Multiple Model Bayesian Estimation for BLE-based Localization and RL-based Decision Support of Autonomous Agents

Parvin Malekzadeh

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

August 2020

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Parvin Malekzadeh**

Entitled: **Multiple Model Bayesian Estimation for BLE-based Localization and RL-based Decision Support of Autonomous Agents**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Omair Ahmed

_____ External Examiner
Dr. Abdessamad Ben Hamza

_____ Examiner
Dr. Wei-Ping Zhu

_____ Supervisor
Dr. Arash Mohammadi

Approved by _____
Dr. Youssef Shayan, Chair
Department of Electrical and Computer Engineering

_____ 2020

_____
Dr. Mourad Debbabi, Dean
Faculty of Gina Cody School of Engineering and Computer Science

Abstract

Multiple Model Bayesian Estimation for BLE-based Localization and RL-based Decision Support of Autonomous Agents

Parvin Malekzadeh

With the rapid emergence of Internet of Things (IoT), we are more and more surrounded by smart connected devices (agents) with integrated sensing, processing, and communication capabilities. In particular, IoT-based positioning has become of primary importance for providing advanced Location-based Services (LBSs) in indoor environments. Several LBSs have been developed recently such as navigation assistance in hospitals, localization/tracking in smart buildings, and providing assistive services via autonomous agents collectively act as an Internet of Robotic Things (IoRT). The focus of the thesis is on the following two research topics when it comes to autonomous agents providing LBSs in indoor environments: (i) *Self-Localization*, which is the autonomous agent's ability to obtain knowledge of its own location, and; (ii) *Localized Decision Support System*, which refers to an autonomous agent's ability to perform optimal actions towards achieving pre-defined objectives. With regards to Item (i), the thesis develops innovative localization solutions based on Bluetooth Low energy (BLE), referred to Bluetooth Smart. Given unavailability of Global Positioning System (GPS) in indoor environments, BLE has attracted considerable attention due to its low cost, low energy consumption, and widespread availability in smart hand-held devices. Because of multipath fading and fluctuations in the indoor environment, however, BLE-based localization approaches fail to achieve high accuracies. To address these challenges, different linear and non-linear Bayesian-based estimation frameworks are proposed in this thesis. Among which, the thesis proposes a novel Multiple-Model and BLE-based tracking framework, referred to as the STUPEFY. The proposed STUPEFY framework uses set-valued information and is designed by coupling a non-linear Bayesian-based estimation model (Box Particle Filter) with fingerprinting-based methodologies to improve the overall localization accuracy. With regards to the Item (ii),

there has been an increasing surge of interest on development of advanced Reinforcement Learning (RL) systems. The objective is development of intelligent approaches to learn optimal control policies directly from smart agents' interactions with the environment. In this regard, Deep Neural Networks (DNNs) provide an attractive modeling mechanism to approximate the value function using sample transitions. DNN-based solutions, however, suffer from high sensitivity to parameter selection, are prone to overfitting, and are not very sample efficient. As a remedy to the aforementioned problems, the thesis proposes an innovative Multiple-Model Kalman Temporal Difference (MM-KTD) framework, which adapts the parameters of the filter using the observed states and rewards. Moreover, an active learning method is proposed to enhance the sampling efficiency of the overall system. The proposed MM-KTD framework can learn the optimal policy with significantly reduced number of samples as compared to its DNN-based counterparts.

Acknowledgments

To my life-coach, my eternal cheerleader, best supervisor ever, Prof. Arash: because I owe it all to you. Many Thanks!

My deep gratitude goes first to my supervisor Prof. Arash Mohammadi, whose support, immense knowledge, motivation, and spirit of adventure with regards to my research made this project possible. His guidance, not only in the matters of my research project but on various matters throughout my Master's program has turned out to be very fruitful. I have been extremely lucky to have a supervisor who cared so much about my work, and who responded to my questions and queries so promptly. He has taught me the methodology to carry out the research and to present the research works as clearly as possible. The most important thing I have learnt from him is how to be a good human being. His friendship, empathy, and personal generosity helped make my time at Concordia University enjoyable. He was kind enough to take me in his research group at I-SIP Lab. Without his support, I could not have overcome the numerous challenges that I faced during the course of my Masters program. Thank you, Prof. Arash for providing such a memorable journey in the past two years. I am extending my heartfelt thanks to his wife, Prof. Farnoosh Naderkhani for her acceptance and patience during the discussions I had with him on research works and thesis preparation.

Besides my supervisor, I would like to thank my thesis committee members: Prof. Abdessamad Ben Hamza, Prof. M. Omair Ahmad, and Prof. Wei-Ping Zhu, for their valuable time and input, and also for their great personality and friendly advice, which encouraged me to widen my research from various perspectives. I was fortunate to study under Prof. Ahmad, the chairman of my committee, for Probabilities and Stochastic course. As my teacher, he has taught me more than I could ever give

him credit for here. He has shown me, by his example, what a good scientist (and person) should be.

I would like to thank my family in Iran, especially my parents, and my brothers for their endless support and faith in me throughout my years of study in my life. I cannot express how I am grateful for all the sacrifices that my family have done for me.

A warm and especial thanks to my husband, Babak, for his unfailing support and continuous encouragement throughout my master study and through the process of researching and writing my thesis, who have provided me through moral and emotional support. This accomplishment would have not been possible without him.

And finally, last but by no means least, I thank my fellow lab mates for the stimulating discussions, and also for all the fun we have had in the last two years, it was great sharing the laboratory with all of you during the last two years.

Contents

List of Figures

List of Tables

Abbreviation

Abbreviation	Description
AI	Artificial Intelligence
AoA	Angle of Arrival
BC	Bhattacharyya Coefficient
BD	Bhattacharyya Distance
BLE	Bluetooth Low Energy
CMACs	Cerebellar Model Articulation Controllers
DNN	Deep Neural Networks
DRL	Deep Reinforcement Learning
DQN	Deep Q-Netwrok
EM	Expectation Mean
FPKF	Fixed-Point Kalman Filter
GM	Gaussian Model
GMM	Gaussian Mixture Model
GMR	Gaussian Mixture Reduction
GPTD	Gaussian Process Temporal Difference
GSF	Gaussian Sum Filter
GPS	Global Positioning System
IoT	Internet of Things
IoAT	Internet of Autonomous Things
IoT-C	Internet of Things Clouds

IoRT	Internet of Robotic Things
IR	Infrared
ISM	Industrial, Scientific and medical
KF	Kalman Filter
KLD	Kullback–Leibler divergence
K-NN	K-Nearest Neighbor
KTD	Kalman Temporal Difference
LBS	Location based Services
LSTD	Least Square Temporal Difference
MC	Monte-Carlo
MMAE	Multiple Model Adaptive Estimation
MM-KTD	Multiple Model Kalman Temporal Difference
MMSE	Minimum Mean Square Error
MoE	Mixture of Experts
MSE	Mean Square Error
NFQ	Neural Fitted Q Learning
NLOS	Non Line-of-Sight
PDF	Probability Density Function
PDR	Pedestrian Dead-Recking
PF	Particle Filter
RBFs	Radial Basis Functions
RF	Radio Frequency
RGD	Restricted Gradient Descent
RL	Reinforcement Learning
RSSI	Received Signal Strength Indicator
SBPF	Set-valued Box Particle Filtering
SMC	Sequential Monte Carlo
TD	Temporal Difference

ToF	Time of Flight
UKF	Unscented Kalman filter
WD	Wasserstein Distance

Chapter 1

Thesis Introduction

Recent developments and advancements in signal and information processing, machine learning, communication systems, and networking technologies have resulted in the widespread emergence of the promising paradigm of the Internet of Things (IoT) [1–5]. The IoT is a new and emerging paradigm rapidly gaining ground in different applications of significant engineering importance particularly for development of smart buildings. The basic motivation behind the IoT concept is to provide advanced residential and enterprise solutions through the latest technologies in an energy efficient, secure, and reliable fashion without jeopardizing the service and comfort level. Of particular interest to this thesis is application of IoT-based technologies within the context of autonomous agents performing pre-defined tasks in an indoor environment. An autonomous agent needs to maintain a model of its surrounding physical environment while localizing itself within that environment. In brief, the focus of the thesis is on the following two research areas when it comes to autonomous agents providing Location-Based Services (LBSs) in an indoor environment: (i) *Self-Localization*, which is the autonomous agent's ability to obtain knowledge of its own location, and; (ii) *Decision Support System*, which refers to an autonomous agent's ability to perform optimal actions towards achieving pre-defined objectives.

The first area, i.e., indoor localization for providing LBSs, is of primary importance within the IoT context. Localization of an autonomous agent, however, becomes particularly challenging in Global Positioning System (GPS)-denied indoor environments. In the past, WiFi networks were the main-stream technology used for indoor localization. However, due to the low cost and low

energy consumption, Bluetooth Low Energy (BLE)-based localization has attracted a great surge of recent interest as an efficient alternative [6–8]. In this context, the main localization approach is to use the Received Signal Strength Indicator (RSSI) [9–12]. RSSI-based solutions, however, are prone to multipath fading and drastic fluctuations in the indoor environment, necessitating the need to resort to multitude of advanced signal processing solutions. In this regard, the focus of the thesis is on development of advanced linear/nonilnear and Gaussian/non-Gaussian Bayesian frameworks to improve the overall BLE-based localization performance.

For the second area (Item (ii) defined above), i.e., design of a localized decision support system for an autonomous agent, the focus of the thesis is on Reinforcement Learning (RL)-based solutions. Inspired by exceptional learning capabilities of human beings, RL systems have emerged aiming to form optimal control policies merely by relying on the knowledge about the past interactions of an autonomous agent with its environment. Different from most forms of machine learning techniques, e.g., supervised learning, an RL system does not require availability of carefully labelled data, which is typically difficult to acquire. On the contrary, to choose the best action possible at each time step, a RL system rather relies on the reward (feedback) received from each move (action) [13–16]. Given their phenomenal potentials, there has been an increasing surge of interest on advancement of RL systems as the decision making component of an autonomous agent. In this regard, Deep Reinforcement Learning (DRL) solutions provide an attractive mechanism to guide agents' decisions. DRL-based solutions, however, suffer from high sensitivity to parameters selection, are prone to overfitting, and are not very sample efficient. To deal with these issues, the thesis use principles from Bayesian-estimation techniques developed for indoor localization and proposes a novel multiple-model RL framework that promotes superior sample efficiency.

1.1 Indoor Localization

Extensive amount of research work has been performed over the years leading to advancement of several indoor positioning/localization systems. Such system can be classified based on the utilized signal and measurement technologies [17]. Generally speaking, different indoor localization systems can be categorized into the following groups based on the incorporated signal technologies:

(1) Optical Systems.

(2) Ultrasound-based systems.

(3) Infrared (IR)-based systems.

(4) Radio Frequency (RF)-based systems, e.g., Wi-Fi, RFID, Zigbee, Bluetooth, UWB, and FM.

In recent years, numerous indoor localization systems have been developed based on the latter category, i.e, the RF signals, because of their unique properties such as strong penetration power through obstacles/walls. These technologies can be built with different methods such as Filtering, Lateration, Dead Reckoning, Fingerprinting, and received signal strength. A reliable indoor localization system should possess specific set of important features including:

- Having low cost for both the hardware and software installation.
- Being adaptive in different situations.
- Being portable.
- Being robust for implementation in dynamic environments.

It has been shown that BLE protocol can provide the aforementioned features effectively. The most important feature of the BLE protocol is its widespread availability in smart phones, robotics systems, and tablets making it an effective standard technology for development of IoT applications.

1.1.1 Bluetooth Low Energy

Bluetooth Low Energy is a wireless personal area network technology, which functions in 2.4 GHz frequency band, referred to as the Industrial, Scientific and medical (ISM). BLE provides same communication range as the conventional Bluetooth protocol while significantly reduces the power consumption. This low energy technology makes the BLE an efficient alternative to the classical communication technologies for different sectors such as robotic systems, smart homes, health-care, sport, and fitness.

One of the important differences between BLE and the classical Bluetooth standard is the radio interface. The BLE utilizes 40 channels each one 2 MHz wide, instead of 79 channels with 1 MHz

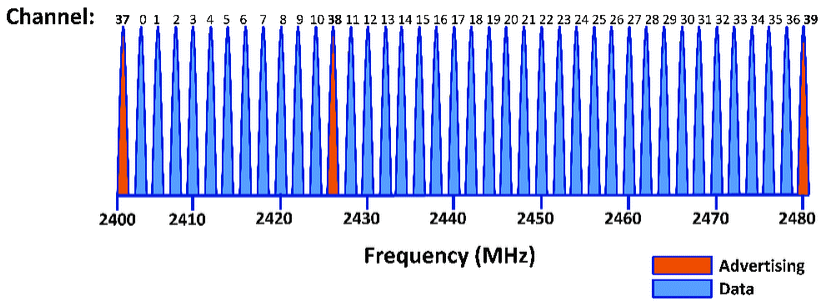


Figure 1.1: BLE frequency channels [18].

wide that is used in classical Bluetooth standard, which makes interoperability unlikely at physical layers [18]. In BLE, also, broadcast channels are reduced to three to reduce the scanning time. Fig 1.1 depicts the BLE frequency channels.

1.1.2 BLE-based Indoor Localization

The data package transmitted by BLE devices include Received Signal Strength Indicator (RSSI) values. The RSSI values can be simply measured by installing an App without the need for any additional hardware. The RSSI values are related to the distance (via pathloss model) between a Bluetooth device and a BLE sensor (typically called a Beacon). By knowing location of BLE sensors as reference nodes, the target's location can be approximated via the pathloss model within an indoor venue. Fig. 1.2 represents an illustrative example of a BLE-based indoor localization system.

1.2 Reinforcement Learning

Inspired by learning mechanisms of different animal species, Reinforcement Learning is emerged as a type of machine learning approach to enable autonomous agents learn through interacting with their environments [19]. Fig. 1.3 represents three different types of machine learning models. Unlike the supervised learning techniques, RL does not require labelled data, instead, it relies on the rewards and penalties received after taking an action. RL is also different from unsupervised learning in terms of the overall objective. The goal in unsupervised learning is to explore the similarities

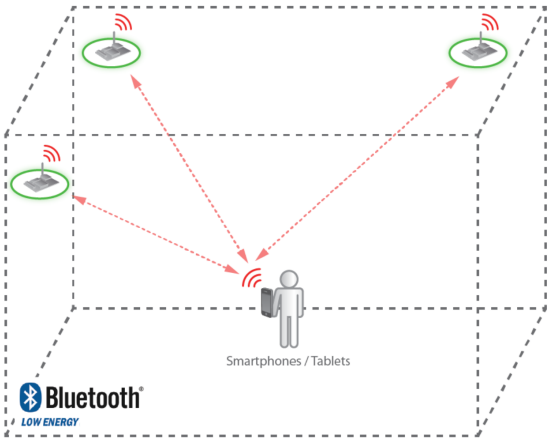


Figure 1.2: BLE-based indoor localization system.[1]

and diversities between data while the goal in a RL problem is to maximize the received rewards during the learning process. Fig. 1.4 depicts the main setup of a RL problem. Some important terminologies used commonly within the RL context are as follows:

- *Environment*: Physical world in which an autonomous agent operates, and which responds to the agent's actions.
- *State*, which shows the current situation of the agent within the environment.
- *Reward*: Feedback received from the environment, which measures the success or failure of the agent's actions.
- *Policy*: The strategy that maps the agent's states to actions.
- *Value*: The expected future rewards received by taking an action in a particular state. The value function relies on the policy by which the agent select its actions.

1.2.1 Reinforcement Learning Applications

In the past, applications of RL were limited due to unavailability of high computational resources. Recent advances in the computational technologies, in particular, evolution of modern

[1] //www.libelium.com/development/waspmote/documentation/bluetooth-low-energy-networking-guide/.

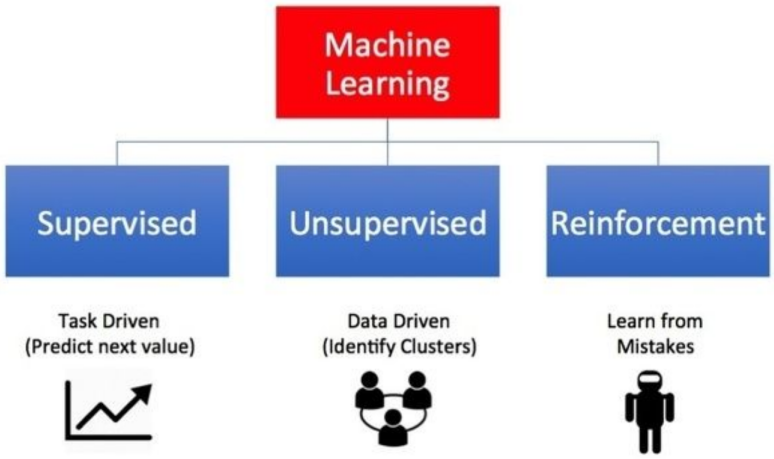


Figure 1.3: Types of machine learning.[2]

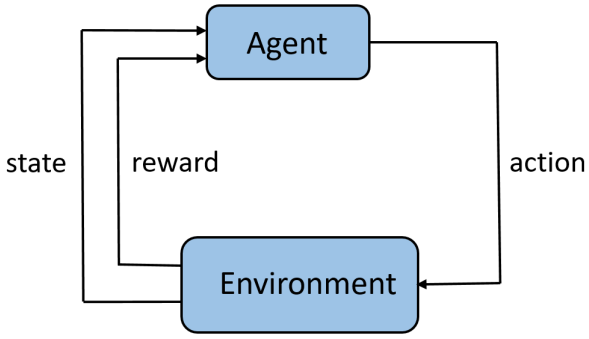


Figure 1.4: Reinforcement Learning illustration.

Graphics Processing Units (GPUs), have rapidly changed the landscape of RL applications. Nowadays, practical applications involving RL problems with large or continuous state spaces are being tackled including but not limited to the following categories [20, 21]:

(1) Building AI for playing computer games such as Atari games, and Backgammon.

(2) Building AI for robotics and industrial automation.

(3) Learning optimal treatment policies in health care and RL-based agents for online stock trading.

[2]http://ginkgobilobahelp.info/?q=Machine+Learning+With+Big+Data++Coursera.

1.2.2 Reinforcement Learning Algorithms

Within the RL context, an autonomous agent following an optimal policy, selects potential actions in such a way that its cumulative reward is maximized over time. In order to obtain the optimal policy, an agent should decide (a trade off) between exploring new states by learning a new policy, or exploiting immediate states based on the currently learnt policy. Since, the cumulative reward is estimated by the value function, the optimal policy is the one that maximizes the value function for a specific state. Solving the RL problem is, therefore, equivalent to learn a way of controlling the system so as to maximize the total reward. Generally speaking, a desirable RL algorithm should have the following key features:

(1) Allowing immediate exploiting, which enables online learning of the system.

(2) Dealing with large and continuous state spaces.

(3) Being sample-efficient, which means learning the optimal policy from as few actions as possible.

(4) Considering uncertainty of the system, which can be used as a useful factor for handling the dilemma between exploration and exploitation.

It is worth mentioning that, all these features are seldom addressed together by state-of-the-art RL algorithms.

1.3 Contributions

The main contributions [22–25] of my thesis research work are briefly outlined below:

(1) **STUPEFY: BLE-based Set-Valued Box Particle** [22]: A novel BLE-based tracking framework, referred to as the STUPEFY, is proposed, which incorporates set-valued information within box particle filtering context. More specifically, the proposed multiple-model STUPEFY framework consists of three integrated modules as briefly described below:

- ***Smoothing Module***, which is developed based on Kalman filtering (KF) with the following two objectives: (i) To reduce the RSSI fluctuations, and; (ii) To construct a

Gaussian Model (GM) of RSSI values to be used within the coordination module for finding the predicted zone of the target. This is achieved via computation of the Bhattacharyya Distance (BD) [26] between the Gaussian output of the KF and the learned GMs of different zones.

- ***Coordination Module***: In the coordination module, similar in nature to fingerprinting approaches, the venue is divided into N_{FP} number of zones (grid of points such as square, hexagonal, or random grid), where Zone l, for ($1 \leq l \leq N_{\text{FP}}$), is defined based on its geometric coordinates. Each zone is referred to as a training point where several observations (signal features) are collected for compensating the time-varying nature (e.g., shadowing/ fading effects) of the associated propagation environment. Different from existing fingerprinting techniques, in the offline phase of the coordination module, we model the RSSI values associated with each zone as a multivariate Gaussian and construct the smallest axes-aligned box containing the ellipsoid associated with each zone to be used for creating set-valued measurements.
- ***Set-Valued Box Particle Filtering (SBPF) Approach (Micro-Localization Module)***: The SBPF model is composed of two steps to micro-locate the target: (i) An approximation of the optimal proposal distribution in terms of a predicted box is used for generating predicted particles from a bivariate Gaussian distribution inscribed in the predicted box (hence boxed particle filtering (PF)), and; (ii) In contrary to conventional PF-based solutions that use point-valued RSSI measurements, the proposed SBPF computes the probability that the measured RSSI from each active beacon belongs to the set identified by the Coordination module (hence set-valued PF).

(2) **Gaussian Mixture-based Indoor Localization via BLE Sensors**: [23] A novel Gaussian Mixture Model (GMM)-based probabilistic framework is proposed for assigning real-time observed RSSI vectors to different fingerprinting zones using Bhattacharyya Coefficient/Distance (BC/BD). The proposed BD-GMM framework consists of the following two data-driven learning components:

- ***Offline Phase***: First, raw RSSI values are collected from active BLE beacons across the

venue, which are then smoothed via a KF. Different from existing fingerprinting techniques, the RSSI values associated with each zone is then modeled with a multivariate GMM.

- ***Online Phase***: KF is applied on the observed measurements to: (i) First, reduce RSSI fluctuations, and; (ii) Second, provide a GM of the online RSSI vector. This probabilistic output is then compared in distribution with GMMs, learned for each zone, via utilization of BD between Gaussians and GMMs. For estimating the user's location, two scenarios are proposed: (i) The BD distance between the output of the KF and the trained GMM associated with all the zones are calculated and the one with minimum distance is identified as the target's zone, and; (ii) A weighted K-Nearest Neighbor (K-NN) algorithm is applied for location estimation, where the weights are computed based on the BD between the output of the KF and the GMM distributions of K nearest zones to the identified one.

(3) **Gaussian Sum Filtering Coupled with Wasserstein Distance for Non-Gaussian BLE-based Localization** [24]: In the context of linear state estimation, KF was used by assuming that the state and observation noises have single Gaussian distributions. In the scenarios where noises have non-Gaussian behaviour, KF cannot provide reliable results. A Gaussian sum filter (GSF)-based algorithm is proposed to model the non-Gaussian RSSI measurement noise as a GMM. The number of components in the GSF is then collapsed into a single Gaussian term with a novel Wasserstein Distance (WD)-Based Gaussian Mixture Reduction (GMR) algorithm. The proposed framework consists of the following two components:

- ***Offline Phase***: After measuring the RSSI values, the values at each zone is modelled with a multivariate GM distribution.
- ***Online Phase***: The proposed novel GSF algorithm, which is an extension of our previous works on the RSSI smoothing, is applied on the real-time RSSI values. This is achieved via incorporation of a WD-based clustering GMR for smoothing the RSSI values and modeling the non-Gaussian RSSI measurement noise as a GMM. Output mixtures are then compared in distribution with GMs, learned for each zone, via utilization

of the BD.

(4) **MM-KTD: Multiple Model Kalman Temporal Differences for Reinforcement Learning** [25]: An innovative Multiple Model Kalman Temporal Difference (MM-KTD) framework is proposed for the value function approximation to address the problems of Deep Neural Networks (DNN)-based value function approximation solutions including overfitting, high sensitivity to parameter selection, and not being very sample efficient. The proposed MM-KTD framework benefits from the following characteristics:

- ***Adoption of an Off-Policy Q-Learning***: To improve the sample efficiency of the proposed MM-KTD, the off-policy Q-learning is adopted to learn the optimal policy from the behaviour policy.
- ***Adoption of an Innovative MMAE Scheme***: To compensate for the lack of information about the measurement noise covariance as the most important parameter of the filter, an innovative Multiple Model Adaptive Estimation (MMAE) scheme is adopted within the RL process. The MMAE structure updates the mean and covariance of Radial Basis Functions (RBFs), which are used to approximate the value function, by exploiting the restricted gradient descent method.
- ***Incorporation of Active Learning***: Within the proposed MM-KTD framework, the estimated uncertainty of the value functions are exploited to form the behaviour policy leading to more visits to less certain values, therefore, improving the overall learning sample efficiency of the system.

1.4 Thesis Organization

The rest of the thesis is organized as follows:

- Chapter 2 provides an overview of the literature on BLE-based indoor localization and RL-based problems. In addition, this chapter provides the background required to follow developments presented in the reminder of the thesis.

- Chapter 3 considers application of multiple-model (hybrid) solutions for BLE-based tracking. In this regard, the chapter presents the proposed STUPEFY framework, which incorporates set-valued information within box particle filtering context.

- Chapter 4 consists of two parts. First, the proposed probabilistic GMM of the RSSI values together with BD-based K-NN approach are presented. Second, the proposed GSF-based algorithm and the WD-based GMR technique are developed.

- Chapter 5 introduces the proposed MM-KTD framework, which adapts the parameters of the filter using the observed states and rewards for RL problems with continuous state space.

- Chapter 6 concludes the thesis and explains some directions for future research works.

Chapter 2

Literature Review and Background

Different techniques have been investigated and proposed for self-localization and RL-based decision making of an autonomous agent as discussed in the previous chapter. The Bayesian state estimation-based approach, which is the building block of this thesis, is a well known framework for formulating and dealing with such problems. In this chapter, first, an overview of the recent literature on both indoor localization and RL problems are presented. Then, a brief introduction is included on the basic underlying principles of Bayesian family of solutions, i.e., the Kalman Filter (KF) (for linear systems) and the Particle Filter (PF) (for non-linear systems).

2.1 Indoor Localization: Literature Review

Generally speaking, existing BLE-based estimation techniques can be classified into the following main categories: (i) Fingerprinting (learning-based) techniques [17], where the indoor venue is divided into several smaller zones to create a training database in an offline fashion. In the online phase, the receiving RSSI signals are classified via different classification models. K-Nearest Neighbors (K-NN) is one popular classification (matching) algorithms [6, 27–30]; (ii) Pedestrian Dead-Recking (PDR), where the user's movement history obtained from sensors embedded in a hand-held device is used for localization purposes, and; (iii) Bayesian-based approaches via utilization of linear (KF or its variants) or non-linear estimators (e.g., PF) [31]. Accuracy of fingerprinting (Item (i)) is heavily dependent on the training database, therefore, RSSI fluctuations significantly

decrease the achievable accuracy [32]; The PDR (Item (ii)) is dependent on the initial location estimate, therefore, position error accumulates over time degrading the overall performance, and; Fluctuation, unreliability, and instability of the RSSI measurements adversely affect computation of the likelihoods in the Bayesian-based techniques (Item (iii)) [33–37]. It is, however, expected that a hybrid model obtained by combining two or more models can overcome these shortcomings and improve the localization accuracy. While single-model BLE-based tracking has been investigated from different aspects, multi-model (hybrid) solutions [38–42] are still in their infancy. Bayesian state estimation approaches have been widely explored in target tracking [43, 44] since KF appeared in 1960 [45]. Kalman filter is an estimator for linear state-space models with Gaussian additive noises, however, in the systems with non-Gaussian behaviour, KF cannot provide reliable results [46]. Non-Gaussian behavior is a common phenomenon in most of real-valued systems. Gaussian mixture (GM) models are used for modeling non-Gaussian densities. For such problems, Gaussian Sum Filter (GSF) [46] are rich solutions to approximate the non-Gaussian noise with GMMs. While KF-based algorithms approximate the state probability density function (PDF) with a single Gaussian, GSF approximates the PDF with a weighted sum of Gaussian distributions. But, the number of GM components tends to increase exponentially during the time resulting in introducing of significant computational overhead. To tackle this challenge, at each step of GSF algorithm, GMs is approximates with lower number of Gaussian components by using the Gaussian mixture reduction (GMR) algorithms [47]. Generally speaking, the GMR algorithms are classified [26] into two main categories: (i) *Bottom-up approaches*, where the desire GM starts with a single Gaussian function and iteratively add more components to reach to the desired GMs model, and; (ii) *Top-down approaches*, where the number of Gaussian mixtures are decreased gradually either by removing an unimportant component or by using the clustering methods that merges the similar components. This reduction is done locally or globally based on different information-probabilistic similarity measures such as the Kullback–Leibler divergence (KLD) [48] or the BD. Then the similar components are replaced by a single Gaussian component. However, both KLD and BD optimize the information gain achieved by GMR, in many applications like machine learning and computer vision, the Geometric shape of the GM is important. WD computes the geometric shape difference between two PDFs by minimizing the cost of transferring one PDF to another [44].

2.2 Reinforcement Learning: Literature Review

When an accurate model of the environment is available, methods such as dynamic programming [19] may be used to find the optimal policy. An accurate model of the environment, however, is rarely available in practice leading to the need to resort to model-based RL or model-free approaches. In the model-based RL approaches, the goal is to learn the model of the system using the past transitions of the system and then apply a model-based control scheme (e.g., model predictive controller [20]) to reach the desired goal [21]. While model-based RL approaches are proven to be sample efficient, their performance, typically, degrades over time by the uncertainties and bias in the constructed model of the environment. On the other hand, model-free approaches directly acquire the optimal policy for the system, without developing a model of the system. Due to their superior asymptotic performance, model-free approaches are commonly preferred over their model-based counterparts especially when sampling the agent's past trajectories is inexpensive.

A common practice in model-free RL schemes is to calculate the state (or alternatively state-action) value function for each state. While this can be done with relative ease for a system with limited number of states [49], it is impractical to find the value function for each state when the number of states is large or the states are continuous (due to the curse of dimensionality [50]). In such cases, which entail most realistic applications, the value function needs to be approximated. In a large group of works, e.g., [51–55], artificial neural networks were employed to approximate the value function over the entire state-space. Despite few successful trials, most early attempts of such were not very successful due to the overfitting problem. However, this problem was overcome in [56], where simple though efficient techniques were employed to avoid the overfitting problem. This work was later extended to systems with continuous actions [57]. However, it was shown later that such approaches are highly sensitive to parameter selection, therefore, required to be revised further to be applicable to a larger variety of problems. Despite continuous attempts [58–61] to overcome such problems, the training of the neural network has remained an open problem in the field.

Another well-practiced approach to approximate the value function is to employ a set of weighted local estimators and convert the approximation problem to a weight estimation problem. Various

local estimator were proposed in the literature, among which Radial Basis Functions (RBFs) [62], and Cerebellar Model Articulation Controllers (CMACs) [63] are most popular. It was shown that RBFs are more suitable than CMACs in systems with continuous states, due to their continuous nature [64]. More recently, Fourier basis was proposed as the local estimator function, however, the performance of the system was shown to be comparable to those using RBFs [65]. Due to its advantages, we exploit the RBFs for the value function approximation. The parameters of the RBFs are usually computed based on the knowledge of the problem. However, it is possible to adapt these parameters using the observed transitions to enhance the autonomy of the method. Cross entropy and gradient descent methods were proposed by [66] for that matter. Stability of the latter was later enhanced using a restrictive technique in [67], which is adopted in this work.

Once the structure of the value function is determined, a suitable algorithm has to be picked to train the value function approximation. Various approaches were proposed in the literature to gradually bring the value approximations close to their real values in all states. These methods were generally categorized as bootstrapping approaches (such as Fixed-Point Kalman Filter or FPKF [68]), residual approaches (e.g., Gaussian Process Temporal Difference or GPTD [69]), and projected fixed-point approaches (e.g., Least Square Temporal Difference or LSTD [70]). Among these approaches, Kalman Temporal difference (KTD) [71,72] stands out as it provides not only the Minimum Mean Square Error (MMSE) estimation of the value function (given the selected structure), but also their uncertainty in terms of their error covariance which could be exploited further to reach higher sample efficiency [72]. It was also shown in [72] that GPTD is a special case of KTD. However, like other Kalman-based schemes, KTD requires the parameters of the filter (such as process and measurement noise covariances) to be known a priori, which is not the case in most practical scenarios.

Filter parameter estimation is a well-studied problem for Kalman filters and has led to numerous adaptive schemes in the literature. These methods may be roughly categorized as innovation-based adaptive methods (e.g., [73] and multiple model adaptive schemes (e.g., [74]). The latter has the advantage of fast adaptability when the mode of the system is changing. Most recently, the effective methods in multiple model adaptive estimation (MMAE) was discussed in [75], where various averaging and weighting schemes were proposed and compared to achieve superior results. Multiple

model approaches were previously used in RL problems. A model-based multiple model approach was introduced in [76] where the uncertainty of the system model was challenged using a set of models for the system. Moreover, a model selection approach was proposed in [77] for a multiple model KTD to overcome the filter parameter uncertainty problem, yet the models and the selection scheme were naive and therefore was not suitable for more general tasks.

2.3 Bayesian Estimation: Formulation

Throughout the thesis, we use the following notation: Non-bold capital letter X denotes a scalar variable; Lowercase bold letter $\boldsymbol{x}$ represents a vector, and; Capital bold letter $\boldsymbol{X}$ denotes a matrix. Transpose of a matrix $\boldsymbol{X}$ is denoted $\boldsymbol{X}^T$. We consider an estimation problem where its states is expresses as a vector of n_x state variables:

$$\boldsymbol{x} = [X^1, X^2, \ldots, X^{n_x}]^T. \tag{2.1}$$

The transition of states over time and the measurements models are represented as follows

$$\boldsymbol{x}_k = \boldsymbol{f}\left(\boldsymbol{x}_{k-1}\right) + \boldsymbol{w}_k \tag{2.2}$$

$$\text{and} \quad \boldsymbol{z}_k = \begin{bmatrix} Z_k^{(1)} \\ \vdots \\ Z_k^{(N_b)} \end{bmatrix} = \underbrace{\begin{bmatrix} h^{(1)}(\boldsymbol{x}_k) + v_k^{(1)} \\ \vdots \\ h^{(N_b)}(\boldsymbol{x}_k) + v_k^{(N_b)} \end{bmatrix}}_{\boldsymbol{h}(\boldsymbol{x}_k)+\boldsymbol{v}_k}, \tag{2.3}$$

where $\boldsymbol{z}_k \in \mathbb{R}^{N_b}$ denotes a sensor measurement vector at iteration k; functions $\boldsymbol{f}(\cdot)$ and $\boldsymbol{h}(\cdot)$, respectively, are the state and observation models; terms $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ represent uncertainties and are assumed to be mutually uncorrelated and can have white Gaussian or non-Gaussian distributions.

In sequential Bayesian estimation, the current state variables of vector $\boldsymbol{x}_k$ (e.g., 2-dimensional location of a target within an area) at time step k only depends only on the previous values $\boldsymbol{x}_{k-1}$,

i.e.,

$$P\big(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{z}_{1:\,k-1}\big) = P\big(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}\big), \tag{2.4}$$

where $P\big(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}\big)$ is known as the *state transition model* of the estimation. Based on the Bayesian conditional independence rule, the observation vector $\boldsymbol{z}_k$ is conditionally independent of the prior states variables given current state values $\boldsymbol{x}_k$, i.e.,

$$P\big(\boldsymbol{z}_k|\boldsymbol{x}_{0:k}\big) = P\big(\boldsymbol{z}_k|\boldsymbol{x}_k\big). \tag{2.5}$$

The final aim in a Bayesian framework is to estimate the conditional filtering density $P(\boldsymbol{x}_k|\boldsymbol{z}_{1:k},)$, i.e., the probability distribution of the state variables for all time instances $k > 0$ given the received observations upto and including the current iteration. Using the Bayes' rule, the filtering density can be calculated as follows

$$P\big(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}\big) = \frac{\overbrace{P(\boldsymbol{z}_k|\boldsymbol{x}_k)}^{\text{Likelihood}}\overbrace{P(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})}^{\text{Predicted Density}}}{\underbrace{P(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})}_{\text{Normalization}}}, \tag{2.6}$$

where denominator $P(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})$ is independent of the state variables and can be expressed as the normalizing constant, i.e., $P(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1}) = \alpha$ and the term $P(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})$ can be computed in terms of the *state transition model* and the filtering density $P(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{k-1})$ as follows

$$P\big(\boldsymbol{x}_k|\boldsymbol{z}_{1:\,k-1}\big) = \int P(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) \times P(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:\,k-1}) d\boldsymbol{x}_{k-1}. \tag{2.7}$$

Eq. (2.6) is referred to as the *measurement update step*, and Eq. (2.7) is referred to as the *prediction step*. The Bayesian state-estimation problems can be divided as:

1 *Prediction Update*: Given $P(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:\,k-1})$ compute $P(\boldsymbol{x}_k|\boldsymbol{z}_{1:\,k-1})$.

2 *Normalization Update*: Compute the normalization factor $P(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})$.

3 *Measurement Update*: Using the likelihood function $P(\boldsymbol{z}_k|\boldsymbol{x}_k)$ to compute $P(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$.

One method, referred to as the maximum a posteriori (MAP) estimator, obtains the state estimate $\hat{\boldsymbol{x}}_k$ by determining the value of $\boldsymbol{x}_k$ that maximizes $P(\boldsymbol{x}_k|\boldsymbol{z}_{1:\,k})$. When several sensors produce their own measurements vector of the states $\boldsymbol{z}_k^{(l)}$ for $(1 \leq l \leq N)$, the conditional probability $P(\boldsymbol{z}_k^{(l)}|\boldsymbol{x}_k)$ then acts the role of a sensor model and can be used in the Bayesian state estimation algorithms. The conditional independence of Bayes' rule in the multi-sensor situation, results in the following likelihood function

$$P\big(\boldsymbol{z}_k|\boldsymbol{x}_k\big) = P\big(\boldsymbol{z}_k^{(1)}, \ldots, \boldsymbol{z}_k^{(N)}|\boldsymbol{x}_k\big) = \prod_{l=1}^{N} P\big(\boldsymbol{z}_k^{(l)}|\boldsymbol{x}_k\big). \tag{2.8}$$

By replacing Eq. (2.8) in Eq. (2.6), we have

$$P\big(\boldsymbol{x}_k|\boldsymbol{z}_k^{(1)}, \ldots, \boldsymbol{z}_k^{(N)}\big) = \alpha P\big(\boldsymbol{x}_k|\boldsymbol{z}_{1:\,k-1}\big) \prod_{l=1}^{N} P\big(\boldsymbol{z}_k^{(l)}|\boldsymbol{x}_k\big), \tag{2.9}$$

In the next section, we introduce the KF algorithm as a classical Bayesian estimator.

2.3.1 Kalman Filter

The Kalman filter is a Bayesian estimation algorithm, which estimates the unknown state variables when the system and observation dynamics are linear. In the prediction step, the KF estimates the current state variables, along with their uncertainties without incorporation of the newly observed measurement. Once the measurement, corrupted with some amount of random noise, is observed , the predicted estimates are updated using a weighted average, with more weight being given to estimates with higher certainty. Kalman filters are used to estimate states based on the MMSE. The prediction model is as follows

$$\boldsymbol{x}_k \quad = \quad \boldsymbol{F}_k \boldsymbol{x}_{k-1} + \boldsymbol{w}_k, \tag{2.10}$$

where $\boldsymbol{F}_k$ is the state transition matrix applied to the previous state vector, and $\boldsymbol{w}_k$ is the process noise vector that is assumed to be zero-mean Gaussian with the covariance $\boldsymbol{Q}_k$, i.e., $\mathcal{N}(0, \boldsymbol{Q}_k)$. The

transition model is paired with the linear measurement model as:

$$\boldsymbol{z}_k = \boldsymbol{H}_k \boldsymbol{x}_k + \boldsymbol{v}_k \tag{2.11}$$

where $\boldsymbol{z}_k$ is the measurement vector, $\boldsymbol{H}_k$ is the measurement matrix, and $\boldsymbol{v}_k$ is the measurement noise vector that is assumed to be zero-mean Gaussian with the covariance R_k , i.e., $\boldsymbol{v}_k = \mathcal{N}(0, \boldsymbol{R}_k)$. The KF provides the optimal solution for these two models based on the following recursions

Prediction Step:

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}_k \hat{\boldsymbol{x}}_{k-1|k-1} \tag{2.12}$$

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_k \boldsymbol{P}_{k-1|k-1} \boldsymbol{F}_k^T + \boldsymbol{Q}_k. \tag{2.13}$$

Update Step:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \big(\boldsymbol{z}_k - \boldsymbol{H}_k \hat{\boldsymbol{x}}_{k|k-1}\big) \tag{2.14}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T \big(\boldsymbol{H}_k \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T + \boldsymbol{R}_k\big)^{-1} \tag{2.15}$$

$$\boldsymbol{P}_{k|k} = \boldsymbol{P}_{k|k-1} - \boldsymbol{K}_k \boldsymbol{H}_k^T \boldsymbol{P}_{k|k-1}. \tag{2.16}$$

In the KF with its transition and measurement model and its statistical properties, the posterior follows a Gaussian distribution, i.e.,

$$P(\boldsymbol{x}_k | \boldsymbol{Z}_k) \sim \mathcal{N}(\hat{\boldsymbol{x}}_{k|k}, \boldsymbol{P}_{k|k}). \tag{2.17}$$

This section completes the introduction to the KF for the linear systems with Gaussian uncertainties. Next, we review particle filtering for nonlinear/non-Gaussian estimation problems.

2.3.2 Particle Filter

The KF performs poor for nonlinear systems with non-Gaussian uncertainties. Alternatively, one can utilize numerical Sequential Monte Carlo (SMC) solutions, also referred to as the particle filters (PFs) [34,78], as approximates to the Bayesian estimators. A PF is a recursive, Bayesian state estimator that uses a set of particles to approximate the non-Gaussian posterior distribution. The set

of all particles is utilized to produce the final state estimate. PF is useful especially when the problem includes highly nonlinear or complicated system models and/or non-Gaussian distributions. To formulate the PF, we need to define transition model and measurement model, which are described respectively as

$$\boldsymbol{x}_k = \boldsymbol{g}(x_{k-1}) + \boldsymbol{w}_k, \tag{2.18}$$

$$\text{and} \quad \boldsymbol{z}_k = \boldsymbol{h}(x_k) + \boldsymbol{v}_k, \tag{2.19}$$

where both $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are mutually independent sequences with known probability density functions and $\boldsymbol{g}(.)$ and $\boldsymbol{h}(.)$ are known functions. Once the initial system parameters such as the number of particles N_p, the initial values of particles and the observations are specified, the particles $\mathbb{X}_k^{(i)}$ for $(1 \leq i \leq N_p)$ and their associated weights $(W_k^{(i)})$ at time index k can be obtained based on the two main steps of the PF as:

$$\text{Prediction Step:} \qquad \mathbb{X}_k^{(i)} \sim P\big(\boldsymbol{x}_k | \boldsymbol{x}_{0:k-1}, \boldsymbol{z}_{1:k}\big). \tag{2.20}$$

The next step is to update the weights as follows

$$\text{Observation Update Step:} \qquad W_k^{(i)} \propto W_{k-1}^{(i)} \frac{P\big(\boldsymbol{z}_k | \mathbb{X}_k^{(i)}\big) P\big(\mathbb{X}_k^{(i)} | \mathbb{X}_{k-1}^{(i)}\big)}{P\big(\boldsymbol{x}_k | \boldsymbol{x}_{0:k-1}, \boldsymbol{z}_{1:k}\big)}, \tag{2.21}$$

where the likelihood function $P(\boldsymbol{z}_k | \mathbb{X}_k^{(i)})$ is derived from the observation equation. The overall filtering distribution of the state vector at iteration $k-1$ can be expressed in terms of the particles and their associated weights as

$$P(\boldsymbol{x}_{k-1} | \boldsymbol{z}_{1:k-1}) \approx \sum_{i=1}^{N_p} W_{k-1}^{(i)} \delta\big(\boldsymbol{x}_{k-1} - \mathbb{X}_{k-1}^{(i)}\big), \tag{2.22}$$

where $\delta(\cdot)$ denotes the Dirac delta function. This completes the introduction of the PF.

2.4 Conclusion

In this chapter, an overview of the existing solutions proposed previously in the literature for both BLE-based indoor localization and RL-based systems are presented. The basics of Bayesian estimation approaches are also discussed together with a brief introduction to Kalman filtering and Particle filtering as the required background material.

Chapter 3

STUPEFY: BLE-based Set-Valued Box Particle

With the rapid emergence of IoT, we are more and more surrounded by smart connected devices with integrated sensing, processing, and communication capabilities. As stated previously, BLE, referred to as Bluetooth Smart, is considered as the main-stream technology to perform identification and localization/tracking in IoT applications. While single-model BLE-based tracking has been investigated from different aspects, application of multi-model (hybrid) solutions are still in their infancy. In this regard, the chapter proposes a novel BLE-based tracking framework, referred to as the STUPEFY, which incorporates set-valued information within box particle filtering context. More specifically, the proposed multiple-model STUPEFY framework consists of the following three integrated modules:

(i) The Smoothing Module developed based on Kalman filtering to reduce the RSSI fluctuations and facilitate comparison of Gaussian models of the RSSI values in distribution with the learned ones;

(ii) A learning-based model (Cooperation Module) utilized in an intuitive fashion to provide/construct a coarse estimate of the target's location together with the smallest axes-aligned box containing the ellipsoid associated with each zone's learned RSSI distribution, and;

(iii) A novel set-valued box particle filtering (SBPF) approach (Micro-Localization Module).

The proposed STUPEFY framework is evaluated via proof-of-concept experiments based on real BLE datasets and results illustrate its significant potentials in terms of the improving the overall achievable tracking accuracy.

The rest of this chapter is organized as follows: Section 3.1 presents the proposed STUPEFY framework. Simulation results are provided in Section 3.2. Finally, Section 3.3 concludes the chapter.

3.1 The STUPEFY Framework

A 2-D indoor licalization/tracking problem is considered with a single user walking within the surveillance region equipped with $N_b > 1$ number of BLE sensors. The following general non-linear state-space model is considered

$$\boldsymbol{x}_k = \boldsymbol{f}\left(\boldsymbol{x}_{k-1}\right) + \boldsymbol{w}_k \tag{3.1}$$

$$\text{and} \quad \boldsymbol{z}_k = \begin{bmatrix} Z_k^{(1)} \\ \vdots \\ Z_k^{(N_b)} \end{bmatrix} = \underbrace{\begin{bmatrix} h^{(1)}(\boldsymbol{x}_k) + v_k^{(1)} \\ \vdots \\ h^{(N_b)}(\boldsymbol{x}_k) + v_k^{(N_b)} \end{bmatrix}}_{\boldsymbol{h}(\boldsymbol{x}_k)+\boldsymbol{v}_k}, \tag{3.2}$$

where $\boldsymbol{z}_k \in \mathbb{R}^{N_b}$ denotes the sensor's measurement vector at iteration k; $\boldsymbol{x}_k = [X_k, Y_k]^T \in \mathbb{R}^2$ denotes the state vector (2-D location of the target); functions $\boldsymbol{f}(\cdot)$ and $\boldsymbol{h}(\cdot)$, respectively, are the state and observation models; terms $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ represent uncertainties and are assumed to be mutually uncorrelated white Gaussian noises. The proposed STUPEFY framework consists of three main sub-systems as shown in Fig. 3.1 and are covered in the following sub-sections.

3.1.1 Smoothing Module

The initial step of the proposed SBPF is KF-based smoothing [43, 79] with two objectives:

(i) To smooth fluctuations in the RSSI values, and;

(ii) To construct a Gaussian model of the RSSI values to be used within the coordination module

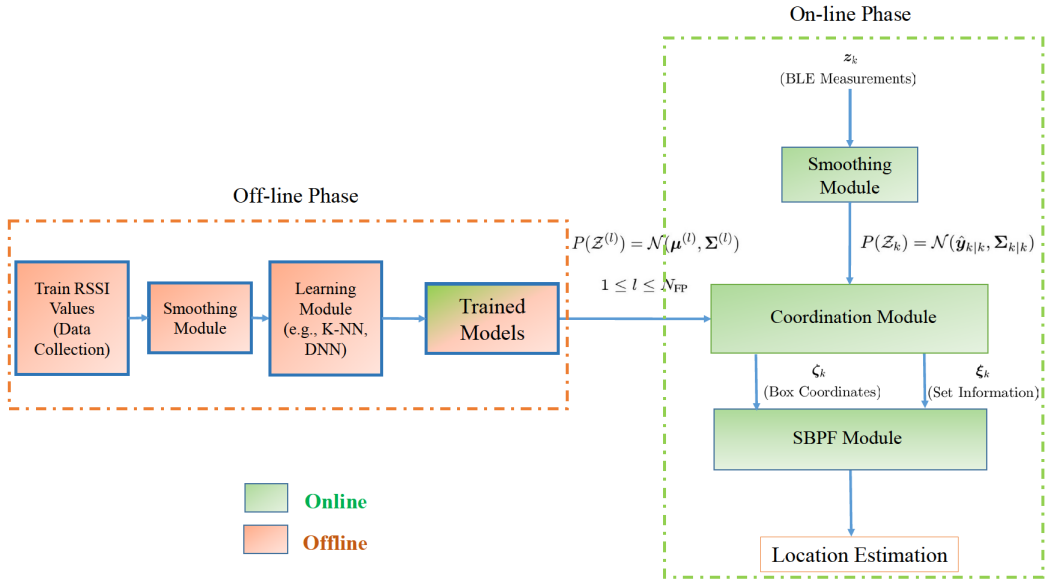


Figure 3.1: Architecture of the proposed STUPEFY framework.

for finding the predicted zone of the target via BD [26].

In other words, Gaussian approximation of the smoothed RSSI values provided by the KF filter is compared in distribution with the ones learned to model each fingerprinting zone. In this regard, the RSSI values obtained from the j^{th} active BLE beacon are given by

$$Z_k^{(j)} = \underbrace{-10\, N\, \log(\frac{D_k^{(j)}}{D_0}) + C_0}_{h^{(j)}(\boldsymbol{x}_k)} + v_k^{(j)}, \tag{3.3}$$

where $D_k^{(j)} = \sqrt{(X_k - X_k^{(j)})^2 + (Y_k - Y_k^{(j)})^2}$ with $\boldsymbol{x}_k^{(j)} = [X_k^{(j)}, Y_k^{(j)}]^T$ denoting 2-D location of the j^{th} sensor; D_0 is the reference distance; C_0 is the average RSSI value at reference distance; N is the pathloss exponent, and; the overall observation vector is constructed as $\boldsymbol{z}_k = [Z_k^{(1)}, \dots Z_k^{(j)}, \dots Z_k^{(N_b)}]^T$. To formulate the KF, we define an intermediate state vector $\boldsymbol{y}_k \in \mathbb{R}^{N_b}$, which models the smoothed RSSIs based on the following state-model

$$\boldsymbol{y}_k = \boldsymbol{y}_{k-1} + \boldsymbol{\nu}_k. \tag{3.4}$$

The measured RSSI values $\boldsymbol{z}_k$ are used as the input observation to the KF with the following observation model

$$\boldsymbol{z}_k = \boldsymbol{y}_k + \boldsymbol{u}_k. \tag{3.5}$$

Terms $\boldsymbol{\nu}_k$ and $\boldsymbol{u}_k$ are zero-mean Gaussian uncertainties used in the smoothing model with their second-order statistics ($\boldsymbol{Q}_{\text{RSSI}}$ and $\boldsymbol{R}_{\text{RSSI}}$) being learned through an initial calibration phase. The output of the KF at each iteration is, therefore, denoted by $\mathcal{N}(\hat{\boldsymbol{y}}_{k|k}, \boldsymbol{\Sigma}_{k|k})$, which represents a Gaussian model of smoothed RSSI vector.

3.1.2 Coordination Module

In the coordination module, similar in nature to fingerprinting approaches [38–40], the venue is divided into N_{FP} number of zones (grid of points such as square, hexagonal, or random grid), where Zone l, for ($1 \leq l \leq N_{\text{FP}}$), is defined based on its geometric coordinates. Each zone is referred to as a training point where several observations (signal features) are collected for compensating the time-varying nature (e.g., shadowing/fading effects) of the associated propagation environment. The coordination module consists of the following two main phases:

(a) The Offline Phase

This is a data-driven learning approach that can be roughly decomposed into raw signal feature collection (RSSI values collected from active BLE beacons), and fingerprint creation step, i.e., construction of a training database from smoothed (via Smoothing Module) RSSI values.

Different from existing fingerprinting techniques, in the offline phase of the coordination module, we model the RSSI values associated with the l^{th}-zone, for ($1 \leq l \leq N_{\text{FP}}$), as a multivariate Gaussian distribution $p(\mathcal{Z}^{(l)}) \sim \mathcal{N}(\boldsymbol{\mu}^{(l)}, \boldsymbol{\Sigma}^{(l)})$. Based on the learned statistics (i.e., $\boldsymbol{\mu}^{(l)}$ and $\boldsymbol{\Sigma}^{(l)}$), the smallest axes-aligned box $\boldsymbol{\xi}^{(l)}$ containing the ellipsoid associated with the l^{th}-zone is constructed. Fig. 3.2 shows the ellipsoid for three BLE beacons ($N_b = 3$). The smallest axes-aligned bounding box is completely determined by the boundary points, which are obtained by projecting

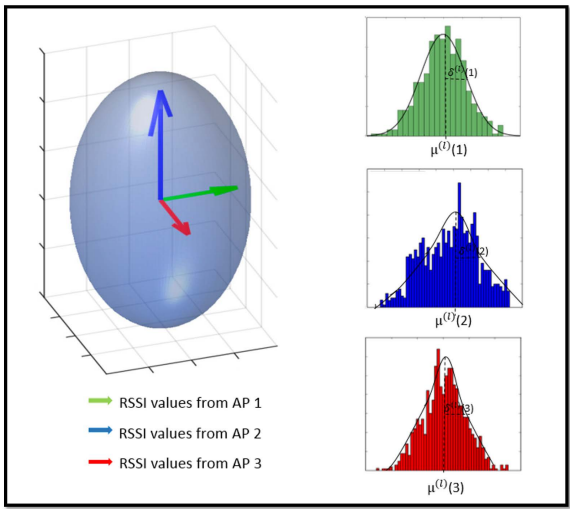


Figure 3.2: Ellipsoid associated with the l^{th}-zone for $N_b = 3$.

the ellipsoid on to different axes. An analytical smallest axes-aligned set is constructed by minimization along the j^{th} coordinate axis as the solution of the following constrained optimization problem

$$\begin{cases} \min_{\boldsymbol{z}} & \boldsymbol{e}^T(j)\boldsymbol{z} \\ \text{subject to} & (\boldsymbol{z}-\boldsymbol{\mu}^{(l)})^T\left(\boldsymbol{\Sigma}^{(l)}\right)^{-1}(\boldsymbol{z}-\boldsymbol{\mu}^{(l)}) \leq \eta^2, \end{cases} \tag{3.6}$$

where $\boldsymbol{e}(j)$ is a $(N_b \times 1)$ vector with one non-zero element equal to 1 at the jth location, and Term η is selected depending on the probability mass the designer chooses to be captured by the ellipsoid. The problem in Eq. (3.6) is convex and admits the following closed-form solution

$$\xi^{(l)}(j)_{\text{min}} = \mu^{(l)}(j) - \eta\sqrt{\Sigma^{(l)}(j,j)}, \tag{3.7}$$

where $\Sigma^{(l)}(j,j)$ denotes (j,j) scalar element of $\boldsymbol{\Sigma}^{(l)}$. Solving the related maximization problem yields the maximum along the j^{th} coordinate axis, which will complete construction of the smallest axes-aligned box $\boldsymbol{\xi}^{(l)}$ for the l^{th} coordination zone.

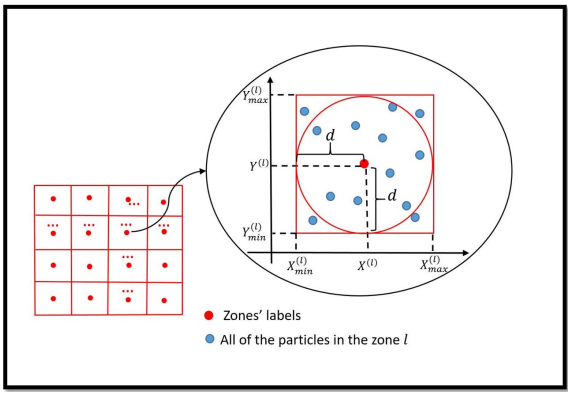


Figure 3.3: An illustrative example showing the box PF approach.

(b) The Online Phase

The online phase consists of pattern matching and post-processing steps. In the online phase, the RSSI vector of a real time user is measured, then compared to the pre-recorded training RSSI vectors to identify the zone of user's location.

Again and in contrary to existing solutions for pattern matching, we propose to measure (via statistical distant measures) the distance between two probability distributions, i.e., $p(\mathcal{Z}^{(l)}) = \mathcal{N}(\boldsymbol{\mu}^{(l)}, \boldsymbol{\Sigma}^{(l)})$ and $p(\mathcal{Z}_k) = \mathcal{N}(\hat{\boldsymbol{y}}_{k|k}, \boldsymbol{\Sigma}_{k|k})$. In particular, in this chapter, we use BD [26] between the two distributions defined as follows

$$d_b(p(\mathcal{Z}_k), p(\mathcal{Z}^{(l)})) = -\ln\Big(\int_{\mathbb{R}^{N_b}} \sqrt{p(\mathcal{Z}_k)p(\mathcal{Z}^{(l)})}\, d\boldsymbol{z}\Big). \tag{3.8}$$

The distance between the measurement vector $\boldsymbol{z}_k$ and all the N_{FP} zones are computed and the one with smallest distance, i.e., $\hat{\mathcal{Z}}_k = \arg\min_l\{d_b(\mathcal{Z}_k, \mathcal{Z}^{(l)})\}$ is selected as the candidate zone. Finally, geometric coordinate of the predicted zone where the target is located together with the learned Gaussian model ($p(\hat{\mathcal{Z}}) = \mathcal{N}(\boldsymbol{\mu}^{(l)}, \boldsymbol{\Sigma}^{(l)})$) representing the RSSI values associated with the selected zone $\hat{\mathcal{Z}}_k$ will be provided as the inputs to the SBPF module, which is described next.

3.1.3 Micro-Localization Module

Once predicted zone of the user is estimated by the coordination module, the proposed SBPF is implemented based on the following two steps to micro-locate the target. In what follows, we assume that l^{th} zone is predicted as the target's zone by the coordination module, i.e., $\hat{\mathcal{Z}}_k = \mathcal{Z}^{(l)}$.

1. Prediction Step: To form predicted particles denoted by $\mathbb{X}_k^{(i)}$, the proposed SBPF uses a box $\boldsymbol{\zeta}_k^{(l)}$ identified by coordinates of the predicted zone (Zone l), which is defined as

$$\boldsymbol{\zeta}_k^{(l)} \triangleq \left\{ \boldsymbol{x} : \min_m \boldsymbol{x}_k^{(l)} \preceq \boldsymbol{x} \preceq \max_m \boldsymbol{x}_k^{(l)} \right\}, \tag{3.9}$$

where the minimum, maximum, and inequalities should be understood component-wise. More specifically, in the prediction step N_p particles are generated within the box (defined in Eq. (3.9)) associated with the predicted zone of the user (provided by the coordination module). In other words, predicted particles are generated via the following proposal distribution

$$\mathbb{X}_k^{(i)} \sim p(\boldsymbol{x}_k \in \boldsymbol{\zeta}_k^{(l)} | \boldsymbol{z}_{k-1}) = p(\min_m \boldsymbol{x}_k^{(l)} \preceq \boldsymbol{x}_k \preceq \max_m \boldsymbol{x}_k^{(l)} | \boldsymbol{z}_{k-1}). \tag{3.10}$$

Intuitively speaking, the proposal distribution defined in Eq. (3.10) can be considered as an approximation of the posterior distribution, which in turn becomes an attractive and intelligent choice for the proposal distribution. In other words, the proposed SBPF uses an approximation $p(\boldsymbol{x}_k \in \boldsymbol{\zeta}_k^{(l)} | \boldsymbol{z}_{k-1})$ of the optimal proposal distribution.

Boxed Particle Generation: The following three scenarios are introduced to generate boxed particles: (i) N_p particles are generated with the ones falling within the identified box (Eq. (3.9)) participating in the update step, i.e., reduced and varying number of particles used in the update steps; (ii) Second, upsampling particles until N_p falling within the predicted box, and; (iii) N_p particles generated from a bivariate Gaussian distribution inscribed in the predicted box (Fig. 3.3).

2. Update Step: In contrary to conventional PF-based [80–82] solutions developed for BLE-based tracking, which form point-valued likelihood $P(\boldsymbol{z}_k | \mathbb{X}_k^{(i)})$ for each particle based on the RSSI values, the proposed SBPF computes the probability that the measured RSSI from each active beacon belongs to the set $\boldsymbol{\xi}_k^{(l)}$ (Eq. (3.7)) identified by the coordination module. In other words, instead of

Algorithm 1 STUPEFY IMPLEMENTATION

Input: $\{\mathcal{Z}^{(l)}\}_{l=1}^{N_{\text{FP}}}$, $\{\mathbb{X}_{k-1}^{(i)}, W_{k-1}^{(i)}\}_{i=1}^{N_p}$, and $\boldsymbol{z}_k$.
Output: $\{\mathbb{X}_k^{(i)}, W_k^{(i)}\}_{i=1}^{N_s}$, $\hat{\boldsymbol{x}}_{k|k}$ and $\boldsymbol{P}_{k|k}$.
Offline Phase:
    Construct training database: $\mathcal{N}(\boldsymbol{\mu}^{(l)}, \boldsymbol{\Sigma}^{(l)}), (1 \leq l \leq N_{\text{FP}})$.
    Calculate $\boldsymbol{\zeta}^{(l)}$, $\xi^{(l)}(j)_{\max}$ and $\xi^{(l)}(j)_{\min}$, via Eq. (3.7).
Online Phase (Smoothing and Coordination Modules):
    Smooth $\boldsymbol{z}_k$ and compute $P(\mathcal{Z}_k) = \mathcal{N}(\hat{\boldsymbol{y}}_{k|k}, \boldsymbol{\Sigma}_{k|k})$.
    Identify predicted zone $\hat{\mathcal{Z}}_k = \arg\min_l \{d_b(\mathcal{Z}_k, \mathcal{Z}^{(l)})\}$.
Online Phase (SBPF Module Prediction Step):
    Boxed predicted particle generation via Eq. (3.10).
Online Phase (SBPF Module Update Step):
    Compute set-valued likelihoods via Eq. (3.11).
    Update particles' wights via Eq. (3.12).

using point-valued measurements ($Z_k^{(j)}$), set-valued [35, 36] observations are used to update particles, i.e., $W_k^{(i)} \propto p\big(\boldsymbol{z}_k \in \boldsymbol{\xi}_k^{(l)} | \mathbb{X}_k^{(i)}\big)$. For instance, for the j^{th} smoothed RSSI observation $Z_k^{(j)}$, set-valued likelihood function is computed as follows

$$\begin{aligned} p\big(\xi^{(l)}(j)_{\min} \leq Z_k^{(j)} \leq \xi^{(l)}(j)_{\max} | \boldsymbol{x}_k\big) &= p\big(\xi^{(l)}(j)_{\min} \leq h^{(j)}(\boldsymbol{x}_k) + v_k^{(j)} \leq \xi^{(l)}(j)_{\max} | \boldsymbol{x}_k\big) \\ &= \Phi\left(\frac{\xi^{(l)}(j)_{\max} - h^{(j)}(\boldsymbol{x}_k)}{\sqrt{R_k^{(j)}}}\right) - \Phi\left(\frac{\xi^{(l)}(j)_{\min} - h^{(j)}(\boldsymbol{x}_k)}{\sqrt{R_k^{(j)}}}\right), \end{aligned} \tag{3.11}$$

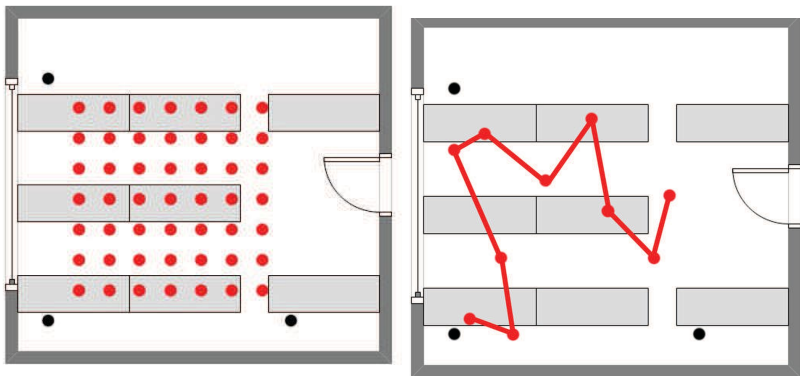
where $\Phi(\cdot)$ is the cumulative Gaussian distribution with zero mean and variance 1. Finally, the weight associated with each particle $\mathbb{X}_k^{(i)}$ is updated as follows

$$W_k^{(i)} = W_{k-1}^{(i)} \frac{p\big(\boldsymbol{z}_k \in \boldsymbol{\xi}_k^{(l)} | \mathbb{X}_k^{(i)}\big) p(\mathbb{X}_k^{(i)} | \mathbb{X}_{k-1}^{(i)})}{p(\boldsymbol{x}_k \in \boldsymbol{\xi}_k^{(l)} | \boldsymbol{z}_{k-1})}. \tag{3.12}$$

Algorithm 1 outlines the steps of the STUPEFY.

3.2 Experimental Results

The proposed STUPEFY localization framework is evaluated via a proof-of-concept experimental testbed, where RSSI measurements are collected in a $(6.0m \times 5.5m)$ low-noise meeting room environment [10]. Three Gimbal Series 10 Beacons, developed by Qualcomm, were used as BLE



(a) (b)

Figure 3.4: Experimental dataset [10]: (a) Fingerprinting data. (b) Test trajectory.

transmitters, which were placed $4m$ apart from one another in the shape of a triangle as shown in Fig. 3.4(a). Fingerprinting points were taken with a 0.5m spacing between the centres, resulting in 49 zones. For testing, several random trajectories are created (one example is shown in Fig. 3.4(b)) based on 10 randomly selected test points for which test RSSI values are computed. First, similar algorithm proposed in [40] is applied on the test data where particles are weighted by their probability computed from the fingerprinting method. Second, proposed STUPEFY algorithm is applied on the test data; in the coordination module K-NN classifier is applied (K = 1,3,5,7,9). The three boxed particle generation scenarios, described in Sub-section II-C, are then applied for varying number of particles ranging from $N_p = 100$ to $N_p = 1100$. The mean square error (MSE) is calculated as a measure of performance. As Table 3.1 shows, K-NN classifier with $K=5$, gives the best accuracy. In comparison to conventional PF approach, the average performance improvements achieved via the proposed STUPEFY framework based on Scenarios (ii) and (iii), are 58% and 83%, respectively. To show the efficacy of the proposed STUPEFY framework, Fig. 3.5 illustrates the mean MSE results and their variations for $K = 5$ as a function of the number of particles, computed based on 100 randomly selected different target tracks. It can be observed that the accuracy achieved from the proposed STUPEFY algorithm significantly outperforms its counterparts. Upper bounds on the computation times associated with Schemes (i)-(iii) are 4, 25, and 12 seconds, respectively, which are computed by averaging over 50 runs based on $K = 5$ and $N_p = 1000$. We note that BLE

Table 3.1: MSE comparison of proposed algorithms for various number of k with $N = 500$.

Algorithm	$K=1$	$K=3$	$K=5$	$K=7$
K-NN Approach	$123cm$	$123cm$	$73cm$	$134cm$
Conventional PF	$43.9cm$	$43cm$	$36cm$	$45.5cm$
Boxed Particle, Scheme (i)	$38.82\ cm$	$38cm$	$36cm$	$36.5cm$
Boxed Particle, Scheme (ii)	$27\ cm$	$25.49cm$	$15cm$	$15cm$
Boxed Particle, Scheme (iii)	$13.4cm$	$10.4cm$	$5.49cm$	$9cm$

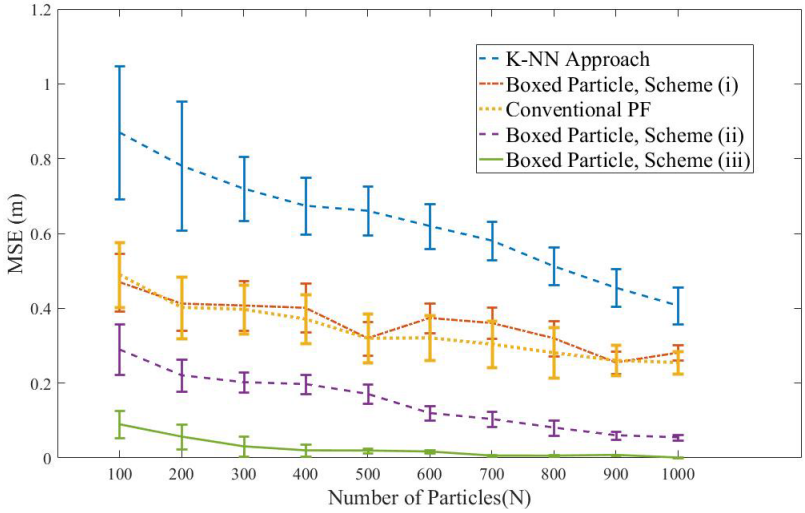


Figure 3.5: MSE computation for PF and STUPEFY (5-NN) as a function of the number of particles.

signals experience more interference in large environments, in presence of obstacles, and/or in non line-of-sight (NLoS) scenarios resulting in higher fluctuations of the RSSI values. In our ongoing research, we are focusing on adaption of mode-matched pathloss models to extend the proposed STUPEFY to better coupe with environmental variations. As a final note, we note that increasing the number of transmitters results in increased computational burden of the offline phase, however, accuracy improvements is expected due to presence of more observations.

3.3 Conclusion

In this chapter, a multi-model BLE-based tracking framework is proposed, referred to as the STUPEFY, which uses set-valued information within box particle filtering. In contrary to existing BLE-based solutions, BD is used for finding a predicted zone by comparing real-time Gaussian

models of the RSSI values; Smallest axes-aligned box containing the ellipsoid associated with each zone is introduced and used for creating set-valued RSSI measurements, and; An approximation of the optimal proposal distribution in terms of a predicted box is used for generating predicted particles. The proposed STUPEFY is evaluated based on real BLE datasets and results illustrate significant potentials in terms of improving overall BLE-based achievable tracking accuracy.

Chapter 4

BLE-based Bayesian Estimation Coupled with Distribution Matched Algorithms

In the previous chapter, RSSI values are assumed to have normal statistical properties. The main challenge is that multipath fading and drastic fluctuations in the indoor environment result in complex Gaussian/non-Gaussian RSSI measurements, necessitating the need to smooth RSSIs for development of BLE-based localization applications. In the first part of this chapter, we assume that RSSI values can be modeled as Gaussian distributions. Then, to deal with the fact that RSSI-based solutions are prone to drastic fluctuations, GMMs are trained to more accurately represent the underlying distribution of the RSSI values. For assigning real-time observed RSSI vectors to different zones, first a Kalman Filter is applied to smooth the RSSI vector and form its Gaussian model, which is then compared in distribution with learned GMMs based on BD and via a weighted K-NN approach. I the second part of the chapter, a Gaussian Sum Filter (GSF) approach is designed to more realistically model the non-Gaussian nature of RSSIs. To maintain acceptable computational load, the number of components in the GSF is collapsed into a single Gaussian term with a novel WD-Based GMR algorithm. The simulation results based on real collected RSSI signals confirm the success of the proposed WD-based GSF framework compared to its conventional counterparts.

The reminder of this chapter is organized as follows: Section 4.1 presents the proposed GMM-based localization framework. Section 4.2 presents the proposed Non-Gaussian BLE-based indoor localization framework, where Gaussian sum filtering is coupled with WD. Finally, Section 4.3 concludes the chapter.

4.1 Gaussian Mixture-based Indoor Localization via BLE Sensors

In this section, we propose a novel BD-GMM framework for assigning real-time observed RSSI vectors to different fingerprinting zones using Bhattacharyya coefficient/distance [26]. More specifically, after measuring the RSSI data, values associated with each location are modelled by a multivariate GMM distribution. In the online phase, KF is applied on the observed measurements to: (a) First, reduce RSSI fluctuations, and; (b) Second, provide a Gaussian model of the online RSSI vector. This probabilistic output is then compared in distribution with GMMs, learned for each zone, via utilization of BD between Gaussians and GMMs [26]. For estimating the user's location, two scenarios are proposed:

(i) The BD between the output of the KF and the trained GMM associated with all the zones are calculated and the one with minimum distance is identified as the target's zone, and;

(ii) A weighted K-NN algorithm is applied for location estimation, where the weights are computed based on the BD between the output of the KF and the GMM distributions of K nearest zones to the identified one.

In what follows, first the problem at hand is formulated in Section 4.1.1 where the proposed GMM-based localization framework is also presented. Data collection and experimental results together with comparisons are then provided in Section 4.1.2.

4.1.1 GMM-based Indoor Localization via BLE Sensors

Generally speaking, for indoor localization with fingerprinting approach, the RSSI values received from active BLE sensors across the surveillance venue are compared with the offline RSSI

Algorithm 2 PROPOSED GMM-BASED ZONE DETECTION

Input: $\{\boldsymbol{RSSI}_n^{(l)}\}_{n=1}^{N_{\text{Train}}}$ for $(1 \leq l \leq N_{\text{FP}})$, and $\boldsymbol{z}_k$.
Output: $\hat{\mathcal{Z}}_k$.

Offline Phase:
  Construct training database $\{\boldsymbol{R}^{(l)}\}_{l=1}^{N_{\text{FP}}}$ based on Eq. (4.1).
  Model each zone with N_c GMM, i.e.,
  $P(\mathcal{Z}^{(l)}) = \sum_{i=1}^{N_c} \phi_i \, \mathcal{N}(\boldsymbol{\mu}_i^{(l)}, \boldsymbol{\sigma}_i^{(l)})$.
Online Phase:
  Smooth $\boldsymbol{z}_k$ and compute $P(\mathcal{Z}_k^{\text{KF}}) = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k)$.
  First Estimation Scenario:
    Compute $d(\mathcal{Z}_k^{\text{KF}}, \mathcal{Z}^{(l)})$ via Eq. (4.3).
    Identify the user's zone via
    $\hat{\mathcal{Z}}_k = \arg\min_l \{d_b(\mathcal{Z}_k, \mathcal{Z}^{(l)})\}$.
  Second Estimation Scenario:
    Find K nearest neighbors to $\boldsymbol{z}_k$ between $\{\boldsymbol{RSSI}_n^{(l)}\}_{n=1}^{N_{\text{Train}}}$ for $(1 \leq l \leq N_{\text{FP}})$ based on $d(\mathcal{Z}_k^{\text{KF}}, \mathcal{Z}^{(l)})$.
    Calculate weights, $\frac{1}{d(\mathcal{Z}_k^{\text{KF}}, \mathcal{Z}^{(l)})}$, associated with the K selected neighbors.
    Compute the weight of each zone $W^{(l)}$.
    Select the zone as follows: $\hat{\mathcal{Z}}_k = \arg\max_l \{W^{(l)}\}$.

values at each zone to identify the most similar zone to the unknown location and associate the centre of selected zone as the predicted location. Common approaches for finding the aforementioned similarity is finding the minimum Euclidean distance between the average RSSI values computed during the training phase with the test RSSI value. On the contrary, the proposed algorithm uses probability distribution of RSSI values instead of their mean during the Offline and Online phases. In other words, we develop a probabilistic BLE-based fingerprinting method, where during the Offline phase the RSSIs associated with each zone is modelled by a multivariate GMM.

For the purpose of localizing the target, the venue is divided into N_{FP} number of monitoring zones (grid of zones such as square, hexagonal, or random grid), where Zone l denoted by $\mathcal{Z}^{(l)}$, for $(1 \leq l \leq N_{\text{FP}})$, is defined based on its centre coordinates. We consider tracking a single target within these zones identified in the surveillance region monitored with N_b number of BLE-enabled sensors. Each zone is referred to as a training point where several RSSI values from all the active N_b sensors are collected for compensating the time-varying nature (e.g., shadowing/fading effects)

of the associated propagation environment. The proposed GMM-based localization framework consists of the following two main phases:

(i) *The Offline Phase*, which is a data-driven learning approach that can be roughly decomposed into collection of raw RSSI values from N_b active BLE beacons across the venue $\boldsymbol{RSSI}_n^{(l)} = [RSSI_n^{(1,l)}, \ldots, RSSI_n^{(N_b,l)}]^T$, for $(1 \leq n \leq N_{\text{Train}})$, and fingerprint creation step, i.e., construction of the GMM training database based on RSSI values.

More specifically, during the Offline phase, N_{Train} number of training RSSI vectors are measured from all N_b sensors collectively represented as follows

$$\boldsymbol{R}^{(l)} = \begin{bmatrix} RSSI_1^{(1,l)}, & \ldots, & RSSI_{N_{\text{Train}}}^{(1,l)} \\ \vdots & \vdots & \vdots \\ RSSI_1^{(N_b,l)}, & \ldots, & RSSI_{N_{\text{Train}}}^{(N_b,l)} \end{bmatrix}, \tag{4.1}$$

for $(1 \leq l \leq N_{\text{FP}})$. Then these N_{Train} training RSSI vectors for Zone $\mathcal{Z}^{(l)}$ collected in matrix $\boldsymbol{R}^{(l)}$ are each modeled with a GMM, denoted by $P(\mathcal{Z}^{(l)})$, consisting of N_c Gaussian components and parameterized by $\theta_i^{(l)} = \{\phi_i^{(l)}, \boldsymbol{\mu}_i^{(l)}, \boldsymbol{\sigma}_i^{(l)}\}$, i.e.,

$$P(\mathcal{Z}^{(l)}) \;\; = \;\; \sum_{i=1}^{N_c} \phi_i \, \mathcal{N}(\boldsymbol{\mu}_i^{(l)}, \boldsymbol{\sigma}_i^{(l)}), \tag{4.2}$$

where $\phi_i^{(l)}$ is the i^{th} component's weight, $\mu_i^{(l)}$ is its mean vector with its associated covariance matrix denoted by $\Sigma_i^{(l)}$. The parameters in Eq. (4.2) are calculated in the Offline phase via the Expectation Maximization (EM) algorithm. As stated previously, different from existing fingerprinting techniques, in the Offline phase of the proposed framework, we model the RSSI values associated with the l^{th}-zone, for $(1 \leq l \leq N_{\text{FP}})$, as a multivariate GMM $P(\mathcal{Z}^{(l)})$ given in Eq. (4.2).

(ii) *The Online Phase*, which consists of pattern matching and post-processing steps. In the testing (Online) phase, the real time RSSI vector is measured, then a KF is applied for two reasons: (a) To smooth fluctuations in the RSSI values, and; (b) To construct a Gaussian model of the RSSIs to be used for finding the predicted zone of the target via the two estimation scenarios described below. In other words, Gaussian approximation of the smoothed RSSIs provided by the KF is compared

with GMM learned to model each fingerprinting zone.

To formulate the KF in this section, we define an intermediate state vector $\boldsymbol{y}_k \in \mathbb{R}^{N_b}$, which models the smoothed RSSIs based on a random walk model $\boldsymbol{y}_k = \boldsymbol{y}_{k-1} + \boldsymbol{v}_k$ as the state-model. The measured RSSI values $\boldsymbol{z}_k$ are used as the input observation to the KF with $\boldsymbol{z}_k = \boldsymbol{y}_k + \boldsymbol{\mu}_k$ as the observation model. Terms $\boldsymbol{v}_k$ and $\boldsymbol{\mu}_k$ are zero-mean Gaussian uncertainties used in the smoothing model with their second-order statistics being learned through an initial calibration phase. The output of the KF at each iteration is, therefore, denoted by $P(\mathcal{Z}_k^{\text{KF}}) = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k)$, which represents a Gaussian model of smoothed RSSI vector. Again and in contrary to existing solutions for pattern matching, we propose to measure (via statistical distant measures) the distance between two probability distributions $P(\mathcal{Z}_k) = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k)$ and $P(\mathcal{Z}^{(l)}) = \sum_{i=1}^{N_c} \phi_i \, \mathcal{N}(\boldsymbol{\mu}_i^{(l)}, \boldsymbol{\sigma}_i^{(l)})$. In particular, in this section we use BD between two distributions as follows

$$d(\mathcal{Z}_k^{\text{KF}}, \mathcal{Z}^{(l)}) = -\ln\Big(\int_{R^{N_b}} \sqrt{P(\mathcal{Z}_k^{\text{KF}}) P(\mathcal{Z}^{(l)})}\, dz\Big). \tag{4.3}$$

Based on the BD specified in Eq. (4.3), the following two estimation scenarios are considered:

S1. The first estimation scenario is applied by computing the BD between the measurement vector $\boldsymbol{z}_k$ and all the N_{FP} GMM distributions and associating the one with smallest distance, i.e., $\hat{\mathcal{Z}}_k = \arg\min_l \{d_b(\mathcal{Z}_k, \mathcal{Z}^{(l)})\}$.

S2. In the second estimation scenario, first K nearest neighbors to the observation $\boldsymbol{z}_k$ are detected from all the raw RSSI vectors available for training ($\{\boldsymbol{RSSI}_n^{(l)}\}_{n=1}^{N_{\text{Train}}}$ for $1 \leq l \leq N_{\text{FP}}$). This is done based on the corresponding BD between $\mathcal{Z}_k^{\text{KF}}$ and distribution of the associated zone $\mathcal{Z}^{(l)}$. Then, total weight of each zone ($W^{(l)}$) is computed with the sum of the weights of neighbors belonging to Zone l, and the weight of each neighbor is computed as follow $\frac{1}{d(\mathcal{Z}_k^{\text{KF}}, \mathcal{Z}^{(l)})}$. For location estimation, the zone with maximum weight is detected as the user's location, i.e., $\hat{\mathcal{Z}}_k = \arg\max_l \{W^{(l)}\}$.

Algorithm 2 outlines the steps of the proposed GMM-based BLE fingerprinting. Next, we present real experiments to evaluate performance of the proposed framework.

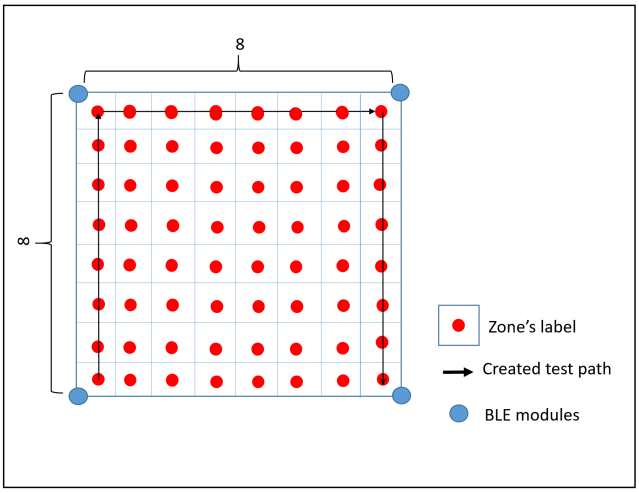


Figure 4.1: Data Collection Setup.

4.1.2 Experimental Results

Two proposed GMM-based localization algorithms are applied on a real dataset consisting of RSSI measurements collected in a $(4.0m \times 4.0m)$ room divided into 64 square zones with dimension $(0.5m \times 0.5m)$ as shown in Fig. 4.1. For fingerprinting, $1,000$ RSSI vectors were measured in each zone while 4 BLE sensors were located in the corners of the venue with 4m distance from each others. For evaluation of the proposed GMM-based approaches, a trajectory is created (as shown in Fig. 4.2) based on 22 selected test points covering the area for which test RSSI values are computed. We have $1,000$ offline measured RSSI vectors in each zone, which are modeled by GMMs with 2, 4 or 8 components. Then KF is applied on the test RSSI vectors and a GMM with 2 components is associated to each vector of test data. For comparison purposes, two scenarios are applied on the data based on the Euclidean distance between mean of a Gaussian distribution of the current test data and the mean of single Gaussian distributions (not GMMs) of all 64 zones. The best achieved zone classification accuracy for first scenario and second scenario ($K = 5$) are 14% and 77%, respectively. On the other hand, the BD between different combination of trained models (with different number of Gaussian components within the GMMs) and test data are computed for both scenarios. Table 4.1 and 4.2 show the accuracy of the two scenarios with different combination of (N_c). It is observed that combination of BD with GMM method based on two Scenarios provide a better accuracy $(45\%, 90\%)$ compared to the Euclidean distance method $(14\%$ and $77\%)$.

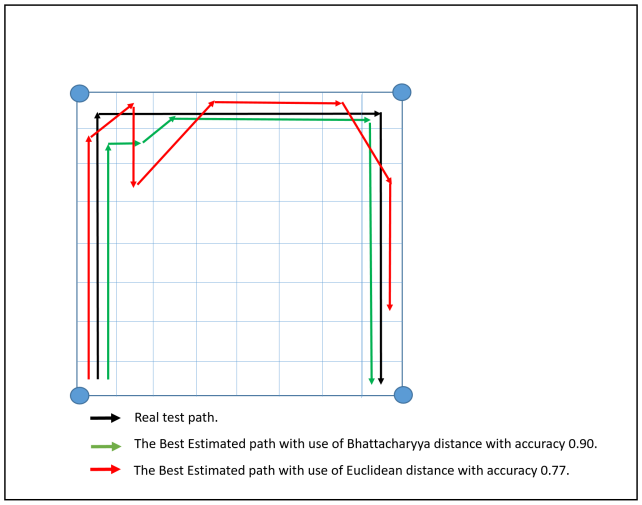


Figure 4.2: Estimated target's path.

Table 4.1: Accuracy of first scenario for various number of N_c.

Number of Components (N_c)	**GM**	**2-GMM**	**4-GMM**	**8-GMM**
GM	40%	**45**%	30%	14%
2-GMM	**36**%	30%	25%	9%

Table 4.2: Accuracy of the first scenario with $K = 5$ for various number of N_c.

Number of Components (N_c)	**GM**	**2-GMM**	**4-GMM**	**8-GMM**
GM	88%	**90**%	88%	43%
2-GMM	**70**%	**70**%	43%	20%

Based on these experiments, it is concluded that using 2 components within GMMs representing fingerprinting zones and a single Gaussian model provided by the KF (representing the probabilistic model of the real-time RSSI values) coupled with utilization of the BD and Scenario S2 provides the best results of 90%. This high accuracy becomes even more intriguing considering the fact that the considered zones are fairly close and relatively small number of RSSI values are collected per each zone.

4.2 Gaussian Sum Filtering Coupled with Wasserstein Distance for Non-Gaussian BLE-based Localization

This section proposes a novel GMM-based probabilistic framework for assigning real-time observed RSSI vectors to different fingerprinting zones using BD and Wasserstein distance (WD) [26]. More specifically, after measuring the RSSI values, the values at each zone is modelled with multivariate GM distribution. In the online phase, the proposed novel GSF algorithm is applied on the real-time RSSI values via incorporation of a WD-based clustering GMR for smoothing the RSSI values and simulating the non-Gaussian RSSI measurement noise as a GMMs distribution. These distributions outputs are then compared in distribution with GMs, learned for each zone, via utilization of the BD [26]. Two scenarios are used for user's location estimation: (i) The BD between the output of the GSF and the trained GM associated with all the zones are calculated and the one with the minimum distance is identified as the target's zone, and; (ii) A weighted K-NN algorithm is applied for location estimation, where the weights are computed based on the BD between the output of the GSF and the GMs distributions of K nearest zones to the identified one.

4.2.1 GMM-based Indoor Localization via BLE Sensors

During the offline phase, RSSI values are measured from the active BLE beacons across the surveillance venue including all the zones. In the online phase, upon collection of the RSSI vector at each measurement epoch, a similarity measure is used to compare the observed RSSI value with the ones associated with the fingerprinting zones learned during the offline phase, and select one as the predicted zone of the user. In this section, the probability distribution of RSSI values are used during the online and offline phase to find this similarity. In other words, the RSSI values measured at each zone during the offline phase is modeled by a multivariate GMs and the RSSI vector received in the online phase is smoothed and modelled as a multivariate GMs by using the GSF based on the WD reduction methodology.

We consider tracking a user in an environment monitored with N_b BLE-enabled sensors. For fingerprinting, the venue is divided into N_{FP} grid of squares, where zone l is denoted by $\mathcal{Z}_l$ for $(1 \leq l \leq N_{\text{FP}})$. The center of each zone is referred to as a "training point".

The proposed GM-based localization framework consists of an offline phase and an online phase. The former consists of the following two main steps:

Step 1. Collecting N_{Train} number of raw RSSI vectors from N_b number of active beacons across Zone l, for ($1 \leq l \leq N_{\text{FP}}$), and constructing the following zone-specific training RSSI matrix

$$\boldsymbol{R}_l = \begin{bmatrix} RSSI^{1}_{(1,l)}, & \ldots, & RSSI^{1}_{(N_{\text{Train}},l)} \\ \vdots & \vdots & \vdots \\ RSSI^{N_b}_{(1,l)}, & \ldots, & RSSI^{N_b}_{(N_{\text{Train}},l)} \end{bmatrix}, \tag{4.4}$$

associated with Zone l, for ($1 \leq l \leq N_{\text{FP}}$).

Step 2. Modelling the N_{Train} number of RSSI vectors in matrix $\boldsymbol{R}_l$ with GM components, denoted by $P(\mathcal{Z}_l)$, consisting of N_c Gaussian components parameterized by $\theta_l^{(i)} = \{\phi_l^{(i)}, \boldsymbol{\mu}_l^{(i)}, \boldsymbol{\sigma}_l^{(i)}\}$, i.e.,

$$P(\mathcal{Z}_l) \quad = \quad \sum_{i=1}^{N_c} \phi^{(i)} \, \mathcal{N}(\boldsymbol{\mu}_l^{(i)}, \boldsymbol{\sigma}_l^{(i)}), \tag{4.5}$$

where $\phi_l^{(i)}$ is the i^{th} component's weight, $\boldsymbol{\mu}_l^{(i)}$ is the mean vector with its associated covariance matrix denoted by $\boldsymbol{\sigma}_l^{(i)}$.

It is worth mentioning that different from existing fingerprinting techniques, in the offline phase of the proposed framework, we model the RSSI values associated with the l^{th}-zone, for ($1 \leq l \leq N_{\text{FP}}$), as a multivariate GMM ($P(\mathcal{Z}_l)$) defined by Eq. (4.5). After completion of the offline phase, these learned statistics are used for real-time indoor localization as is described in the following sub-section.

The Online Phase

In the real-time implementation of the proposed algorithm at measurement epoch $k > 1$, first RSSI vector $\boldsymbol{z}_k = \left[Z_k^{(1)} \ldots Z_k^{(N_b)}\right]^T$ is constructed from measurements obtained from the N_b active beacons. Then, the GSF algorithm is applied on the observation vector $\boldsymbol{z}_k$ for the following two reasons:

(i) *Pre-processing*, which consists of RSSI smoothing to compensate the fading effects on the measured RSSI values, and;

(ii) *Construction of the Real-time GMM*, where the observed and smoothed RSSI vector corresponding to the unknown location at iteration k is modeled with a GM.

The output of the GSF would then be utilized for pattern matching to find the user's zone via identifying the zone, which has the most similar distribution to that of the online measured RSSI vector. Finally, the following statistical assumptions are incorporated to develop the GSF [46]:

- It is assumed that initial distribution $P(\boldsymbol{x_0}|\boldsymbol{z_0})$ is equal to the prior density $P(\boldsymbol{x_0})$.
- It is assumed that the process noise $\boldsymbol{w}_k$ is an additive white Gaussian noise with known covariance matrices $\boldsymbol{Q} > 0$.
- The measurement noise is a non-Gaussian additive approximated by a mixture of m_c mixing components as

$$P(\boldsymbol{v}_k) \quad = \quad \sum_{i=1}^{m_c} \mu^{(i)} \mathcal{N}(\boldsymbol{v}^{(i)}, \boldsymbol{R}^{(i)}) \tag{4.6}$$

- It is assumed that measurement noise $\boldsymbol{v}_k$ is stable over all steps.

To formulate the WD-GSF algorithm, we define the state vector $\boldsymbol{x}_k = [\boldsymbol{y}_k, \Delta\boldsymbol{y}_k]^T$, which consists of RSSI value $\boldsymbol{y}_k$ and its associated rate of change denoted by $\Delta\boldsymbol{y}_k$. Term $\Delta\boldsymbol{y}_k$ is dependent on the environment and shows how the RSSI values are fluctuating. The higher the noise in the environment, the higher will be the fluctuation. We consider the following dynamic model to represent the evolution of the constructed state vector ($\boldsymbol{x}_k = [\boldsymbol{y}_k, \Delta\boldsymbol{y}_k]^T$) over time

$$\begin{bmatrix} \boldsymbol{y}_k \\ \Delta\boldsymbol{y}_k \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{F}_k} \begin{bmatrix} \boldsymbol{y}_{k-1} \\ \Delta\boldsymbol{y}_{k-1} \end{bmatrix} + \begin{bmatrix} \boldsymbol{w}_k^{y} \\ \boldsymbol{w}_k^{\Delta y} \end{bmatrix}, \tag{4.7}$$

and $\boldsymbol{w}_k = \mathcal{N}(0, \boldsymbol{Q})$, with its covariance matrix $\boldsymbol{Q}$ being learned during the offline phase, represents the forcing term of the model, which is a design parameter. For WD-GSF implementation, we

consider the following observation model

$$\boldsymbol{z}_k = \underbrace{[1 \;\; 0]}_{\boldsymbol{H}_k} \begin{bmatrix} \boldsymbol{y}_k \\ \Delta \boldsymbol{y}_k \end{bmatrix} + \boldsymbol{v}_k, \tag{4.8}$$

which $\boldsymbol{v}_k$ has the probability distribution in Eq. (4.6). The proposed WD-GSF approximates the likelihood function $p(\boldsymbol{z}_k|\boldsymbol{x}_k)$ with a GM as

$$P(\boldsymbol{z}_k|\boldsymbol{x}_k) \approx \sum_{i=1}^{m_c} w_k^{(i)} \mathcal{N}(\hat{\boldsymbol{z}}_{k|k-1}^{(i)}, \boldsymbol{S}_k^{(i)}), \tag{4.9}$$

where

$$\hat{\boldsymbol{z}}_{k|k-1}^{(i)} = \boldsymbol{H}_k \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{v}^{(i)}, \tag{4.10}$$

$$\boldsymbol{S}_k^{(i)} = \boldsymbol{H}_k \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T + \boldsymbol{R}_k^{(i)}, \tag{4.11}$$

and

$$w_k^{(i)} = \frac{\exp\{||\hat{\boldsymbol{z}}_{k|k-1}^{(i)} - \boldsymbol{z}_k||^2_{[\boldsymbol{S}_k^{(i)}]^{-1}}\}}{\pi^{N_x/2} |\boldsymbol{S}_k^{(i)}|^{1/2} \times \boldsymbol{\zeta}_k} \mu^{(i)}, \tag{4.12}$$

where

$$\boldsymbol{\zeta}_k = \sum_{i=1}^{m_c} \frac{\exp\{||\hat{\boldsymbol{z}}_{k|k-1}^{(i)} - \boldsymbol{z}_k||^2_{[\boldsymbol{S}_k^{(i)}]^{-1}}\}}{\pi^{N_x/2} |\boldsymbol{S}_k^{(i)}|^{1/2}} \mu^{(i)}. \tag{4.13}$$

Based on the Bayesian theory, Eq. (4.9) results in a GMM of the posterior distribution $p(\boldsymbol{x}_k|\boldsymbol{z}_k)$ as

$$P(\boldsymbol{x}_k|\boldsymbol{z}_k) \approx \sum_{i=1}^{m_c} w_k^{(i)} p(\boldsymbol{x}_k|\boldsymbol{z}_{k-1}) \mathcal{N}(\hat{\boldsymbol{z}}_{k|k-1}^{(i)}, \boldsymbol{S}_k^{(i)}). \tag{4.14}$$

Then, GMR reduction algorithm based on WD (Algorithm 3) is applied on Eq. (4.14) to collapse the resulting GM into one single Gaussian distribution $\mathcal{N}(\hat{\boldsymbol{x}}_{k|k}, \boldsymbol{P}_{k|k})$. In other words, WD-GSF is applied based on the following steps

Step 1. Prediction Step: Assuming that the posteriori distribution $P(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{k-1})$ is collapsed into one single Gaussian denoted by $\mathcal{N}(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{P}_{k-1|k-1})$ via WD, the prediction step of the WD-GSF for computing $\hat{\boldsymbol{x}}_{k|k-1}$ is given by

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_k \boldsymbol{P}_{k-1|k-1} \boldsymbol{F}_k^H + \boldsymbol{Q}_k \tag{4.15}$$

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}_k \hat{\boldsymbol{x}}_{k-1|k-1} \tag{4.16}$$

Step 2. Filtering and Collapsing: In this step, WD-GSF simultaneously filters the m_c components identified in Eq. (4.9) and update the overall point estimation $\hat{\boldsymbol{x}}_{k|k}$ and its error covariance $\boldsymbol{P}_{k|k}$ by collapsing the m_c filtered components into one single Gaussian via Wasserstein-Based Clustering GMR algorithm.

In other words, $P(\boldsymbol{z}_k|\boldsymbol{x}_k)$ is a GM with m_c components and $P^{(i)}(\boldsymbol{z}_k|\boldsymbol{x}_k) = \mathcal{N}(\hat{\boldsymbol{z}}^{(i)}_{k|k-1}, \boldsymbol{S}^{(i)}_k)$, for $(1 \leq i \leq m_c)$, represents the i^{th} Gaussian component of this GM model. The goal of WD-GSF is to approximate $P(\boldsymbol{z}_k|\boldsymbol{x}_k)$ with single Gaussian distribution $Q(\boldsymbol{z}_k|\boldsymbol{x}_k) = \mathcal{N}(\hat{\boldsymbol{z}}_{k|k-1}, \boldsymbol{S}_k)$. In contrary to the common approach of using the KLD for GMR purposes, we consider WD-based as the statistical measure. The WD is a symmetric and metric probabilistic similarity measure, which minimizes the cost of transferring a PDF to another one by considering the difference between the shapes of the underlying PDFs. The WD between the aforementioned two Gaussian components is measured as follows [44]

$$\begin{aligned} D_i = D_W^2(P^{(i)}\big(\boldsymbol{z}_k|\boldsymbol{x}_k\big), Q(\boldsymbol{z}_k|\boldsymbol{x}_k)\big) = \text{tr}\big\{\boldsymbol{S}^{(i)}_k + \boldsymbol{S}_k - 2\big((\boldsymbol{S}^{(i)}_k)^{1/2}\boldsymbol{S}_k(\boldsymbol{S}^{(i)}_k)^{1/2}\big)^{1/2}\big\} \\ + \quad ||\hat{\boldsymbol{z}}^{(i)}_{k|k-1} + \hat{\boldsymbol{z}}_{k|k-1}||_2^2. \end{aligned} \tag{4.17}$$

To perform GMR on $P(\boldsymbol{z}_k|\boldsymbol{x}_k)$ with m_c components and reducing it to $Q(\boldsymbol{z}_k|\boldsymbol{x}_k)$ with $1 < m_c$ components WD (Eq. (4.17)) is used by first constructing an initial candidate for $Q(\boldsymbol{z}_k|\boldsymbol{x}_k)$. Then the components of $P(\boldsymbol{z}_k|\boldsymbol{x}_k)$ are associated with the closet components of $Q(\boldsymbol{z}_k|\boldsymbol{x}_k)$ based on the WD between the two components. The group of $P(\boldsymbol{z}_k|\boldsymbol{x}_k)$ components that are assigned to the same component of $Q(\boldsymbol{z}_k|\boldsymbol{x}_k)$ is called a cluster. Finally, the associated component of $Q(\boldsymbol{z}_k|\boldsymbol{x}_k)$ is replaced with the center of its cluster and its weight becomes equal to sum of the cluster members'

Algorithm 3 THE PROPOSED WD-GSF

Input: $\boldsymbol{H}(\boldsymbol{x}), \boldsymbol{F}(\boldsymbol{x}), P(\boldsymbol{v}_k), \boldsymbol{z}_k, \boldsymbol{Q}_k, \hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{P}_{k-1|k-1}, \boldsymbol{R}_l (1 \leq l \leq N_{\text{FP}})$.

Output: $\hat{\boldsymbol{x}}_{k|k}, \boldsymbol{P}_{k|k}, \hat{\mathcal{Z}}_k$.

Prediction Step:

  $\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_k \boldsymbol{P}_{k-1|k-1} \boldsymbol{F}_k^H + \boldsymbol{Q}_k$

  $\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}_k \hat{\boldsymbol{x}}_{k-1|k-1}$

for $i = 1 : m_c$ **do**

  Compute $\hat{\boldsymbol{z}}_{k|k-1}^{(i)}, \boldsymbol{S}_k^{(i)}$ via Eq. (4.10) and Eq. (4.11) respectively.

  Compute $w_k^{(i)}$ via Eq. (4.12) and Eq. (4.13).

end for

$P(\mathcal{Z}_k) = P(\boldsymbol{z}_k|\boldsymbol{x}_k) \approx \sum_{i=1}^{m_c} w_k^{(i)} \mathcal{N}(\hat{\boldsymbol{z}}_{k|k-1}^{(i)}, \boldsymbol{S}_k^{(i)})$.

Collapse $P(\boldsymbol{z}_k|\boldsymbol{x}_k)$ into $Q(\boldsymbol{z}_k|\boldsymbol{x}_k) = \mathcal{N}(\hat{\boldsymbol{z}}_{k|k-1}, \boldsymbol{S}_k)$ with WD-based GMR.

Update Step:

  $\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k \boldsymbol{S}_k^{-1}$

  $\boldsymbol{P}_{k|k} = [\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k] \boldsymbol{P}_{k|k-1}$

  $\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k (\hat{\boldsymbol{z}}_k - \hat{\boldsymbol{z}}_{k|k-1})$

Pattern Matching:

  Construct $P(\mathcal{Z}_l) = \sum_{i=1}^{N_c} \phi^{(i)} \mathcal{N}(\boldsymbol{\mu}_l^{(i)}, \boldsymbol{\sigma}_l^{(i)})$ for $(1 \leq l \leq N_{\text{FP}})$.

  First Scenario (S1):

    $\hat{\mathcal{Z}}_k = \arg\min_l \{d_b(\mathcal{Z}_k, \mathcal{Z}_l)\}$.

  Second Scenario (S2):

    $\hat{\mathcal{Z}}_k$ is estimated with weighted K-NN algorithm.

weights.

After completion of the WD-based GMR component of the WD-GSF, the state vector is constructed as follows

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k \boldsymbol{S}_k^{-1} \tag{4.18}$$

$$\boldsymbol{P}_{k|k} = [\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k] \boldsymbol{P}_{k|k-1} \tag{4.19}$$

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k (\hat{\boldsymbol{z}}_k - \hat{\boldsymbol{z}}_{k|k-1}) \tag{4.20}$$

After smoothing the RSSI values during the online phase ($\hat{\boldsymbol{x}}_{k|k}$), for pattern matching, against the excising solution, now we propose to measure (via statistical distant measures) the distance between two probability distributions, i.e.,

$$P(\mathcal{Z}_l) = \sum_{i=1}^{N_c} \phi^{(i)} \mathcal{N}(\boldsymbol{\mu}_l^{(i)}, \boldsymbol{\sigma}_l^{(i)}),$$

$$\text{and} \quad P(\mathcal{Z}_k) = P(\boldsymbol{z}_k|\boldsymbol{x}_k) = \sum_{i=1}^{m_c} w_k^{(i)} \mathcal{N}(\hat{\boldsymbol{z}}_{k|k-1}^{(i)}, \boldsymbol{S}_k^{(i)}).$$

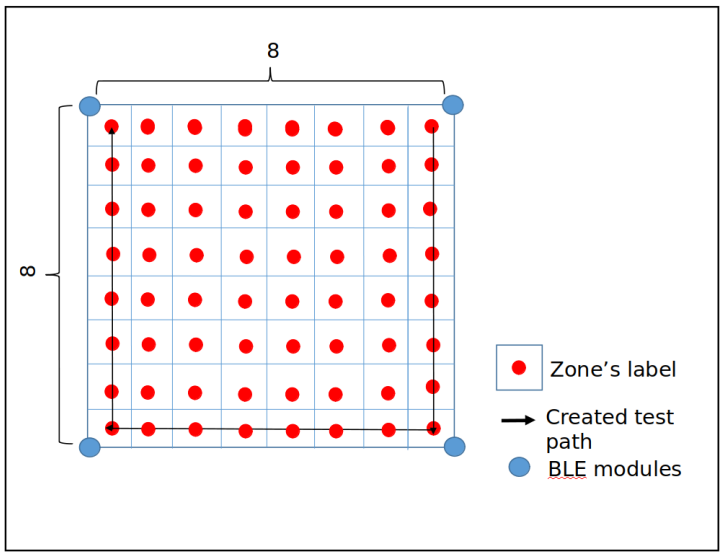


Figure 4.3: Data measurement setup.

We use BD as follows

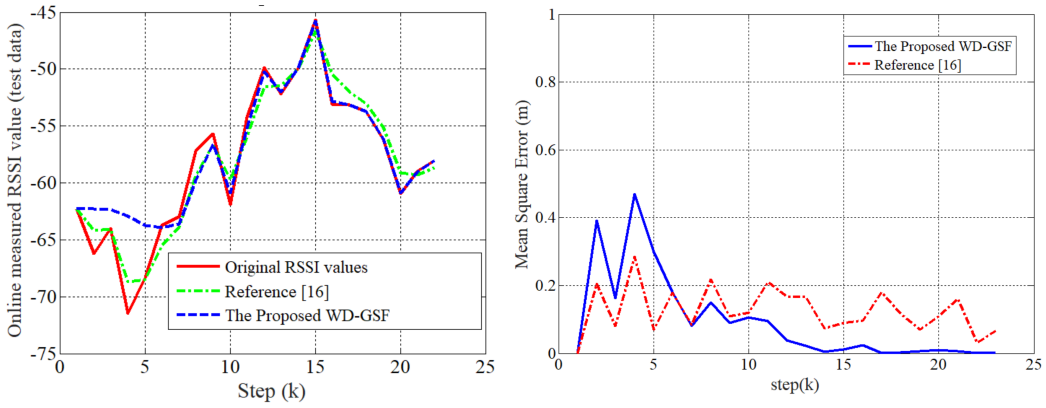
$$d_b(P(\mathcal{Z}_k), P(\mathcal{Z}_l)) = -\ln\Big(\int_{\mathbb{R}^{N_b}} \sqrt{P(\mathcal{Z}_k)P(\mathcal{Z}_l)}\, d\boldsymbol{z}\Big). \tag{4.21}$$

Based on the BD specified in Eq. (4.21), the WD-GSF can be implemented in terms of either of the following two variants:

S1. The first estimation scenario is applied by computing the BD between the measurement vector $\boldsymbol{z}_k$ and all the N_{FP} GM distributions and associating the one with smallest distance, i.e. $\hat{\mathcal{Z}}_k = \arg\min_l\{d_b(\mathcal{Z}_k, \mathcal{Z}_l)\}$.

S2. In the second estimation scenario, first K nearest neighbors to the observation $\boldsymbol{z}_k$, between all fingerprinting zones are detected based on their corresponding BD. Then, weighted average of the locations of the K selected neighbours, with weights computed as follows $\frac{1}{d(\boldsymbol{Z}_k, Z_l)}$, is considered as the user's location

This complete description of the proposed WD-GSF framework, which is summarized in Algorithm 3.



(a) (b)

Figure 4.4: (a) RSSI smoothing resulting from conventional KF and GSF. (b) MSE comparison between KF-based algorithm and WD-GSF algorithm.

4.2.2 Experimental Results

Two different variants of the proposed WD-GSF are applied and evaluated based on a real dataset consisting of RSSI measurements collected in a $(4.0m \times 4.0m)$ room divided into 64 square zones with dimension $(0.5m \times 0.5m)$ as shown in Fig. 4.3. For fingerprinting, $1,000$ RSSI vectors were measured at each zone while 4 BLE sensors were located in the corners of the venue with 4m distance from each others.

Similar to our previous work presented in Section 4.1, for evaluation of the proposed GMM-based approaches, a trajectory is created (as shown in Fig. 4.3) based on 22 selected test points covering the area for which test RSSI values are computed. We have $1,000$ offline measured RSSI vectors in each zone, which are modeled by one single Gaussian and GMMs with 2, and 4 components for comparison purposes. During the online phase, the WD-GSF is applied on the test RSSI vector 100runs to (i) Smooth the RSSI vector, and; (ii) Model the RSSI vector at each step with one single Gaussian and a GM with 2 components. Fig. 4.4(a) illustrates the smoothed RSSI resulting from WD-GSF and KF-based algorithm in Reference [23]. Two scenarios are applied on the data based on the BD between Gaussian distribution of the current test data (resulting from WD-GSF and KF model) and Gaussian distribution of all 64 zones. The BD between different combination of trained models (with different number of Gaussian components within the GMMs) and test data are computed for both scenarios. Table 4.3 and 4.4 show the accuracy of the two scenarios

Table 4.3: Accuracy comparison of first location estimation algorithms for various number of GMM components.

# of Components (N_c)	**GM**	**2-GMM**	**4-GMM**	**8-GMM**
GM	0.40	0.45	0.25	0.25
2-GMM	0.36	**0.62**	0.45	0.30

Table 4.4: Accuracy comparison of second location estimation algorithms with $K = 5$ for various number of GMM components.

# of Components (N_c)	**GM**	**2-GMM**	**4-GMM**	**8-GMM**
GM	0.88	0.90	0.88	0.74
2-GMM	0.74	**0.95**	0.80	0.62

with different combination of N_c and m_c getting from the WD-GSF algorithm. The best achieved zone classification accuracy for first scenario and second scenario ($K = 5$) are 62% and 95%, respectively. Based on these experiments, it is concluded that using 2 components within GMMs representing fingerprinting zones and 2 components of GMM provided by the WD-GSF (representing the probabilistic model of the real-time RSSI values) coupled with utilization of the BD and Scenario S2 provides the better result (95%) compared to modelling the real-time RSSI values with KF-based localization algorithm described in Section 4.1 (45% and 90%). Fig. 4.4(b) compares the MSE in estimating the location resulting of two mentioned algorithm based on 100 runs.

4.3 Conclusion

To deal with the fact that RSSI-based solutions are prone to multipath fading and drastic fluctuations in the indoor environment, this chapter proposed to jointly use Bayesian estimation approaches coupled with distribution matching algorithms. Instead of using conventional solutions, distribution matching is performed based on the BD and WD of the distribution of online and offline RSSI values. The experimental results show that performances of the proposed BD-GMM and WD-GSF frameworks are superior in terms of accuracy when compared to algorithms developed based on a single Gaussian model, and Euclidean distance-based matching algorithm.

Chapter 5

MM-KTD: Multiple Model Kalman Temporal Differences for Reinforcement Learning

The previous chapters focused on development of localization algorithms that can be used by an autonomous agent (e.g., a rescue robot) to navigate through indoor environments. In this chapter, the focus is on the decision process of that autonomous agent to achieve its pre-defined objective (e.g., reaching a person in need in that rescue mission) by performing sequential actions in an optimal or near-optimal fashion. In this regard, RL is an attractive domain providing autonomous agents with a set of tools to perform sophisticated and required behaviors. Conversely, the existing challenges in autonomous agents provide both inspiration, impact, and validation for developments in the RL.

As described previously in Chapter 2, in model-free RL methods with continuous state-space, typically, the value function of the states need to be approximated. In this regard, Deep Neural Networks (DNNs) algorithms provide an attractive modeling mechanism to approximate the value function using sample transitions. DNN-based solutions, however, suffer from high sensitivity to parameter selection, are prone to overfitting, and are not very sample efficient. Kalman-based methodologies, on the other hand, could be used as an efficient alternative. Such an approach, however, commonly requires a-priori information about the system (such as noise statistics) to perform

efficiently. The main objective of this chapter is to address this issue. As a remedy to the aforementioned problems, this chapter proposes an innovative Multiple Model Kalman Temporal Difference (MM-KTD) framework, which adapts the parameters of the filter using the observed states and rewards. Moreover, an active learning method is proposed to enhance the sampling efficiency of the system. More specifically, the estimated uncertainty of the value functions are exploited to form the behaviour policy leading to more visits to less certain values, therefore, improving the overall learning sample efficiency. As a result, the proposed MM-KTD framework can learn the optimal policy with significantly reduced number of samples as compared to its DNN-based counterparts. To evaluate performance of the proposed MM-KTD framework, a comprehensive set of experiments has been performed based on three RL benchmarks, namely, Inverted Pendulum; Mountain Car, and; Lunar Lander. Experimental results show superiority of the proposed MM-KTD framework in comparison to its state-of-the-art counterparts.

The remaining of this chapter is organized as follows: In Section 5.1, the basic techniques of RL is briefly outlined. Section 5.2 presents the proposed MM-KTD framework. Experimental results based on three RL benchmarks are presented in Section 5.3 illustrating effectiveness and superiority of the proposed MM-KTD framework. Finally, Section 5.4 concludes the chapter.

5.1 Reinforcement Learning (RL): Formulation

In a typical RL scenario, one deals with an agent being placed in an unknown environment. At each time, the agent is considered to be in a specific state within the available set of states denoted by $\mathcal{S}$, and can take an action from the action set $\mathcal{A}$ using specific policy that takes the agent to another state. More specifically, at time step k, the agent at state $\boldsymbol{s}_k \in \mathcal{S}$ takes an action $a_k \in \mathcal{A}$ exploiting the policy π_k, which takes the system to the state $\boldsymbol{s}_{k+1} \in \mathcal{S}$ with the transition probability of $P(\boldsymbol{s}_k, a_k, \boldsymbol{s}_{k+1}) \in \mathcal{P}_a$ resulting in a reward $r_k \in \mathcal{R}$. The 5-tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}_a, \mathcal{R}, \gamma\}$, for $(0 \leq \gamma \leq 1)$ denoting the discount factor, defines the process known as the Markov Decision Process (MDP), which defines the domain of work in RL. The agent starts at an initial state denoted by $\boldsymbol{s}_0$ and continues to explore the states until it reaches a terminal state denoted by $\boldsymbol{s}_T$, where an episode is completed.

Generally speaking, the RL goal is to find a policy through a number of experimental episodes that maximizes the expected sum of discounted rewards over the future states. In achieving so, it is useful to define the state value function, $V_\pi(\boldsymbol{s})$ as

$$V_\pi(\boldsymbol{s}) = \mathbb{E}\left\{\sum_{k=0}^{T} \gamma^k r_k | \boldsymbol{s}_0 = \boldsymbol{s}, a_k = \pi(\boldsymbol{s}_k)\right\}, \tag{5.1}$$

where $\mathbb{E}\{\cdot\}$ represents the expectation function. If the value function (and also the transition probability) is known for a policy, the policy may be improved by selecting a greedy action at each step leading to the next state, which has the highest value (policy improvement) [83]. If the transition probability is unknown, it is more useful to exploit the state-action value function defined as follows

$$Q_\pi(\boldsymbol{s}, a) = \mathbb{E}\left\{\sum_{k=0}^{T} \gamma^k r_k | \boldsymbol{s}_0 = \boldsymbol{s}, a_0 = a, a_k = \pi(\boldsymbol{s}_k)\right\}. \tag{5.2}$$

Using the state-action value function ($Q_\pi(\boldsymbol{s}, a)$) defined in Eq. (5.2) has the advantage of direct selection of the greedy action as compared to the state value function ($V_\pi(\boldsymbol{s})$) defined in Eq. (5.1), where the leading states have to be identified.

Following the Bellman update idea [84], sample transitions may be exploited to gradually update the value functions. In other words, at each transition (from one state to the next by taking an action), a one-step update may be performed, which is best known as a Temporal Difference (TD) update [84]. If the current policy is used to select actions for such an update, the procedure is called "on-policy learning". For instance, in SARSA method [85, 86], which is an on-policy learning method, the state-action value function is updated as follows

$$Q_\pi(\boldsymbol{s}_k, a_k) = Q_\pi(\boldsymbol{s}_k, a_k) + \alpha\Big(r_k + \gamma\, Q_\pi(\boldsymbol{s}_{k+1}, a_{k+1}) - Q_\pi(\boldsymbol{s}_k, a_k)\Big), \tag{5.3}$$

where α is the learning rate. On-policy samples are usually not very sample efficient, as the value function is learned for the current policy and not based on the optimal one. Besides, exploring new states is challenging in the on-policy methods as they follow a particular policy. On the other hand, the "off-policy learning" methods, also referred to as behavior policies, allow for updating the optimal policy using the information gained from other policies. In most cases, a stochastic (e.g.,

random) policy is selected as the behaviour policy to ensure enough exploration of new states. One of the most practiced off-policy methods is known as Q-learning [86–89], which updates the value function using the Bellman optimality equation as follows

$$Q_{\pi^*}(\boldsymbol{s}_k, a_k) = Q_{\pi^*}(\boldsymbol{s}_k, a_k) + \alpha\Big(r_k + \gamma \max_{a \in \mathcal{A}} Q_{\pi^*}(\boldsymbol{s}_{k+1}, a) - Q_{\pi^*}(\boldsymbol{s}_k, a_k)\Big), \tag{5.4}$$

where π^* represents the optimal policy. For the greedy policy, the state value function is related to the state action value function as follows

$$V_{\pi^*}(\boldsymbol{s}) = \max_{a \in \mathcal{A}} Q_{\pi^*}(\boldsymbol{s}_k, a). \tag{5.5}$$

It is noteworthy to mention that action a_k in Eq. (5.4) is selected based on the behaviour policy. Once the system has converged, the optimal policy may be used as follows

$$a_k = \arg\max_{a \in \mathcal{A}} Q_{\pi^*}(\boldsymbol{s}_k, a). \tag{5.6}$$

When the number of states are finite and small, it is relatively easy to update the value function by visiting every state of the system (e.g., using Q-Learning). When the states are continuous, however, it is not viable to visit all the states, therefore, requiring to approximate the value function. Deep learning techniques provide powerful supervised learning solutions for such purposes by approximating highly nonlinear functions using labelled data. However, neural networks (used in deep learning) are notorious for problems such as over-fitting and brittleness to parameter tuning, therefore, should be used with extra care.

Alternatively, the value function may be approximated using basis functions. In this approach, the value function is estimated with a weighted sum of local basis functions, each of which is active in a local region of the state-space. The value function is then formed as follows

$$Q_{\pi}(\boldsymbol{s}_k, a_k) = \boldsymbol{\phi}(\boldsymbol{s}_k, a_k)^T \boldsymbol{\theta}_k, \tag{5.7}$$

where $\boldsymbol{\phi}(\boldsymbol{s}, a)$ is a vector of basis functions (will be described later in Section 5.2.3) and $\boldsymbol{\theta}$ is a

weight vector. Various basis function may be used for such approximations, among which RBFs are proven [64] to be one of the most suitable options and are therefore selected in this work for value function approximation. This completes our background discussion on RL. Next, we present the proposed MM-KTD framework.

5.2 MM-KTD: Multiple Model Kalman Temporal Difference

For RL tasks with continuous state-space, the sample transitions of the system and the gained rewards are used as the data for the purpose of value function approximation and to estimate the weights. To avoid overfitting problems, supervised learning methods such as deep learning require this data to be stored and then used in batches (batch learning) for training of a system (e.g., neural networks). Generally speaking, neural networks have considerably high memory requirements to store the input data, weight parameters, and activations as an input propagates through the network. In training, activation from a forward pass must be retained until they can be used to calculate the error gradients in the backwards pass. As an example, the 50-layer ResNet network has 26 million weight parameters and computes 16 million activations in the forward pass. Measuring, roughly, the memory requirement of the ResNet-50 training stage with a mini-batch of 32 requires a, typically, high performance GPU and over 7.5 GB of local DRAM. In the contrary, sequential data processing methods such as multiple-model filters [22, 26, 43, 90] can adapt the system with the last measurement (assuming a one-step Markov property), without the need to store the whole measurements for learning, which results in much less memory requirement. Using a Kalman-based approach, the posteriori of the weights $\boldsymbol{\theta}_k$ can be calculated recursively using the Bayes rule as follows

$$P(\boldsymbol{\theta}_k|\boldsymbol{Y}_k) = \frac{P(\boldsymbol{y}_k|\boldsymbol{\theta}_k, \boldsymbol{Y}_{k-1})P(\boldsymbol{\theta}_k|\boldsymbol{Y}_{k-1})}{P(\boldsymbol{y}_k|\boldsymbol{Y}_{k-1})}, \tag{5.8}$$

where $\boldsymbol{y}_k$ is the measurements vector of the system (i.e, transition information) at time step k and $\boldsymbol{Y}_k$ is the set of all measurements from time Step 1 to time Step k. Utilizing the probabilistic model in Eq. (5.8), this chapter proposes a Kalman-based off-policy learning scheme, which is detailed below.

5.2.1 Kalman Temporal Difference Method

The optimal value function may be approximated from its one-step approximation using the TD method shown in Eq. (5.4), i.e.,

$$Q_{\pi^*}(\boldsymbol{s}_k, a_k) \approx r_k + \gamma \max_{a \in \mathcal{A}} Q_{\pi^*}(\boldsymbol{s}_{k+1}, a). \tag{5.9}$$

With a change in the order of the variables, the reward at time Step k may be considered as a noisy measurement from the system as follows

$$r_k = Q_{\pi^*}(\boldsymbol{s}_k, a_k) - \gamma \max_{a \in \mathcal{A}} Q_{\pi^*}(\boldsymbol{s}_{k+1}, a) + v_k, \tag{5.10}$$

where v_k is assumed to be a zero-mean Gaussian noise with variance of R. In this chapter, the value function is approximated as discussed in Eq. (5.7), rendering Eq. (5.10) to have the following form

$$\begin{aligned} r_k &= \boldsymbol{\phi}(\boldsymbol{s}_k, a_k)^T \boldsymbol{\theta}_k - \gamma \max_{a \in \mathcal{A}} \boldsymbol{\phi}(\boldsymbol{s}_{k+1}, a)^T \boldsymbol{\theta}_k + v_k \\ &= \left[\boldsymbol{\phi}(\boldsymbol{s}_k, a_k)^T - \gamma \max_{a \in \mathcal{A}} \boldsymbol{\phi}(\boldsymbol{s}_{k+1}, a)^T \right] \boldsymbol{\theta}_k + v_k. \end{aligned} \tag{5.11}$$

Considering,

$$\boldsymbol{h}_k = \boldsymbol{\phi}(\boldsymbol{s}_k, a_k) - \gamma \max_{a \in \mathcal{A}} \boldsymbol{\phi}(\boldsymbol{s}_{k+1}, a), \tag{5.12}$$

where $\max_{a \in \mathcal{A}} \boldsymbol{\phi}(\boldsymbol{s}_{k+1}, a_k)$ is found from $\max_{a \in \mathcal{A}} \boldsymbol{\phi}(\boldsymbol{s}_{k+1}, a_k)^T \boldsymbol{\theta}_k$, Eq. (5.11) defines the measurement of the system (the reward) as a linear function of the weight function (i.e., $\boldsymbol{\theta}$), which is to be estimated. Assuming the weight vector to be modelled by a linear dynamic system, i.e.,

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{F} \boldsymbol{\theta}_k + \boldsymbol{w}_k, \tag{5.13}$$

where $\boldsymbol{F}$ is the transition matrix and $\boldsymbol{w}_k$ is a zero-mean Gaussian noise with the covariance of $\boldsymbol{Q}$, the Kalman filtering formulation may be employed to estimate the weights in a MMSE sense. To

be more precise, the weights and their error covariance are first initialized, i.e.,

$$\hat{\boldsymbol{\theta}}_0 = \boldsymbol{\theta}(0) \tag{5.14}$$

$$\text{and } \boldsymbol{P}_{\boldsymbol{\theta},0} = \boldsymbol{P}_{\boldsymbol{\theta}}(0). \tag{5.15}$$

Then at each time step, first the weights and their error covariance are predicted as

$$\hat{\boldsymbol{\theta}}_{k|k-1} = \boldsymbol{F}\boldsymbol{\theta}_k \tag{5.16}$$

$$\boldsymbol{P}_{\boldsymbol{\theta},k|k-1} = \boldsymbol{F}\boldsymbol{P}_{\boldsymbol{\theta},k}\boldsymbol{F}^T, \tag{5.17}$$

and then the estimations are updated using the observed reward from the transition from State $\boldsymbol{s}_k$ to the next state ($\boldsymbol{s}_{k+1}$) as follows

$$\boldsymbol{K}_k = \boldsymbol{P}_{\boldsymbol{\theta},k|k-1}\boldsymbol{h}_k\big(\boldsymbol{h}_k^T\boldsymbol{P}_{\boldsymbol{\theta},k|k-1}\boldsymbol{h}_k + R\big)^{-1} \tag{5.18}$$

$$\hat{\boldsymbol{\theta}}_k = \hat{\boldsymbol{\theta}}_{k|k-1} + \boldsymbol{K}_k\big(r_k - \boldsymbol{h}_k^T\hat{\boldsymbol{\theta}}_{k|k-1}\big) \tag{5.19}$$

$$\boldsymbol{P}_{\boldsymbol{\theta},k} = \big(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{h}_k^T\big)\boldsymbol{P}_{\boldsymbol{\theta},k|k-1}\big(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{h}_k^T\big)^T + \boldsymbol{K}_k R\boldsymbol{K}_k^T. \tag{5.20}$$

The value function for each set of state and action is easily reconstructed through Eq. (5.7), and the optimal policy would be to select the action with the highest state-action value function at each state similar to Eq. (5.6).

When the parameters of the filter and system initial values (i.e., $\boldsymbol{\theta}_0$, $\boldsymbol{P}_{\boldsymbol{\theta},0}$, $\boldsymbol{F}$, $\boldsymbol{h}_k$, $\boldsymbol{Q}$, and R) are known a priori, the system will provide accurate estimations. However, these values are usually not available and need to be approximated using the measurements obtained from the environment. Among these, the measurement mapping function ($\boldsymbol{h}_k$) and the measurement noise variance (R) are of most importance since they regulate the flow of information from new measurements. The adaptation of these parameters is the topic of the following two subsections. Other filter parameters may be selected as design parameters.

As a final note, we would like to point out that the basic component of the constructed state-space model is the measurement equation (Eq. (5.11)), which relates the reward to the basis functions. In this chapter, following the formulation in Eq. (5.7) we are dealing with a linear combination of the basis functions with unknown (to be estimated) weight vectors $\boldsymbol{\theta}_k$. This formulation results in a linear measurement model of Eq. (5.11). An interesting direction for future work is to consider nonlinear measurement model [72] for estimating the value function. With regards to the dynamics of the weight vectors, at one hand, it is a common practice [22] to use the linear model of Eq. (5.13) when an auxiliary and unknown dynamic is introduced for the variable of interest to be estimated. Intuitively speaking, the dynamical model of Eq. (5.13) is introduced to make it possible to use state-space based solutions such as Kalman-based filters or Particle filters. As its true nature is unknown a-priori, the intuition is to consider it to be constant (hence having an identity type sate model $\boldsymbol{F}$) with changes being controlled by the covariance of the state model noise ($\boldsymbol{w}_k$). An interesting direction for further investigations is to learn dynamics of the introduced state model as well, e.g., using a separate neural-based module, which is the focus of our ongoing research.

5.2.2 Multiple Model Adaptive Estimation

Kalman filter is a powerful tool for accurate estimation of hidden variables when the estimation model is fully known. However, usually full knowledge about the filter parameters and initial values is not available in practical scenarios, which lead to deterioration of the system performance. Adaptive estimation is a one powerful way to remedy such a problem. In this work, a multiple model adaptation scheme [22, 26, 43, 90] is adopted due to simplicity and effectiveness of multiple-model solutions. In such schemes, multiple candidates are considered for each of the uncertain parameters and values and the estimations made based on each set of candidates is weighted exploiting the measurement likelihood function. The number of candidates increases exponentially with the number of uncertain parameters (curse of dimensionality), therefore, it is desirable to limit the adaptation to some of the most important parameters. In a Kalman-based estimation framework, the measurement noise variance (R) is one of the most important parameters and is, therefore, considered for the adaptation in this section using the multiple model technique. After observing $\boldsymbol{s}_{k+1}$ and r_k by taking action a_k, the measurement model ($\boldsymbol{h}_k$) is calculated. Then, different values ($R^{(i)}$) for

the measurement noise variance is considered in the proposed MM-KTD scheme and a bank of mode-matched Kalman filters are implemented for adaptation of the observation noise covariance. Eqs. (5.18)-(5.20) are, therefore, replaced with the following

$$\begin{aligned}
\boldsymbol{K}_k^{(i)} &= \boldsymbol{P}_{\boldsymbol{\theta},k|k-1}\boldsymbol{h}_k\big(\boldsymbol{h}_k^T\boldsymbol{P}_{\boldsymbol{\theta},k|k-1}\boldsymbol{h}_k + R^{(i)}\big)^{-1} && (5.21)\\
\hat{\boldsymbol{\theta}}_k^{(i)} &= \hat{\boldsymbol{\theta}}_{k|k-1} + \boldsymbol{K}_k^{(i)}\big(r_k - \boldsymbol{h}_k^T\hat{\boldsymbol{\theta}}_{k|k-1}\big) && (5.22)\\
\boldsymbol{P}_{\boldsymbol{\theta},k}^{(i)} &= \big(\boldsymbol{I} - \boldsymbol{K}_k^{(i)}\boldsymbol{h}_k^T\big)\boldsymbol{P}_{\boldsymbol{\theta},k|k-1}^T\big(\boldsymbol{I} - \boldsymbol{K}_k^{(i)}\boldsymbol{h}_k^T\big) + \boldsymbol{K}_k^{(i)}R^{(i)}\boldsymbol{K}_k^{(i)^T}, && (5.23)
\end{aligned}$$

where superscription i denotes the Kalman filter value for the i^{th} filter which exploits $R^{(i)}$ as its measurement noise covariance. The posteriori of each mode-matched filter is weighted based on its associated and normalized weight, which are then added together to form the system's posteriori, i.e.,

$$P(\boldsymbol{\theta}_k|\boldsymbol{Y}_k) = \sum_{i=1}^{M}\omega^{(i)}P(\boldsymbol{\theta}_k|\boldsymbol{Y}_k, R^{(i)}), \tag{5.24}$$

where M is number of mode-matched filter within the MM-KTD framework and,

$$\omega^{(i)} = P(r_k|\boldsymbol{\theta}_{k|k-1}, R^{(i)}) = c.e^{\frac{-1}{2}\left(r_k-\boldsymbol{h}_k^T\hat{\boldsymbol{\theta}}_{k|k-1}\right)^T\left(\boldsymbol{h}_k^T\boldsymbol{P}_{\boldsymbol{\theta},k|k-1}\boldsymbol{h}_k+R^{(i)}\right)^{-1}\left(r_k-\boldsymbol{h}_k^T\hat{\boldsymbol{\theta}}_{k|k-1}\right)}, \tag{5.25}$$

where $w^{(i)}$ is the normalized measurement likelihood for the i^{th} filter and,

$$c = \frac{1}{\sum_{i=1}^{M} w^{(i)}}, \tag{5.26}$$

Using Eq. (5.24), the weight and its error covariance are then updated as follows

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_k &= \sum_{i=1}^{M}\omega^{(i)}\hat{\boldsymbol{\theta}}_k^{(i)} && (5.27)\\
\boldsymbol{P}_{\boldsymbol{\theta},k} &= \sum_{i=1}^{M}\omega^{(i)}\left(\boldsymbol{P}_{\boldsymbol{\theta},k}^{(i)} + (\hat{\boldsymbol{\theta}}^{(i)} - \hat{\boldsymbol{\theta}})(\hat{\boldsymbol{\theta}}^{(i)} - \hat{\boldsymbol{\theta}})^T\right) && (5.28)
\end{aligned}$$

As a final note, it is interesting to highlight the connections between the above multiple model frame work of the proposed MM-KTD with other methods, in particular gating approach of Mixture of

Experts (MoE) [91]. In a general setting, the goal of the weight update step in a multiple-model framework is to find the conditional probability density function (PDF) corresponding to mode i, denoted by m^i for $(1 \leq i \leq M)$, given all the observations up to the current time, i.e.,

$$p\big(m_k^{(i)}|\boldsymbol{Y}_k\big) = \frac{p\big(\boldsymbol{y}_k|\boldsymbol{Y}_{k-1}, m_k^{(i)}\big)p\big(m_k^{(i)}|\boldsymbol{Y}_{k-1}\big)}{\sum_{j=1}^{M} p\big(\boldsymbol{y}_k|\boldsymbol{Y}_{k-1}, m_k^{(j)}\big)p\big(m_k^{(j)}|\boldsymbol{Y}_{k-1}\big)}, \tag{5.29}$$

where the denominator is a normalizing factor to ensure that $p(m_k^{(l)}|\boldsymbol{Y}_k)$ is a proper PDF. Term $\mathcal{L}_k^{(i)} \triangleq p(\boldsymbol{y}_k|\boldsymbol{Y}_{k-1}, m_k^{(i)})$ in the nominator of Eq. (5.29) is the likelihood function of mode i, which is proportional to the exponential term in Eq. (5.25). Therefore, the corresponding weight for the filter matched to mode i, for $(1 \leq i \leq M)$, can be simplified as

$$\omega_k^{(i)} \triangleq p(m_k^{(i)}|\boldsymbol{Y}_k) = \frac{\omega_{k-1}^{(i)}\mathcal{L}_k^{(i)}}{\sum_{j=1}^{M} \omega_{k-1}^{(j)}\mathcal{L}_k^{(j)}}. \tag{5.30}$$

On the contrary, the MoE, typically, uses the soft max adaptation for Gaussian gating to obtain the associated weight, denoted here by $\omega_k^{(i,MoE)}$, for each model (expert). The weight is obtained by multiplying the input (in our case, the estimated state vector $\hat{\boldsymbol{\theta}}_k^{(i)}$ by model i, for $(1 \leq i \leq M)$) by a trainable weight matrix $\boldsymbol{W}_g$ and then applying the Softmax function as follows

$$\omega_k^{(i,MoE)} = \frac{\exp\left(\hat{\boldsymbol{\theta}}_k^{(i)}.\boldsymbol{W}_g\right)}{\sum_{j=1}^{M} \exp\left(\hat{\boldsymbol{\theta}}_k^{(j)}.\boldsymbol{W}_g\right)}. \tag{5.31}$$

Comparing Eq. (5.31) with Eq. (5.30) reveals the potentials of the multiple model approach in comparison to the MoE. This completes our discussion on multiple model adaptive estimation. Next, we discuss the update process for computation of the basis functions.

5.2.3 Basis Function Update

Vector $\boldsymbol{h}_k$, defined in Eq. (5.12), is the measurement mapping function and is required to be computed for accurate estimations within the context of Kalman-based filtering schemes. Such a prior knowledge, however, is typically not available, therefore $\boldsymbol{h}_k$ should be adapted to its correct value. Since $\boldsymbol{h}_k$ is formed by the basis functions, its adaptation necessitates the adaptation of the

basis functions. The vector of basis functions shown in Eq. (5.7) is formed as follows,

$$\boldsymbol{\phi}(\boldsymbol{s}_k, a_k) = [\phi_{1,a_1}, \dots \phi_{N,a_1}, \phi_{1,a_2}, \dots \phi_{N,a_D}]^T, \tag{5.32}$$

where N is the number of basis functions per action and D is the total number of actions (the arguments of the basis functions are omitted for brevity). Each basis function $\phi_{n,a_d}(\boldsymbol{s}_k, a_k)$ is selected as a RBF, is defined based on its mean vector $\boldsymbol{\mu}_{n,a_d}$ and covariance matrix $\boldsymbol{\Sigma}_{n,a_d}$ as follows

$$\boldsymbol{\phi}_{n,a_d}(\boldsymbol{s}_k, a_k) = e^{\frac{-1}{2}(\boldsymbol{s}_k - \boldsymbol{\mu}_{n,a_d})^T \boldsymbol{\Sigma}_{n,a_d}^{-1} (\boldsymbol{s}_k - \boldsymbol{\mu}_{n,a_d})}, \tag{5.33}$$

where $\boldsymbol{\mu}_{n,a_d}$ and $\boldsymbol{\Sigma}_{n,a_d}$ are the mean and covariance of this radial basis function.Due to the large number of parameters associated with the measurement mapping function (i.e., $2 \times N \times D$), it is reasonable to adapt these parameters through a gradient descent-based adaptation scheme rather than the multiple model method. For that purpose, the Restricted Gradient Descent (RGD) method proposed in [67] is adopted in this work. Using RGD, first the gradient of the object function with respect to the parameters of each basis function is calculated using partial derivations. The goal is to minimize the objective function, which is defined as the difference between the estimated value function and its one-step (TD) update, i.e.,

$$\boldsymbol{S}_k = \left(Q_{\pi^*}(\boldsymbol{s}_k, a_k) - r_k - \gamma \max_{a \in \mathcal{A}} Q_{\pi^*}(\boldsymbol{s}_{k+1}, a) \right)^2. \tag{5.34}$$

The gradient of the object function with respect to the parameters of the RBFs is calculated using the chain rule,

$$\Delta\boldsymbol{\mu} = -\frac{\partial \boldsymbol{S}_k}{\partial \boldsymbol{\mu}} = -\frac{\partial \boldsymbol{S}_k}{\partial Q_{\pi^*}} \frac{\partial Q_{\pi^*}}{\partial \boldsymbol{\phi}} \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{\mu}} \tag{5.35}$$

$$\text{and } \Delta\boldsymbol{\Sigma} = -\frac{\partial \boldsymbol{S}_k}{\partial \boldsymbol{\Sigma}} = -\frac{\partial \boldsymbol{S}_k}{\partial Q_{\pi^*}} \frac{\partial Q_{\pi^*}}{\partial \boldsymbol{\phi}} \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{\Sigma}}, \tag{5.36}$$

where the partial derivations are calculated using Eqs. (5.7), (5.33), and (5.34) as follows

$$\frac{\partial \boldsymbol{S}_k}{\partial Q_{\pi^*}} = 2\boldsymbol{S}^{\frac{1}{2}} \tag{5.37}$$

$$\frac{\partial Q_{\pi^*}}{\partial \boldsymbol{\phi}} = \boldsymbol{\theta}_k^T \tag{5.38}$$

$$\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{\mu}} = \boldsymbol{\phi}\boldsymbol{\Sigma}^{-1}(\boldsymbol{s}_k - \boldsymbol{\mu}) \tag{5.39}$$

$$\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{\Sigma}} = \boldsymbol{\phi}\boldsymbol{\Sigma}^{-1}(\boldsymbol{s}_k - \boldsymbol{\mu})(\boldsymbol{s}_k - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}. \tag{5.40}$$

The mean and covariance of the RBFs are then adapted using the calculated partial derivative as follows

$$\boldsymbol{\mu}_{n,a_d} = \boldsymbol{\mu}_{n,a_d} + \lambda_{\boldsymbol{\mu}}\Delta\boldsymbol{\mu} = \boldsymbol{\mu}_{n,a_d} - 2\lambda_{\boldsymbol{\mu}}\boldsymbol{S}^{\frac{1}{2}}Q_{\pi^*}\boldsymbol{\Sigma}^{-1}(\boldsymbol{s}_k - \boldsymbol{\mu}_{n,a_d}), \tag{5.41}$$

$$\boldsymbol{\Sigma}_{n,a_d} = \boldsymbol{\Sigma}_{n,a_d} + \lambda_{\boldsymbol{\Sigma}}\Delta\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{n,a_d} - 2\lambda_{\boldsymbol{\Sigma}}\boldsymbol{S}^{\frac{1}{2}}Q_{\pi^*}\boldsymbol{\Sigma}_{n,a_d}^{-1}(\boldsymbol{s}_k - \boldsymbol{\mu})(\boldsymbol{s}_k - \boldsymbol{\mu})^T\boldsymbol{\Sigma}_{n,a_d}^{-1}, \tag{5.42}$$

where $\lambda_{\boldsymbol{\mu}}$ and $\lambda_{\boldsymbol{\Sigma}}$ are the adaptation rates. To make the system more stable, only one of the updates shown in Eqs. (5.41) and (5.42) will be performed as discussed in [67]. To be more precise, when the size of the covariance is decreasing (i.e., $\boldsymbol{S}^{\frac{1}{2}}Q_{\pi^*} > 0$), the covariances of the RBFs are updated using Eq. (5.42), otherwise, their means are updated using Eqs. (5.41). Using this approach, unlimited expansion of the RBF covariances is avoided.

5.2.4 Active Learning

In order to ensure enough exploration of the states in off-policy learning methods, the behaviour policy is usually chosen to be a stochastic policy (e.g., a random policy). However, such a choice commonly leads to sample inefficiency, which is already a big problem in model-free RL methods. One advantage that the proposed MM-KTD learning framework offers over other optimization-based techniques (e.g., gradient descent-based methods) is the calculation of the uncertainty for the weights ($\boldsymbol{P}_{\boldsymbol{\theta}}$), which is directly related to the uncertainty of the value function. This information can then be used at each step to select the actions, which lead to most reduction in the uncertainty of the weights. Using the information of the Kalman filter (information filter [43]), the information

of the weights, which is denoted by the inverse of $\boldsymbol{P}_{\boldsymbol{\theta}}$ is updated as follows

$$\boldsymbol{P}_{\boldsymbol{\theta},k}^{-1} = \boldsymbol{P}_{\boldsymbol{\theta},k|k-1}^{-1} + \boldsymbol{h}_k R^{-1} \boldsymbol{h}_k^T. \tag{5.43}$$

Since only the second element (i.e., $\boldsymbol{h}_k R^{-1} \boldsymbol{h}_k^T$) in the right hand side of Eq. (5.43) is affected by the choice of the action (as it changes $\boldsymbol{h}_k$), the action is selected such that this term is maximized. More specifically, the action is obtained by maximizing the information of the weights, i.e.,

$$a_k = \arg\max_a \Big(\boldsymbol{h}_k(\boldsymbol{s}_k, a) R^{-1} \boldsymbol{h}_k^T(\boldsymbol{s}_k, a) \Big) = \arg\max_a \Big(\boldsymbol{h}_k(\boldsymbol{s}_k, a) \boldsymbol{h}_k^T(\boldsymbol{s}_k, a) \Big). \tag{5.44}$$

The second equality in Eq. (5.44) is constructed as R is a scalar. The proposed behavior policy in Eq. (5.44) is different from that of Reference [72], where a random policy was introduced, which favored actions with less certainty of the value function. Even though favoring the actions that reduce the uncertainty of the value function is a good idea, the random nature of such policies make them less sample efficient than expected. The proposed MM-KTD framework is briefed in Algorithm 4. It is worth further clarifying computation of Step 6 in Algorithm 4. For learning the optimal policy, the control action a_k is selected based on the behavioral policy, which leads to most reduction in the uncertainty of the weights in Eq. (5.44). Once the system has been converged, the resulted optimal policy will be used based on Eq. (5.6) to select the actions during the testing phase. Because the state-space is continuous, we have approximated the value function of Eq. (5.6) using RBFs. The value function is estimated with a weighted sum of RBFs ($\boldsymbol{\phi}(\boldsymbol{s}_k, a_k)$) with the weight vectors $\boldsymbol{\theta}_k$. For estimating the weights, the sample transition of the system, i.e., $\boldsymbol{s}_k, a_k$ and the gained reward r_k are used in a Kalman-based approach as in Eqs. (5.11), and (5.13). Therefore, the control action in Step 6 will be found based on the proposed active learning behavioral policy in Eqs. (5.12), and (5.44). Because the matrix $\big(\boldsymbol{h}_k(\boldsymbol{s}_k, a) \boldsymbol{h}_k^T(\boldsymbol{s}_k, a)\big)$ that generates the control action cannot be maximized, we have maximized its trace $\big(\boldsymbol{h}_k^T(\boldsymbol{s}_k, a) \boldsymbol{h}_k(\boldsymbol{s}_k, a)\big)$. Finally, we note that the proposed MM-KTD algorithm is designed for systems with finite number of actions. It is worth mentioning that having infinite number of actions per state is typical of continuous control tasks [87, 92]. Extension of the proposed framework for application to infinite-dimensional [93] action space is an interesting direction for future research work.

Algorithm 4 THE PROPOSED MM-KTD FRAMEWORK

Learning Phase:
Set $\boldsymbol{\theta}_0, \boldsymbol{P}_{\boldsymbol{\theta},0}, \boldsymbol{F}, \boldsymbol{\mu}_{n,i_d}, \boldsymbol{\Sigma}_{n,i_d}$ for $n = 1, 2, \ldots, N$ and $i_d = 1, 2, \ldots, D$
Repeat (for each episode):
  Initialize $\boldsymbol{s}_k$
  While $\boldsymbol{s}_k \neq \boldsymbol{s}_T$ **do**:
    $a_k = \arg\max_a \left(\boldsymbol{h}_k(\boldsymbol{s}_k, a) \boldsymbol{h}_k^T(\boldsymbol{s}_k, a) \right)$
    Take action a_k , observe $\boldsymbol{s}_{k+1}, r_k$
    Calculate $\boldsymbol{\phi}(\boldsymbol{s}, a)$ via Eqs. (5.32) and (5.33)
    $\boldsymbol{h}_k(\boldsymbol{s}_k, a_k) = \boldsymbol{\phi}(\boldsymbol{s}_k, a_k) - \gamma \arg\max_a \boldsymbol{\phi}(\boldsymbol{s}_{k+1}, a)$
    $\hat{\boldsymbol{\theta}}_{k|k-1} = \boldsymbol{F}\hat{\boldsymbol{\theta}}_k$
    $\boldsymbol{P}_{\boldsymbol{\theta},k|k-1} = \boldsymbol{F}\boldsymbol{P}_{\boldsymbol{\theta},k-1}\boldsymbol{F}^T + \boldsymbol{Q}$
    for $i = 1 : M$ **do**:
      $\boldsymbol{k}_k^{(i)} = \boldsymbol{P}_{\boldsymbol{\theta},k|k-1}\boldsymbol{h}_k(\boldsymbol{h}_k^T\boldsymbol{P}_{\boldsymbol{\theta},k|k-1}\boldsymbol{h}_k + R^{(i)})^{-1}$
      $\hat{\boldsymbol{\theta}}_k^{(i)} = \hat{\boldsymbol{\theta}}_{k|k-1} + \boldsymbol{k}_k^{(i)}(r_k - \boldsymbol{h}_k^T\hat{\boldsymbol{\theta}}_{k|k-1})$
    $\boldsymbol{P}_{\boldsymbol{\theta},k}^{(i)} = (\boldsymbol{I} - \boldsymbol{K}_k^{(i)}\boldsymbol{h}_k^T)\boldsymbol{P}_{\boldsymbol{\theta},k|k-1}(\boldsymbol{I} - \boldsymbol{K}_k^{(i)}\boldsymbol{h}_k^T)^T + \boldsymbol{K}_k^{(i)}R^{(i)}\boldsymbol{K}_k^{(i)^T}$
    end for
    Compute the value of c and $w^{(i)}$ by using $\sum_{i=1}^M w^{(i)} = 1$ and Eq. (5.25)
    $\hat{\boldsymbol{\theta}}_k = \sum_{i=1}^M w^{(i)}\hat{\boldsymbol{\theta}}_k^{(i)}$
    $\boldsymbol{P}_{\boldsymbol{\theta},_k} = \sum_{i=1}^M \omega^{(i)} \left(\boldsymbol{P}_{\boldsymbol{\theta},_k}^{(i)} + (\hat{\boldsymbol{\theta}}^{(i)} - \hat{\boldsymbol{\theta}})(\hat{\boldsymbol{\theta}}^{(i)} - \hat{\boldsymbol{\theta}})^T \right)$
    RBFs Parameters Update:
    $S_k = \left(Q_{\pi^*}(\boldsymbol{s}_k, a_k) - r_k - \gamma \max_{a \in \mathcal{A}} Q_{\pi^*}(\boldsymbol{s}_{k+1}, a) \right)^2$
    if $S^{\frac{1}{2}}Q_{\pi^*} > 0$ **then**:
      Update $\boldsymbol{\Sigma}_{n,a_d}$ via Eq. (5.41)
    else:
      Update $\boldsymbol{\mu}_{n,a_d}$ via Eq. (5.42)
    end if
  end while
Testing Phase:
  Repeat (for each trial episode):
    While $\boldsymbol{s}_k \neq \boldsymbol{s}_T$ **do**:
    $a_k = \arg\max_a \boldsymbol{\phi}(\boldsymbol{s}_k, a)^T\boldsymbol{\theta}_k$
    Take action a_k, and observe $\boldsymbol{s}_{k+1}, r_k$
    End While

5.3 Experimental Results

In this section, performance of the proposed MM-KTD framework is evaluated. In order to demonstrate the effectiveness and sample efficiency of the MM-KTD, which is a model-free and multiple model RL scheme, three popular RL benchmarks, i.e., Inverted Pendulum, Mountain Car,

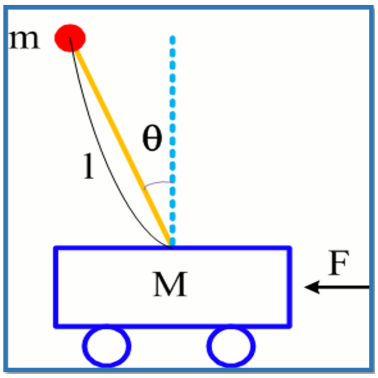


Figure 5.1: The Inverted Pendulum platform.[1]

and Lunar Lander are considered, experimented and different comparisons are performed. For the implementation of the proposed MM-KTD, a hardware with a 2.6 GHz Intel Core i7 processor and 12GB RAM has been used. To have a fair comparison, we have used optimized parameters for the NFQ method, and the KTD approach as specified in [51, 72], respectively. It is also worth mentioning that one benefit of the proposed MM-KTD framework (which comes from its multiple-model architecture) is its superior ability to deal with scenarios where enough information about the underlying parameters is not fully available.

5.3.1 Inverted Pendulum

In the first experiment, the Inverted Pendulum platform is considered, which is shown in Fig. 5.1. The weight of the base is 8 kg, while the weight of the pendulum is assumed to be 2 kg. The length of the pendulum is 0.5 m. The pendulum is initially left at an angle (arbitrarily close to the upright position) and then its base (object with mass of 8 kg) is moved to keep its balance. The base may be moved to the left or right with a force of 50N, or not moved. The episode ends once the pendulum is fallen behind a horizontal line. The goal is to prevent the pendulum from falling below the horizontal line as long as possible. The state of the system is determined as the pendulum angle from the upright position and its angular velocity (i.e., $\boldsymbol{s}_k = [\theta, \dot{\theta}]^T$). At each time step, given the state ($\boldsymbol{s}_k$) and the taken action (a_k), the next state of the system is determined through the dynamics of the system (which is not known to the agent). If the angle of the pendulum in the next state

[1] http://large.stanford.edu/courses/2007/ph210/lee2/.

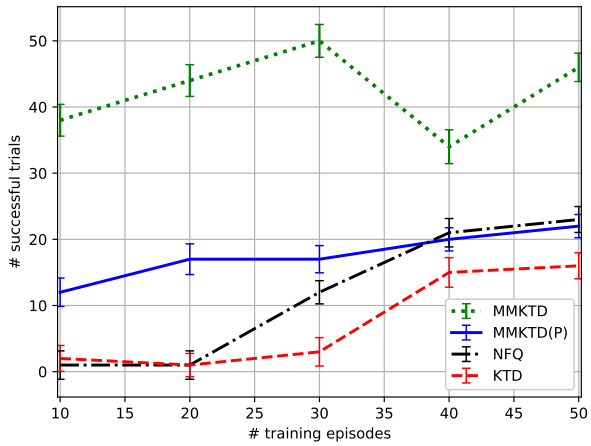


Figure 5.2: The average (lines) and 95% confidence interval (error bars) of the number of successful trials over 50 trials of the proposed MM-KTD scheme as compared with other RL methods for the Inverted Pendulum environment.

is beyond the horizontal line (i.e., $|\theta| > \pi/2$), then a reward of -1 will be fed back to the agent. Otherwise, the reward is kept as 0.

The proposed MM-KTD is employed in this problem using 9 RBFs and a bias parameter. Since there are three possible actions (i.e., $\mathcal{A} = \{-50, 0, +50\}$), the size of the feature vector is 30. The mean and covariance of the RBFs are initialized as follows

$$\boldsymbol{\mu}_{n,a_d} \in \{-\pi/4, 0, +\pi/4\} \times \{-0.5, 0, +0.5\}, \tag{5.45}$$

$$\boldsymbol{\Sigma}_{n,a_d} = \boldsymbol{I}_2, \tag{5.46}$$

where $\boldsymbol{I}_2$ is the identity matrix of size (2×2). The vector of basis functions in Eq. (5.32) are, therefore, given by

$$\boldsymbol{\phi}(\boldsymbol{s}_k, a_k = +50) = [1, \phi_{1,a_d}, \ldots, \phi_{9,a_d}, 0, \ldots 0, 0, \ldots, 0]^T \tag{5.47}$$

$$\boldsymbol{\phi}(\boldsymbol{s}_k, a_k = -50) = [0, \ldots, 0, 1, \phi_{1,a_d}, \ldots \phi_{9,a_d}, 0, \ldots, 0]^T \tag{5.48}$$

$$\boldsymbol{\phi}(\boldsymbol{s}_k, a_k = 0) = [0, \ldots, 0, 0, \ldots, 0, 1, \phi_{1,a_d}, \ldots \phi_{9,a_d}]^T, \tag{5.49}$$

where the value of ϕ_{n,a_d} for $n \in \{1, 2, \ldots, 9\}$ is calculated via Eq. (5.33). The initial values of $\lambda_{\boldsymbol{\mu}}$

and $\lambda_{\boldsymbol{\Sigma}}$ are selected as 200 and 100, respectively by using trial and error to keep the system stable. The time step is selected to be 10 milliseconds and the discount factor is selected as 0.95, (i.e., $\gamma = 0.95$). The process noise covariance is selected small ($\boldsymbol{Q}_k = 10^{-3}\boldsymbol{I}_{30}$), and the transition matrix is selected to be the identity matrix ($\boldsymbol{F} = \boldsymbol{I}_{30}$). The measurement noise covariance candidates are selected from the following set,

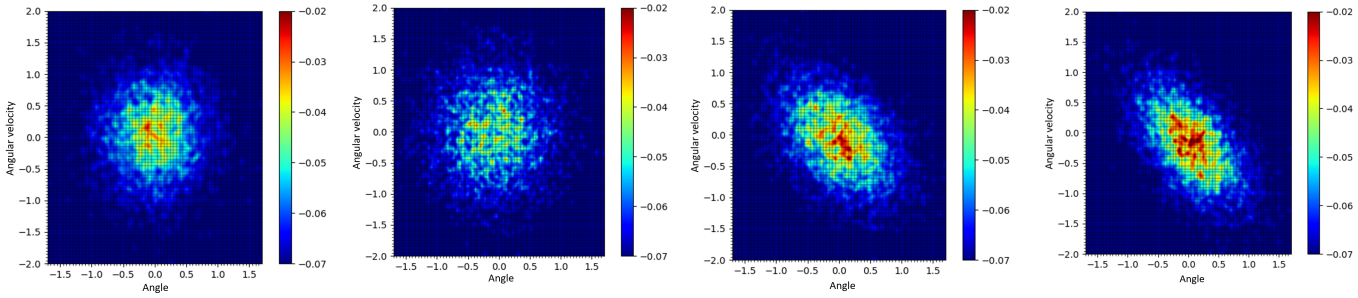
$$R^{(i)} \in \{0.01, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100\}. \tag{5.50}$$

The initial weights are selected to be zero, (i.e., $\boldsymbol{\theta}_0 = \mathbf{0}_{30}$), while the initial error covariance is chosen as $\boldsymbol{P}_{\boldsymbol{\theta},0} = 10\boldsymbol{I}_{30}$. Each experiment is started from an angle randomly selected from a normal distribution with mean zero and standard deviation of 0.1. The agent is first trained through a set of episodes, then tested in 50 episodes to find if it can keep the pendulum in balance for at least 5 seconds (500 samples). Various number of episodes are used for training the system. To highlight the effectiveness of the proposed MM-KTD, the achieved results are compared with that of KTD method of [72], MM-KTD with no active learning (denoted by MM-KTD (P)), and Neural Fitted Q (NFQ) learning method of [51], which performs the update session by considering an entire set of transition experiences in an off-line fashion, instead of using an online approach for updating the neural value function. More specifically, the NFQ scheme optimizes the sequence of the loss function using Rprop algorithm, which is a supervised learning method for batch learning to update the parameters of the Q-network. The available historical observations without performing any exploration that is used in the NFQ leads to a substantially lower amount of training experience required for learning optimal policies. In this approach, we just have a loop over trained steps. The NFQ uses the current policy in order to get the target value, while the DQN uses the target network to achieve the goal. As the nature of this chapter is to use the proposed algorithm to restrict the number of learning episodes with reduced number of training data, which is critical for practical application in real scenarios, we focused only on the algorithms with the least required learning episodes.

Each testing scenario is repeated 10 times for a specific number of training episodes to minimize the randomness of the achieved number of successful trials out of total 50 trials. For evaluation

Table 5.1: The number (percentage) of successful trials of the proposed MM-KTD scheme as compared with other RL methods for the Inverted Pendulum environment.

# Training Episodes	KTD	MM-KTD (P)	MM-KTD	NFQ
10	2.1/50 (4.2%)	12.3/50 (24.6%)	38.5/50 (77%)	1.9/50 (3.8%)
20	1/50 (2%)	17.4/50 (34.8%)	44/50 (88%)	1.5/50 (3%)
30	3.6/50 (7.2%)	17.6/50 (35.2%)	50/50 (100%)	21.6/50 (43.2%)
40	15/50 (30%)	20.7/50 (41.4%)	34/50 (68%)	15/50 (30%)
50	16.5/50 (33%)	22.0/50 (44%)	46.6/50 (93.2%)	23/50 (46%)



(a) (b) (c) (d)

Figure 5.3: The state value function of the greedy policy ($V_{\pi^*}(\boldsymbol{s})$) after 10 episodes of training using: (a) MM-KTD (b) MM-KTD (P) (c) NFQ, and (d) KTD.

purposes, we formed the average and 95% confidence interval of the number of successful trials out the 10 repetitions. Fig. 5.2 shows the resulted mean (lines) and 95% confidence interval (error bars) for different number of training episodes ranging from 10 to 50, which are also briefed in Table 5.1. As it was expected, the proposed MM-KTD is the most sample efficient algorithm of all. In addition, the proposed method offers the highest asymptotic performance in the performed experiments. This superior performance comes as a result of adaptive estimation of the value function through active Q-learning. It is expected that the performance of MM-KTD (P) and NFQ become better with more training episodes and get closer to that of the proposed MM-KTD. However, original KTD cannot reach that level of performance as it lacks accurate knowledge of the filter's parameters.

The state value function of the methods after 10 training episodes are depicted in Fig. 5.3, which the x and y axis represent θ and $\dot{\theta}$ of the pendulum, respectively. It is worth mentioning that when the state value function has the higher values around the vertical (origin) position of the Inverted Pendulum (states close to $\theta = \dot{\theta} = 0$), it means that the selected actions, resulted from the behavioral policy, have led to the states near the vertical position, which is the expected result i.e., the pendulum is above the horizontal line. Therefore, based on the Fig. 5.3, the proposed active learning method causes higher sample efficiency of MM-KTD scheme compared to the other schemes. Due to the higher sample efficiency, MM-KTD can quickly concentrate higher values for states closer to the origin. However, other practiced methods are still far from this stage, therefore, fail to perform in an acceptable fashion with restricted amount of experience.

Finally, to evaluate stability of the utilized RBFs, we have conducted the following Monte-Carlo (MC) study. In particular, we have fixed the number of RBFs to be 9 and their means as selected in Eq. (5.45). Then, the proposed MM-KTD scheme has been performed on the Inverted Pendulum task for 200 different trials. The entire process has been repeated 100 times (i.e., through a MC simulation of 100 runs) using three different values of the widths ($\mathbf{\Sigma}$) of the RBFs. The number of steps to the goal was averaged over the 100 runs. Fig. 5.4 illustrates potential stability of the utilized RBFs. As can be observed from Fig. 5.4, steady state performance can be achieved by using the RBFs for the value function approximation.

5.3.2 Mountain Car

In the second experiment, the Mountain Car platform, is shown in Fig. 5.5, is chosen for the evaluation. Mountain Car is a classic RL problem where the objective is to create an algorithm which learns to climb a steep hill to reach the goal marked by a flag. The car's engine is not powerful enough to drive up the hill without a head start, therefore, the car must drive up the left hill to obtain enough momentum to scale the steeper hill to the right and reach the goal. The state of the system is the position of the car and its velocity (i.e. $\boldsymbol{s} = [x, \dot{x}]^T$). The possible actions are restricted within ($\mathcal{A} = \{0, 1, 2\}$) which are "push left", "no push", and the "push right", respectively. The road ends at the position $-1.2m$, i.e., the position must be greater that $-1.2m$. The task is to reach the top where the position must be larger or equal to $0.5m$. There would be a -1 reward for each

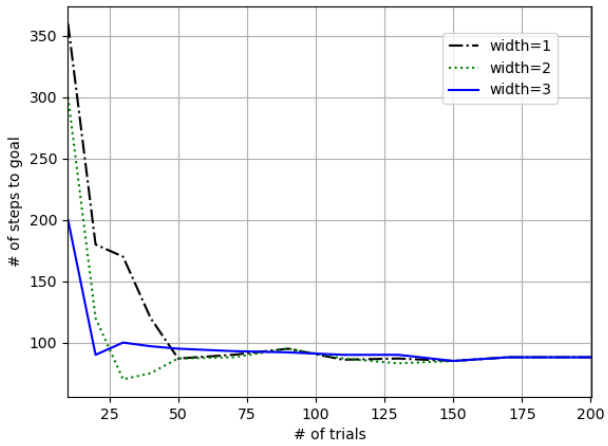


Figure 5.4: Stability analysis of the RBFs.

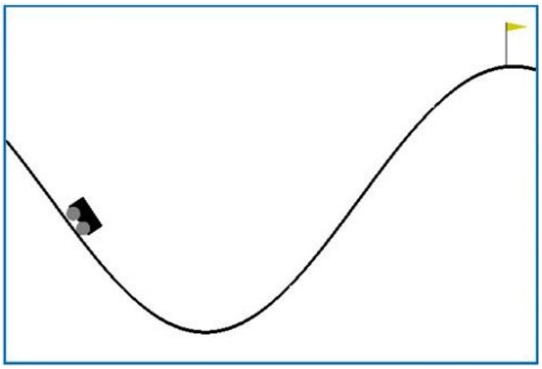


Figure 5.5: The Mountain Car environment.[2]

step where the car is unable to mount the hill to reach the goal and there is no penalty for climbing the left hill acting as a wall when is reached. Each episode starts with a random position ranging from $-0.6m$ to $-0.4m$ with no velocity.

Similar to the Inverted Pendulum experiment presented in Section 5.3.1, 9 RBFs and a bias parameter are used for the proposed MM-KTD algorithm. Therefore, the size of the feature vector

[2]https://medium.com/@ts1829/solving-mountain-car-with-q-learning-b77bf71b1de2.

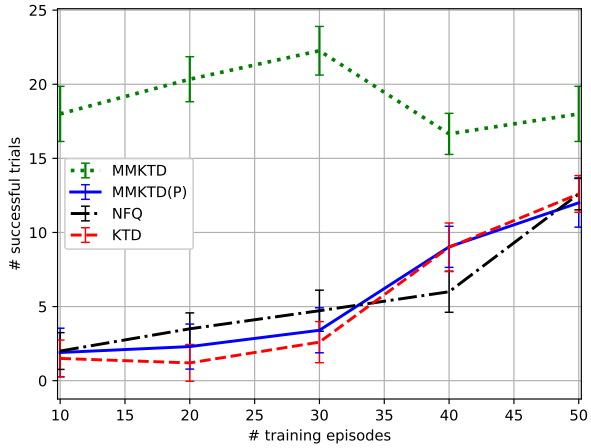


Figure 5.6: The average (lines) and 95% confidence interval (error bars) of the number of successful trials over 50 trials of the proposed MM-KTD scheme as compared with other RL methods for the Mountain Car environment.

is 30. The mean and covariance of the RBFs are initialized as follows

$$\boldsymbol{\mu}_{n,a_d} \in \{-0.775, -0.35, +0.775\} \times \{-0.035, 0, +0.035\} \tag{5.51}$$

$$\boldsymbol{\Sigma}_{n,a_d} = \boldsymbol{I}_2. \tag{5.52}$$

The initial values of $\lambda_{\boldsymbol{\mu}}$ and $\lambda_{\boldsymbol{\Sigma}}$ are selected as 100 and 80, respectively. The time step is selected to be 50 milliseconds and the discount factor is chosen as 0.95. The process noise covariance is set to $\boldsymbol{Q}_k = 10^{-3}\boldsymbol{I}_{30}$, and the transition matrix is selected to be the identity matrix ($\boldsymbol{F} = \boldsymbol{I}_{30}$). Candidate values for R are selected from the same set was selected for the Inverted Pendulum environment.

The initial weights are selected to be zero (i.e., $\boldsymbol{\theta}_0 = \boldsymbol{0}_{30}$), whereas the initial error covariance is chosen as $\boldsymbol{P}_{\boldsymbol{\theta},0} = 10\boldsymbol{I}_{30}$. The agent first is trained through different number of episodes, then tested 10 times for 50 episodes to find if it can reach the flag during 200 samples of each episode. Fig. 5.6 shows the results of KTD [72], MM-KTD, MM-KTD (P), and NFQ learning method of [51]. As expected, the performance of KTD, MM-KTD (P) and NFQ improves with increased training episodes. However, KTD, MM-KTD (P) and NFQ can't provide that level of performance is achieved by MM-KTD with the lowest number of training episodes in this experiment (10). Based on the achieved results shown in Table 5.2, MM-KTD is the most sample efficient

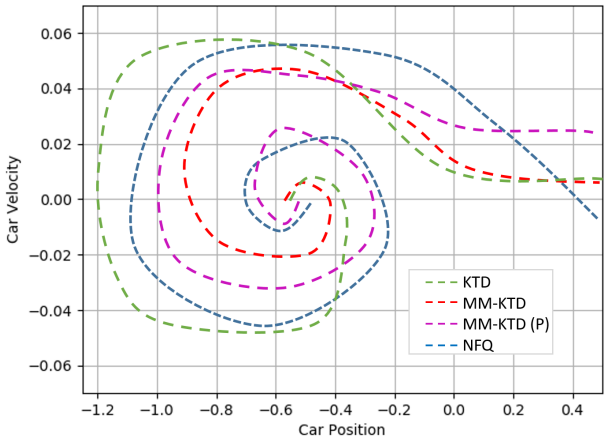


Figure 5.7: State space trajectory of the Mountain Car environment at episode number 50.

Table 5.2: The number (percentage) of successful trials of the proposed MM-KTD scheme as compared with other RL methods for the Mountain Car environment.

# Training Episodes	KTD	MM-KTD (P)	MM-KTD	NFQ
10	1.5/50 (3%)	1.9/50 (3.8%)	18.0/50 (36%)	2.0/50 (4%)
20	1.2/50 (2.4%)	2.3/50 (4.6%)	20.3/50 (40.6%)	3.5/50 (7%)
30	2.6/50 (5.2%)	3.4/50 (6.8%)	22.2/50 (44.4%)	4.72/50 (9.44%)
40	9.2/50 (18.4%)	9/50 (18%)	16.6/50 (33.2%)	6/50 (12%)
50	12.3/50 (24.6%)	12.0/50 (24%)	18.0/50 (36%)	12.6/50 (25.2%)

approach of all tested algorithms. Fig. 5.7 depicts the trajectories of the system's states (position, velocity) for the Mountain Car environment at episode number 50. As can be observed from the Fig. 5.7, the episode starts from zero velocity and a random position in the range $[-0.6, -0.4]$, and ends when the position reaches to $0.5m$. Fig. 5.8 shows the optimal agent's actions resulted from applying the MM-KTD scheme over a combination of positions and velocities. Based on Fig. 5.8, the agent moves left when its velocity is negative. Most of the times that the velocity is positive, the agent moves to the right and sometimes does nothing.

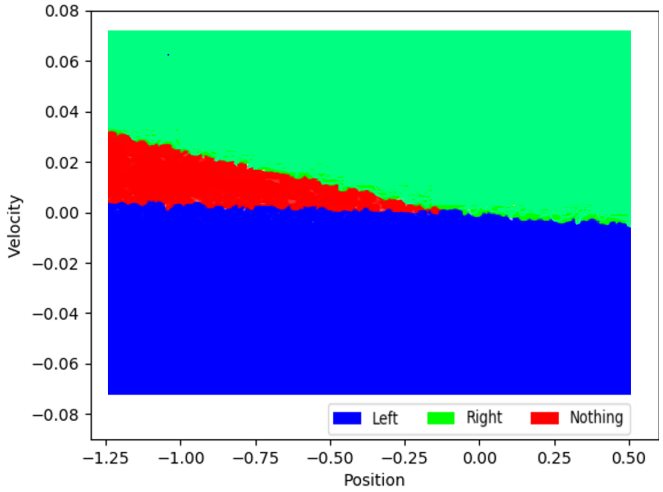


Figure 5.8: Actions resulted from optimal policy of MM-KTD algorithm at the episode number 50 for Mountain Car environment.

5.3.3 Lunar Lander

In the third experiment, we focus on the Lunar Lander environment, which is a more complicated environment compared to the two previous ones. In the Lunar Lander environment, the goal is for the RL agent to learn to land successfully on a landing pad located at coordinate $(0, 0)$ in a randomly generated surface on the moon as shown in Fig. 5.9. The state space of the system consists of the agent's position (x, y) in space, horizontal and vertical velocity (v_x, v_y), orientation in space θ, and angular velocity $\dot{\theta}$. The agent has four possible actions, i.e., do nothing; firing the left engine, firing the main engine, and; firing the right engine ($\mathcal{A} = \{0, 1, 2, 3\}$). Reward for landing on the pad is about 100 to 140 points, varying on the lander placement on the pad. If the lander moves away from the landing pad it loses reward. Each episode terminates if the lander lands or crashes, receiving additional +100 or -100 points, respectively. Each leg ground contact worth +10 points. Firing the main engine results in a -0.3 point penalty for each frame. The problem is considered solved if the agent receives +200 points over 100 iterations. The RBFs of order two are considered for each state variable resulting in 64 RBFs for each action. Consequently, the size of the feature vector $\boldsymbol{\phi}(\boldsymbol{s}_k, a_k)$ will be 256. Based on the useful range of each variable of state vector, the initial

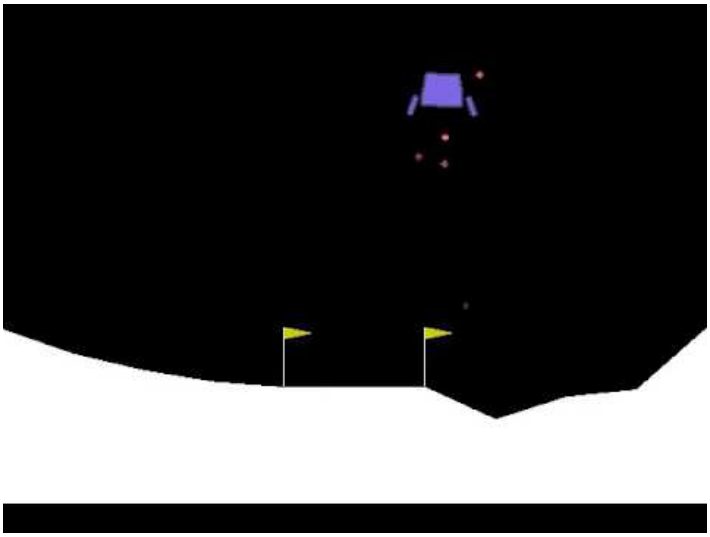


Figure 5.9: The Lunar Lander environment.[3]

mean and covariance of the RBFs are chosen as follows

$$\boldsymbol{\mu}_{n,a_d} \in \{-0.333, +0.333\}^6, \tag{5.53}$$

$$\boldsymbol{\Sigma}_{n,a_d} = 2\boldsymbol{I}_6. \tag{5.54}$$

The initial values of $\lambda_{\boldsymbol{\mu}}$ and $\lambda_{\boldsymbol{\Sigma}}$ are both selected as 200 to keep the system stable. The discount factor is selected as 0.99. The process noise covariance is set to $\boldsymbol{Q}_k = 10^{-1}\boldsymbol{I}_{256}$. Candidate values for R are selected from the same set as was used for the Inverted Pendulum environment. Like the two previous environments, first, the agent is trained through different number of episodes changing from 10 to 50, then tested 10 times for 50 trials to find if the agent can successfully land on the landing pad over 100 steps. Fig. 5.10 depicts the results of applying the KTD [72], MM-KTD, MM-KTD (P)), and NFQ learning [51] scheme. It can be observed that the proposed approach outperforms its counterparts.

[2]https://towardsdatascience.com/solving-lunar-lander-openaigym-reinforcement-learning-785675066197.

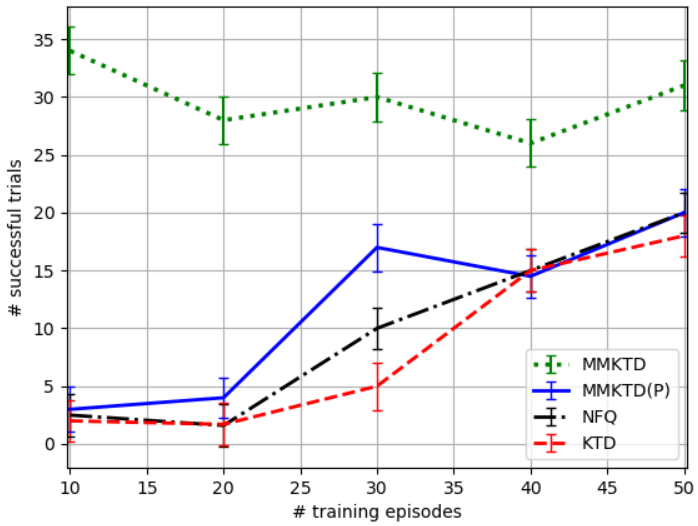


Figure 5.10: The average (lines) and 95% confidence interval (error bar) of the number of successful trials over 50 trials of the proposed MM-KTD scheme as compared with other RL methods for the Lunar Lander environment.

5.3.4 Parameters Selection

It is worth providing more details on different values assigned to the underlying parameters in the three simulations. First, it is worth mentioning that one aspect of the proposed multiple-model type framework is to address the issues of reducing dependence of RL performance on its parameter values. For adaptation of the RBFs' parameters, the initial centers of the RBFs are, typically, distributed evenly along each dimension of the state vector, leading to n^d centers for d state variables and a given order n. Initial variance of each state's variable (σ^2) is often set to $\frac{2}{n-1}$. For the Inverted Pendulum and Mountain Car, the value of $n = 3$ is selected following the KTD approach of [72] for having fair comparison. For the Lunar Lander, the value of n is selected to be 2. Furthermore, the values for $\lambda_{\boldsymbol{\mu}}$ and $\lambda_{\boldsymbol{\Sigma}}$ are selected in such a way to keep the system stable. In addition, based on Fig. 5.4, it can be observed that RBFs are fairly stable against selection of their underlying parameters. The discount factor denoted by γ affects how much weight is given to the future rewards in the value function. A discount factor $\gamma = 0$ will result in state/action values representing the

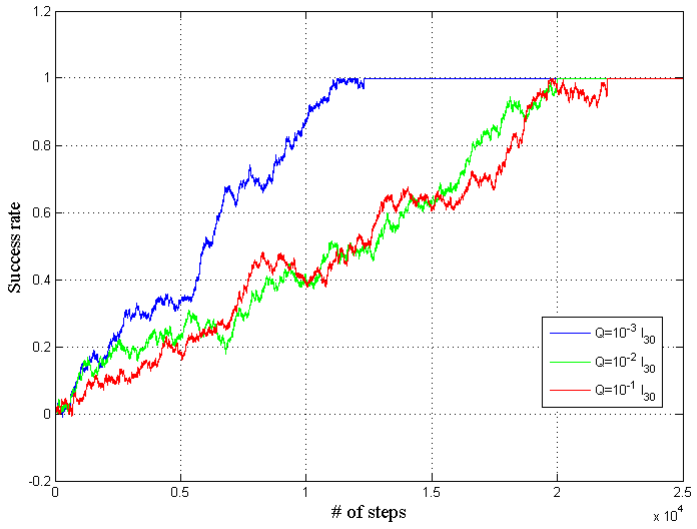


Figure 5.11: Success rate during 50 training episodes for Inverted Pendulum environment.

immediate reward, while a higher discount factor will result in the values representing the cumulative discounted future reward that an agent is expected to receive (behaving under a given policy). Commonly (as is the case in the implemented environments), a large portion of the reward is earned upon reaching the goal. To prioritize this final success, we expect an acceptable γ to be close 1. In our manuscript, γ is set to the high values of 0.95 and 0.99.

The prior $\boldsymbol{\theta}_0$ should be initialized to a value close to the expected optimal value based on historical data, or to a default value, e.g., the zero vector. The prior $\boldsymbol{P}_0$ quantifies the certainty that the user have in the initialized prior $\boldsymbol{\theta}_0$, the lower the less certainty. The process noise covariance is a design parameter, the proposed MM-KTD algorithm allows, systematically, to use a set of different values of process noise covariance ($\boldsymbol{Q}$). We have conducted a sensitivity analysis experiment for the Inverted Pendulum environment to evaluate sensitivity to this design parameter for the scenario where only a single initial value can be assigned. Fig. 5.11 presents the agent's success rate during the training phase over 50 episodes based on three different assignments to $\boldsymbol{Q}$. It can be observed that different values of the process noise covariance affect the time which took the agent to complete the training process. Finally and as stated previously, variable R, i.e., the measurement noise variance, is one of the most important parameters to be identified. To select this parameter, our intuition in the

proposed MM-KTD framework is to cover the potential range of the measurement noise variance using M mode-matched filters. The parameter M is set a-priori denoting the number of candidates $R^{(i)}$, for ($1 \leq i \leq M$), for the observation noise variance. For example, in the experiments, we have selected M to be equal to 11.

5.4 Conclusion

In this chapter, a Multiple-Model Kalman Filter-based Temporal Differences framework, referred to as the MM-KTD, was proposed, which deals with the problems of sample efficiency, online learning, prior information and memory of other RL algorithms. The proposed MM-KTD algorithm was evaluated based on three RL experiments. Based on the achieved results, the proposed algorithm achieved the highest number of successful trials while its number of training episodes for learning the best policy was considerably less in comparison to its counterparts. The need for restricted number of learning episodes results in the reduced training data, which is critical for practical applications in the real scenarios.

Chapter 6

Summary and Future Research Directions

This chapter concludes the thesis with a list of important contributions made in the dissertation and some proposed directions for future work.

6.1 Summary of Thesis Contributions

The research works performed in this thesis are motivated by recent advancements and developments of autonomous agents in Internet of Robotic Things applications. Such applications require the autonomous agents to be able to localize/navigate point-to-point within an indoor environments and perform conditional tasks with minimum human intervention. Current commercial indoor localization and agents decision support systems, typically, rely on BLE-based solutions and RL-based approaches, respectively. The focus of the thesis is on addressing the identified drawbacks of existing solutions for BLE-based indoor localization and RL-based decision making problems in the autonomous systems. In this regard, the thesis made a number of contributions [22–25] as briefly outlined below:

(1) **STUPEFY: BLE-based Set-Valued Box Particle** [22]: A novel multiple-model STUPEFY framework was proposed, which estimates the target's coarse location via an intuitive learning fashion using Gaussian distribution of the RSSI values of each zone. Then, a novel set-valued

box particle filtering (SBPF) is used to micro-locate the target in the predicted zone achieved from the learning process. More specifically, the proposed STUPEFY framework consists of the following three components:

- ***Smoothing Module***, which is developed based on the KF with the following two objectives: (i) To reduce the RSSI fluctuations, and; (ii) To model the RSSI values associated with each zone as a multivariate Gaussian and construct the smallest axes-aligned box containing the ellipsoid associated with each zone to be used for creating set-valued measurements.
- ***Coordination Module***: In contrary to existing BLE-based pattern matching solutions, output of the KF is utilized for finding the predicted zone of the target via BD. Intuitively speaking, real-time Gaussian models of the RSSI values are compared in distribution with the learned ones. An approximation of the optimal proposal distribution in terms of a predicted box is used for generating predicted particles from a bivariate Gaussian distribution inscribed in the predicted box (hence boxed PF).
- ***Micro-Localization Module***: In contrary to the conventional PF-based solutions that use point-valued RSSI measurements, the proposed SBPF computes the probability that the measured RSSI from each active beacon belongs to the set identified by the Coordination module (hence set-valued PF).

(2) **Gaussian Mixture-based Indoor Localization via BLE Sensors** [23]: To deal with the fact that RSSI-based solutions are prone to drastic fluctuations, GMMs are trained to more accurately represent the underlying distribution of RSSI values. For assigning real-time observed RSSI vectors to different zones, first a KF is applied to smooth the RSSI vector and form its Gaussian model, which is then compared in distribution with learned GMMs based on BD and via a weighted K-NN approach. More specifically, the proposed framework, referred to as BD-GMM, is composed of the following two data-driven components:

- ***Offline Phase***: The RSSI values at all of the fingerprinting zones are measured, smoothed via KF, and values associated with each location are modelled by a multivariate GMM distribution.

- ***Online Phase***: This phase consists of pattern matching and post-processing steps. In the online phase, the real time RSSI vector is measured, then a KF is applied for two reasons: (i) To smooth fluctuations in the RSSI values, and; (ii) To construct a Gaussian model of the RSSI values to be used for finding the predicted zone of the target via the two estimation scenarios: (i) The BD distance between the output of the KF and the trained GMM associated with all the zones are calculated and the one with minimum distance is identified as the target's zone, and; (ii) A weighted K-Nearest Neighbor (K-NN) algorithm is applied for location estimation, where the weights are computed based on the BD between the output of the KF and the GMM distributions of K nearest zones to the identified one.

(3) **Gaussian Sum Filtering Coupled with Wasserstein Distance for Non-Gaussian BLE-based Localization** [24]: The thesis proposed a novel GMM-based probabilistic framework for assigning real-time observed RSSI vectors to different fingerprinting zones using BD and WD. The proposed framework consists of two integrated modules as briefly described below:

- ***Offline Phase***: The RSSI values are measured from the active BLE beacons across the surveillance venue including all the zones. Then, the values at each zone is modeled with a multivariate GM distribution.
- ***Online Phase***: The proposed novel GSF algorithm, referred to as WD-GSF, is applied on the real-time RSSI values via incorporation of a WD-based clustering GMR for smoothing the RSSI values and simulating the non-Gaussian RSSI measurement noise as a GMM. These distributions outputs are then compared in distribution with GMs, learned for each zone, via utilization of the BD. Two scenarios are used for user's location estimation: (i) The BD between the output of the GSF and the trained GM associated with all the zones are calculated and the one with the minimum distance is identified as the target's zone, and; (ii) A weighted K-NN algorithm is applied for location estimation, where the weights are computed based on the BD between the output of the GSF and the GMs distributions of K nearest zones to the identified one.

(4) **MM-KTD: Multiple Model Kalman Temporal Differences for Reinforcement Learning** [25]: In the thesis, an innovative Multiple Model Kalman Temporal Differences (MM-KTD) framework is proposed for value function approximation using Multiple Model Adaptive Estimation (MMAE) of KTD to address the parameter uncertainty and overfitting problems of DNN-based approaches. The proposed MM-KTD framework benefits from the key following characteristics:

- ***An Off-policy Q-learning Adaptation***: To improve the sample efficiency of the proposed MM-KTD, an off-policy Q-learning is adopted to learn the optimal policy from the behaviour policy.

- ***MMAE-based Value Function Approximation***: An innovative MMAE scheme is adopted within the RL process, referred to as MM-KTD, to compensate for the lack of information about the measurement noise covariance as the most important parameter of the filter, while the mean and covariance of the RBFs are updated exploiting the restricted gradient descent.

- ***Incorporation of an Active Learning***: The estimated uncertainty of the value functions are exploited via an active learning approach to form the behaviour policy. This, in turn, leads to more visits to less certain values, therefore, improving the overall learning sample efficiency.

6.2 Future Research

Below, potential future research directions are discussed to further improve the proposed contributions made throughout this thesis:

(1) The proposed indoor localization frameworks have been evaluated based on real datasets collected via 4 and 5 BLE devices in a $6\,m \times 5\,m$ room. It would be interesting, as a direction for future works, to investigate performance of the proposed approaches in larger indoor environments with more Bluetooth devices.

(2) As described previously in Chapter 2, model-based RL algorithms calculates a set of candidate actions from scratch whereas model-free algorithms make choices more efficient but in

a less flexible way by storing pre-computed action values. There is an intermediate family of RL algorithms referred to as Successor Representation (SR), which makes a trade-off between efficiency and flexibility of the system. The proposed MM-KTD approach can be further improved along extending the MM-KTD framework to apply on the SR algorithm families. Extension of MM-KTD approach to the SR can be a direction for the future research.

(3) Another potential direction for future research is to explore possibility of using other BLE-based indoor localization techniques such as Time of Flight (ToF) or Angle of Arrival (AoA). This can be achieved by customizing the hardware and using new radio and directional antennas or an antenna grid.

(4) Another subject would be to explore if a hybrid solution can be developed by utilization of cheap BLE devices and other infrastructures or different signaling technologies (e.g., WiFi and/or UWB) to help improve RF based localization.

(5) In this thesis, a linear and Gaussian class (KF) of Bayesian estimation techniques is used within the RL context. It would be beneficial to apply other non-linear and non-Gaussian groups of Bayesian estimation family such as the PF. It would be interesting to investigate potentials of PF in the RL problems and evaluate its combination with other popular RL algorithms such as DNN-based solutions.

Bibliography

[1] C. Xu, Y. Sun, K.N. Plataniotis, and N. Lane "Editorial - Signal Processing and the Internet of Things," *IEEE Signal Processing Magazine*, vol. 35, no. 5, 2018.

[2] P. Spachos, I. Papapanagiotou, and K.N. Plataniotis, "Microlocation for Smart Buildings in the Era of the Internet of Things: A Survey of Technologies, Techniques, and Approaches," *IEEE Signal Processing Magazine*, vol. 35, no. 5, 2018.

[3] S. Milici, A. Lazaro, R. Villarino, D. Girbau, and M. Magnarosa, "Wireless Wearable Magnetometer-Based Sensor for Sleep Quality Monitoring," *IEEE Sensors Journal*, vol. 18, no. 5, pp. 2145-2152, March, 2018.

[4] C. Xu, L. Yang, and P. Zhang, "Practical Backscatter Communication Systems for Battery-Free Internet of Things: A Tutorial and Survey of Recent Research," *IEEE Signal Processing Magazine*, vol. 35, no. 5, pp. 16-27, Sept. 2018.

[5] S. Tarkoma, and H. Ailisto, "The Internet of Things program: The finnish perspective," *IEEE Communication Magazine*, vol. 51, no. 3, pp. 10-11, Mar. 2013.

[6] M.C. Cara, J.L. Melgarejo, G.B. Rocca, L.O. Barbosa, and I.G. Varea, "An Analysis of Multiple Criteria and Setups for Bluetooth Smartphone-based Indoor Localization Mechanism," *Journal of Sensors*, 2017.

[7] S. Alletto *et al.*, "An Indoor Location-Aware System for an IoT-Based Smart Museum," *IEEE Internet of Things Journal*, vol. 3, no. 2, pp. 244-253, April 2016.

[8] D. Ramachandram, and G. W. Taylor, "Deep Multimodal Learning: A Survey on Recent Advances and Trends," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 96-108, Nov. 2017.

[9] J. Luo, and H. Gao, "Deep Delief Networks for Fingerprinting Indoor Localization using Ultrawideband Technology," *International Journal Distributed Sensor Networks*, no. 18, 2016.

[10] S. Sadowski, and P. Spachos, "RSSI-Based Indoor Localization With the Internet of Things," *IEEE Access*, vol. 6, pp. 30149-30161, 2018.

[11] F. Zafari, I. Papapanagiotou, M. Devetsikiotis, and T.J. Hacker, "An iBeacon based Proximity and Indoor Localization System," *arXiv/1703.07876*, 2017.

[12] S. Spano, G.C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Matta, A. Nannarelli, and M. Re, "An Efficient Hardware Implementation of Reinforcement Learning: The Q-Learning Algorithm," *IEEE Access*, vol. 7, pp. 186340-186351, 2019.

[13] M. Seo, L.F. Vecchietti, S. Lee, and D. Har, "Rewards Prediction-Based Credit Assignment for Reinforcement Learning With Sparse Binary Rewards," *IEEE Access*, vol. 7, pp. 118776-118791, 2019.

[14] A. Toubman *et al.*, "Modeling behavior of Computer Generated Forces with Machine Learning Techniques, the NATO Task Group approach," *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Budapest, 2016, pp. 001906-001911.

[15] J. J. Roessingh *et al.*, "Machine Learning Techniques for Autonomous Agents in Military Simulations - Multum in Parvo," *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Banff, AB, 2017, pp. 3445-3450.

[16] M. Atashi, M. Salimibeni, P. Malekzadeh, M. Barbulescu, K.N. Plataniotis, and A. Mohammadi, "Multiple Model BLE-based Tracking via Validation of RSSI Fluctuations under Different Conditions," *International Conference on Information Fusion (FUSION)*, Ottawa, ON, Canada, 2019, pp. 1-6.

[17] Z. Nan, Z. Hongbo, F. Wenquan, and W. Zulin, "A Novel Particle Flter Approach for Indoor Positioning by Fusing WIFI and Inertial Sensors," *Chinese Journal of Aeronautics*, vol. 28, no. 6, pp.1725-1734, 2015.

[18] J. Tosi, F.Taffoni, M. Santacatterina, R. Sannino, and D. Formica "Performance Evaluation of Bluetooth Low Energy: A Systematic Review. Sensors," Sensors, vol. 17, pp. 2898, 2017.

[19] R. Bellman, "The Theory of Dynamic Programming," *Bulletin of the American Mathematical Society*, vol. 60, no. 6, pp. 503-515, 1954.

[20] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information Theoretic MPC for Model-based Reinforcement Learning," *International Conference on Robotics and Automation (ICRA)*, 2017.

[21] S. Ross, and J. A. Bagnell, "Agnostic System Identification for Model-based Reinforcement Learning," *arXiv:1203.1007*, 2012.

[22] P. Malekzadeh, A. Mohammadi, M. Barbulescu, and K.N. Plataniotis, "STUPEFY: Set-Valued Box Particle Filtering for BLE-based Indoor Localization" *IEEE Signal Processing Letters*, vol. 26, no. 12, pp. 1773-1777, Dec. 2019.

[23] P. Malekzadeh, M. Salimibeni, M. Atashi, M. Barbulescu, K. N. Plataniotis, and A. Mohammadi, "Gaussian Mixture-based Indoor Localization via Bluetooth Low Energy Sensors," *2019 IEEE SENSORS*, Montreal, QC, Canada, 2019, pp. 1-4.

[24] P. Malekzadeh, S. Mehryar, P. Spachos, K. N. Plataniotis, and A. Mohammadi, "Non-Gaussian BLE-Based Indoor Localization Via Gaussian Sum Filtering Coupled with Wasserstein Distance," *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020, pp. 9046-9050.

[25] P. Malekzadeh, M. Salimibeni, A. Mohammadi, A. Assa, and K.N. Plataniotis, "MM-KTD: Multiple Model Kalman Temporal Differences for Reinforcement Learning," *IEEE Access*, vol. 8, pp. 128716-128729, 2020.

[26] A. Mohammadi, and K. N. Plataniotis, "Improper Complex-Valued Bhattacharyya Distance," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 5, pp. 1049-1064, 2016.

[27] F.S. Danis, and A.T. Cemgil, "Model-based Localization and Tracking using Bluetooth Low-Energy Beacons," *Sensors*, vol. 17, no. 11, 2017.

[28] Z. Iqbal *et al.*, "Accurate Real Time Localization Tracking in a Clinical Environment using Bluetooth Low Energy and Deep Learning," *PLoS One*, vol. 13, no. 10, e0205392.

[29] C.S. Chen, "Artificial Neural Network for Location Estimation in Wireless Communication Systems," *Sensors*, vol. 12, pp. 2798-2817, 2012.

[30] S.R Jondhale, and R.S. Deshpande, "Effecient Localization of Sensor Node using Generalized Regression Neural Networks in Large Scale Farmland," *International Journal of Communication System, Willey Online*, vol. 32, Issue 16, 2019.

[31] S.R Jondhale, and R.S. Deshpande, "Modified Kalman Filtering Framework Based Real Time Target Tracking Against Environmental Dynamicity in Wireless Sensor Networks," *Ad Hoc & Sensor Wireless Networks*, vol. 40, pp. 119-143, 2018.

[32] A.Kushki, K.N. Plataniotis, and A.N. Venetsanopoulos, "WLAN Positioning Systems: Principles and Applications in Location-based Services," *WLAN Positioning Systems: Principles and Applications in Location-Based Services*, pp. 1-148, 2012.

[33] A. Mohammadi, and A. Asif, "Distributed Consensus + Innovation Particle Filtering for Bearing/Range Tracking With Communication Constraints," *IEEE Transactions on Signal Processing*, vol. 63, no. 3, pp. 620-635, 2015.

[34] A. Mohammad, and A. Asif, "Diffusive Particle Filtering for Distributed Multisensor Estimation," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, 2016, pp. 3801-3805.

[35] A. Mohammadi, S. Davar, and K. N. Plataniotis, "Ternary-Event-Based State Estimation With Joint Point, Quantized, and Set-Valued Measurements," *IEEE Signal Processing Letters*, vol. 25, no. 5, pp. 665-669, 2018.

[36] D.R. Morrell, and W. C. Stirling, "Set-values Filtering and Smoothing," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 1, pp. 184-193, 1991.

[37] H. Song, P. Shi, C. Lim, W. Zhang, and L. Yu, "Set-Membership Estimation for Complex Networks Subject to Linear and Nonlinear Bounded Attacks," IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 1, pp. 163-173, Jan. 2020.

[38] YB. Bai, S. Wu, G. Retscher, A. Kealy, L. Holdend, M. Tomkoe, A. Borriakd, B. Hua, H.R Wuf and K. Zhang "A New Method for Improving Wi-Fi-based Indoor Positioning Accuracy," *Journal of Location Based Services*, vol. 8, no. 3, pp. 35-147, 2014.

[39] M. Khan, Y.D Kai, and H.U Gul, "Indoor Wi-Fi Positioning Agorithm based on Combination of Location Fingerprint and Unscented KaIman Filter," *IEEE International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, 2017, pp. 693-698.

[40] Z. Wu, E. Jedari, R. Muscedere, and R. Rashidzadeh, "Improved Particle Filter based on WLAN RSSI Fngerprinting and Smart Sensors for Indoor Localization," *Computer Communications*, vol. 83, pp. 64-71, 2016.

[41] S.R Jondhale, and R.S. Deshpande, "Kalman Filtering Framework-based Real Time Target Tracking in Wireless Sensor Networks using Generalized Regression Neural Networks," *IEEE Sensors Journal*, vol. 19, no. 1, pp. 224-233, 2019.

[42] S.R Jondhale, and R.S. Deshpande, "GRNN and KF Framework based Real Time Target Tracking using PSOC BLE and Smartphone," *Ad Hoc Networks*, vol. 84, pp.19-28, 2019.

[43] A. Mohammadi, and K. N. Plataniotis, "Event-Based Estimation With Information-Based Triggering and Adaptive Update," *IEEE Transactions on Signal Processing*, vol. 65, no. 18, pp. 4924-4939, 2017.

[44] A. Assa, and K.N. Plataniotis, " Wasserstein-Distance-Based Gaussian Mixture Reduction," *IEEE Signal Processing Letters*, vol. 25, no. 10, pp. 1465-1469, 2018.

[45] G. Terejanu, *et al*, "Adaptive Gaussian sum Filter for Nonlinear Bayesian Estimation," *IEEE Transactions on Automatic Control*, vol. 56, no. 9, pp. 2151-2156, 2011.

[46] A. Mohammadi, and K.N. Plataniotis, "Complex-valued Gaussian Sum Filter for Nonlinear Filtering of non-Gaussian/non-circular Noise," *IEEE Signal Processing Letters*, vol. 22, no. 4, pp. 440-444, 2014.

[47] D. Schieferdecker, and MF. Huber, "Gaussian Mixture Reduction via Clustering," *IEEE International Conference on Information Fusion*, pp. 1536-1543, 2009.

[48] AR. Runnalls, "Kullback-Leibler Approach to Gaussian Mixture Reduction," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 3, pp. 989-999, 2007.

[49] A. Lazaric, M. Restelli, and A. Bonarini, "Reinforcement Learning in Continuous Action Spaces Through Sequential Monte Carlo Methods," *Advances in Neural Information Processing Systems*, 2008, pp. 833-840.

[50] E. Keogh, and A. Mueen, "Curse of Dimensionality," *Encyclopedia of Machine Learning*, Springer, 2011, pp. 257-258.

[51] M. Riedmiller, "Neural Fitted Q Iteration-first Experiences with a Data Efficient Neural Reinforcement Learning Method," *European Conference on Machine Learning*, Springer, 2005, pp. 317-328.

[52] Y. Tang, H. Guo,T. Yuan, X. Gao, X. Hong, Y. Li, J. Qiu, Y. Zuo, and J. Wu, "Flow Splitter: A Deep Reinforcement Learning-Based Flow Scheduler for Hybrid Optical-Electrical Data Center Network," *IEEE Access*, vol. 7, pp. 129955-129965, 2019.

[53] C. Hu, and M. Xu, "Adaptive Exploration Strategy With Multi-Attribute Decision-Making for Reinforcement Learning," *IEEE Access*, vol. 8, pp. 32353-32364, 2020.

[54] M. Kim, S. Lee, J. Lim, J. Choi, and S.G. Kang, "Unexpected Collision Avoidance Driving Strategy Using Deep Reinforcement Learning," *IEEE Access*, vol. 8, pp. 17243-17252, 2020.

[55] J. Xie, Z. Shao, Y. Li, Y. Guan, and J. Tan, "Deep Reinforcement Learning with Optimized Reward Functions for Robotic Trajectory Planning," *IEEE Access*, vol. 7, pp. 105669-105679, 2019.

[56] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *arXiv:1312.5602*, 2013.

[57] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous Control with Deep Reinforcement Learning," *arXiv:1509.02971*, 2015.

[58] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-critic Methods," *arXiv:1802.09477*, 2018.

[59] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Offpolicy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," *arXiv:1801.01290*, 2018.

[60] H. Van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," *AAAI*, vol. 2. Phoenix, AZ, 2016, p. 5.

[61] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," *International Conference on Machine Learning*, 2016, pp. 1928-1937.

[62] S. Haykin, "Neural Networks: A Comprehensive Foundation," *Prentice Hall PTR*, 1994.

[63] W. T. Miller, F. H. Glanz, and L. G. Kraft, "Cmas: An Associative Neural Network Alternative to Backpropagation," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1561-1567, 1990.

[64] R. M. Kretchmar and C. W. Anderson, "Comparison of CMACs and Radial Basis Functions for Local Function Approximators in Reinforcement Learning," *International Conference on Neural Networks*, vol. 2. IEEE, 1997, pp. 834-837.

[65] G. Konidaris, S. Osentoski, and P. S. Thomas, "Value Function Approximation in Reinforcement Learning using the Fourier Basis," *AAAI*, vol. 6, 2011, p. 7.

[66] I. Menache, S. Mannor, and N. Shimkin, "Basis Function Adaptation in Temporal Difference Reinforcement Learning," *Annals of Operations Research*, vol. 134, no. 1, pp. 215-238, 2005.

[67] A. d. M. S. Barreto and C. W. Anderson, "Restricted Gradient-descent Algorithm for Value-function Approximation in Reinforcement Learning," *Artificial Intelligence*, vol. 172, no. 4-5, pp. 454-482, 2008.

[68] D. Choi and B. Van Roy, "A generalized Kalman filter for Fixed Point Approximation and Efficient Temporal-difference Learning," *Discrete Event Dynamic Systems*, vol. 16, no. 2, pp. 207-239, 2006.

[69] Y. Engel, "Algorithms and Representations for Reinforcement Learning," *Hebrew University of Jerusalem*, 2005.

[70] S. J. Bradtke and A. G. Barto, "Linear Least-squares Algorithms for Temporal Difference Learning," *Machine Learning*, vol. 22, no. 1-3, pp. 33-57, 1996.

[71] M. Geist and O. Pietquin, "Algorithmic Survey of Parametric Value Function Approximation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 6, pp. 845-867, 2013.

[72] M. Geist and O. Pietquin, "Kalman Temporal Differences," *Journal of Artificial Intelligence Research*, vol. 39, pp. 483-532, 2010.

[73] R. Mehra, "On the Identification of Variances and Adaptive Kalman Filtering," *IEEE Transactions on Automatic Control*, vol. 15, no. 2, pp. 175-184, 1970.

[74] D. G. Lainiotis, "Partitioning: A Unifying Framework for Adaptive Systems, i: Estimation," *Proceedings of the IEEE*, vol. 64, no. 8, pp. 1126-1143, 1976.

[75] A. Assa and K. N. Plataniotis, "Similarity-based Multiple Model Adaptive Estimation," *IEEE Access*, vol. 6, pp. 36 632-36 644, 2018.

[76] K. Doya, K. Samejima, K.-i. Katagiri, and M. Kawato, "Multiple Model-based Reinforcement Learning," *Neural Computation*, vol. 14, no. 6, pp. 1347-1369, 2002.

[77] T. Kitao, M. Shirai, and T. Miura, "Model Selection based on Kalman Temporal Differences Learning," *IEEE International Conference on Collaboration and Internet Computing (CIC)*, 2017, pp. 41-47.

[78] A. Mohammadi, and A. Asif, "Distributed Particle Filter Implementation With Intermittent/Irregular Consensus Convergence," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2572-2587, 2013.

[79] G. Kalantar, A. Mohammadi, and S.N. Sadrieh, "Analyzing the Effect of Bluetooth Low Energy (BLE) with Randomized MAC Addresses in IoT Applications," *IEEE International Conference on Internet of Things*, 2018.

[80] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.J Nordlund, "Particle Filters for Positioning, Navigation, and Tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp.425-437, 2005.

[81] Y.H. Ruan, P. Willett, and A. Marrs, "Fusion of Quantized Measurements via Particle Filtering," *Proceedings of the 2003 Aerospace Conference*, pp. 1967-1978, 2003.

[82] R. Karlsson, and F. Gustafsson, "Particle Filtering for Quantized Sensor Information," *European Signal Processing Conference*, 2005.

[83] R. S. Sutton, *et al.*, "Reinforcement Learning: An Introduction," *MIT Press*, 1998.

[84] M. Hutter, and S. Legg, "Temporal Difference Updating without a Learning Rate," *Advances in Neural Information Processing Systems*, 2008, pp. 705-712.

[85] R. S. Sutton, "Generalization in Reinforcement Learning: Successful Examples using Sparse Coarse Coding," *Advances in Neural Information Processing Systems*, 1996, pp. 1038-1044.

[86] W. Xia, C. Di, H. Guo, and S. Li, "Reinforcement Learning Based Stochastic Shortest Path Finding in Wireless Sensor Networks," *IEEE Access*, vol. 7, pp. 157807-157817, 2019.

[87] J. Li, T. Chai, F. L. Lewis, Z. Ding, and Y. Jiang, "Off-Policy Interleaved Q -Learning: Optimal Control for Affine Nonlinear Discrete-Time Systems," *IEEE Transactions on Neural Networks and Learning Systems,*, vol. 30, no. 5, pp. 1308-1320, May 2019.

[88] C. J. Watkins, and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279-292, 1992.

[89] Y. Ge, F. Zhu, X. Ling, and Q. Liu, "Safe Q-Learning Method Based on Constrained Markov Decision Processes," *IEEE Access*, vol. 7, pp. 165007-165017, 2019.

[90] A. Mohammadi, and K. N. Plataniotis, "Distributed Widely Linear Multiple-Model Adaptive Estimation," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 3, pp. 164-179, Sept. 2015.

[91] N. Shazeer, A., Mirhoseini, K., Maziarz, A. Davis, Q., Le, G. Hinton, and J. Dean, "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer," *ArXiv preprint*, 2017.

[92] A. Al-Tamimi, M. Abu-Khalaf, and F. L. Lewis, "Model-Free Q- Learning Designs for Discrete-Time Zero-Sum Games with ApplicationtoH-InfinityControl," *Automatica*, vol. 43, pp. 473-481, 2007.

[93] M. Z. Win, F. Meyer, Z. Liu, W. Dai, S. Bartoletti, and A. Conti "Efficient Multi-sensor Localization for the Internet of Things: Exploring a New Class of Scalable Localization Algorithms," *IEEE Signal Process. Magazine*, vol. 35, no. 5, 2018.