BEARING-ONLY TRACKING
WITH
KALMAN FILTERS USING SMOOTHED MEASUREMENTS

by

Murat Karayaka

B.S., Electrical and Electronics Engineering, Istanbul Technical University, 2016

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering

Boğaziçi University

2021

ACKNOWLEDGEMENTS

Firstly, I sincerely thank my mother Nebahat, my father Hüseyin and my sisters Sevil and Sevilay who have supported me in all my decisions all the time. I have always felt how lucky I am to have them as my family.

I would like to express my deep gratitude to my professors from my undergraduate study, Dr. Fatih Erden and Dr. Deniz Kumlu. They have always been more than instructors to me. Their mentorship and encouragement will never be forgotten.

I am extremely grateful to my thesis supervisor Dr. Emin Anarım as well as Dr. Mutlu Koca for their valuable guidance and patience throughout my graduate study. I know they would help me find my way when I needed. I would also like to thank my thesis jury, including Dr. Tayfun Akgül, for their inspiring feedbacks.

I would also like to extend a special appreciation to the countless people who have been involved in my academic life or inspired me in any way.

I greatly appreciate the assistance I received from MarineTraffic, who let me use their data services in order to collect data that were used in this thesis's simulations. This thesis has been partially supported by MarineTraffic.

Finally, I would like to thank the Scientific and Technological Research Council of Turkey (TÜBİTAK). This thesis has been supported by the 2210-A National Scholarship Programme for MSc Students from TÜBİTAK.

ABSTRACT

BEARING-ONLY TRACKING WITH KALMAN FILTERS USING SMOOTHED MEASUREMENTS

Kalman filter-based solutions proposed for nonlinear systems are frequently used in bearing-only tracking applications. Due to the physical conditions of these tracking applications, the measurements gathered may contain a high amount of noise. For example, if the measurement sensors are too far from the target being tracked, a small error in the calculated bearing or a small amount of noise exposure will cause the uncertainty in the tracking system to increase significantly. Since the effect of this large amount of noise can only be eliminated to a certain extent by Kalman filter-based solutions, tracking performance may decrease in these applications. In this thesis, various statistical and machine learning-based noise removal methods will be applied to reduce the noise in bearing measurements obtained with sensors. Then, these noise-reduced measurements will be used in Kalman filter-based solutions in bearing-only tracking. The effects of noise reduction methods on tracking performance will be compared with simulations on real vessel trajectories.

ÖZET

KALMAN SÜZGEÇLERİ İLE DÜZGÜNLEŞTİRİLMİŞ ÖLÇÜMLER KULLANARAK SADECE KERTERİZ İLE TAKİP

Doğrusal olmayan sistemler için Kalman süzgeci tabanlı çözümler, sadece kerteriz ile takip uygulamalarında sıklıkla kullanılmaktadır. Bu takip uygulamalarının fiziksel koşulları nedeniyle, elde edilen ölçümler yüksek miktarda gürültü içerebilir. Örneğin, ölçüm sensörlerinin takip edilen hedefe çok uzak olması durumunda, hesaplanan kerterizdeki küçük bir hata veya ölçümün maruz kalacağı az miktarda bir gürültü, takip sistemindeki belirsizliğin önemli ölçüde artmasına neden olacaktır. Kalman süzgeci tabanlı çözümlerle bu yüksek gürültünün etkisi ancak belirli miktarlarda giderilebildiği için bu uygulamalarda takip performansı düşebilir. Bu tezde, sensörlerle elde edilen kerteriz ölçümlerinde gürültüyü azaltmak için çeşitli istatistiksel ve makine öğrenmesi tabanlı gürültü giderme yöntemleri uygulanacaktır. Ardından, bu gürültüsü azaltılmış ölçümler, sadece kerteriz ile takip uygulamasında Kalman süzgeci tabanlı çözümlerde kullanılacaktır. Gürültü azaltma yöntemlerinin takip performansına etkileri, gerçek deniz taşıtı yörüngeleri üzerinde gerçekleştirilen takip simülasyonları sonuçları kullanılarak karşılaştırılacaktır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a_{WT}	Scaling parameter of the mother wavelet in WT
b_0	Parameter of linear regression
b_1	Parameter of linear regression
b_{WT}	Shifting parameter of the mother wavelet in WT
c^{DL}	Cell state in DL-based filtering
C_t	Cross-correlation matrix in the UKF at time t
D_{t+1}	Covariance matrix in smoothing at time $t+1$
e	Ellipticity of an oblate spheroid in stereographic projection
f	Nonlinear state transition function
F	State transition matrix
G_t	Smoothing gain at time t
h	Nonlinear measurement function
h^{DL}	Hidden state in DL-based filtering
H	Measurement matrix
I	Identity matrix
J_f	Jacobian matrix of the nonlinear state transition function f
J_h	Jacobian matrix of the nonlinear measurement function h
K_t	Kalman gain at time t
n_{LR}	Number of the dependent variables in linear regression
N^G	Window size in Gaussian smoothing
N^{MA}	Window size in moving average filtering *or* smoothing
N_m	Number of Monte Carlo simulations
N_p	Number of particles
N_s	State vector dimension
N^{DL}	Batch size in DL-based filtering
$p(x_t\|Z_t)$	Updated posterior probability distribution function at time t
$p(x_t\|Z_{t-1})$	Predicted posterior probability distribution function at time t

$p(x_t\|x_{t-1})$	Probabilistic model of the state transition at time t
$p(z_t\|Z_{t-1})$	Normalizing constant in Bayes' rule at time t
$p(z_t\|x_t)$	Likelihood function at time t
$p(z_t\|\dot{v}_t^n)$	Likelihood function of the particle n at time t
$P_{t\|t}^s$	Smoothed covariance matrix at time t
$P_{t\|t-1}$	Prediction for covariance matrix at time t
Q	Process noise covariance matrix
R	Measurement noise covariance matrix
R_L	Local radius in stereographic projection
R_e	Equatorial radius in stereographic projection
$s(t)$	Continuous-time signal
S_t	Predicted measurement covariance matrix in the UKF at time t
$S_{FT}(\omega)$	Fourier transform of signal $s(t)$
$S_{STFT}(\tau, \omega)$	Short-time Fourier transform of the signal $s(t)$
$S_{WT}(a_{WT}, b_{WT})$	Wavelet transform of the signal $s(t)$
T	Length of the estimated trajectory
$\dot{v}^n$	Particle vector of the particle n
v_t^W	White Gaussian measurement noise at time t
$\overline{vx}_t$	Velocity on the x-axis at time t
$\overline{vy}_t$	Velocity on the y-axis at time t
w_{t-1}^W	White Gaussian process noise at time $t-1$
$\dot{w}^n$	Weight of the particle n
w_{STFT}	Window function in Short-time Fourier transform
w_n^G	Weight in Gaussian smoothing with index n
W^{DL}	Weight matrix in DL-based filtering
W_i^m	Weight for mean vector calculation in the UKF
W_i^c	Weight for covariance matrix calculation in the UKF
x_t	State vector of system at time t
x	The x-axis in Cartesian coordinates
$\bar{x}_t$	Position of target on the x-axis at time t

$x_{t\|t-1}$	Prediction for state vector at time t
$\bar{x}_t$	Ground truth position of target on the x-axis at time t
$\bar{x}_{t\|t}$	Estimated position of target on the x-axis at time t
x_{LR}	Predictor variable in linear regression
x^{MA}	Data point in moving average filtering *or* smoothing
x_t^{ES}	Data point in exponential smoothing at time t
x^{DL}	Data point in DL-based filtering
x_n^G	Data point in Gaussian smoothing with index n
$x_{t\|t}^s$	Smoothed state vector at time t
y	The y-axis in Cartesian coordinates
$\bar{y}_t$	Ground truth position of target on the y-axis at time t
$\bar{y}_{t\|t}$	Estimated position of target on the y-axis at time t
$\bar{y}_t$	Position of target on the y-axis at time t
y_t^{MA}	Result of moving average filtering *or* smoothing at time t
y_t^{ES}	Result of exponential smoothing at time t
y^{DL}	Result of DL-based filtering
y_{LR}	Dependent variable in linear regression
y_t^G	Result of Gaussian smoothing at time t
z_t	Measurement vector of system at time t
Z_t	Measurement set at time t
α^{ES}	Smoothing parameter in exponential smoothing
α_{UKF}	Parameter of the UKF
β_{UKF}	Parameter of the UKF
$\dot{\chi}_t$	Sigma points in the UKF at time t
Δt	Sampling time
κ	Parameter of the UKF
λ	Longitude in stereographic projection
λ_0	Central longitude in stereographic projection
μ_t	Predicted measurement mean in the UKF at time t
σ	Standard deviation of a probability distribution

σ_z	Standard deviation of measurement noise
τ	Parameter for window function in STFT
ϕ	Latitude in stereographic projection
ϕ_1	Central latitude in stereographic projection
χ_C	Conformal latitude in stereographic projection
Ψ	Mother wavelet in WT
Ψ_t^p	Particle set at time t
ω	Frequency in FT

LIST OF ACRONYMS/ABBREVIATIONS

1D	One-Dimensional
2D	Two-Dimensional
AFD	Adaptive Fourier Decomposition
AIS	Automatic Identification System
API	Application Programming Interface
AUX-MMPF	Auxiliary Multiple Model Particle Filter
BLSTM	Bidirectional Long Short-Term Memory
BLUE	Best Linear Unbiased Estimator
BRNN	Bidirectional Recurrent Neural Network
CNN	Convolutional Neural Network
CRLB	Cramér–Rao Lower Bound
DL	Deep Learning
ECG	Electrocardiogram
EEG	Electroencephalography
EEMD	Ensemble Empirical Mode Decomposition
EKF	Extended Kalman Filter
EMD	Empirical Mode Decomposition
FISST	Finite Set Statistics
fMRI	Functional Magnetic Resonance Images
FT	Fourier Transform
GM-PHD	Gaussian Mixture Probability Hypothesis Density Filter
GPS	Global Positioning System
HSV	Hue Saturation Value
ICA	Independent Component Analysis
IMM	Interacting Multiple Model
IMM-EKF	Interacting Multiple Model Extended Kalman Filter
IMM-UKF	Interacting Multiple Model Unscented Kalman Filter
IMO	International Maritime Organization

JMS-PF	Jump Markov System Particle Filter
KF	Kalman Filter
LSTM	Long Short-Term Memory
MABWT	Multi-Adaptive Bionic Wavelet Transform
MAP	Maximum a Posteriori
MC	Measurement-Conditioned
MCs	Mono-Components
ML	Machine Learning
MLP	Multilayer Perceptron
MMPF	Multiple Model Particle Filter
MMSE	Minimum Mean Square Error
MPC	Modified Polar Coordinate
MPE	Mean Position Error
NC	Nested Conditioning
NDCG	Normalized Discounted Cumulative Gain
NLP	Natural Language Processing
pdf	Probability Density Function
PF	Particle Filter
PHD	Probability Hypothesis Density
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
TMA	Target Motion Analysis
SD	Standard Deviation
SMC-PHD	Sequential Monte Carlo Probability Hypothesis Density Filter
SNR	Signal-to-Noise Ratio
SOLAS	Safety of Life at Sea
ST	Stockwell Transform
STFT	Short-Time Fourier Transform
SVD	Singular Vector Decomposition
UHF	Ultra-High Frequency
UKF	Unscented Kalman Filter

UKS	Unscented Kalman Smoother
UT	Unscented Transform
VTS	Vessel Traffic Services
WT	Wavelet Transform

1. INTRODUCTION

We have all been excited about the recent, mind-blowing progress of autonomous vehicle technology. In order to replace the human driver in the driver's seat with the management systems that are built in those cars, human behaviors are to be imitated by these management systems. One of the most important human behaviors is observing their surrounding and deciding about what next move to make to drive safely (e.g., a driver watches the actions of the car in front of him and steps on the brake to avoid crashing). The target tracking algorithms in the literature are applied to help management systems in autonomous vehicles perform this task instead of humans.

This is just one of the most popular applications of target tracking. Target tracking has been one of the most studied subjects in many technological fields, from robotics to computer vision, over the years. The main task of target tracking is to determine the state vector, which consists of the position and the velocity of a moving target at any given time. In tracking applications, a sensor or multiple sensors are responsible for supplying observations from the target to be used in the tracking process. Various types of sensors, depending on the tracking environment, can be utilized to collect necessary data. Sensors can be utilized in variable numbers, from one to many, and they can be either static or moving. In general, the collected data, regardless of which sensor is used, is subject to noise. This noise originates from several factors, such as an inability to observe the target directly, insensitive measurement devices, or environmental effects like inconvenient weather conditions. So, it may not always be possible to detect the source of the noise in order to take precautions; even if it is possible, taking precautions is not easy. However, there are methods for decreasing the effect of noise in the collected data. This process is called "denoising". In order to achieve satisfying results in target tracking, denoising should be performed with suitable tools. There are numerous denoising approaches in the literature, but their performances depend on the noise level. If the noise level is high, these denoising approaches are only able to remove the effect of the noise in limited amounts.

A tracking application in which bearings are used for measurements is called a bearing-only tracking application. In this thesis, we will focus on tracking vessels on the sea using bearings from sensors on the ground as one of the problems in the bearing-only tracking studies. Estimation of trajectories via bearing-only tracking is also known as target motion analysis (TMA). Bearing-only tracking applications can be sorted into two groups based on the movements of sensors. In the first group, a moving bearing sensor is used for obtaining observations. For this case, a single sensor is enough for the TMA. In the second group, sensors that are fixed in terms of positions are used. But in this case, at least two sensors are required at the same time for the TMA. In our study, bearings from two fixed sensors will be used in tracking. We will investigate solutions for increasing the tracking performance in bearing-only tracking applications when the noise level is more than acceptable. Actually, this is one of the common problems in bearing-only tracking applications, since the tracking environment can not be controlled easily, and the sensors are far away to the vessels. So, even the small errors, which occurred while measuring, have serious impacts on the measured data. To overcome this problem, we will use additional denoising methods from the literature as pre-processing steps for noisy bearings before applying bearing-only tracking algorithms.

In Chapter 2, we will present studies relating to target tracking and data denoising approaches from the literature in order to give the general perspective of the problem addressed in this thesis. Following that, in the beginning of Chapter 3, the problem statement of the target tracking for linear and nonlinear systems will be developed. Later, the backgrounds of the target tracking approaches, which will be implemented in this study, will be introduced, and the results of our experiments with these tracking approaches on an example tracking problem will be given. Bearing measurements in the bearing-only tracking will be handled as time series data throughout our study. In Chapter 4, the time series denoising methods that will be used as pre-processing steps in our bearing-only tracking experiments will be explained with results of simulations on example time series data. In Chapter 5, details of our bearing-only tracking experiments and analysis of the experimental results will be covered.

Our study will be completed with Chapter 6 in which we will sum up our study and share our ideas about current methods and our trials about bearing-only tracking, as well as our plans for future studies.

2. LITERATURE REVIEW

In this chapter, we will give details of approaches relating to the addressed problem in this thesis from the literature. In Section 2.1, we will present target tracking studies that are about tracking in different coordinate systems, tracking of multiple targets, tracking of maneuvering targets, bearing-only tracking methods, bearing-only tracking in a special coordinate system and motion constraints in a bearing-only tracking application. In Section 2.2, we will discuss time series denoising studies based on statistical and mathematical backgrounds of time series.

2.1. Target Tracking

In target tracking, most of the time the measurement coordinates and the system dynamic coordinates are different. Thus, the measurement to be exploited must be converted into these system coordinates to make sense for the selected filtering method. Unfortunately, this conversion results in information loss and bias in general. In [1], the best linear unbiased estimator (BLUE) filter was proposed for tracking in the linear systems when nonlinear measurements exist. The BLUE filter equations, which posses flawless measurement conversions, were derived for tracking in both spherical and polar coordinates. In their experiments, the authors demonstrated that the BLUE filter outperformed the other two filters that use measurement conversion methods, which were the measurement-conditioned (MC) and the nested conditioning (NC) methods. The details of these two conversion methods are given in [2].

In real world implementations of target tracking, tracking algorithms are implemented for tracking multiple targets simultaneously in addition to tracking a single target. For multiple target tracking, the measurement-track association problem should be solved, since a set of observations is obtained at each filtered time step. Moreover, this set may not contain an observation for each target or may contain false alarms.

In [3], a filtering method named the probability hypothesis density (PHD) filter was proposed as a multiple target tracking algorithm. Here, only the first order moment of the multi-target posterior distribution, namely PHD, was used for estimation, differently from other multi-target Bayes filters. In this study, the target sets and the measurement sets for multiple target tracking were modeled as random finite sets and recursive Bayes filter equations were derived using the finite set statistics (FISST). For the linear Gaussian systems, a closed form solution of the PHD filter in which the PHD was expressed with a Gaussian mixture model, called the Gaussian mixture probability hypothesis density filter (GM-PHD), was proposed in [4]. Sequential Monte Carlo implementation of the PHD filter (SMC-PHD) was proposed in [5] for nonlinear and non-Gaussian systems.

In the tracking of maneuvering targets, it is not possible to find a single suitable system dynamic equation which covers all possible situations. By looking at targets' actions, three groups of situations can be identified: moving with constant velocity, moving with constant acceleration, or turning. The interacting multiple model (IMM) filtering algorithm was proposed for tracking of maneuvering targets in [6]. Here, several filtering methods with different modes were run at the same time. The mode stands for one of the several situations explained before, such as a target with constant velocity. It was assumed the target state transition happens according to one of these modes at each time step, and changes between these modes was modeled as a Markov chain throughout tracking. While tracking, the inputs of a filter with a specific mode at any time were computed using the last outputs of all filters from the previous time, and the probabilities of the system switched from these filters' modes to this specific filter mode. The output of the IMM algorithm was calculated with outputs of the individual filters and their model probabilities. These model probabilities were updated using the observation at each time step. In conclusion, the author stated that the IMM filter outperformed the other maneuvering target tracking algorithms with less computational complexity. Compact versions of this tracking algorithm are given in [7, 8].

2.1.1. Bearing-Only Tracking

In [9], several particle filter (PF)-based methods were proposed for maneuvering targets in the bearing-only tracking problem. Those methods are multiple model PF (MMPF), auxiliary MMPF (AUX-MMPF) and jump Markov system PF (JMS-PF). The details of their methods were provided by the study references. Their performances were compared against IMM-aided traditional tracking filters, namely IMM-extended Kalman filter (IMM-EKF) and IMM-unscented Kalman filter (IMM-UKF). The authors tested their proposed algorithms in three different cases: single-sensor, multi-sensor and tracking with hard constraints. In their study, these constraints were applied as limiting the target speed range. For the first two cases, they were able to derive the Cramér-Rao lower bound (CRLB) as the best possible performance, in order to evaluate filtering results. In their experiments, the authors observed that the PF-based methods outperformed the traditional IMM-based filters, due to the fact that they were more compatible with non-linear systems.

In [10], due to the EKF's instabilities in bearing-only tracking, an alternative coordinate system, called the modified polar coordinate (MPC) system, was proposed. State and measurement vectors were re-defined and derived in these new coordinates. In this study, it was specified that, with the new definition of the state vector, the components of this vector, which were computable without observer maneuvering, and the other components, which required maneuver for computation, were separated. The authors claim that this separation resolved instability of the EKF by fixing ill-conditioned covariance matrix. The proposed EKF performance was tested on realistic scenarios, and comparisons of this filter with Cartesian and pseudo-linear filter equivalents were provided. In their experiments, the authors observed that the proposed method has given unbiased and stable results in comparison with the others. Additionally, for pseudo-linear filters that were tested in these experiments, researchers attempted to linearize measurement models by using pseudo-state measurements, which are computed as a nonlinear transformation of a bearing measurement, in place of bearing measurements in [11].

In [12], observability in TMA was discussed. Several types of scenarios, depending on the observer's maneuvers, were tested. In the first scenario, the observability of a target which travels with constant speed and course, monitored by two sensors that have constant distance from each other, was examined. It was stated for this case that, if the target does not travel on the connection line of two sensors, observability is ensured by triangulation, without any requirements on the observer's maneuvers. In the second scenario, a target which travels with constant speed and a moving sensor with constant speed and course were used. But this time, the observed frequency from the target was taken into account. For this case, observability was satisfied if the bearings were not stable. In the third scenario, a target and a sensor moved on parallel horizontal trajectories with constant velocities. The distance between those trajectories were assumed to be known. In this case, only bearing measurements were required for observability. Moreover, in this study, examples of inefficient maneuvers for tracking targets with constant velocity and constant accelerations were presented with their theoretical explanations. In the final scenario, it was stated that a maneuver is required by a sensor if a target and a sensor move on different horizontal trajectories.

2.2. Time Series Denoising

In [13], a nonlinear adaptive model of a free noise reduction algorithm for time series was proposed. In this method, time series were divided into smaller groups and polynomials with predetermined degrees that fit the data in these smaller groups were computed. In both experiments on denoising the chaotic Lorenz data and eliminating the electrocardiogram (ECG) portions from the electroencephalography (EEG) data, the authors demonstrated that the proposed method outperformed the widely used wavelet shrinkage method.

In [14], the modified EKF-based framework for ECG signal denoising and compression was proposed. These modifications enabled the use of the EKF algorithm with 17-dimensional data.

The performance of the proposed framework was compared with the traditional EKF (that has two state variables) and a wavelet-based filtering methods called the multiadaptive bionic wavelet transform (MABWT) that was proposed in [15]. It was stated that the proposed method outperformed the other competitors in evaluation metrics for denoising and compressions.

In [16], a statistical tool called independent component analysis (ICA) was proposed for denoising the ECG signal and eliminating unintended artifact components of it. Moreover, a new, automated independent component order-determining method, was presented for online applications to replace visual analysis. This method benefits statistical differences of the signal ICA components. A three-channel ECG signal was used in this study and several combinations of noise and artifact existence in those channels, such as one noisy channel or two noisy channels with artifacts, were examined for demonstration of the algorithmic results. It was stated that the proposed method was able to denoise the ECG signal up to a certain amount of noise and eliminate artifact components up to certain amplitudes.

In [17], a signal denoising method based on the Stockwell transform (ST) [18], which is used for analyzing signal in time–frequency domain, was proposed. The ST is carried out similar to the short-time Fourier transform (STFT) and the wavelet transform (WT). But, in the ST, varying window size depending on frequency is used, unlike a fixed-length window size in the STFT. Moreover, a time-frequency signal representation is obtained in the ST instead of a time-scale one, which is used in the WT. Advantages of the ST include progressive resolution, frequency-invariant amplitude response, and absolutely-referenced phrase information. For denoising, firstly, the ST of the noisy signal is computed. Then filtering and masking are applied to this new representation of the noisy signal. Finally, the denoised signal is reconstructed by applying the inverse of the ST. The performance of the presented method is tested against several noise types, such as white Gaussian and muscle artifact, by observing the signal-to-noise ratio (SNR) and root mean square error (RMSE) values.

It was stated that the proposed method outperformed the other WT-based approaches, namely WT-Soft [19] and WT-Subband [20], in simulations on the real case and artificially generated scenarios.

In [21], the adaptive Fourier decomposition (AFD)-based signal denoising method was proposed. The AFD extracts signal energy information that enables us to discriminate between signals with the same frequency but different energy distribution. In the AFD, at each decomposition level, mono-components (MCs) and a standard remainder are computed from the signal. Earlier-computed MCs correspond to the higher energy components of the signal, which are accepted as the noise-free parts. Thus, in order to denoise the signal, decomposition should be stopped at a suitable level before computation of values from noisy parts. In this study, the decomposition level that maximizes the SNR was selected iteratively for ending decomposition. Finally, the denoised signal was reconstructed using these computed values. The presented algorithm effectiveness was verified on both synthetic and real ECG signals, by comparisons with WT-based, ST-based [17], empirical mode decomposition (EMD)-based and ensemble empirical mode decomposition (EEMD)-based denoising methods [22].

3. TARGET TRACKING

In this chapter, we will introduce important concepts and studies from the literature relating to target tracking. As we stated before, this topic has been widely studied by numerous researchers; thus, the literature is quite extensive. In the scope of our study, target tracking will be handled as two groups: target tracking in linear systems and target tracking in nonlinear systems. Although tracking in linear systems is not very common in the real world, information about this will help us to build a base of knowledge about tracking.

In Section 3.1, we will cover some basic concepts about dynamics of linear and nonlinear systems, respectively. These concepts will be used to develop algorithms in the following sections. In Section 3.2, we will present one of the most used solutions from the literature for tracking, the Bayesian filtering method. Following that, a solution for linear systems based on the Bayesian filtering method will be given in Section 3.3, whereas several solutions for nonlinear systems will be given in Section 3.4. We will conclude this chapter with Section 3.5, in which tracking simulations using the presented solutions are given.

3.1. Preliminaries

3.1.1. Linear System and Measurement Models

The dynamic model for a linear system can be expressed as

$$x_t = Fx_{t-1} + w_{t-1}^W, \tag{3.1}$$

where x_t represents the state vector of the system at a time step t. In tracking systems, this state vector consists of velocities and positions of the target in both x and y coordinates.

F is the state transition matrix and w_{t-1}^{W} stands for a white Gaussian process noise. We would like to rewrite the linear system dynamics in matrix form as

$$x_t = \begin{bmatrix} \bar{x}_t \\ \bar{y}_t \\ \overline{vx}_t \\ \overline{vy}_t \end{bmatrix} = F \begin{bmatrix} \bar{x}_{t-1} \\ \bar{y}_{t-1} \\ \overline{vx}_{t-1} \\ \overline{vy}_{t-1} \end{bmatrix} + w_{t-1}^{W}, \tag{3.2}$$

where

$$F = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.3}$$

in order to adapt them to the implementation easily. In these equations, $\bar{x}_t$ and $\bar{y}_t$ are positions, and $\overline{vx}_t$ and $\overline{vy}_t$ are velocities in both x and y coordinates at time step t. Δt is a sampling time for the tracking.

We also need to know the system measurement model for tracking. The measurement model in linear systems can be expressed as

$$z_t = Hx_t + v_t^{W}, \tag{3.4}$$

where z_t is a measurement vector, H is a measurement matrix and v_t^{W} is a white Gaussian measurement noise.

In the same manner, we can describe the measurement model in matrix form as

$$z_t = \begin{bmatrix} \bar{x}_t \\ \bar{y}_t \end{bmatrix} + v_t^W = H \begin{bmatrix} \bar{x}_t \\ \bar{y}_t \\ \overline{vx}_t \\ \overline{vy}_t \end{bmatrix} + v_t^W, \tag{3.5}$$

where

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{3.6}$$

If the system dynamic model and the system measurement model are linear, and the process and the measurement noise follow Gaussian distributions, this system is known as a linear Gaussian system.

3.1.2. Nonlinear System and Measurement Models

Nonlinear system and measurement models can only be defined by nonlinear functions as

$$x_t = f(x_{t-1}) + w_{t-1}^W \tag{3.7}$$

$$z_t = h(x_t) + v_t^W, \tag{3.8}$$

contrary to the matrix definitions given in Section 3.1.1. In these equations, f is a nonlinear state transition function, whereas h is a nonlinear measurement function.

3.2. The Bayesian Filtering

In tracking with the Bayesian approach, the posterior pdf (probability density function) of the system state is computed using information about the system that has been collected until the computation time [23].

This posterior pdf can be used for extracting information, such as an optimal state estimate, in tracking applications. This extracted information must be updated when new system information is obtained. Computing a new posterior pdf for a state estimation at each time step is not an efficient solution for tracking applications. Thus, a recursive filtering method, which uses the collected information sequentially, is a commonly accepted solution in tracking [24].

We will explain the recursive filtering method with statistical tools. Suppose that we have a measurement set Z_t, as

$$Z_t = \{z_i : i = 1, 2, ..., t\}, \tag{3.9}$$

for making an estimation at the time step t. In order to compute the posterior pdf $p(x_t|Z_t)$ recursively, we will use the computed posterior pdf $p(x_{t-1}|Z_{t-1})$, from the previous time step $t-1$.

Firstly, in recursive filtering, the prediction for the posterior pdf $p(x_t|Z_{t-1})$ is computed with the Chapman-Kolmogorov equation as

$$p(x_t|Z_{t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|Z_{t-1})dx_{t-1}. \tag{3.10}$$

This is the so-called prediction step of the recursive filtering. In this equation, $p(x_t|x_{t-1})$ should be calculated using the system dynamic model. Later, the predicted posterior pdf is updated as

$$\begin{aligned} p(x_t|Z_t) &= p(x_t|z_t, Z_{t-1}) \\ &= \frac{p(z_t|x_t, Z_{t-1})p(x_t|Z_{t-1})}{p(z_t|Z_{t-1})} \\ &= \frac{p(z_t|x_t)p(x_t|Z_{t-1})}{p(z_t|Z_{t-1})}, \end{aligned} \tag{3.11}$$

using the new measurement z_t, with the help of Bayes' theorem.

The computation of the posterior pdf $p(x_t|Z_t)$ is defined as the update step of recursive filtering. In order to use the equation above, $p(z_t|Z_{t-1})$ should be obtained with

$$p(z_t|Z_{t-1}) = \int p(z_t|x_t)p(x_t|Z_{t-1})dx_t, \tag{3.12}$$

using $p(z_t|x_t)$ that can be computed from the system measurement model. Finally, the updated posterior pdf $p(x_t|Z_t)$ is used for extracting estimations. For example, using this pdf, one can obtain the minimum mean square error (MMSE) estimate as

$$\hat{x_t}^{MMSE} = E\{x_t|Z_t\} = \int x_t p(x_t|Z_t)dx_t, \tag{3.13}$$

or the maximum a posteriori (MAP) estimate as

$$\hat{x_t}^{MAP} = \arg\max_{x_t} p(x_t|Z_t). \tag{3.14}$$

The computations of the pdf in the prediction step and in the update step of the recursive filtering are intractable, unless there are any restrictions on the probability distributions as linear Gaussian systems.

3.3. Target Tracking in Linear Systems

3.3.1. The Kalman Filter (KF)

If the pdf to be calculated with the Bayesian recursive filtering belongs to a Gaussian distribution, the recursive filtering equations can be solved analytically because only two parameters (a distribution mean and a distribution covariance) are required for expressing a Gaussian distribution. It was proved that in linear systems, the posterior distribution is a Gaussian distribution if the prior distribution is a Gaussian distribution [23].

The Kalman filter (KF) is one of the most important algorithms in estimation studies. It was introduced as the optimal mean squared error filter in linear Gaussian systems by Rudolf Kalman in [25]. The KF is accepted as a special case solution for the Bayesian recursive filtering. Since the KF is recursive, it only needs the estimation from the previous time step and the measurement of the current time step to make an estimation. Thus, its computation is efficient, and it can be used in real time.

In the target tracking systems, the KF can be utilized to filter noisy observations in order to estimate the mean vector and the covariance matrix of the target state. Filtering with the KF is carried out in the following two steps, namely the prediction and the update steps.

In the prediction step of the KF, predictions for the state vector $x_{t|t-1}$ and the covariance matrix $P_{t|t-1}$ are computed using the latest updated state vector $x_{t-1|t-1}$ and the covariance matrix $P_{t-1|t-1}$ as

$$x_{t|t-1} = F x_{t-1|t-1} \tag{3.15}$$

$$P_{t|t-1} = F P_{t-1|t-1} F^T + Q, \tag{3.16}$$

where Q is the process noise covariance matrix, and F is the state transition matrix of the system.

After the prediction step, the KF updates its predictions with the help of a measurement vector z_t belonging to the current time step. The state vector is updated using a Kalman gain K_t, which can be thought of as the weight for the residual between estimation and observation. A Kalman gain is calculated with

$$K_t = P_{t|t-1} H^T \Big(H P_{t|t-1} H^T + R \Big)^{-1}, \tag{3.17}$$

where R is the measurement noise covariance matrix and H is the measurement matrix of the system.

Lastly, the predicted state vector and the predicted covariance matrix are updated with the computed Kalman gain, as

$$x_{t|t} = x_{t|t-1} + K_t\Big(z_t - Hx_{t|t-1}\Big) \tag{3.18}$$

$$P_{t|t} = \Big(I - K_t H\Big)P_{t|t-1}, \tag{3.19}$$

respectively. Here I is the identity matrix.

3.4. Target Tracking in Nonlinear Systems

3.4.1. The Extended Kalman Filter (EKF)

It was stated that the KF is the optimal estimator in linear Gaussian systems in Section 3.3.1. Unfortunately, in real world implementations, most systems are nonlinear. Modifications and improvements are required to use the KF in this kind of system. The extended Kalman filter (EKF) is one of the KF-based approaches that is modified to nonlinear system conditions [26].

In the EKF, nonlinear system and measurement equations are linearized locally using the Taylor series. Jacobian matrices of the state transition function J_f and the measurement function J_h are evaluated at the latest position estimation $x_{t-1|t-1}$ and the current position prediction $x_{t|t-1}$ as

$$J_f(x_{t-1|t-1}) = \left.\frac{\partial f}{\partial x}\right|_{x_{t-1|t-1}} \tag{3.20}$$

$$J_h(x_{t|t-1}) = \left.\frac{\partial h}{\partial x}\right|_{x_{t|t-1}}. \tag{3.21}$$

The computed values of $J_f(x_{t-1|t-1})$ and $J_h(x_{t|t-1})$ replace the state transition matrix F and the measurement matrix H, respectively, in the KF algorithm.

Firstly, in the prediction step of EKF algorithm, state and covariance matrix predictions are calculated as

$$x_{t|t-1} = f(x_{t-1|t-1}) \tag{3.22}$$

$$P_{t|t-1} = J_f(x_{t-1|t-1})P_{t-1|t-1}J_f^T(x_{t-1|t-1}) + Q. \tag{3.23}$$

The only difference with the KF in this prediction computation is that $J_f(x_{t-1|t-1})$ is used instead of F.

In the update step of the EKF, the Kalman gain

$$K_t = P_{t|t-1}J_h^T(x_{t|t-1})\Big(J_h(x_{t|t-1})P_{t|t-1}J_h^T(x_{t|t-1}) + R\Big)^{-1}, \tag{3.24}$$

is required to update predictions as before in the KF. The only difference with the KF in this update computation is that $J_h(x_{t|t-1})$ is used instead of H. The predicted state vector and the predicted covariance matrix are updated with the equations below:

$$x_{t|t} = x_{t|t-1} + K_t\Big(z_t - h(x_{t|t-1})\Big) \tag{3.25}$$

$$P_{t|t} = \Big(I - K_t J_h(x_{t|t-1})\Big)P_{t|t-1}. \tag{3.26}$$

3.4.2. The Unscented Kalman Filter (UKF)

Another widely applied approach for tracking in nonlinear systems is the unscented Kalman filter (UKF) [27]. The UKF uses the unscented transform (UT) method to estimate the new state distribution. In this transform, carefully selected sigma points are propagated through time, and final state and covariance estimations are computed as a weighted sum of these points.

In the prediction step of the UKF, sigma points χ at the time step $t-1$ are calculated with

$$\chi_{t-1}^{0} = x_{t-1|t-1} \tag{3.27}$$

$$\chi_{t-1}^{i} = x_{t-1|t-1} + \left(\sqrt{(N_s+\lambda)P_{t-1|t-1}}\right)_i \qquad i = 1, ..., N_s \tag{3.28}$$

$$\chi_{t-1}^{i+N} = x_{t-1|t-1} - \left(\sqrt{(N_s+\lambda)P_{t-1|t-1}}\right)_i \qquad i = 1, ..., N_s, \tag{3.29}$$

where $\left(\sqrt{M}\right)_i$ stands for ith row of the square root of the matrix M inside. In these equations, superscripts are the indices of sigma points. N_s indicates the state vector dimension, and λ is calculated with control parameters α_{UKF} is set as $\frac{1}{100}$ and κ is set as 0 with

$$\lambda = \alpha_{UKF}^2(N_s + \kappa) - N_s. \tag{3.30}$$

Afterwards, these sigma points χ are propagated through time with a nonlinear state transition function f, as

$$\hat{\chi}_t^i = f(\chi_{t-1}^i) \qquad i = 0, ..., 2N_s. \tag{3.31}$$

As the last thing for the prediction step, the predicted state mean $x_{t|t-1}$ and the predicted state covariance $P_{t|t-1}$ are computed as

$$x_{t|t-1} = \sum_{i=0}^{2N_s} W_i^m \hat{\chi}_t^i \tag{3.32}$$

$$P_{t|t-1} = \sum_{i=0}^{2N_s} W_i^c (\hat{\chi}_t^i - x_{t|t-1})(\hat{\chi}_t^i - x_{t|t-1})^T + Q, \tag{3.33}$$

respectively.

In these equations, W_i^m is the weight for the mean vector calculation, whereas W_i^c is the weight for the covariance matrix calculation, and they can be obtained as

$$W_0^m = \frac{\lambda}{\lambda + N_s} \tag{3.34}$$

$$W_0^c = \frac{\lambda}{\lambda + N_s} + (1 - \alpha_{UKF}^2 + \beta_{UKF}) \tag{3.35}$$

$$W_i^m = W_i^c = \frac{1}{2(\lambda + N_s)} \qquad i = 1, ..., 2N_s, \tag{3.36}$$

where the parameter β_{UKF} is set as 2 for a Gaussian distribution.

In the update step of the UKF, new sigma points $\dot{\chi}_t$ are calculated using the predicted state mean $x_{t|t-1}$ and the predicted state covariance $P_{t|t-1}$ as

$$\dot{\chi}_t^0 = x_{t|t-1} \tag{3.37}$$

$$\dot{\chi}_t^i = x_{t|t-1} + \left(\sqrt{(N_s + \lambda)P_{t|t-1}}\right)_i \qquad i = 1, ..., N_s \tag{3.38}$$

$$\dot{\chi}_t^{i+N} = x_{t|t-1} - \left(\sqrt{(N_s + \lambda)P_{t|t-1}}\right)_i \qquad i = 1, ..., N_s. \tag{3.39}$$

Then, these new sigma points are transformed with a nonlinear measurement function h as

$$\xi_t^i = h(\dot{\chi}_t^i) \qquad i = 0, ..., 2N_s. \tag{3.40}$$

The predicted measurement mean μ_t and the predicted measurement covariance matrix S_t are computed by

$$\mu_t = \sum_{i=0}^{2N_s} W_i^m \xi_t^i \tag{3.41}$$

$$S_t = \sum_{i=0}^{2N_s} W_i^c (\xi_t^i - \mu_t)(\hat{\xi}_t^i - \mu_t)^T + R. \tag{3.42}$$

The cross-correlation matrix C_t and the Kalman gain K_t are obtained as

$$C_t = \sum_{i=0}^{2N_s} W_i^c (\dot{\chi}_t^i - x_{t|t-1})(\xi_t^i - \mu_t)^T \tag{3.43}$$

$$K_t = C_t S_t^{-1}, \tag{3.44}$$

respectively. Lastly, the updated mean $x_{t|t}$ and the updated covariance matrix $P_{t|t}$ are calculated with

$$x_{t|t} = x_{t|t-1} + K_t (z_t - \mu_t) \tag{3.45}$$

$$P_{t|t} = P_{t|t-1} - K_t S_t K_t^T, \tag{3.46}$$

where z_t is the measurement for the estimation time step.

3.4.3. The Particle Filter (PF)

In the particle filter (PF), lots of points called "particles" are used for estimation [28]. Utilization of these particles makes it easier to estimate an approximate non-Gaussian distribution. The PF is one of the most popular algorithms in nonlinear tracking.

In the first step of the PF, particle vectors are generated using an initial state vector and an initial covariance matrix. Then, initial weight values are assigned to each vector. In general, all particle vectors have the same weight in the beginning. We can think of it as those particles and their weights defining a particle set Ψ_t^p at time step t, as

$$\Psi_t^p = \{\dot{v}_t^1, \dot{w}_t^1, ..., \dot{v}_t^n, \dot{w}_t^n\} \qquad n = 1, ..., N_p, \tag{3.47}$$

where N_p represents the total number of particles, and $\dot{v}^n$ and $\dot{w}^n$ are the vector and the weight of the particle n, respectively.

At each time step, those particles are transferred from the previous time step to the current time step using a state transition function f as

$$\dot{v}_t^n = f(\dot{v}_{t-1}^n) \qquad n = 1, ..., N_p. \tag{3.48}$$

The weights of the particles are updated with the measurement z_t as

$$\dot{w}_t^n = \dot{w}_{t-1}^n p(z_t|\dot{v}_t^n) \qquad n = 1, ..., N_p. \tag{3.49}$$

In this equation $p(z_t|\dot{v}_t^n)$, can be thought of as the measurement likelihood. Finally, the estimation of the PF $x_{t|t}$, is calculated with the weighted sum of the particles by

$$x_{t|t} = \sum_{n=1}^{N_p} \dot{w}_t^n \dot{v}_t^n. \tag{3.50}$$

But, particle weights should be normalized before this computation.

One of the most important points in the PF is observing the weights of the particles. As time passes, some particle weights shrink to zero, and their contributions become negligible. So, the performance of the PF decreases when it uses a smaller number of particles. This problem is known as particle degeneration in the literature. In order to avoid this problem, particles should be regenerated from the current estimation with resampling when needed. The need for resampling can be discovered by checking the value of effective particle counts

$$\overline{N_t} = \frac{1}{\sum_{n=1}^{N_p} (\dot{w}_t^n)^2}. \tag{3.51}$$

If this value is below a determined threshold, it means that particle degeneration is likely. Lots of resampling algorithms are present in the literature. Examples of comprehensive studies about resampling are given in [29, 30]. In our study, we have implemented the commonly used systematic resampling method.

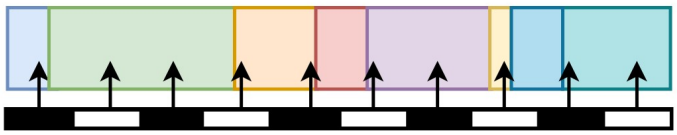

Figure 3.1. Visualization of systematic resampling method. Adapted from [30].

Systematic Resampling. In this method, it can be thought that the particles are represented as part of a weight line with lengths that are proportional to their weights, as in Figure 3.1. A random distance value (the first black arrow in the figure) from the starting point is selected from a uniform distribution in $(0, 1/N_p]$. The particle, whose distance from the starting point is the selected distance value, is taken as the first new particle. The other particles are selected using a fixed distance $1/N_p$, starting from the first selected distance value. Since the particles with higher weights cover longer parts on the weight line, they are likely to be resampled. After the resampling, all of the particles get the same weight as $1/N_p$.

3.5. Target Tracking Simulations

3.5.1. Evaluation Metrics

In order to evaluate the performances of tracking algorithms, we used two metrics, which are explained below:

3.5.1.1. Root Mean Square Error (RMSE). Root mean square error (RMSE) value is computed as,

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T}(\bar{x}_t - \bar{x}_{t|t})^2 + (\bar{y}_t - \bar{y}_{t|t})^2}{1/T}} \tag{3.52}$$

where T is the length of the estimated trajectory; $\bar{x}_t$ and $\bar{y}_t$ are the ground truth positions of a target on x and y coordinates at time step t, respectively; and $\bar{x}_{t|t}$ and $\bar{y}_{t|t}$ stand for corresponding estimated positions.

3.5.1.2. Mean Position Error (MPE). Mean position error (MPE) of the estimated trajectories for a desired time step t over simulations is computed with

$$MPE(t) = \frac{1}{N_m} \sum_{n=1}^{N_m} \sqrt{(\bar{x}_t^n - \bar{x}_{t|t}^n)^2 + (\bar{y}_t^n - \bar{y}_{t|t}^n)^2}, \tag{3.53}$$

where N_m represents the number of Monte Carlo simulations and n indicates the simulation number to which position or estimation belongs to. The definitions of the other variables are the same as given in Section 3.5.1.1. This metric was used to observe position errors based on time, since the metric, explained in Section 3.5.1.1, is calculated over entire estimated and ground truth trajectories.

3.5.2. Simulation Results

For target tracking simulations, the trajectory given in Figure 3.2 was created artificially. In tracking, measurements from two sensors were used. A white Gaussian measurement noise, with standard deviation (SD) $\sigma_z = 5$, was added to these measurements in order to simulate real world conditions.

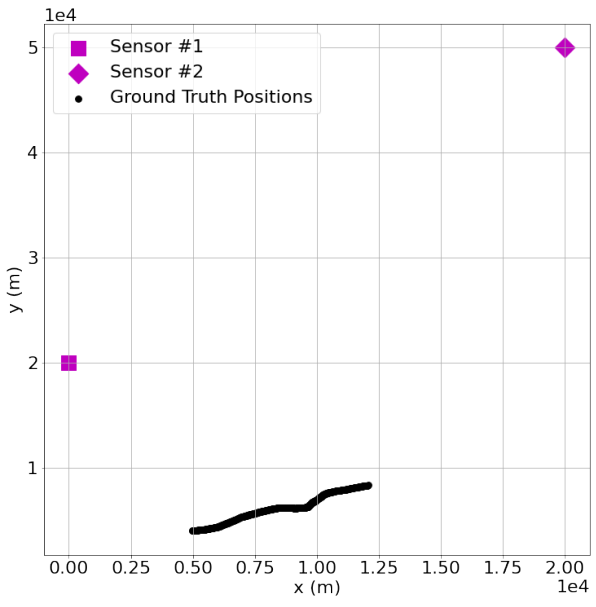

Figure 3.2. Artificially generated trajectory.

The MPE values of estimations over 100 Monte Carlo simulations, from different tracking algorithms, are given in Figure 3.3. It can be concluded from this figure that all the filtering methods were successful in noise reduction and tracking. It is clear that both the EKF and the UKF performed similarly well in tracking, whereas the PF showed relatively weaker performance.

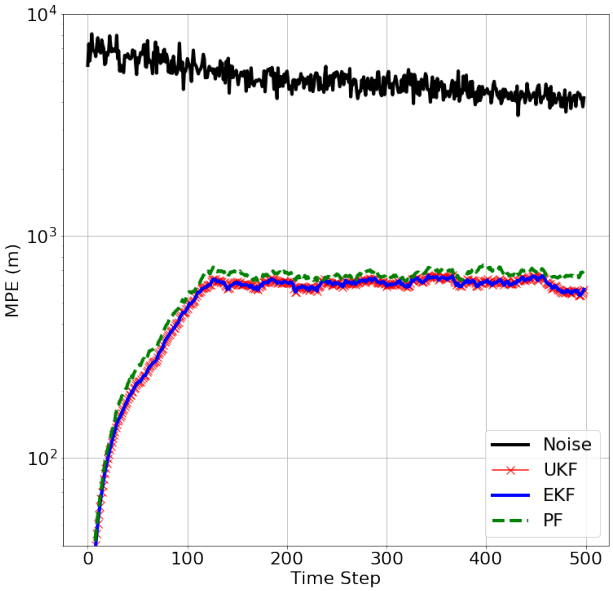

Figure 3.3. MPE values of different filtering methods.

In Table 3.1, the RMSE values of different filtering methods are presented. It can be observed from this table that the RMSE values are consistent with the MPE values of estimations. There is no significant difference between the errors of the EKF and the UKF in these values. The RMSE of the PF is a little higher, but it is still acceptable considering the noise effect.

In our experiments, we observed that the PF underperformed the other two filtering methods. In addition, the most important drawback of the PF is that it has the highest computation time in comparison with the other methods. The main reason for this amount of time is the requirement of many calculations for numerous particles. On the other hand, the EKF implementation was the hardest one due to the computations of the Jacobian matrices. In general, these computations are not straightforward.

Table 3.1. RMSE values of different filtering methods and noise.

	RMSE
UKF	1182.5838
EKF	1183.5435
PF	1918.6287
Noise	11440.2123

Even more, these Jacobian matrices may not exist in some systems that have discontinuities, or singularities, or discretized states and measurements. Moreover, EKF linearization is not successful in highly nonlinear systems in which linear approximation is not satisfactory [27].

In this study, we will be using the UKF in our follow-up tracking experiments. It was selected due to showing the best performance in our simulations, according to both metrics (RMSE and MPE) and having applicability to various kind of systems easily.

Additionally, we tested the UKF performance under different measurement noise conditions. The MPE values of the estimations are given in Figure 3.4, whereas the corresponding RMSE values are given in Table 3.2. Three different values are selected for the measurement noise SD σ_z: 1, 3, and 5.

Table 3.2. RMSE values of UKF with different SD values of measurement noise.

SD of the Measurement Noise	**RMSE of the UKF**
$\sigma_z = 1$	195.6332
$\sigma_z = 3$	314.2813
$\sigma_z = 5$	452.1720

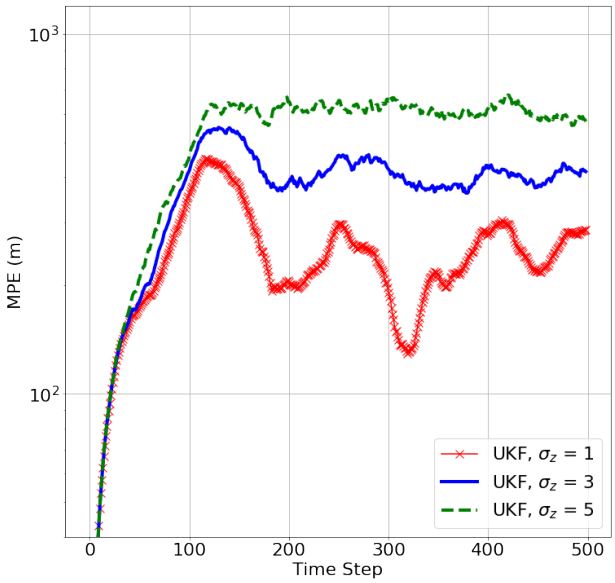

Figure 3.4. MPE values of UKF with different SD values of measurement noise.

Our results verified, as expected, that when the noise level is lower, the tracking performance gets better. Based on this observation, we will investigate several methods for noise reduction in Chapter 4, on measurements, to increase bearing-only tracking performance in our study.

4. TIME SERIES DENOISING

A time series is defined as a set that consists of sequential data points with time information. This time information can help reveal relations between time series data points, giving it an advantage over traditional datasets. To use time information, time series data should be processed by special tools. Thanks to its wide use in numerous real-world applications, such as values of stock options in the stock market or ECG signals in the medical field, time series have attracted the interest of academic studies. Lots of statistical tools are developed in those studies to satisfy needs in time series processing. Thus, it is wise to model time-dependent data as a time series for further analysis.

Examples of time series data are given in Figure 4.1. Time-based noisy bearing measurements (taken with a sensor) of a moving target on the sea, as well as the ground truth bearings, are depicted here. In the scope of this study, we will focus on only this kind of time series.

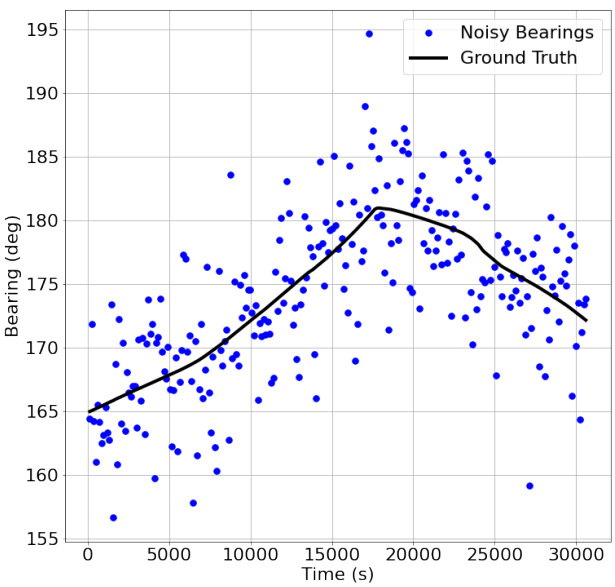

Figure 4.1. Examples of time series data.

In real-world scenarios, it is not usually possible to obtain data directly without the unintended effects of environmental conditions. When collecting bearing measurements, these effects can emerge from the sensor itself, or weather or geographical conditions. We can label all the differences from the original data points as noise, regardless of their sources. Noise should be removed before analyzing time series data in order to avoid making incorrect inferences. Denoising is an operation to decrease the effect of the noise or to remove it completely, if possible. We will give details about several denoising methods implemented in our study. In Section 4.1, filtering approaches, in which only previous data points and the current data points are used to clean the current data point, will be discussed. On the other hand, in some cases, we may have an opportunity to access data points from the future of the current data point. Considering these future data points, in addition to the previous ones, results in better denoising performance. This method is called smoothing in the literature. Several smoothing approaches implemented in our study will be given in Section 4.2.

4.1. Time Series Filtering

In this section, we will introduce filtering methods that are intended for time series denoising. These methods use only historical values and the current value to clean the current data point; however, they have the advantage of being able to be used in real-time applications.

4.1.1. Linear Regression

In machine learning (ML), linear regression is a method in which the relationship between two variables is modeled using observed data values. When only one variable is present, the linear regression model is used to estimate the corresponding variable. It can be identified as one of the simplest learning approaches. Moreover, other advanced regression based approaches, such as nonlinear regression and multiple linear regression, are discussed in the literature.

The linear regression model is expressed as a line equation as

$$y_{LR} = b_1 x_{LR} + b_0, \tag{4.1}$$

where x_{LR} is the predictor and y_{LR} is the dependent variable. Parameters of the model, b_1 and b_0, are calculated with

$$b_1 = \frac{n_{LR}\Sigma x_{LR} y_{LR} - \Sigma x_{LR}\Sigma y_{LR}}{n_{LR}\Sigma x_{LR}^2 - (\Sigma x_{LR})^2} \tag{4.2}$$

$$b_0 = \frac{1}{n_{LR}}\Sigma y_{LR} - b_1 \frac{1}{n_{LR}}\Sigma x_{LR}, \tag{4.3}$$

where n_{LR} is the number of the dependent variables that are used in the linear regression model, using a least squares solution [31].

As we can observe from example time series in Figure 4.1, bearings and corresponding time indexes have a kind of relation that could be captured by a linear regression model. In this study, we accept the measurement as a dependent variable and the time index as a predictor one. A batch of noisy measurements is used to find model parameters. These parameters ensure that the error, which is the sum of the distance between the data points and the fitted line, is as small as possible. We presume that deviations from the model line are noise in time series data and plan to use the linear regression model for estimating denoised data points in the future, which are positioned on the line. A sliding window approach can be used for selecting data points to calculate parameters in (4.2) and (4.3). For each data point, the historical data points in the window are considered in this calculation.

In Figure 4.2, the linear regression estimations for bearing measurements are shown. Here, the sliding window width was set as 15. We can conclude that the denoising effect is visible, because the estimated bearings are closer to the ground truth bearings than the received noisy bearings. The implementation of this method is simple and computationally inexpensive. The only disadvantage of this method is the requirement of having many measurements to perform well in denoising.

In a linear regression application, window width selection is critical and may be dependent on the characteristics of data. Thus, it is difficult to select a single window width value suited for different applications.

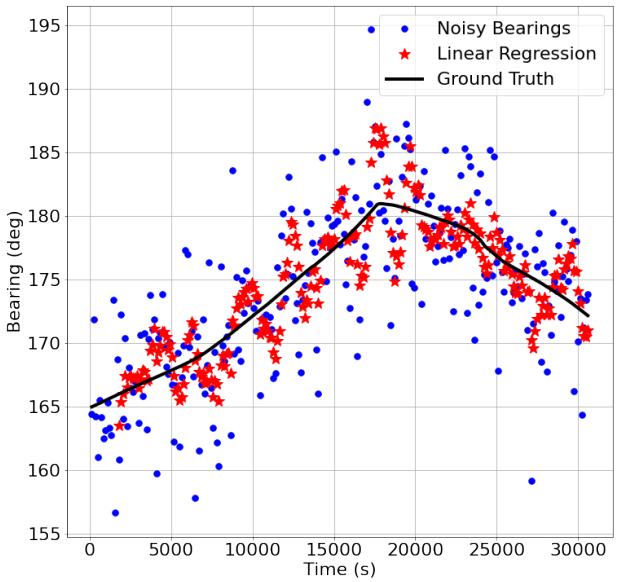

Figure 4.2. Linear regression results.

4.1.2. Moving Average Filtering

In order to reduce the effect of noise on data, the first solution that may come to mind is an approach that takes the average of several data points in the neighborhood of the current one and uses this average instead of directly using a single data point. This simple yet effective solution is called the moving average filter in the literature.

The weights of all the data points are the same in the moving average filtering. But for some applications, a special weighting scheme may be desired (e.g., the Savitzky-Golay filter uses polynomial fitting via the least squares solution to find a weight coefficient for each data point [32]).

A comparison study for commonly used moving average filtering approaches on the problem of forex forecasting in finance is given in [33]. In this study, it was stated that the weighting strategies that assigned linearly or exponentially increasing weight values to the later data points resulted in a better solution to the addressed problem.

In implementations of this method, a sliding window is used to select data points for averaging in the moving average filtering method. The width of this window should be set as a suitable value for the application by trying several values and observing the results. In our implementation with this filtering method, we placed the end of the window on the current data point and used only the previous points to average. Filtering by placing the sliding window in this way is also known as the trailing moving average [34].

We can express the moving average filtering mathematically as

$$y_t^{MA} = \frac{1}{N^{MA}+1} \sum_{n=t-N^{MA}}^{t} x_n^{MA}, \tag{4.4}$$

where N^{MA} represents the window size. Subscripts indicate the time step which the data point x^{MA} belongs to. Finally, y_t^{MA} is the result of the moving average filtering operation at time t.

From Figure 4.3, we can observe that even a simple averaging operation with the moving average filtering can yield acceptable results in noise reduction. The sliding window width was set to 15 in our implementation. It is clear that this denoising method outperformed the method introduced in Section 4.1.1 (linear regression). In general, there are no sudden changes in estimates; we can say that the estimated points are consistent with each other. Implementing this method is also simple and computationally inexpensive.

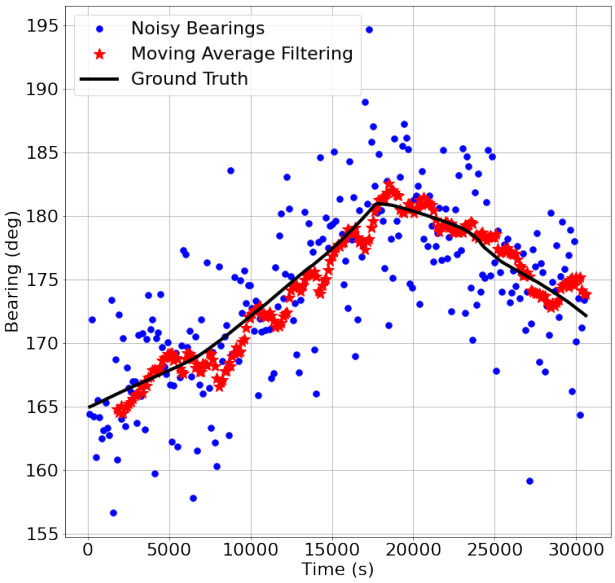

Figure 4.3. Moving average filtering results.

4.1.3. Exponential Smoothing

Exponential smoothing, one of the simplest approaches for time series denoising, was proposed by Charles C. Holt in 1957. The reprinted version of the original report, in which this smoothing concept was introduced, is given in [35]. Despite the fact that its name includes smoothing, we are addressing this method in Section 4.1 (Time Series Filtering) because it only uses past observations, like the other filtering approaches discussed here.

In one study, the exponential smoothing method was used for removing sudden changes in traffic flow data before training a neural-network based traffic flow forecasting model in [36]. It was stated that the exponential smoothing step prevented learning of sudden changes by the model and helped to train a model with more generalization capability. In another study, the exponential smoothing method was used as one of the time series prediction sub-methods that provide input to the proposed neural network model by the authors for the vehicle traffic flow estimation [37].

Moreover, an extended version of the exponential smoothing that can be used for prediction of time series with seasonal patterns or trends was proposed in [38].

In the exponential smoothing, all the previous data points contribute to estimate the noise-free current data point as

$$y_t^{ES} = \alpha^{ES} x_t^{ES} + (1 - \alpha^{ES}) y_{t-1}^{ES}, \tag{4.5}$$

where y_t^{ES} is the result of the exponential smoothing at time t; x_t^{ES} is the current data point at time t; a smoothing parameter, α^{ES}, is used for weighting these contributing data points. α^{ES} can be equal to any value in the interval $[0, 1]$. We can think of it as this parameter controlling how much we use the current and the previous information. It is wise to use small values as a smoothing parameter in highly noisy data to decrease the contribution of the current data point, but it should be determined by application requirements. It can be seen in

$$\begin{aligned} y_t^{ES} &= \alpha^{ES} x_t^{ES} + (1 - \alpha^{ES})[\alpha^{ES} x_{t-1}^{ES} + (1 - \alpha^{ES}) y_{t-1}^{ES}] \\ &= \alpha^{ES}[x_t^{ES} + (1 - \alpha^{ES}) x_{t-1}^{ES} + (1 - \alpha^{ES})^2 x_{t-2}^{ES} + ... + (1 - \alpha^{ES})^{t-1} x_1^{ES}] \\ &\quad + (1 - \alpha^{ES})^t x_0^{ES}, \end{aligned} \tag{4.6}$$

that measurements from the first time step to the current time step are used in a manner of weighted averages. The weights get smaller exponentially while going back in time, which is the reason of such naming for this method.

The result of denoising by exponential smoothing on sample time series data can be seen in Figure 4.4. The smoothing parameter was set as 0.25 in our experiment. It is clear that the denoised data points have closer values to the ground truth and smaller deviations than the noisy data. We can say that this method was able to estimate the ground truth bearings with a bias. Because this method has high computational efficiency and does not require keeping all of the previous data, it can work in real time without any further requirements.

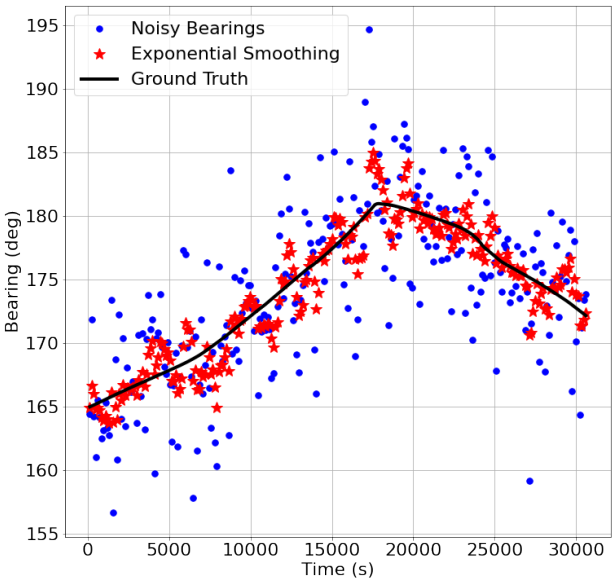

Figure 4.4. Exponential smoothing results.

4.1.4. Deep Learning-Based Filtering (DL-Based Filtering)

ML-based approaches improve results that are solved through traditional methods. We can define a ML method as a function that is built using training data in the training process, and that takes an input and produces an output [39]. In general, training data is transformed into special feature vectors that are used in the training, depending on the application requirements, in the step called the feature extraction. For example, one can transform colored images into HSV (for hue, saturation, value) vectors to use them as inputs in the training process of a color-based image classification application. Again, in this example application, the learning function that results from the training process can be used to make inferences about new images that were not included in the training data, but these new images also need to be transformed into vectors with the same feature extraction step of the training process.

Deep learning (DL) is one of the most popular subfields of ML-based approaches. Here, the learning function, which is used for producing the expected output, can consist of numerous sub-functions that are known as layers of a DL model.

This layered learning architecture, which is the main reason for the usage of the word "deep" in this method's name, extracts higher-level patterns from the data [40]. DL architecture can be thought of as a more complex version of the multilayer perceptron (MLP) that is based on the concept introduced in [41], the ancestor of modern DL networks. Here, the term "complexity" refers to increasing layer sizes, the number of neurons, or the number of nodes. In DL, unlike common ML algorithms, specific feature extraction steps are not necessary. Feature extractions are assumed to be performed by deep layers implicitly. Generally, training data is given to a deep neural network after several simple preprocessing steps, such as normalization of the data values or ensuring the representation of the data with certain length vectors. Complex problems in which the feature extraction can not be performed with ease are the main interest in DL studies.

A DL-based solution for one-dimensional (1D) data, in which the input data is processed in deep layers consecutively, is known as a deep feedforward network, whereas for two-dimensional (2D) data and special types of 1D it is known as a convolutional neural network (CNN) [40]. Apart from these applications, a DL-based solution can be used in processing sequential data (sequential data can be defined as a set of data points which have a specific kind of dependencies to each other, i.e., stock prices, texts, speech signals, or any time series data). The crucial thing here is that patterns between data points are also to be considered in data processing instead of just data values. Recurrent neural network (RNN) is a DL-based solution that performs processing tasks specially on sequential data [42]. In the scope of this study, we will be focusing on this special kind of DL method.

Image captioning, machine translation, sentiment classification, and object tracking are popular research topics in RNN studies in the literature. A time series prediction study with RNN is given in [43]. In this paper, a preprocessing step that removes outliers in the training set is proposed for training the RNN. This preprocessing step operates like the EKF [26] and filters the training data to obtain robust data estimations.

It was confirmed with the training simulations with both filtered and unfiltered data that the proposed preprocessing step boosted the performance of RNN on estimations in comparison with a conventional neural network with the least squares fitting. In another study, the RNN-based phoneme recognition method that had the best recognition score on the TIMIT dataset [44] at the time was proposed in [45]. In this paper, deep RNN that has multiple layers for capturing higher level representation of the data were used for the first time to solve the addressed problem.

In RNN training, two approaches, the connectionist temporal classification [46] and the RNN Transducer [47], were used with a regularization method that adds noise to model weights in order to achieve the end-to-end training. Training with those approaches lacks input-output alignment necessity, which severely limits the usage of RNN on this problem. It was stated that the proposed speech recognition method showed state-of-the-art performance on the experimented dataset. In [48], a RNN-based sentence embedding scheme (a sentence embedding scheme generates an embedding vector that consists of semantic information of a sentence, later these vectors can be used in numerous applications, such as sentiment classification or machine translation) for natural language processing (NLP) applications was proposed. In this study, the RNN-based embedding method was used for information retrieval from the web. The proposed method was compared with widely used state-of-the-art DL-based and statistical methods on the addressed problem of the paper. It has been shown that this method outperformed its competitors due to its superior ability to select useful information and discard useless information. The mean of the normalized discounted cumulative gain (NDCG) [49] was used as the evaluation metric in this comparison. Additionally, it has been stated that the proposed method is robust against noise, as it mainly focuses on keywords in the embedding process. The visual analysis of the embedding operation is given in order to investigate how embedding is being performed by neural networks.

In Figure 4.5, an example architecture of RNN is presented. In order to estimate y^{DL}, a batch of x^{DL}'s (here $N^{DL}+1$ x^{DL}'s in total), are fed to the neural network.

Here, subscripts denote time indexes. h^{DL}'s represent hidden states that carry the contribution of previous data points forward to the current time step. Finally, W^{DL}'s are weights for corresponding states that are indicated in their subscripts.

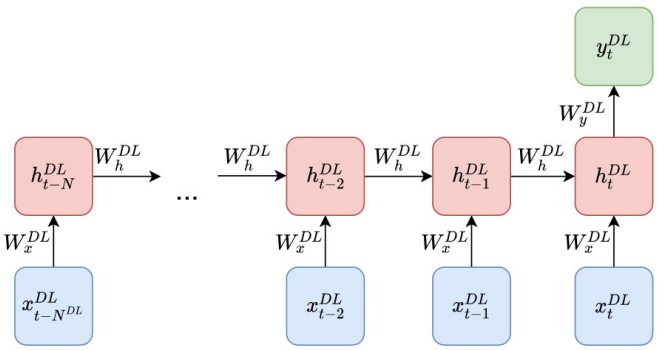

Figure 4.5. Architecture of RNN. Adapted from [50].

In considering applications requirements, the several types of RNN can be classified as one-to-many, many-to-many, or many-to-one, as shown in Figure 4.5. In this section of our study, we will be using many-to-one RNN for filtering time series data.

When conducting training of DL models, loss value of a desired loss function (i.e., mean squared error loss or binary cross-entropy) is calculated with the model parameters at each training step. Then these parameters are updated using the gradient of the loss function, with respect to weights, in order to minimize the computed loss. In RNN, this gradient is computed by multiplying the gradients of layers by the chain rule. If these layers' gradient values are greater than 1, the total gradient will have a much greater value due to numerous multiplications. On the contrary, if these gradient values are less than 1, the total gradient will have a much smaller value. These problems are known as exploding gradients and vanishing gradients, respectively [40]. Having extreme gradient values makes it impossible to update parameters and, finally, training. Several methods, such as using different activation functions or initial parameter selection in a smart way, exist in the literature to overcome these problems. Below, we will focus on as one of these methods, long short-term memory (LSTM) network, which has extra gates in the architecture for controlling information flow.

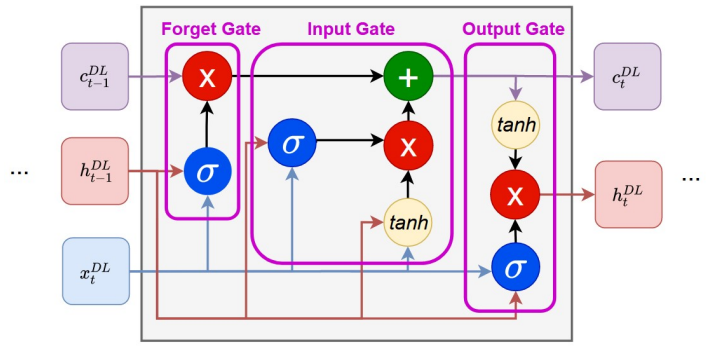

Figure 4.6. One unit of LSTM. Adapted from [52].

4.1.4.1. Long Short-Term Memory (LSTM). Because of the vanishing and the exploding gradients problems that are described above, simple RNN has problems with capturing long-term dependencies. These dependencies are critical for most of the sequential data processing applications (e.g., the first word of a sentence may contain important information in order to predict a word at the end of this sentence).

LSTM, which was proposed in [51], is designed for enabling useful long-term dependencies. In Figure 4.6, one unit of LSTM is given (multiple units can be concatenated to capture long-term dependencies). Unlike simple RNN, here an extra state called the "cell state" (c^{DL} in Figure 4.6) and special gates are utilized. The cell state is the memory of the network, whereas the gates control information flow from past to future. The tasks of each gate, in one cycle of information flow in LSTM, can be expressed as follows:

- Forget Gate: Selects the information for removal from the cell state or keeping it in the cell state.
- Input Gate: Updates the relevant information in the cell state.
- Output Gate: Determines the next hidden state using the previous hidden state and the current input.

Sigmoid and tanh functions, multiplication, and addition operators are used in these gates to perform the tasks described above.

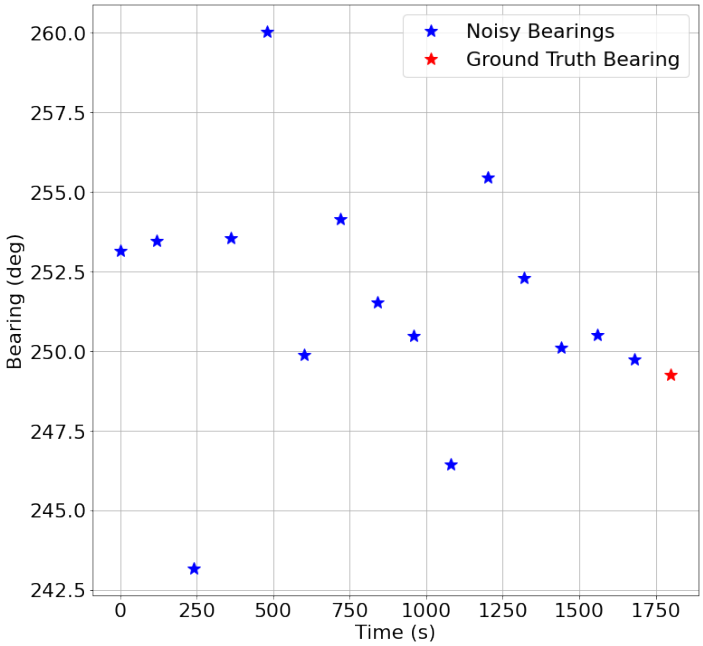

Figure 4.7. Example from DL-based filtering training dataset.

In our study, we have trained LSTM-based DL model to denoise bearing measurements. In the training process, the noisy bearing measurements until the current time were used as inputs to the network, whereas the ground truth bearing of the current time step was used as their label. An example from our training data is plotted in Figure 4.7.

The results of DL-based time series filtering are presented in Figure 4.8. It is clear that this filtering method was able to clean data with acceptable results. This method can be selected for any denoising applications, provided that application-based model training is performed.

4.2. Time Series Smoothing

In this section, we will introduce smoothing methods for time series denoising. In order to clean the current data point, not only past observations but also observations from the future will be used.

We can give examples of smoothing problems as follows:

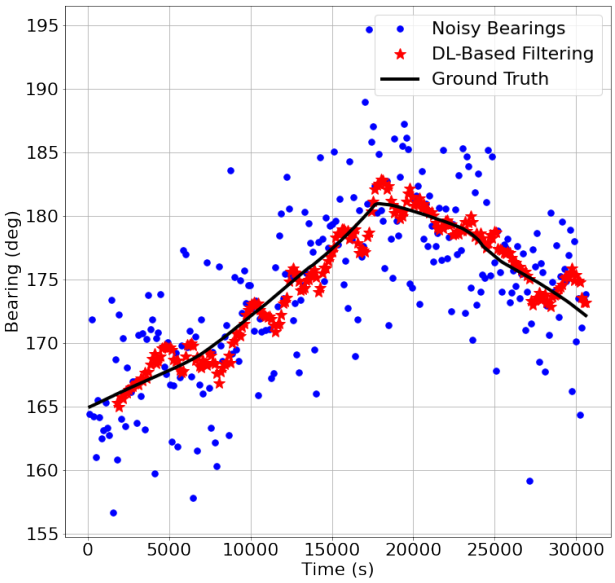

Figure 4.8. DL-based filtering results.

- estimation of the trajectory of a space vehicle tracked by a radar when the all radar measurements are available [53],
- removing noise from images or speech signals,
- determining history of a chemical or biological process using noisy measurements.

4.2.1. Moving Average Smoothing

The sliding window, which was described in Section 4.1.2 (moving average filtering), is used as being centered at the current data point in order to use the previous and the future values in the moving average smoothing. The smoothing with placing the sliding window like as explained, also known as the centered moving average in the literature [34]. We can express this process mathematically as

$$y_t^{MA} = \frac{1}{2N^{MA}+1} \sum_{n=t-N^{MA}}^{t+N^{MA}} x_n^{MA}. \tag{4.7}$$

It should be noted that the future values (belonging to $t+1$, $t+2$, ..., $t+N^{MA}$ time steps) are also considered in this method unlike the filtering method from Section 4.1.2 (moving average filtering).

From Figure 4.9, it can be inferred that denoised estimations are closer to the ground truth bearings than the estimations belonging to filtering approaches we have discussed so far. The sliding window width was set as 15 in our implementation. There are negligible fluctuations in estimations. It is impressive that, the estimations are consistent with the ground truth bearing even when there are sudden changes in the ground truth values.

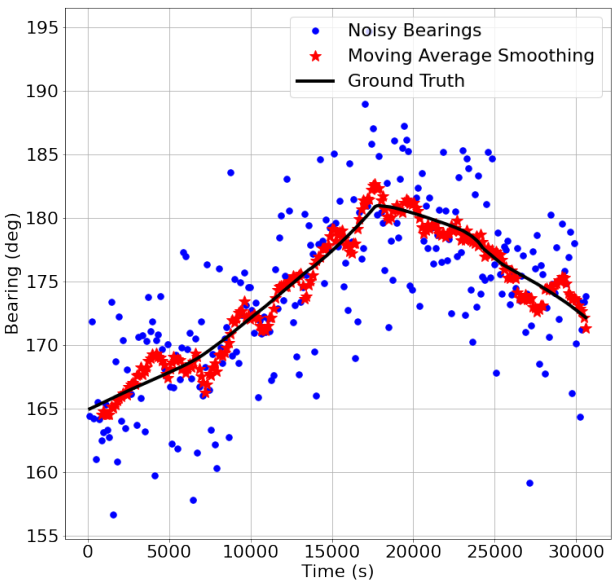

Figure 4.9. Moving average smoothing results.

4.2.2. Gaussian Smoothing

Gaussian smoothing, was proposed in [54], is widely used for noise removal in computer vision implementations. Any weighting coefficients were not used in Section 4.2.1, when summing all the data points from different time steps; however, in this method, we use discrete values sampled from a Gaussian distribution for assigning weights to these data points.

The current data point takes the highest weight, and this weight decreases gradually when moving forward and backward in time. This smoothing approach can be thought of as a weighted moving average smoothing.

In one study, Gaussian smoothing for the edge detection problem in image processing was discussed in [55]. A smoothing method is required for eliminating noise in the images in order not to detect them as edges, because an edge detection algorithm tries to find high frequency components, that mainly include edges and sometimes noise, of images. It was stated by the author that, Gaussian smoothing has few flaws such as removal of edges or detecting unreal edges in linear cases. For nonlinear cases, in which Gaussian smoothing boosts edge detection performance significantly, implementation of Gaussian smoothing requires heavy computations due to its implementation complexity.

Gaussian smoothing can be expressed mathematically as

$$y_t^G = \frac{1}{2N^G + 1} \sum_{n=t-N^G}^{t+N^G} x_n^G w_n^G. \tag{4.8}$$

In this equation, y_t^G is the denoised data point at time t, N^G is the number of data points used from the previous and future time steps, and w_n^G is the weight that is assigned to a data point x_n^G, which is one of the data points around the current data point, which can be computed by

$$w_n^G = \frac{1}{\sigma\sqrt{2\pi}} e^{-(n-t)^2/2\sigma^2}, \tag{4.9}$$

with a σ value for the SD of the Gaussian distribution.

In Figure 4.10, the Gaussian smoothing result on example time series is given when N^G value is set as 7 (15 bearings were used for each time step). From this figure, we can conclude that this method underperformed the moving average smoothing method by visual inspection.

The estimated points have high variations as the ones belonging to the denoising approaches discussed in Section 4.1 (time series filtering).

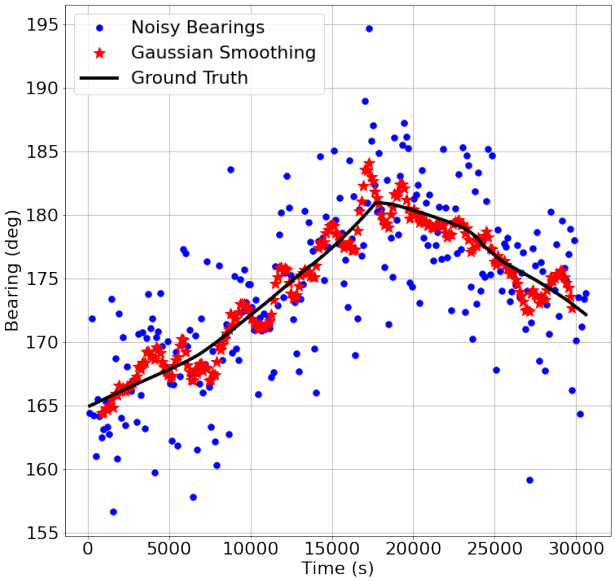

Figure 4.10. Gaussian smoothing results.

The critical step in this method is selecting the SD of the Gaussian distribution that is used for weight computation. If it is set too high, the weight coefficients have closer values. Thus, the denoising result gets similar to the result of denoising with the method discussed in Section 4.2.1 (moving average smoothing). On the other hand, if it is set too low, the contribution of the current data point dominates the denoising result, and the performance of denoising process decreases when there are high noise effects on data points. Because in this case, it is very likely that the current data point is deviated from the ground truth significantly. The calculated weight values used in our implementation, when σ is set as 4, are illustrated in Figure 4.11.

4.2.3. Wavelet Smoothing

We have high resolution in time while working on a time series data, as illustrated in Figure 4.12. This condition enables us to implement lots of useful statistical tools in time domain.

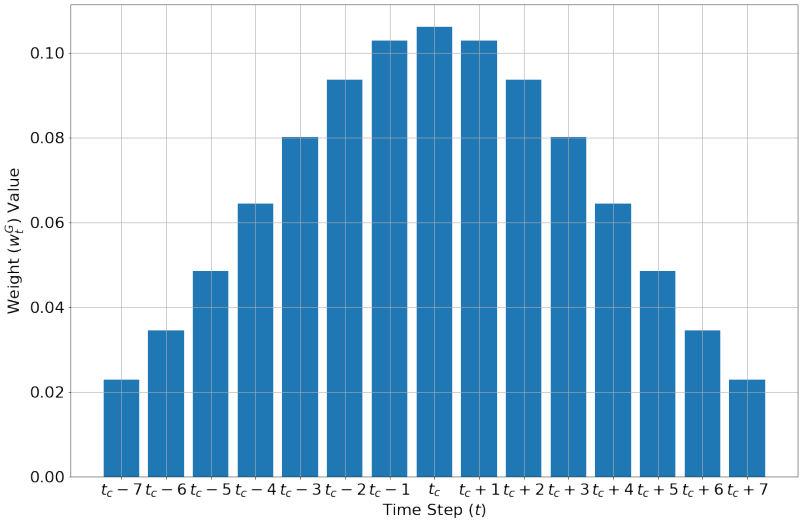

Figure 4.11. Weight values used in this study for Gaussian smoothing.

But transformations of domains are also to be considered for solving some specific problems with ease. In the scope of our study, we will focus on "time domain to frequency domain transformation" as one of these transformations in order to solve noise problem in data processing.

In frequency domain, a signal's noise and information components are more separable than their time domain equivalents. That is why a band pass filter can eliminate noise components easily, after transformation of a time domain signal into this domain. In general, following this kind of filtering, the noise-free signal is reconstructed from the signal's remaining frequency components.

Fourier analysis, that is named after Joseph Fourier, can be accepted as a main method for frequency domain operations. In this analysis, a signal, which is in the time domain, is defined as a sum of sinusoids (these sinusoids are orthogonal basis functions in the frequency domain) with different frequencies and amplitudes in the frequency domain.

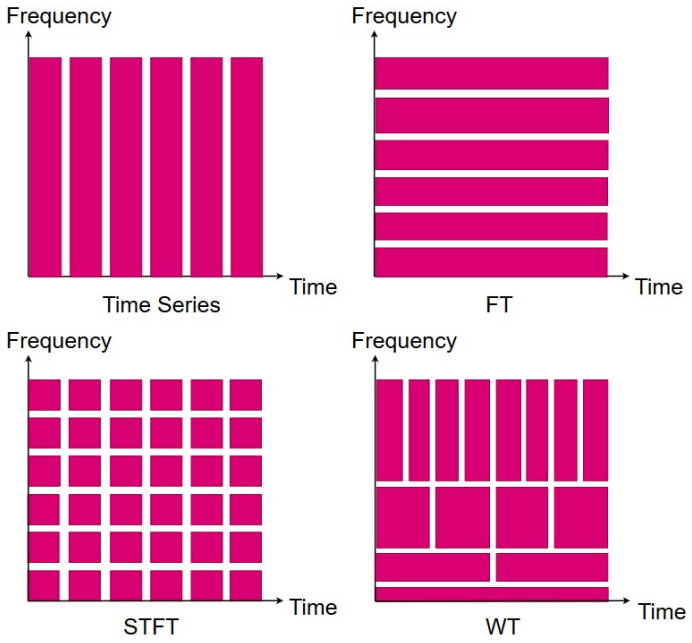

Figure 4.12. Comparisons of time-frequency resolutions of different approaches. Adapted from [56].

The Fourier transform (FT) $S_{FT}(\omega)$ of a continuous-time signal $s(t)$ is computed as

$$S_{FT}(\omega) = \int_{-\infty}^{+\infty} s(t)e^{-j\omega t}\,dt, \tag{4.10}$$

whereas the inverse of this transformation is computed as

$$s(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} S_{FT}(\omega)e^{j\omega t}\,d\omega. \tag{4.11}$$

For the sake of simplicity, we will go over only continuous-time signals' transformation, but this transformation can also be applied to discrete-time signals. In fact, in real world applications, these transformations mostly occur in discrete time, but intuition is the same in transformations of both types of signals.

The FT results in having high resolution in the frequency domain but no resolution in the time domain. This result is illustrated in Figure 4.12. Signals, which have differences in time but with the same frequencies, have the same FT result.

This common problem limits the usage of frequency domain analysis with the FT, especially for the non-stationary signals (statistical features of the signals, such as a mean and a variance, change over time in non-stationary signals). One way to increase time resolution, in FT, is partitioning the signal into smaller groups and applying the FT to each group separately. So one can analyze the frequency representations that are roughly time localized. This method, was proposed in [57], is known as the short-time Fourier transform (STFT). An illustration of resulting resolutions, in time and frequency, are given in Figure 4.12. In this method, a window function w_{STFT} is used for selecting one part of the signal $s(t)$ while zeroing out values in other parts. This window is shifted throughout the entire signal by a parameter τ, to compute the STFT. The STFT equation is given by

$$S_{STFT}(\tau, \omega) = \int_{-\infty}^{+\infty} s(t) w_{STFT}(t - \tau) e^{-j\omega t}\, dt. \tag{4.12}$$

Another transformation approach for localization of the frequency components in time is the wavelet transform (WT) [58]. Unlike the FT, the orthogonal basis functions, which are called as wavelets in this transform, are not just sinusoids. The WT is computed with

$$S_{WT}(a_{WT}, b_{WT}) = \frac{1}{\sqrt{a_{WT}}} \int_{-\infty}^{+\infty} s(t) \Psi \left(\frac{t - b_{WT}}{a_{WT}} \right) dt, \tag{4.13}$$

where Ψ is a mother wavelet, and a_{WT} and b_{WT} are parameters for scaling and shifting this mother wavelet in the WT. A lot of mother wavelet options such as Daubechies wavelet or Haar wavelet are available for different type of signals [59].

Using wavelets in different scales and positions yields in high resolution in both time and frequency domain simultaneously. Unlike the STFT that has a fixed size window function, several window sizes exist in a single WT. It helps us to have multi-scale frequency domain representation, as seen in Figure 4.12.

In order to use the WT in noise removal, resulting wavelet coefficients in different scales are used. Coefficients, which have small amplitudes, are considered as belonging to noise and eliminated with one of the several threshold methods that were proposed in the literature. Following that, the noiseless signal is reconstructed with remaining coefficients [60].

An image denoising method for functional magnetic resonance images (fMRI), using wavelets, is presented in [61]. Here, threshold operation was performed on 2D wavelet coefficients for image denoising instead of 1D ones as seen in common applications. Wavelet-based denoising and Gaussian smoothing, that is a conventional fMRI denoising method, were compared in terms of SNR and the false discovery rates that belong to the statistical parametric mappings of the data. It has been stated that if the SNR is too low, Gaussian smoothing is preferable due to the fact that its strong smoothing effect. On the other hand, if the SNR is high or acceptable, wavelet-based denoising is a good choice since it does not cause serious deformations unlike Gaussian smoothing. A comparison study about the WT-based denoising methods on EEG signals using different types of wavelets, is given in [62] whereas a study about increasing the WT-based denoising method's performance with an extra post-processing step on wavelet coefficients with Singular Vector Decomposition (SVD), is given in [63].

Another common application with the WT in the literature is data compression. A WT-based data compression method for the smart grid is proposed in [64]. It was stated this compression method has the advantage of being able to operate in real time as well as denoising certain type of noise. The wavelet data compression can be performed easily by discarding negligibly small wavelet coefficients following the WT. Later data can be reconstructed with the remaining coefficients. The wavelet type and decomposition level that have the highest wavelet energies were selected for the WT. The performance of the proposed method was verified with experiments on simulated data via observing values relating data compression and denoising.

In Figure 4.13, wavelet smoothing result on example time series is illustrated. A batch of noisy observations with 150 bearings were used at each time step. We can say that the performance of this method is remarkable in comparison with other methods have been discussed in this study so far. The estimates are close to each other for most of the time.

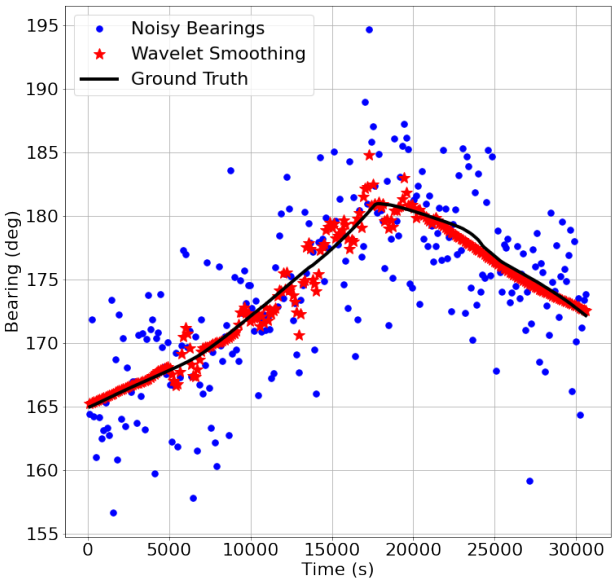

Figure 4.13. Wavelet smoothing results.

4.2.4. Deep Learning-Based Smoothing (DL-Based Smoothing)

In this smoothing method, an extension of the RNN, which was introduced in Section 4.1.4, namely, the Bidirectional Recurrent Neural Network (BRNN) [65] is used for data denoising. The difference here is that the input data is used as input of two separate layers, which are forward and backward layers. The forward layer behaves as traditional RNN. The information flows from past to future time steps. However, in the backward layer, the information flows backward in time. This layer can be thought of as the forward layer, which has the flipped data as an input. The both layers contribute to the output of the BRNN.

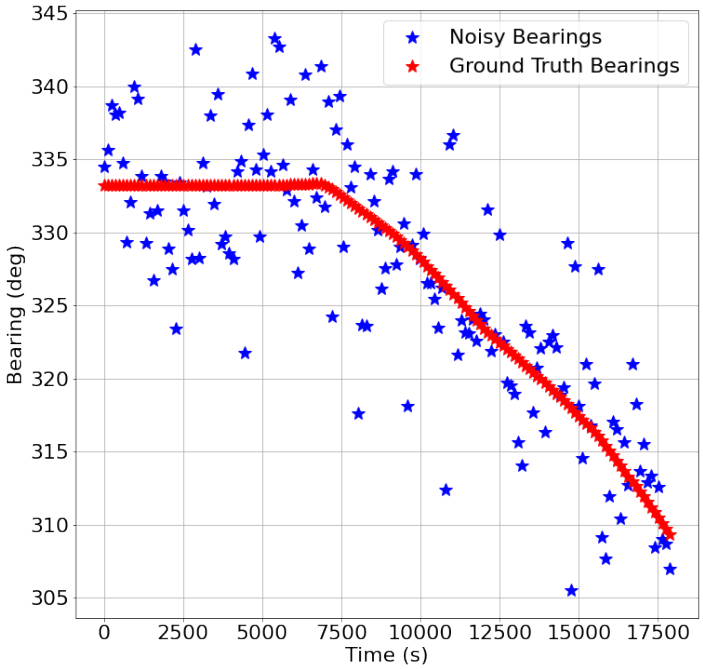

Figure 4.14. Example from DL-based smoothing training dataset.

In order to avoid problems of traditional RNN, that were discussed in Section 4.1.4, LSTM architecture was used in this smoothing method. The Bidirectional LSTM (BLSTM), was proposed in [66], is a modified version of LSTM based upon the intuition of BRNN. In the scope of our study, BLSTM-based DL model was used to estimate ground truth bearings. A batch of noisy observations (150 bearings were used at each time step) was used as input, whereas their ground truth values were used as their labels in training of neural networks. A sample from our training set is given in Figure 4.14.

The resulting smoothing performance of this method is presented in Figure 4.15. It is obvious that, the estimates are really close to the ground bearings, and they are consistent with each other as expected in a real case scenario. There are almost no bias in estimations except for the cases when there are sudden changes in bearings. Since it is not very likely to have these sudden changes in bearing measurements when tracking on the sea, this method seems promising for most of the cases in our bearing-only tracking problem.

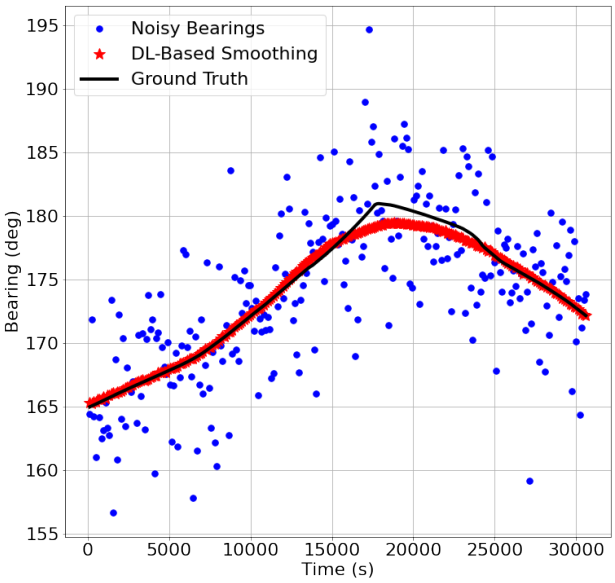

Figure 4.15. DL-based smoothing results.

5. EXPERIMENTS AND RESULTS

5.1. Testing Environment

We implemented all algorithms in the Python programming language to evaluate their performances. In order to show the results of a possible real-world application of our study, we used real data, detailed information of which is given in Section 5.1.1.

5.1.1. Dataset

The automatic identification system (AIS) is one of the systems that ensures voyage safety on the sea. In this system, AIS messages that consist of information of interest about the vessel's voyage to other vessels and vessel traffic services (VTS), such as vessel's speed or vessel's destination port, are shared automatically with interested parties. The AIS requires installation of two devices on vessels: the AIS transponder and the AIS receiver. The AIS transponders send AIS messages either to VTS stations when the vessel is near coastal areas or to satellites when the vessel is on the open sea, whereas the AIS receivers are used to collect these transmitted AIS messages. The AIS enables information sharing without being affected by weather conditions or being restricted by distance. It is a lifesaving device on the sea, because some communication appliances, such as ones use ultra-high frequency (UHF) waves, offer very short transmission range.

The requirements for carrying navigational system and equipment are regulated with Safety of Life at Sea (SOLAS) regulation V/19 introduced in 1974. As a revision to this regulation, the International Maritime Organization (IMO) introduced an addendum in 2000 that required all ships to carry the AIS that provide information to other ships and to coastal authorities [67]. According to this new regulation, AIS devices should be deployed in the ships meeting one of the following conditions by 31 December 2004:

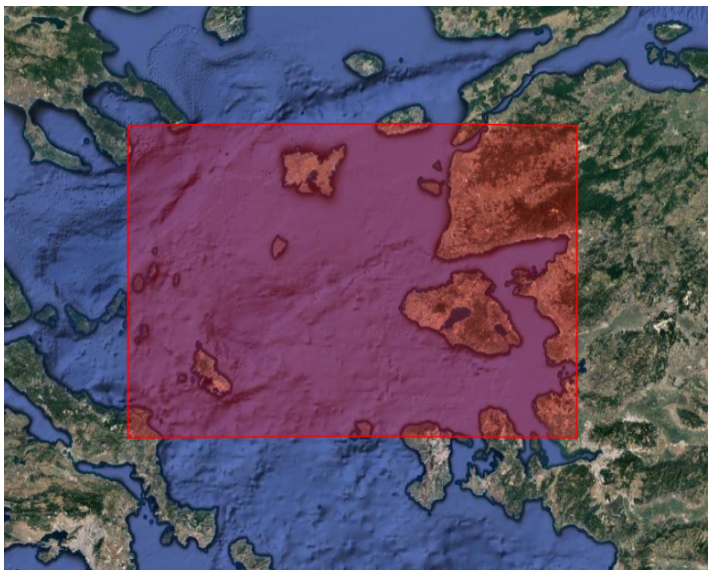

Figure 5.1. Coverage region of collected AIS data.

- all passenger ships, regardless of their size;
- all ships of 300 gross tonnage and above that are engaged on international voyages;
- all cargo ships of 500 gross tonnage and above that are not engaged on international voyages.

For our experiments, we collected AIS messages over the course of three days using data services from [68]. The collected AIS data was used to obtain vessels' trajectories on the sea. The coverage region of the collected AIS data is illustrated as a red square in the Aegean Sea in Figure 5.1.

AIS messages from fixed locations such as VTS, sea buoys, and anchored vessels, as well as AIS messages from vessels that have a mean speed of below 1 knot were removed from the collected AIS data. Moreover, the AIS messages of the vessels with less than 200 AIS messages in total were ignored, since our wavelet and DL-based smoothing approaches require at least 150 data points for denoising. The remaining vessel trajectories were plotted on maps in Figure 5.2 and Figure 5.3. We used Day #1 and Day #2 AIS data, which consisted of 55 different trajectories in total, as a training dataset for our DL-based filtering and smoothing approaches. We used Day #3 AIS data, which consisted 34 different trajectories in total, as test set in all simulations.

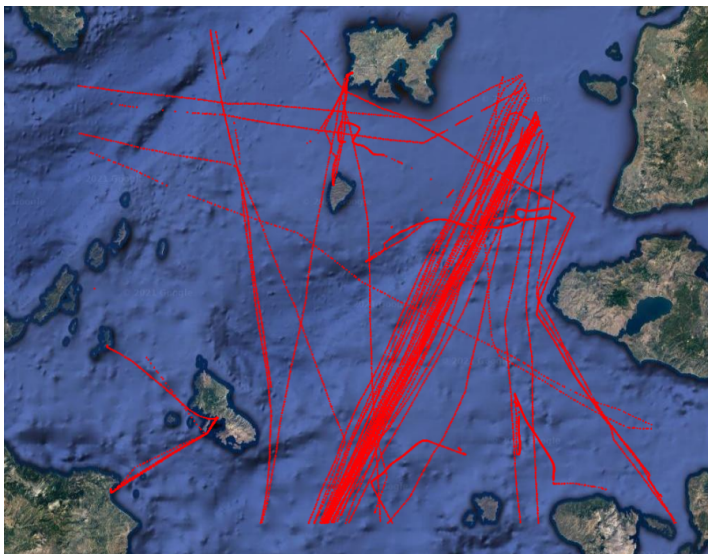

Figure 5.2. Day #1 and Day #2 vessel trajectories.

While preparing our dataset, vessels' positions in Global Positioning System (GPS) coordinates (ϕ (latitude), λ (longitude)) in the AIS data were transformed into Cartesian coordinates (x, y) using stereographic projection equations [69].

The equations for this transformation are

$$x = k.cos\phi.sin(\lambda - \lambda_0) \tag{5.1}$$

$$y = k[cos\phi_1.sin\phi - sin\phi_1.cos\phi.cos(\lambda - \lambda_0)] \tag{5.2}$$

$$k = \frac{2.R}{1 + sin\phi_1.sin\phi + cos\phi_1.cos\phi.cos(\lambda - \lambda_0)} \tag{5.3}$$

$$R_L = \frac{R_e.cos\phi}{(1 - e^2.sin^2\phi)cos\chi_C} \tag{5.4}$$

$$\chi_C = 2.tan^{-1}\left\{tan\left(\frac{1}{4}\pi + \frac{1}{2}\phi\right)\left(\frac{1 - esin\phi}{1 + esin\phi}\right)^{e/2}\right\} - \frac{1}{2}\pi, \tag{5.5}$$

where ϕ_1 is the central latitude, λ_0 is the central longitude, R_L is the local radius and χ_C is the conformal latitude. In these equations, the equatorial radius R_e is equal to 6378137, and the ellipticity of an oblate spheroid e is equal to 0.082095044176503. We used the interpolation method to fill missing position and speed values in the collected AIS data. An example of a vessel trajectory in both coordinate system are illustrated in Figure 5.4 and Figure 5.5.

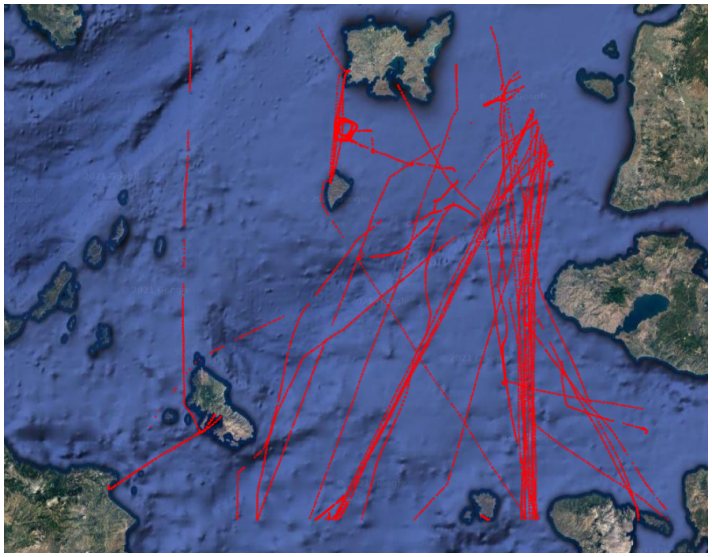

Figure 5.3. Day #3 vessel trajectories.

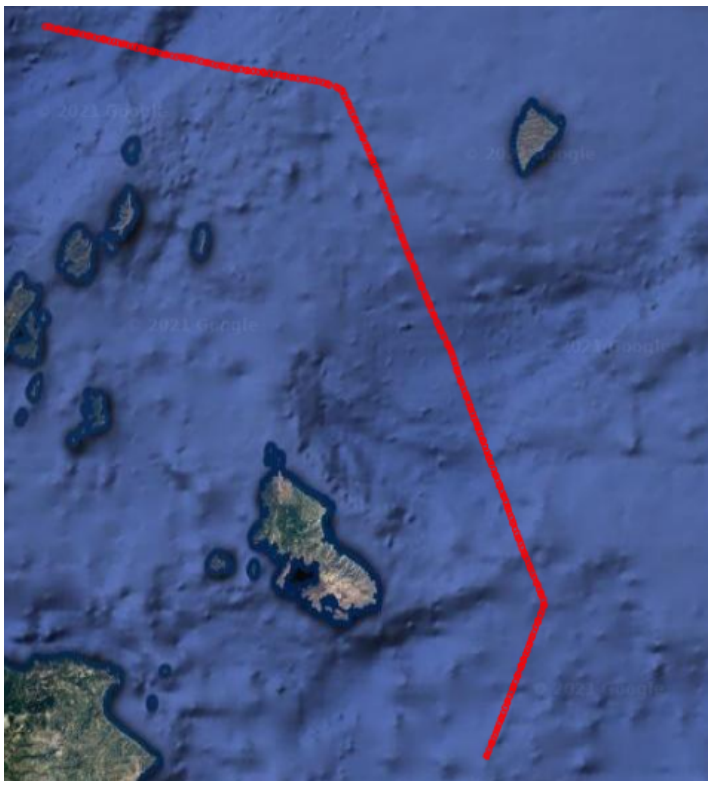

Figure 5.4. Vessel trajectory in GPS coordinates.

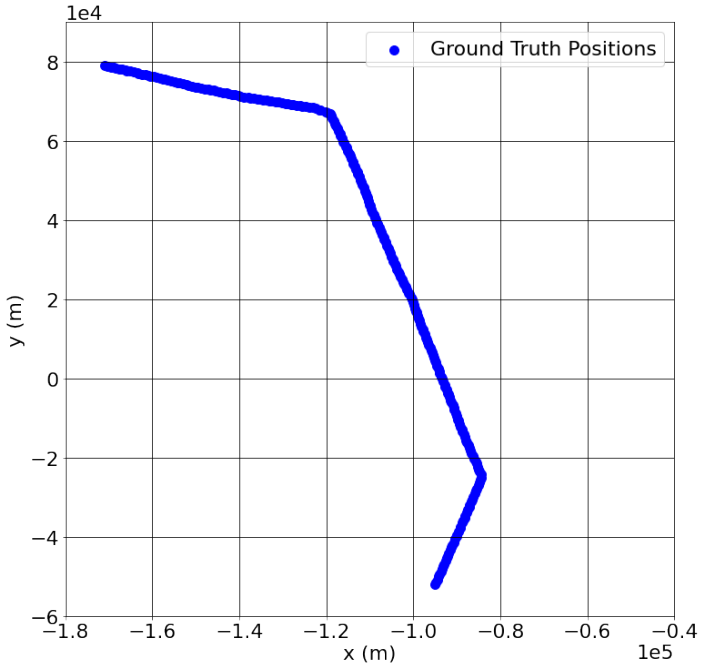

Figure 5.5. Vessel trajectory in Cartesian coordinates.

Following the computations of the vessel positions in Cartesian coordinates, bearings measurements were calculated. Two sensors were assumed to have been deployed on the ground to obtain bearing measurements from the vessels. Bearings are computed with

$$z_t = 90 - tan^{-1}\left(\frac{y_t - y_0}{x_t - x_0}\right), \tag{5.6}$$

where (x_0, y_0) is the sensor, (x_t, y_t) is the vessel coordinates in Cartesian plane at time t, and z_t is the bearing from that vessel at time t. Gaussian white noise, the covariance matrix of which is given by

$$R = \begin{bmatrix} 25 & 0 \\ 0 & 25 \end{bmatrix}, \tag{5.7}$$

was added to the measurements to adapt them to real world scenarios. An example of the resulting noisy bearings and ground truth bearings are given in Figure 5.6. We generated 25 different measurement sets for each trajectory (for 55 different trajectories in total) while we were preparing our training set for DL-based solutions.

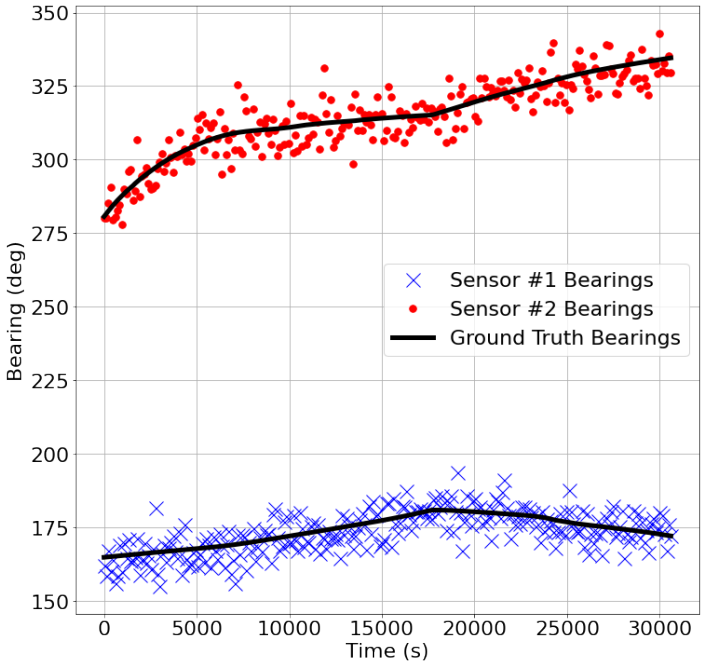

Figure 5.6. Examples of received bearings with two sensors and ground truth bearings.

5.2. Simulations

The experimented denoising methods have been divided into two groups, namely filtering and smoothing approaches. In Section 5.2.2, bearing-only tracking simulations with filtering approaches (in which only the bearing's history was used to reduce the effect of noise on the current bearing) will be given, whereas in Section 5.2.5, simulations with smoothing approaches (in which values from both bearings' history and future were used for denoising the current bearing) will be presented. These denoising approaches were applied as pre-denoising processes on bearings that were used as inputs to the tracking algorithms.

In the beginning of the tracking, the initialization vectors for each target, in Cartesian coordinates, were obtained by noise addition to the ground truth positions of targets. While tracking, a single bearing measurement from one sensor was used for a state update at each time step. The measurements of the sensors were used sequentially.

The estimated positions' MPE and RMSE values of Monte Carlo simulations were calculated over 34 different vessel trajectories. 40 different measurement sets for each trajectory were generated for tracking simulations.

5.2.1. The UKF

In this study, the UKF was selected from among the tracking algorithms were introduced in Chapter 3, for bearing-only tracking. The reasons for this selection are its mathematical robustness and easy computations, as well as its performance considering its RMSE and MPE values in test simulations against other competitors.

The estimated trajectory by the UKF from noisy bearings, are given in Figure 5.7, whereas the MPE values of simulations by the UKF can be seen in Figure 5.8. We can conclude that estimated positions are consistent with the ground truth positions, except for the case that when target maneuvers. There are biases in the estimations, but they can be accepted considering the noise effect. It is also can be inspected from the MPE values that, the UKF performed noise reduction in significant amounts in positions.

The UKF determines the contributions of measurements to estimations, according to their variances. If the measurement variance is high, the UKF results are dominated by the prediction part of the filter, and the UKF uses small contribution from measurements. Thus, if the variance of measurements are reduced by denoising before giving measurements as inputs to the filter, the UKF results can be improved. In order to analyze performance changes based upon that idea, we implemented denoising methods, that were introduced in Chapter 4, as pre-processing steps on bearing measurements in bearing-only tracking.

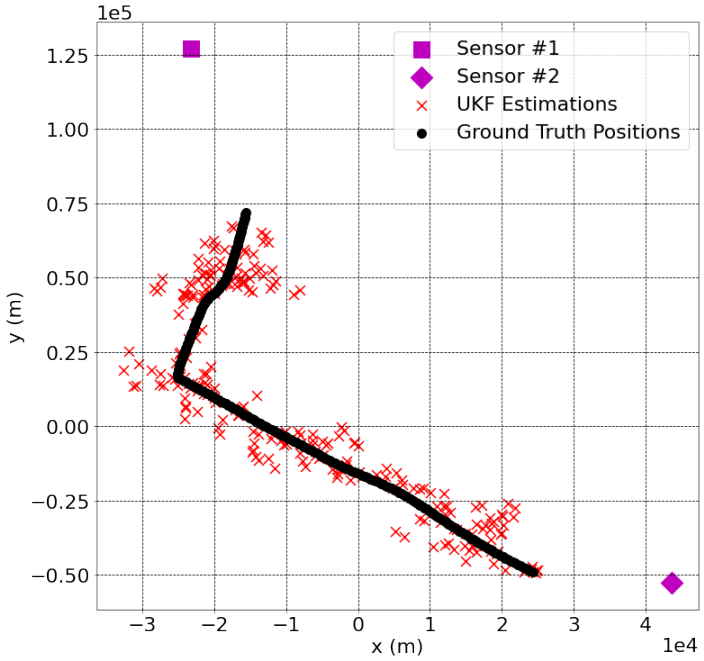

Figure 5.7. Estimated trajectory by UKF.

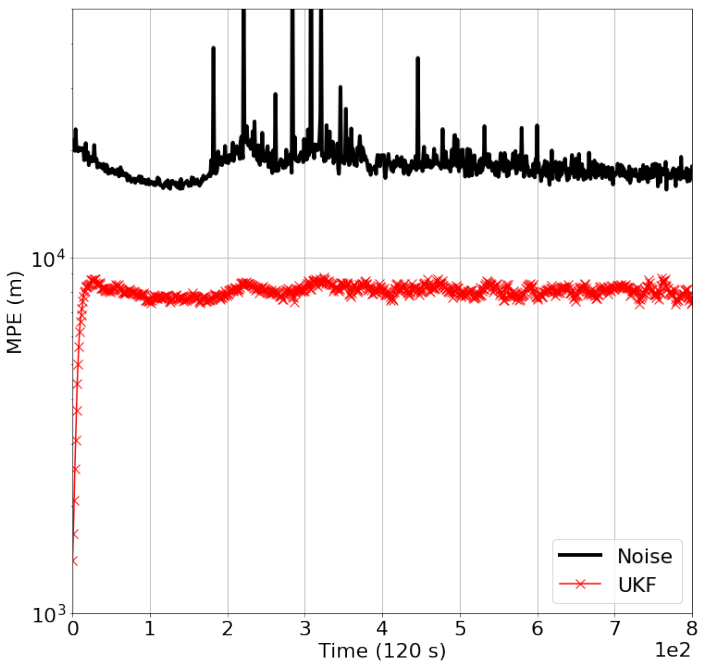

Figure 5.8. MPE values of estimations by UKF.

5.2.2. Simulations with Filtered Bearings

In this section, we will give details about our implementations of filtering approaches, which were introduced in Chapter 4, with the UKF.

5.2.2.1. The UKF with Linear Regression. In this tracking approach, noisy bearings were used in batches to compute linear regression parameters at each estimation time step. Then, these computed parameters were used to estimate denoised bearing measurements, which were used as inputs for the UKF. The batch size was set as 15 in our simulations. In order to estimate targets' states before obtaining the 15th measurements, only the current measurements belonging to the estimation time steps were used in the same way as the UKF.

5.2.2.2. The UKF with Moving Average Filtering. In this tracking approach, the averages, that were computed using the latest and the previous values of the measurements, were used to estimate denoised bearing measurements at each estimation time step. These measurement estimations were as inputs to the UKF. In average computations, 15 bearing values were used from the previous measurements in addition to the latest measurements. In order to estimate targets' states before obtaining the 15th measurements, only the current measurements belonging to the estimation time steps were used in the same way as the UKF.

5.2.2.3. The UKF with Exponential Smoothing. In this tracking approach, in order to estimate denoised bearing measurements to be used as inputs to the UKF at state estimation time steps, the latest measurement value and only one value from the previous time steps, which was the last smoothing result were used. This approach has the advantage of being applied easily in comparison with the other methods we have presented so far. In addition, there was no need to collect more than two measurements. The bearing denoising process was able to start from the second estimation time step.

In order to estimate targets' states at the first estimation time steps, only the current measurements were used in the same way as the UKF.

5.2.2.4. The UKF with DL-Based Filtering. In this tracking approach, a DL model with LSTM architecture, that was introduced in Section 4.1.4, was trained to estimate denoised bearing measurements. Then these measurements were used as inputs to the UKF at state estimation time steps. A single-layer LSTM with 16 hidden units were trained with the Adam optimization algorithm [70] and RMSE loss function. Keras Application Programming Interface (API) [71] was used to implement TensorFlow DL library [72] in our study. 15 bearings were used as inputs to the network in order to estimate the noise-free bearing at each time step. In order to estimate targets' states before obtaining the 15th measurements, only the current measurements belonging to the estimation time steps were used in the same way as the UKF.

5.2.3. Results of Tracking Simulations with Filtered Bearings

5.2.3.1. The UKF with Linear Regression. An example of estimated trajectory by the UKF with Linear Regression, is given in Figure 5.9. In comparison with the results of the UKF, we can see that there are fewer jumps in estimates, but fluctuations exist throughout the trajectory. Local improvements on the estimated trajectory are visible. From Figure 5.10, it is also clear that linear regression denoising on bearings leaded to decrease of the MPE while estimating different trajectories.

5.2.3.2. The UKF with Moving Average Filtering. An example of estimated trajectory by the UKF with Moving Average Filtering is given in Figure 5.11. There are improvements on the estimated trajectory. The estimated positions are consistent and close to each other, which are common cases in real world scenarios. Moreover, there are fewer fluctuations in estimations than the ones in Figure 5.9. From Figure 5.12, it can be concluded that moving average filtering reduced the MPE values of the UKF.

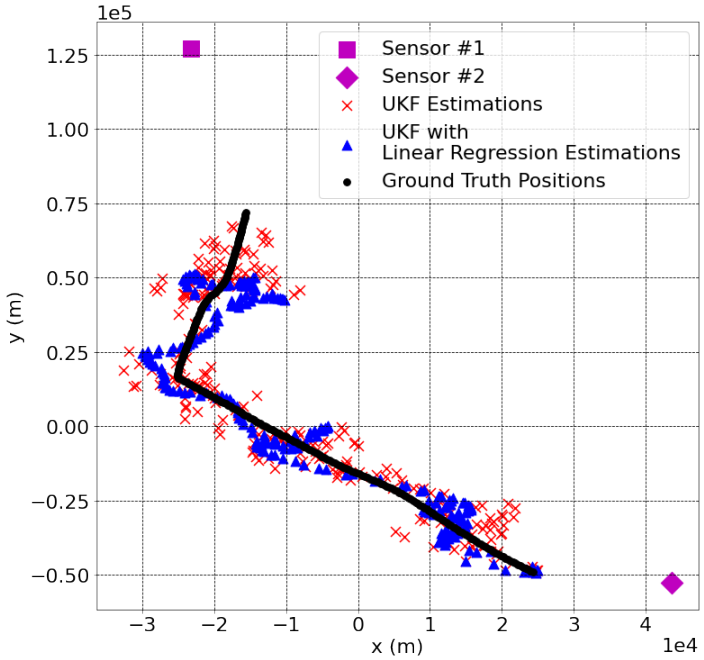

Figure 5.9. Estimated trajectory by UKF with Linear Regression.

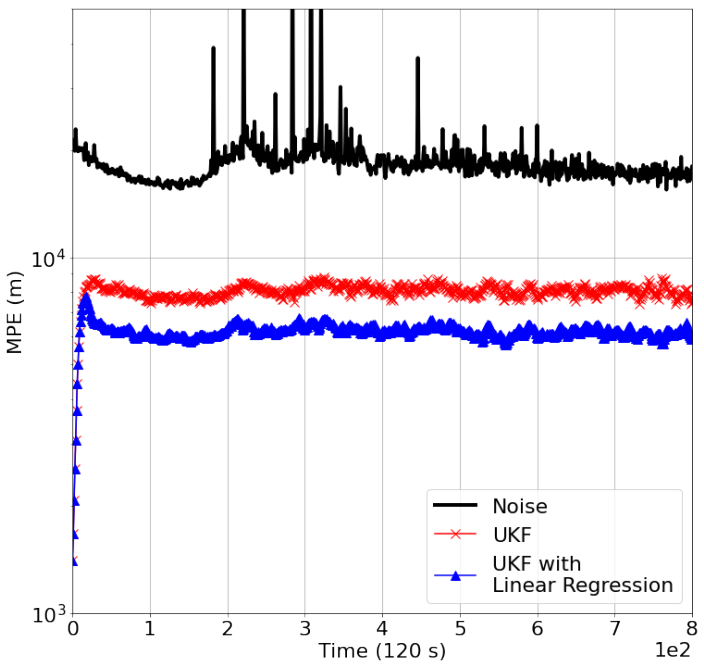

Figure 5.10. MPE values of estimations by UKF with Linear Regression.

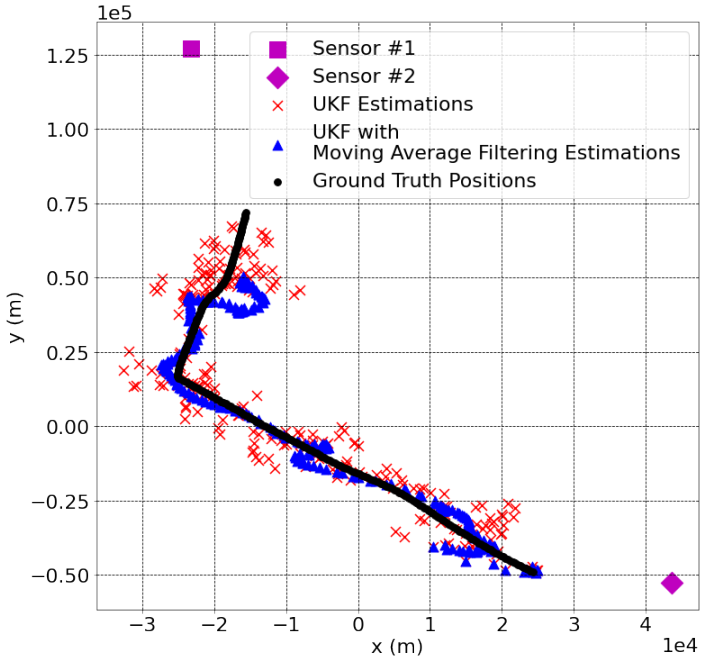

Figure 5.11. Estimated trajectory by UKF with Moving Average Filtering.

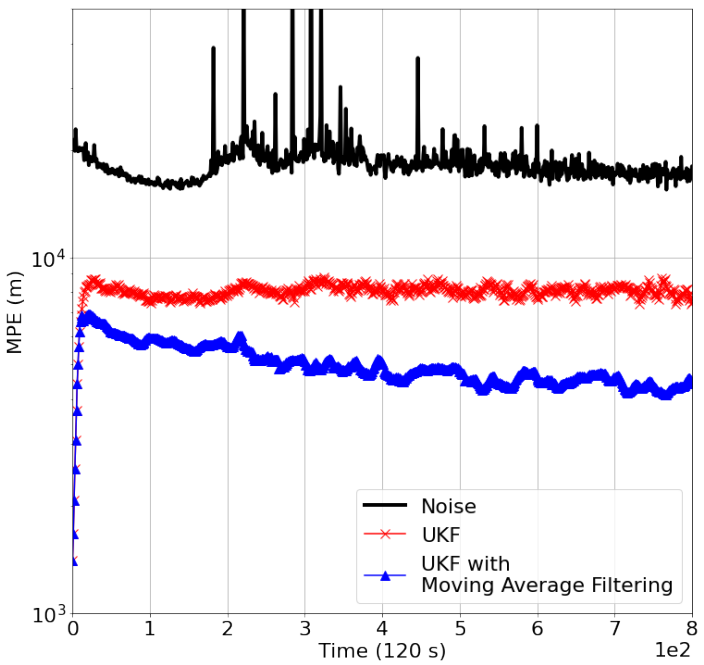

Figure 5.12. MPE values of estimations by UKF with Moving Average Filtering.

5.2.3.3. The UKF with Exponential Smoothing. An example of estimated trajectory by the UKF with Exponential Smoothing is given in Figure 5.13. We can see improvements in predicted positions in comparison with the UKF. Also, these improvements are visible at MPE results given in Figure 5.14. It can be concluded that estimated positions and the MPE results are similar to ones belonging to the UKF with Moving Average Filtering in Figure 5.11 and Figure 5.12. But, since this filtering method can start denoising process from the second time step, the MPE values are lower in the beginning of the estimations in comparison with the other methods have been discussed so far.

5.2.3.4. The UKF with DL-Based Filtering. An example of estimated trajectory by the UKF with DL-based filtering given in Figure 5.15. Here, we can observe that estimations are quite close to the ground truth positions. There are no sudden changes in estimated positions, which happened very often in the UKF. Also, the MPE values of estimating different trajectories, which are given in Figure 5.16, were reduced in simulations.

The overall tracking MPE values' comparisons with experimented filtering methods so far are given in Figure 5.17. It is clear that all the approaches have succeeded to improve estimation results. It can be stated that, ordinary changes in bearings due to maneuvers of the vessels could not be captured easily with linear regression, since linear regression always tries to fit a line for modelling bearings. We think that this was the main reason of linear regression weak performance. The other three methods have showed similar performance boosting in tracking. Even the simple averaging method showed recognizable differences in the simulation MPE values.

The RMSE values belonging to our simulations with filtering approaches are presented in Table 5.1. It can be concluded that the UKF with Exponential Smoothing and DL-Based Filtering performed similarly well in tracking.

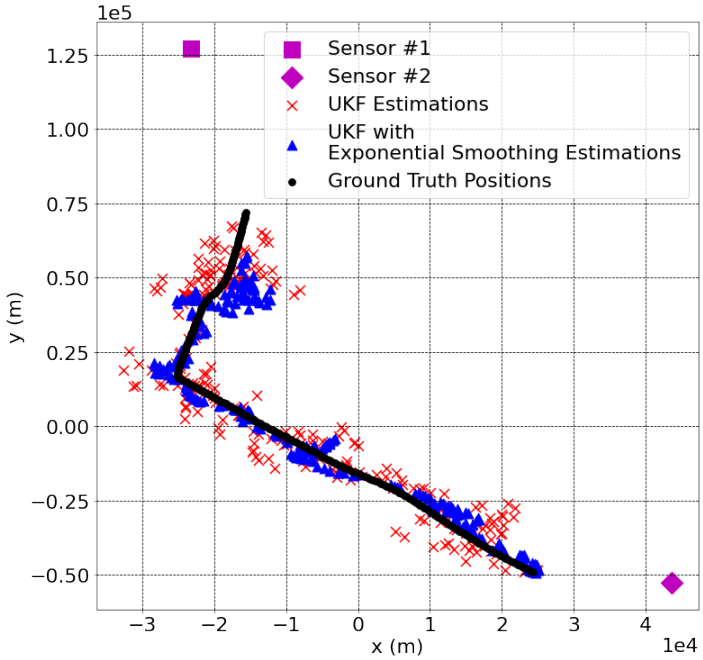

Figure 5.13. Estimated trajectory by UKF with Exponential Smoothing.

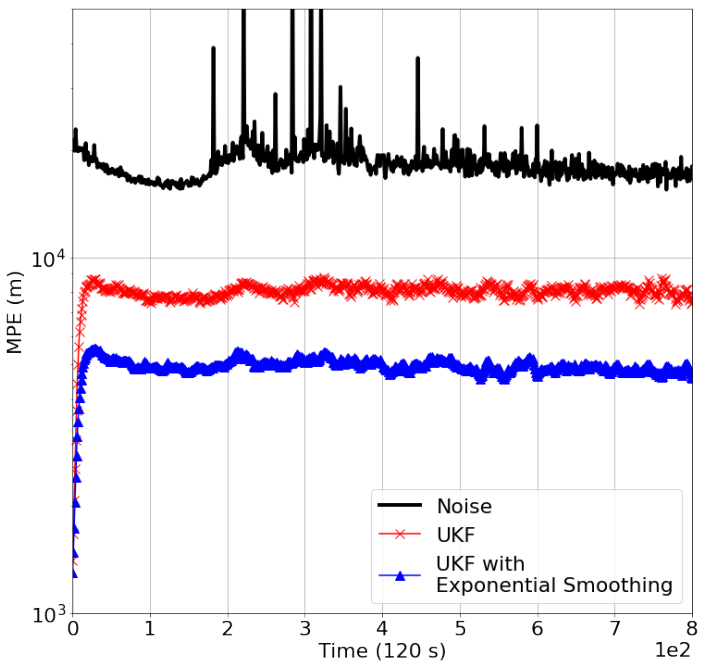

Figure 5.14. MPE values of estimations by UKF with Exponential Smoothing.

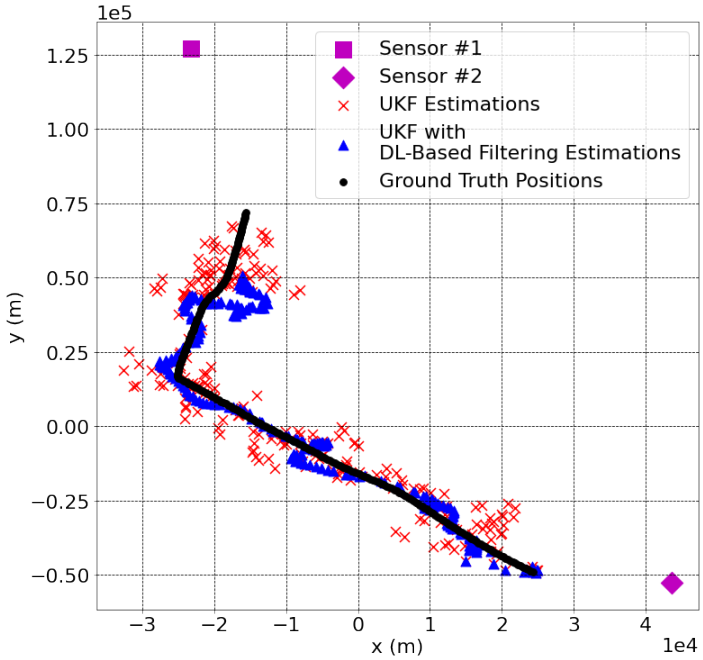

Figure 5.15. Estimated trajectory by UKF with DL-based filtering.

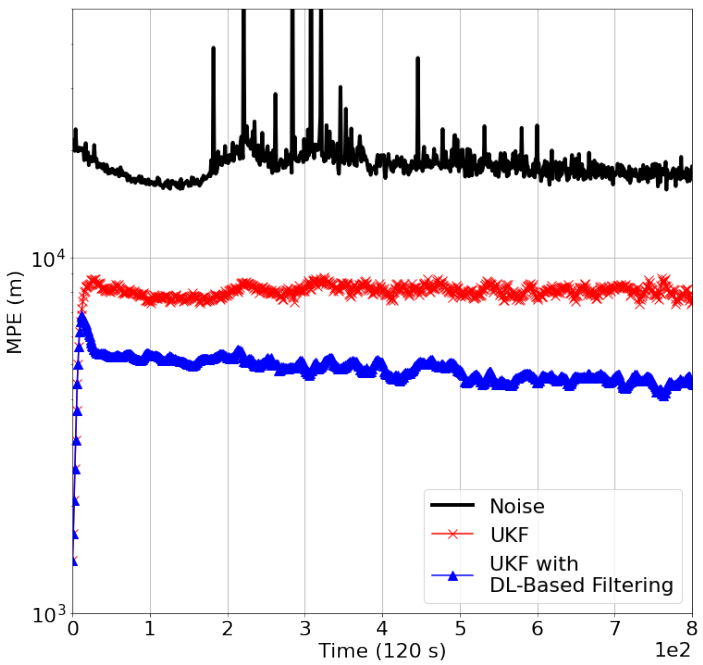

Figure 5.16. MPE values of estimations by UKF with DL-based filtering.

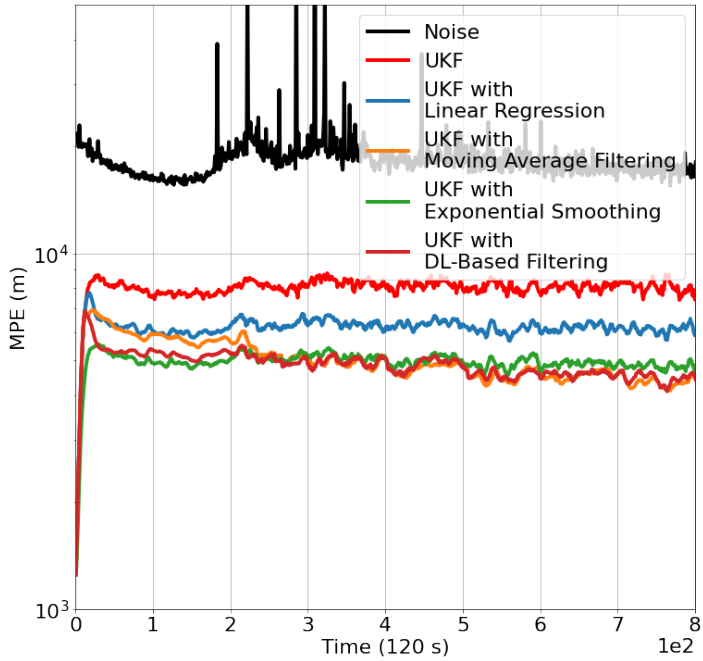

Figure 5.17. MPE values' comparison of UKF with filtered bearings.

However, exponential smoothing may be the best choice for a real world tracking application, since it does not have any training requirements as in DL-based filtering, and it does not have to collect many measurements before denoising as in moving average filtering or linear regression. Moreover, exponential smoothing has the lowest RMSE value over 1360 simulations.

Table 5.1. RMSE values of UKF with filtered bearings and noise.

	RMSE
UKF	6459.5670
UKF with Linear Regression	5084.1322
UKF with Moving Average Filtering	4634.1170
UKF with Exponential Smoothing	4118.3033
UKF with DL-Based Filtering	4258.4119
Noise	26598.3480

5.2.4. The Unscented Kalman Smoother (UKS)

In some applications, we need to estimate the position of a vessel belonging to the previous time step. It means that we will have not only the past measurements but also measurements from the future for prediction. So, it is wise to use all available information to make more accurate predictions. For example, we may need to investigate a ship collision and may not have an exact position information belonging to the collision time or any important time which has meaning for investigation. GPS information may not be available at these times, or existing information may not always be clean enough for evaluating important events. In order to use tracking methods in these applications, two tracking filters that operate in forward and backward in time are used at time same time. But using tracking approaches backward in time does not give satisfying results. The Rauch Tung smoothing (RTS) [73] based the unscented Kalman smoother (UKS) approach was proposed in [53]. The additional step, which is not included in the UKF, in the UKS algorithm is given by

$$D_{t+1} = \sum_{i=0}^{2N} W_i^c (\chi_t^i - x_{t|t})(\dot{\chi}_{t+1}^i - x_{t+1|t})^T \tag{5.8}$$

$$G_t = D_{t+1}[P_{t+1|t}]^{-1} \tag{5.9}$$

$$x_{t|t}^s = x_{t|t} + G_t(x_{t+1|t+1}^s - x_{t+1|t}) \tag{5.10}$$

$$P_{t|t}^s = P_{t|t} + G_t(P_{t+1|t+1}^s - P_{t+1|t})G_t^T, \tag{5.11}$$

where $x_{t|t}^s$ and $P_{t|t}^s$ are smoothed state vector and smoothed covariance matrix, respectively at time t. Here, G_t is used as a smoothing gain, and it is computed with the estimated covariance matrix $P_{t+1|t}$ and the covariance matrix D_{t+1} that is calculated using the estimated state vector $x_{t+1|t}$ and the updated state vector $x_{t|t}$ and their corresponding sigma points $\dot{\chi}_{t+1}$ and χ_t^i, respectively.

The estimated trajectory by the UKS from noisy bearings, is given in Figure 5.18, whereas the MPE values of simulations by the UKS can be seen in Figure 5.19. It is clear that estimated positions are consistent with each other.

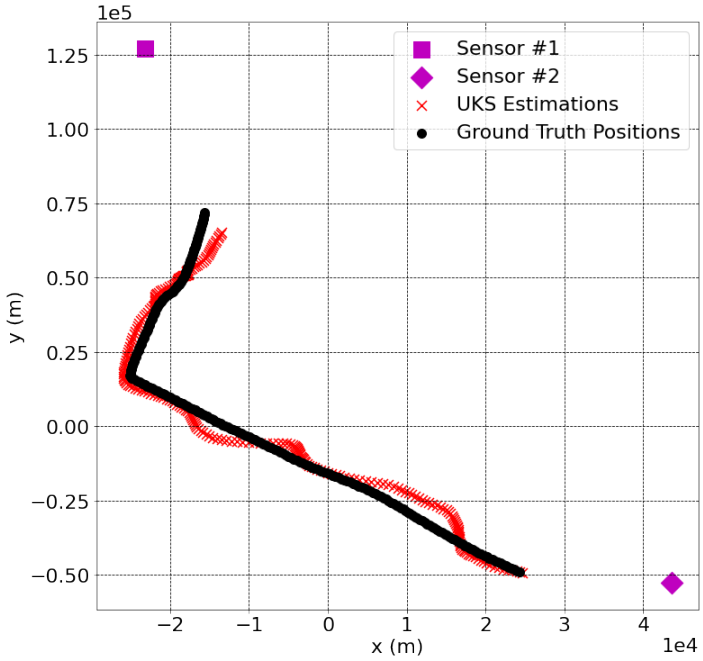

Figure 5.18. Estimated trajectory by UKS.

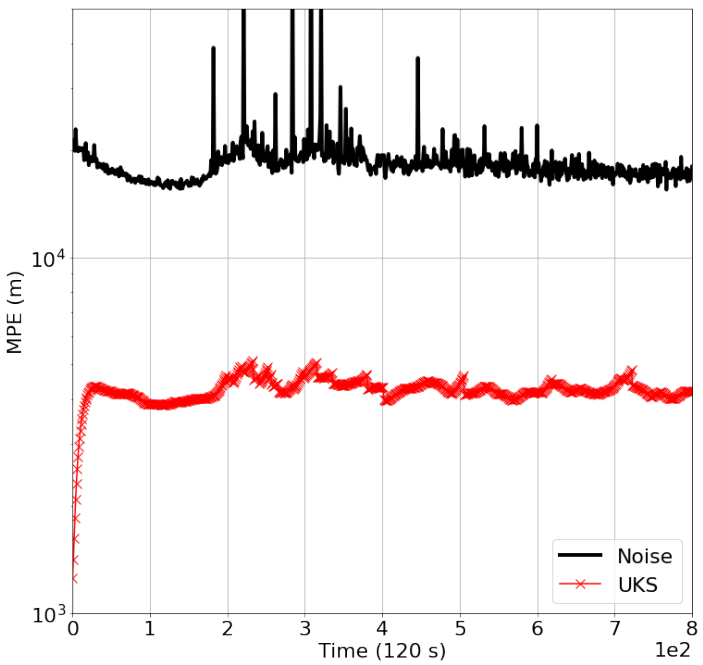

Figure 5.19. MPE values of estimations by UKS.

Here, the MPE values are lower than the ones belonging to the filtering methods have been presented so far.

5.2.5. Simulations with Smoothed Bearings

In this section, we will give details about our implementations of the smoothing approaches that were introduced in Chapter 4, as pre-processing steps on bearing measurements in bearing-only tracking with the UKS.

5.2.5.1. The UKS with Moving Average Smoothing. In this tracking approach, the averages, that were computed using the future and the previous values of the measurements, were used to estimate denoised bearing measurements at each estimation time step. These measurement estimations were used as inputs to the UKS. In average computations, 15 bearing values in total were used. In order to estimate targets' states before obtaining the 7th measurements (this the number of bearings were used from the previous bearing values), only the current measurements belonging to the estimation time steps were used in the same way as the UKS.

5.2.5.2. The UKS with Gaussian Smoothing. In this tracking approach, moving average smoothing, that were discussed in section 5.2.5.1, was implemented with an additional step that assigns weight coefficients from a Gaussian distribution, to bearings in average computation.

5.2.5.3. The UKS with Wavelet Smoothing. In this tracking approach, the WT-based smoothing method was used to denoise bearings. The resulting noise-free bearings were used as inputs to the UKS. In our experiments, Coiflet 4 wavelet, which is illustrated in Figure 5.20, was used in the WT. The decomposition level was selected as 5. In this smoothing method, 150 bearings were used for denoising at a time. These bearings were selected using a sliding window, when there were more than 150 bearings for a trajectory estimation.

It should be taken into account that, in this method, the denoising results are highly dependent on mother wavelet and level size selections. Several combinations should to be tested for data with different characteristics. We used scikit-image Python library [74] in our implementation.

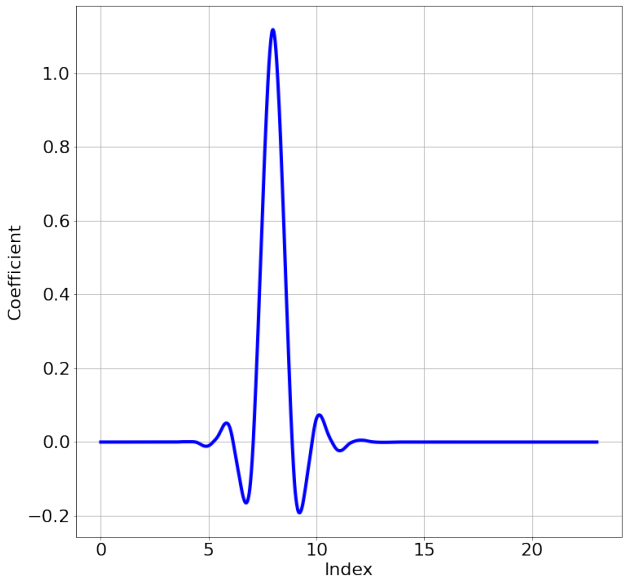

Figure 5.20. Coiflet 4 wavelet.

5.2.5.4. The UKS with DL-Based Smoothing. In this tracking approach, a DL model with BLSTM architecture, that was introduced in Section 4.2.4, was trained in order to estimate denoised bearing measurements to be used as an inputs to the UKS at state estimation time steps. A single-layer BLSTM with 32 hidden units was trained in the same way as expressed in Section 5.2.2.4. But in this smoothing method, 150 bearings were used as inputs to the network in order to estimate noise-free bearings. These bearings were selected using a sliding window, when there were more than 150 bearings for a trajectory estimation.

5.2.6. Results of Tracking Simulations with Smoothed Bearings

5.2.6.1. The UKS with Moving Average Smoothing. An example of estimated trajectory by the UKS with Moving Average Smoothing is given in Figure 5.21. We can observe from this figure that, the estimated positions are quite close to the ground truth positions throughout the trajectory, even when the target maneuvers. It is clear that using of the future values has been useful to denoise bearings if we compare the filtering results with the ones from Section 5.2.3. Moreover, the superior performance of this denoising method is also visible in the MPE values that are plotted in 5.22. These values are lower than all the values belonging to the denoising methods that we have discussed so far.

5.2.6.2. The UKS with Gaussian Smoothing. An example of estimated trajectory by the UKS with Gaussian Smoothing is given in Figure 5.23. It is clear that overall estimation performance is improved in comparison with the UKS. There are no significant divergences in estimated positions from the ground truth positions throughout the trajectory, but estimations in the beginning. It can be concluded that this estimated trajectory is not smooth as the one belonging to moving average smoothing in Figure 5.21; however, it is still closer to ground truth trajectory than results of the filtering methods that we have presented in Section 5.2.3. Furthermore, the MPE values given in Figure 5.24 are not low as the ones in Figure 5.22, but they are more acceptable than the ones belonging to the UKS in Figure 5.19.

5.2.6.3. The UKS with Wavelet Smoothing. An example of estimated trajectory by the UKS with Wavelet Smoothing is given in Figure 5.25. It is clear that the estimated trajectory is compatible with the ground truth trajectory. There are no sudden changes in estimations due to usage of so many measurements in smoothing at the same time, which ensures that the patterns between measurements are extracted easily. The satisfying performance of this method is also observed in the MPE values of simulations given in Figure 5.26.

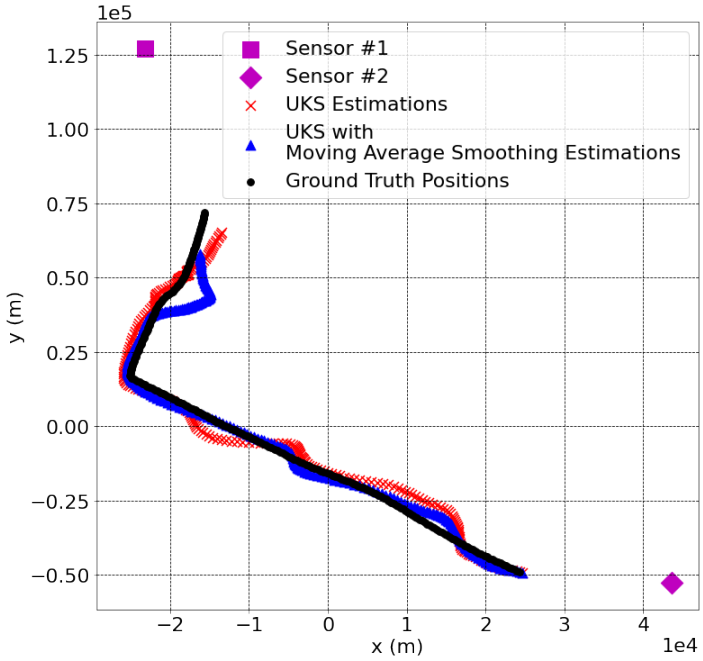

Figure 5.21. Estimated trajectory by UKS with Moving Average Smoothing.

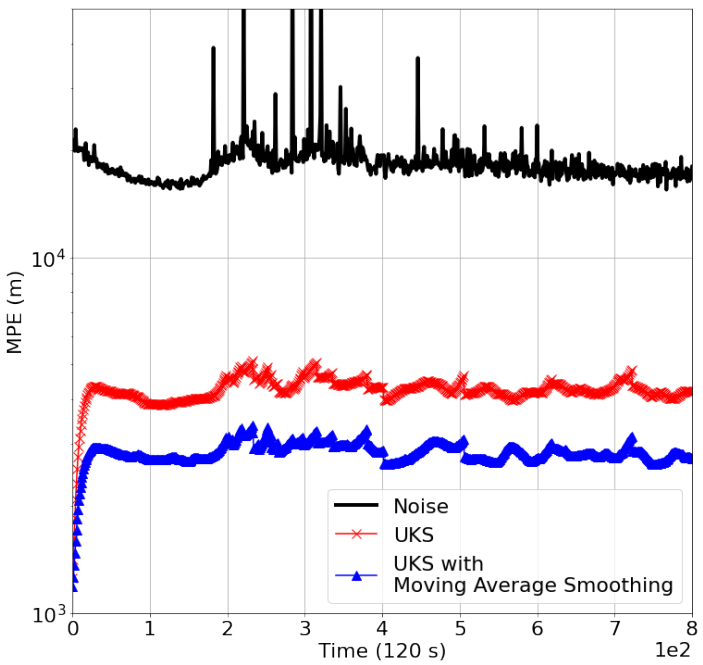

Figure 5.22. MPE values of estimations by UKS with Moving Average Smoothing.

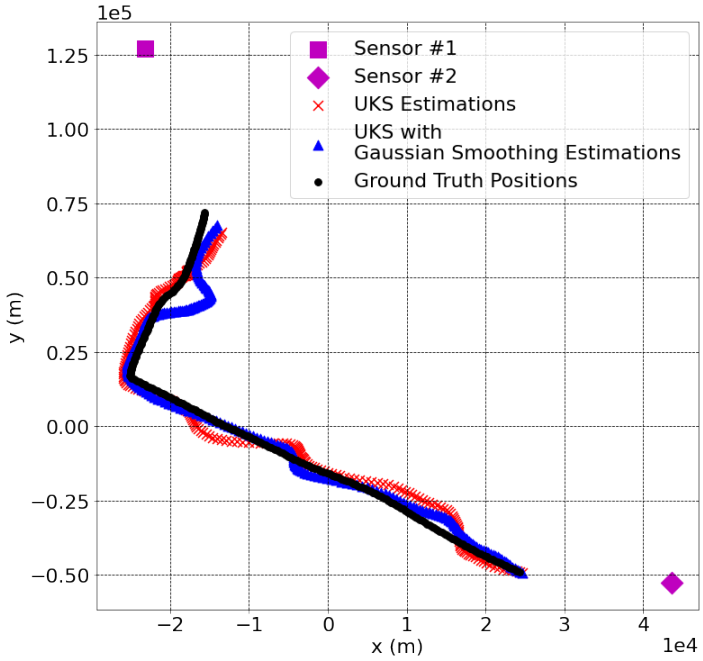

Figure 5.23. Estimated trajectory by UKS with Gaussian Smoothing.

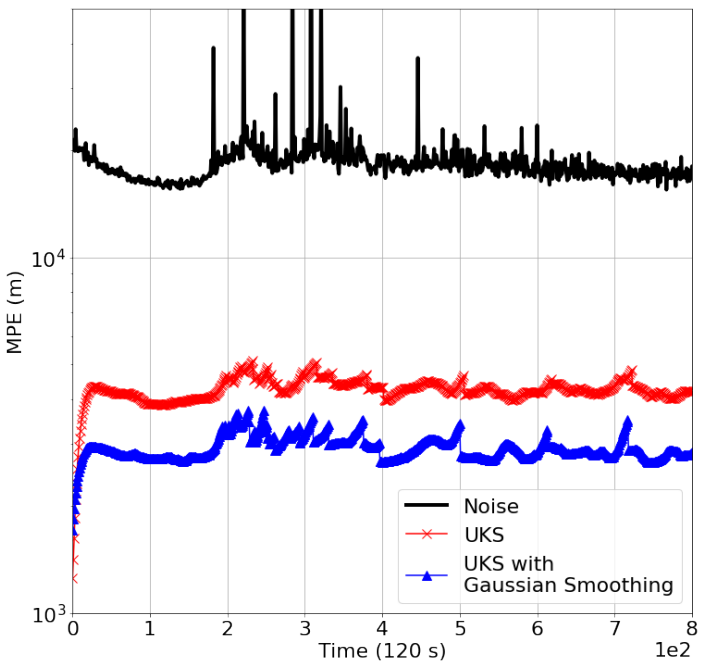

Figure 5.24. MPE values of estimations by UKS with Gaussian Smoothing.

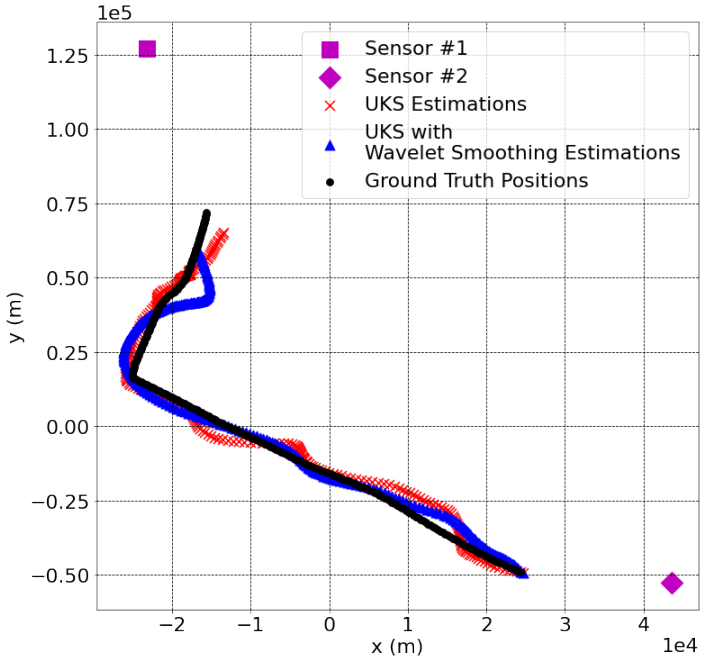

Figure 5.25. Estimated trajectory by UKS with Wavelet Smoothing.

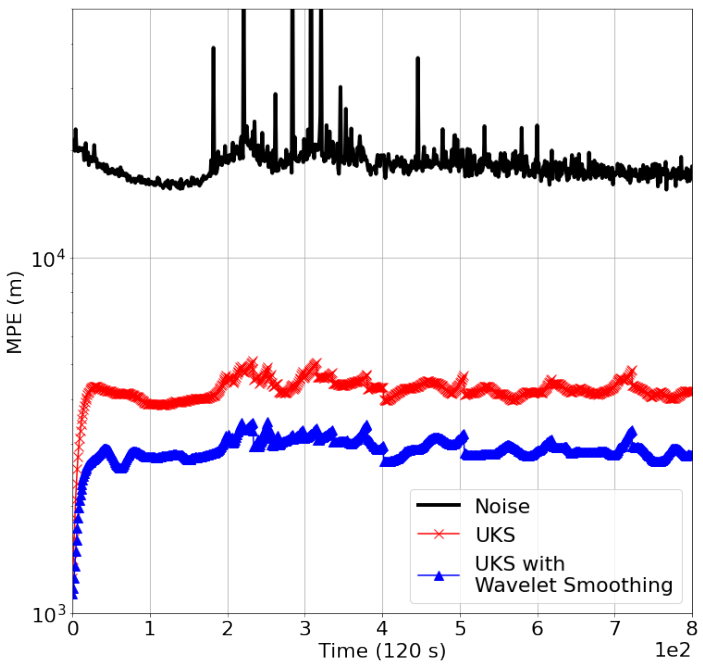

Figure 5.26. MPE values of estimations by UKS with Wavelet Smoothing.

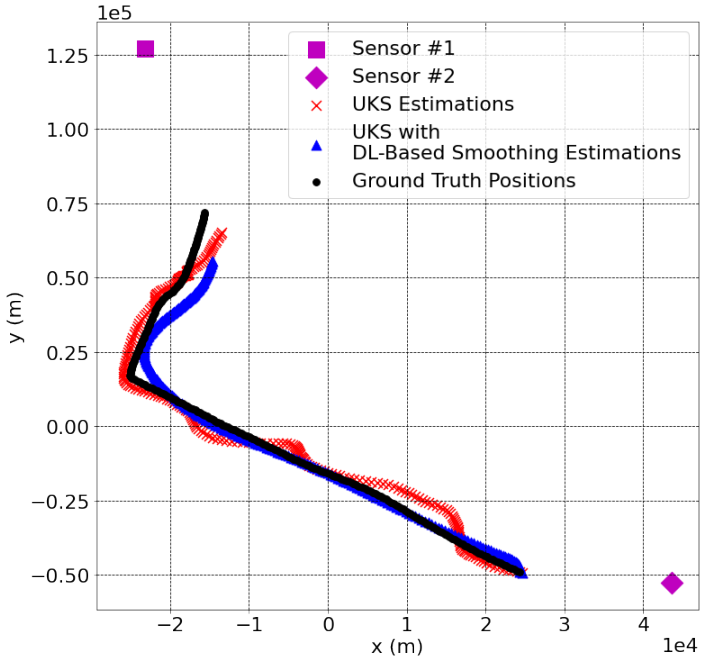

Figure 5.27. Estimated trajectory by UKS with DL-Based Smoothing.

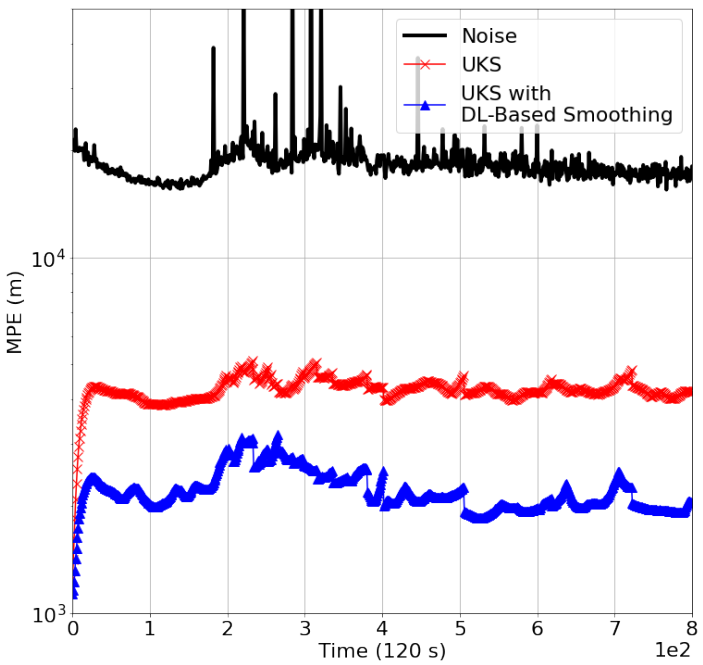

Figure 5.28. MPE values of estimations by UKS with DL-Based Smoothing.

5.2.6.4. The UKS with DL-Based Smoothing. An example of estimated trajectory by the UKS with DL-Based Smoothing is given in Figure 5.27. It is obvious that estimated positions are closer to ground truth positions than the estimated positions belonging to other methods implemented in this study. Even when the target maneuvers, estimations are consistent with these maneuvers. The MPE values, which are by far the lowest ones in our experiments, of the different trajectory estimations is given in Figure 5.28. From this figure, it can be concluded that this method's excellent performance is not only seen in the trajectory given in Figure 5.27, but also in all the other ones extracted from real data.

The overall tracking MPE values' comparisons with the implemented smoothing methods are given in Figure 5.29. The performance of the UKS with Gaussian Smoothing is the lowest one, as seen here. This result may be caused from the fact that the usage of a Gaussian weighting scheme increases the effect of a single data point which is at the center of the group of the data points, while suppressing the effect of the other ones. If the data point at the center is not close to the ground truth value, the performance of the Gaussian smoothing decreases. The performance of the UKS with Moving Average Smoothing is slightly better in comparison with the UKS with Wavelet Smoothing. But it should be noted that the wavelet smoothing's performance is subject to selection of the mother wavelets and decomposition levels. For other applications, these selections may perform better than moving average smoothing. The UKS with DL-Based Smoothing method has the best result in our simulations. It should be noted that the performances of the DL models are highly dependent on the training data set. It means that the performance of the UKS with DL-Based Smoothing may increase by using richer training data.

The RMSE values belonging to our simulations with smoothing approaches are presented in Table 5.2. We can see that these values are consistent with the MPE values in Figure 5.29. The UKS with DL-Based Smoothing has the lowest value over Monte Carlo simulations by a relatively large margin.

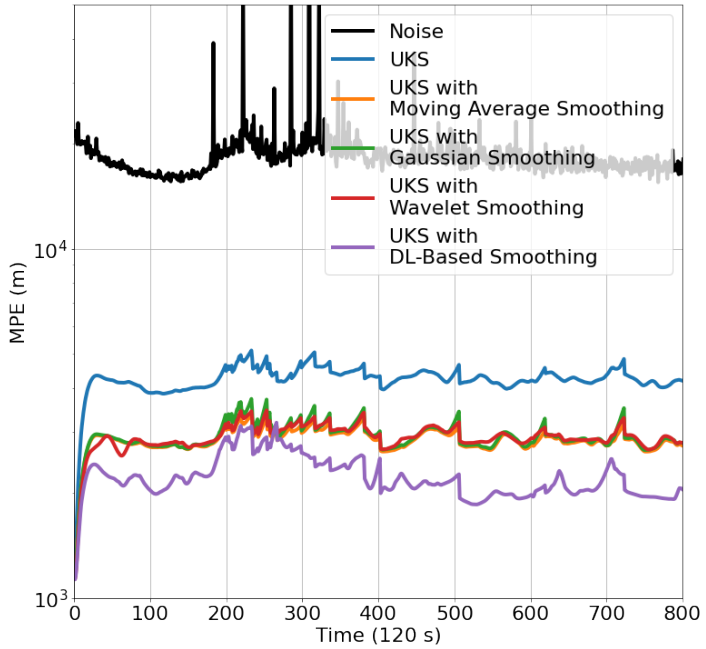

Figure 5.29. MPE values' comparisons of UKS with smoothed bearings.

Table 5.2. RMSE values of UKS with smoothed bearings and noise.

	RMSE
UKS	3462.3333
UKS with Moving Average Smoothing	2341.9210
UKS with Gaussian Smoothing	2462.9628
UKS with Wavelet Smoothing	2364.7350
UKS with DL-Based Smoothing	1964.2874
Noise	26598.3480

5.3. Simulation Results

Based upon analysis of the results of our experiments, it can be concluded that implementing pre-processing step as denoising on bearings before implementing the tracking method reduced the estimation RMSE values.

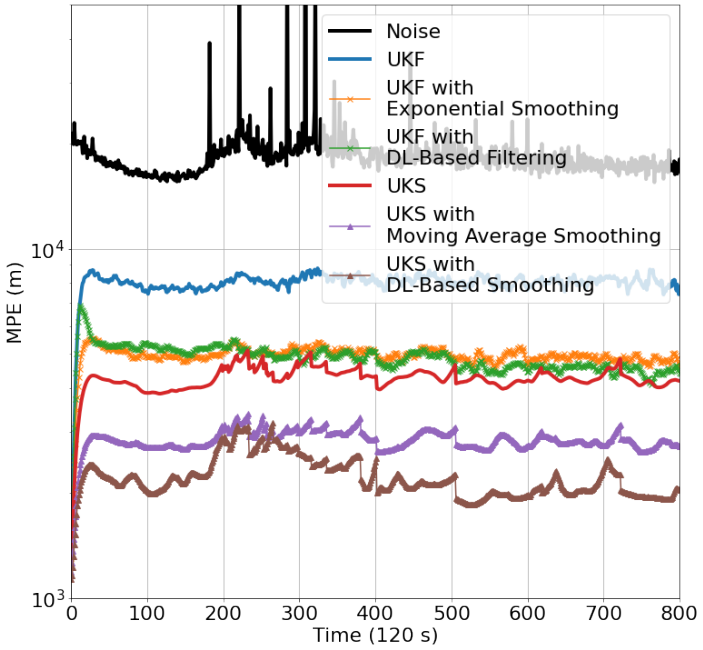

Figure 5.30. MPE values' comparisons of UKF and UKS with denoising methods.

The online denoising methods, which we named as filtering approaches in our study, could not catch up with the results of the offline denoising methods, which we named as smoothing approaches. The usage of online denoising methods are advised when the future information is not available, because they result in significant improvements in simulation results. As we have stated before, the exponential smoothing method may be the best choice here due to its ease of application and performance. On the other hand, for the cases in which the future information is available, the DL-based smoothing method is the best choice, since it outperformed the other competitors in our experiments. The only drawback of this method is the burden of collecting a comprehensive dataset that consists of all the possible cases in target tracking.

The MPE values' comparisons of the two best filtering and smoothing approaches are given in Figure 5.30.

6. CONCLUSION & FUTURE WORK

6.1. Conclusion

In this thesis, bearing-only tracking, one of the most studied target tracking methods, was studied.

Firstly, we presented the backgrounds of commonly used target tracking methods in the literature: the KF, the EKF, the UKF and the PF. Later, we tested these methods' tracking performances on an artificially generated tracking scenario. It was observed from our experiments that the UKF was one of the best solutions for the addressed problem of this study. Considering its performance and implementation advantages, we decided to use it as a tracking method throughout our study. Moreover, we conducted a brief experiment on the effect of the different measurement noise conditions on tracking performance, and we observed that when the noise effect on measurements is low, the tracking performance of the UKF becomes better. Based on this observation, we investigated noise reduction approaches on a time-series data that covers bearing measurements in our tracking problem, in order to boost the performance of the UKF on real case applications. We have grouped noise reduction approaches into two groups, regarding requirements of their applicability depending on times of available information:

- In the first group, the methods that do not require information from the future called "filtering methods" in this study, were discussed. The most important advantage of these filtering methods is that they can be used in real time tracking problems. It was verified via experimental results on real target trajectories obtained from the AIS data that denoising bearing measurements with these filtering methods, namely linear regression, moving average filtering, exponential smoothing, DL-based filtering methods, increased the tracking performance of the UKF.

Additionally, it was stated that the filtering with exponential smoothing would be a reasonable choice due to the fact that it does not require any training to use, and it does not need to have a certain amount of measurements to operate.

- In the second group, the methods that require information from the future called "smoothing methods" in this study, were discussed. These smoothing methods can be applied to real world situations in which the previous position of the target is to be discovered. The smoothing methods, namely moving average smoothing, the Gaussian smoothing, the wavelet smoothing and the DL-based smoothing, showed significant effect on performance increase of the UKS. The DL-based smoothing would be the best choice due to its flexibility in different applications using modifications on its training step, in addition to having the best performance increase for the UKS.

We can conclude that, applying any pre-denoising methods on sensor bearings helps to increase tracking performance of bearing-only tracking applications. Presented denoising methods can be selected depending on the application requirements of a real world tracking problem.

6.2. Future Work

We implemented several hybrid solutions for bearing-only tracking application in this study. But in real world implementation, using compact solutions are more preferable due to being less error-prone and being able to integrate different applications easily. We experienced that DL-based solutions are promising in tracking applications, as well as the impressive performances of the tracking algorithms were introduced years ago. We plan to build a single solution that exploits advantages of the both approaches, implicitly in our future studies.

REFERENCES

1. Zhao, Z., T. Rong Li and V. Jilkov, "Best Linear Unbiased Filtering With Nonlinear Measurements for Target Tracking", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 40, No. 4, pp. 1324–1336, 2004.

2. Li, X. R. and V. P. Jilkov, "Survey of Maneuvering Target Tracking: III. Measurement Models", O. E. Drummond (Editor), *Signal and Data Processing of Small Targets 2001*, Vol. 4473, pp. 423–446, International Society for Optics and Photonics, SPIE, 2001.

3. Mahler, R., "Multitarget Bayes Filtering via First-Order Multitarget Moments", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 39, No. 4, pp. 1152–1178, 2003.

4. Vo, B.-N. and W. Ma, "The Gaussian Mixture Probability Hypothesis Density Filter", *IEEE Transactions on Signal Processing*, Vol. 54, No. 11, pp. 4091–4104, 2006.

5. Vo, B., S. Singh and A. Doucet, "Sequential Monte Carlo Methods for Multitarget Filtering With Random Finite Sets", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 41, No. 4, pp. 1224–1245, 2005.

6. Blom, H. and Y. Bar-Shalom, "The Interacting Multiple Model Algorithm for Systems With Markovian Switching Coefficients", *IEEE Transactions on Automatic Control*, Vol. 33, No. 8, pp. 780–783, 1988.

7. Bar-Shalom, Y., K. Chang and H. Blom, "Tracking a Maneuvering Target Using Input Estimation Versus The Interacting Multiple Model Algorithm", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 25, No. 2, pp. 296–300, 1989.

8. Genovese, A. F., "The Interacting Multiple Model Algorithm for Accurate State Estimation of Maneuvering Targets", *Johns Hopkins APL technical digest*, Vol. 22, No. 4, pp. 614–623, 2001.

9. Arulampalam, M., B. Ristic, N. Gordon and M. T., "Bearings-Only Tracking of Manoeuvring Targets Using Particle Filters", *EURASIP Journal on Applied Signal Processing*, Vol. 15, p. 2351–2365, 2004.

10. Aidala, V. and S. Hammel, "Utilization of Modified Polar Coordinates for Bearings-Only Tracking", *IEEE Transactions on Automatic Control*, Vol. 28, No. 3, pp. 283–294, 1983.

11. Whitcombe, D. W., "Pseudo State Measurements Applied to Recursive Nonlinear Filtering", *Proc. 3rd Symp. Nonlinear Estimation Theory and Its Application*, 1972.

12. Jauffret, C. and D. Pillon, "Observability in Passive Target Motion Analysis", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 32, No. 4, pp. 1290–1300, 1996.

13. Gao, J., H. Sultan, J. Hu and W.-W. Tung, "Denoising Nonlinear Time Series by Adaptive Filtering and Wavelet Shrinkage: A Comparison", *IEEE Signal Processing Letters*, Vol. 17, No. 3, pp. 237–240, 2010.

14. Sayadi, O. and M. B. Shamsollahi, "ECG Denoising and Compression Using a Modified Extended Kalman Filter Structure", *IEEE Transactions on Biomedical Engineering*, Vol. 55, No. 9, pp. 2240–2248, 2008.

15. Sayadi, O. and M. Shamsollahi, "Multiadaptive Bionic Wavelet Transform: Application to ECG Denoising and Baseline Wandering Reduction", *EURASIP Journal on Advances in Signal Processing*, Vol. 2007, pp. 1–11, 2007.

16. He, T., G. Clifford and L. Tarassenko, "Application of Independent Component

Analysis in Removing Artefacts From the Electrocardiogram", *Neural Computing and Applications*, Vol. 20, No. 15, pp. 105–116, 2005.

17. Ari, S., M. K. Das and A. Chacko, "ECG Signal Enhancement Using S-Transform", *Computers in Biology and Medicine*, Vol. 43, No. 6, pp. 649–660, 2013.

18. Stockwell, R., L. Mansinha and R. Lowe, "Localization of the Complex Spectrum: The S Transform", *IEEE Transactions on Signal Processing*, Vol. 44, No. 4, pp. 998–1001, 1996.

19. Ercelebi, E., "Electrocardiogram Signals De-noising Using Lifting-Based Discrete Wavelet Transform", *Computers in Biology and Medicine*, Vol. 34, No. 6, pp. 479–493, 2004.

20. Poornachandra, S., "Wavelet-Based Denoising Using Subband Dependent Threshold for ECG Signals", *Digital Signal Processing*, Vol. 18, No. 1, pp. 49–55, 2008.

21. Wang, Z., F. Wan, C. M. Wong and L. Zhang, "Adaptive Fourier Decomposition Based ECG Denoising", *Computers in Biology and Medicine*, Vol. 77, pp. 195–205, 2016.

22. Chang, K. and S. Liu, "Gaussian Noise Filtering from ECG by Wiener Filter and Ensemble Empirical Mode Decomposition", Vol. 64, p. 249–264, 2010.

23. Ho, Y. and R. Lee, "A Bayesian Approach to Problems in Stochastic Estimation and Control", *IEEE Transactions on Automatic Control*, Vol. 9, No. 4, pp. 333–339, 1964.

24. Ristic, B., S. Arulampalam and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*, Vol. 19, Artech House Radar Library, 2004.

25. Kalman, R. E., "A New Approach to Linear Filtering and Prediction Problems", *Journal of Basic Engineering*, Vol. 82, pp. 35–45, 1960.

26. Jazwinski, A. H., *Stochastic Processes and Filtering Theory*, Academic Press, 1970.

27. Julier, S. J. and J. K. Uhlmann, "Unscented Filtering and Nonlinear Estimation", *Proceedings of the IEEE*, Vol. 92, pp. 401–422, 2004.

28. Gordon, N. J., D. J. Salmond and A. F. Smith, "Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation", *IEE Proceedings F-Radar and Signal Processing*, Vol. 140, No. 2, pp. 107–113, 1993.

29. Douc, R. and O. Cappe, "Comparison of Resampling Schemes for Particle Filtering", *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, pp. 64–69, 2005.

30. Li, T., M. Bolic and P. M. Djuric, "Resampling Methods for Particle Filtering: Classification, Implementation, and Strategies", *IEEE Signal Processing Magazine*, Vol. 32, No. 3, pp. 70–86, 2015.

31. Montgomery, D. C., E. A. Peck and G. G. Vining, *Introduction to Linear Regression Analysis*, John Wiley & Sons, 2012.

32. Savitzky, A. and M. J. Golay, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures", *Analytical Chemistry*, Vol. 36, No. 8, pp. 1627–1639, 1964.

33. Hansun, S. and M. B. Kristanda, "Performance Analysis of Conventional Moving Average Methods in Forex Forecasting", *International Conference on Smart Cities, Automation Intelligent Computing Systems*, pp. 11–17, 2017.

34. Shmueli, G., P. C. Bruce, I. Yahav, N. R. Patel and K. C. Lichtendahl Jr, *Data Mining for Business Analytics: Concepts, Techniques, and Applications in R*, John Wiley & Sons, 2017.

35. Holt, C. C., "Forecasting Seasonals and Trends by Exponentially Weighted Moving

Averages", *International Journal of Forecasting*, Vol. 20, No. 1, pp. 5–10, 2004.

36. Chan, K. Y., T. S. Dillon, J. Singh and E. Chang, "Neural-Network-Based Models for Short-Term Traffic Flow Forecasting Using a Hybrid Exponential Smoothing and Levenberg–Marquardt Algorithm", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 13, No. 2, pp. 644–654, 2012.

37. Tan, M.-C., S. C. Wong, J.-M. Xu, Z.-R. Guan and P. Zhang, "An Aggregation Approach to Short-Term Traffic Flow Prediction", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 10, No. 1, pp. 60–69, 2009.

38. Winters, P. R., "Forecasting Sales by Exponentially Weighted Moving Averages", *Management science*, Vol. 6, No. 3, pp. 324–342, 1960.

39. Bishop, C. M., *Pattern Recognition and Machine Learning*, Vol. 128, 2006.

40. Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning*, MIT press, 2016.

41. Rosenblatt, F., "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain", *Psychological Review*, Vol. 65, No. 6, p. 386, 1958.

42. Rumelhart, D. E., G. E. Hinton and R. J. Williams, "Learning Representations by Back-Propagating Errors", *nature*, Vol. 323, No. 6088, pp. 533–536, 1986.

43. Connor, J., R. Martin and L. Atlas, "Recurrent Neural Networks and Robust Time Series Prediction", *IEEE Transactions on Neural Networks*, Vol. 5, No. 2, pp. 240–254, 1994.

44. Garofolo, J. S., "TIMIT Acoustic-Phonetic Continuous Speech Corpus", *Linguistic Data Consortium*, 1993.

45. Graves, A., A.-r. Mohamed and G. Hinton, "Speech Recognition With Deep Recurrent Neural Networks", *IEEE International Conference on Acoustics, Speech*

and Signal Processing, pp. 6645–6649, 2013.

46. Graves, A., S. Fernández, F. Gomez and J. Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data With Recurrent Neural Networks", *Proceedings of the 23rd International Conference on Machine learning*, pp. 369–376, 2006.

47. Graves, A., "Sequence Transduction With Recurrent Neural Networks", *International Conference of Machine Learning Workshop on Representation Learning*, 2012.

48. Palangi, H., L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song and R. Ward, "Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 24, No. 4, pp. 694–707, 2016.

49. Järvelin, K. and J. Kekäläinen, "IR Evaluation Methods for Retrieving Highly Relevant Documents", *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, p. 41–48, 2000.

50. Amidi, A. and S. Amidi, *Recurrent Neural Networks Cheatsheet*, 2019, `https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks`, accessed in August 2021.

51. Hochreiter, S. and J. Schmidhuber, "Long Short-Term Memory", *Neural Computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

52. Olah, C., *Understanding LSTM Networks*, 2015, `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, accessed in August 2021.

53. Särkkä, S., "Unscented Rauch-Tung-Striebel Smoother", *IEEE Transactions on Automatic Control*, Vol. 53, No. 3, pp. 845–849, 2008.

54. Marr, D. and E. Hildreth, "Theory of Edge Detection", *Proceedings of the Royal Society of London. Series B. Biological Sciences*, Vol. 207, No. 1167, pp. 187–217, 1980.

55. Basu, M., "Gaussian-Based Edge-Detection Methods-A Survey", *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 32, No. 3, pp. 252–260, 2002.

56. Gençay, R., F. Selçuk and B. Whitcher, *Introduction to Wavelets and Other Filtering Methods in Finance and Economics*, Academic Press, 2002.

57. Allen, J., "Short Term Spectral Analysis, Synthesis, and Modification by Discrete Fourier Transform", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 25, No. 3, pp. 235–238, 1977.

58. Mallat, S., "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 7, pp. 674–693, 1989.

59. Daubechies, I., *Ten Lectures on Wavelets*, SIAM, 1992.

60. Donoho, D. L. and J. M. Johnstone, "Ideal Spatial Adaptation by Wavelet Shrinkage", *Biometrika*, Vol. 81, No. 3, pp. 425–455, 1994.

61. Wink, A. and J. Roerdink, "Denoising Functional MR Images: A Comparison of Wavelet Denoising and Gaussian Smoothing", *IEEE Transactions on Medical Imaging*, Vol. 23, No. 3, pp. 374–387, 2004.

62. Patil, S. S. and M. K. Pawar, "Quality Advancement of EEG by Wavelet Denoising for Biomedical Analysis", *International Conference on Communication, Information Computing Technology*, pp. 1–6, 2012.

63. Patil, R., "Noise Reduction Using Wavelet Transform and Singular Vector Decom-

position", *Procedia Computer Science*, Vol. 54, pp. 849–853, 2015.

64. Ning, J., J. Wang, W. Gao and C. Liu, "A Wavelet-Based Data Compression Technique for Smart Grid", *IEEE Transactions on Smart Grid*, Vol. 2, No. 1, pp. 212–218, 2011.

65. Schuster, M. and K. Paliwal, "Bidirectional Recurrent Neural Networks", *IEEE Transactions on Signal Processing*, Vol. 45, No. 11, pp. 2673–2681, 1997.

66. Graves, A. and J. Schmidhuber, "Framewise Phoneme Classification With Bidirectional LSTM Networks", *IEEE International Joint Conference on Neural Networks*, Vol. 4, pp. 2047–2052, 2005.

67. *International Maritime Organization*, 2019, `https://www.imo.org`, accessed in August 2021.

68. *MarineTraffic – Global Ship Tracking Intelligence*, 2007, `https://www.marinetraffic.com`, accessed in August 2021.

69. Weisstein, E. W., *Stereographic Projection*, 2004, `https://mathworld.wolfram.com/StereographicProjection.html`, accessed in August 2021.

70. Kingma, D. P. and J. Ba, "Adam: A Method for Stochastic Optimization", *International Conference on Learning Representations*, 2014.

71. Chollet, F. *et al.*, *Keras*, 2015, `https://github.com/fchollet/keras`, accessed in August 2021.

72. Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan,

F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu and X. Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015, `https://www.tensorflow.org/`, accessed in August 2021.

73. Rauch, H. E., F. Tung and C. T. Striebel, "Maximum Likelihood Estimates of Linear Dynamic Systems", *AIAA journal*, Vol. 3, No. 8, pp. 1445–1450, 1965.

74. Van der Walt, S., J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart and T. Yu, "scikit-image: Image Processing in Python", *PeerJ*, Vol. 2, p. e453, 2014.