performance, especially in mmWave communications, sensor fusion results to be a smart opportunity and easy to integrate in the existing approaches. Works [9] [4] [5] provide state-of-the-art approaches in the 3D mmWave beam tracking using data fusion strategy.

The idea proposed in this work was to integrate the state-space approach with the data fusion, to create an efficient algorithm for tracking the parameters of the channel paths. Two different strategies have been implemented: the first one uses AoA measurements between the received multipath and the user to track the AoA itself and its rate of change, while the second one uses AoA and distance measurements in order to estimate the relative position and speed of scatterers present in the scenario from which infer the AoA of the multipath components.

After a first stage of mathematical modeling of the two approaches, they have been implemented and tested in MATLAB, by developing a simulator in which the user can set multiple parameters depending on the desired scenario. The analysis done showed the robustness of the proposed strategies compared to the reference ones in different operating conditions. In particular:

- The two approaches provide the same performances in not critical conditions and with data fusion, but the latter can give also a position estimate. Moreover, the first approach is more stable in absence of data fusion.

- In general, data fusion provides better performances for both the algorithms, mostly in critical conditions of high mobile user speed and high device rotation speed.

- An excessive measurements and sensors noise can cause a divergence of both the algorithms and a sharp performances degradation. Moreover, there are some process noise spectral density values that provide the best compromise between the RMSE and the outlier rates.

- The evaluated performances are better compared to the one of the reference papers [4] [5] in the same scenarios.

Despite the promising results obtained, there are for sure many issues that should be tackled to further improve the performance. First of all, the state vector could be enriched with more parameters, e.g., Time-of-Arrival, amplitude and phase of the receiving multipath components, in order to provide a better estimation and knowledge of the CSI.

Moreover, the algorithms proposed can be improved by performing a chain approach where the estimates of the first one could be used as "filtered measurement" of the second one in order to make the latter more robust to noise. A further improvement could be the implementation of an adaptive algorithm that sets the best process noise spectral density depending on the operating condition.

Finally, many of the noise parameters were set empirically, therefore it could be useful to test the proposed strategies in real devices and environments, to obtain reliable results.

Appendix 1: Figures

Effect of the absence of distance measurements in the second approach

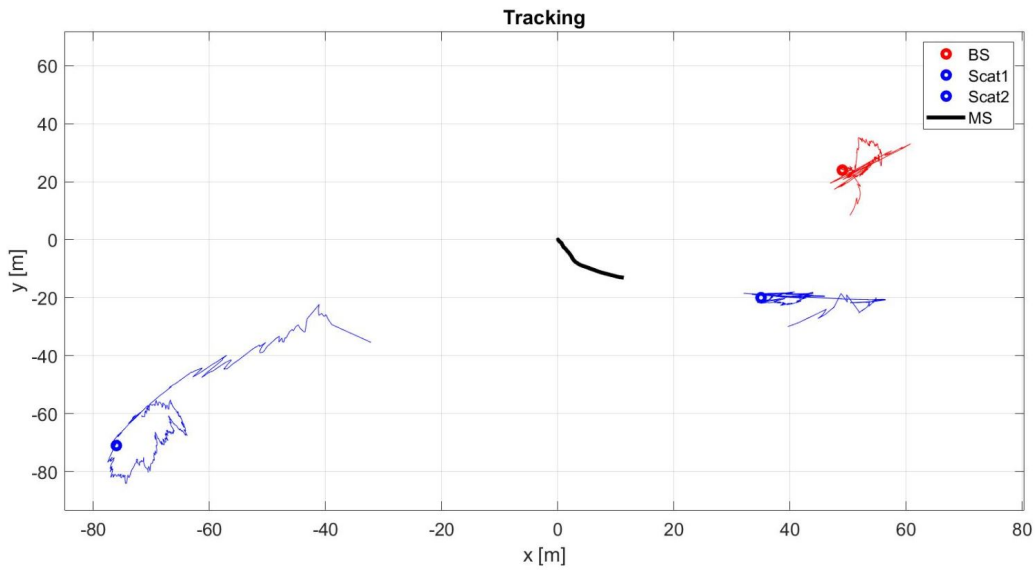

Figure 5.15: Default scenario, second approach. Effect of the disabling of the distance measurement on the relative base station and scatterers positions tracking

Effect of the simulation time

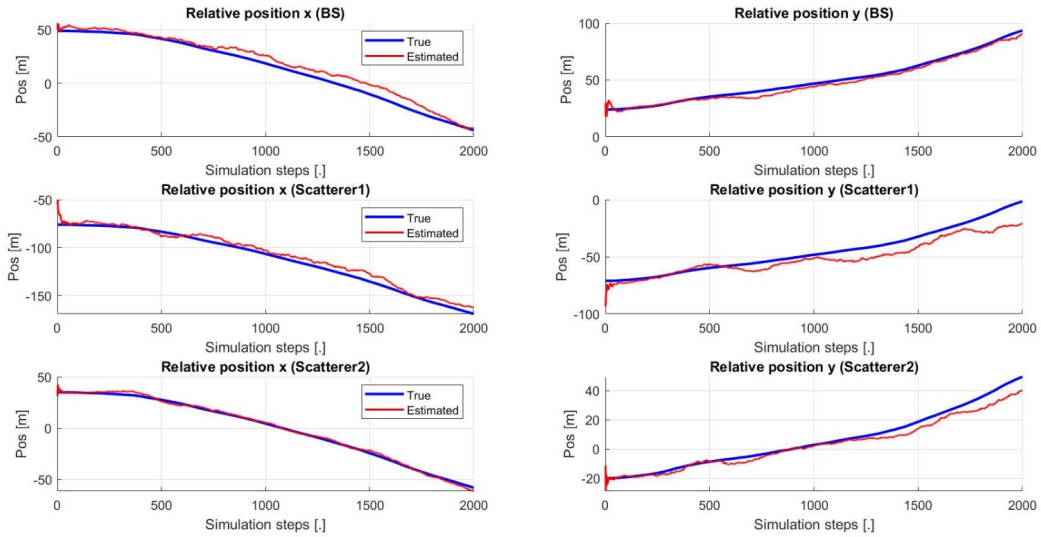

Figure 5.16: Default scenario, second approach. Effect of the simulation time increase on the position estimate

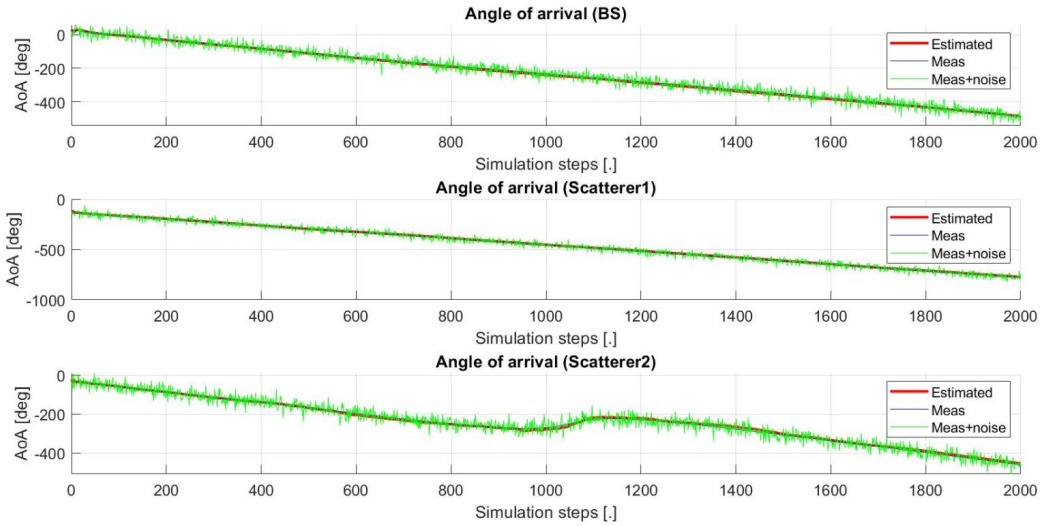

Figure 5.17: Default scenario, second approach. Effect of the simulation time increase on the AoA estimate

Effect of the area reduction

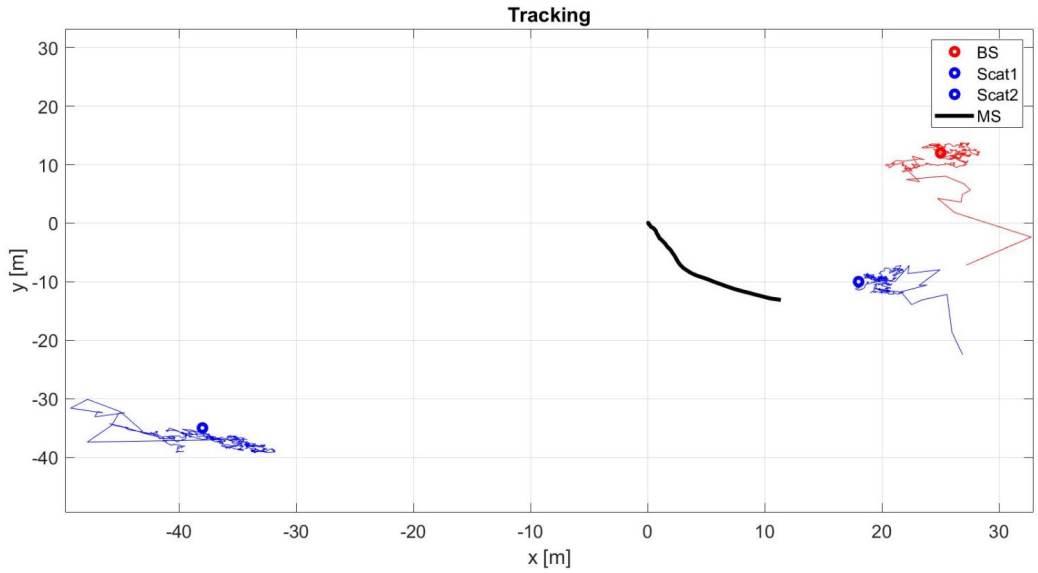

Figure 5.18: Default scenario, second approach. MU random walk and rotation in the 2D space, with position estimates in a $100 \times 100m$ rectangular area

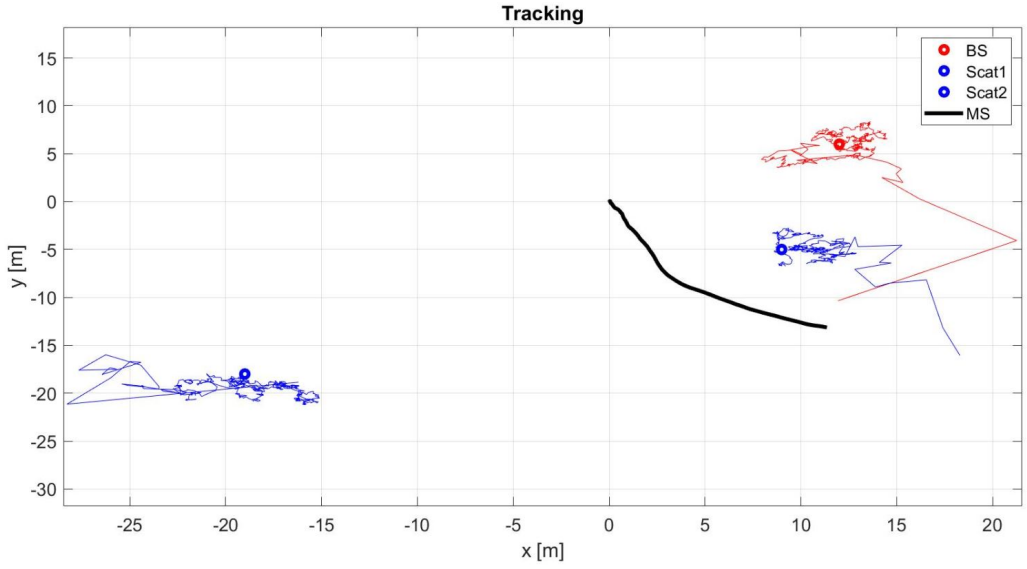

Figure 5.19: Default scenario, second approach. MU random walk and rotation in the 2D space, with position estimates in a $50 \times 50m$ rectangular area

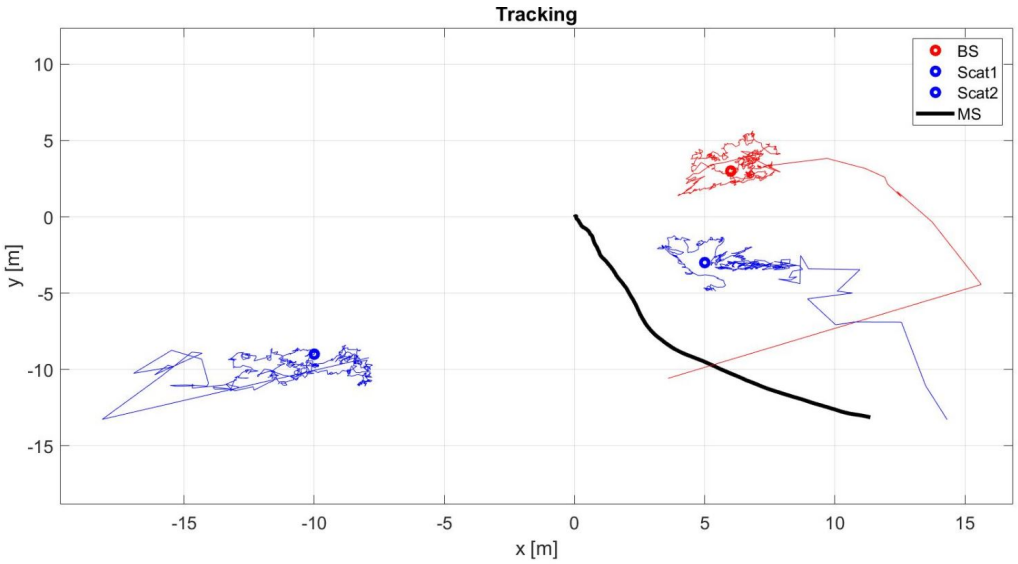

Figure 5.20: Default scenario, second approach. MU random walk and rotation in the 2D space, with position estimates in a $25 \times 25m$ rectangular area

Effect of device speed rotation

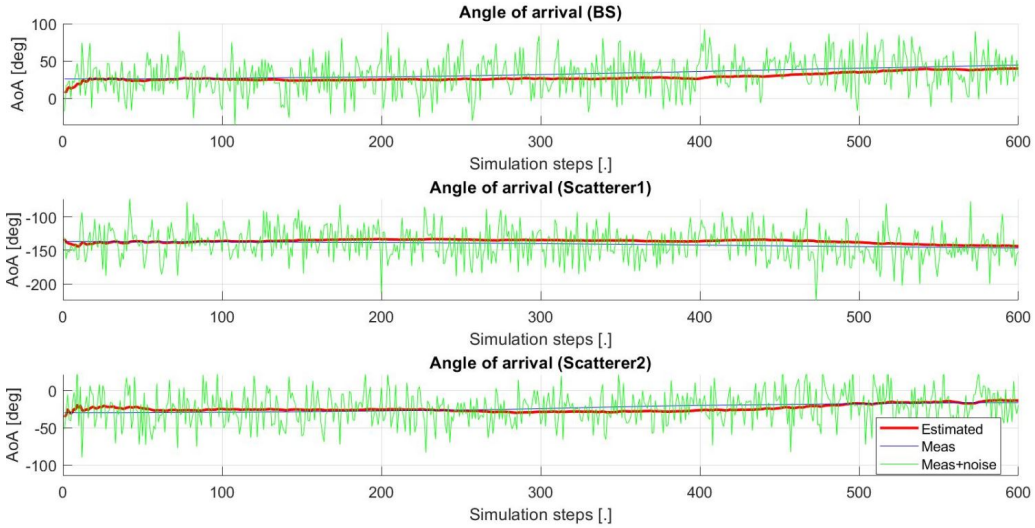

Figure 5.21: Default scenario, second approach, MU random walk and non-rotating in the 2D space. Effect on the AoA estimate

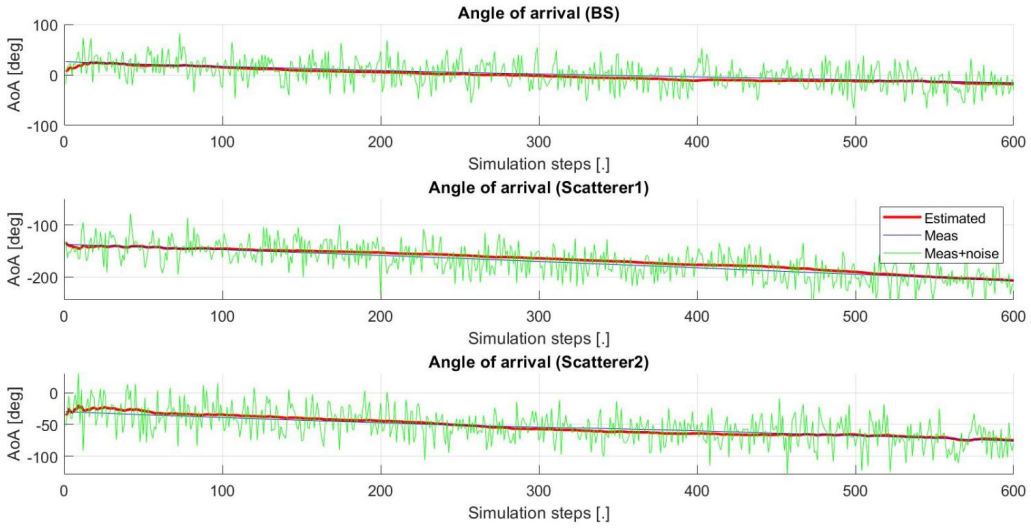

Figure 5.22: Default scenario, second approach, MU random walk and rotating at 10 degrees per second in the 2D space. Effect on the AoA estimate

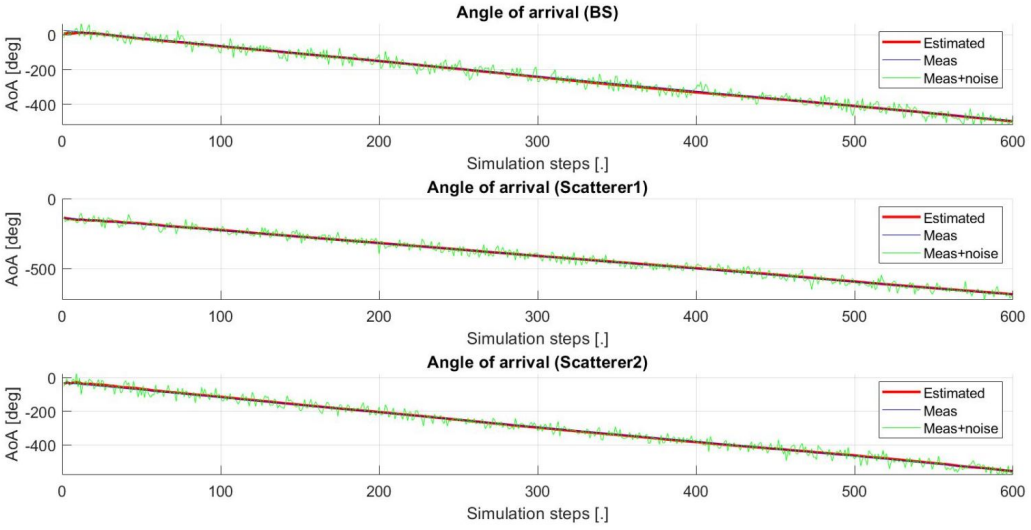

Figure 5.23: Default scenario, second approach, MU random walk and rotating at 90 degrees per second in the 2D space. Effect on the AoA estimate

Effect of the process noise spectral density of the mobile user

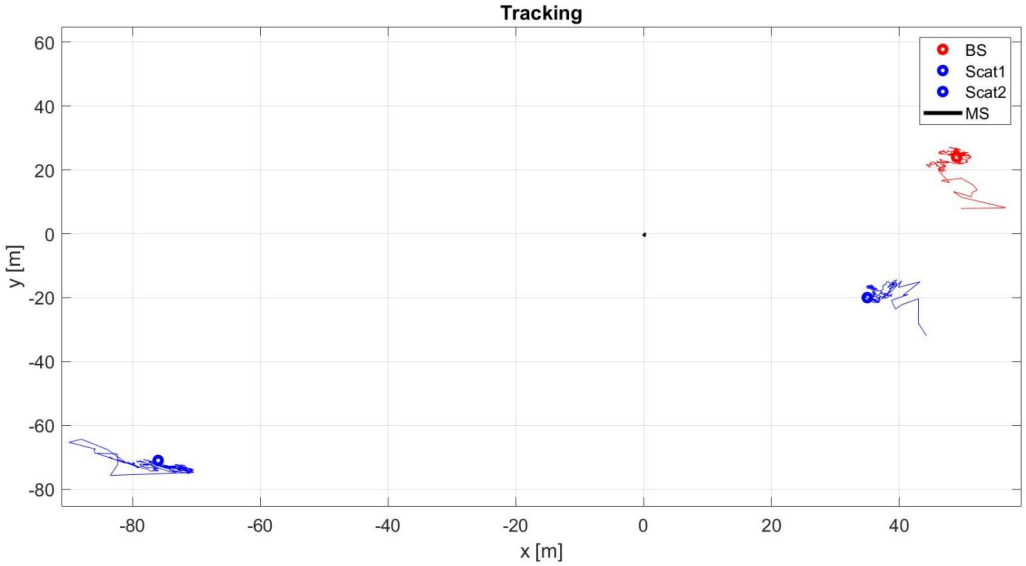

Figure 5.24: Default scenario, second approach. MU fixed and rotating in the 2D space

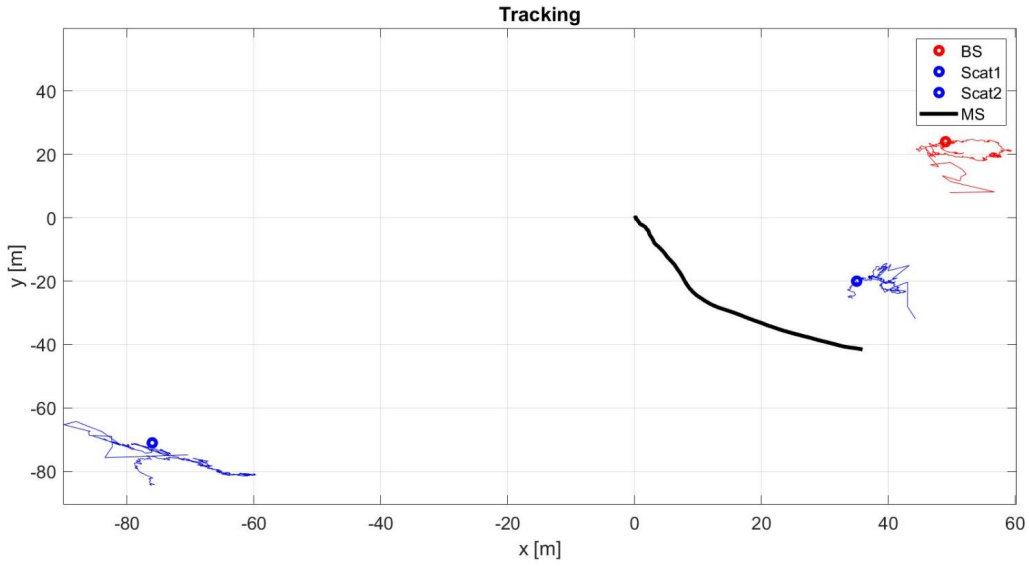

Figure 5.25: Default scenario, second approach. MU random walk with $\boldsymbol{W_u} = 10$ and rotation in the 2D space

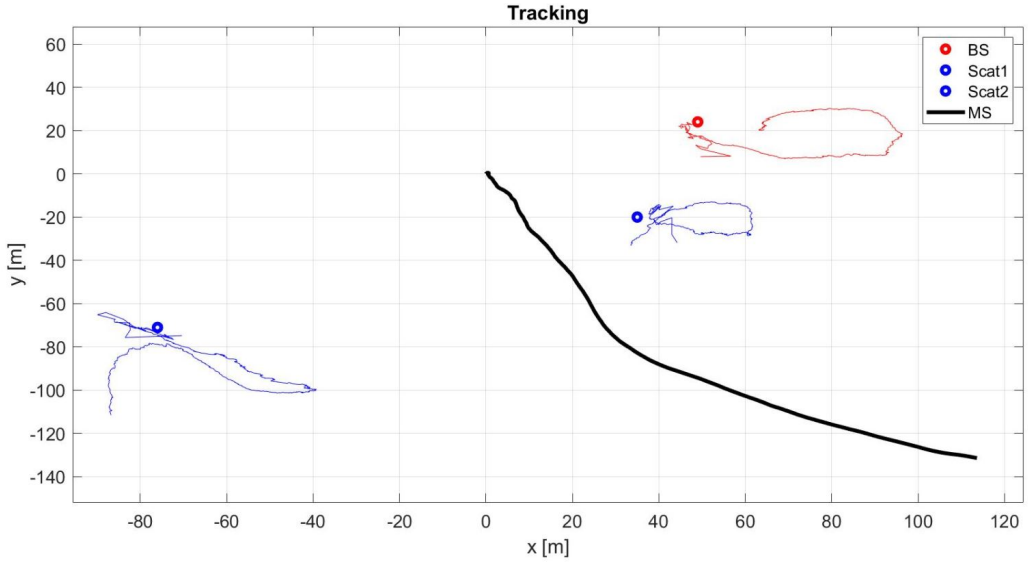

Figure 5.26: Default scenario, second approach. MU random walk with $\boldsymbol{W_u} = 100$ and rotation in the 2D space

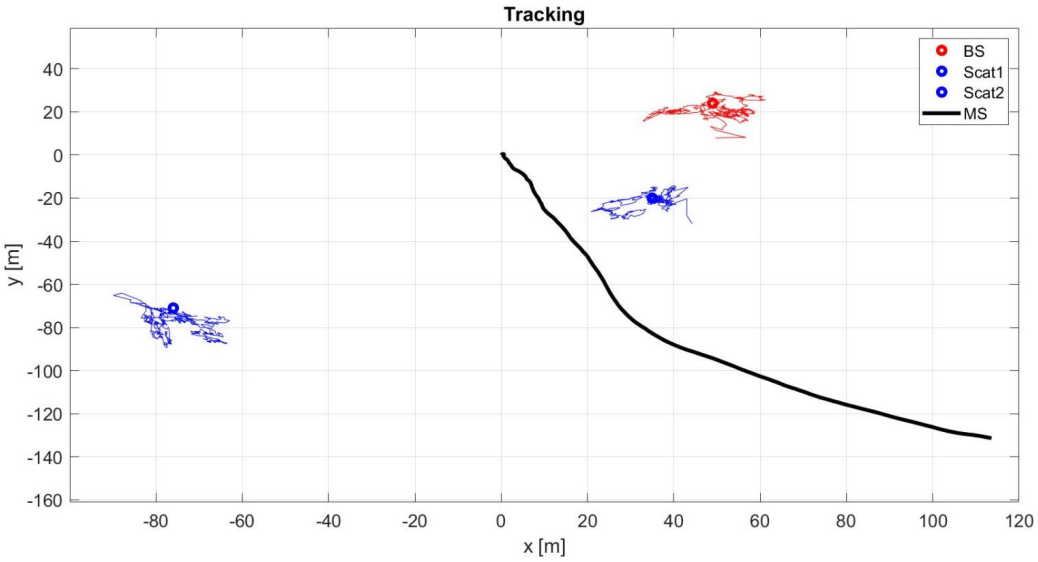

Figure 5.27: Default scenario, second approach. MU random walk with $\boldsymbol{W_u} = 100$ and rotation in the 2D space. Effect of the increase of the process noise spectral density of the mobility model ($\boldsymbol{W_e} = 100$)

Effect of the process noise spectral density of the mobility model

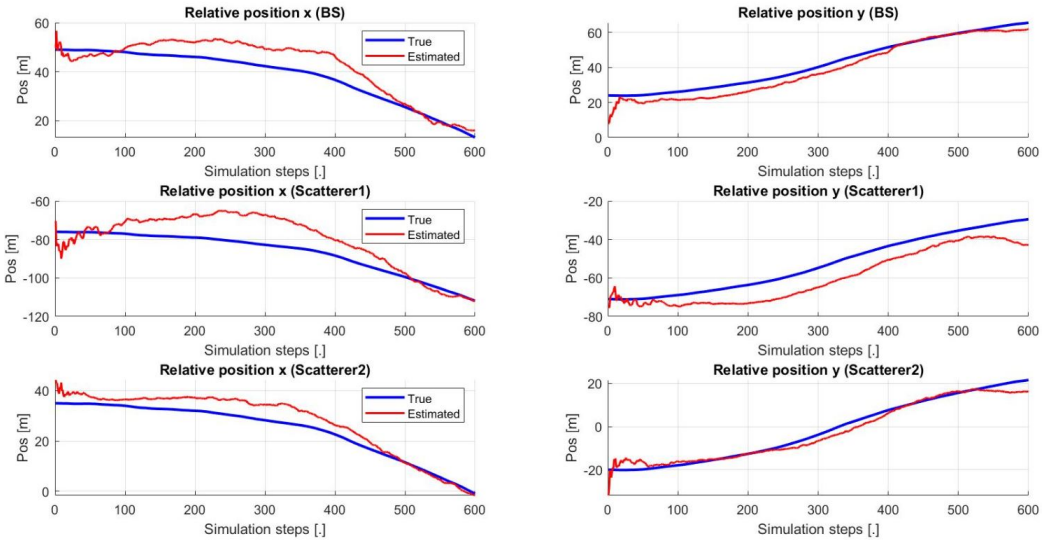

Figure 5.28: Default scenario, second approach. MU random walk with $\boldsymbol{W_u} = 10$ and rotation in the 2D space. Effect of a low $\boldsymbol{W_e}$ on the position estimate

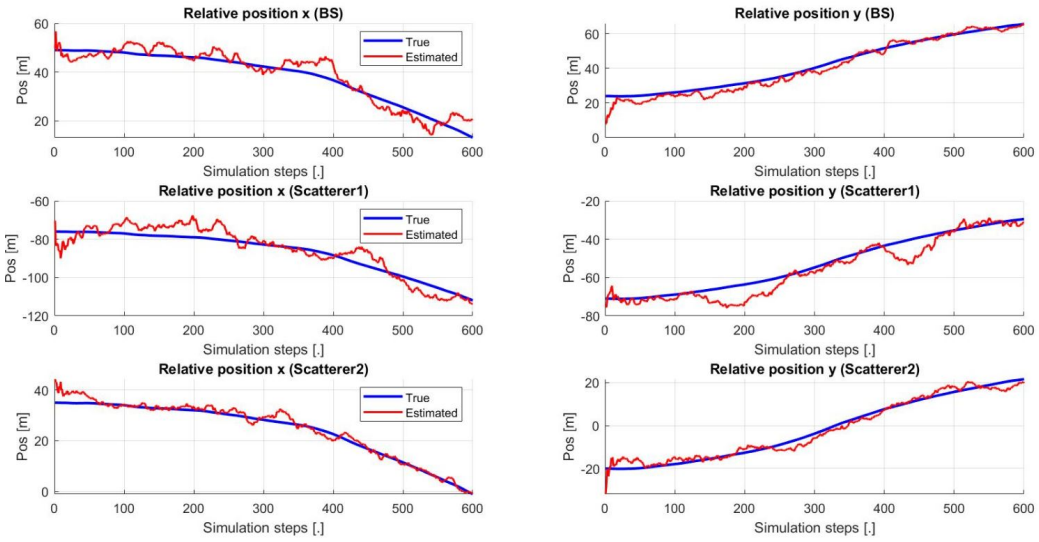

Figure 5.29: Default scenario, second approach. MU random walk with $\boldsymbol{W_u} = 10$ and rotation in the 2D space. Effect of an high $\boldsymbol{W_e}$ on the position estimate

Comparison with state-of-the-art approaches

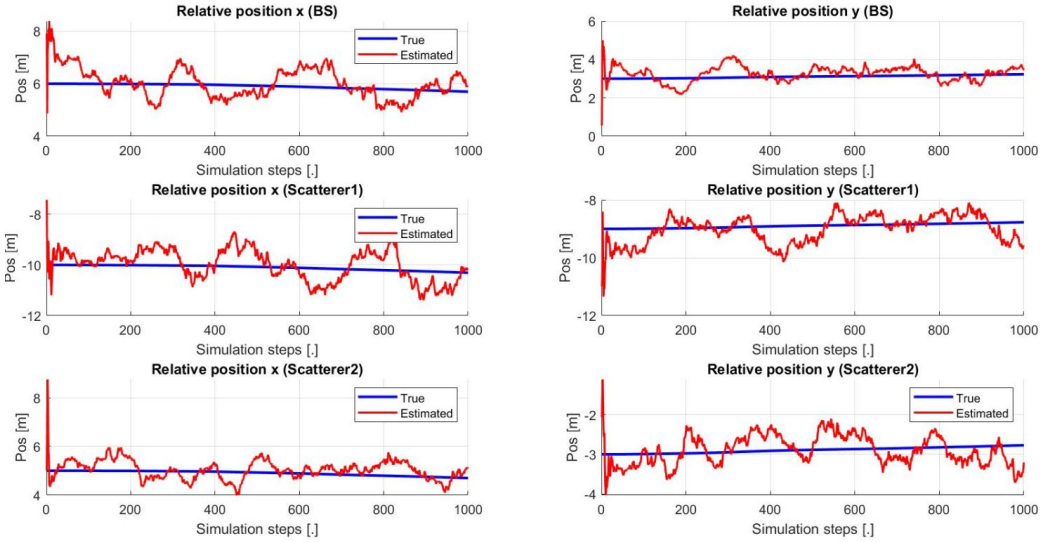

Figure 5.30: Second approach. Positions measurements and estimates in the local reference system using the settings proposed by the reference paper [4]

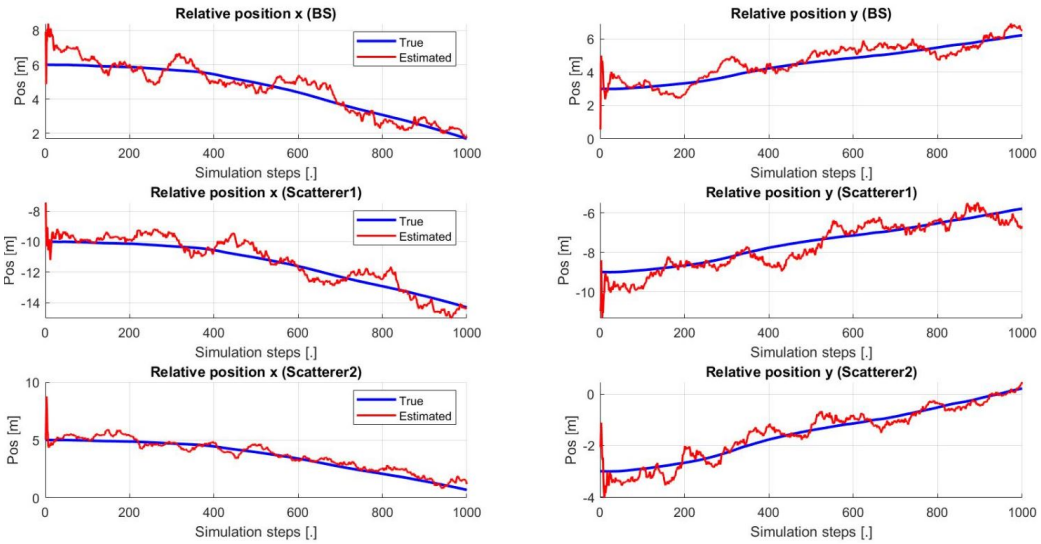

Figure 5.31: Second approach. Positions measurements and estimates in the local reference system using the settings proposed by the reference paper [5]

Appendix 2: Matlab code

First approach

```
% Simulator %

% Purpose: track the position of a mobile user walking with a random
% trajectory in time and 2D-space.
%
% Description: bayesian approach, using Extended Kalman Filter to the state
% estimate (angles) and eventually data fusion to improve the mobility
% model.
% MonteCarlo simulations, with randomness given by the random walk and the
% measurement noise.
%
% Info: Andrea Nicolini & Davide Dardari, February 20, 2019, version 4.0

clear all;
close all;
clc;

% Input parameters
% n_scat = input('Set the number of scatter objects (in addition to one ...
%   BS): ');
% n_meas = input('Set the number of parameters to measure: ');
% df = input('Set 1 to enable data fusion, 0 otherwise: ');
% W0 = input('Set the power spectral density of the process noise (ex. ...
%   0.001): ');
% W0_ms = input('Set the power spectral density of the process noise ...
%   for the movement (ex. 1): ');
% sigma = input('Set the std dev for the angle estimate (ex. pi/8): ');
% sigmag = input('Set the std dev for the gyroscope (ex. 0.1): ');
% self_rot = input('Set the degree per second of device's self rotation ...
%   (ex. 30): ');
% sigmaKal = input('Set the std dev for the Kalman filtering for angle ...
%   (ex. pi/6): ');
% sigmaKal1 = input('Set the std dev for the kalman filtering for rate ...
```

```
%   of change (ex. 0.01): ');
% frac = input('Set the portion of steps for RMSE evaluation (ex. ...
%   9/10): ');
% dt = input('Set the sampling time[s] (ex. 0.01): ');
% area = input('Set the range of the problem[m] (ex. 80): ');
% time = input('Set the simulation time[s] (ex. 6): ');
% threshold = input('Set the threshold value for the RMSE (ex. 10): ');
% M = input('Set the number of MonteCarlo simulations (ex. 10000): ');
n_scat = 2;
n_meas = 3;
df = 1;
W0 = 0.001;
W0_ms = 1;
sigma = pi/8;
sigmag = 0.1;
self_rot_speed_deg = 30;                          % deg/s
self_rot_speed = deg2rad(self_rot_speed_deg);     % rad/s
sigmaKal = pi/6;
sigmaKal1 = 0.01;
frac = 9/10;
dt = 0.01;
area = 80;
time = 6;
sim_steps = time/dt;            % simulation steps
threshold = deg2rad(10);        % 10 deg
M = 10000;
theta_size = 2+(n_scat*2);      % track 2 param (angle & rate) per object

% MonteCarlo simulations %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rmseBS_temp = zeros(1,M);
rmseS1_temp = zeros(1,M);
rmseS2_temp = zeros(1,M);
outlierBS = 0;
outlierS1 = 0;
outlierS2 = 0;
for mc=1:M
    rng(mc,'v5normal');                           % seed
% Scenario %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Initialization
    pos = zeros(1,(n_scat+1)*2);          % true scatterers (&BS) positions
    pos_rel = zeros((n_scat+1)*2,sim_steps);    % relative positions
    x_ms = zeros(4,1);                    % true MS position (center of the
                                          % reference system) and speed
    X_ms = zeros(4,sim_steps);            % true MS position & speed matrix
    rot = zeros(1,sim_steps);                   % device self-rotation
    w_ms = zeros(4,1);                          % correlated Gaussian RVs
    true_meas = zeros(n_meas,sim_steps);        % noise free measurements
    true_meas_temp = zeros(n_meas,sim_steps);   % noise free measurements
                                                % (no self-rotation)
```

```
Z = zeros(n_meas,sim_steps);                % measurements (AoA) vector
u = zeros(theta_size,sim_steps);            % gyroscope matrix

% True scatterers (&BS) positions
for i=1:2:2*(n_scat+1)
    pos(i) = randi([-area area],1,1);
    pos(i+1) = randi([-area area],1,1);
end

% Mobility model & Covariance matrix of the MS
PHI_ms = [1 0 dt 0; 0 1 0 dt; 0 0 1 0; 0 0 0 1];
Q_ms = W0_ms*[dt^3/3 0 dt^2/2 0; 0 dt^3/3 0 dt^2/2; dt^2/2 0 dt 0; ...
    0 dt^2/2 0 dt];

% Random walk, self rotation, relative positions & noise-free
% measurement acquisition
for n=1:sim_steps
    % Random walk
    w_ms = chol(Q_ms)'*randn(size(PHI_ms,1),1);
    x_ms = PHI_ms*x_ms+w_ms;
    X_ms(:,n) = x_ms;
    % Self rotation
    if n>1
        rot(n) = rot(n-1)+self_rot_speed*dt;
    else
        rot(n) = self_rot_speed*dt;
    end
    % Relative positions & noise-free measurement acquisition
    for j=1:n_meas
        pos_rel(2*j-1,n) = pos(2*j-1)-X_ms(1,n);  % relative position x
        pos_rel(2*j,n) = pos(2*j)-X_ms(2,n);      % relative position y
        % Angle measurements
        true_meas_temp(j,n) = atan2(pos_rel(2*j,n),pos_rel(2*j-1,n))';
    end
end
true_meas_temp = unwrap(true_meas_temp')';  % phase correction
% Noise-free measurement acquisition
for n=1:sim_steps
    for j=1:n_meas
        true_meas(j,n) = true_meas_temp(j,n)-rot(n);
    end
end
true_meas = unwrap(true_meas')';            % phase correction

% Noisy Measurement acquisition (angle) & Data fusion
for n=1:sim_steps
    for j=1:n_meas
        Z(j,n) = true_meas(j,n)+sigma*randn;
    end
```

```
        % Data fusion
        if n>1 && df>0
            for j=1:theta_size/2
                % Gyroscope
                u(j,n) = (-self_rot_speed+sigmag*randn)*dt;  % rad every dt
            end
        end
    end

% Mobility model %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    F = [zeros(theta_size/2), eye(theta_size/2); ...
        zeros(theta_size/2,theta_size)];
    PHI = eye(theta_size)+dt*F;
    % Gyroscope model
    G = eye(theta_size);
    B = G;
    % Process noise
    L = [zeros(theta_size/2,theta_size); eye(theta_size/2), ...
        zeros(theta_size/2)];
    Q = Q_eval(F,L,dt,W0);

% Measurement model %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    H = [eye(n_meas), zeros(n_meas)];
    % Measurement noise
    R = sigma^2*eye(n_meas);    % covariance matrix

% Extended Kalman Filter %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Initialization
    phi = zeros(n_meas,sim_steps);           % mapping to polar coordinates
    v = zeros(n_meas,sim_steps);                    % innovation in EKF
    a = zeros(1,n_meas);                            % angle+noise prior
    angle_est_error = zeros(n_meas,sim_steps);      % estimate error
                                                    % with tracking
    angle_est_error_raw = zeros(n_meas,sim_steps);  % raw estimate error

    % Priors -> priors position is known with uncertainty
    % (first measurement -> first estimate)
    for j=1:n_meas
        a(j) = true_meas(j,1)+sigma*rand;
    end
    m = [a zeros(1,theta_size/2)]';
    P = [sigmaKal^2*eye(theta_size/2), zeros(theta_size/2);...
        zeros(theta_size/2), sigmaKal1^2*eye(theta_size/2,theta_size/2)];

    % Algorithm iteration
    for n=1:sim_steps
        % Prediction
        m = PHI*m;                      % state prediction
        if (df==1), m=m+B*u(:,n); end % state prediction (with data fusion)
```

```
        P = PHI*P*(PHI')+Q;            % state prediction uncertainty
        % Update
        v(:,n) = Z(:,n)-H*m;             % innovation
        for j=1:n_meas
            if v(j,n)<-pi, v(j,n) = v(j,n)+2*pi; end    % adjust phase
            if v(j,n)>pi, v(j,n) = v(j,n)-2*pi; end     % adjust phase
        end
        % Equivalent but more stable version
        PHt = (H*P)';                % Matlab is column-major, so (H*PX)'
                                     % is more efficient than PX*H'
        S = H*PHt+R;
        S = make_symmetric(S);       % ensure that the matrix is symmetric
                                     % (see the bottom of the script)
        SChol = chol(S);
        SCholInv = inv(SChol);       % triangular matrix
        K1 = PHt*SCholInv;
        K = K1*SCholInv';
        m = m+K*v(:,n);          % a-posteriori state estimate
        P = P-K1*K1';            % a-posteriori state estimate uncertainty
        % Tracking error of the AoA
        for j=1:n_meas
            angle_est_error(j,n) = true_meas(j,n)-m(j);       % with EKF
            angle_est_error_raw(j,n) = true_meas(j,n)-Z(j,n); % without EKF
        end
    end

    % RMSE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        rmseBS_temp(mc) = sqrt(mean(sum(angle_est_error...
            (1,sim_steps*frac:sim_steps).^2,1)));
        if rmseBS_temp(mc)>threshold
            outlierBS = outlierBS+1;
            rmseBS_temp(mc) = 0;
        end
        rmseS1_temp(mc) = sqrt(mean(sum(angle_est_error...
            (2,sim_steps*frac:sim_steps).^2,1)));
        if rmseS1_temp(mc)>threshold
            outlierS1 = outlierS1+1;
            rmseS1_temp(mc) = 0;
        end
        rmseS2_temp(mc) = sqrt(mean(sum(angle_est_error...
            (3,sim_steps*frac:sim_steps).^2,1)));
        if rmseS2_temp(mc)>threshold
            outlierS2 = outlierS2+1;
            rmseS2_temp(mc) = 0;
        end
end
rmseBS = rad2deg((sum(rmseBS_temp))/(M-outlierBS))
outlierBS_rate = outlierBS/M
rmseS1 = rad2deg((sum(rmseS1_temp))/(M-outlierS1))
```

```
outlierS1_rate = outlierS1/M
rmseS2 = rad2deg((sum(rmseS2_temp))/(M-outlierS2))
outlierS2_rate = outlierS2/M

% Functions %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % To ensure that a matrix is symmetric
    function P = make_symmetric(P)
        P = (P+P')*0.5;
    end
        % State noise covariance matrix
    function Q = Q_eval(F,L,dt,W0)
        Q = W0*(L*L'*dt+1/2*dt^2*(F*L*L'+L*L'*F')+1/3*dt^3*F*L*L'*F');
  % Q = W0*[eye(theta_size/2)*(1/3*dt^3), eye(theta_size/2)*(1/2*dt^2); ...
  %     eye(theta_size/2)*(1/2*dt^2), eye(theta_size/2)*dt];
    end

% End
```

Second approach

```
% Simulator %

% Purpose: track the position of a mobile user walking with a random
% trajectory in time and 2D-space.
%
% Description: bayesian approach, using Extended Kalman Filter to the state
% estimate (positions) and eventually data fusion to improve the mobility
% model.
% MonteCarlo simulations, with randomness given by the random walk and the
% measurement noise.
%
% Info: Andrea Nicolini & Davide Dardari, February 20, 2019, version 6.0

clear all;
close all;
clc;

% Input parameters
% n_scat = input('Set the number of scatter objects (in addition to ...
%   one BS): ');
% n_meas = input('Set the number of parameters to measure: ');
% df = input('Set 1 to enable data fusion, 0 otherwise: ');
% ranging = input('Set 1 if ranging (distance) measurements are ...
%   available, 0 otherwise: ');
% W0 = input('Set the power spectral density of the process noise ...
%   (ex. 1): ');
```

```
% W0_ms = input('Set the power spectral density of the process noise ...
%   for the movement (ex. 1): ');
% sigma = input('Set the std dev for the angle estimate (ex. pi/8): ');
% sigma1 = input('Set the std dev for the distance estimate (ex. 10): ');
% sigmag = input('Set the std dev for the gyroscope (ex. 0.1): ');
% sigmaa = input('Set the std dev for the accelerometer (ex. 10): ');
% self_rot_speed_deg = input('Set the degree per seconds of device's ...
%   self rotation (ex. 30): ');
% sigmaKal = input('Set the std dev for the Kalman filtering for ...
%   distance (ex. 20): ');
% sigmaKal1 = input('Set the std dev for the kalman filtering for ...
%   speed (ex. 0.01): ');
% frac = input('Set the portion of steps for RMSE evaluation ...
%   (ex. 9/10): ');
% dt = input('Set the sampling time[s] (ex. 0.01): ');
% area = input('Set the range of the problem[m] (ex. 80): ');
% time = input('Set the simulation time[s] (ex. 6): ');
% threshold = input('Set the threshold value for the angle RMSE ...
%   (ex. 10): ');
% thresholdPos = input('Set the threshold value for the position RMSE ...
%   (ex. 5): ');
% M = input('Set the number of MonteCarlo simulations (ex. 10000): ');
n_scat = 2;
n_meas = 6;
df = 1;
ranging = 1;
W0 = 1;
W0_ms = 1;
sigma = pi/8;
sigma1 = 10;
sigmag = 0.1;
sigmaa = 10;
self_rot_speed_deg = 30;                          % deg/s
self_rot_speed = deg2rad(self_rot_speed_deg);   % rad/s
sigmaKal = 20;
sigmaKal1 = 0.01;
frac = 9/10;
dt = 0.01;
area = 80;
time = 6;
sim_steps = time/dt;                % simulation steps
threshold = deg2rad(10);            % 10 deg
thresholdPos = 5;                   % 5 meters
M = 10000;
theta_size = 4+(n_scat*4);    % track 4 param (position & speed) per object

% MonteCarlo simulations %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rmseBS_temp = zeros(1,M);
rmseS1_temp = zeros(1,M);
```

```
rmseS2_temp = zeros(1,M);
rmsePosBS_temp = zeros(1,M);
rmsePosS1_temp = zeros(1,M);
rmsePosS2_temp = zeros(1,M);
outlierBS = 0;
outlierS1 = 0;
outlierS2 = 0;
outlierPosBS = 0;
outlierPosS1 = 0;
outlierPosS2 = 0;
for mc=1:M
    rng(mc,'v5normal');                                % seed
% Scenario %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Initialization
    pos = zeros(1,(n_scat+1)*2);          % true scatterers (&BS) positions
    pos_rel = zeros((n_scat+1)*2,sim_steps);    % relative positions
    x_ms = zeros(4,1);          % true MS position (center of the reference
                                % system) and speed
    X_ms = zeros(4,sim_steps);            % true MS position & speed matrix
    rot = zeros(1,sim_steps);             % device self-rotation (rad)
    rot_deg = zeros(1,sim_steps);         % device self-rotation (deg)
    x_self = zeros(1,2,sim_steps);              % self rotation graphic x
    y_self = zeros(1,2,sim_steps);              % self rotation graphic y
    w_ms = zeros(4,1);                          % correlated Gaussian RVs
    true_meas = zeros(n_meas,sim_steps);        % noise free measurements
    true_meas_temp = zeros(n_meas,sim_steps);   % noise free measurements
                                                % (no self-rotation)
    Z = zeros(n_meas,sim_steps);    % measurements (AoA & distance) vector
    Z_gyro = zeros(1,sim_steps);    % gyroscope measurements (rad every dt)
    x_gyro = zeros(n_scat+1,sim_steps);         % gyroscope x displacement
    y_gyro = zeros(n_scat+1,sim_steps);         % gyroscope y displacement
    u = zeros(theta_size,sim_steps);       % gyroscope+accelerometer matrix

    % True scatterers (&BS) positions
    for i=1:2:2*(n_scat+1)
        pos(i) = randi([-area area],1,1);
        pos(i+1) = randi([-area area],1,1);
    end

    % Mobility model & Covariance matrix of the MS
    PHI_ms = [1 0 dt 0; 0 1 0 dt; 0 0 1 0; 0 0 0 1];
    Q_ms = W0_ms*[dt^3/3 0 dt^2/2 0; 0 dt^3/3 0 dt^2/2; dt^2/2 0 dt 0; ...
        0 dt^2/2 0 dt];

    % Random walk, self rotation, relative positions & noise-free
    % measurement acquisition
    for n=1:sim_steps
        % Random walk
        w_ms = chol(Q_ms)'*randn(size(PHI_ms,1),1);
```

```
    x_ms = PHI_ms*x_ms+w_ms;
    X_ms(:,n) = x_ms;
    % Self rotation
    if n>1
        rot(n) = rot(n-1)+self_rot_speed*dt;
    else
        rot(n) = self_rot_speed*dt;
    end
    % rot_deg(n) = rad2deg(rot(n));
    % Relative positions & noise-free measurement acquisition
    for j=1:n_meas/2
        pos_rel(2*j-1,n) = pos(2*j-1)-X_ms(1,n);  % relative position x
        pos_rel(2*j,n) = pos(2*j)-X_ms(2,n);      % relative position y
        % Angle measurements (first part)
        true_meas_temp(j,n) = atan2(pos_rel(2*j,n),pos_rel(2*j-1,n))';
        % Ranging/distance measurements (second part)
        true_meas(j+n_meas/2,n) = sqrt(pos_rel(2*j,n)^2+...
            pos_rel(2*j-1,n)^2)';
    end
end
true_meas_temp = unwrap(true_meas_temp')';  % phase correction
% Noise-free measurement acquisition
for n=1:sim_steps
    for j=1:n_meas/2
        true_meas(j,n) = true_meas_temp(j,n)-rot(n);
    end
end
true_meas = unwrap(true_meas')';            % phase correction
% true_meas_deg = rad2deg(true_meas);

% Noisy Measurement acquisition (angle & distance) & Data fusion
for n=1:sim_steps
    % Measurements
    Z_gyro(n) = (self_rot_speed+sigmag*randn)*dt;  % gyro: rad every dt
    for j=1:n_meas/2
        Z(j,n) = true_meas(j,n)+sigma*randn;       % AoA
        Z(j+n_meas/2,n) = true_meas(j+n_meas/2,n)+...
            sigma1*randn;                          % ranging
        % Gyroscope (x and y variations due to reference system rot)
        if n>1 && df>0
            x_gyro(j,n) = true_meas(j+n_meas/2,n-1)*(cos...
                (true_meas(j,n-1)-Z_gyro(n))-(cos(true_meas(j,n-1)))...
                *(cos(Z_gyro(n))));
            y_gyro(j,n) = true_meas(j+n_meas/2,n-1)*(sin...
                (true_meas(j,n-1)-Z_gyro(n))-(sin(true_meas(j,n-1)))...
                *(cos(Z_gyro(n))));
        end
    end
    % Data fusion
```

```
        if n>1 && df>0
            for j=1:2:theta_size/2-1
                % Gyroscope
                u(j,n) = x_gyro((j+1)/2,n);             % every dt
                u(j+1,n) = y_gyro((j+1)/2,n);           % every dt
                % Accelerometer
                u(theta_size/2+j,n) = (X_ms(3,n)-X_ms(3,n-1))...
                    +sigmaa*randn*dt;       % every dt
                u(theta_size/2+j+1,n) = (X_ms(4,n)-X_ms(4,n-1))...
                    +sigmaa*randn*dt;       % every dt
            end
        end
    end

% Mobility model %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    F =[zeros(theta_size/2), eye(theta_size/2); ...
        zeros(theta_size/2,theta_size)];
    PHI = eye(theta_size)+dt*F;
    % Accelerometer & gyroscope model
    G = eye(theta_size);
    B = G;
    % Process noise
    L = [zeros(theta_size/2,theta_size); eye(theta_size/2), ...
        zeros(theta_size/2)];
    Q = Q_eval(F,L,dt,W0);

% Measurement model %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % NL model -> h: phi_i = arctg(x_i/y_i) i=1,..n_meas
    H = zeros(n_meas,theta_size);   % linearised model (Jacobian matrix)
    % Measurement noise
    R = [sigma^2*eye(n_meas/2), zeros(n_meas/2); zeros(n_meas/2), ...
        sigma1^2*eye(n_meas/2)];    % covariance matrix

% Extended Kalman Filter %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Initialization
    phi = zeros(n_meas,sim_steps);           % mapping to polar coordinates
    pos_est = zeros(1,n_meas);      % first measurement -> first estimate
    v = zeros(n_meas,sim_steps);                        % innovation in EKF
    mv = zeros(theta_size/2,sim_steps);           % state estimate vector
                                                  % (relative coords)
    mv_real = zeros(theta_size/2,sim_steps);      % state estimate vector
                                                  % (absolute coords)
    m_angle = zeros(n_meas/2,sim_steps);            % AoA estimate vector
    angle_est_error = zeros(n_meas/2,sim_steps);    % estimate error with
                                                    % tracking (ang)
    angle_est_error_raw = zeros(n_meas/2,sim_steps); % raw estimate error
    pos_est_error = zeros(n_meas/2,sim_steps);      % estimate error with
                                                    % tracking (pos)
    pos_est_error_raw = zeros(n_meas/2,sim_steps);   % raw estimate error
```

```
%   pos_est_error_x = zeros(n_meas/2,sim_steps);    % estimate error with
                                                    % tracking (x)
%   pos_est_error_y = zeros(n_meas/2,sim_steps);    % estimate error with
                                                    % tracking (y)

    % Priors -> priors position is known with uncertainty
    % (first measurement -> first estimate)
    for j=1:n_meas/2
        a = true_meas(j,1)+sigma*rand;          % angle+noise
        r = sigma1*rand;                        % distance noise
        if ranging==1
            r = r+true_meas(j+n_meas/2,1);      % distance+noise
        else
            r = r+area/2;           % no info -> use average distance
        end
        r = max(1,r);
        pos_est(2*j-1) = r*cos(a);          % x initial estimate position
        pos_est(2*j) = r*sin(a);            % y initial estimate position
    end
    m = [pos_est zeros(1,theta_size/2)]';
    P = [sigmaKal^2*eye(theta_size/2), zeros(theta_size/2); ...
        zeros(theta_size/2), sigmaKal1^2*eye(theta_size/2,theta_size/2)];

    % Algorithm iteration
    for n=1:sim_steps
        % Prediction
        m = PHI*m;                      % state prediction
        if (df==1), m=m+B*u(:,n); end % state prediction (with data fusion)
        P = PHI*P*(PHI')+Q;             % state prediction uncertainty
        % Update
        for j=1:n_meas/2
            % Angle measurements
            phi(j,n) = atan2(m(2*j),m(2*j-1));
            H(j,2*j-1) = -m(2*j)/(m(2*j)^2+m(2*j-1)^2);
            H(j,2*j) = m(2*j-1)/(m(2*j)^2+m(2*j-1)^2);
            % Ranging measurements
            if ranging==1
                phi(j+n_meas/2,n) = sqrt(m(2*j)^2+m(2*j-1)^2);
                H(j+n_meas/2,2*(j)-1) = m(2*j-1)/sqrt(m(2*j)^2+m(2*j-1)^2);
                H(j+n_meas/2,2*(j)) = m(2*j)/sqrt(m(2*j)^2+m(2*j-1)^2);
            end
        end
        phi(1:n_meas/2,:) = unwrap(phi(1:n_meas/2,:)')'; % phase correction
        v(:,n) = Z(:,n)-phi(:,n);                        % innovation
        % Equivalent but more stable version
        PHt = (H*P)';           % Matlab is column-major, so (H*PX)'
                                % is more efficient than PX*H'
        S = H*PHt+R;
        S = make_symmetric(S);  % ensure that the matrix is symmetric
```

```
                                % (see the bottom of the script)
        SChol = chol(S);
        SCholInv = inv(SChol);  % triangular matrix
        K1 = PHt*SCholInv;
        K = K1*SCholInv';
        m = m+K*v(:,n);         % a-posteriori state estimate
        P = P-K1*K1';           % a-posteriori state estimate uncertainty
        % Consistency check: speed and range limits
%          for j=1:theta_size/2
%             if (abs(m(j))>100), m(j) = 100*m(j)/abs(m(j)); ...
%                 end % maximum range
%             if (abs(m(j+theta_size/2))>1)
%                 m(j+theta_size/2) = 1*m(j+theta_size/2)/abs...
%                   (m(j+theta_size/2)); % maximum speed
%             end
%          end
        % Vectors update
        for j=1:2:length(pos)
            m_angle((j+1)/2,n) = atan2(m(j+1),m(j));    % AoA
        end
        m_angle = unwrap(m_angle')';        % phase correction
        for j=1:2:length(pos)
            mv(j,n) = (m(j)/cos(m_angle(((j+1)/2),n)))...
                *cos(m_angle(((j+1)/2),n)+rot(n));    % relative position x
            mv(j+1,n) = (m(j)/cos(m_angle(((j+1)/2),n)))...
                *sin(m_angle(((j+1)/2),n)+rot(n));    % relative position y
        end
        for j=1:2:length(pos)
            mv_real(j,n) = mv(j,n)+X_ms(1,n);        % position x
            mv_real(j+1,n) = mv(j+1,n)+X_ms(2,n);    % position y
        end
        % Tracking error of the AoA & position
        for j=1:n_meas/2
            angle_est_error(j,n) = true_meas(j,n)-m_angle(j,n); % with EKF
            angle_est_error_raw(j,n) = true_meas(j,n)-Z(j,n); % without EKF
            pos_est_error_raw(j,n) = true_meas(n_meas/2+j,n)-...
                Z(n_meas/2+j,n);                          % pos without EKF
            pos_est_error(j,n) = true_meas(n_meas/2+j,n)-...
                sqrt(mv(2*j-1,n)^2+mv(2*j,n)^2);             % pos with EKF
%          pos_est_error_x(j,n) = pos_rel(2*j-1,n)-mv(2*j-1,n); % x with EKF
%          pos_est_error_y(j,n) = pos_rel(2*j,n)-mv(2*j,n);     % y with EKF
        end
    end

    % RMSE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % AoA
        rmseBS_temp(mc) = sqrt(mean(sum(angle_est_error...
            (1,sim_steps*frac:sim_steps).^2,1)));
        if rmseBS_temp(mc)>threshold
```

```
            outlierBS = outlierBS+1;
            rmseBS_temp(mc) = 0;
        end
        rmseS1_temp(mc) = sqrt(mean(sum(angle_est_error...
            (2,sim_steps*frac:sim_steps).^2,1)));
        if rmseS1_temp(mc)>threshold
            outlierS1 = outlierS1+1;
            rmseS1_temp(mc) = 0;
        end
        rmseS2_temp(mc) = sqrt(mean(sum(angle_est_error...
            (3,sim_steps*frac:sim_steps).^2,1)));
        if rmseS2_temp(mc)>threshold
            outlierS2 = outlierS2+1;
            rmseS2_temp(mc) = 0;
        end
        % Position
        rmsePosBS_temp(mc) = sqrt(mean(sum(pos_est_error...
            (1,sim_steps*frac:sim_steps).^2,1)));
        if rmsePosBS_temp(mc)>thresholdPos
            outlierPosBS = outlierPosBS+1;
            rmsePosBS_temp(mc) = 0;
        end
        rmsePosS1_temp(mc) = sqrt(mean(sum(pos_est_error...
            (2,sim_steps*frac:sim_steps).^2,1)));
        if rmsePosS1_temp(mc)>thresholdPos
            outlierPosS1 = outlierPosS1+1;
            rmsePosS1_temp(mc) = 0;
        end
        rmsePosS2_temp(mc) = sqrt(mean(sum(pos_est_error...
            (3,sim_steps*frac:sim_steps).^2,1)));
        if rmsePosS2_temp(mc)>thresholdPos
            outlierPosS2 = outlierPosS2+1;
            rmsePosS2_temp(mc) = 0;
        end
end
rmseBS = rad2deg((sum(rmseBS_temp))/(M-outlierBS))
outlierBS_rate = outlierBS/M
rmseS1 = rad2deg((sum(rmseS1_temp))/(M-outlierS1))
outlierS1_rate = outlierS1/M
rmseS2 = rad2deg((sum(rmseS2_temp))/(M-outlierS2))
outlierS2_rate = outlierS2/M

rmsePosBS = (sum(rmsePosBS_temp))/(M-outlierPosBS)
outlierPosBS_rate = outlierPosBS/M
rmsePosS1 = (sum(rmsePosS1_temp))/(M-outlierPosS1)
outlierPosS1_rate = outlierPosS1/M
rmsePosS2 = (sum(rmsePosS2_temp))/(M-outlierPosS2)
outlierPosS2_rate = outlierPosS2/M
```

```
% Functions %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % To ensure that a matrix is symmetric
    function P = make_symmetric(P)
        P = (P+P')*0.5;
    end
    % State noise covariance matrix
    function Q = Q_eval(F,L,dt,W0)
        Q = W0*(L*L'*dt+1/2*dt^2*(F*L*L'+L*L'*F')+1/3*dt^3*F*L*L'*F');
%   Q = W0*[eye(theta_size/2)*(1/3*dt^3), eye(theta_size/2)*(1/2*dt^2); ...
%      eye(theta_size/2)*(1/2*dt^2), eye(theta_size/2)*dt];
    end

% End
```

Bibliography

[1] C. Jeong, J. Park, and H. Yu, "Random access in millimeter-wave beamforming cellular networks: issues and approaches," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 180–185, January 2015.

[2] J. Salmi, A. Richter, and V. Koivunen, "Detection and tracking of mimo propagation path parameters using state-space approach," *IEEE Transactions on Signal Processing*, vol. 57, no. 4, pp. 1538–1550, April 2009.

[3] J. Bao, D. Sun, and H. Li, "Motion sensor aided beam tracking in mobile devices of millimeter-wave communications," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–7.

[4] Z. Qi and W. Liu, "Three-dimensional millimetre wave beam tracking based on handset mems sensors with extended kalman filtering," in *Radio Propagation and Technologies for 5G (2016)*, Oct 2016, pp. 1–6.

[5] ——, "Three-dimensional millimetre-wave beam tracking based on smart phone sensor measurements and direction of arrival/time of arrival estimation for 5g networks," *IET Microwaves, Antennas Propagation*, vol. 12, no. 3, pp. 271–279, 2018.

[6] M. Giordani, M. Mezzavilla, and M. Zorzi, "Initial access in 5g mmwave cellular networks," *IEEE Communications Magazine*, vol. 54, no. 11, pp. 40–47, November 2016.

[7] J. G. Andrews, T. Bai, M. N. Kulkarni, A. Alkhateeb, A. K. Gupta, and R. W. Heath, "Modeling and analyzing millimeter wave cellular systems," *IEEE Transactions on Communications*, vol. 65, no. 1, pp. 403–430, Jan 2017.

[8] X. Yin, G. Steinbock, G. E. Kirkelund, T. Pedersen, P. Blattnig, A. Jaquier, and B. H. Fleury, "Tracking of time-variant radio propagation paths using particle filtering," in *2008 IEEE International Conference on Communications*, May 2008, pp. 920–924.

[9] D. Shim, C. Yang, J. Kim, J. Han, and Y. Cho, "Application of motion sensors for beam-tracking of mobile stations in mmwave communication systems," pp. 19 622–19 638, 2014.

[10] A. Yassin, "Context inference, localization and mapping in indoor environments using mmwaves," November 2018.

Acknowledgments

This work represents, for me, the end of a journey started five years ago. It has been very hard and in several moments it seemed impossible to go on, but thanks to many people I managed to keep going. For this reason I want to thank:

- My supervisors and professors, cause they taught me a lot in these years. In particular Prof. Davide Dardari, who followed me in my thesis work and gave me lot of hints and teachings.
- All of my university mates, with whom I shared projects, exams, hardships and laughs.
- My best friends, which helped me when I was down and celebrated me in the best moments.
- All those who accompanied me for all the journey or only for a part of it, all of the old friendships and the new people I met, that taught me the true essence of life.
- My family all, my brother and my parents, that always trust me more than I do, always permit me to do whatever I want and always want my happiness. I love you.

Thank you all again.