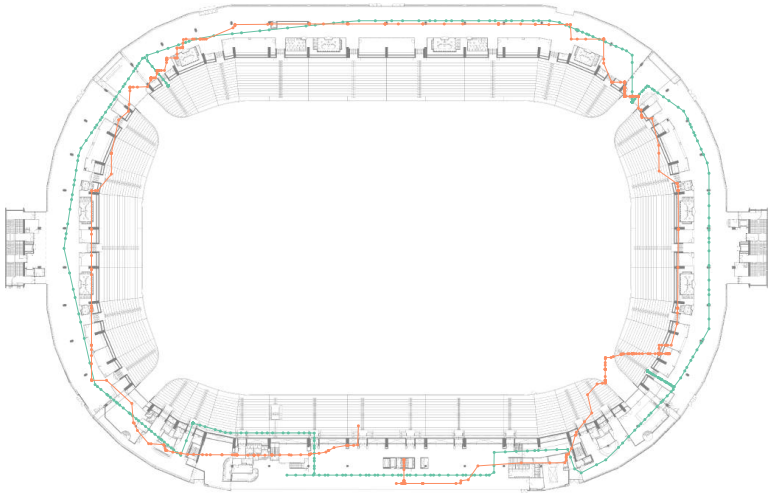
Computer Science — University of Amsterdam

Bachelor Computer Science

University of Amsterdam



A Performance Analysis of Filtering Methods applied to WiFi-based Position Reconstruction

Nima Motamed

August 24, 2018

Supervisor: dr. M. H. Lees

Abstract

WiFi-based position reconstruction is becoming an increasingly important field of study, with applications in both ubiquitous computing and crowd analysis. Many approaches to this problem provide high accuracy at the cost of low ease of use, by often requiring either specialised hardware of costly offline phases. We compare and analyze the use of several filtering methods in order to increase the accuracy of a low-cost and out-of-the-box trilateration approach to WiFi-based position reconstruction, by applying the filtering methods to positions reconstructed from walks through the largest stadium in the Netherlands. The compared filtering methods consist of exponential filtering, Gaussian filters, Savitzky-Golay filters, median filters and Kalman filters, along with several variations. In addition to the performance comparison, we apply sensitivity analysis to the parameters of the methods, determining which are of most importance to resulting accuracy. We find that the median filter and some of its variations perform the best, improving upon the baseline accuracy by seven metres. Other high performers include the Gaussian and Savitzky-Golay filters, which like the median filter, are simple in their procedures and require few parameters.

Acknowledgement

I would like to thank Michael Lees and Philip Rutten for their direct supervision during the writing of this thesis. Philip was always available to give advice about the more technical parts of the project, and Mike guided the thesis as a whole in the right direction.

For being so gracious and providing such an interesting experimental setting, I thank the people at the Amsterdam ArenA and KPMG.

Being able to finish this project would have been impossible from the start, were it not for Robert Belleman and Sander van Splunter's work in making sure I could hand in this thesis later than the original deadline. I owe a lot to your help.

Special thanks go out to Rick Quax, who sparred with me on the topic of statistical analysis, Jan Schutte, who aided with the making of visualizations of paths and proofreading, and Sanne Bouwmeester, Laura Doan, and Anne van Oers, who were also involved with proofreading. I am greatly indebted to all these proofreaders, who pointed out mistakes and provided ideas I would never have come across on my own.

Above all, I express sincere and deep gratitude to my loved ones, and especially my mother. Finishing this thesis took much more time than expected due to the accumulation of several obstacles and problems, and often I wondered whether it would ever work out. After all these months, I now have the pleasure of presenting this thesis to her. I know you were worried, but here we finally are. This is dedicated to you.

Contents

CHAPTER 1

Introduction

Position reconstruction has become an increasingly important field of study for the purposes of both ubiquitous computing and crowd analysis. For the former, it can aid context-aware computing by making applications provide services based on the location of the user, contributing to an improved user experience. The latter requires reliable real world data of movements within crowds in order to build and validate models, contributing to security and efficiency.

Such position reconstruction is currently often performed using the Global Positioning System (GPS), being available on virtually all modern mobile devices. As its functionality relies on satellites, it has pervasive coverage and a high accuracy of up to $5 - 10$ meters in outdoor environments [1]. Nonetheless, GPS comes with significant drawbacks. It carries a relatively high energy cost, requiring mobile devices to either increase in size, or risk early draining of the battery [2]. Additionally, GPS is largely reliant on free line-of-sight transmission between the device and several satellites, making it unsuited for indoor position reconstruction, which is a vital part of both ubiquitous computing and crowd analysis.

Due to their increased proliferation and cost efficiency, wireless networks such as WiFi are a popular alternative to GPS for indoor position reconstruction. There is a multitude of different techniques for position reconstruction that can be applied to WiFi systems [3], which can generally be divided into techniques that model the propagation of the signals, and techniques that attempt to 'learn' how to associate signals to positions from training data [4].

The latter of these techniques, often referred to as fingerprinting, performs well but cannot be applied out-of-the-box. It requires additional hardware and a preparatory phase in order to build a training data set that is capable of generalizing the signal strengths to positions, which might be prohibitive cost-wise [5]–[7]. In contrast, model-based methods can function in an existing wireless network with little to no modifications and no need for training data [3].

Model-based methods modelling the distances between access points and devices are in common use due to their ease of use and low cost, but often suffer from lower accuracy than other methods [8]. Van Engelen, Van Lier, Takes, *et al.* [4], Laviola [9], and Lee, Oka, Pollakis, *et al.* [10] attempt to mitigate this by applying noise-reducing filtering methods to the positions produced by the distance-based method. Such methods increase accuracy whilst maintaining the ease of use and low cost. This thesis aims to expand upon the methods they used, and more thoroughly compare and analyze the use of filtering methods in order to increase the accuracy of low-cost WiFi and distance-based position reconstruction.

In order to determine the accuracy of each method, we need to compare position estimates produced from experimentally obtained measurements to a ground truth consisting of the actual positions that correspond to said measurements. While such experiments can easily be performed in a lab setting, we are particularly interested in the performance of the methods in practice. To this end, previous experiments in indoor position reconstruction have usually been performed on floors of common buildings with a relatively small amount of access points [4], [11]–[13]. We perform our experiments in the Johan Cruyff ArenA, the largest stadium in the Netherlands, with over 600 access points, making it one of the largest wireless sensor networks suitable for position reconstruction 'in the wild'.

1.1 Thesis outline

This thesis is structured as follows. Chapter 2 explains the formal process of position reconstruction based on signal strength measurements that is used to obtain position estimates. Chapter 3 gives an overview of the workings of each filtering method as applied to the position estimates obtained earlier. Chapter 4 describes the experiment performed in the ArenA in order to obtain the measurements, along with the ways we analyze the final data. In Chapter 5 we present and discuss the results, and we finally state our conclusions and suggestions for future research in Chapter 6.

CHAPTER 2

Position reconstruction

Position reconstruction using WiFi data relies on frequent (attempted) communication between devices capable of wireless networking (which we will simply refer to as *mobile devices*, regardless of actual size and mobility), and access points (which we will henceforth refer to as *AP's*). While such communication needs to be frequent when the device is actively running applications with networking components, communication still occurs in several ways even when mobile devices are not actively being used.

In order to initiate communications, mobile devices need to be able to find and connect to AP's in their vicinity. The IEEE 802.11 standard for WiFi communications [14] defines two ways in which both mobile devices and AP's can both locate one another. Firstly, AP's broadcast beacon frames (a frame is an unit of WiFi data) containing their networking information, allowing mobile devices to recognize their presence. Secondly, mobile devices can also actively attempt to locate AP's by sending out frames called probe requests, containing their MAC address (an unique identifier for the device) and other information relevant to WiFi communications. On average, mobile devices send out anywhere between 55 and 2000 probe requests in an hour, but this number can be much higher, depending on device build and usage [15].

The following section will describe the format and preprocessing of the measurements of the ordinary active communications and probe requests obtained by the AP's. The second section will introduce the basic model used to describe the transmission of WiFi signals between mobile devices and AP's. The third section will use this model in the context of position reconstruction, using trilateration. Finally, the last section will describe the optimization procedure used to actually find the best positions given the measurements, producing a path.

2.1 AP measurement preprocessing

By having sensors in or by an AP (we will not distinguish between sensors and AP's, and will refer to both by the latter name), we can sniff out the frames sent to it, measuring the Received Signal Strength Indication (RSSI) of the received signal associated with the request. These measurements can be expressed in units of Watts, but for reasons that will become clear in Section 2.2, we will instead use decibels-milliwatts (the signal strength in decibels with reference to one milliwatt).

Now, let N be the amount of AP's we are collecting measurements from. Formally, we denote these AP's by vectors $\mathbf{s}_1, \ldots, \mathbf{s}_N$, with $\mathbf{s}_i \in \mathbb{R}^3$ representing the x, y, and z-coordinates of the i-th AP, expressed in meters w.r.t. some arbitrary origin. Letting k be the amount of measurements some i-th AP has obtained for the mobile device, we associate to that AP a sequence of RSSI measurements $P_i = (P_{i_1}, \ldots, P_{i_k})$, and a sequence of corresponding measurement timestamps (in seconds) $t_i = (t_{i_1}, \ldots, t_{i_k})$.

In order to determine the device's position at a certain time u, we need to collect measurements P_{i_j} such that $t_{i_j} = u$; i.e. all measurements (for any AP) obtained exactly at the specified time. However, as both the frequency with which the device sends probe requests and

the speed at which the AP's generate measurements are generally not known, it is often the case that at most a single AP measured the device at any given time. To solve this, we divide the measurements into groups representing time windows w seconds in length.

These groups are represented by a sequence of timestamps $t' = (t'_1, \ldots, t'_\beta)$, with each t'_i in the middle of its window. Let $\beta = \lceil \frac{g-l}{w} \rceil$ ($\lceil \cdot \rceil$ denotes the ceiling function), $l = \min(t_1 \| \ldots \| t_N)$ and $g = \max(t_1 \| \ldots \| t_N)$ ($\cdot \| \cdot$ concatenates sequences). Using this notation, we can define the sequence of group timestamps t by stating that $t'_i = l + (i - \frac{1}{2})w$. [1] To this sequence of timestamps, we finally associate a sequence M of sets of measurements alongside their AP vectors, with

$$M_i = \{(P_{j_k}, \mathbf{s}_j) \mid j, k \in \mathbb{N}, (t'_i - \frac{w}{2}) \leqslant t_{j_k} < (t'_i + \frac{w}{2})\}. \tag{2.1}$$

2.2 Friis' transmission equation and path loss model

Before moving to fit the mobile device's positions to the measurements grouped in the previous section, we need to first model the transmission of the WiFi signals. Friis [16] proposed the following equation for the radio transmission between two antennas:

$$\frac{P_{\text{receiving}}}{P_{\text{transmitting}}} = \frac{A_{\text{receiving}} A_{\text{transmitting}}}{d^2 \lambda^2}, \tag{2.2}$$

where $P_{\text{receiving}}$ and $P_{\text{transmitting}}$ are respectively the power used by the transmitting antenna and the signal's power at the receiving antenna, $A_{\text{receiving}}$ and $A_{\text{transmitting}}$ are the effective areas of the corresponding antennas, d is the distance between the antennas, and λ is the wavelength of the signal. All quantities are expressed in the corresponding SI units (Watts for power, (square) meters for distance and area).

Comtemporary literature [17]–[19] uses a slightly reworked (but equivalent [20]) version of Equation (2.2) that incorporates the efficiency and directivity (a measure of how focused in a single direction an antenna's signals are) into a value for the so-called gain of the antennas:

$$\frac{P_{\text{receiving}}}{P_{\text{transmitting}}} = G_{\text{receiving}} G_{\text{transmitting}} \left(\frac{\lambda}{4\pi d} \right)^2, \tag{2.3}$$

where $G_{\text{receiving}}$ and $G_{\text{transmitting}}$ represent the values for the gain of the corresponding antennas, in Watts.

Equation (2.3) assumes that the signal is transmitted through free space, with no refraction or other difficulties occurring [16]. This assumption is captured by the exponent in $\left(\frac{\lambda}{4\pi d}\right)^2$. Hata [21] proposed a transmission equation containing a parameter measuring the influence of obstacles, by replacing this exponent by a parameter known as the *path loss exponent*, γ [4], [22]. This exponent is $\gamma = 2$ under free space assumptions. Situations in which the signal suffers from reflection, diffraction, scattering or similar issues call for $\gamma > 2$. It must be noted that there also exist conditions wherein it is possible that $\gamma < 2$ [23].

Now, having added this path loss exponent, we rewrite Equation (2.3) by moving $P_{\text{transmitting}}$ to the right-hand side, and we simplify the equation further by expressing power and gain in dBm (decibel-milliwatts), resulting in

$$P_{\text{receiving}} = P_{\text{transmitting}} + G_{\text{receiving}} + G_{\text{transmitting}} + 10\gamma \log_{10} \left(\frac{\lambda}{4\pi d} \right). \tag{2.4}$$

2.3 Trilateration

As said before, and as visualized by Figure 2.1, Equation (2.4) models the relation between the distance d and the value of the signal's power at both ends of the transmission. By applying the theory of trilateration, we can translate such distances to actual position vectors.

[1] It should be noted that this method does not guarantee all the groups are equal in length, but we for our purposes it is deemed sufficient.

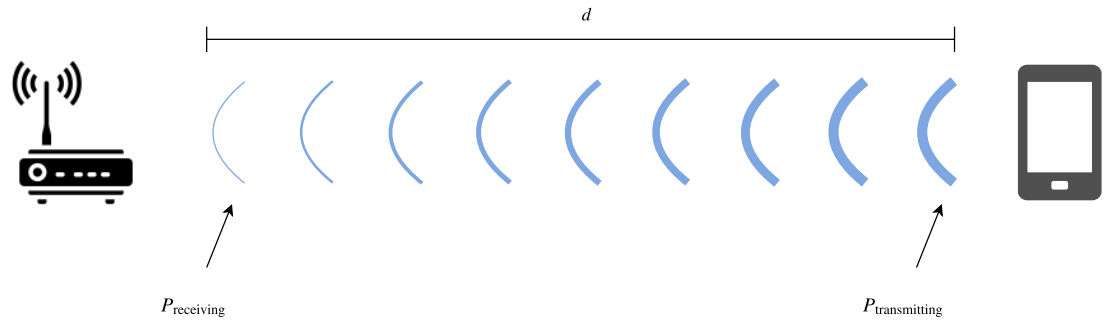
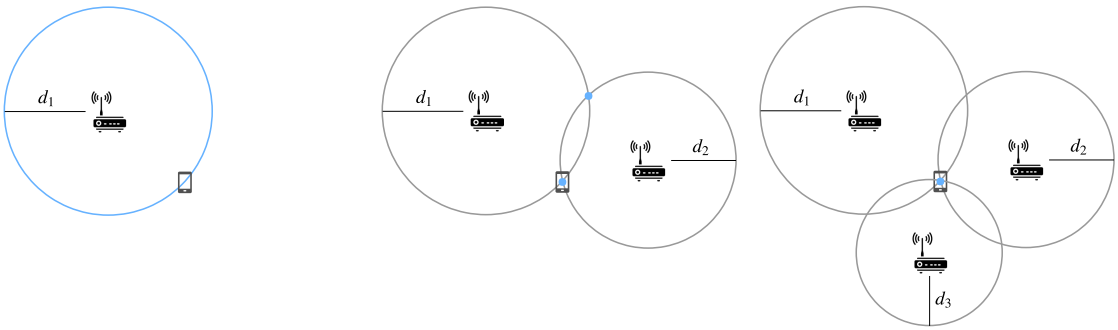


Figure 2.1: The transmission of a WiFi signal between a mobile device and an AP. Equation (2.4) allows us to estimate the distance d between them based on the decrease in the signal's power. Note that free space assumptions are used here ($\gamma = 2$).

Trilateration is the process of determining an object's position by using the geometrical properties of several simultaneous distance measurements from different measuring stations. As visualized in Figure 2.2, the problem in two dimensions boils down to finding the intersection between (at least) three circles around measuring stations. Algebraically, we express the problem as finding the solution for a system of equations [24]

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 = d_1^2 \\ (x - x_2)^2 + (y - y_2)^2 = d_2^2 \\ \vdots \\ (x - x_n)^2 + (y - y_n)^2 = d_n^2, \end{cases} \tag{2.5}$$

where x and y are the unknowns, x_i and y_i are respectively the x- and y-coordinates of the measuring station that produced the i-th measurement, d_i is the i-th measurement, and $n \geq 3$ is the amount of measuring stations that produced measurements.



(a) With one measurement, all points on the circle are possible candidates.

(b) With two measurements, the possible candidates are reduced to at most two points.

(c) With three measurements, there is only a single viable candidate.

Figure 2.2: Geometric properties of two-dimensional trilateration, as used to locate a mobile device based on three distance measurements d_1, d_2, d_3. The measured distances are considered to be noiseless, and the AP's are not collinear.

Extending the theory to the three-dimensional setting is trivial. While the main principles stay the same, we now need to account for the properties of spheres, instead of those of circles. Because of this, three measurements now only narrow the candidates down to at most two, as visualized in Figure 2.3. Four measurements are generally needed to obtain a single optimal

position. The system of equations in Equation (2.5) is simply extended to

$$\begin{cases} (x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2 = d_1^2 \\ (x-x_2)^2 + (y-y_2)^2 + (z-z_2)^2 = d_2^2 \\ \vdots \\ (x-x_n)^2 + (y-y_n)^2 + (z-z_n)^2 = d_n^2, \end{cases} \tag{2.6}$$

where z is also an unknown, and z_i is the z-coordinate of the measuring station that produced the i-th measurement.

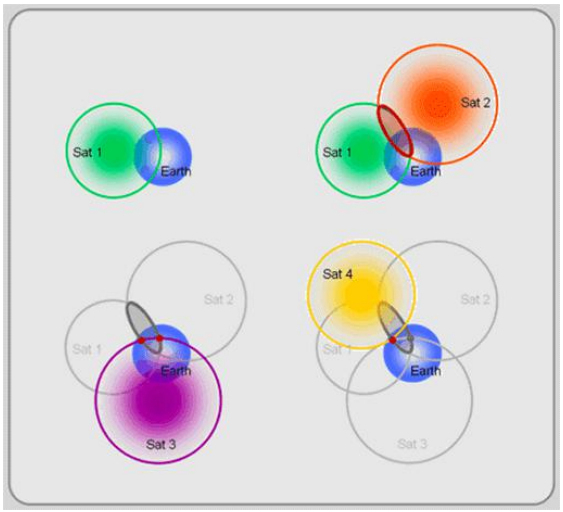


Figure 2.3: Geometric properties of three-dimensional trilateration. The assumptions made are the same as those in Figure 2.2. Image used from Schmandt [25].

As the equations in Equation (2.6) are nonlinear, finding a solution is a nontrivial problem. Analytical solutions do exist [24], [26], [27], but these only account for cases where there are exactly three noncollinear measuring stations, and where the spheres have exactly one intersection [28]. In practice, distance measurements are noisy, making such precise intersections a rarity. The use of more than three or four measurements intuitively should aid in the finding of an optimal position. Because of such constraints associated with analytical solutions, numerical methods are generally used in order to approximate an optimal position [28].

2.4 Model fitting

The theory of trilateration as decribed in the previous section requires distance measurements to be provided. However, the general path loss model in Equation (2.4) can only be used to obtain a value for the distance d when the values for the gains, received and transmitting power, and path loss exponent are known. Of these, our measurements as described in Section 2.1 only provide values for the received power. The other values need to be estimated by some method.

We could estimate these other values, using them to obtain a value for the distance, which we will then use to estimate a position. Instead, we modify the model in Equation (2.4) to use absolute positions in place of distance values, giving

$$P_{\text{receiving}} = P_{\text{transmitting}} + G_{\text{receiving}} + G_{\text{transmitting}} + 10\gamma \log_{10}\left(\frac{\lambda}{4\pi\|\mathbf{x}_\text{t} - \mathbf{x}_\text{r}\|}\right), \tag{2.7}$$

where $\mathbf{x}_\text{t}, \mathbf{x}_\text{r} \in \mathbb{R}^3$ are the coordinates of the transmitting and receiving antenna, respectively, and $\|\cdot\|$ is the Euclidean norm. Using positions as parameters, we can estimate these positions alongside the other values, removing the need to perform our position reconstruction in separate steps.

As all values other than the path loss exponent, received power, and the two positions are generally unknown, we simplify the model by rewriting Equation (2.7) to

$$\begin{aligned} P_{\text{receiving}} &= P_{\text{transmitting}} + G_{\text{receiving}} + G_{\text{transmitting}} + 10\gamma \log_{10}\left(\frac{\lambda}{4\pi\|\mathbf{x}_\text{t} - \mathbf{x}_\text{r}\|}\right) \\ &= P_{\text{transmitting}} + G_{\text{receiving}} + G_{\text{transmitting}} + 10\gamma \log_{10}\left(\frac{\lambda}{4\pi}\right) - 10\gamma \log_{10}\|\mathbf{x}_\text{t} - \mathbf{x}_\text{r}\| \\ &= \eta - 10\gamma \log_{10}\|\mathbf{x}_\text{t} - \mathbf{x}_\text{r}\|, \end{aligned} \tag{2.8}$$

where $\eta = P_{\text{transmitting}} + G_{\text{receiving}} + G_{\text{transmitting}} + 10\gamma \log_{10}\left(\frac{\lambda}{4\pi}\right)$ is a bias term that needs to be estimated alongside the actual location [4], [29].

Now, having obtained a 'final' model of signal transmissions, we can use this model in conjunction with our grouped measurements from Section 2.1, and the theory of trilateration from Section 2.3. Akin to the system in Equation (2.6), we wish to create a system of equations for each timestamp. Using the geometric properties of trilateration as visualised in Figure 2.3, we see that theoretically we need at least four (approximately) simultaneous measurements to be able to pinpoint an object's location with confidence. To this end, we prune our sequence of measurement sets M, alongside the corresponding sequence of timestamps t, producing new sequences $M' = (M'_1, \ldots, M'_{\beta'})$ (which all contain at least four measurements), and the corresponding timestamps $T = (T_1, \ldots, T_{\beta'})$. Here, $\beta' = |\{i \mid 1 \leqslant i \leqslant \beta, |M_i| \geqslant 4\}|$, with $|\cdot|$ being the set cardinality. The ordering of the timestamps in T is unchanged w.r.t. t'.

To each timestamp T_i with measurement set M'_i we now associate a system of equations, consisting of the following equation for each $(P_j, \mathbf{s}_j) \in M'_i$:

$$\eta_i - 10\gamma \log_{10}\| \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^\text{T} - \mathbf{s}_j\| = P_j, \tag{2.9}$$

where η_i, x_i, y_i, and z_i are the unknowns. The path loss exponent is assumed to be constant across all AP's and measurements. [2]

As our measurements are noisy, and the system of equations generally has no solution, we aim to obtain a least squares fit of the unknowns, as is often done with trilateration problems [4], [32], [33]. This involves finding the parameter vector $\hat{\boldsymbol{\theta}}_i$ for which the sum of squared errors for M'_i is minimal:

$$\hat{\boldsymbol{\theta}}_i \in \arg\min_{\boldsymbol{\theta}} \sum_{(P,\mathbf{s}) \in M'_i} \left(P - \left(\eta_i - 10\gamma \log_{10}\| \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^\text{T} - \mathbf{s}\|\right)\right)^2 \quad , \tag{2.10}$$

where $\boldsymbol{\theta} = \begin{bmatrix} \eta_i & x_i & y_i & z_i \end{bmatrix}^\text{T}$.

2.5 Optimization

In order to approximate Equation (2.10), we need to apply some optimization method, as said before in Section 2.3. Press, Teukolsky, Vetterling, *et al.* [34] describe a multitude of local and global function optimization methods that we could apply, each with their own (dis-)advantages and conditions. Instead of using such a general optimization method, we opt to make use of the properties of nonlinear least squares fitting. While many general optimization methods attempt to either approximate or do away with first and second order derivatives for the purpose of generality, nonlinear least squares fitting involves a known model function, with oftentimes known analytical derivatives [35]. We will therefore use a method that can use these derivatives.

Two such methods for nonlinear least squares fitting are the gradient descent and Gauss-Newton methods. The gradient descent method attempts to find a minimum by repeatedly moving in the direction of the steepest descent of the sum of squared errors (i.e. the direction opposite that of the gradient), starting from an initial guess. While this method does guarantee convergence by the definition of the gradient, it is often quite slow, and may even fail to find

[2] This need not be the case in practice, and other authors do in fact attempt approaches in which the path loss exponent is estimated per sensor or measurement [4], [30], [31].

a minimum within reasonable computational bounds [36]. The Gauss-Newton method takes a slightly more sophisticated approach by assuming the sum of squared errors is quadratic near the minimum, and then attempting to find said minimum using linearized least squares approximations. This can result in much higher convergence rates when done with a reasonable initial estimate, but it could also cause diverge or head towards a saddle point, resulting in a nonoptimal solution [37].

These methods perform well in specific parts of the optimization procedure: gradient descent in the initial stage, and Gauss-Newton in the final stage [38]. It is for this reason that there are methods that can be interpreted as being a hybrid between these two methods, interpolating between them each iteration. Of such hybrids, variations of the Levenberg-Marquardt method based on work by Levenberg [39] and Marquardt [40] are in widespread use [41]. In this thesis we will use the variation proposed by Moré [42] and as implemented in the SciPy library for Python [43], because of its increased robustness and availability. We will only give an overview of the 'basic' Levenberg-Marquardt method (based on Madsen, Nielsen, and Tingleff [38], Gavin [41], and Lourakis [44]), as applied to the computation of Equation (2.10). For a thorough discussion of the full method and robustness modifications involved, we refer to the aforementioned literature instead.

We can interpret the problem of least squares minimization in Equation (2.10) as the minimization of a function of the form

$$F(\mathbf{x}) = \frac{1}{2}\sum_{j=1}^{m} f_j^2(\mathbf{x}), \tag{2.11}$$

where $\mathbf{x} = \boldsymbol{\theta}$ is the parameter vector, $(P_1, \mathbf{s}_1), \ldots, (P_m, \mathbf{s}_m)$ are the measurements and AP vectors in M_i', and $f_j(\mathbf{x}) = P_j - \left(\eta_i - 10\gamma \log_{10} \| \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^{\mathrm{T}} - \mathbf{s}_j \|\right)$ is the j-th error value. [3] The Levenberg-Marquardt method specifically requires that $m \geq n$, which is always satisfied in our case.

Let $\mathbf{f} : \mathbb{R}^4 \to \mathbb{R}^m$ with $\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) & \ldots & f_m(\mathbf{x}) \end{bmatrix}^{\mathrm{T}}$ be a vector-valued function containing all errors. We can then define the *Jacobian* $\mathbf{J}(\mathbf{x})$ of $\mathbf{f}(\mathbf{x})$ (the matrix consisting of all its first-order partial derivatives) as

$$\begin{aligned}
\mathbf{J}(\mathbf{x}) &= \begin{bmatrix} \frac{\partial f_1}{\partial \eta_i}(\mathbf{x}) & \frac{\partial f_1}{\partial x_i}(\mathbf{x}) & \frac{\partial f_1}{\partial y_i}(\mathbf{x}) & \frac{\partial f_1}{\partial z_i}(\mathbf{x}) \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_m}{\partial \eta_i}(\mathbf{x}) & \frac{\partial f_m}{\partial x_i}(\mathbf{x}) & \frac{\partial f_m}{\partial y_i}(\mathbf{x}) & \frac{\partial f_m}{\partial z_i}(\mathbf{x}) \end{bmatrix} \\
&= \begin{bmatrix} (\nabla f_1(\mathbf{x}))^{\mathrm{T}} \\ \vdots \\ (\nabla f_m(\mathbf{x}))^{\mathrm{T}} \end{bmatrix} \\
&= \begin{bmatrix} -1 & \frac{10\gamma}{\ln(10)\|\mathbf{v} - \mathbf{s}_1\|^2}(\mathbf{v} - \mathbf{s}_1)^{\mathrm{T}} \\ \vdots & \vdots \\ -1 & \frac{10\gamma}{\ln(10)\|\mathbf{v} - \mathbf{s}_m\|^2}(\mathbf{v} - \mathbf{s}_m)^{\mathrm{T}} \end{bmatrix},
\end{aligned} \tag{2.12}$$

where $\mathbf{v} = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^{\mathrm{T}}$.

Using the Jacobian, we can give the following expression for the gradient $\mathbf{F}'(\mathbf{x}) = \nabla F(\mathbf{x})$:

[3] The factor $\frac{1}{2}$ is included here for the simplification of derivatives.

$$\begin{aligned}\mathbf{F}'(\mathbf{x}) &= \nabla\left(\frac{1}{2}\sum_{j=1}^{m} f_j^2(\mathbf{x})\right)\\ &= \sum_{j=1}^{m} f_j(\mathbf{x})\left(\nabla f_j(\mathbf{x})\right)\\ &= \left(\mathbf{J}(\mathbf{x})\right)^{\mathrm{T}}\mathbf{f}(\mathbf{x}).\end{aligned} \tag{2.13}$$

In addition to the gradient, we also need the matrix of all second partial derivatives of F: the Hessian matrix $\mathbf{F}''(\mathbf{x})$. For an arbitrary second partial derivative of F with $a, b \in \{\eta_i, x_i, y_i, z_i\}$, we have

$$\frac{\partial^2 F}{\partial a \partial b} = \sum_{j=1}^{m}\left(\frac{\partial f_j}{\partial a}(\mathbf{x})\frac{\partial f_j}{\partial b}(\mathbf{x}) + f_j(\mathbf{x})\frac{\partial^2 f_j}{\partial a \partial b}(\mathbf{x})\right). \tag{2.14}$$

From this, we can derive the following expression for the Hessian:

$$\mathbf{F}''(\mathbf{x}) = \left(\mathbf{J}(\mathbf{x})\right)^{\mathrm{T}}\mathbf{J}(\mathbf{x}) + \sum_{j=1}^{m} f_j(\mathbf{x})\mathbf{f}_j''(\mathbf{x}), \tag{2.15}$$

where $\mathbf{f}_j''(\mathbf{x})$ is the Hessian matrix of $f_j(\mathbf{x})$.

As with the Gauss-Newton method, we linearly approximate the error values $f_j(x)$, making the error value Hessians $\mathbf{f}_j''$ negligible, allowing us to simplify Equation (2.15) to

$$\mathbf{F}''(\mathbf{x}) \approx \left(\mathbf{J}(\mathbf{x})\right)^{\mathrm{T}}\mathbf{J}(\mathbf{x}). \tag{2.16}$$

Using our computed gradient and approximated Hessian, we can now proceed with the Levenberg-Marquardt method. The method, like all other nonlinear least squares methods, attempts to iteratively refine the current, $(k+1)$-th parameter vector $\mathbf{x}_{k+1}$ based on the previous estimate $\mathbf{x}_k$, with a given initial estimate $\mathbf{x}_1$, which we will compute as $\mathbf{x}_1 = \begin{bmatrix}0 & \frac{1}{m}\sum_{j=1}^{m}\mathbf{s}_j^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$, i.e. the mean of all AP's that produced measurements at the time in question, with the bias set to 0. As the Levenberg-Marquardt method combines gradient descent with the Gauss-Newton method, we will first give their respective methods for obtaining $\mathbf{x}_{k+1}$.

Gradient descent updates the current estimate by moving in the direction of steepest descent, scaled by $\lambda > 0$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \lambda\mathbf{F}'(\mathbf{x}_k). \tag{2.17}$$

This method is not always computationally feasible in terms of its convergence rate. The Gauss-Newton method is more efficient in practice, assuming the current estimate is near the solution. It achieves this efficiency by approximating the function around the current estimate by a quadratic function. This is done by approximating the gradient by a linear function using its Taylor series around the current estimate:

$$\mathbf{F}'(\mathbf{x}_{k+1}) \approx \mathbf{F}'(\mathbf{x}_k) + \mathbf{F}''(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k). \tag{2.18}$$

Solving this for $\mathbf{F}'(\mathbf{x}_{k+1}) = \mathbf{0}$ gives the Gauss-Newton update

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\mathbf{F}''(\mathbf{x}_k)\right)^{-1}\mathbf{F}'(\mathbf{x}_k). \tag{2.19}$$

Levenberg [39] combines Equations (2.17) and (2.19) by introducing a so-called *damping parameter* $\mu_k > 0$ with given initial value μ_1:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\mathbf{F}''(\mathbf{x}_k) + \mu_k\mathbf{I}\right)^{-1}\mathbf{F}'(\mathbf{x}_k), \tag{2.20}$$

where $\mathbf{I}$ is the identity matrix. The damping parameter allows each iteration to alternate between gradient descent (large values of μ_k) and Gauss-Newton (small values of μ_k). Marquardt [40]

pointed out that Equation (2.20) completely neglects the Hessian $\mathbf{F}''(\mathbf{x}_k)$ for large values of μ_k, which is not necessarily advantageous. Even when solely using gradient descent, the Hessian can be used to give information about the curvature of the function, by moving a greater amount in directions where the gradient itself is smaller. This can be implemented by scaling the movement size λ in Equation (2.17) by the Hessian:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\mathbf{F}''(\mathbf{x}_k) + \mu_k \mathbf{diag}\left(\mathbf{F}''(\mathbf{x}_k)\right)\right)^{-1} \mathbf{F}'(\mathbf{x}_k), \tag{2.21}$$

where $\mathbf{diag}\left(\mathbf{F}''(\mathbf{x}_k)\right) \in \mathbb{R}^{4\times 4}$ is the diagonal matrix with the same diagonal as $\mathbf{F}''(\mathbf{x}_k)$. Expressing this using the Jacobian gives us the final equation

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\left(\mathbf{J}(\mathbf{x}_k)\right)^{\mathrm{T}} \mathbf{J}(\mathbf{x}_k) + \mu_k \mathbf{diag}\left(\left(\mathbf{J}(\mathbf{x}_k)\right)^{\mathrm{T}} \mathbf{J}(\mathbf{x}_k)\right)\right)^{-1} \left(\mathbf{J}(\mathbf{x}_k)\right)^{\mathrm{T}} \mathbf{f}(\mathbf{x}_k). \tag{2.22}$$

The general idea of the Levenberg-Marquardt method now is to attempt to perform this update, getting a new estimate $\widetilde{\mathbf{x}}_{k+1}$, and to then compute the sum of squared errors again. If the value is significantly smaller than the previous value, i.e. if $F(\mathbf{x}_k) - F(\widetilde{\mathbf{x}}_{k+1}) \geqslant \epsilon_{\text{est}}$ for some threshold ϵ_{est}, we accept the new estimate as $\mathbf{x}_{k+1} = \widetilde{\mathbf{x}}_{k+1}$. [4] Since this means we are getting closer to a solution, we increase μ_{k+1} w.r.t. μ_k by some method, such as scaling it by some factor $c_\uparrow$ as $\mu_{k+1} = c_\uparrow \mu_k$. On the other hand, if $F(\mathbf{x}_k) - F(\widetilde{\mathbf{x}}_{k+1}) < \epsilon_{\text{est}}$, we do not accept the new estimate, and set $\mathbf{x}_{k+1} = \mathbf{x}_k$. The approximation did not perform well apparently, and thus we decrease μ_{k+1}, which we can also simply do by downscaling with a factor $c_\downarrow$ to $\mu_{k+1} = \frac{\mu_k}{c_\downarrow}$.

We continue the process like this until we either obtain convergence in the gradient (if it approaches $\mathbf{0}$) or in the sum of squared errors. Alternatively, we can stop when we reach a maximum amount of iterations. Figure 2.4 visualizes the iterations of the Levenberg-Marquardt method as applied to the computation of Equation (2.10) in two dimensions.

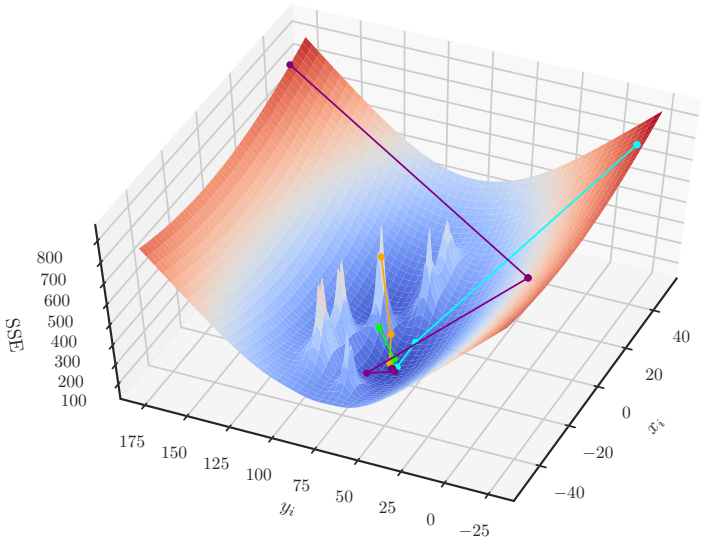


Figure 2.4: The progress of the Levenberg-Marquardt method for four different initial estimates, used to find the $\begin{bmatrix} x_i & y_i \end{bmatrix}^{\mathrm{T}}$ that minimizes the sum of squared errors SSE for a measurement set M_i'. The equation used is the same as in Equation (2.10), with constant values for η_i and z_i. The green line represents the method performed with $\mathbf{x}_1 = \frac{1}{m}\sum_{j=1}^{m} \mathbf{s}_j$.

[4] In practice, the sums of squared errors are normalized in some way, so the choice of ϵ_{est} is less problem-dependent. We refer to aforementioned literature for methods.

2.6 Final processing

Performing the Levenberg-Marquardt method for each measurement set M_i', we obtain a sequence of parameter vectors $\hat{\boldsymbol{\theta}}_i = \begin{bmatrix} \eta_i & x_i & y_i & z_i \end{bmatrix}^{\mathrm{T}}$ containing not only the coordinates we seek, but also the bias term, which is of no interest to the sought after positions. Like the bias term, the z-coordinates are of no interest to our problem of position reconstruction as formulated in Chapter 1, as the altitude at which people walk does not fluctuate over time in an ordinary building, except when moving to another floor. Accounting for stairs and other floor transitions introduces unneccessary complexity to our method. We thus obtain a final time series of position vectors $\mathbf{X}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^{\mathrm{T}}$ with corresponding timestamps T_i, as visualized in Figure 2.5.

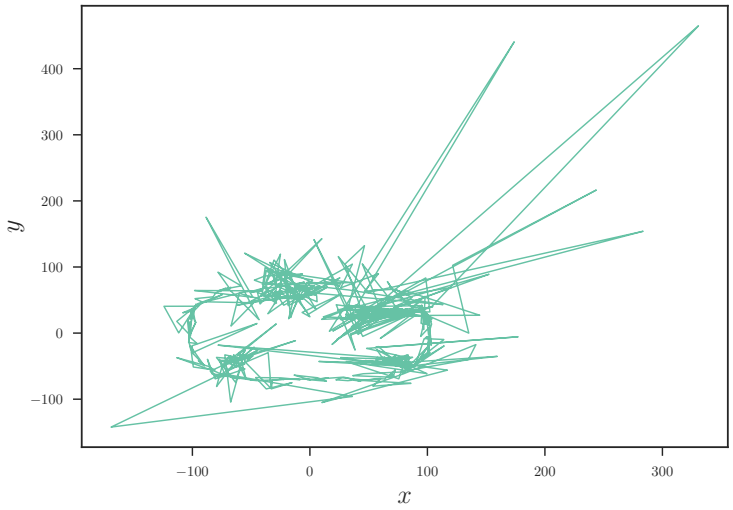


Figure 2.5: The resulting path of $\mathbf{X}_i$ vectors obtained by applying the Levenberg-Marquardt method to a sample walk.

CHAPTER 3

Filtering methods

The inherent noise encountered in WiFi-based position reconstruction requires some modification or additional method to be applied to either the original measurements or the $\mathbf{X}_i$ positions obtained through the procedure described in Chapter 2. As mentioned by Van Engelen, Van Lier, Takes, *et al.* [4], opting to apply noise-reducing methods (such as the signal filtering methods in Section 3.1) to the $\mathbf{X}_i$ positions is a viable option based on the knowledge that the position of a mobile device is expected to changes at a slow rate. This is not the case for the RSSI measurements, since these can fluctuate heavily because of interference or the multipath phenomenon [45].

For the noise-reduction of the position time series, we opt to use filtering methods. Such methods generally function relatively efficiently, due to them usually producing a filtered value based on either preceding or (a limited amount of) surrounding values. Using those methods that have this latter property opens up the possibility of building applications that perform (near) real-time position reconstruction and noise filtering. [5]

This chapter will discuss a variety of filtering methods, divided into signal and probabilistic filtering methods. For each method, we will give the necessary equations and procedures to compute a sequence of filtered positions $\boldsymbol{\mathcal{X}}_i$ from observed positions $\mathbf{X}_i$.

3.1 Signal filtering

Signal filtering methods are named in such a manner because of their wide range of applications within the area of digital signal processing. These methods can usually (but not always) be formulated as a single equation consisting of common formulas, representing a so-called discrete-time system [46]. Such equations do not attempt to model the process behind or the distribution of the positions, as opposed to the methods suggested in Section 3.2, which often need to be expressed as algorithms or procedures.

In this section we will discuss four general signal filtering methods, three of which were already proposed by Van Engelen, Van Lier, Takes, *et al.* [4] for position reconstruction. We will also discuss several variations on all these methods.

3.1.1 Exponential filtering

Exponential filtering (often known as exponential smoothing) is a widely-used approach for the filtering of time series. Its popularity can be owed to its simplicity and high accuracy when used for time series forecasting [47]. The simplest form of the method as applied to scalar values can be expressed as [48]

$$s_i = \alpha x_i + (1 - \alpha) s_{i-1}, \tag{3.1}$$

[5] While such applications are of great interest, they are beyond the scope of this thesis. Throughout this thesis we will always make the assumption that the *entirety* of the data is available from the beginning.

where $0 < \alpha < 1$ is a smoothing parameter, s_i and s_{i-1} are the i- and $(i-1)$-th filtered value, and x_i is the i-th observation. By assuming that the x- and y-coordinates are not codependent, we can derive from Equation (3.1) the equation as used to obtain the filtered positions

$$\boldsymbol{\mathcal{X}}_i = \alpha \mathbf{X}_i + (1-\alpha)\, \boldsymbol{\mathcal{X}}_{i-1}\,. \tag{3.2}$$

Decreasing α in Equation (3.2) causes more of the history of the positions to be taken into account for its filtering. This history requires some choice to be made for the initial filtered value $\boldsymbol{\mathcal{X}}_1$. Some common heuristics are the following [49], [50]:

(a.) Setting it to the first observation: $\boldsymbol{\mathcal{X}}_1 = \mathbf{X}_1$.

(b.) Reversing the order of the observed positions, obtaining reverse filtered positions $\boldsymbol{\mathcal{X}}^*_i$ with $\boldsymbol{\mathcal{X}}^*_1 = \mathbf{X}_l$ (where l is the amount of observed positions $\mathbf{X}_i$), and setting $\boldsymbol{\mathcal{X}}_1 = \boldsymbol{\mathcal{X}}^*_l$.

(c.) Setting it to the mean of the first p (which is usually $p = 3$) observations: $\boldsymbol{\mathcal{X}}_1 = \frac{1}{p}\sum_{i=1}^{p} \mathbf{X}_i$.

In addition to the variety of methods for choosing the initial filtered value, there are some possible modifications to the filtering equation in Equation (3.2). These modifications involve accounting for the irregular frequency of the time values, filtering the observations more than once, and combining these.

Irregular time filtering Exponential filtering usually assumes the observations are spaced equally in time. Contrary to this, intuition tells us that more recent observations should have more weight during filtering. By introducing a time window duration $\tau > 0$, also expressed in seconds, we can describe a version of Equation (3.2) for irregularly spaced positions that takes the position timestamps T_i into account [51]:

$$\boldsymbol{\mathcal{X}}_i = \alpha_i \mathbf{X}_i + (1-\alpha_i)\, \boldsymbol{\mathcal{X}}_{i-1}, \tag{3.3}$$

where $\alpha_i = 1 - \exp\left(\frac{-(T_i - T_{i-1})}{\tau}\right)$.

3.1.1.1 Double exponential filtering

Simple exponential smoothing as described in Equation (3.2) does not perform very well in situations where the time series displays a steady trend. For such observations, double exponential filtering is an apt choice, because of its adjustment for the trend [52].

Its procedure requires Equation (3.2) to be used in order to obtain first-level filtered positions $\boldsymbol{\mathcal{X}}'_i$. Treating these positions like observations, the second-level filtered positions $\boldsymbol{\mathcal{X}}''_i$ are obtained by performing simple exponential filtering again with $\boldsymbol{\mathcal{X}}''_1 = \boldsymbol{\mathcal{X}}'_i$ (which is obtained by one of the heuristics named earlier):

$$\boldsymbol{\mathcal{X}}''_i = \alpha\, \boldsymbol{\mathcal{X}}'_i + (1-\alpha)\, \boldsymbol{\mathcal{X}}''_{i-1}\,. \tag{3.4}$$

Using these two levels of filtered positions, we obtain the final doubly exponential filtered positions

$$\boldsymbol{\mathcal{X}}_i = \left(2\, \boldsymbol{\mathcal{X}}'_i - \boldsymbol{\mathcal{X}}''_i\right) + \frac{\alpha}{1-\alpha}\left(\boldsymbol{\mathcal{X}}'_i - \boldsymbol{\mathcal{X}}''_i\right). \tag{3.5}$$

In order to also account for irregularity in the time samples, it is sufficient to replace each α in Equation (3.5) (and in the computation of $\boldsymbol{\mathcal{X}}'_i$ and $\boldsymbol{\mathcal{X}}''_i$) by α_i as used in Equation (3.3). [6]

[6] $\boldsymbol{\mathcal{X}}_1$ needs to be defined as $\mathbf{X}_1$ due to the lack of a previous time value. Equation (3.5) did not require this to be defined separately, as that equation already computes $\boldsymbol{\mathcal{X}}_1 = \mathbf{X}_1$.

3.1.2 Gaussian filter

Gaussian filters are a staple in the area of image processing, where they are referred to as Gaussian blurs, used to reduce noise for applications like edge detection and human vision modelling [53]. Those images are essentially treated like two-dimensional signals, and the end-result is obtained by the discrete convolution of the image with a so-called sampled Gaussian function. When applied to our sequence of positions $\mathbf{X}_i$, which is a vector-valued signal, we can also express the filter as a moving weighted average with weights sampled from the one-dimensional Gaussian function $G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$, where $\sigma > 0$ represents the scale (or standard deviation) of the underlying Gaussian. As we use a discretely sampled Gaussian function, the sum of all weights will not sum to 1, calling for the inclusion of a normalization constant N. This gives us [4], [53]

$$\begin{aligned}\boldsymbol{\mathcal{X}}_i &= (\frac{1}{N} G_\sigma * \mathbf{X})(i) \\ &= \frac{1}{N} \sum_{j=-\infty}^{\infty} G_\sigma(j-i)\mathbf{X}_j \\ &= \frac{1}{\sum_{k=-\infty}^{\infty} G_\sigma(k-i)} \sum_{j=-\infty}^{\infty} G_\sigma(j-i)\mathbf{X}_j, \end{aligned} \tag{3.6}$$

where $\mathbf{X}_j = \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathrm{T}}$ if there is no j-th observed position.

As most of the Gaussian distribution lies within a certain band around the mean, we can rewrite Equation (3.6) to the more computationally feasible (and still approximately equivalent)

$$\boldsymbol{\mathcal{X}}_i = \frac{1}{\sum_{k=i-\lceil 5\sigma \rceil}^{i+\lceil 5\sigma \rceil} G_\sigma(k-i)} \sum_{j=i-\lceil 5\sigma \rceil}^{i+\lceil 5\sigma \rceil} G_\sigma(j-i)\mathbf{X}_j. \tag{3.7}$$

Irregular time filtering As with simple exponential filtering, Gaussian filters also assume that observations are spaced equally in time. In order to extend Equation (3.7) to account for the irregularly sampled timestamps, we need to use the time values $T_{\sigma,i}$ in the Gaussian functions:

$$\boldsymbol{\mathcal{X}}_i = \frac{1}{\sum_{k \in \mathcal{T}_{\sigma,i}} G_\sigma(T_k - T_i)} \sum_{j \in \mathcal{T}_{\sigma,i}} G_\sigma(T_j - T_i)\mathbf{X}_j, \tag{3.8}$$

where $\mathcal{T}_{\sigma,i} = \{k \mid (T_i - 5\sigma) \leqslant T_k \leqslant (T_i + 5\sigma)\}$.

3.1.3 Savitzky-Golay filter

The Savitzky-Golay filter is a simple filter proposed by Savitzky and Golay [54], widely used in spectroscopy. The principle behind the filter is to fit a polynomial to a window of observations around the observation to be filtered using linear least squares, with the observations being assigned 'local' coordinates with respect to the observation to be filtered. The filtered value is then equal to the value of the fitted polynomial in the middle of the window, i.e. at $x = 0$ (where x is the independent variable of the polynomial).

While the original paper describes this process through a convolution with certain (possible pre-computed) coefficients (thus being a type of moving average), Persson and Strang [55] show that the coefficients $\mathbf{c} = \begin{bmatrix} c_0 & \dots & c_\kappa \end{bmatrix}^{\mathrm{T}}$ for the best fitted polynomial $c_0 + c_1 x + \cdots + c_\kappa x^\kappa$ of degree κ, fitted to $2m+1$ subsequent and equally spaced values $\mathbf{b} = \begin{bmatrix} b_{-m} & \dots & b_m \end{bmatrix}^{\mathrm{T}}$ with window length m, can be computed as the least squares solution of

$$\mathbf{V}\mathbf{c} = \mathbf{b}, \tag{3.9}$$

where

$$\mathbf{V} = \begin{bmatrix} (-m)^0 & \dots & (-m)^\kappa \\ (-m+1)^0 & \dots & (-m+1)^\kappa \\ \vdots & \ddots & \vdots \\ (m-1)^0 & \dots & (m-1)^\kappa \\ m^0 & \dots & m^\kappa \end{bmatrix}, \mathbf{c} = \begin{bmatrix} c_0 \\ \vdots \\ c_\kappa \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_{-m} \\ \vdots \\ b_m \end{bmatrix}.$$

The least squares solution of the system in Equation (3.9) is computed using the Moore-Penrose pseudoinverse of $\mathbf{V}$:

$$\mathbf{c} = (\mathbf{V}^\mathrm{T}\mathbf{V})^{-1}\mathbf{V}^\mathrm{T}\mathbf{b}. \tag{3.10}$$

As the filtered value we are interested in is equal to the value of the polynomial at the independent variable $x = 0$, we only require the coefficient c_0 to get the filtered value s:

$$\big[s\big] = \underbrace{\big[1 \quad 0 \quad \dots \quad 0\big]}_{(\kappa+1) \text{ values}}(\mathbf{V}^\mathrm{T}\mathbf{V})^{-1}\mathbf{V}^\mathrm{T}\mathbf{b}. \tag{3.11}$$

Applying this to our case, position vectors $\mathbf{X}_i$, we see that we need to extend the system in Equation (3.9) to perform a least squares fit for both the x- and y-directions of the position:

$$\mathbf{VC} = \mathbf{B}_i, \tag{3.12}$$

where

$$\mathbf{C} = \begin{bmatrix} c_{x,0} & c_{y,0} \\ \vdots & \vdots \\ c_{x,\kappa} & c_{y,\kappa} \end{bmatrix}, \mathbf{B}_i = \begin{bmatrix} \mathbf{X}_{i-m}^\mathrm{T} \\ \vdots \\ \mathbf{X}_{i+m}^\mathrm{T} \end{bmatrix}.$$

Using the extended system of Equation (3.12), we can give the following expression for the filtered positions:

$$\boldsymbol{\mathcal{X}}_i = \left(\underbrace{\big[1 \quad 0 \quad \dots \quad 0\big]}_{(\kappa+1) \text{ values}}(\mathbf{V}^\mathrm{T}\mathbf{V})^{-1}\mathbf{V}^\mathrm{T}\mathbf{B}_i \right)^\mathrm{T}. \tag{3.13}$$

Besides the choice of window size m and polynomial degree κ, the filter requires us to decide how the filtered positions of the observed data $\mathbf{X}_1, \dots, \mathbf{X}_m$ and $\mathbf{X}_{l-m+1}, \dots, \mathbf{X}_l$ are computed, where $\mathbf{X}_l$ is the final observation. Some common heuristics are the following:

(a.) Pad the observations with fictitious positions $\mathbf{X}_{-m+1}, \dots, \mathbf{X}_0$ and $\mathbf{X}_{l+1}, \dots, \mathbf{X}_{l+m}$, and then compute the filtered positions as usual. These padded positions $\mathbf{X}_j$ are defined by the mirroring of the edge they are on:

$$\mathbf{X}_j = \begin{cases} \mathbf{X}_{1-j} & \text{if } j \leqslant 0 \\ \mathbf{X}_{2l-j+1} & \text{if } j > l. \end{cases} \tag{3.14}$$

(b.) Similarly pad the observations with the same amount of fictitious positions, but define them as being equal to the last 'real' position on their edge:

$$\mathbf{X}_j = \begin{cases} \mathbf{X}_1 & \text{if } j \leqslant 0 \\ \mathbf{X}_l & \text{if } j > l. \end{cases} \tag{3.15}$$

(c.) Determine the lowest and highest indices $j_\mathrm{min} = m + 1$ and $j_\mathrm{max} = l - m$ (which can be equal) for which Equation (3.13) can be computed without problems. Let $\mathbf{C}^*_\mathrm{min}$ and

$\mathbf{C}^*_{\max}$ be the full coefficient matrices computed in Equation (3.13) for indices $j_{\min}$ and $j_{\max}$ respectively. Then we compute the remaining filtered positions as

$$\boldsymbol{\mathcal{X}}_j = \begin{cases} \left(\begin{bmatrix} 1 & j-j_{\min} & (j-j_{\min})^2 & \dots & (j-j_{\min})^\kappa \end{bmatrix} \mathbf{C}^*_{\min}\right)^{\mathrm{T}} & \text{if } j < j_{\min} \\ \left(\begin{bmatrix} 1 & j-j_{\max} & (j-j_{\max})^2 & \dots & (j-j_{\max})^\kappa \end{bmatrix} \mathbf{C}^*_{\max}\right)^{\mathrm{T}} & \text{if } j > j_{\max}, \end{cases} \tag{3.16}$$

i.e. use the least squares fits computed at the edges to fill in all remaining indices. It should be noted that this heuristic does require that $l \geqslant 2m+1$, as at least one position needs to be filtered using the 'basic' Savitzky-Golay procedure.

Irregular time filtering Extending Equation (3.13) to account for the irregular spacing of positions poses no theoretical difficulties. We merely need to change the 'local' coordinates in Equation (3.12) to coordinates based on the spacing of the corresponding time values:

$$\mathbf{V}_i\mathbf{C} = \mathbf{B}_i, \tag{3.17}$$

where

$$\mathbf{V}_i = \begin{bmatrix} (T_{i-m}-T_i)^0 & \dots & (T_{i-m}-T_i)^\kappa \\ (T_{i-m+1}-T_i)^0 & \dots & (T_{i-m+1}-T_i)^\kappa \\ \vdots & \ddots & \vdots \\ (T_{i+m-1}-T_i)^0 & \dots & (T_{i+m-1}-T_i)^\kappa \\ (T_{i+m}-T_i)^0 & \dots & (T_{i+m}-T_i)^\kappa \end{bmatrix}.$$

Similarly to the heuristic chosen for the computation of edge-case filtered positions, this method also requires a definition for time values before or after the given sequence of time values. We make the assumption that such undefined time values are spaced in equal intervals starting from the edges, with the intervals equal in size to that of the two time values on that edge:

$$T_j = \begin{cases} T_1 - (1-j)(T_2 - T_1) & \text{if } j \leqslant 0 \\ T_l + (j-l)(T_l - T_{l-1}) & \text{if } j > l. \end{cases} \tag{3.18}$$

Using this heuristic and the matrices in Equation (3.17), we obtain the following equation for the filtered position:

$$\boldsymbol{\mathcal{X}}_i = \left(\underbrace{\begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}}_{(\kappa+1) \text{ values}} (\mathbf{V}_i^{\mathrm{T}}\mathbf{V}_i)^{-1}\mathbf{V}_i^{\mathrm{T}}\mathbf{B}_i \right)^{\mathrm{T}}. \tag{3.19}$$

3.1.4 Median filter

Similarly to the Gaussian filter, the median filter is widely used in image processing. Its popularity can be attributed to its property of reducing noise whilst preserving sharp edges [56]. The filter operates using a moving window of size $2m+1$ over some data sequence. The filtered value $\hat{x}_i$ for each data value x_i is defined to be the median of the window $x_{i-m}, \dots, x_{i+m}$ around that value.

To obtain the median, we begin by sorting the window around x_i, obtaining sorted indices $i_1, \dots, i_{2m+1}$ such that $x_{i_j} \leqslant x_{i_{j+1}}$ for $1 \leqslant j \leqslant 2m+1$. As the window is of an odd size, we see that

$$\hat{x}_i = x_{i_{m+1}}. \tag{3.20}$$

Just as with the Savitzky-Golay filter, the median filter requires some choice to be made for the way boundaries are treated. To this end, we choose to pad the data with values $x_{1-m}, \dots, x_0$ and $x_{l+1}, \dots x_{l+m}$ (where l is again the size of the original data), defined identically to Equation (3.15).

The filtered position vectors are computed by applying this procedure separately to the sequences of x- and y-coordinates, combining their filtered values into

$$\boldsymbol{\mathcal{X}}_i = \begin{bmatrix} \hat{x}_i & \hat{y}_i \end{bmatrix}^{\mathrm{T}}. \tag{3.21}$$

Irregular time filtering As with the other filters discussed so far, the median filter also assumes equally spaced data. We opt to account for irregularly spaced data by using the weighted median filter as described by Yin, Yang, Gabbouj, *et al.* [57]. To each value in the window around the value at time i we associate positive weight values $w_{i_1}, \ldots, w_{i_{2m+1}}$. The goal of the weighted median filter is then to compute

$$\hat{x}_i \in \arg\min_{\theta} \sum_{j=1}^{2m+1} w_{i_j} |x_{i_j} - \theta|. \tag{3.22}$$

Yin, Yang, Gabbouj, *et al.* [57] suggest computing this by iterating over the sorted window values from highest to lowest, and setting the weighted median to the first value for which the cumulative sum of the weights up to the corresponding weight is greater than or equal to the half of the sum of all weights. This can be expressed as

$$\hat{x}_i = x_{i_u}, \text{ where } u = \max\left\{ j \;\middle|\; 1 \leqslant j \leqslant 2m+1, \sum_{k=j}^{2m+1} w_{i_k} \geqslant \frac{1}{2} \sum_{k=1}^{2m+1} w_{i_k} \right\}. \tag{3.23}$$

It might however be the case that half of the weight sum can be obtained at a different (preceding) value by computing the cumulative sums from the opposite direction. We compute the final weighted median value by taking the mean of the medians computed in both directions (which can be equal):

$$\hat{x}_i = \frac{1}{2}(x_{i_u} + x_{i_d}), \text{ where } d = \min\left\{ j \;\middle|\; 1 \leqslant j \leqslant 2m+1, \sum_{k=1}^{j} w_{i_k} \geqslant \frac{1}{2} \sum_{k=1}^{2m+1} w_{i_k} \right\}. \tag{3.24}$$

In order to use Equation (3.24), we still need to define some scheme to compute the weights for each window, based on the (differences in) time values. We suggest two such schemes.

Gaussian weights Using the Gaussian function from Equation (3.8) and a scale parameter σ, we can compute weights based on their distance in time to the central value of the window:

$$w_{i_j} = G_\sigma(T_j - T_i). \tag{3.25}$$

Shifted time differences Instead of having to introduce an additional parameter to compute the weights, we propose a weighting scheme that only relies on the data. Let $\Delta T(i,j) = |T_i - T_j|$. We associate each of the sorted data values x_{i_j} in the window around x_i to the value $\Delta T(i,j)$, creating a sequence $D = (\Delta T(i,1), \ldots, \Delta T(i,2m+1))$. Using these values, we define the weights as

$$w_{i_j} = \max(D) - \Delta T(i,j) + \min_2(D), \tag{3.26}$$

where $\min_2(D)$ is the second smallest value in D. This term is added to guarantee that all weights are positive.

3.2 Probabilistic filtering

While signal filtering methods are generally quite efficient, they do not use any information about the nature of the sequence of positions. By attempting to model the way the subject of the measurements is walking using some physical process model, we should be able to see

that some positions are either impossible or extremely unlikely. Probabilistic filtering methods combine the use of such a process model with the modeling of the (un)certainty in both the process model and the observed positions.

In this section we will discuss one such probabilistic filtering method: the Kalman filter. We will explore the basic theory of the method and some interesting variations, along with an overview of process models used for position filtering.

3.2.1 Kalman filters

The Kalman filter, proposed by several authors but most famously by Kálmán [58], is a procedure that attempts to estimate the (hidden) state of some system governed by a (usually) linear discrete-time process, based on its previous estimates and measurements. It has found great success within the area of navigation and position reconstruction, being used in GPS and spacecraft guidance [59], and can more generally be used for all kinds of time series analysis.

3.2.1.1 Basic Kalman filter

We will discuss the basic procedure largely based on Faragher [59], and Bishop and Welch [60], but with notation more in tune with that used throughout preceding parts of this chapter. Instead of giving a complete derivation of the filter, we will only describe the intuitive approach, and refer to Welling [61] for the former.

The filtering procedure attempts to estimate the true, unobservable state $\mathbf{x}_i \in \mathbb{R}^n$ at a time step i. This state is assumed to have evolved from the state at the previous time step through the following model:

$$\mathbf{x}_i = \mathbf{F}_i \mathbf{x}_{i-1} + \mathbf{B}_i \mathbf{u}_i + \mathbf{w}_i. \tag{3.27}$$

Here, $\mathbf{F}_i$ is the process matrix that relates the previous state $\mathbf{x}_{i-1}$ to the current state, $\mathbf{u}_i \in \mathbb{R}^m$ is the so-called control vector containing known systematic 'inputs' at time i (which are not applicable in our case, but are present in many other systems, such as the input from a steering wheel for vehicular navigation), $\mathbf{B}_i$ is the control input matrix relating the control input to its effect on the state, and $\mathbf{w}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_i)$ is a random vector representing the noise in the process for the state, drawn from the Gaussian distribution with mean $\mathbf{0}$ and covariance matrix $\mathbf{Q}_i$. Control inputs and control input matrices will be omitted throughout the rest of this thesis.

As stated before, the Kalman filter uses both the previous state estimates, and observable measurements. These measurements $\mathbf{X}_i \in \mathbb{R}^2$ are also assumed to have been obtained from the real system through a linear model:

$$\mathbf{X}_i = \mathbf{H}_i \mathbf{x}_i + \mathbf{v}_i. \tag{3.28}$$

Here, $\mathbf{H}_i$ is the measurement matrix that transforms the state at time i into a measurement, and $\mathbf{v}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_i)$ is a random vector representing the noise in the measuring process, with a Gaussian distribution similar to that of $\mathbf{w}_i$, but with covariance matrix $\mathbf{R}_i$.

The goal of the filter is to produce filtered state estimates by modeling the distribution of the (un)certainty in the states. We model this distribution using a multivariate Gaussian probability density function with mean $\hat{\mathbf{x}}_i$ (which is also the filtered estimate for time i, as it is the most probable) and covariance matrix $\mathbf{P}_i$. At each time step, we compute $\hat{\mathbf{x}}_i$ based on the previous mean and covariance, and the current measurement, with some heuristic for the initial state and covariance.

The filter computes these two state estimates in two separate stages, commonly referred to as the prediction and update stages. During the prediction stage, the filter applies the process model to the previous distribution given by estimates $\hat{\mathbf{x}}_{i-1}$ and $\mathbf{P}_{i-1}$, creating a predicted Gaussian with mean $\hat{\mathbf{x}}_i^-$ and covariance $\mathbf{P}_i^-$. As in Equation (3.27), the process model at time i is represented by the matrix $\mathbf{F}_i$. By basic (multivariate) probability theory, we know that the mean of the new Gaussian distribution obtained by applying $\mathbf{F}_i$ to the distribution of states is simply

$$\hat{\mathbf{x}}_i^- = \mathbf{F}_i \hat{\mathbf{x}}_{i-1}. \tag{3.29}$$

The covariance of the new Gaussian can also be derived through similar means:

$$\mathbf{P}_i^- = \mathbf{F}_i \mathbf{P}_{i-1} \mathbf{F}_i^{\mathrm{T}} + \mathbf{Q}_i, \tag{3.30}$$

where $\mathbf{Q}_i$ is the covariance matrix of the process noise in Equation (3.27).

During the update stage, the Gaussian of the prediction is combined with the measurement $\mathbf{X}_i$. The filter weighs the estimate and the measurement based on their respective uncertainties. When the covariance matrix $\mathbf{P}_i^-$ of the prediction is generally small, it is given more weight, and similarly for the covariance matrix $\mathbf{R}_i$ of the measurements. The notion of this weighing is captured in the so-called Kalman gain matrix

$$\mathbf{K}_i = \mathbf{P}_i^- \mathbf{H}_i^{\mathrm{T}} \left(\mathbf{H}_i \mathbf{P}_i^- \mathbf{H}_i^{\mathrm{T}} + \mathbf{R}_i \right)^{-1}. \tag{3.31}$$

Using this, we can complete our definition of the update stage and give the equation for the filtered mean

$$\hat{\mathbf{x}}_i = \hat{\mathbf{x}}_i^- + \mathbf{K}_i \left(\mathbf{X}_i - \mathbf{H}_i \hat{\mathbf{x}}_i^- \right). \tag{3.32}$$

Instead of the usual definition for the filtered covariance matrix, we use the so-called Joseph form as described by Brown and Hwang [62], which is numerically more stable and performs at least as well for problems which do not mesh well with the linear and Gaussian assumptions of the filter:

$$\mathbf{P}_i = (\mathbf{I} - \mathbf{K}_i \mathbf{H}_i) \mathbf{P}_i^- (\mathbf{I} - \mathbf{K}_i \mathbf{H}_i)^{\mathrm{T}} + \mathbf{K}_i \mathbf{R}_i \mathbf{K}_i \tag{3.33}$$

Finally, in order to obtain a filtered position vector from these estimates, we merely need to transform the filtered mean $\hat{\mathbf{x}}_i$ using the measurement matrix:

$$\boldsymbol{\mathcal{X}}_i = \mathbf{H}_i \hat{\mathbf{x}}_i. \tag{3.34}$$

Figure 3.1 contains a schematic overview of the Kalman filter procedure. As this overview shows, a lot of parameters are used in the filter. Most of these are dependent on the process models used, as described in the following sections.

Brownian motion model Brownian motion attempts to describe the movement of some target by way of a purely stochastic process, and is often used for the modelling of movement in dense human crowds [63], [64]. Such motion is a type of random walk, and each change in state can be fully attributed to a zero-mean Gaussian distributed random variable.

The state being tracked is identically formatted to the measurements $\mathbf{X}_i$, consisting of x- and y-coordinates, and does not evolve through a process model, but through a stochastic process:

$$\hat{\mathbf{x}}_i = \begin{bmatrix} \hat{x}_i \\ \hat{y}_i \end{bmatrix}, \mathbf{H}_i = \mathbf{I}, \mathbf{F}_i = \mathbf{I}. \tag{3.35}$$

While the initial mean can be defined using a simple heuristic where the first observation is used, defining the initial covariance is less intuitive. Labbe Jr. [65] proposes setting it to a diagonal matrix (as the covariance between different entries is hard to estimate). Entries that are present in both the state and measurements are set to the corresponding entry in the measurement covariance matrix $\mathbf{R}_1$. Other entries are set to a realistic 'maximum' value. As our state and measurements contain the same type of information, we can simply define our initialization as

$$\hat{\mathbf{x}}_1 = \mathbf{X}_1, \mathbf{P}_1 = \mathbf{diag}\left(\mathbf{R}_1\right). \tag{3.36}$$

Defining the covariance matrix for the process and measurement noise is a non-trivial and important task, as the filter does not automatically update these. We have no reason to think

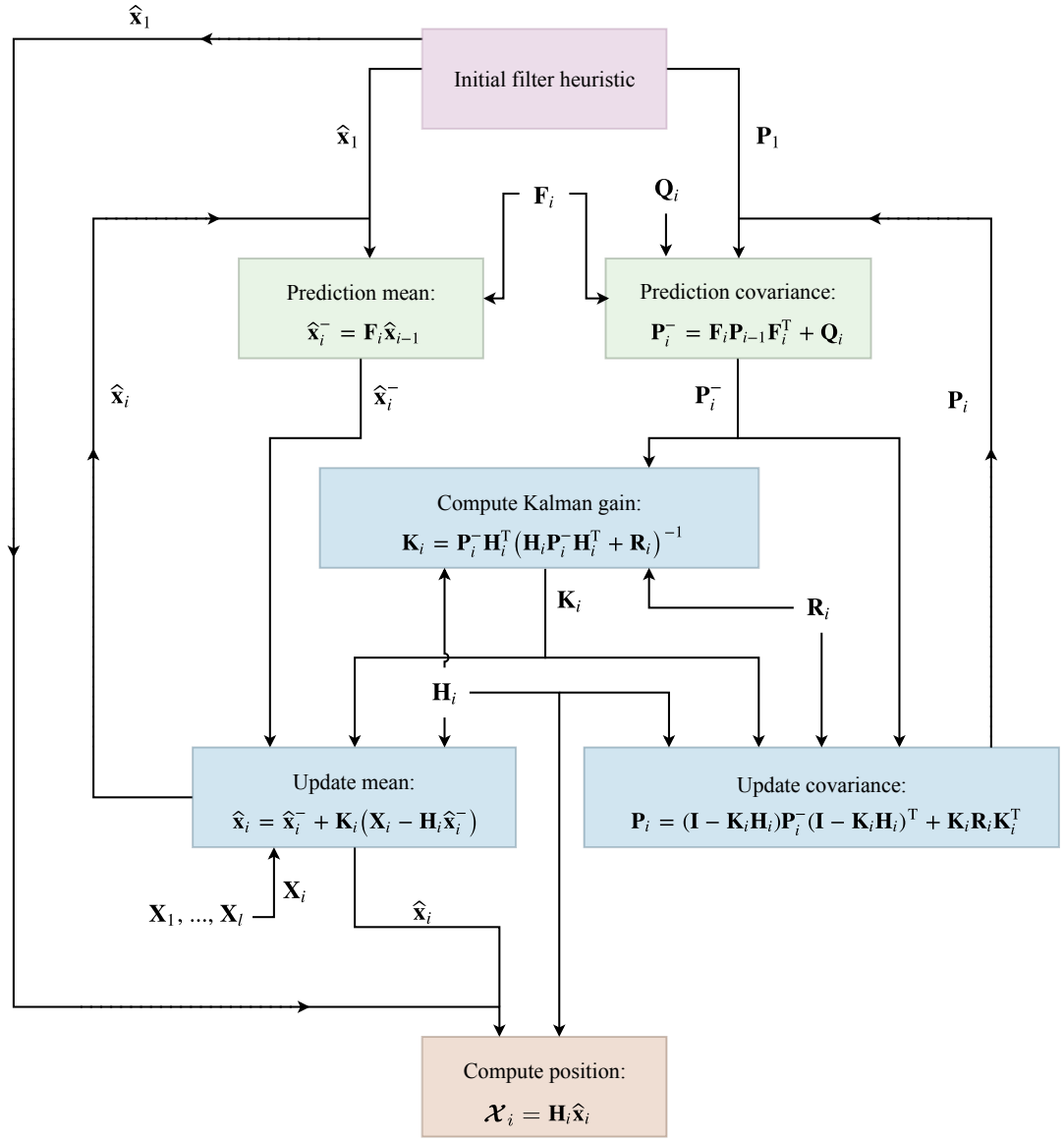


Figure 3.1: The Kalman filter procedure. Arrows represent inputs and outputs, with arrows going to the same place being the same input for that stage (only possibly differing in index). The arrows have labels clarifying the data flow along that arrow. The initial filter heuristic is only used once at the beginning of the procedure.

a measurement at one time has more uncertainty than at another time, so we define the measurement covariance matrix through the same standard deviation parameter $\sigma > 0$ for each time:

$$\mathbf{R}_i = \mathbf{R} = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}. \tag{3.37}$$

Labbe Jr. [66] also describes a way to compute the process noise matrix. By treating the process as a continuous system with similarly continuous noise (i.e. noise defined at each time, instead of at only discrete timesteps), we compute a process noise matrix by discretizing this noise matrix. This is done by integrating the continuous noise over the time intervals between discrete time steps.

The continuous noise depends on a spectral density parameter ϕ, derived from the treatment of the noise as a signal in the frequency domain using Fourier transformations. As our true state

is considered to remain constant, and is only varying due to noise, we can define the continuous noise covariance matrix $\mathbf{C}$ by only assigning values to those elements on the diagonal which vary solely due to noise:

$$\mathbf{C} = \phi\mathbf{I}. \tag{3.38}$$

For consistency in notation for later process models, we also define the continuous process matrix $\mathbf{F}'(t)$, which happens to be equal to the discrete variant $\mathbf{I}$ in this case. Using this matrix and the continuous noise matrix, we can compute the discrete process noise covariance matrix for use in the filter using the following equation:

$$\mathbf{Q}_i = \int_0^{\Delta T_i} \mathbf{F}'(t)\mathbf{C}\mathbf{F}'(t)^{\mathrm{T}} dt \tag{3.39}$$

$$= \phi\Delta T_i\mathbf{I}, \tag{3.40}$$

where $\Delta T_i = T_i - T_{i-1}$.

Constant velocity We can also model the motion through the use of the classical Newtonian kinematic equations. By also including velocities $\dot{x}, \dot{y}$, for the x- and y-coordinates respectively, we can use these equations for the process transition from x_{i-1} to x_i (and similarly for y):

$$x_i = x_{i-1} + \Delta T_i\dot{x}_{i-1}. \tag{3.41}$$

We can initialize the velocities by computing the difference between the first two observed positions, and dividing that by the difference in their time values, giving us the following mean state estimate vector

$$\hat{\mathbf{x}}_i = \begin{bmatrix} x_i \\ \dot{x}_i \\ y_i \\ \dot{y}_i \end{bmatrix}, \tag{3.42}$$

with $\begin{bmatrix} x_1 & y_1 \end{bmatrix}^{\mathrm{T}} = \mathbf{X}_1$, and $\begin{bmatrix} \dot{x}_1 & \dot{y}_1 \end{bmatrix}^{\mathrm{T}} = \frac{1}{\Delta T_2}(\mathbf{X}_2 - \mathbf{X}_1)^{\mathrm{T}}$.

The initial estimated covariance matrix is again reliant on the measurement noise covariance matrix (which is identical to the one in the previous section for all process models), but that is only true for the x and y entries. As suggested in the previous section, we set the entries for the velocities to some maximum velocity $v_{\max}$:

$$\mathbf{P}_1 = \begin{bmatrix} \sigma^2 & 0 & 0 & 0 \\ 0 & v_{\max}^2 & 0 & 0 \\ 0 & 0 & \sigma^2 & 0 \\ 0 & 0 & 0 & v_{\max}^2 \end{bmatrix}. \tag{3.43}$$

The process and measurement matrices are easily derived from Equations (3.41) and (3.42) as

$$\mathbf{F}_i = \begin{bmatrix} 1 & \Delta T_i & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta T_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{H}_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{3.44}$$

We compute the process noise covariance matrix as we did in the previous section. This time, the process matrix does actually differ based on time, so we need to define the following continuous process matrix:

$$\mathbf{F}'(t) = \begin{bmatrix} 1 & t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.45}$$

along with the continuous noise covariance matrix, which assumes that the velocities only vary because of noise:

$$\mathbf{C} = \phi \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.46}$$

We can now finally compute $\mathbf{Q}_i$ using Equation (3.39) as

$$\mathbf{Q}_i = \phi \Delta T_i \begin{bmatrix} \frac{(\Delta T_i)^2}{3} & \frac{\Delta T_i}{2} & 0 & 0 \\ \frac{\Delta T_i}{2} & 1 & 0 & 0 \\ 0 & 0 & \frac{(\Delta T_i)^2}{3} & \frac{\Delta T_i}{2} \\ 0 & 0 & \frac{\Delta T_i}{2} & 1 \end{bmatrix}. \tag{3.47}$$

Constant acceleration By also adding constant acceleration $\ddot{x}, \ddot{y}$ to the previous model, we can account for systematic changes in velocity. The process transition equations for x (which are similarly formed for y) become

$$x_i = x_{i-1} + \Delta T_i \dot{x}_{i-1} + \frac{1}{2}(\Delta T_i)^2 \ddot{x}_{i-1}, \tag{3.48}$$

$$\dot{x}_i = \dot{x}_{i-1} + \Delta T_i \ddot{x}_{i-1}, \tag{3.49}$$

with (initial) mean state estimate

$$\hat{\mathbf{x}}_i = \begin{bmatrix} x_i \\ \dot{x}_i \\ \ddot{x}_i \\ y_i \\ \dot{y}_i \\ \ddot{y}_i \end{bmatrix}, \hat{\mathbf{x}}_1 = \begin{bmatrix} x_1 \\ \dot{x}_1 \\ \ddot{x}_1 \\ y_1 \\ \dot{y}_1 \\ \ddot{y}_1 \end{bmatrix}, \tag{3.50}$$

where

$$\begin{bmatrix} \ddot{x}_1 & \ddot{y}_1 \end{bmatrix}^{\mathrm{T}} = \frac{1}{\Delta T_2} \left(\frac{1}{\Delta T_3}(\mathbf{X}_3 - \mathbf{X}_2) - \frac{1}{\Delta T_2}(\mathbf{X}_2 - \mathbf{X}_1) \right)^{\mathrm{T}}.$$

By introducing a maximum acceleration parameter $a_{\max}$, we can define the initial covariance matrix similarly to Equation (3.43):

$$\mathbf{P}_1 = \begin{bmatrix} \sigma^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & v_{\max}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{\max}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & v_{\max}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & a_{\max}^2 \end{bmatrix}. \tag{3.51}$$

The (discrete and continuous) process and measurement matrices follow intuitively from

Equations (3.48) to (3.50):

$$\mathbf{F}_i = \begin{bmatrix} 1 & \Delta T_i & \frac{(\Delta T_i)^2}{2} & 0 & 0 & 0 \\ 0 & 1 & \Delta T_i & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta T_i & \frac{(\Delta T_i)^2}{2} \\ 0 & 0 & 0 & 0 & 1 & \Delta T_i \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{F}'(t) = \begin{bmatrix} 1 & t & \frac{t^2}{2} & 0 & 0 & 0 \\ 0 & 1 & t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & t & \frac{t^2}{2} \\ 0 & 0 & 0 & 0 & 1 & t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.52}$$

$$\mathbf{H}_i = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

As $\mathbf{R}_i$ is the same for all these process models, we need to only compute $\mathbf{Q}_i$. To do this, we redefine the continuous noise covariance matrix

$$\mathbf{C} = \phi \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.53}$$

allowing us to use Equation (3.39) again to compute

$$\mathbf{Q}_i = \phi \Delta T_i \begin{bmatrix} \frac{(\Delta T_i)^4}{20} & \frac{(\Delta T_i)^3}{8} & \frac{(\Delta T_i)^2}{6} & 0 & 0 & 0 \\ \frac{(\Delta T_i)^3}{8} & \frac{(\Delta T_i)^2}{3} & \frac{\Delta T_i}{2} & 0 & 0 & 0 \\ \frac{(\Delta T_i)^2}{6} & \frac{\Delta T_i}{2} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{(\Delta T_i)^4}{20} & \frac{(\Delta T_i)^3}{8} & \frac{(\Delta T_i)^2}{6} \\ 0 & 0 & 0 & \frac{(\Delta T_i)^3}{8} & \frac{(\Delta T_i)^2}{3} & \frac{\Delta T_i}{2} \\ 0 & 0 & 0 & \frac{(\Delta T_i)^2}{6} & \frac{\Delta T_i}{2} & 1 \end{bmatrix}. \tag{3.54}$$

3.2.1.2 Kalman smoothing

Kalman filters are recursive filters, where the estimates for time i only depend on that of time $i-1$. Due to this property, the filter can have a hard time differentiating between measurements that contain a lot of noise, and actual changes to the system. If the filter were also capable of looking at future measurements or estimates, however, it could more accurately estimate the state of the system by observing whether a measurement or estimate is part of a greater overall trend.

Rauch, Striebel, and Tung [67] proposed a very simple approach to such a filter, commonly known as the RTS smoother or the Kalman smoother. It consists of two passes over the data, the first of which is identical to the ordinary Kalman filter. For each time step i, the mean estimate $\hat{\mathbf{x}}_i$ and covariance matrix estimate $\mathbf{P}_i$ are kept track of. During the second pass, smoothed estimates $\hat{\mathbf{x}}'_i$ and $\mathbf{P}'_i$ are produced by passing over the first estimates backwards, beginning at the second-to-last time step $l-1$, and letting $\hat{\mathbf{x}}'_l = \hat{\mathbf{x}}_l$ and $\mathbf{P}'_l = \mathbf{P}_l$.

This second pass also consists of a prediction and an update stage, like the basic Kalman filter. During the prediction stage at time step i, the smoother creates predictions $\hat{\mathbf{x}}^{'-}_{i+1}$ and $\mathbf{P}^{'-}_{i+1}$ of the mean and covariance matrix respectively, at time $i+1$:

$$\hat{\mathbf{x}}^{'-}_{i+1} = \mathbf{F}_{i+1}\hat{\mathbf{x}}_i \tag{3.55}$$

$$\mathbf{P}^{'-}_{i+1} = \mathbf{F}_{i+1}\mathbf{P}_i\mathbf{F}^{\mathrm{T}}_{i+1} + \mathbf{Q}_{i+1}. \tag{3.56}$$

This predicted second covariance matrix is then used alongside the original covariance matrix in the update stage to compute a second Kalman gain

$$\mathbf{K}'_i = \mathbf{P}_i \mathbf{F}_{i+1}^{\mathrm{T}} (\mathbf{P}'^{-}_{i+1})^{-1}, \tag{3.57}$$

which is then used to compute $\hat{\mathbf{x}}'_i$ and $\mathbf{P}'_i$, based on the predictions and the smoothed estimates for $i+1$:

$$\hat{\mathbf{x}}'_i = \hat{\mathbf{x}}_i + \mathbf{K}'_i \left(\hat{\mathbf{x}}'_{i+1} - \hat{\mathbf{x}}'^{-}_{i+1} \right), \tag{3.58}$$

$$\mathbf{P}'_i = \mathbf{P}_i + \mathbf{K}'_i \left(\mathbf{P}'_{i+1} - \mathbf{P}'^{-}_{i+1} \right) \mathbf{K}'^{\mathrm{T}}_i. \tag{3.59}$$

After each such cycle, we can (again) obtain the filtered positions by using the measurement matrix:

$$\boldsymbol{\mathcal{X}}_i = \mathbf{H}_i \hat{\mathbf{x}}'_i. \tag{3.60}$$

3.2.1.3 Multi-modal Kalman filter

The Kalman filter attempts to estimate the true state of a system by assuming that some process model is in fact the correct model for the observed data. This is often not the case, and it is for this reason that those using a Kalman filter need to extensively compare the performance of different process models for their data.

This process of comparing filter performance can be incorporated into the filtering process itself. Bishop and Welch [60] describe a multi-modal Kalman filter that uses multiple models $\mu_1, \ldots, \mu_r$ (of which one is correct) at the same time by running r Kalman filters in parallel, and defining the multi-modal estimates at time i to be a weighted combination of the individual estimates of each model.

This weighted combination uses weights representing the probability that a certain model μ_j is the correct model at time i. These probabilities are based on the likelihood $f(\mathbf{X}_i \mid \mu)$ that we observe a measurement $\mathbf{X}_i$, conditioned on μ being the correct model at time i:

$$f(\mathbf{X}_i \mid \mu) = \frac{1}{(2\pi)^{\frac{n_\mu}{2}} \sqrt{\det\left(\mathbf{C}_{\mu,i}\right)}} \exp\left(-\frac{1}{2} (\mathbf{X}_i - \mathbf{H}_{\mu,i} \hat{\mathbf{x}}^-_{\mu,i})^{\mathrm{T}} \mathbf{C}^{-1}_{\mu,i} (\mathbf{X}_i - \mathbf{H}_{\mu,i} \hat{\mathbf{x}}^-_{\mu,i}) \right), \tag{3.61}$$

where the subscripts with μ, i represent that vector or matrix as used in model μ at time i, n_μ is the amount of dimensions of the state vector in model μ, $\det(\cdot)$ is the determinant, and $\mathbf{C}_{\mu,i} = \mathbf{H}_{\mu,i} \mathbf{P}^-_{\mu,i} \mathbf{H}^{\mathrm{T}}_{\mu,i} + \mathbf{R}_{\mu,i}$.

With this likelihood function, after each prediction step we can compute the probability $p_j(i)$ that μ_j is the correct model at time i:

$$p_j(i) = \frac{f(\mathbf{X}_i \mid \mu_j) p_j(i-1)}{\sum_{k=1}^{r} f(\mathbf{X}_i \mid \mu_k) p_k(i-1)}. \tag{3.62}$$

As we generally have no prior belief that some model is a better fit than another, we initialize the probabilities for each model μ_j with equal initial probabilities

$$p_j(1) = \frac{1}{r}. \tag{3.63}$$

Using the probability values, we can compute the multi-modal estimates after the update steps of all the individual Kalman filters. While Bishop and Welch assume that all models have the same state vector, this is not the case for our models. Because of this, we cannot compute a weighted combination of state estimates, but instead opt to compute a weighted combination of resulting position vectors, giving

$$\boldsymbol{\mathcal{X}}_i = \sum_{j=1}^{r} p_j(i) \mathbf{H}_{\mu_j,i} \hat{\mathbf{x}}_{\mu_j,i}. \tag{3.64}$$

Dynamic multi-modal filter The probabilities as computed by Equation (3.62) generally converge to fixed values, representing the correct model having been found. As Bishop and Welch describe, this is not appropriate for systems where the choice of said correct model is changing during the running of the filter. This can be remedied by allowing the probabilities to vary each time step.

This method operates in almost the same way as the usual multi-modal filter, but replaces the probability in Equation (3.62) by

$$p_j(i) = \frac{f(\mathbf{X}_i \mid \mu_j)}{\sum_{k=1}^{r} f(\mathbf{X}_i \mid \mu_k)}. \tag{3.65}$$

CHAPTER 4

Experiments

In order to compare the filtering methods, we apply them to experimentally obtained RSSI measurements. For each method, we analyse its performance on these measurements, along with the sensitivity in its parameters. The experimental setup used to obtain the measurements along with the methods used to analyze the filters will be described in this chapter.

4.1 Experimental setup

We analyze the filtering methods based on measurements obtained from previously done experiments on the 8^{th} of February 2018. These experiments were not performed with the same goal as this project, but are sufficient for the goals of this thesis.

4.1.1 The ArenA

The Johan Cruyff ArenA (formerly and colloquially known as the Amsterdam ArenA) is the largest stadium in the Netherlands, with dimensions $235 \times 180 \times 77$ metres. Its maximum capacity is between $35,000$ and $68,000$ people, depending on the event taking place. There is a total of 618 AP's throughout the ArenA, of which we know the exact locations of 591. Figure 4.1 visualizes all these known AP's, with coordinates expressed in metres with respect to the centre of the ground floor as the origin.

4.1.2 Path walking

The experiment took place on the fourth floor of an empty ArenA. Two testers walked around the floor with a mobile testing device for a period of time approximately 30 minutes in duration, taking note of their location within the ArenA at specific times. The path they walked as shown in Figure 4.2, consisted mostly of walking through the corridors. The testers also stood still by balconies, stood between bleachers, and walked through parts of the bleachers during the final part of the experiment.

4.1.3 Data collection

The mobile device used for testing was an Intel Edison module with a built-in wireless adapter capable of packet injection. The Edison sent out probe requests at a constant frequency using the 2.4 and 5 GHz frequency bands. These probe requests were picked up by the existing AP's in the ArenA. The AP's, off-the-shelf Huawei models, monitored the requests (producing RSSI's) at a constant frequency of approximately 1 Hz, and did this each time for a duration of time taking roughly 60 milliseconds.

At an unknown, constant frequency, each AP compiled and sent out a report of the requests, as a UDP datagram consisting of an identifier for that AP, the MAC addresses of the request sources and the RSSI's. The reports were sent to a central server run by KPMG, which stored

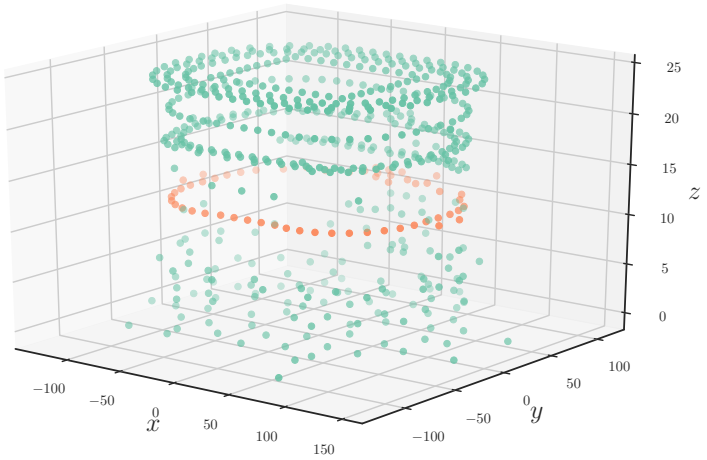


Figure 4.1: The distribution of AP's throughout the ArenA. AP's on the fourth floor are colored orange.

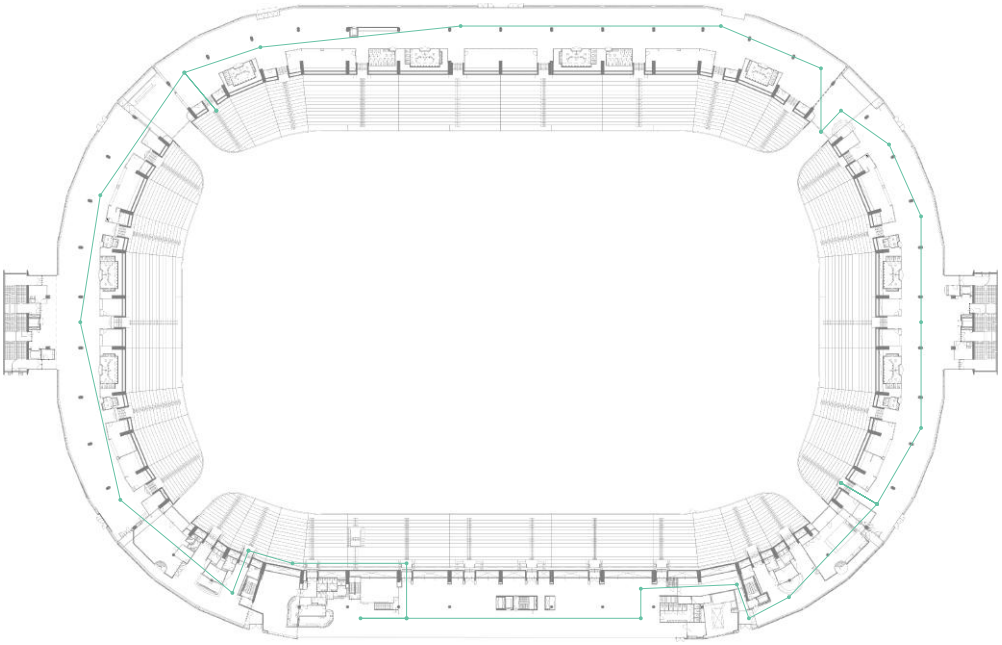


Figure 4.2: The path walked through the fourth floor in the experiment.

them locally in a relational database for the experiment. KPMG filtered out these measurements containing the MAC address of our testing device, and made those specific measurements available to us.

4.2 Performance comparison

In order to compare the performance of the filtering methods, we need to define a performance measure based on the accuracy of the positions they produce. Let $\boldsymbol{\mathcal{X}}_1^\mu, \ldots, \boldsymbol{\mathcal{X}}_l^\mu$ be the positions produced by some filtering method (also including the plain position reconstruction described in Chapter 2) μ, with corresponding timestamps $T_1, \ldots, T_l$. Given a 'ground truth' sequence of positions $\boldsymbol{\mathfrak{X}}_1, \ldots, \boldsymbol{\mathfrak{X}}_l$, we define our accuracy measure for μ as the mean Euclidean distance MED between the positions of these two sequences:

$$\text{MED}(\mu) = \frac{1}{l} \sum_{i=1}^{l} \| \boldsymbol{\mathfrak{X}}_i - \boldsymbol{\mathcal{X}}_i^\mu \|. \tag{4.1}$$

This formulation relies on the assumption that we have ground truth positions at the same times as the corresponding filtered positions. Since we generally do not know where the subject was exactly at each time T_i, we perform linear interpolation between ground truth positions at known times, assuming constant velocity between successive positions, which fits our experimental setup.

4.2.1 Parameter optimization

All used methods rely on some number of parameters which influence that method's accuracy. In order to compare the accuracy of different methods, we choose to perform this comparison when they are at their highest performance, by finding the parameters for each method such that the MED is minimal. We compute the optimal parameters by running a bounded limited-memory Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [68] from a multitude of initial parameter values, and using the parameters that give a minimal MED.

This method is first applied to find the optimal path loss exponent γ for the plain position reconstruction. All subsequent filtering methods are then used on the positions obtained by the position reconstruction using this optimal γ.

Table 4.1 contains a list of all methods and their to-be optimized parameters, along with constants for some. The shorthands introduced in this table will be used interchangeably throughout this thesis. [7]

4.2.2 Statistical analysis

For each method μ in Table 4.1, we report both MED(μ), along with violin plots [70] for the MED values of bootstrapped samples of the individual Euclidean distances $\| \boldsymbol{\mathfrak{X}}_i - \boldsymbol{\mathcal{X}}_i^\mu \|$. All of these are using the method with optimal parameters.

To test whether one method μ_1 produces a statistically significantly different mean Euclidean distance than another method μ_2, we use a non-parametric confidence interval approach. This approach involves drawing from a bootstrapped distribution under the null hypothesis that the mean difference between the Euclidean distances is zero.

Let $\text{ED}(\mu, i) = \| \boldsymbol{\mathfrak{X}}_i - \boldsymbol{\mathcal{X}}_i^\mu \|$ be the Euclidean distance w.r.t. the ground truth for the i-th position produced by method μ. We compute the differences in distance $\delta_1, \ldots, \delta_l$ between the two methods as $\delta_i = \text{ED}(\mu_1, i) - \text{ED}(\mu_2, i)$, and denote their mean by $\overline{\delta} = \frac{1}{l} \sum_{i=1}^{l} \delta_i$.

As the null hypothesis states that the true mean of the distribution that these differences were drawn from is equal to zero, we change the mean of our observed differences to zero, obtaining shifted differences $\delta_1^*, \ldots, \delta_l^*$ with $\delta_i^* = \delta_i - \overline{\delta}$.

We approximate the underlying distribution by now obtaining bootstrapped samples from these shifted differences. This means we draw N[8] sequences of length l with replacement from $\delta_1^*, \ldots, \delta_l^*$ and compute the means of these sequences, obtaining N means $\overline{\delta}_1^*, \ldots, \overline{\delta}_N^*$. These means approximate a distribution of mean Euclidean distance differences between μ_1 and μ_2 under the null hypothesis.

[7] The values for the maximum velocity and acceleration were based on Gómez, Marquina, and Gómez [69].

[8] We will report the amount of samples used for any bootstrapping method used throughout this thesis in the captions of the relevant figures.

Table 4.1: Methods used to produce positions (including both the plain position reconstruction scheme, as well as the filtering methods), alongside shorthands for ease of notation, and parameters/constants.

Method	Shorthand	Parameters	Constants
Plain position reconstruction	base	γ	$w = 1.5$
Exponential filtering	expon	α, heuristic	N/A
Irregular time exponential filtering	irexp	τ, heuristic	N/A
Double exponential filtering	doubexp	α, heuristic	N/A
Irregular time double exponential filtering	doubirexp	τ, heuristic	N/A
Gaussian filter	gauss	σ	N/A
Irregular time Gaussian filter	irgauss	σ	N/A
Savitzky-Golay filter	sav	m, κ, heuristic	N/A
Irregular time Savitzky-Golay filter	irsav	m, κ, heuristic	N/A
Median filter	median	m	N/A
Gaussian weights median filter	gaussmed	m, σ	N/A
Shifted time differences median filter	shiftmed	m	N/A
Brownian motion Kalman filter	brown	σ, ϕ	N/A
Constant velocity Kalman filter	vel	σ, ϕ	$v_{\max} = 12.0$
Constant acceleration Kalman filter	acc	σ, ϕ	$v_{\max} = 12.0, a_{\max} = 10.0$
Brownian motion Kalman smoothing	smbrown	σ, ϕ	N/A
Constant velocity Kalman smoothing	smvel	σ, ϕ	$v_{\max} = 12.0$
Constant acceleration Kalman smoothing	smacc	σ, ϕ	$v_{\max} = 12.0, a_{\max} = 10.0$
Multimodal Kalman filter	mult	$\sigma, \phi_{\text{brown}}, \phi_{\text{vel}}, \phi_{\text{acc}}$	$v_{\max} = 12.0, a_{\max} = 10.0$
Dynamic multimodal Kalman filter	dynmult	$\sigma, \phi_{\text{brown}}, \phi_{\text{vel}}, \phi_{\text{acc}}$	$v_{\max} = 12.0, a_{\max} = 10.0$

Using this distribution, we compute a bootstrapped 95% confidence interval

$$\left[P_{0.025}(\overline{\delta}_1^*, \ldots, \overline{\delta}_N^*),\ P_{0.975}(\overline{\delta}_1^*, \ldots, \overline{\delta}_N^*)\right], \tag{4.2}$$

where $P_\alpha(\cdot)$ is the (100α)-th percentile of the supplied sequence. If $\overline{\delta}$ lies outside this confidence interval, we can reject the null hypothesis at significance level 0.05.

We choose to also report p-values using this bootstrapped distribution. The approximate p-value is computed as

$$p = \frac{\#\left\{i \;\middle|\; 1 \leqslant i \leqslant N \text{ and } \left(-|\overline{\delta}| \geqslant \overline{\delta}_i^* \text{ or } \overline{\delta}_i^* \geqslant |\overline{\delta}|\right)\right\}}{N}, \tag{4.3}$$

where $\#\{\cdot\}$ is the set cardinality.

4.2.3 Sensitivity analysis

Besides looking at optimal performance, we are also interested in the influence of different parameters on the accuracy of a method. Randomly sampling all parameters and producing scatterplots of the accuracy plotted against the values for a single parameter is a possible way to qualitatively judge the sensitivity of a method in said parameter. The same can be said of plots of the accuracy against all possible values of one or two parameters.

The downside of these approaches is that they do not take into account any interactions between parameters, which may downplay the influence of parameters that the method is sensitive to. The comparison of parameters also becomes difficult when methods display highly nonlinear, periodic or otherwise irregular behavior. Numerical global sensitivity analysis offers a measure for the comparison of parameters that does not suffer from these issues.

Sobol [71] proposes a variance-based global sensitvity analysis method that decomposes the variance in the output of a model (the accuracy of the method, in our case), and estimates the contribution of each parameter at any order of interaction to this variance. This contribution is expressed through sensitivity indices that express proportions of total variance, which are

computed by sampling over the parameter space and analyzing the output for each parameter vector.

In this thesis, we focus on so-called first-order and total-order indices for each parameter, which respectively represent the expected proportion of the total variance that would be explained if that parameter is fixed, and the expected proportion of the variance that would remain if all parameters but the one in question were fixed. The first-order indices only contain information about the parameter by itself, while total-order indices contain information about all interactions of that parameter with other parameters.

We report the indices for methods with more than one parameter, along with their accuracy in the form of a 95% confidence interval, obtained using a normal approximation applied to some number of resamples of the MED values computed for the sampled parameters from the parameter space. The usual boostrapping method is not suited to sensitivity indices, as their computation is already expensive.

CHAPTER 5

Results

5.1 Optimal performance comparison

The optimal parameters and corresponding MED (mean Euclidean distance between position estimates and actual positions) values for all methods are listed in Table 5.1, with corresponding violin plots in Figure 5.1. Plots of the MED against the parameters are included in Appendix A, and visualizations of the optimal MED paths are in Appendix B.

Table 5.1: Optimal parameters and MED for each method.

Method	Optimal parameters	MED
base	$\gamma = 0.3$	19.15
doubexp	$\alpha = 0.193,$ heuristic = (b.)	17.31
vel	$\sigma = 1.28 \times 10^3, \phi = 303$	17.11
expon	$\alpha = 0.383,$ heuristic = (b.)	17.06
doubirexp	$\tau = 15.2,$ heuristic = (b.)	17.05
acc	$\sigma = 0.767, \phi = 1.53 \times 10^{-8}$	17.01
brown	$\sigma = 88.6, \phi = 657$	16.87
mult	$\sigma = 25.7, \phi_{\text{brown}} = 55.4, \phi_{\text{vel}} = 51.7, \phi_{\text{acc}} = 51.7$	16.87
dynmult	$\sigma = 251, \phi_{\text{brown}} = 4.58 \times 10^3, \phi_{\text{vel}} = 256, \phi_{\text{acc}} = 1.24 \times 10^3$	16.83
irexp	$\tau = 6.57,$ heuristic = (b.)	16.61
smacc	$\sigma = 7.50 \times 10^3, \phi = 0.151$	13.44
smbrown	$\sigma = 79.2, \phi = 26.3$	13.39
smvel	$\sigma = 9.35 \times 10^3, \phi = 254$	13.38
gauss	$\sigma = 11.6$	13.35
irgauss	$\sigma = 35.3$	13.35
sav	$m = 44, \kappa = 2,$ heuristic = (c.)	12.94
irsav	$m = 44, \kappa = 2,$ heuristic = (c.)	12.72
shiftmed	$m = 51$	12.11
median	$m = 32$	12.03
gaussmed	$m = 33, \sigma = 215$	12.00

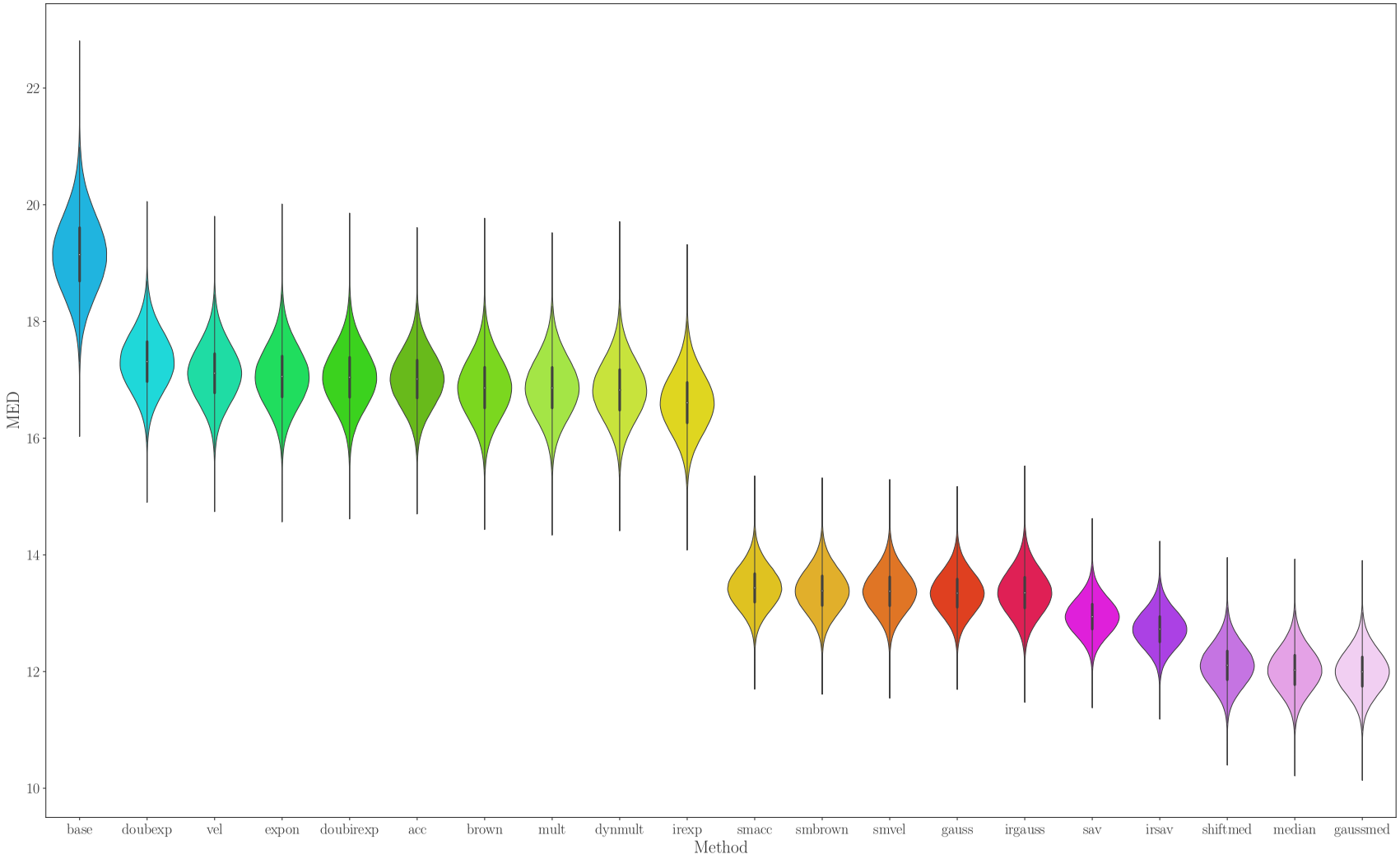


Figure 5.1: Violin plots of the bootstrapped MED values for each method. Bootstrapping was performed with $N = 1 \times 10^6$.

5.2 Hypothesis tests

Figure 5.2 contains the results of the hypothesis tests performed, showing which methods produce a statistically significantly different MED than other methods. The confidence intervals computed for these hypothesis tests are displayed in Appendix C.

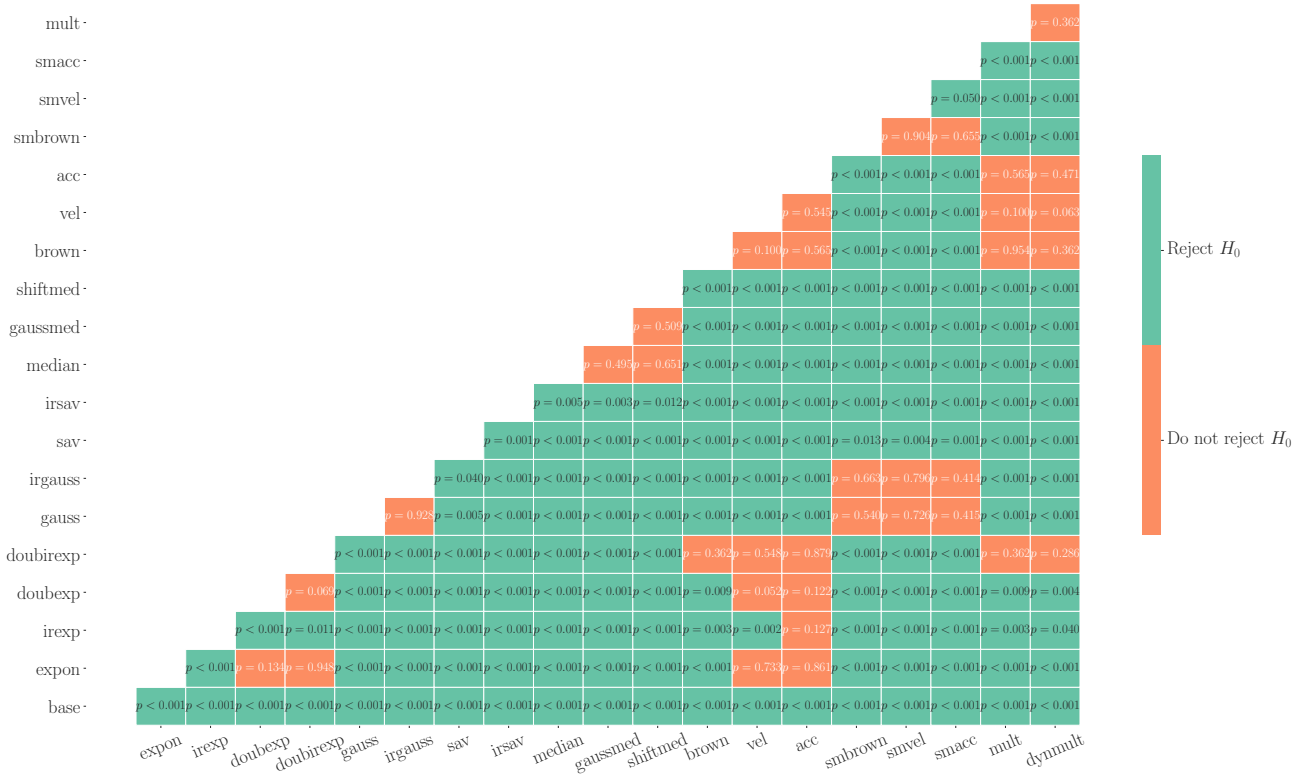


Figure 5.2: Results of the testing of the significance of the difference in MED between the method on the row and the method on the column, computed using the confidence interval approach described in Section 4.2.2, alongside approximated p-values. Bootstrapping was performed with $N = 1 \times 10^7$.

5.3 Sensitivity indices

Figures 5.3 to 5.8 contain the first-order and total-order sensitivity indices for the parameters of methods with more than two parameters, along with their accuracy expressed as a 95% confidence interval through error bars. All indices were computed by sampling 5×10^5 points from their parameter space, and resampling these points 1×10^3 times to compute the confidence intervals. The lower bounds used for sampling were either the theoretical lower bound (such as $m = 1$ for methods using window sizes), or a value close to zero. Upper bounds were either the theoretical upper bound (a value close to one for the smoothing parameter α in exponential filtering), or a value large enough to include the optimal value somewhere approximately in the middle of the bounds.

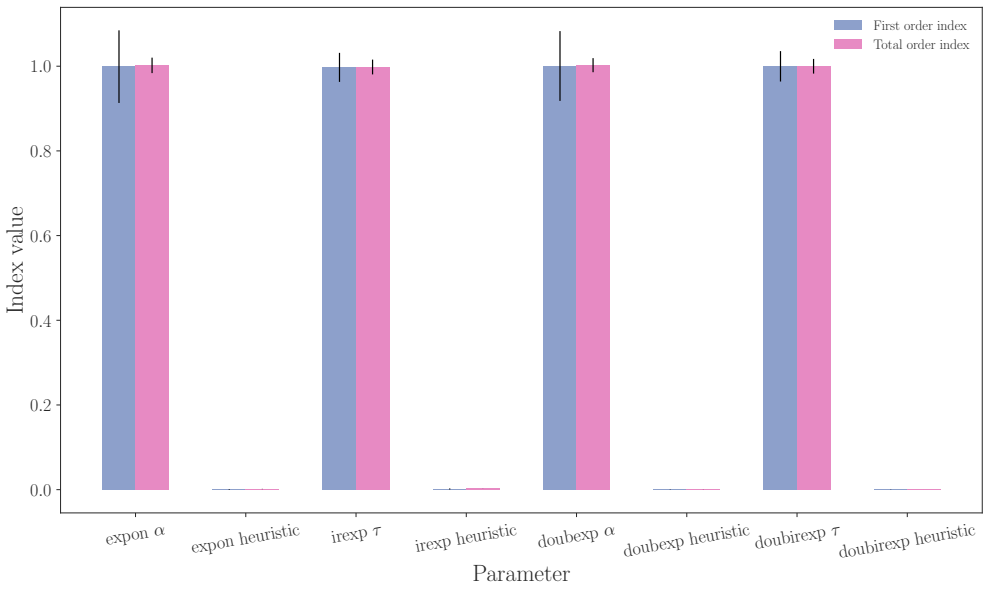


Figure 5.3: Sensitivity indices for the exponential filtering methods.

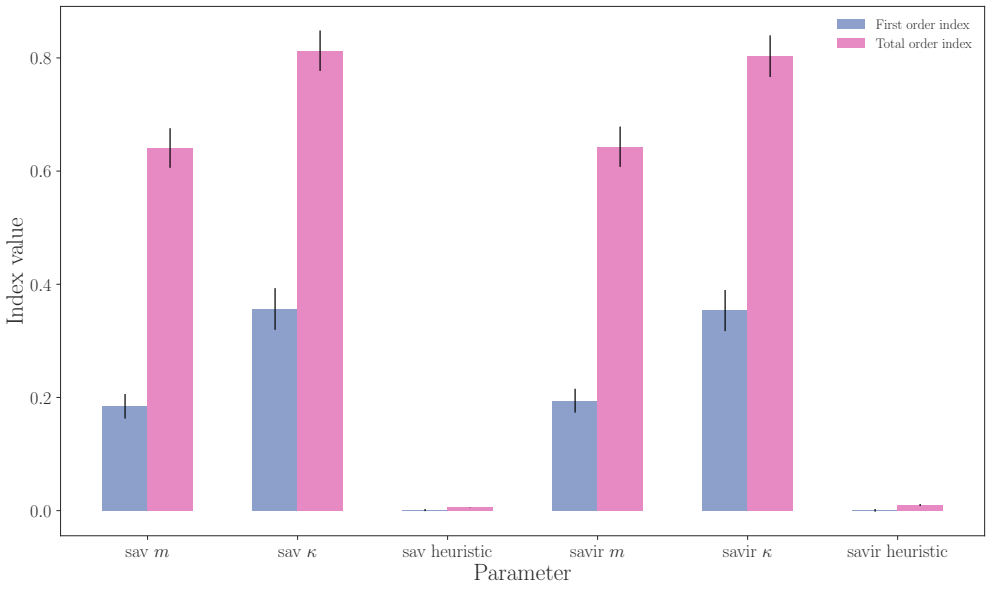


Figure 5.4: Sensitivity indices for the Savitzky-Golay methods.

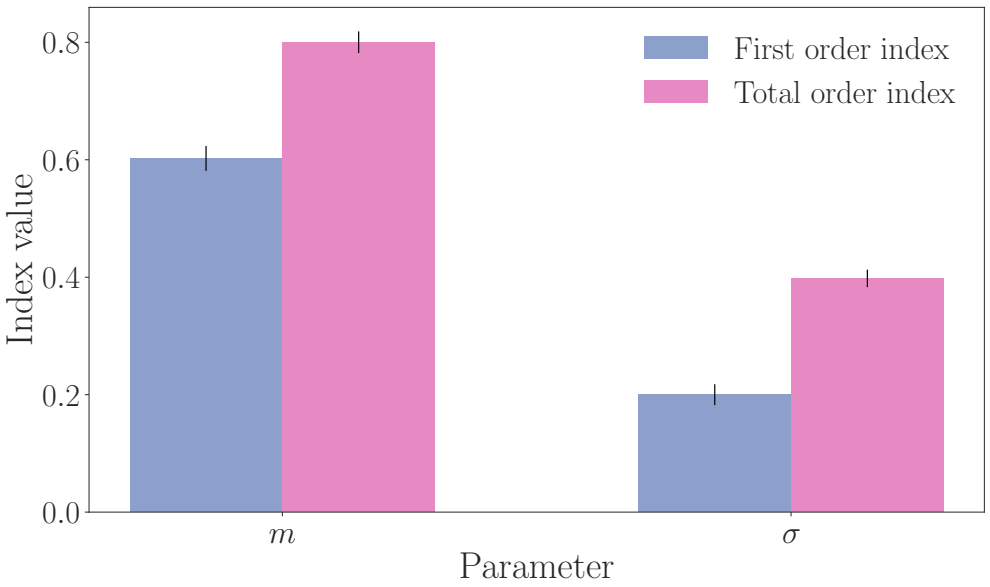


Figure 5.5: Sensitivity indices for the weighted median filter with Gaussian weights.

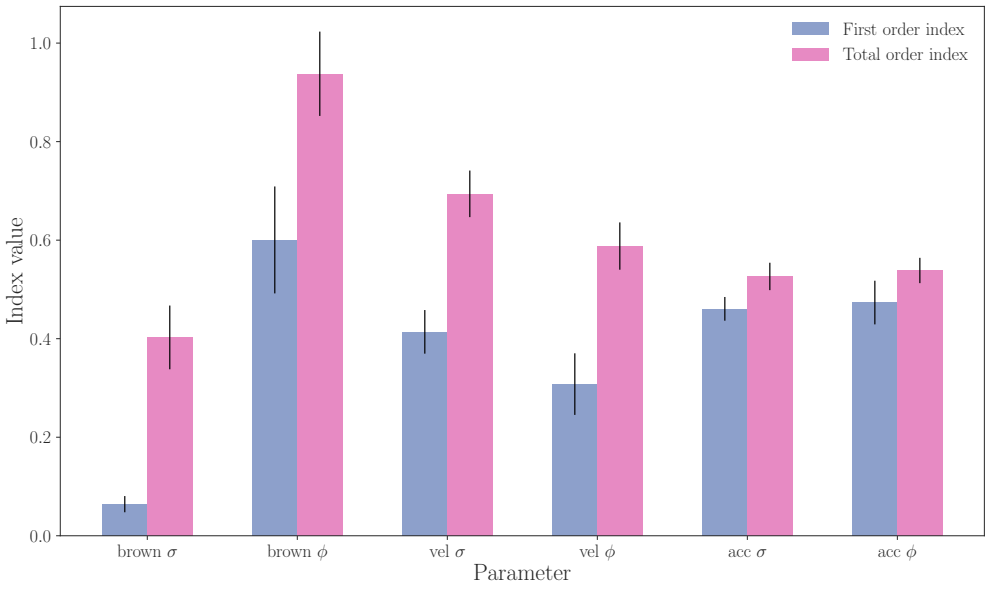


Figure 5.6: Sensitivity indices for the basic Kalman filters.

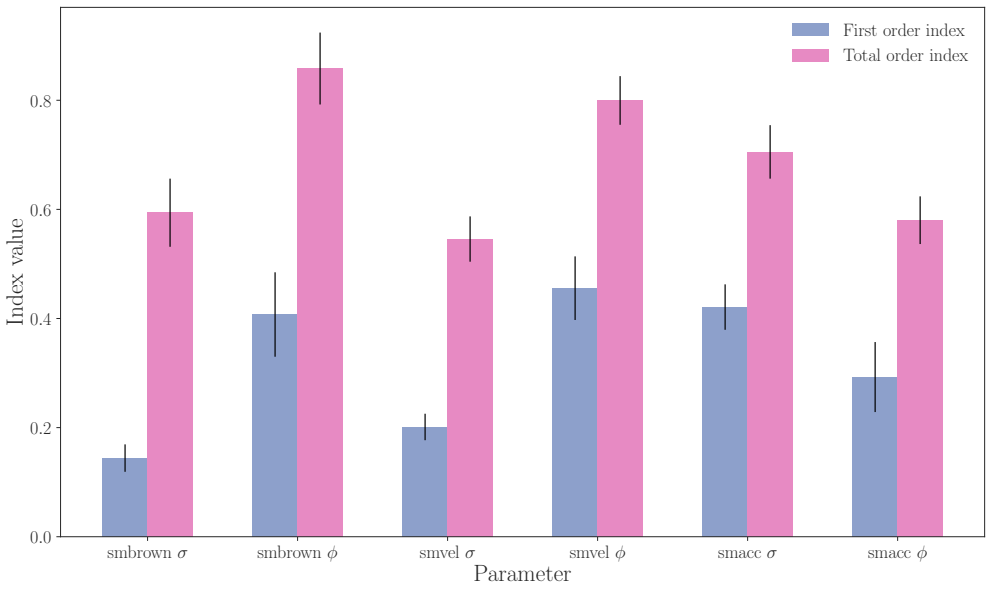


Figure 5.7: Sensitivity indices for the Kalman smoothing methods.

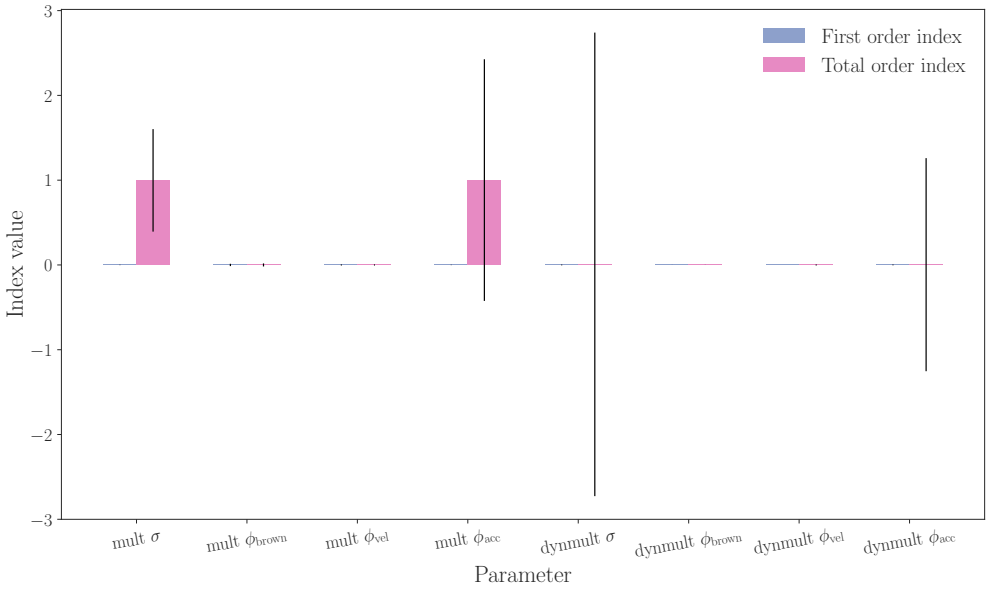


Figure 5.8: Sensitivity indices for the multimodal Kalman filters.

CHAPTER 6

Discussion

In order to improve upon the accuracy of WiFi-based position reconstruction, we have compared several filtering methods. From Table 5.1 and Figure 5.1 we can infer that all methods used do increase accuracy compared to that of plain position reconstruction, but that a large amount of methods do not produce a significantly different MED than others, roughly dividing the methods into three tiers based on their performance. Exponential filtering methods and both the basic and multimodal Kalman filters make up the least accurate tier, and all only improve the accuracy of the plain method by a few metres. Kalman smoothing, Gaussian filters, and the Savitzky-Golay filters make up the middle tier, which already improves on the previous tier by three to four metres. Finally, the top tier consists of the median filters, adding approximately another metre of accuracy.

The results of the hypothesis tests in Figure 5.2 support these observations. All filtering methods differ statistically significantly in accuracy compared to the plain position reconstruction, with $p < 0.001$. Within the aforementioned tiers, most filtering methods display no statistically significant difference in accuracy. Additionally, note that the paths produced by methods within the same tier are visually quite similar as well, as seen in Appendix B, which supports the lack of significant difference.

Of particular interest is the complexity of the methods with high performance. The three top types of filters (median filters, Savitzky-Golay filters, and Gaussian filters) are all simple in concept and procedure. They outperform complex methods like multimodal Kalman filters, which combine several Kalman filters yet fail to improve significantly upon them, and Kalman smoothing, which requires two passes in order to properly use all measurements.

We also observe that while irregular time variations are mostly statistically significantly different than their base methods, except for irregular time double exponential filtering and the Gaussian filter, the increase in accuracy they provide is always of the order of decimetres. Combined with their oftentimes increased computational requirements, they do not appear to be as viable a choice as the base methods.

From the computed sensitivity indices in Figures 5.3 and 5.4, we observe that the choice of heuristic for initial filtered position is almost completely inconsequential, and can safely be ignored and set to the optimal heuristics reported in Table 5.1. We can also see that the window size and degree in Figure 5.4 display high amounts of interaction, with total-order indices at least twice as high as the first-order indices. The indices for the degree κ are higher than those of the window size, whilst the range of values usually used for the degree is much smaller than that of the window size [54]. This means finding a good degree for the filter is both more important and easier than selecting a window size.

The opposite is the case with the weighted median filter with Gaussian weights in Figure 5.5. The window size is approximately twice as important than the other parameter, the scale of the Gaussian. This agrees with our knowledge of the filter's workings, as large window sizes can be made to function similarly to smaller sizes by choosing smaller standard deviations. This causes the weights of values that are further away to be assigned weights nearing zero.

The indices of the Kalman filters in Figure 5.6 do not differ much between σ and ϕ, with

slightly higher values for σ, except for those of the Brownian motion Kalman filter. Table 5.1 suggests that the Brownian model is the most appropriate model for our data out of all process models we compared, which explains the increased importance of a well-tuned process model, compared to measurements. This likely has to do with the nature of the experimental walk, which contains several instances of standing still for extended periods of time. Such motion is better handled by a Brownian process model than by one assuming constant motion.

This increased importance of ϕ compared to σ is also seen across the Kalman smoothing methods in Figure 5.7. These smoothing methods all make optimal use of the information in the measurements, which again explains the increased importance of the process model. The exception in this case is constant acceleration Kalman smoothing, where the opposite is the case. We assume this stems from the inadequacy of such models for walks like ours, as is the case with the Brownian motion Kalman filter.

The sensitivity indices of the multimodal filters in Figure 5.8 display drastically different behaviour than those of the other methods discussed so far. All first-order indices appear to be approximately zero, with high confidence in these values. The total-order indices are also zero for most parameters, except for the σ and ϕ_{acc} of the basic multimodal Kalman filter. It appears that ϕ_{brown} and ϕ_{vel} are of virtually no importance compared to these two parameters. We do notice that this might not be the case for ϕ_{acc}, as its confidence interval crosses zero. [9] Trusting the analysis, it would also appear that none of the parameters of the dynamic multimodal Kalman filter are of any importance, as all total-order indices are zero, some with large confidence intervals. We believe these results are due to a very high degree of interactions between the parameters. This means we requires significantly larger sample sizes to reliably compute these indices.

6.1 Limitations and future work

It should be noted that while deemed sufficient, the used experimental setup is far from ideal, due to the ground truth relying on very subjective indications of the subject's positions at certain times. We have actually ran more recent experiments where we have much more confidence in the constructed ground truth, by walking paths from AP to AP, which we can precisely pinpoint on a floor map of the ArenA. The measurements from these experiments were not available yet to be used at this thesis' time of writing, due to unfinalized legal agreements. While such measurements would be preferred for similar projects in the future, they do have the downside of the walk not representing realistic human walking behaviour. Care should be taken to perform experiments where such behaviour is maintained, while simultaneously constructing a reliable ground truth.

The conditions of the setting of the walk are also problematic. While our goal was to compare the effect of the filtering methods in a realistic setting, the experiment performed took place in an empty ArenA. Performing walks throughout the ArenA during a concert or a football match with tens of thousands of mobile devices communicating with the AP's would give much more conclusive results about the filters' performance in practice.

Also of interest is the optimal path loss exponent $\gamma = 0.3$ we found for our data, which is a value that is usually not encountered in literature on path loss modelling. In such literature, the use of environments with a high amount of signal interference and refraction results in $\gamma > 2$. We believe the unusual value most likely has to do with the quality of the ground truth data it has been optimized with, as said ground truth has been constructed in a way prone to human error.

Unfortunately, due to computational limitations, the sensitivity analysis was performed with an initial sample size of 5×10^5, with 1×10^3 resamples for the confidence interval estimation. While adequate for simple filters with a small amount of parameters, this seems to be too small for complex, high-interaction methods such as the multimodal filters. Similar works should take this into account, and perform such analyses using greater computational resources, or opt for other, more efficient global sensitivity analysis methods than Sobol's method.

[9] While individual indices can only lie between zero and one, the confidence intervals might cross these values due to the normal approximation used.

While not actual limitations for the scope of our research, there are interesting directions for future work to proceed in. While our focus was on the comparison of the filtering methods on positions obtained from simple WiFi based position reconstruction using trilateration, there is no need to limit similar research to this approach. Studies comparing these filtering methods should also focus on more costly methods such as fingerprinting and other approaches to position reconstruction.

6.2 Conclusion

In this thesis, we aimed to thoroughly compare and analyze several filtering methods and their influence on the accuracy of low-cost WiFi-based position reconstruction. All filtering methods compared improve upon accuracy compared to the plain position reconstruction method, netting a maximum increase in accuracy of approximately seven metres. We also explored irregular time variations on the methods, but found no significant improvement compared to their base methods.

The median filters, which are the best performing methods, have only one or two parameters, and their procedures rely on a single pass of the data using everyday statistics, keeping in line with the aim of maintaining ease of use. They outperform complex methods like multimodal Kalman filters, which combine several Kalman filters yet fail to improve significantly upon them, and Kalman smoothing, which requires two passes in order to properly use all measurements. Other high performers include the Savitzky-Golay and Gaussian filters, both types of moving averages.

APPENDIX A

Parameter plots

Figures A.1 to A.10 plot the MED against the parameter values for each filtering method (including the plain position reconstruction). When doing so would improve legibility, we instead plot the logarithm of the MED

Except for plots where the filtering method's parameters have a domain with fixed bounds, all plots use parameter values in the vicinity of the optimal value. For the two filtering methods with four parameters each, we plot the MED against each parameter, setting the other three parameters to their optimal values.

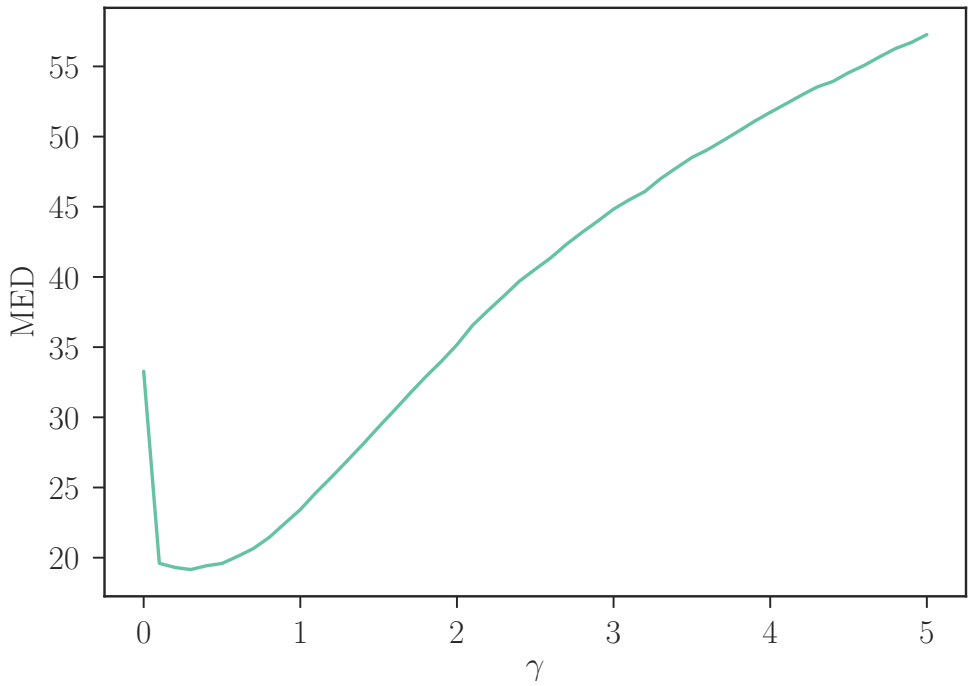
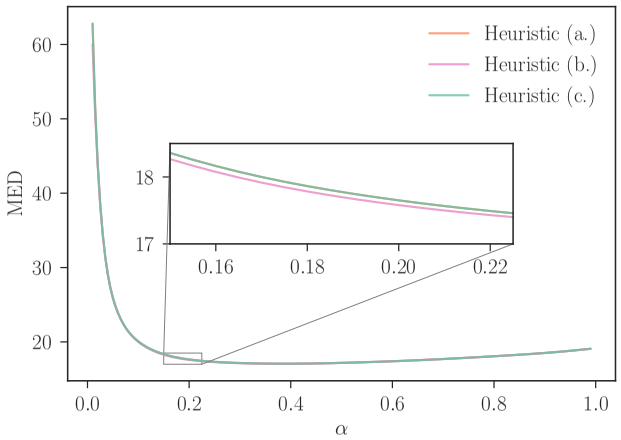
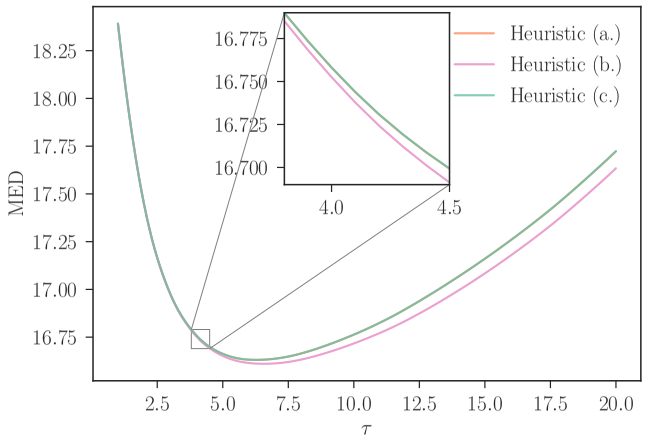


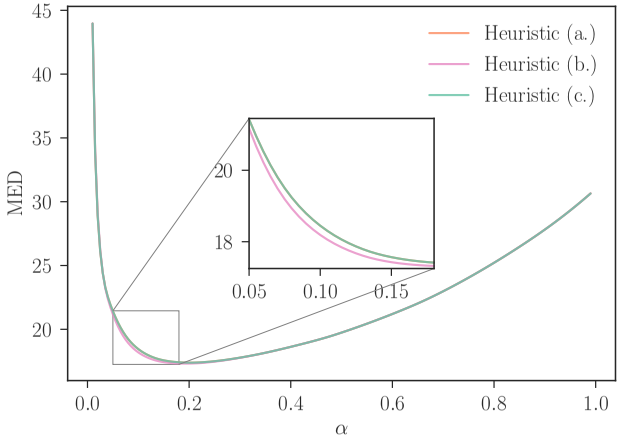
Figure A.1: MED plotted against the path loss exponent for plain position reconstruction.



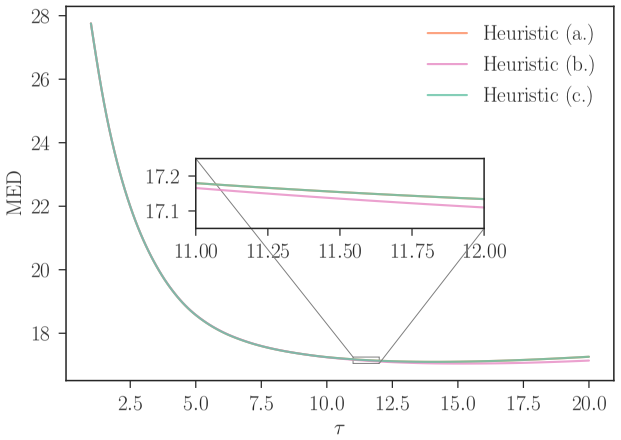
(a) Basic exponential filtering



(b) Irregular time exponential filtering

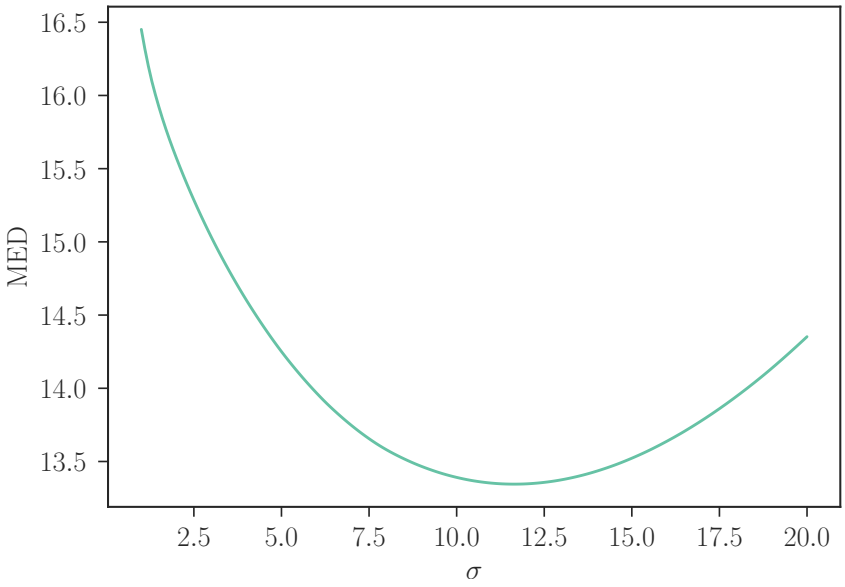


(c) Double exponential filtering

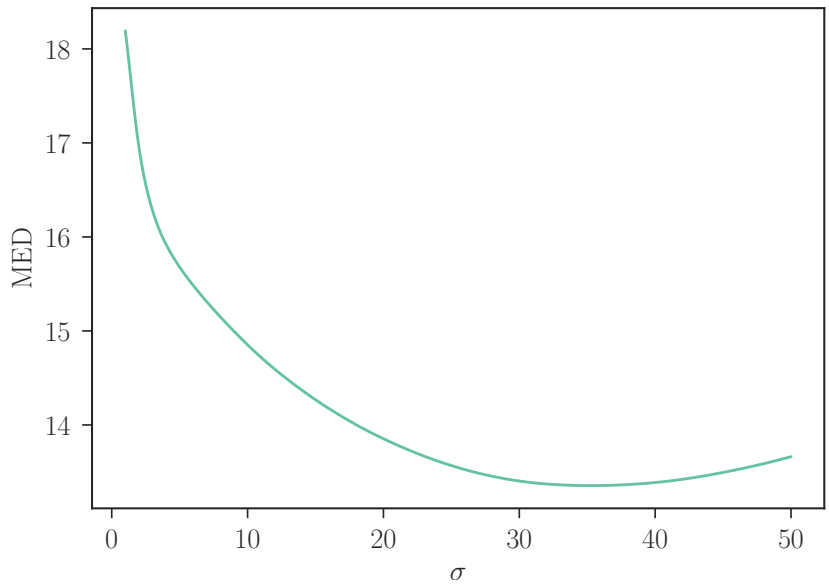


(d) Irregular time double exponential filtering

Figure A.2: MED plotted against the smoothing parameters and time window durations of the exponential filtering methods.

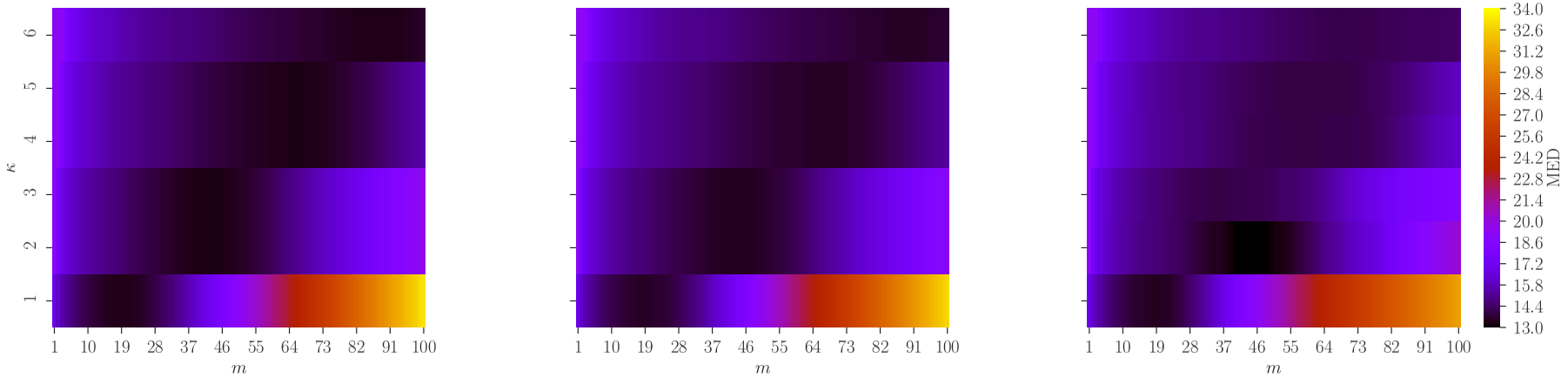


(a) Basic Gaussian filter



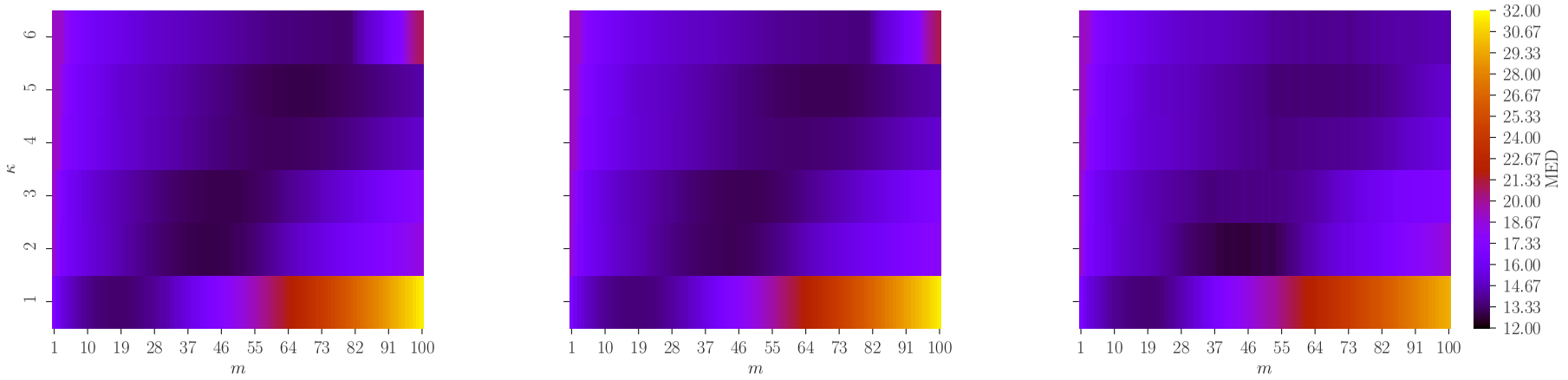
(b) Irregular time Gaussian filter

Figure A.3: MED plotted against the scale of the Gaussian filters.



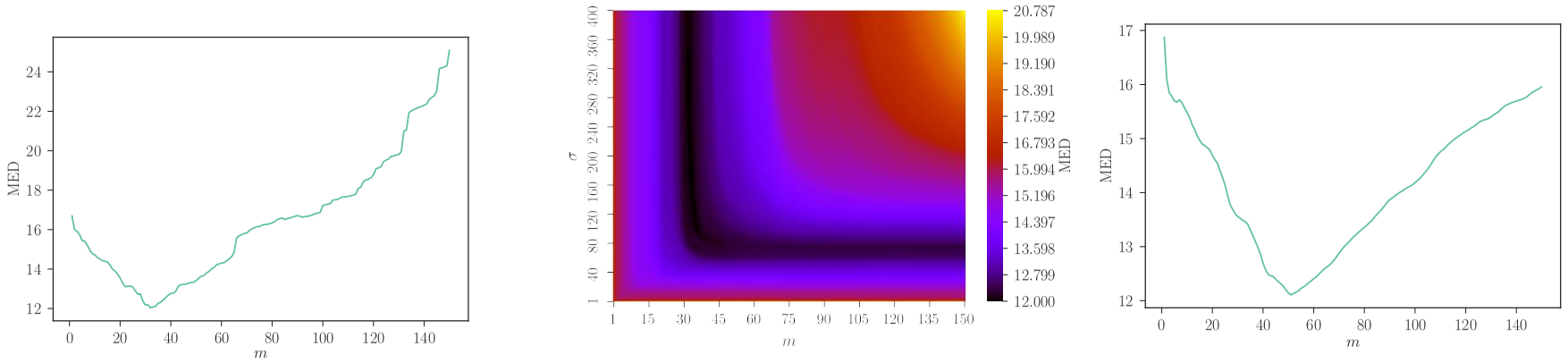
(a) Heuristic (a.). (b) Heuristic (b.). (c) Heuristic (c.).

Figure A.4: MED plotted against the window length and the degree for the basic Savitzky-Golay method.



(a) Heuristic (a.). (b) Heuristic (b.). (c) Heuristic (c.).

Figure A.5: MED plotted against the window length and degree for the irregular time Savitzky-Golay method.

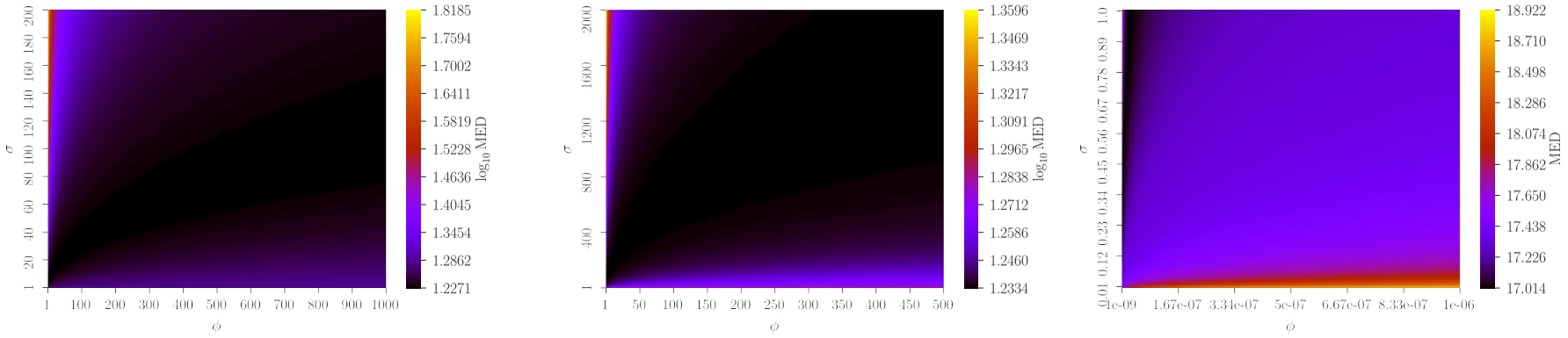


(a) Basic median filter

(b) Weighted median filter with Gaussian weights

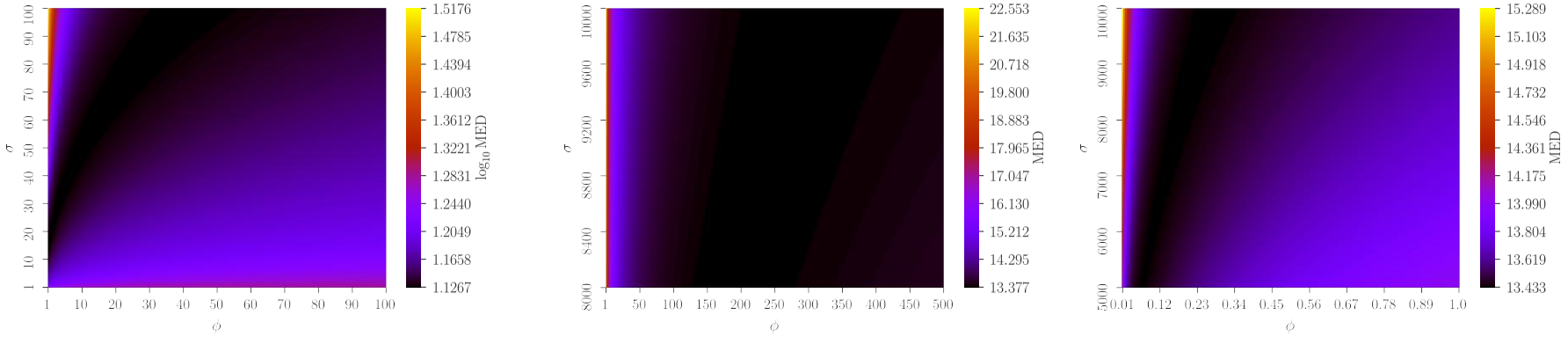
(c) Weighted median filter with shifted time differences

Figure A.6: MED plotted against the window length for the median filters, along with the scale for the weighted filter with Gaussian weights.



(a) Brownian motion (b) Constant velocity (c) Constant acceleration

Figure A.7: MED plotted against the noise spectral density and standard deviation for the basic Kalman filter.

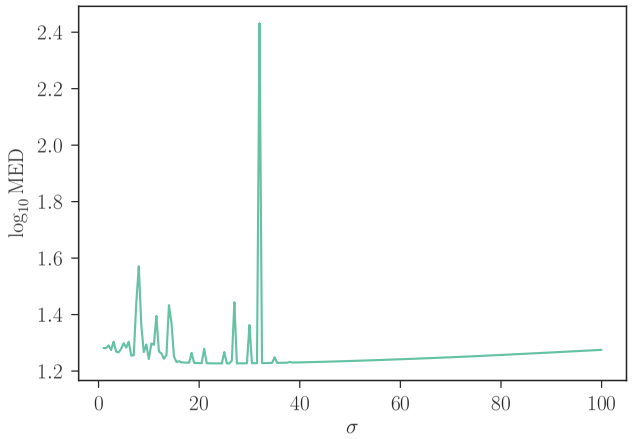


(a) Brownian motion

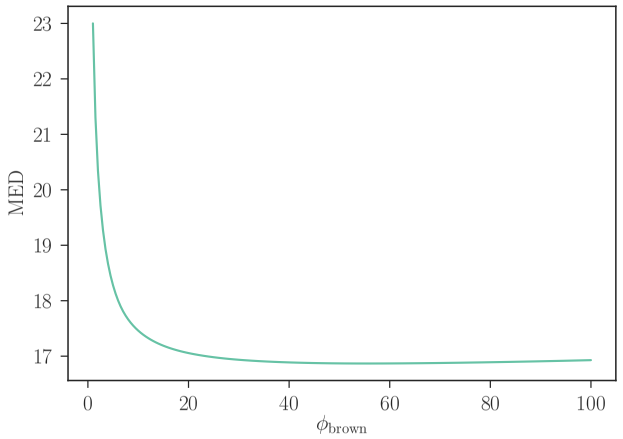
(b) Constant velocity

(c) Constant acceleration

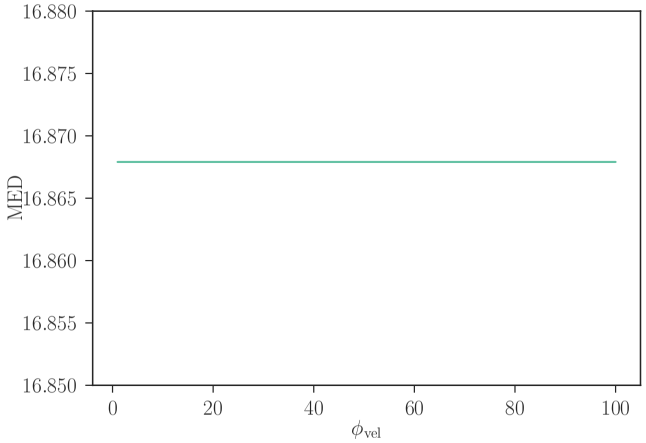
Figure A.8: MED plotted against the noise spectral density and standard deviation for the Kalman smoothing methods.



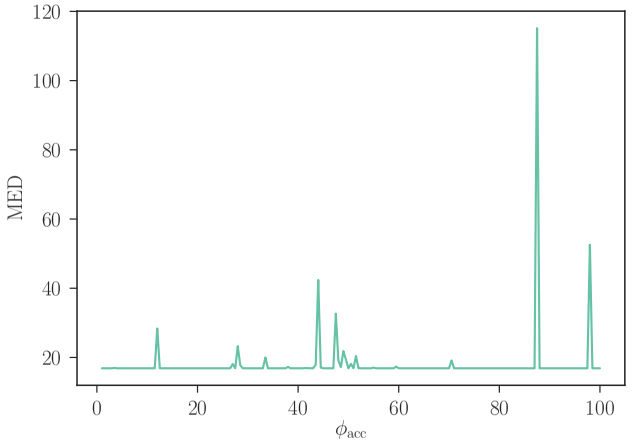
(a) $\phi_{\text{brown}} = 55.37, \phi_{\text{vel}} = 51.72, \phi_{\text{acc}} = 51.72$



(b) $\sigma = 25.72, \phi_{\text{vel}} = 51.72, \phi_{\text{acc}} = 51.72$

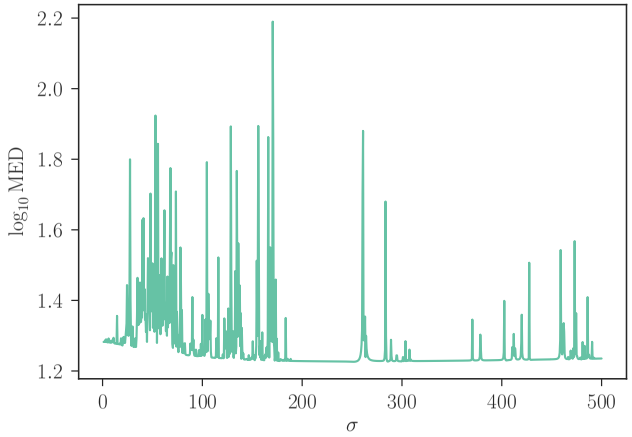


(c) $\sigma = 25.72, \phi_{\text{brown}} = 55.37, \phi_{\text{acc}} = 51.72$

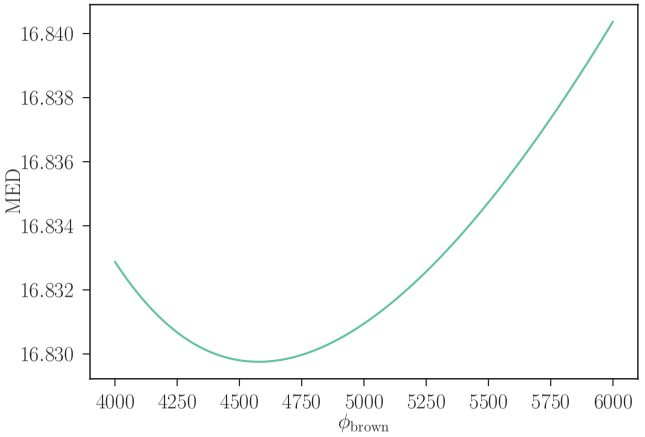


(d) $\sigma = 25.72, \phi_{\text{brown}} = 55.37, \phi_{\text{vel}} = 51.72$

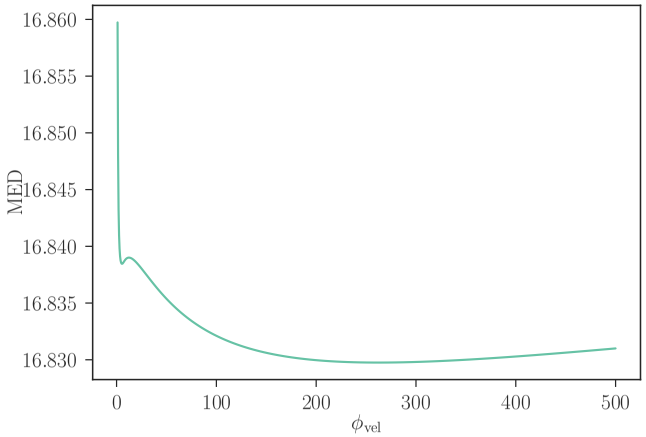
Figure A.9: MED plotted against single parameters of the multimodal Kalman filter, setting other parameters to their optimal values.



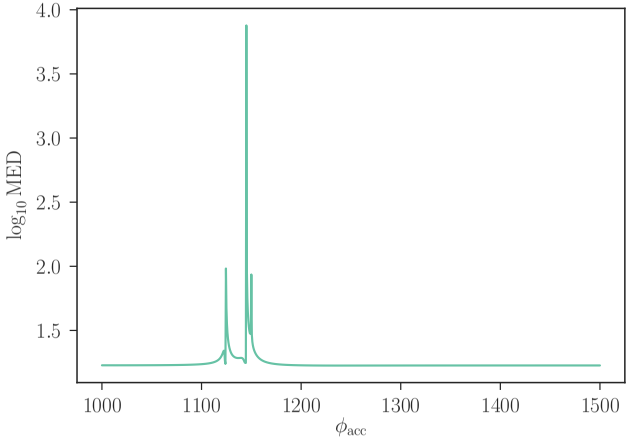
(a) $\phi_{\text{brown}} = 4575.81, \phi_{\text{vel}} = 255.56, \phi_{\text{acc}} = 1235.12$



(b) $\sigma = 251.37, \phi_{\text{vel}} = 255.56, \phi_{\text{acc}} = 1235.12$



(c) $\sigma = 251.37, \phi_{\text{brown}} = 4575.81, \phi_{\text{acc}} = 1235.12$



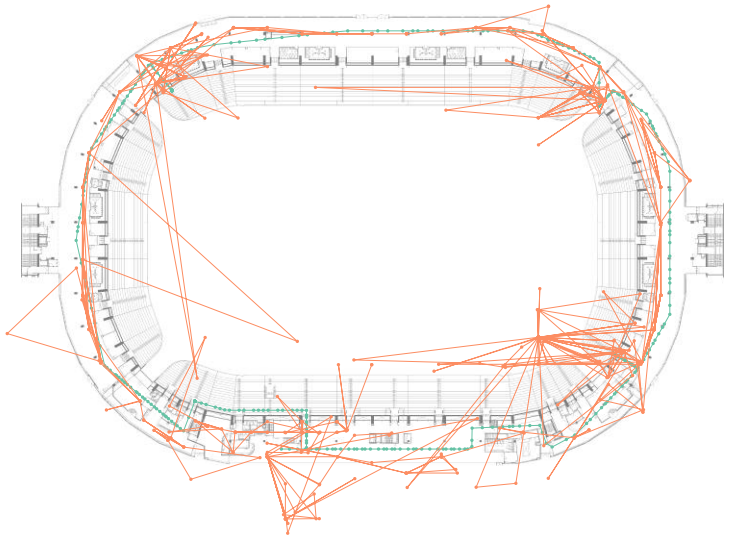
(d) $\sigma = 251.37, \phi_{\text{brown}} = 4575.81, \phi_{\text{vel}} = 255.56$

Figure A.10: MED plotted against single parameters of the dynamic multimodal Kalman filter, setting other parameters to their optimal values.

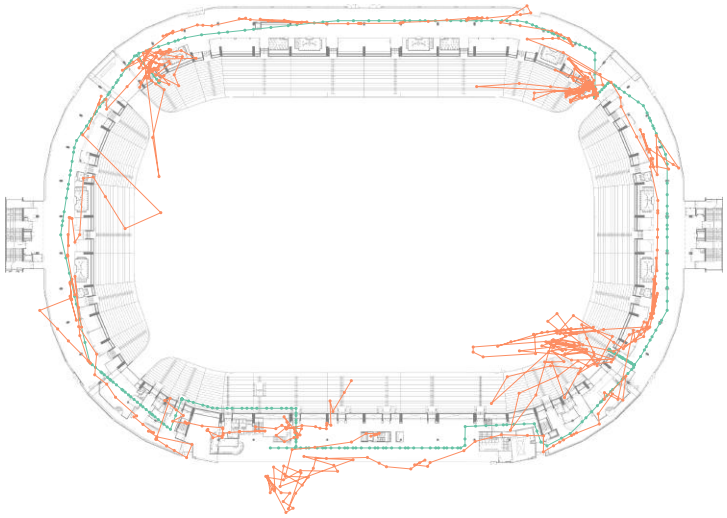
APPENDIX B

Path visualizations

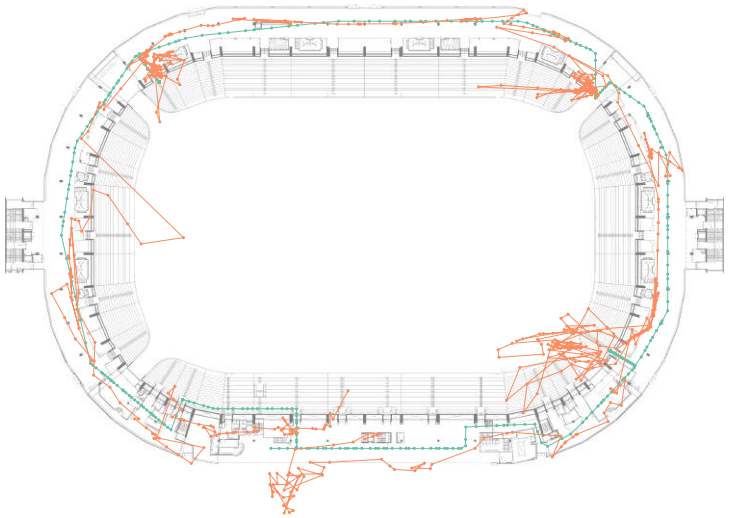
Figure B.1 visualizes the paths obtained by each method using optimal parameter settings. In all these figures the ground truth is colored green, while the estimated path is orange.



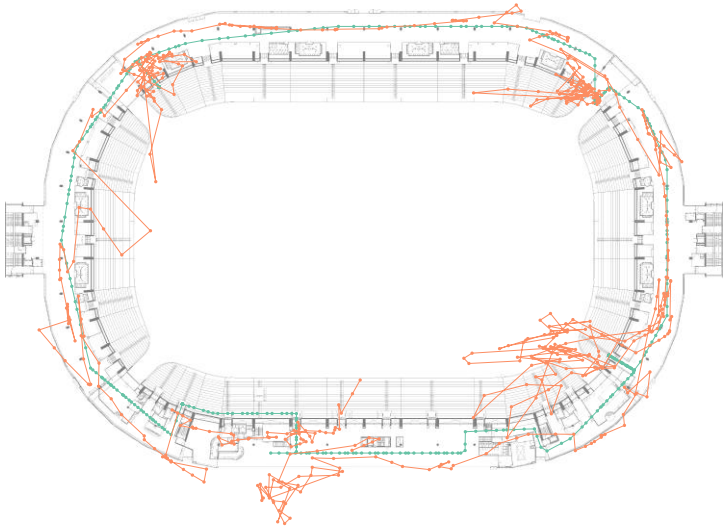
(a) Plain position reconstruction



(b) Basic exponential filtering

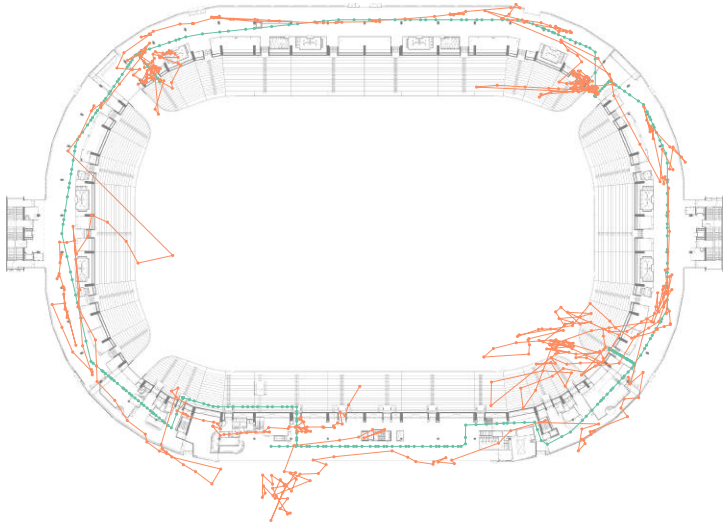


(c) Irregular time exponential filtering

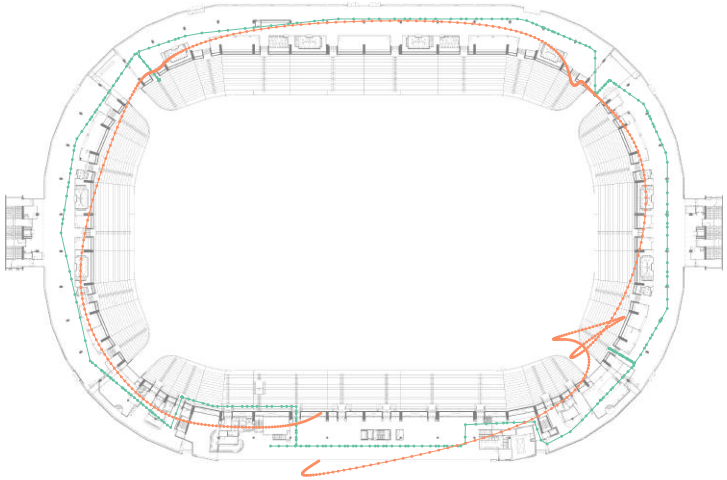


(d) Double exponential filtering

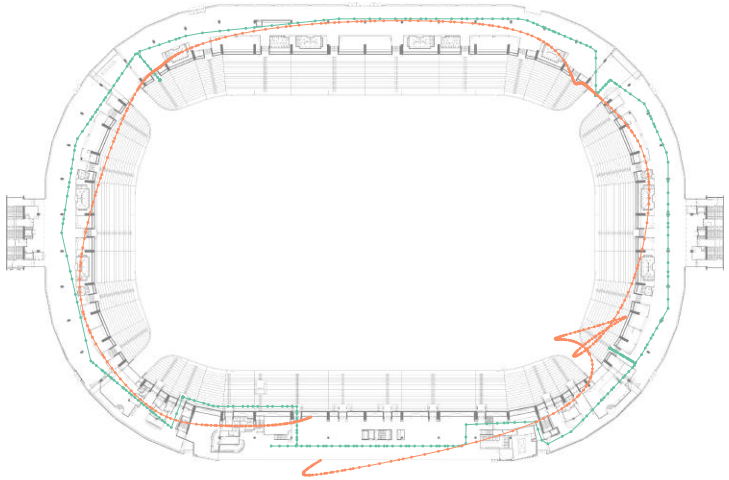
Figure B.1: Optimal path visualizations in orange for all methods, plotted alongside the ground truth in green. Continued on following pages.



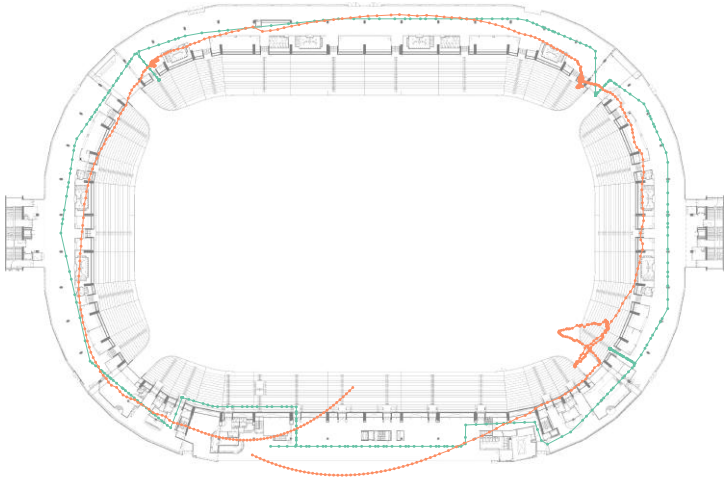
(e) Irregular time double exponential filtering



(f) Gaussian filter

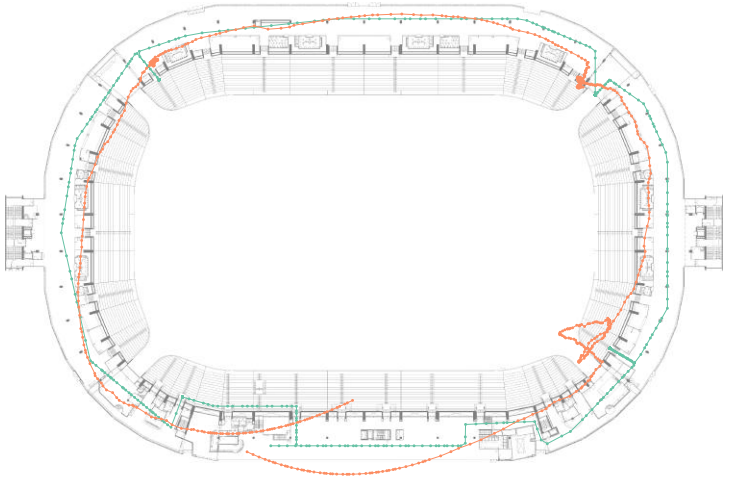


(g) Irregular time Gaussian filter

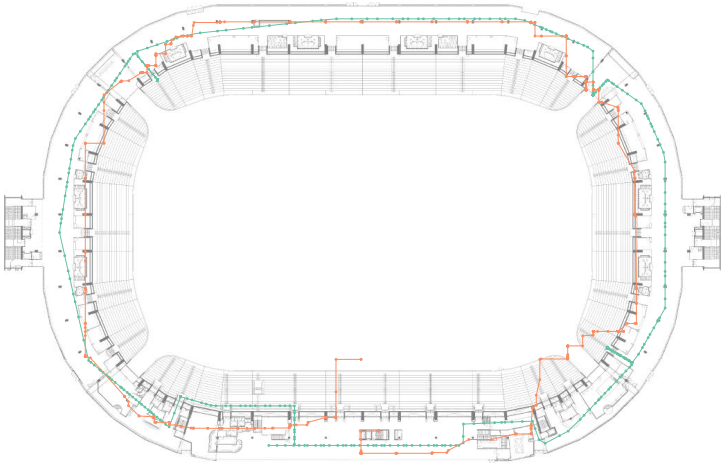


(h) Basic Savitzky-Golay filter

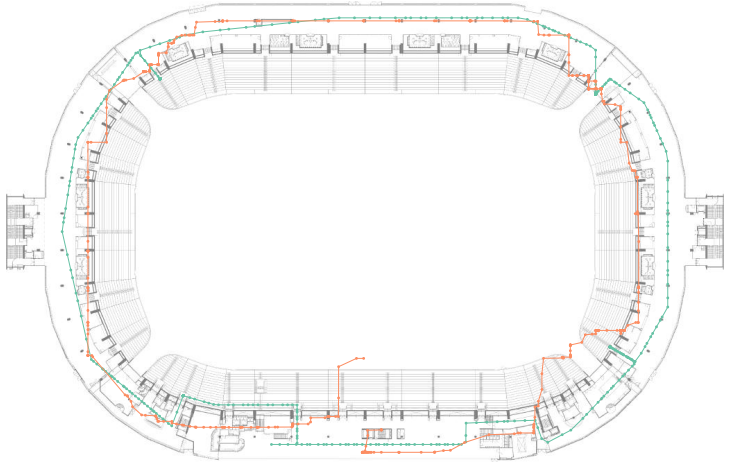
Figure B.1: (continued)



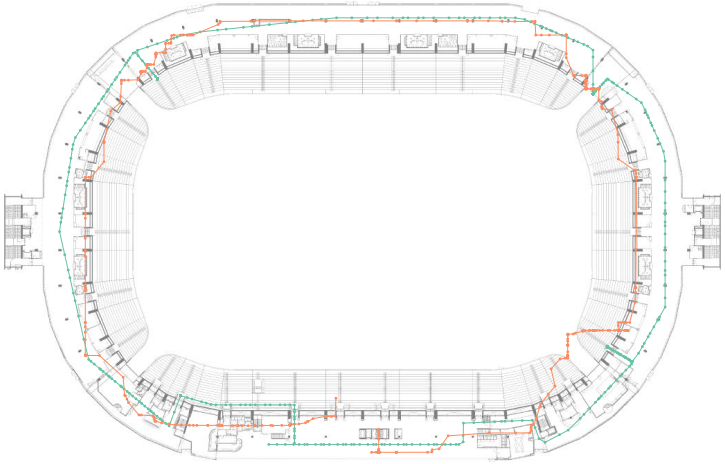
(i) Irregular time Savitzky-Golay filter



(j) Median filter

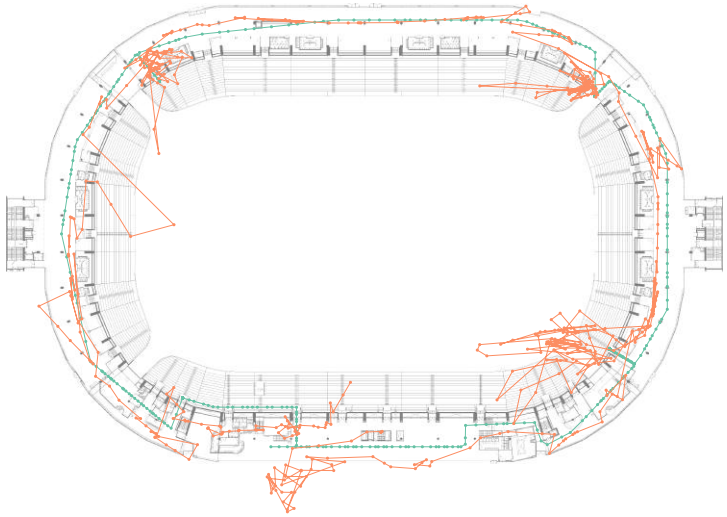


(k) Weighted median filter with Gaussian weights

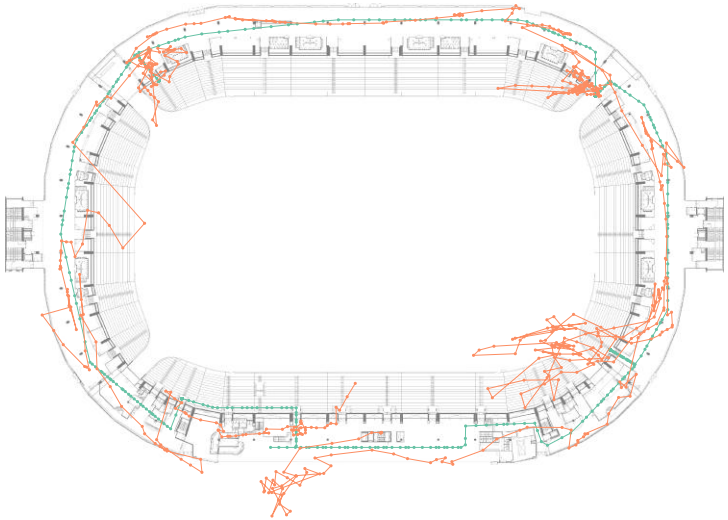


(l) Weighted median filter with shifted time differences

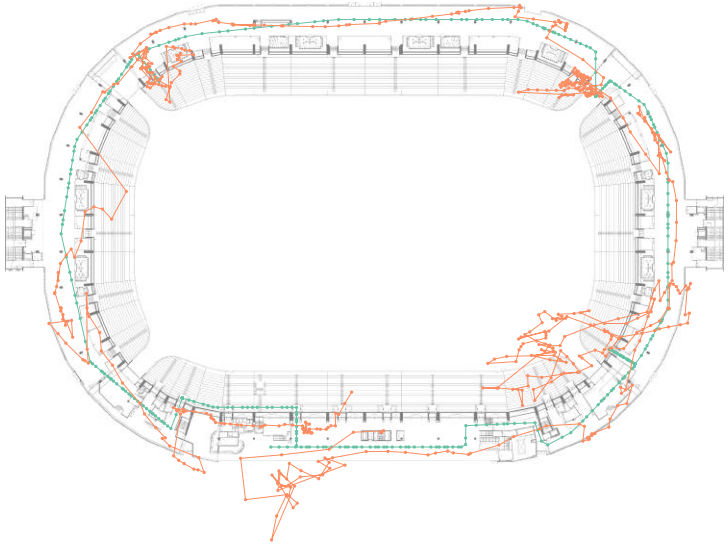
Figure B.1: (continued)



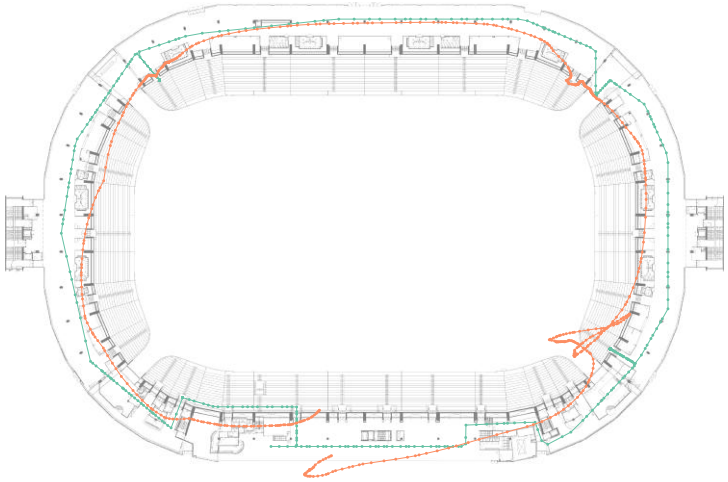
(m) Brownian motion Kalman filter



(n) Constant velocity Kalman filter

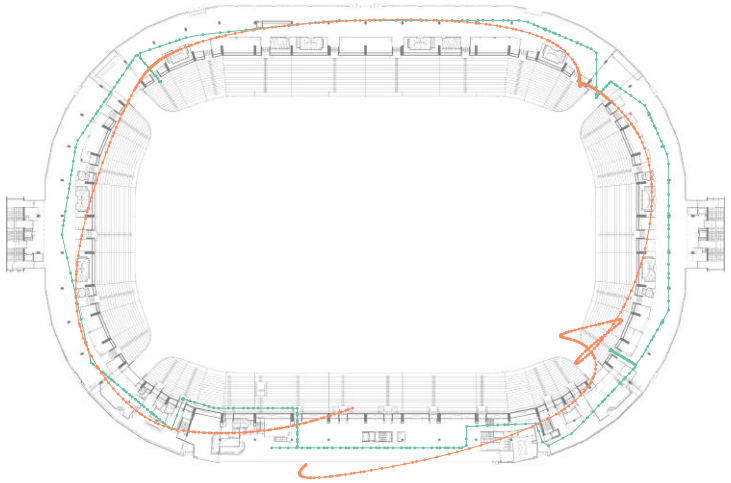


(o) Constant acceleration Kalman filter

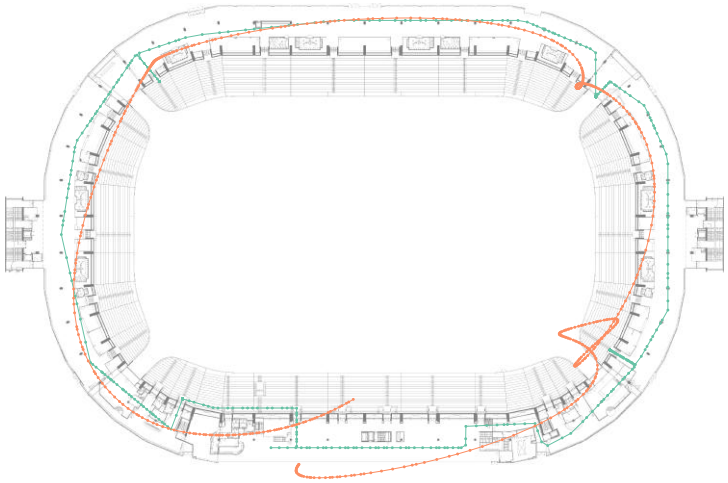


(p) Brownian motion Kalman smoothing

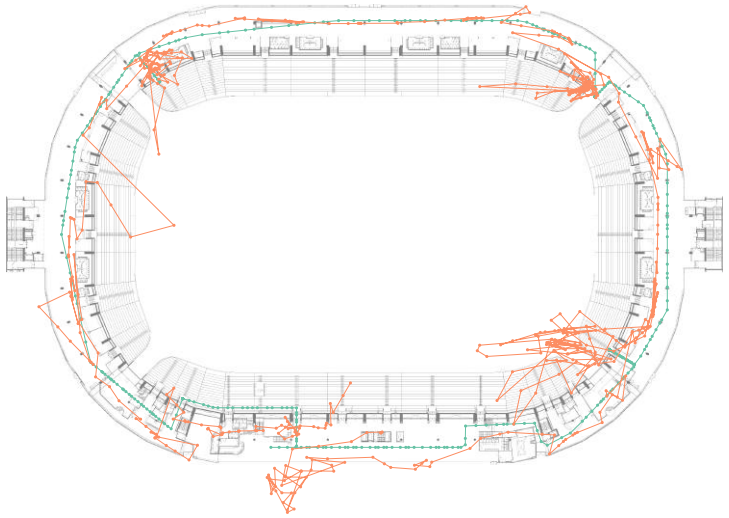
Figure B.1: (continued)



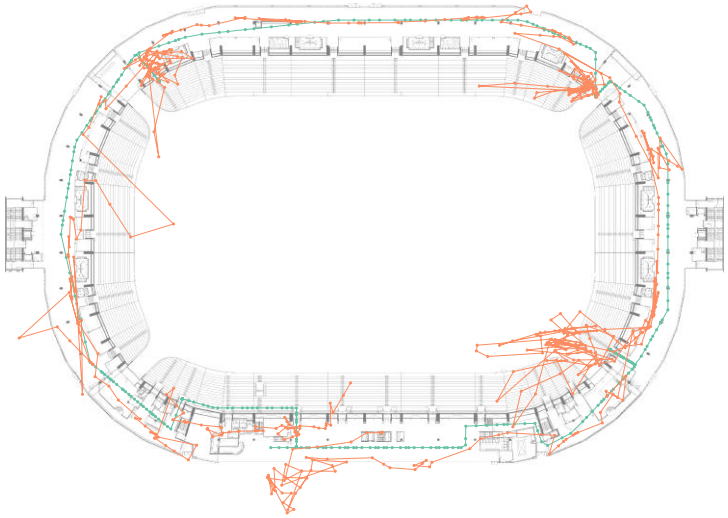
(q) Constant velocity Kalman smoothing



(r) Constant acceleration Kalman smoothing



(s) Multimodal Kalman filter



(t) Dynamic multimodal Kalman filter

Figure B.1: (continued)

APPENDIX C

Hypothesis confidence intervals

Figure C.1 plots histograms of the bootstrapped differences between two filtering methods under the null hypothesis, alongside the 95% confidence interval (colored purple) and the actual observed mean difference (colored blue). Histograms where the observed value does not lie within the confidence interval are colored green to signifiy statistical significance of the observation. Histograms where this is not the case are colored orange.

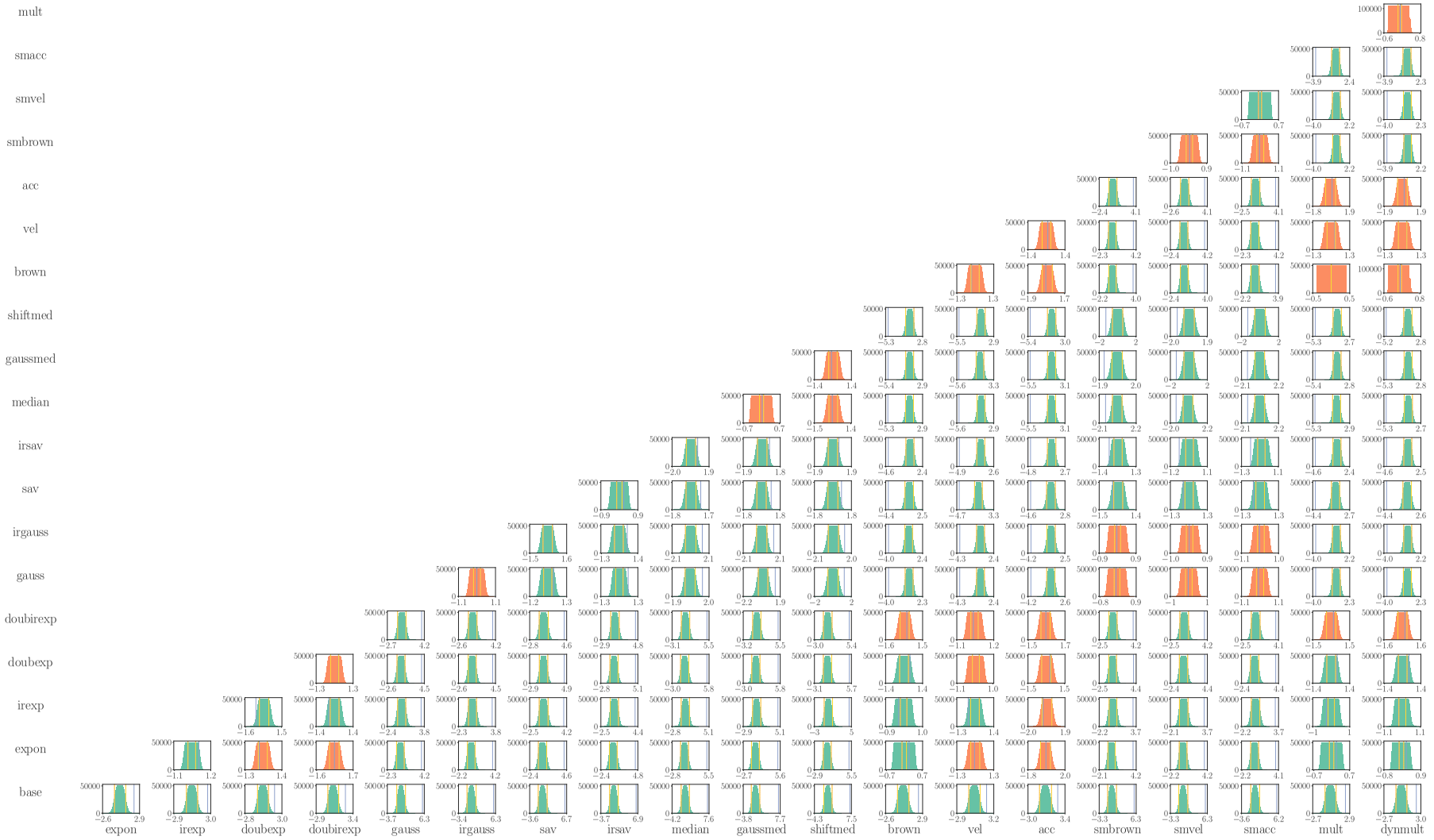


Figure C.1: Histograms of the bootstrapped differences between filtering methods, with the column method being subtracted from the row method. The 95% confidence intervals are drawn using yellow dashed lines, and the observed mean difference is drawn using a solid blue line. Statistical significance of the observation is denoted by the histogram being colored green, and the opposite is denoted by an orange color.

References

[1] M. G. Wing, A. Eklund, and L. D. Kellogg, "Consumer-grade Global Positioning System (GPS) accuracy and reliability", *Journal of Forestry*, vol. 103, no. 4, pp. 169–173, Jun. 2005. DOI: 10.1093/jof/103.4.169.

[2] A. Sayed, A. Tarighat, and N. Khajehnouri, "Network-based wireless location: Challenges faced in developing techniques for accurate wireless location information", *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 24–40, Jul. 2005. DOI: 10.1109/MSP.2005.1458275.

[3] S. Pandey and P. Agrawal, "A survey on localization techniques for wireless networks", *Journal of the Chinese Institute of Engineers*, vol. 29, no. 7, pp. 1125–1148, Oct. 2006, ISSN: 0253-3839. DOI: 10.1080/02533839.2006.9671216.

[4] J. E. Van Engelen, J. J. Van Lier, F. W. Takes, and H. Trautmann, "Crowd flow analysis based on indoor WiFi positioning data (under review)", p. 36, 2017.

[5] K. Kaemarungsi and P. Krishnamurthy, "Modeling of indoor positioning systems based on location fingerprinting", in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, 2004, pp. 1012–1022, ISBN: 0780383559. DOI: 10.1109/INFCOM.2004.1356988.

[6] F. Subhan, H. Hasbullah, A. Rozyyev, and S. T. Bakhsh, "Indoor positioning in Bluetooth networks using fingerprinting and lateration approach", in *2011 International Conference on Information Science and Applications, ICISA 2011*, IEEE, Apr. 2011, pp. 1–9, ISBN: 9781424492244. DOI: 10.1109/ICISA.2011.5772436.

[7] J. Yim, C. Park, J. Joo, and S. Jeong, "Extended Kalman filter for wireless LAN based indoor positioning", *Decision Support System*, vol. 45, no. 4, pp. 960–971, 2008. DOI: 10.1016/j.dss.2008.03.004.

[8] H. P. Mistry and N. H. Mistry, "RSSI based localization scheme in wireless sensor networks: A survey", in *2015 Fifth International Conference on Advanced Computing & Communication Technologies*, IEEE, Feb. 2015, pp. 647–652, ISBN: 978-1-4799-8488-6. DOI: 10.1109/ACCT.2015.105.

[9] J. J. Laviola, "Double exponential smoothing: An alternative to Kalman filter-based predictive tracking", in *Eurographics Workshop on Virtual Environments*, 2003, pp. 199–206. DOI: 10.1145/769953.769976.

[10] K. C. Lee, A. Oka, E. Pollakis, and L. Lampe, "A comparison between unscented Kalman filtering and particle filtering for RSSI-based tracking", in *Proceedings of the 2010 7th Workshop on Positioning, Navigation and Communication, WPNC'10*, IEEE, Mar. 2010, pp. 157–163, ISBN: 9781424471577. DOI: 10.1109/WPNC.2010.5650817.

[11] B. Ferris, D. Fox, and N. D. Lawrence, "WiFi-SLAM using Gaussian process latent variable models", in *IJCAI'07 Proceedings of the 20th international joint conference on Artifical intelligence*, vol. 7, 2007, pp. 2480–2485.

[12] J. Hightower and G. Borriello, "Particle filters for location estimation in ubiquitous computing: A case study", in *International conference on ubiquitous computing*, 2004, pp. 88–106. DOI: 10.1007/978-3-540-30119-6_6.

[13] T. Roos, P. Myllymäki, H. Tirri, P. Misikangas, and J. Sievänen, "A probabilistic approach to WLAN user location estimation", *International Journal of Wireless Information Networks*, vol. 9, no. 3, pp. 155–164, 2002. DOI: 10.1023/A:1016003126882.

[14] IEEE, *IEEE standard for wireless LAN medium access control (MAC) and physical layer (PHY)*, 2012. DOI: 10.1109/IEEESTD.2012.6178212. [Online]. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=6178209.

[15] J. Freudiger, "Short: How talkative is your mobile device? an experimental study of Wi-Fi probe requests", in *WiSec '15 Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2015, pp. 1–6, ISBN: 9781450336239. DOI: 10.1145/2766498.2766517.

[16] H. T. Friis, "A note on a simple transmission formula", *Proceedings of the IRE*, vol. 34, no. 5, pp. 254–256, 1946, ISSN: 00968390. DOI: 10.1109/JRPROC.1946.234568.

[17] W. L. Stutzman and G. A. Thiele, *Antenna Theory and Design*, 3rd ed. Wiley, 2013, p. 822, ISBN: 0470576642.

[18] C. A. Balanis, *Antenna Theory: Analysis and Design*, 3rd ed. Wiley-Interscience, 2005, p. 1117, ISBN: 0471714623.

[19] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall PTR, 2002, p. 707, ISBN: 0130422320.

[20] J. A. Shaw, "Radiometry and the Friis transmission equation", *American Journal of Physics*, vol. 81, no. 1, pp. 33–37, Jan. 2013. DOI: 10.1119/1.4755780.

[21] M. Hata, "Empirical formula for propagation loss in land mobile radio services", *IEEE Transactions on Vehicular Technology*, vol. 29, no. 3, pp. 317–325, Aug. 1980. DOI: 10.1109/T-VT.1980.23859.

[22] A. Bose and C. H. Foh, "A practical path loss model for indoor WiFi positioning enhancement", in *2007 6th International Conference on Information, Communications and Signal Processing, ICICS*, 2008, ISBN: 1424409837. DOI: 10.1109/ICICS.2007.4449717.

[23] J. Röbesaat, P. Zhang, M. Abdelaal, and O. Theel, "An improved BLE indoor localization with Kalman-based fusion: An experimental study", *Sensors (Switzerland)*, vol. 17, no. 5, p. 26, Apr. 2017, ISSN: 14248220. DOI: 10.3390/s17050951.

[24] F. Thomas and L. Ros, "Revisiting trilateration for robot localization", *IEEE Transactions on Robotics*, vol. 21, no. 1, pp. 93–101, Feb. 2005, ISSN: 1552-3098. DOI: 10.1109/TRO.2004.833793.

[25] M. Schmandt, *GIS Commons: An introductory textbook on geographic information systems*, 2010. [Online]. Available: http://giscommons.org/chapter-2-input/ (visited on 05/10/2018).

[26] D. Manolakis, "Efficient solution and performance analysis of 3-D position estimation by trilateration", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 32, no. 4, pp. 1239–1248, 1996. DOI: 10.1109/7.543845.

[27] B. T. Fang, "Trilateration and extension to Global Positioning System navigation", *Journal of Guidance, Control, and Dynamics*, vol. 9, no. 6, pp. 715–717, Nov. 1986. DOI: 10.2514/3.20169.

[28] Y. Zhou, "An efficient least-squares trilateration algorithm for mobile robot localization", in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Oct. 2009, pp. 3474–3479, ISBN: 978-1-4244-3803-7. DOI: 10.1109/IROS.2009.5354370.

[29] N. Wagle and E. Frew, "A particle filter approach to WiFi target localization", in *AIAA Guidance, Navigation, and Control Conference*, 2010, ISBN: 978-1-60086-962-4. DOI: 10.2514/6.2010-7750.

[30] S.-q. Bai, W.-h. Liang, and S. Qin, "Accurate path-loss exponent correcting location method", in *Proceedings of the 33rd Chinese Control Conference*, IEEE, Jul. 2014, pp. 472–475, ISBN: 978-9-8815-6387-3. DOI: 10.1109/ChiCC.2014.6896669.

[31] G. Mao, B. D. Anderson, and B. Fidan, "Path loss exponent estimation for wireless sensor network localization", *Computer Networks*, vol. 51, no. 10, pp. 2467–2483, Jul. 2007, ISSN: 1389-1286. DOI: 10.1016/J.COMNET.2006.11.007.

[32] R. S. Marin, "Wireless indoor localization using pathloss-based techniques", MSc thesis, Tampere University of Technology, 2015, p. 65.

[33] G. Mao, B. Fidan, and B. D. Anderson, "Wireless sensor network localization techniques", *Computer Networks*, vol. 51, no. 10, pp. 2529–2553, Jul. 2007, ISSN: 13891286. DOI: 10.1016/j.comnet.2006.11.018.

[34] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "Minimization or maximization of functions", in *Numerical Recipes: The Art of Scientific Computing*, 3rd ed., Cambridge University Press, 2007, ch. 10, pp. 487–562, ISBN: 9780521880688.

[35] ——, "Nonlinear models", in *Numerical Recipes: The Art of Scientific Computing*, 3rd ed., Cambridge University Press, 2007, ch. 15.5, pp. 799–806, ISBN: 9780521880688.

[36] P. E. Frandsen, K. Jonasson, H. B. Nielsen, and O. Tingleff, "Unconstrained optimization", Department of Mathematical Modelling, Technical University of Denmark, Kongens Lyngby, Tech. Rep., 1999.

[37] S. Gratton, A. S. Lawless, and N. K. Nichols, "Approximate Gauss–Newton methods for nonlinear least squares problems", *SIAM Journal on Optimization*, vol. 18, no. 1, pp. 106–132, Jan. 2007. DOI: 10.1137/050624935.

[38] K. Madsen, H. B. Nielsen, and O. Tingleff, "Methods for non-linear least squares problems", Informatics and Mathematical Modelling, Technical University of Denmark, Tech. Rep., 2004.

[39] K. Levenberg, "A method for the solution of certain non-linear problems in least squares", *Quarterly of Applied Mathematics*, vol. 2, no. 2, pp. 164–168, 1944. DOI: 10.2307/43633451.

[40] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters", *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, Jun. 1963. DOI: 10.1137/0111030.

[41] H. P. Gavin, "The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems", Department of Civil and Environmental Engineering, Duke University, Tech. Rep., 2017.

[42] J. J. Moré, "The Levenberg-Marquardt algorithm: Implementation and theory", in *Proceedings of the Biennial Conference Held at Dundee*, G. A. Watson, Ed., Springer, Berlin, Heidelberg, 1977, pp. 105–116. DOI: 10.1007/BFb0067700.

[43] E. Jones, T. Oliphant, and P. Peterson, *SciPy: Open source scientific tools for Python*, 2001. [Online]. Available: http://www.scipy.org/ (visited on 05/18/2018).

[44] M. I. A. Lourakis, "A brief description of the Levenberg-Marquardt algorithm implemented by levmar", Institute of Computer Science, Foundation for Research and Technology - Hellas, Heraklion, Crete, Tech. Rep., 2005.

[45] S. Thummalapalli, "Wi-Fi indoor positioning", MSc thesis, Halmstad University, 2012, p. 49.

[46] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete Time Signal Processing*, 2nd ed. New Jersey: Prentice Hall, 1999, ISBN: 0137549202.

[47] S. Makridakis, M. Hibon, and C. Moser, "Accuracy of forecasting: An empirical investigation", *Journal of the Royal Statistical Society. Series A (General)*, vol. 142, no. 2, p. 97, 1979. DOI: 10.2307/2345077.

[48] E. S. Gardner, "Exponential smoothing: The state of the art", *Journal of Forecasting*, vol. 4, no. 1, pp. 1–28, 1985. DOI: 10.1002/for.3980040103.

[49] S. Makridakis and M. Hibon, "Exponential smoothing: The effect of initial values and loss functions on post-sample forecasting accuracy", *International Journal of Forecasting*, vol. 7, no. 3, pp. 317–330, Nov. 1991. DOI: 10.1016/0169-2070(91)90005-G.

[50] S. Çapar, "Importance of initial value in exponential smoothing methods", *Sosyal Bilimler Enstitüsü Dergisi*, vol. 17, no. 3, pp. 291–302, 2015. DOI: 10.16953/deusbed.12175.

[51] A. Eckner, "Algorithms for unevenly spaced time series: Moving averages and other rolling operators", 2017, [Online]. Available: http://www.eckner.com/research.html.

[52] D. E. Gardner, "Weight factor selection in double exponential smoothing enrollment forecasts", *Research in Higher Education*, vol. 14, no. 1, pp. 49–56, 1981, ISSN: 0361-0365. DOI: 10.1007/BF00995369.

[53] I. T. Young and L. J. Van Vliet, "Recursive implementation of the Gaussian filter", *Signal Processing*, vol. 44, no. 2, pp. 139–151, Jun. 1995. DOI: 10.1016/0165-1684(95)00020-E.

[54] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures", *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964. DOI: 10.1021/ac60214a047.

[55] P.-O. Persson and G. Strang, "Smoothing by Savitzky-Golay and Legendre filters", in *Mathematical Systems Theory in Biology, Communications, Computation, and Finance*, J. Rosenthal and D. S. Gilliam, Eds., Springer, New York, NY, 2003, pp. 301–315. DOI: 10.1007/978-0-387-21696-6_11.

[56] L. Rabiner, M. Sambur, and C. Schmidt, "Applications of a nonlinear smoothing algorithm to speech processing", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 6, pp. 552–557, Dec. 1975. DOI: 10.1109/TASSP.1975.1162749.

[57] L. Yin, R. Yang, M. Gabbouj, and Y. Neuvo, "Weighted median filters: A tutorial", *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 43, no. 3, pp. 157–192, 1996. DOI: 10.1109/82.486465.

[58] R. E. Kálmán, "A new approach to linear filtering and prediction problems", *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, Mar. 1960. DOI: 10.1115/1.3662552.

[59] R. Faragher, "Understanding the basis of the Kalman filter via a simple and intuitive derivation [lecture notes]", *IEEE Signal Processing Magazine*, vol. 29, no. 5, pp. 128–132, Sep. 2012. DOI: 10.1109/MSP.2012.2203621.

[60] G. Bishop and G. Welch, "An introduction to the Kalman filter", *SIGGRAPH Course Notes*, 2001.

[61] M. Welling, "The Kalman filter", Pasadena, California, 2010, [Online]. Available: http://www.stat.columbia.edu/~liam/teaching/neurostat-spr12/papers/hmm/KF-welling-notes.pdf.

[62] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, 4th ed. John Wiley & Sons, 2012, pp. 143–147, ISBN: 9780470609699.

[63] A. Aurell and B. Djehiche, "Modeling tagged pedestrian motion: A mean-field type control approach", Stockholm, Sweden, 2018, [Online]. Available: https://arxiv.org/pdf/1801.08777.pdf.

[64] R. Crossland, "Brownian motion and crowd psychology", in *Modernist Physics: Waves, Particles, and Relativities in the Writings of Virginia Woolf and D. H. Lawrence*, Oxford University Press, 2018, ch. 5, ISBN: 9780198815976. DOI: 10.1093/oso/9780198815976.001.0001.

[65] R. R. Labbe Jr., "Multivariate Kalman filters", in *Kalman and Bayesian Filters in Python*, 2015, ch. 6.

[66] ——, "Kalman filter math", in *Kalman and Bayesian Filters in Python*, 2015, ch. 7.

[67] H. E. Rauch, C. T. Striebel, and F. Tung, "Maximum likelihood estimates of linear dynamic systems", *AIAA Journal*, vol. 3, no. 8, pp. 1445–1450, Aug. 1965. DOI: 10.2514/3.3166.

[68] R. H. Byrd, L. Peihuang, and J. Nocedal, "A limited-memory algorithm for bound-constrained optimization", Argonne National Laboratory (ANL), Argonne, IL, Tech. Rep., Mar. 1996. DOI: 10.2172/204262.

[69] J. H. Gómez, V Marquina, and R. Gómez, "On the performance of Usain Bolt in the 100 metre sprint", *European Journal of Physics*, vol. 34, no. 5, 2013. DOI: 10.1088/0143-0807/34/5/1227. arXiv: arXiv:1305.3947v2.

[70] J. L. Hintze and R. D. Nelson, "Violin plots: A box plot-density trace synergism", *The American Statistician*, vol. 52, no. 2, pp. 181–184, May 1998. DOI: 10.1080/00031305.1998.10480559.

[71] I. M. Sobol, "Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates", *Mathematics and Computers in Simulation*, vol. 55, no. 1-3, pp. 271–280, 2001. DOI: 10.1016/S0378-4754(00)00270-6.