YAŞAR UNIVERSITY

GRADUATE SCHOOL

MASTER'S THESIS

MACHINE LEARNING BASED ENERGY-EFFICIENT INDOOR POSITIONING FOR MOBILE INTERNET OF THINGS

ALPER SAYLAM

THESIS ADVISOR: PROF. DR. VOLKAN RODOPLU
CO-ADVISOR: PROF. DR. CÜNEYT GÜZELİŞ

ELECTRICAL AND ELECTRONICS ENGINEERING

PRESENTATION DATE: 28.07.2022

BORNOVA / İZMİR

JULY 2022

We certify that, as the jury, we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Jury Members:	**Signature:**
Prof. (PhD) Volkan RODOPLU Yaşar University	.....
Prof. (PhD) Cüneyt GÜZELİŞ Yaşar University	.....
Prof. (PhD) Mustafa SEÇMEN Yaşar University	.....
Prof. (PhD) Barış ATAKAN Izmir Institute of Technology	.....
Prof. (PhD) Ayşegül UÇAR Fırat University	.....

Prof. (PhD) Yücel ÖZTÜRKOĞLU

Director of the Graduate School

ABSTRACT

MACHINE LEARNING BASED ENERGY-EFFICIENT INDOOR POSITIONING FOR MOBILE INTERNET OF THINGS

Saylam, Alper
MSc, Electrical and Electronics Engineering
Advisor: Prof. Dr. Volkan Rodoplu
Co-Advisor: Prof. Dr. Cüneyt Güzeliş
July 2022

Artificial Intelligence is a promising solution to indoor positioning and tracking systems in overcoming the challenges of positioning accuracy and energy consumption of mobile Internet of Things (IoT) devices. The majority of the past works in the positioning literature have focused on enhancing the positioning accuracy and solving the problem of transmit energy consumption via reactive approaches. In contrast to these past approaches, our approach is to forecast the future trajectory of a mobile IoT device and determine the positioning interval based on these trajectory forecasts. Our approach aims to reduce the transmit energy consumption of a mobile IoT device in indoor positioning and tracking systems.

In this thesis, first, we develop an algorithm called "Dynamic Positioning Interval based on Reciprocal Forecasting Error (DPI-RFE)" in order to achieve energy-efficient indoor positioning. In contrast with existing indoor positioning algorithms, DPI-RFE adapts the positioning interval based on the reciprocal instantaneous forecasting error, thereby dynamically trading off transmit energy consumption against forecasting error. We compare the performance of DPI-RFE with respect to total transmit energy consumption and average forecasting error against those of the Constant Positioning Interval (CPI) and Positioning Interval based on Displacement (PID) algorithms. Our results show that DPI-RFE significantly outperforms both of these benchmark algorithms with respect to transmit energy consumption while achieving a competitive average forecasting error performance.

Second, in order to improve energy efficiency further for mobile IoT devices, we develop a novel architecture called "Machine Learning Enabled Sleep Time Estimation (MLE-STE)". Our MLE-STE architecture forecasts the trajectory of the mobile device and estimates the maximum allowable sleep time of the mobile device with respect to forecast positions subject to a target maximum forecasting error. We compare the performance of our MLE-STE architecture against those of PID and our DPI-RFE

algorithms with respect to transmit energy consumption and forecasting error. Our results show that the MLE-STE architecture outperforms both PID and DPI-RFE.

This thesis paves the way to the development of machine-learning-based indoor positioning and tracking systems that achieve high energy efficiency for mobile IoT devices.

Keywords: Indoor Positioning and Tracking, mobile Internet of Things (IoT), trajectory forecasting, Artificial Intelligence (AI), Machine Learning (ML), energy efficiency

ÖZ

MOBİL NESNELERİN İNTERNETİ İÇİN MAKİNE ÖĞRENİMİNE DAYALI ENERJİ VERİMLİ İÇ MEKANDA KONUMLANDIRMA

Saylam, Alper
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği
Danışman: Prof. Dr. Volkan Rodoplu
Yardımcı Danışman: Prof. Dr. Cüneyt Güzeliş
Temmuz 2022

Yapay Zekâ, mobil Nesnelerin İnterneti (IoT) cihazlarının konumlandırma doğruluğu ve enerji tüketimi ile ilgili zorlukların üstesinden gelmede iç mekan konumlandırma ve izleme sistemleri için umut verici bir çözümdür. Konumlandırma literatüründeki geçmiş çalışmaların çoğu, konumlandırma doğruluğunu artırmaya ve reaktif yaklaşımlar yoluyla iletim enerji tüketimi sorununu çözmeye odaklanmıştır. Bu geçmiş yaklaşımların aksine, yaklaşımımız bir mobil IoT cihazının gelecekteki yörüngesini tahmin etmek ve bu yörünge tahminlerine dayalı olarak konumlandırma aralığını belirlemektir. Yaklaşımımız, iç mekan konumlandırma ve izleme sistemlerinde bir mobil IoT cihazının iletim enerji tüketimini azaltmayı amaçlamaktadır.

Bu tezde ilk olarak, enerji verimli iç mekan konumlandırma elde etmek için "Karşılıklı Tahmin Hatasına Dayalı Dinamik Konumlandırma Aralığı (DPI-RFE)" adlı bir algoritma geliştirilmiştir. Mevcut iç mekan konumlandırma algoritmalarının aksine, DPI-RFE, karşılıklı anlık tahmin hatasına dayalı olarak konumlandırma aralığını uyarlar, böylece iletim enerji tüketimini tahmin hatasına karşı dinamik olarak değiştirir. Toplam iletim enerji tüketimi ve ortalama tahmin hatası açısından DPI-RFE'nin performansı Sabit Konumlandırma Aralığı (CPI) ve Yer Değiştirme tabanlı Konumlandırma Aralığı (PID) ile karşılaştırılmaktadır. Sonuçlarımız, rekabetçi bir ortalama tahmin hatası performansı elde ederken, DPI-RFE'nin iletim enerji tüketimi açısından bu kıyaslama algoritmalarının her ikisinden de önemli ölçüde daha iyi performansa sahip olduğunu göstermektedir.

İkincisi, mobil IoT cihazları için enerji verimliliğini daha da artırmak için "Makine Öğrenimi Etkinleştirilmiş Uyku Süresi Tahmini (MLE-STE)" adlı yeni bir mimari geliştirilmiştir. MLE-STE mimarimiz, mobil cihazın yörüngesini tahmin eder ve hedef maksimum tahmin hatasına tabi olan tahmin pozisyonlarına göre mobil cihazın izin verilen maksimum uyku süresini tahmin eder. MLE-STE mimarimizin performansı, PID ve DPI-RFE algoritmalarının performansıyla, iletim enerji tüketimi ve tahmin

hatası açısından karşılaştırılmaktadır. Sonuçlarımız, MLE-STE mimarisinin hem PID hem de DPI-RFE'den daha iyi performansa verdiğini göstermektedir.

Bu tez, mobil IoT cihazları için yüksek enerji verimliliği sağlayan makine öğrenimi tabanlı iç mekan konumlandırma ve izleme sistemlerinin geliştirilmesine giden yolu açmaktadır.

Anahtar Kelimeler: İç Mekan Konumlandırma ve İzleme, mobil Nesnelerin İnterneti (IoT), yörünge tahmini, Yapay Zeka (AI), Makine Öğrenimi, enerji verimliliği

ACKNOWLEDGEMENTS

Throughout my MSc thesis program, I have been encouraged and supported by many people who are my advisors, teammates, and my family. First, I would like to thank my advisor Prof. Dr. Volkan Rodoplu for his endless support and advice. I have learned from him how to think critically and how to manage long-term projects. Second, I would like to thank my co-advisor Prof. Dr. Cüneyt Güzeliş for his critical contributions and direction throughout the course of my thesis. Third, I would like to thank my teammates Rıfat Orhan Çıkmazel, Nur Keleşoğlu, and Mert Nakıp for their precious friendship and help.

Finally, I would like to thank my family for their limitless support and grateful patience during this journey.

Alper SAYLAM
İzmir, 2022

TEXT OF OATH

I declare and honestly confirm that my study, titled "MACHINE LEARNING BASED ENERGY-EFFICIENT INDOOR POSITIONING FOR MOBILE INTERNET OF THINGS" and presented as a Master's Thesis, has been written without applying to any assistance inconsistent with scientific ethics and traditions. I declare, to the best of my knowledge and belief, that all content and ideas drawn directly or indirectly from external sources are indicated in the text and listed in the list of references.

ALPER SAYLAM

Signature:

_____

July 28, 2022

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SYMBOLS AND ABBREVIATIONS

ABBREVIATIONS:

IoT	Internet of Things
ML	Machine Learning
AI	Artificial Intelligence
IP	Indoor Positioning
Wi-Fi	Wireless Fidelity
IPT	Indoor Positioning and Tracking
PID	Positioning Interval based on Displacement
CPI	Constant Positioning Interval
DPI-RFE	Dynamic Positioning Interval based on Reciprocal Forecasting Error
PS	Position Selection
FEC	Forecasting Error Calculator
PIU	Positioning Interval Update
MLE-STE	Machine Learning Enabled Sleep Time Estimation
STE	Sleep Time Estimator
BoSN	Bank of Sigmoid Neurons
TF	Trajectory Forecaster
AoA	Angle of Arrival
RSSI	Received Signal Strength Indicator
TDOA	Time Difference of Arrival
TOF	Time of Flight
MLP	Multi-layer Perceptron
LSTM	Long Short Term Memory
Adam	Adaptive Moment Estimation
MSE	Mean Square Error

Continued on the next page

ABBREVIATIONS:

ReLU	Rectified Linear Unit Function

SYMBOLS:

G	Gateway
D	Mobile IoT device
R	Deployment region
$\mathscr{M}$	Set of anchors
B	Number of beacons transmitted by the device
T	Duration between successive beacons transmitted by device
T_s	Sleep time
T_c	Positioning interval
E	Transmit energy consumption of the mobile IoT device
k	Discrete time index
$i_x[k]$	Row index of the cell that the position estimate of the device takes place at time instant k
$i_y[k]$	Column index of the cell that the position estimate of the device takes place at time instant k
$\mathbf{x}[k]$	2D vector, composed of the x and y coordinates of the position estimate of the mobile device in slot k
U	Past window
V	Forecast window
$\mathbf{c}[k]$	Estimate cell position of the device at time k
$\hat{\mathbf{c}}^T[k+v]$	v^{th} step ahead forecast cell position, where k is the discrete-time index
F^x	Forecaster that forecasts the row index of the cell positions
F^y	Forecaster that forecasts the column index of the cell positions
$e_{\max}$	Threshold for the forecasting error of the Trajectory Forecaster
γ	Threshold for the displacement in forecast cell positions of the device

CHAPTER 1
INTRODUCTION

The goal of this thesis is the design of machine-learning-based approaches that aim to reduce the transmit energy consumption of mobile IoT devices in indoor positioning and tracking systems. In particular, we develop the "Dynamic Positioning Interval based on Reciprocal Forecasting Error (DPI-RFE)" algorithm and the "Machine Learning Enabled Sleep Time Estimation (MLE-STE)" architecture for energy-efficient indoor positioning and tracking systems.

In this chapter, we discuss the problem addressed in this thesis and the technical background that will be utilized in the following chapters. First, we discuss the importance of indoor positioning and tracking systems for IoT applications. In addition, we review the positioning techniques and present positioning and tracking applications. Second, we review the indoor positioning and tracking system, its challenges, and describe the energy consumption of mobile devices in indoor positioning and tracking system. Third, we review the rudiments of Machine Learning (ML) and provide the context for this work. Fourth, we present the relationship between our study and the state of the art. Finally, we give an outline for the rest of the thesis.

1.1 Positioning Techniques and Mobile Internet of Things

The development of Industry 4.0 exposes useful Internet of Things (IoT) applications in distinct sectors such as logistics, healthcare, energy, and transportation as well. The IoT applications improve the industrial processes, enable fast and effective ways, provide low-cost productions, and provide human safety as well (Fantana et al., 2013). An IoT application consists of numerous static, portable, and mobile IoT devices in order to satisfy the expectations of the industry. Since numerous IoT devices (particularly mobile devices) take part in the industry, the positioning and tracking of those devices become crucial. (Ramakrishnan, Gaur, & Singh, 2016).

Nowadays, the need for positioning and tracking of mobile IoT devices both indoors and outdoors via wireless signal is growing rapidly. The wireless positioning techniques are categorized into two groups: (1) self-positioning, (2) remote positioning. Remote positioning refers to the computation of the position of the device by a central computing entity. In contrast, self-positioning refers to the case in which the position of the mobile

device is computed by a device located on the mobile device (Svalastog, 2007).

Self-positioning includes three main positioning techniques, which are GPS (Global Positioning Techniques), Indoor GPS, and Mobile Terminal Positioning over Satellite UMTS (S-UMTS). In a GPS system, the satellites transmit signals to a mobile device, which computes its position based on those signals. Thus, the mobile device determines its 3D position (latitude, longitude, and altitude). Similarly, the same process is applied in indoor GPS, but positions are not as accurate as outdoors. In contrast to GPS, S-UMTS utilizes only two satellites in order to obtain the mobile device position (Zeimpekis, Giaglis, & Lekakos, 2002).

Remote positioning, which is also utilized for the scope of this thesis, consists of four techniques (Zeimpekis et al., 2002). The first technique is Cell Identification, which provides the approximate position of the mobile device by identifying in which cell the device is located at a given time. The second technique is Angle of Arrival (AoA), which determines the device position by considering the intersection of the two straight lines produced by the distinct signals sent from the mobile device to two distinct receivers (Niculescu & Nath, 2003). The third technique is Time of Arrival (ToA), which obtains the position of the mobile device by observing the flight time of the signal sent from the mobile device to the receiver (Shin & Sung, 2002). The fourth technique is Time Difference of Arrival (TDoA), which computes the position of the mobile device based on the intersection region of the TDoA measurements each of which provides a hyperbolic locus (Y. Wang & Ho, 2016).

1.1.1 Positioning and Tracking Applications

The positioning and tracking of the IoT devices are expected to be in the industry and the daily life of human beings in the near future. In addition, the positioning applications are categorized into two groups regarding the application environments (e.g., indoor and outdoor). Table 1.1 shows the applications and their operation environments (Zeimpekis et al., 2002). (Piras, Marucco, & Charqane, 2010) states that while the outdoor positioning systems guarantee high performance, indoor positioning applications do not guarantee high performance (hence, indoor positioning is more challenging than outdoor). The positioning and tracking of a mobile IoT device in an indoor environment is the main problem that this thesis addresses.

1.2 Indoor Positioning and Tracking System

The indoor positioning and tracking system is comprised of three main components, which are the transmitter, the receiver, and the central computing entity (Svalastog,

Table 1.1. Positioning and tracking applications and their environments

Application	Application Environment
Automotive assistance	Outdoor
Travel service	Outdoor
Product tracking	Outdoor/Indoor
Traffic management	Outdoor
Field personnel support	Outdoor/Indoor
Vehicle tracking	Outdoor
People tracking	Outdoor/Indoor
Fleet Management	Outdoor

2007). Fig. 1.1 shows an illustration of an indoor positioning and tracking system. In the system, the mobile IoT device (shortly, device), which is located on the moving object whose position is computed, operates as a transmitter. Note that the device is battery limited. Each anchor, which is a static device, represents a receiver, and the gateway operates as a central computing entity. In the system, the device first transmits signals, which can be Wi-Fi, Bluetooth, and Zigbee, to the anchors (Bai, Ciravegna, Bond, & Mulvenna, 2020). In Fig. 1.1, dashed lines represent the two distinct transmitted signals that are sent from the device to the two distinct anchors.

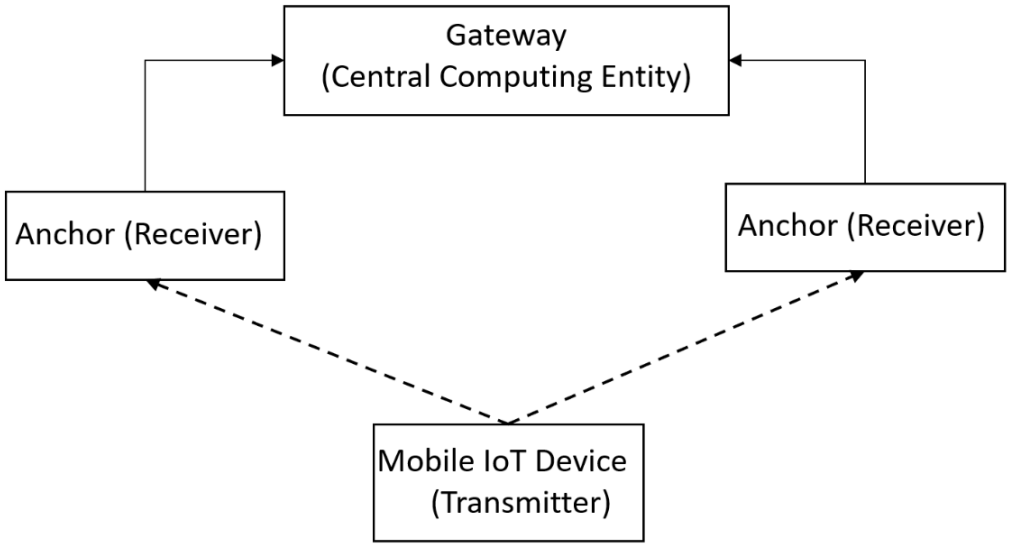

Figure 1.1. Indoor positioning system components

Each anchor computes the positioning indicator such as AoA, TDoA, and ToA based on the received signal (Svalastog, 2007). As shown in Fig. 1.1, each anchor sends the computed position indicator to the gateway. The gateway then determines the position of the mobile device based on the position indicator produced by the anchors. The

DPI-RFE algorithm that we develop in Chapter 2 and the MLE-STE architecture that we develop in Chapter 3 are designed to reside in the central computing entity.

1.2.1 Indoor Positioning and Tracking System Challenges

Indoor positioning is particularly challenging due to the fact that it is highly affected by the unpredictable wireless signal characteristics (Nessa, Adhikari, Hussain, & Fernando, 2020). These characteristics cause limited accuracy, reliability, and robustness in indoor positioning and tracking systems. Moreover, the transmitting signal from the device to the anchor is affected by the following conditions: change of indoor propagation channel due to device motion, the density of the obstacles, attenuation of the transmitted signals, and multipath (Mainetti, Patrono, & Sergi, 2014).

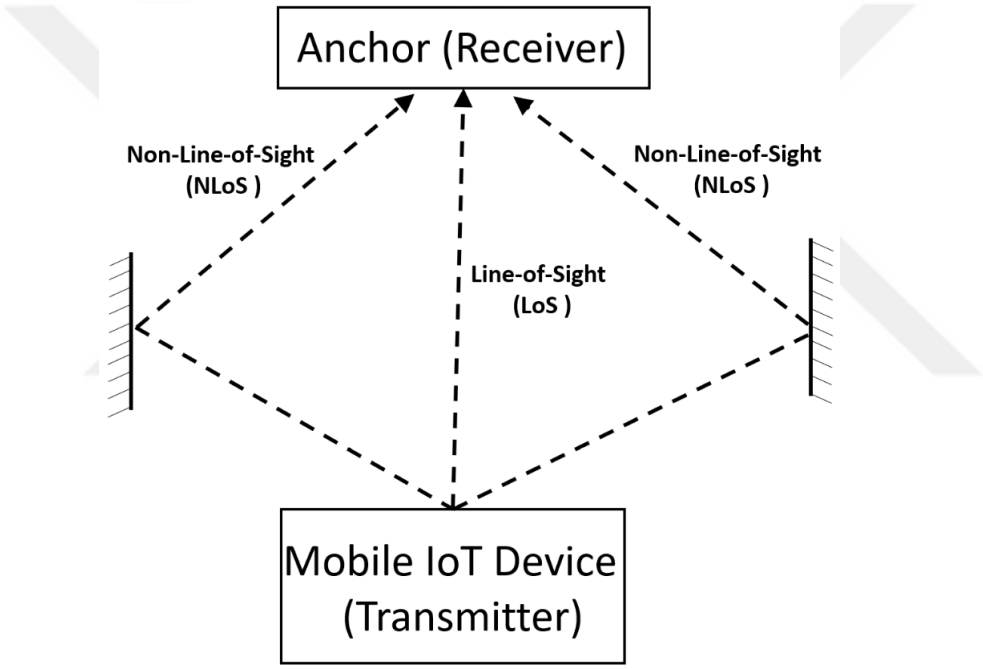

Figure 1.2. Line-of-Sight and Non-Line-of-Sight signals

In an indoor positioning system, there occur distinct paths from the transmitter to the receiver due to the characteristics of wireless communication. The direct path from transmitter to receiver is called the Line of Sight (LoS) path. A path that is produced by reflection from a surface or penetration of an obstacle, which is a penetrable object, is called a Non-Line-of-Sight (NLoS) path.

Fig. 1.2 shows a LoS path and two NLoS paths. Since multiple paths from the transmitter to the receiver exist, there is a multipath profile at the receiver, which reduces the accuracy, the robustness, and the reliability of indoor positioning and tracking systems (García, Poudereux, Hernández, Ureña, & Gualda, 2015). In this thesis, we focus entirely on the end result of the position obtained from positioning indicator, which

already incorporate the effects of multipath. We utilize the past positions obtained in this manner to forecast the future trajectory of the device and focus on reducing the transmit energy consumption while taking into account the forecasting error. Thus, in this thesis, we do not target improvements in obtaining position estimates themselves but rather focus on how such position estimates can be used to forecast future positions. Such forecasts are then used to put the device to sleep over durations for which we expect to achieve accurate forecasts.

1.2.2 Energy Consumption of a Mobile IoT Device in Indoor Positioning and Tracking

The mobile device in an indoor positioning and tracking system is powered by a battery, which has limited capacity (Bisio, Lavagetto, Marchese, & Sciarrone, 2016). Furthermore the device consumes energy for the following operations: scanning the environment to connect to the anchor, and to periodically transmit signal in order to indicate its position. The "positioning interval" refers to the duration between the two successive transmissions that contain positioning indicators. The works in the positioning literature, which focus on the energy consumption of the mobile device, have introduced solutions that reduce the energy consumption by utilizing energy-efficient technology or energy-efficient hardware for the mobile device. Note that Wi-Fi-based positioning systems are much more energy-consuming than Bluetooth Low Energy (BLE) based systems (Kárník & Streit, 2016).

Figure 1.3. Illustration of the signals at different positioning intervals

In Fig. 1.3, we present three distinct scenarios for the positioning interval with which the device transmits. In this figure, each "D" represents the signal transmitted by the mobile device, and each slot represents a time slot. In the first scenario, the device transmits the signal at a period of two time slots, which is the most frequent advertising scenario among these scenarios. In the second scenario, the device transmits signals at a period of three time slots. In the third scenario, the device transmits signals at a period of four time slots. In the first scenario, since the device transmits the signal the most frequently, the device consumes the most transmit energy among all of the scenarios. In the third scenario, since the device transmits signals the least frequently, it consumes

the least transmit energy. This illustration shows that a larger positioning interval results in higher energy efficiency for the mobile device.

1.3 Basic Definitions in Machine Learning and the Context of our Work

ML models have rapidly grown in IoT applications due to their characteristics such as being accurate, fast, and reliable (Tahsien, Karimipour, & Spachos, 2020). An ML model is utilized to find the pattern or to make predictions based on a given data set. The ML model consists of two stages, which are the training and test stages, that are used in order to determine the values of the model parameters such as the weights and biases of the neurons. In the training state, the ML model aims to compute the values of the model parameters over the data set called the "training set". In the test stage, the trained model is tested over the set called the "test set" to observe the accuracy of the trained model.

In this thesis, we develop the DPI-RFE algorithm and MLE-STE architecture, each of which is ML-enabled to forecast the trajectory of a mobile IoT device in an indoor positioning and tracking system and to reduce the transmit energy consumption problem of the mobile device.

1.3.1 Trajectory Forecast of a Mobile IoT Device

One of the operations in our DPI-RFE and MLE-STE is the forecasting of the future trajectory of the mobile IoT device. Throughout this thesis, we consider only the 2D positions (x and y coordinates) of the mobile device. The ML model estimates the future states based on the past states of the data. Thus, in order to estimate the future trajectory of the mobile device, we take into account the past trajectory of the mobile device.

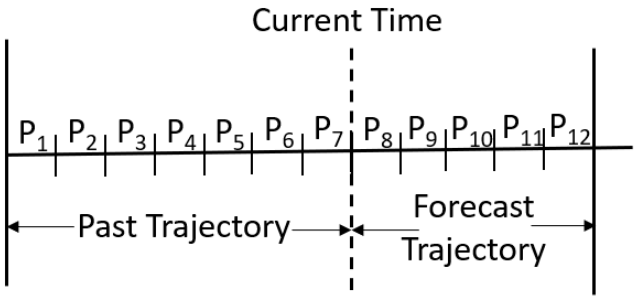

Figure 1.4. Illustration of trajectory forecasting

Fig. 1.4 shows an illustration of trajectory forecasting. In this figure, each "P" represents a 2D position of the mobile device, and the dashed vertical bar represents the current time. The positions that occur before the current time indicate the past trajectory, and

the positions that occur after the current time indicate the forecast trajectory produced by the ML model based on this past trajectory. A key performance metric of trajectory forecasting is the forecasting error, namely the difference between the actual position and the forecast position over the course of the trajectory.

1.4 Relationship to the State of The Art

In this section, we present the relationship between our algorithm and the state of the art. We categorize the works in the indoor positioning literature into three groups: (1) the works that focus only on the trajectory forecasting of the mobile device based on ML models; (2) the works that develop motion-triggered algorithms that dynamically adjust the beacon interval of the mobile device; and (3) the works that forecast the trajectory of the device and provide adaptive solutions in order to reduce the energy consumption of the mobile device.

In the first category, we address the ML enabled algorithms that forecast the future trajectory of a mobile IoT device by using its past trajectory. The majority of the articles on trajectory forecasting in the literature have considered trajectory forecasting of mobile objects (e.g., pedestrians, phones, traffic agents) in order to avoid collisions and to provide efficient navigation in various applications such as autonomous vehicles and social robots (Alahi et al., 2016; Huang, Bi, Li, Mao, & Wang, 2019; Ma et al., 2019). Recent works (Xue, Huynh, & Reynolds, 2018; Pellegrini, Ess, Schindler, & Van Gool, 2009; Vu, Le, & Lee, 2014; H. Liu et al., 2019) have demonstrated that the ML model that forecasts the future trajectory by considering the past trajectory provides promising results. In (C. Wang, Ma, Li, Durrani, & Zhang, 2019), human trajectory forecasting was utilized in order to efficiently allocate the resources for the 5G applications. In contrast to those past works, we forecast the future trajectory of a mobile device in order to reduce its energy consumption.

In the second category, we contrast our work against those works that adjust the sampling interval of the mobile devices that are employed in indoor positioning systems in a motion-triggered manner. In (Zhang, Liu, & Jiang, 2012; X. Liu, Zhan, & Cen, 2018; González, Morillo, Álvarez-García, Ros, & Ruiz, 2019; Abdellatif, Mtibaa, Harras, & Youssef, 2013; Kuxdorf-Alkirata, Spathmann, Koch, & Bruckmann, 2018), the sampling rate of the mobile device is dynamically adjusted by considering the information from an accelerometer that is located on the mobile device and that triggers the system when it detects the motion of device. Similarly, Reference (Shafer & Chang, 2010) develops an algorithm that computes the indoor position only when the accelerometer detects the mobile device's movement. In (Kjærgaard, Langdal, Godsk, & Toftkjær, 2009), an algorithm called EnTracked optimizes the position updates by regarding the

system conditions and mobility of the device. In contrast, our DPI-RFE and MLE-STE dynamically adjust the positioning interval of the mobile device in an adaptive manner.

In the third category, we compare our works against adaptive solutions that aim to reduce the energy consumption of the mobile device in an indoor positioning system. Reference (Constandache, Gaonkar, Sayler, Choudhury, & Cox, 2009) has introduced an energy-efficient algorithm called EnLoc for positioning applications. The EnLoc algorithm first forecasts the mobility patterns of a mobile device and determines the optimal positioning accuracy for a given energy budget. Reference (Chon, Talipov, Shin, & Cha, 2011) has developed an algorithm that estimates the regularity of the mobile device's mobility and forecasts its residence time at a place in order to schedule the beacon interval of a mobile device. In (Saylam, Cikmazel, Kelesoglu, Nakip, & Rodoplu, 2021), an adaptive algorithm called Positioning Interval based on Displacement (PID) adjusts the positioning interval of a mobile device based on when the device is forecast to change its cell. In contrast to the adaptive algorithms in that category, While DPI-RFE determines the positioning interval reciprocal to the forecasting error in an adaptive manner, MLE-STE architecture adaptively determines the positioning interval for given conditions in order to reduce the energy consumption of the mobile device.

1.5 Contributions of this Thesis

The first contribution of this thesis is the development of a novel algorithm, which we call DPI-RFE for energy-efficient indoor positioning and tracking system. Our algorithm uses ML in order to forecast the future trajectory of a mobile IoT device and adapts the positioning interval of the device in a manner that is reciprocal to the instantaneous forecasting error.

The second contribution of this thesis is the development of an architecture, which we call MLE-STE in an energy-efficient indoor positioning and tracking system for mobile IoT device. Our MLE-STE architecture first forecasts the trajectory of the mobile IoT device and then utilizes the ML models in order to estimate a sleep time for the mobile device based on the forecast trajectory. Our architecture considers both forecasting error and device displacement, in estimating sleep time. Thereby, our architecture significantly reduces the transmit energy consumption of the mobile device by adapting the sleep time and keeps the forecasting error below the threshold as well.

1.6 Outline of this Thesis

The rest of the thesis is organized as follows: In Chapter 2, we propose an algorithm called DPI-RFE, which runs at gateway $\mathscr{G}$, dynamically updates the positioning interval

of the mobile device based on its trajectory forecasts. Moreover, the algorithm reduces the energy consumption of the mobile device by adjusting its positioning interval. In addition, we also present the performance of the DPI-RFE algorithm against the benchmarks with respect to the total transmit energy consumption, average forecasting error and positioning interval.

In Chapter 3, we develop an architecture called MLE-STE, which is implemented in $\mathscr{G}$, which consists of a Trajectory Forecaster, an Accumulator, and a Sleep Time Estimator (STE) module in order to enhance the energy efficiency of the mobile device. The advantage of this architecture is that it determines a sleep time that reduces energy consumption and satisfies a given forecasting error. In addition, we also measure the performance of the MLE-STE architecture in terms of transmitting energy consumption and trajectory forecasting error.

In Chapter 4, we present our conclusions and discuss directions for our future work.

CHAPTER 2

DYNAMIC POSITIONING INTERVAL BASED ON RECIPROCAL FORECASTING ERROR (DPI-RFE) ALGORITHM FOR ENERGY-EFFICIENT MOBILE IOT INDOOR POSITIONING

2.1 Introduction

As we have discussed in Chapter 1, Indoor Positioning (IP) is expected to play a key role in next-generation wireless networks in order to enable a plethora of services such as navigation, proximity marketing, asset tracking as well as social distancing (Alrashidi, 2020; Duque Domingo, Cerrada, Valero, & Cerrada, 2017). While providing accurate IP is the main goal of these networks, it is equally important that such accuracy is provided without depleting the scarce battery resources of the mobile devices whose positions are being determined (Constandache et al., 2009). In particular, trading off positioning accuracy against energy consumption is crucial for mobile IoT devices, which constitute a rapidly growing segment of the Internet (Ericsson, Jun. 2021). AI is increasingly playing an important role in the design of wireless networks (Nguyen et al., 2020). Recent work has also applied ML techniques to IP (Alahi et al., 2016; C. Wang et al., 2019; Huang et al., 2019; Xue et al., 2018; Cakan, Şahin, Nakip, & Rodoplu, 2021).

The main contribution of this chapter[1] is the design of a novel algorithm called DPI-RFE which dynamically adapts the positioning interval of the device in a manner that is reciprocal to the instantaneous forecasting error.

Our DPI-RFE algorithm is distinct from existing algorithms that (1) forecast the future trajectory of a mobile device without any regard to energy efficiency (Alahi et al., 2016; C. Wang et al., 2019; Huang et al., 2019; Xue et al., 2018; Ma et al., 2019; Pellegrini et al., 2009), or (2) adjust the positioning interval of a mobile device based on motion detection by the device (i.e. motion-triggered energy-efficient IP) (Chon et al., 2011; Zhang et al., 2012; X. Liu et al., 2018; Yin, Wu, Yang, & Liu, 2017; González et al., 2019), or (3) forecast future trajectory of a mobile device in order to achieve energy-efficient IP (Saylam, Cikmazel, et al., 2021; Constandache et al., 2009). Our results show that with regard to total energy consumption of the mobile IoT

[1] The technical content in this chapter has been published as a conference paper (Saylam, Kelesoglu, Cikmazel, Nakip, & Rodoplu, 2021) at the International Conference on Computer, Information and Telecommunication Systems (CITS) 2021.

device, DPI-RFE significantly outperforms both the Constant Positioning Interval (CPI) algorithm, which serves as a benchmark, and the PID algorithm (Saylam, Cikmazel, et al., 2021), which adapts the sleep duration in response to a significant forecast change in position. Furthermore, DPI-RFE achieves an average forecasting error that is close to that achieved by PID.

The rest of this chapter is organized as follows: In Section 2.2, we state our assumptions and basic system design. In Section 2.3, we describe our DPI-RFE algorithm. In Section 2.4, we discuss our results. In Section 2.5, we summarize this chapter.

2.2 Assumptions and Basic System Design

Throughout this work, we shall focus on a particular mobile IoT device, denoted by D, that roams an indoor deployment region, which we denote by $\mathscr{R}$. Furthermore, we observe this device D over an interval $\mathscr{T}$ in time. We assume that there is a set $\mathscr{M}$ of positioning anchors (or "anchors" for short) with which D is associated over $\mathscr{T}$.

We assume that device D has the ability to send a beacon signal that is heard successfully by each anchor in $\mathscr{M}$. We assume that the anchors in $\mathscr{M}$ are connected to a gateway G. Based on the beacon signals received by each of the anchors in $\mathscr{M}$, gateway G estimates the current position of device D.[2]

Device D wakes up and remains awake for a duration of T seconds, during which it sends L positioning beacons at regular intervals of T_b seconds. Thus, $T = LT_b$. Then, the device goes to sleep for T_s seconds. In our design, T is fixed, whereas T_s is potentially variable in each cycle. We define the "positioning interval", denoted by T_c, as the duration between two successive wake-ups of the device. Thus, $T_c = T + T_s$. Furthermore, in our design, T_s as an integer multiple of T.

We divide up the time axis into slots of duration T. We let k denote the discrete-time index of a time slot. For every k, we let $\mathbf{x}[k]$ denote the 2D vector, composed of the x and y coordinates of the position estimate of the mobile device in slot k. This estimate is based on the L beacons received by the anchors during slot k, if the device is awake in slot k. In particular, no such position estimate is formed at G during a slot if the device D sleeps in that slot.[3]

[2] For example, if Angle of Arrival (AoA) is used as the underlying positioning technology, each anchor records the AoA measurements based on the beacon signal received from D. The gateway G combines these AoA measurements in order to compute the current position of D.

[3] Thus, forecasting is needed for each slot in which no position estimate is available at G.

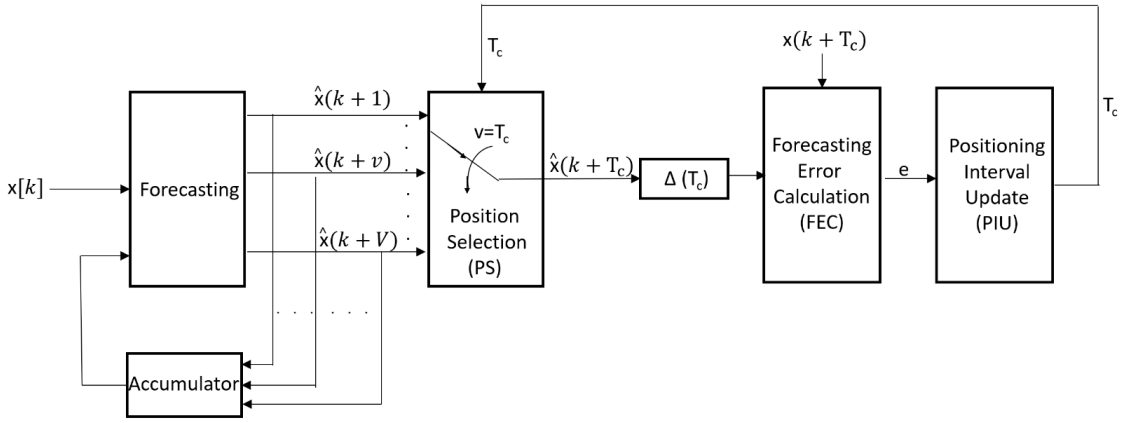

Figure 2.1. Architectural description of the Dynamic Positioning Interval based on Reciprocal Forecasting Error (DPI-RFE) algorithm

2.3 Dynamic Positioning Interval Based On Reciprocal Forecasting Error (DPI-RFE) Algorithm

Fig. 2.1 shows the architecture that implements our DPI-RFE algorithm. We note that this architecture resides at gateway G. The architecture is comprised of four modules: Forecasting; Position Selection (PS); Forecasting Error Calculation (FEC); and Positioning Interval Update (PIU). Below, we describe each of these modules.

First, the Forecasting module in Fig. 2.1 takes as input the past position estimates formed when the device was awake as well as the past forecast positions for those slots at which the device was not awake. (The Accumulator in the figure accumulates the past forecasts. The Forecasting module uses these during those slots at each of which a past position estimate is not available since the device was not awake in that slot.) The Forecasting module takes this superposition of position estimates and position forecasts for a total of U slots into the past and forms position forecasts V slots into the future.[4] (Recall that each slot is of duration T seconds.) The output of the Forecasting module is the vector of forecast positions, denoted by $\hat{\mathbf{x}}(k+v)_{v\in\{1,\ldots V\}}$.[5]

Second, the PS module selects a position among the forecast positions based on the value of the positioning interval T_c that is fed into this module. This selected position is $\hat{\mathbf{x}}(k+T_c)$.

Third, this forecast position $\hat{\mathbf{x}}(k+T_c)$ is input from the PS to the FEC. Recall that device D sends a sequence of L beacons in a slot of duration T when it is awake. The FEC module takes the position estimate of D denoted by $\mathbf{x}[k+T_c]$ and the selected forecast position $\hat{\mathbf{x}}(k+T_c)$ and computes the Euclidean distance between these two positions in

[4]Note that past forecasts are used in all slots for which no position estimates are available in order to form forecasts of future positions.

[5]The argument that appears in the parentheses is the time index *for* which the forecast is formed.

order to obtain the instantaneous forecasting error denoted by e in Fig. 2.1.

Fourth, the PIU module takes the instantaneous forecasting error as an input and determines the next positioning interval, namely T_c, based on this forecasting error. After the positioning interval has been determined, PIU passes to the PS the value of T_c as shown in Fig. 2.1.

Our DPI-RFE algorithm performs a particular selection of T_c based on the forecasting error e, which we now describe. We shall denote the nth positioning interval by $T_c^{(n)}$ and the nth forecasting error by $e^{(n)}$. The DPI-RFE algorithm computes the next positioning interval denoted by $T_c^{(n+1)}$ based on the current positioning interval $T_c^{(n)}$ and the nth forecasting error $e^{(n)}$ as $T_c^{(n+1)} = T_c^{(n)}/e^{(n)}$ provided that this value does not fall below T and does not rise above VT.[6] (DPI-RFE sets $T_c^{(n+1)}$ to the lower or the upper bound, respectively, if $T_c^{(n)}/e^{(n)}$ hits any one of these limits.) That is, the value of the next positioning interval is chosen to be that of the current positioning interval times the reciprocal of the current forecasting error.[7]

2.4 Results

2.4.1 Simulation Setup

In this section, first, we describe our positioning data set. Second, we explain our forecasting methodology for the DPI-RFE algorithm. Third, we describe the benchmark algorithms against which we compare the performance of our algorithm.

2.4.1.1 Description of the Positioning Data Set

The data set (*Positioning Dataset*, 2019) used in this work consists of kinematically collected positions of a human moving in a rectangular deployment region that has a length of 70 m and a width of 35 m. We scaled the data set to a deployment region that has size 4×4 m. In the data set, there are two features, namely the x and y coordinates of the human, who carries the mobile device.[8]

2.4.1.2 Forecasting Methodology

The duration for which the device remains awake, namely T, is set to 1 s. We use a Multi-layer Perceptron (MLP) (Haykin & Haykin, 2001) forecasting scheme in the

[6] The former case corresponds to zero sleep duration; the latter case is the maximum duration for which forecasts into the future are available.

[7] In our future work, we shall explore alternative choices in the functional form that characterizes the dependence of the positioning interval on the forecasting error.

[8] In the data set, there is a total of 5718 samples for training and 1907 samples for testing.

DPI-RFE algorithm. The number of layers and the number of neurons in each layer of the MLP model are optimized in order to minimize Mean Square Error (MSE).

The Forecasting module has a distinct MLP model in order to forecast the future positions on each of the x and y axes. The number of past samples, namely U, is set to 20, and the number of future forecasts, namely V, is set to 10. Each MLP model is comprised of a single hidden layer with 20 neurons and an output layer with 10 neurons. Rectified Linear Unit Function (ReLU) is utilized at both layers. In the training stage, 10-fold cross-validation is applied. For each fold, the number of epochs is 250 and the batch size is 20.

2.4.1.3 Benchmark Algorithms

We examine the performance of our DPI-RFE algorithm against two benchmark algorithms: CPI and PID (Saylam, Cikmazel, et al., 2021).

In the CPI algorithm, the duration between two successive wake-ups of the mobile device, namely T_c, is constant. This algorithm reports forecast positions during those slots at which the device sleeps. In this work, we use the same MLP-based forecasting scheme for CPI as we do for DPI-RFE.[9]

The PID algorithm adaptively chooses the positioning interval based on the forecast displacement of the mobile device D as follows: The deployment region $\mathscr{R}$ is divided into sub-regions called "cells". Based on the forecast trajectory of the device, the PID algorithm sets the positioning interval to be the duration until the first time that the device is forecast to cross over to an adjacent cell.

2.4.2 Performance Evaluation

We shall evaluate the performance of DPI-RFE against the CPI and PID algorithms with respect to both the total transmit energy consumption and the average forecasting error measured over a 300 s observation interval.

Let P_{tx} denote the transmit power consumption incurred by the mobile device when the L beacons are transmitted. We let T_{on} denote the duration during which the L beacons are transmitted.[10] Then, the total transmit energy consumption of device over one cycle of duration T_c is $P_{tx}T_{\text{on}}$. We take $P_{\text{tx}} = 16.25$ mW, and $T_{\text{on}} = 3$ ms in this work.[11]

[9] Hence, the key difference is that while T_c is constant in CPI, it is variable in DPI-RFE.

[10] Note that P_{tx} is incurred only when a beacon is transmitted. We do not quantify the idle power when no transmission occurs while the device is awake. Since we model only beacon transmission and no downlink reception, this model is reasonable in an IP setting.

[11] These are the specifications of the Texas Instruments CC2640R2F evaluation board for an AoA-based positioning system, which is taken as a representative platform.

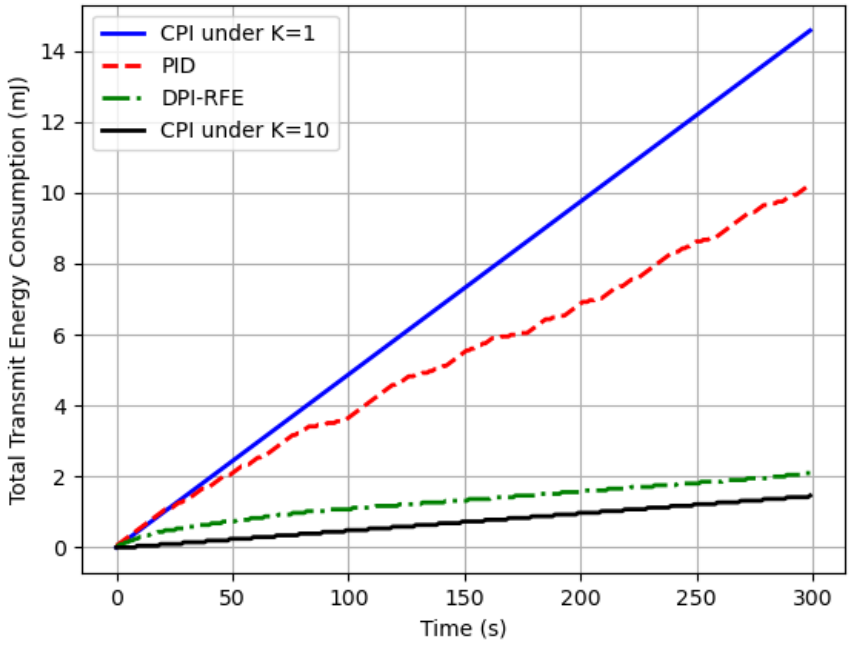

Figure 2.2. Comparison of the total transmit energy consumption incurred by the mobile IoT device under our DPI-RFE algorithm versus those under the CPI and PID algorithms

Fig. 2.2 shows the total transmit energy consumption of each algorithm over the observation interval. We let $K \equiv T_c/T$. Note that this ratio is constant for CPI. In this figure, CPI under $K = 1$ (i.e. zero sleep duration) provides an upper bound to the total transmit energy consumption, while CPI under $K = 10$ serves as a lower bound (for $V = 10$ step ahead forecasting).[12] In the figure, first, we see that the total transmit energy consumption of DPI-RFE remains close to that of the CPI for $K = 10$. Second, we see that DPI-RFE significantly outperforms PID in total energy consumption across the entire observation interval.[13]

Fig. 2.3 displays the time-averaged forecasting error so far (in meters[14]) of each algorithm over the same observation interval. At $t = 300$ s, where approximate convergence has been attained, we see that the average forecasting error of DPI-RFE is slightly higher than that of the PID algorithm. Note that the average forecasting error of both DPI-RFE and PID are above the CPI that has $K = 1$.[15]

[12] The total energy consumption of CPI under $K = 1$ increases linearly as a function of time, as the device consumes constant transmit power and does not sleep in this case. The total energy consumption of CPI under $K = 10$ is a piecewise linear function that is constant whenever the device is asleep.

[13] The plots for the total energy consumption of both DPI-RFE and PID are constant over the intervals on which the device sleeps. Recall that $T = 1$ s throughout our simulations.

[14] Since AoA-based positioning was used in the data set, the average forecasting error is relatively large compared with alternative technologies such as Ultrawideband (UWB). We emphasize that our DPI-RFE algorithm is not specific to AoA and can be used on top of any underlying IP technology.

[15] The forecasting error so far for CPI under $K = 1$ is calculated based on the past vector of position estimates (rather than forecasts) since the device does not sleep at all in this case. Even though the instantaneous forecasting error is not used at all by the CPI algorithm itself under $K = 1$, the average

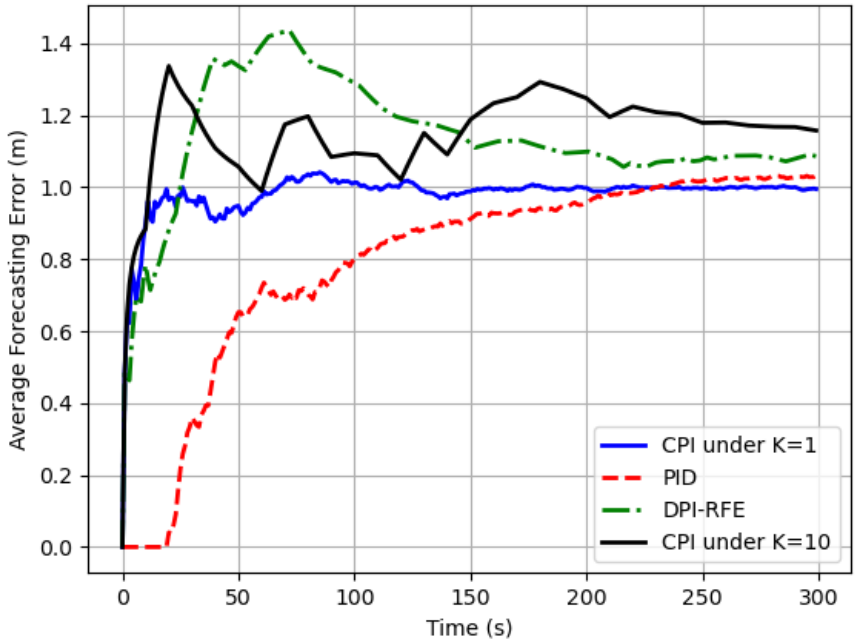

Figure 2.3. Comparison of the forecasting error of our DPI-RFE algorithm against those of the CPI and PID algorithms

Fig. 2.4 displays the instantaneous positioning interval T_c as a function of time. Note that $T_c = 1$ and $T_c = 10$ s for CPI under $K = 1$ and $K = 10$, respectively. In this figure, we see that the positioning interval of DPI-RFE varies between this lower and upper bound, while that of PID fluctuates between 1 and 4 s. The average positioning intervals of PID and DPI-RFE are 1.51 and 5.52 s, respectively. Hence, the average positioning interval of DPI-RFE is significantly higher than that of PID.

In summary, although the average forecasting error of DPI-RFE is slightly higher than that of PID as shown in Fig. 2.3, the total transmit energy consumption of DPI-RFE is significantly lower than that of PID as shown in Fig. 2.2. The reason is that DPI-RFE acts fast in response to the changes in the instantaneous forecasting error, as shown in Fig. 2.4, in order to decrease the energy expenditure.

2.5 Summary

We have developed a novel algorithm called DPI-RFE that dynamically trades off forecasting accuracy in trajectory prediction and total transmit energy consumption for mobile IoT devices by using AI. Our algorithm forecasts the future trajectory of a mobile device and updates the positioning interval based on the instantaneous reciprocal forecasting error. We have demonstrated our DPI-RFE algorithm outperforms CPI and

forecasting error attained after convergence in this case serves as a lower bound for other algorithms that are required to utilize past forecasts for those slots in which the device sleeps.

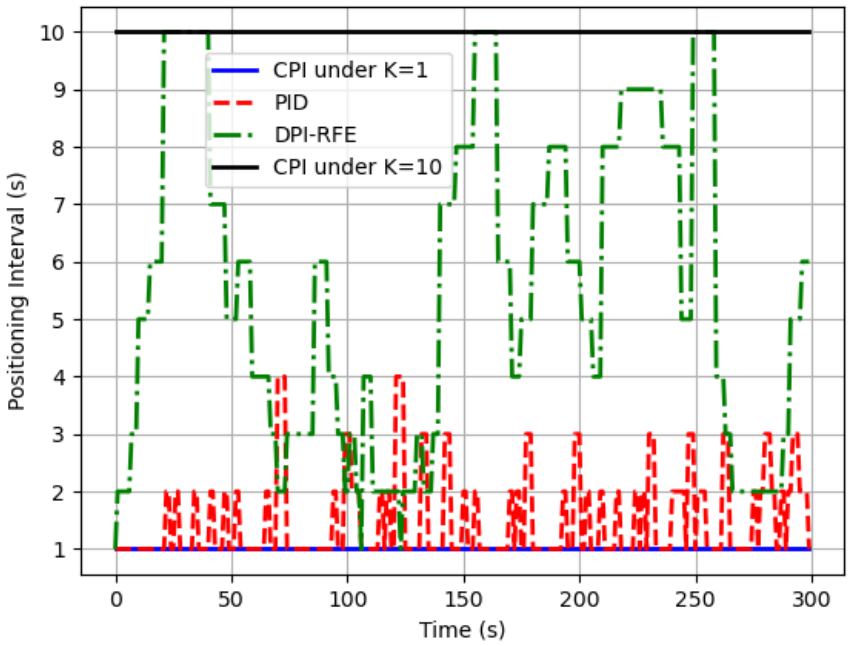

Figure 2.4. Instantaneous positioning interval T_c of our DPI-RFE algorithm versus those of the CPI and PID algorithms

PID algorithms with respect to total transmit energy consumption, while achieving an average forecasting error that is close to that of PID.

CHAPTER 3

MACHINE LEARNING ENABLED SLEEP TIME ESTIMATION (MLE-STE) ARCHITECTURE FOR INDOOR POSITIONING IN ENERGY-EFFICIENT MOBILE INTERNET OF THINGS (IOT)

3.1 Introduction

As we discussed in Chapter 1, along with the increasing demand for IoT, the positioning and tracking of mobile objects, such as pedestrians and assets that carry mobile IoT devices, IoT becomes crucial in indoor environments such as individual homes, hospitals, airports, and smart factories (Pritt, 2013; Mussina & Aubakirov, 2018; Noertjahyana, Wijayanto, & Andjarwirawan, 2017). Indoor positioning and tracking systems (positioning systems for short) span a wide range of IoT applications and take part critical roles in IoT applications. For instance, indoor positioning and tracking system monitors the employees and the assets in a smart factory to avoid the collision between assets and employees. Therefore, indoor positioning and tracking systems are expected to be accurate and efficient.

The challenges in positioning systems are positioning accuracy, energy consumption, reliability, and robustness. In this chapter, we focus on the energy consumption of a mobile IoT devices in a positioning system due to the limited battery source of the mobile IoT devices. For instance, mobile IoT devices consume significantly high energy to transmit signals. In positioning systems, a mobile IoT device periodically transmits signals in order to indicate its position (Gu, Lo, & Niemegeers, 2009; Kasmi, Norrdine, & Blankenbach, 2015; Kwak, Park, Kim, Han, & Kwon, 2018; Gu & Ren, 2015; Yüksel & Töreyin, 2018; Shih, Chiu, Cheng, Lin, & Yi, 2012). Periodic transmission requires a wide power bandwidth for mobile IoT devices since it consumes significantly high energy to transmit the signals.

AI techniques currently play an important role in IoT applications such as those found on autonomous cars, smart grids as well as in farming. In addition, AI is becoming a significant enabler of positioning systems, where significant performance advantages can be obtained (Mukhopadhyay et al., 2021; Cheng, Chang, Wang, & Wu, 2020). In positioning systems, AI techniques are utilized in order to improve positioning accuracy as well as forecast the future trajectory of the mobile IoT device (Yu, Oh, & Kim, 2020; Cakan et al., 2021; Mantini & Shah, 2016).

We categorize the past works that have focused on the trajectory forecast and enhanced the energy efficiency of the mobile IoT device into three groups: First, the past works (Alahi et al., 2016; Huang et al., 2019; Ma et al., 2019; Xue et al., 2018; Pellegrini et al., 2009; C. Wang et al., 2019) present AI techniques that are utilized in order to forecast the future trajectory of the mobile device in indoor positioning and tracking systems. Second, the past works in the positioning literature (Zhang et al., 2012; X. Liu et al., 2018; González et al., 2019; Shafer & Chang, 2010; Kjærgaard et al., 2009) focus on motion-triggered algorithms in order to reduce the advertising energy consumption (namely, transmit energy consumption) of the mobile device. The idea behind motion-triggered algorithms is that the position of the mobile device is updated only when the device detects motion. Third, the recent works (Constandache et al., 2009; Chon et al., 2011; Saylam, Cikmazel, et al., 2021; Saylam, Kelesoglu, Cikmazel, Nakip, & Rodoplu, 2021) introduce algorithms that forecast the trajectory of the mobile device but do not utilize machine-learning-based sleep time estimation.

The main contribution of this chapter[1] is the development of an architecture called MLE-STE for energy-efficient mobile IoT devices in indoor positioning systems. Our architecture first forecasts the future trajectory of the mobile device and then estimates the maximum allowable sleep time under a given acceptable forecasting error and acceptable displacement. Our MLE-STE architecture significantly reduces the transmit energy consumption of the mobile device by adapting the positioning interval. While the recent works (Saylam, Cikmazel, et al., 2021) and (Saylam, Kelesoglu, et al., 2021) for energy-efficient indoor positioning systems have focused on only forecasting error or device displacement, our novel architecture estimates the sleep duration via a ML model that considers both forecasting error and device displacement.

We compare the performance of our architecture against the algorithms in (Saylam, Cikmazel, et al., 2021), (Saylam, Kelesoglu, et al., 2021) called PID and DPI-RFE in terms of total transmit energy consumption and average forecasting error. We see that our architecture outperforms the PID and DPI-RFE algorithms with respect to the total transmit energy consumption.

The rest of this chapter is organized as follows: In Section 3.3, we present our MLE-STE architecture for the indoor positioning. In Section 3.4, we present our simulation methodology, benchmarks, and performance evaluation. In Section 3.5, we summarize our results.

[1] The technical content in this chapter has been submitted as a journal paper (Saylam, Güzeliş, & Rodoplu, n.d.).

3.2 Assumptions

Throughout this chapter, we assume that there is a single mobile IoT device (shortly, device), denoted by D, that moves in a region called "deployment region". We let R denote the deployment region. We assume that there is a set of anchors denoted by $\mathscr{M}$. The device communicates to each anchor by sending a positioning signal called a "beacon". We shall define positioning indicator[2] that is computed by the anchors based on the received beacons.

We let G denote the gateway in whose coverage area the device and the anchors fall. We define the position estimate of the device as a $2D$ position that is estimated by G based on the positioning indicators that the G receives from the anchors. We assume that each anchor communicates its positioning indicator to the gateway on the uplink channel, and the gateway also can communicate with the device via the downlink channel when it is required. Let B denote the number of beacons called successive beacons required by G in order to estimate the current position of the device. We let T denote required duration at which the device transmits B successive beacons when it is awake. Throughout this thesis, we assume that T is an integer. Let E denote the transmit energy consumption that the device requires in order to send B beacons. In addition, we define "sleep time" as the duration that the mobile device does not send any beacon. Let T_s denote the sleep time. We also define the "positioning interval" as the time interval between two position estimates and denote it by T_c. Note that T_c as $T_c = T_s + T$. We assume that $T_s = KT$, where K is a positive integer.

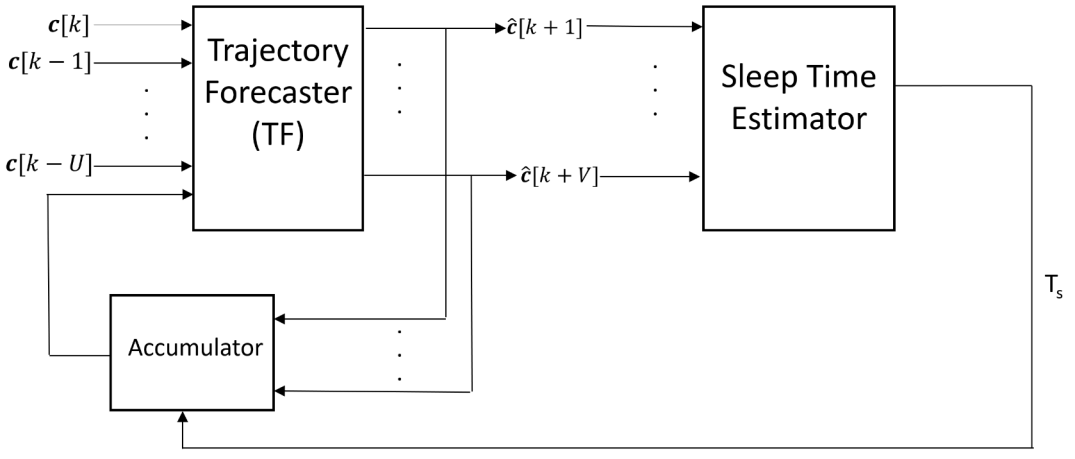

Figure 3.1. Architecture of the Machine Learning Enabled Sleep Time Estimation (MLE-STE) architecture for energy-efficient indoor positioning

[2]The positioning indicator can be chosen as Angle of Arrival (AoA), Received Signal Strength Indicator (RSSI), Time of Flight (TOF), depending on desired positioning accuracy.

3.3 Machine Learning Enabled Sleep Time Estimation Architecture

In this section, we describe our novel architecture for an energy-efficient indoor positioning system. Our architecture is comprised of a Trajectory Forecaster (TF), an accumulator, and a Sleep Time Estimator (STE) module, as shown in Fig. 3.1. First, we shall define cells as rectangular regions that are formed by subdividing the deployment region R into N identical regions each of which has length L and width W as shown in Fig. 3.2.[3] Second, we let $i_x[k] \in \mathbb{N}$ and $i_y[k] \in \mathbb{N}$ denote the row index and the column index of the cell for which the position estimate of the device takes place at time instant k, where let k denote the discrete time. We also define the cell estimate of the device (namely, cell of the device) as $\mathbf{c}[k] = [i_x[k], i_y[k]]^T$.

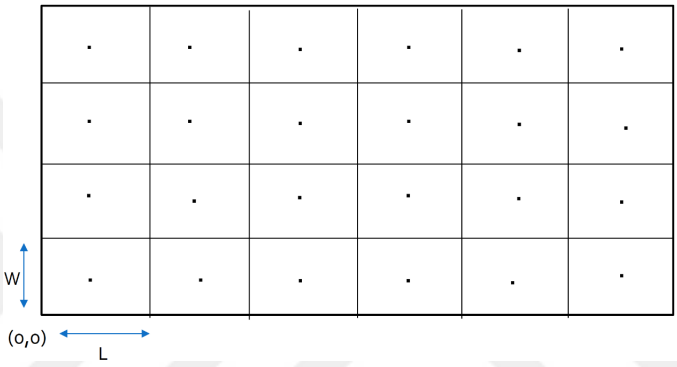

Figure 3.2. Subdivision of the deployment region into identical subregions

TF consists of two distinct forecasters such that the first one individually forecasts the row index and the second one forecasts the column index of the cell of the device. Note that the TF converts each position estimate to a cell estimate before the forecasting. Recall that the cell estimates are available only when the device is awake and transmits beacons.[4] As shown in Fig 3.1, in order to form the trajectory forecasting in each case (i.e. if the device is asleep or the device is awake), whenever TF forecasts, the accumulator stores and delays the forecast cells of the device. We call those cells delayed forecast cells of the device. The delayed positions are utilized to fill the empty inputs of TF which corresponds to cells when device is asleep (The instants where the device is asleep are obtained from the feedback loop of T_s.) for the next forecasting. The accumulator sends only the delayed forecast cells where the device is asleep, which are the empty input for the TF. Note that the input of TF is a collection that consists of the combination of the past estimate cells and the delayed forecast cells of the device and as shown in Fig 3.1. In other words, $\mathbf{c}^T[k-u]_{u\in\{0,\ldots,U\}}$ represents the past cell estimates and filled positions with the delayed forecast cells. We let $\mathbf{c}^T[k-u]_{u\in\{0,\ldots,U\}}$

[3] The deployment region does not have to be rectangular. There exist small enough L and W such that the deployment region can be covered by such cells arbitrary chosen.

[4] There is no beacon transmission from the device to the anchors when the device is asleep.

denote input of TF, where U indicates the past window. The output of TF is a collection that is comprised of the forecast cell of the device (in other words, trajectory forecasts) denoted by $\hat{\mathbf{c}}^T[k+v]_{v\in\{1,\ldots,V\}}$, where V denotes the forecast window.

In Fig. 3.1, the STE module takes $\hat{\mathbf{c}}^T[k+v]_{v\in\{1,\ldots,V\}}$ and updates T_s. We will describe the internal structure of the STE module in Section 3.3.2. Recall that $T_c = T_s + T$.[5]

3.3.1 Trajectory Forecaster

Fig. 3.3 shows the internal structure of TF. Recall that TF is comprised of two distinct forecasters that forecast the row index and column index of the cell of the device. Note that each forecaster in TF is also individually trained. We shall let F^x denote the forecaster that forecasts the row index of the cell of the device, and F^y denote the forecaster that forecasts the column index of the cell of the device.

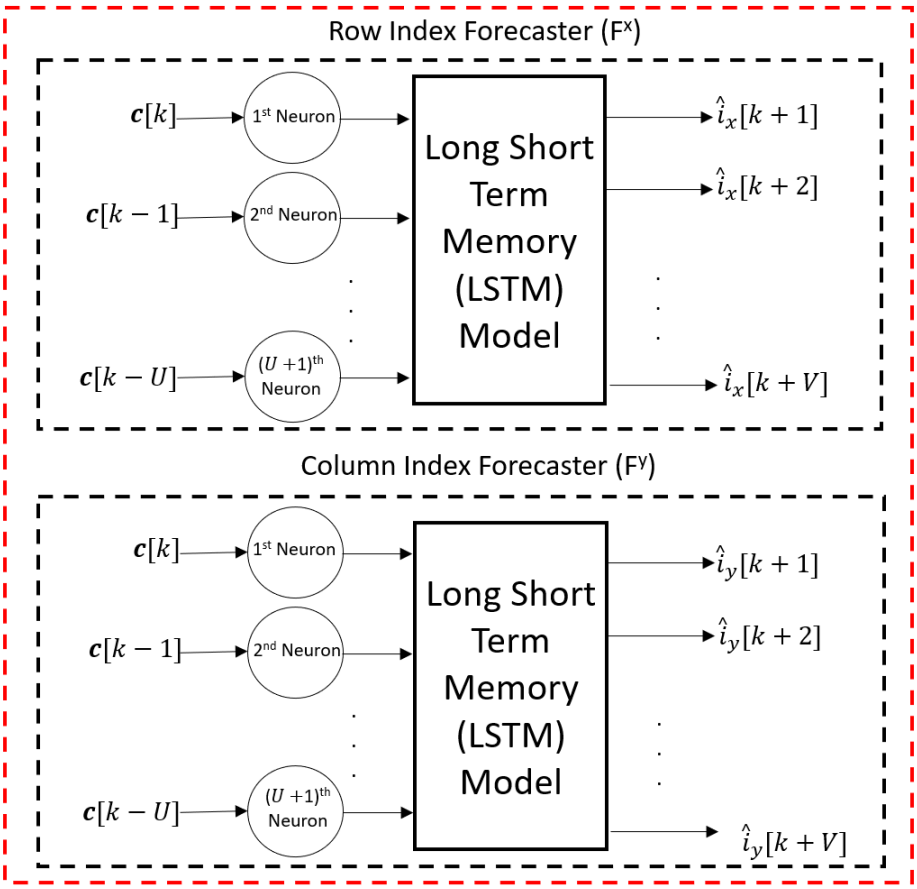

Figure 3.3. Internal structure of the Trajectory Forecaster (TF)

Fig. 3.3, the forecaster F^x is comprised of a collection of neurons with a sigmoid activation function and a Long Short Term Memory (LSTM) model (Smagulova &

[5]In the positioning system, the gateway G communicates the updated sleep time to the device. The device sleeps until the end of that sleep time and then starts to send the beacon.

James, 2019). Recall that $\mathbf{c}^T[k-u]_{u\in\{0,\ldots,U\}}$ is the combination of the past cell estimates and delayed forecast cells. The forecaster F^x has $U+1$ neurons each of which corresponds to a past cell of the device. First, each sigmoid neuron u in F^x takes $\mathbf{c}[k-u]$ in $\mathbf{c}^T[k-u]_{u\in\{0,\ldots,U\}}$ in order to produce features, each of which contains information about the past cell of the device. The LSTM model in F^x then takes the outputs of each sigmoid neuron as its inputs in order to produce the collection of the row index of the forecast cell of the device denoted by $[\hat{\boldsymbol{i}}_x[k+v]_{v\in\{1,\ldots,V\}}]^T$. As shown in Fig. 3.3, F^y has the same inputs and the same internal structure as F^x, but it produces the collection of the column index of the forecast cell of the device denoted by $[\hat{\boldsymbol{i}}_y[k+v]_{v\in\{1,\ldots,V\}}]^T$. When F^x and F^y complete their forecasting processes, TF combines each forecast column index with its related row index in order to produce $\hat{\mathbf{c}}^T[k+v]_{v\in\{1,\ldots,V\}}$.

3.3.2 Sleep Time Estimator Module

In this section, we describe the inputs, the outputs, and the internal structure of the Sleep Time Estimator module, which dynamically updates the sleep time of the device based on the trajectory forecasting error[6] of TF and displacement of the device in the forecast cell positions in order to reduce the transmit energy consumption of the device.

Fig. 3.4 shows the internal structure of the STE module, which is comprised of a Bank of Sigmoid Neurons (BoSN) that has V number of neurons and a Multi-Layer Perceptron (MLP) model. The input of the module is $\hat{\mathbf{c}}^T[k+v]_{v\in\{1,\ldots,V\}}$ and the output of the module is T_s. Note that $\hat{\mathbf{c}}[k]=[\hat{i}_x[k],\hat{i}_y[k]]^T$.

We explain the key idea behind this module is as follows: The STE module estimates T_s for the device by considering the trajectory forecast of the device. However, if the module puts the device to sleep for a long time, the forecasting error of TF increases due to the accumulation process explained in Section 3.3. To this end, the STE module estimates the maximum allowable T_s for the device by considering both the device displacement and a given maximum allowable forecasting error. In Section 3.3.2.1, we will describe via pseudo-code how to determine the set whose elements are the optimal sleep times across trajectory forecasts.

The STE module has an end-to-end trainable[7] algorithm that includes two stages: In the first stage each sigmoid neuron v takes $i_x[k+v]$ and $i_y[k+v]$ as inputs in order to produce f_v, where f_v is the output of the vth sigmoid neuron. Note that each f_v carries the information on the cell of the mobile device at time instant $k+v$. In the second stage, the MLP model takes the collection of features $f_{v\in\{1,\ldots,V\}}$ produced in the first

[6]The trajectory forecasting error refers to the Euclidean distance between the forecast cell of the device and the estimated cell of the device. Let e_{forecast} denote forecasting error of TF.

[7]The phrase "end-to-end trainable" refers to the fact that all parameters are trainable under the same loss function.

stage in order to estimate T_s.

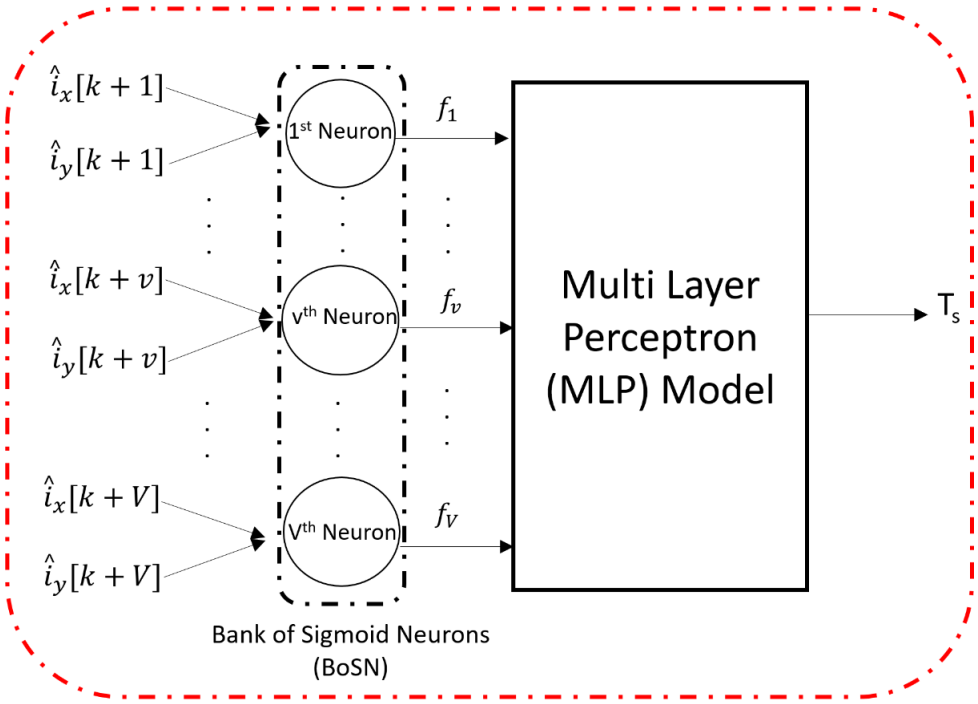

Figure 3.4. Internal structure of the Sleep Time Estimator (STE) module

3.3.2.1 Training Methodology for the STE Module

In order to obtain training data for the STE module, we first run TF, and construct a set that is comprised of the trajectory forecasts. Let $\hat{\mathscr{T}} = \{\hat{\mathbf{c}}^T[k+v]_{v\in\{1,\ldots,V\}} \in \mathbb{N}^{2V}\}_k$ denote a set of the trajectory forecasts. In addition, we let $\mathscr{T} = \{\mathbf{c}^T[k+v]_{v\in\{1,\ldots,V\}} \in \mathbb{N}^{2V}\}_k$ denote the set of future estimate cells of the device, constructed for training of the MLE-STE. We note that each sample in $\mathscr{T}$ is associated with a sample $\hat{\mathscr{T}}$.[8] We let γ denote the threshold for the displacement in forecast cell positions of the device (Which we call device displacement for short), and $e_{\max}$ denote the maximum tolerable forecasting error for the Trajectory Forecaster. Second, we define the "optimal sleep" time as one that satisfies the following conditions: The device displacement is less than γ and the forecasting error of the Trajectory Forecaster is less than $e_{\max}$. In addition, there is an optimal sleep time for each trajectory forecast. We let $\mathscr{S}^*$ denote a set, which is comprised of optimal sleep times, each of which is associated with a trajectory forecast in $\hat{\mathscr{T}}$.

We now describe via pseudo-code how to determine the optimal sleep time values, each of which corresponds to a trajectory forecast. Fig. 3.5 shows the pseudo-code of the DetermineOptimalSleepTime function (or "function" for short) that is utilized to determine an optimal sleep time for each forecast trajectory.

[8]For instance, the nth sample of $\mathscr{T}$ corresponds to the nth sample of $\hat{\mathscr{T}}$.

```
1  Set DetermineOptimalSleepTime(𝒯, 𝒯̂, γ, e_max) {
2  𝒮* = ∅ ;
3  for (i = 0; i < |𝒯̂|; i++) {
4     [flag, m*] = find(argmax_m{||𝒯̂[i][0] − 𝒯̂[i][m]||
4     | ||𝒯̂[i][0] − 𝒯̂[i][m]|| ≤ γ});
5     if(flag = 0)   𝒮*.append(0);
6     else {
7       while (e_forecast ≤ e_max) {
8          e_forecast = ||𝒯[i][m*] − 𝒯̂[i][m*]||;
9          m* = m* + 1;
10         }
11       𝒮*.append(m*);
12    }
13    }
14 return 𝒮*;
15 }
```

Figure 3.5. Training data preparation for the STE module

On Line 2, the function initializes $\mathscr{S}^*$ to the null set. On Line 3, the function selects the ith sample in $\mathscr{T}$ and $\hat{\mathscr{T}}$. On Line 4, the function first computes maximum m value denoted m^* in the range $[1,V]$ that does not exceed γ by taking the weighted Euclidean norm between first and mth forecast position of the ith sample in $\hat{\mathscr{T}}$, where weight is equal to 1.25. The idea behind this line is to find the time when the device changes its position by applying thresholding. Then, the function returns a flag and m^*. If there exist a m^*, flag is 1 and is 0 otherwise. This flag refers that the function finds a m^* where the device position exceeds the threshold. Hence, the function returns m^* if and only if the flag is 1. On Line 5 the function checks the flag. If the flag is 0, the function sets 0 to the optimal sleep time for that trajectory forecast.

Otherwise (if the flag is 1), between Line 7 and 9, the function computes the weighted Euclidean norm between the m^*th forecast cell position in ith sample in $\hat{\mathscr{T}}$ and m^*th future cell position of the ith sample in $\mathscr{T}$ in order to determine the forecasting error of TF, denoted by e_{forecast}. The weight of that Euclidean norm is equal to 1.25. The function then increases the m^* by one. That process continues until the break condition (namely, $\text{e}_{\text{forecast}} \leq e_{\max}$) is satisfied. When the break condition is not satisfied which means that the forecasting error of the TF exceeds the given threshold, the optimal sleep time of that sample is set to m^* (Line 11).

In Fig. 3.6, we demonstrate how to determine desired T_s values for each trajectory forecast that is produced by the TF. To this end, we present two distinct scenarios that show the determination of the desired T_s values for two distinct forecast trajectories. In each scenario, we set γ and $e_{\max}$ which are the design parameters for MLE-STE architecture to 1.25 m.

In the first scenario shown in Fig. 3.6, each slot represents a time slot and the values in the slots represent the center of the cells. According to the process, firstly we check the displacement in the forecast trajectory which is the Euclidean distance between two successive forecast positions. We see that the displacement of the device is zero until the $t = 3$ but that of the device is 1.76 at $t = 4$, which exceeds the threshold value for the device displacement γ. Then, the process completes the displacement check and starts checking the forecasting error, which is the Euclidean distance between a forecast and an actual positions. When we check the forecasting error at $t = 4$, we see that the forecasting error is 1.76 and exceeds threshold $e_{\max}$. Thus, the process terminates at $t = 4$ and declares the $t = 4$ as the desired T_s for that forecast trajectory.

Similarly, in the second scenario, we first compute the device displacement in the forecast trajectory until the displacement exceeds γ. As shown in the second scenario in Fig. 3.6, the device displacement exceeds γ at $t = 3$. After the displacement check, the process computes the forecasting error between the forecast and actual positions at $t = 4$. Since the forecasting error at $t = 4$ does not exceed $e_{\max}$, the process continues with the next forecast and actual positions. We see that the forecasting error at $t = 5$ exceeds $e_{\max}$. Thus, the process terminates at $t = 5$ and determines $t = 5$ as the desired T_s for that forecast trajectory. This process is applied to each trajectory forecast that TF produces in order to determine its desired T_s values.

3.4 Results

3.4.1 Simulation Methodology

In this subsection, we will describe the indoor positioning data set, structure, and the training parameters for the TF and the STE modules.

3.4.1.1 Indoor Positioning Data Set

In this chapter, we use the data set in (Fang, Islam, Munir, & Nirjon, 2020). It is comprised of the 2D position estimates of the human users that carry a mobile IoT device on a 11.84 m $\times$ 18 m rectangular space. The position estimates are obtained via the Wireless Fidelity (Wi-Fi) readings. The system parameters that are used estimating the device position are as follows: T = 50 ms, B = 1, and the energy consumption of the device incurred to send a beacon is 48.75 μJ. In this chapter, we utilize the position estimate of a single user in our simulation.[9] We subdivide the dataset into a train set and a test set, whose portions are as: 80% and 20% respectively.

[9]Our design can be generalized to multiple users by applying this design to each user.

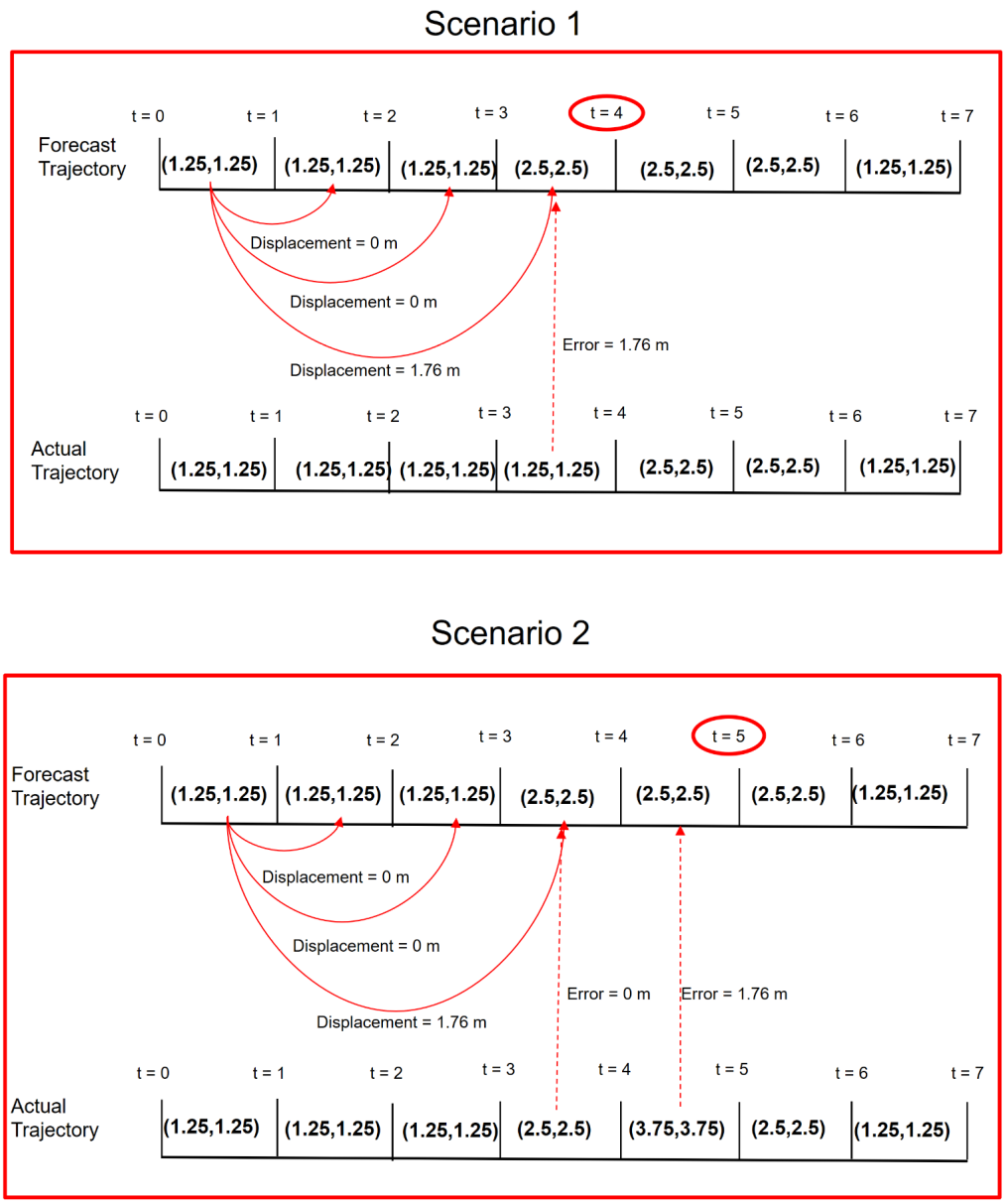

Figure 3.6. Demonstration of how to determine the desired value of T_s

Before training, we convert the 2D estimate positions of the device to the cell of the device as described in Section 3.3. We set $L = 1.25$ m and $W = 1.25$ m.[10]

3.4.1.2 Training Parameters of the Trajectory Forecaster

In this subsection, we describe the training parameters of the TF. We set $U = 300$ and $V = 400$.[11] In addition, we fix the structure of each LSTM model in the Trajectory

[10] We set the length and the width of the cell to 1.25 m because the most of the works in the indoor positioning literature that use the Wi-Fi technology have approximately 1.25 m average positioning error. (H. Liu, Darabi, Banerjee, & Liu, 2007).

[11] Since, the position estimate interval is 50 ms, we set the past window U and forecast window V to large numbers in order to forecast positions over a long time horizon.

Forecaster as follows: One hidden LSTM layer, which has RELU activation function, three dense layers and one output layer, which consists of V linear neurons. In addition, the activation function of each neuron at each of the dense layer is set to RELU. We search the number of neurons for each hidden layer in the range $[2, 256]$ at increments of two in order to obtain the local optimal structure that gives the minimum Mean Squared Error (MSE). The local optimal structure of each forecaster in TF is (in vector notation) $[50, 128, 64, 32, 400]$. Moreover, the training parameters of each network (F^x and F^y) in TF are as follows: We set the number of epochs to 100. We apply early stopping and 5-fold Cross-validation[12] in order to prevent overfitting. Furthermore, we utilize Adaptive moment estimation (Adam) as the optimizer.

3.4.1.3 Training Parameters of the STE Module

We now describe the training parameters of the STE module. The module takes the samples in $\hat{\mathscr{T}}$ as inputs and the samples in $\mathscr{S}^*$ as the desired outputs. Recall that the STE is comprised of the Bank of Sigmoid Neurons (BoSN) and the MLP model. The structure of the module is as follows: BoSN consists of 400 sigmoid neurons; an MLP model consists of 4 hidden layers each of which has a RELU activation function; and one output layer, which is comprised of a single linear neuron. We apply the search described in Section 3.4.1.2 in order to find the local optimal structure for the STE module. The local optimal MLP architecture for that data set is (in vector notation) $[128, 64, 32, 16, 1]$. Moreover, we set the training parameters in Section 3.4.1.2 for the training of the STE module.

3.4.2 Benchmark Algorithms

In this subsection, we will describe two benchmark algorithms that are utilized in order to reduce the transmit energy consumption of the device by adjusting the positioning interval. We will compare the performance of the MLE-STE architecture against these two benchmarks in Section 3.4.3.

The first benchmark is called PID, which first takes the forecast positions of the device and then computes the positioning interval of the device based on the displacement of the device (Saylam, Cikmazel, et al., 2021). This algorithm first divides the deployment region $\mathscr{R}$ into the sub-regions called "cells" and allocates each forecast position of the device to the corresponding cell where the 2D position of the device falls. It then sets the duration, where the device passes to the adjacent cell for the first time as the positioning interval of the device.

[12] In 5-fold cross-validation, the data are split into 5 segments. In each fold 4 of these segments are used for training and 1 segment is used for testing.

The second one, called DPI-RFE which we describe in previous chapter, computes the positioning interval of the device as follows: The DPI-RFE algorithm first takes the forecasts positions of the device and computes the forecasting error by taking the Euclidean distance between the position estimate where the device is awake, and the forecast position that corresponds to this position estimate. It then determines the duration, which is reciprocal to the forecasting error as the positioning interval of the device (Saylam, Kelesoglu, et al., 2021).

3.4.3 Performance Evaluation

In this section, we present our simulation results in order to evaluate the performance of MLE-STE against PID and DPI-RFE algorithms, which are energy-efficient algorithms in indoor positioning literature. We examine the STE module under $e_{\max} = 1.25$ m because this condition refers to the minimum possible forecasting error when the cell size is 1.25 m $\times$ 1.25 m. We also examine MLE-STE under $e_{\max} = 2.5$ m, which refers to double of the minimum possible error. We do not consider values greater than $e_{\max} = 2.5$ m because they do not significantly improve energy efficiency. In addition, we compare these schemes over a 120 s observation interval in terms of total transmit energy consumption of the mobile device, the forecasting error of the trajectory forecasters, and the positioning interval of the device.

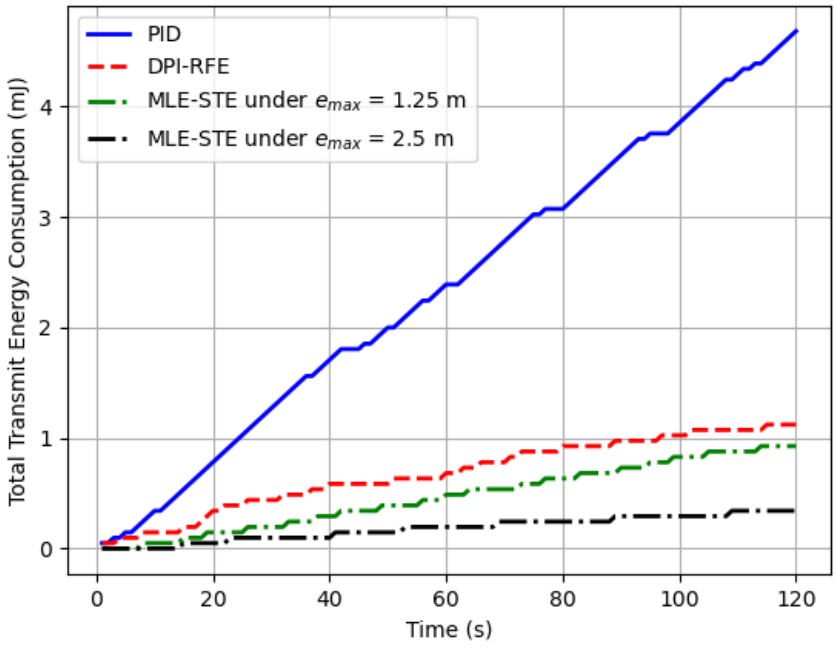

Figure 3.7. Total transmit energy consumption performance under MLE-STE, PID, and DPI-RFE algorithms

Fig. 3.7 shows the total transmit energy consumption of the device for PID, DPI-RFE and MLE-STE under $e_{\max} = 1.25$ m and $e_{\max} = 2.5$ m. We measure the total transmit

energy consumption under each algorithm as follows: If the device is awake, the total transmit energy consumption continually increases. Otherwise (when the device is asleep) the total transmit energy consumption stays constant during the sleep time of the device. When we examine our results in Fig. 3.7, we see that the transmit energy consumption of the PID algorithm is significantly higher than that of DPI-RFE and MLE-STE. We also see that the DPI-RFE is close MLE-STE under $e_{\max} = 1.25$ m but the STE module under $e_{\max} = 2.5$ m significantly outperforms the other algorithms shown in label of Fig. 3.7.

Table 3.1. Forecasting error of the trajectory forecasters under MLE-STE, PID and DPI-RFE

Algorithm Type	Mean (m)	Standard Deviation (m)
PID	2.82	1.08
DPI-RFE	3.4	1.47
MLE-STE under $e_{\max}$ = 1.25 m	3.5	0.9
MLE-STE under $e_{\max}$ = 2.5 m	3.53	0.75

In Table 3.1, we present the mean and standard deviation of the forecasting error of the trajectory forecasters in each algorithm.[13] In this table, we see that while the forecasting error of STE module under $e_{\max} = 1.25$ m and $e_{\max} = 2.5$ m are slightly higher than that of DPI-RFE. We also see that the forecasting error of the PID algorithm is comparable to that of MLE-STE.

Table 3.2. Positioning interval of the mobile IoT device under MLE-STE and benchmark algorithms

Algorithm Type	Mean (s)	Standard Deviation (s)
PID	1.24	0.61
DPI-RFE	6.82	5.22
MLE-STE under $e_{\max}$ = 1.25 m	6.38	1.79
MLE-STE under $e_{\max}$ = 2.5 m	14.96	3.56

In Table 3.2, we present the positioning interval of the device, which is inversely proportional to the energy consumption of the device. We see that the positioning

[13] Each algorithm accumulates the forecast positions when the device is asleep.

interval of MLE-STE under $e_{\max} = 2.5$ m is approximately two times higher than those of DPI-RFE and MLE-STE under $e_{\max} = 1.25$ m. Moreover, the positioning interval of MLE-STE under $e_{\max} = 2.5$ is significantly higher than that of PID, where the difference is one order of magnitude.

In summary, although the forecasting error of MLE-STE under $e_{\max} = 2.5$ m is slightly higher than those of DPI-RFE and MLE-STE under $e_{\max} = 1.25$ m and is comparable to that of PID as shown in Table 3.1, the total transmit energy consumption of the of MLE-STE under $e_{\max} = 2.5$ m outperforms the other algorithms as displayed in Fig. 3.7. The reason is that while the PID algorithm only updates the positioning interval of the device whenever it observes the displacement, and DPI-RFE updates positioning interval reciprocal to the forecasting error, the STE module considers both the displacement of the device and the forecasting error in order to adjust the positioning interval.

3.5 Summary

We have developed a novel architecture called MLE-STE that reduces the energy consumption of mobile IoT devices in indoor positioning systems. Our MLE-STE first forecasts the trajectory of the mobile device and then estimates the sleep time, which satisfies the given forecasting error threshold and the device displacement threshold, based on that forecast trajectory.

We investigated the performance of our MLE-STE against the PID and DPI-RFE algorithms, which served as benchmarks in this chapter, in terms of total transmit energy consumption and forecasting error. Our simulation results showed that MLE-STE architecture outperforms the benchmarks. In our future work, we will implement our MLE-STE architecture on a gateway that serves an energy-efficient indoor positioning system.

CHAPTER 4
CONCLUSIONS

In this thesis, in order to reduce the transmit energy consumption of a mobile IoT device in an indoor positioning system, we have introduced the "Dynamic Positioning Interval based on Reciprocal Forecasting Error (DPI-RFE)" algorithm and the "Machine Learning Enabled Sleep Time Estimation (MLE-STE)" architecture, which provide adaptive solutions that adjust the positioning interval and the sleep time of the mobile IoT device. Each of these first forecasts the trajectory of the mobile IoT device and determines a sleep time for the device based on the forecast trajectory. This reduces the transmit energy consumption of the mobile IoT device since the mobile IoT device is prevented from periodically transmitting signals. In Chapter 2, we proposed the DPI-RFE algorithm, which aims to maximize the energy efficiency of the mobile IoT device by adjusting the sleep time of the device reciprocally to the forecasting error. We also evaluated the performance of the algorithm. In Chapter 3, we proposed the MLE-STE architecture, which determines the sleep time of the mobile IoT device by satisfying the given conditions and is introduced to improve the energy consumption of the device.

In Chapter 2, we proposed our DPI-RFE algorithm and compared the performance of this algorithm in terms of total transmit energy consumption, positioning interval, and average forecasting error against the benchmarks algorithms, which are Constant Positioning Interval (CPI) under $K = 1$, CPI under $K = 10$, and Positioning Interval based on Displacement (PID). We have come to the following conclusions: (1) DPI-RFE significantly outperforms PID and CPI when the $K = 1$. It is also comparable with CPI when the $K = 10$ with respect to the total transmit energy consumption. (2) While the average positioning interval of DPI-RFE is 1.51, that of PID is 5.52. (3) DPI-RFE has a slightly higher average forecasting error than PID and CPI under $K = 1$.

In Chapter 3, we introduced our novel MLE-STE architecture and examined the performance of MLE-STE against the benchmark algorithms, called PID and DPI-RFE, each of which reduces the transmit energy consumption of the mobile device in an indoor positioning system. We compare the performance of MLE-STE under $e_{\max} = 1.25$ m and $e_{\max} = 2.5$ m against PID and DPI-RFE with respect to the total transmit energy consumption, the average positioning interval, and the average forecasting error. First, we showed that our MLE-STE significantly outperforms each benchmark algorithm

under $e_{\max} = 2.5$ m. Second, we showed that while the average positioning interval of our MLE-STE is 14.96 s under $e_{\max} = 2.5$, that of PID is 1.24 s and that of DPI-RFE is 6.82 s. Finally, we demonstrated that under $e_{\max} = 2.5$ m MLE-STE has a slightly higher forecasting error than PID, which achieves the lowest forecasting error among the benchmarks.

In conclusion, we showed that DPI-RFE and MLE-STE are promising for energy-efficient indoor positioning. In addition, our results show that DPI-RFE and MLE-STE significantly improves energy efficiency for mobile IoT devices in an indoor positioning system.

In the future, first, the forecasting performance of the trajectory forecasting can be improved. Second, the system can be generalized from the case of a single mobile IoT device to multiple mobile IoT devices. Third, positioning errors due to the multipath effect can be addressed in our system.

REFERENCES

Abdellatif, M., Mtibaa, A., Harras, K. A., & Youssef, M. (2013). Greenloc: An energy efficient architecture for WiFi-based indoor localization on mobile phones. In *2013 IEEE international conference on communications (icc)* (pp. 4425–4430).

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., & Savarese, S. (2016). Social LSTM: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (pp. 961–971).

Alrashidi, M. (2020). Social distancing in indoor spaces: an intelligent guide based on the Internet of Things: COVID-19 as a case study. *Computers*, *9*(4), 91.

Bai, L., Ciravegna, F., Bond, R., & Mulvenna, M. (2020). A low cost indoor positioning system using Bluetooth Low Energy. *IEEE Access*, *8*, 136858–136871.

Bisio, I., Lavagetto, F., Marchese, M., & Sciarrone, A. (2016). Smart probabilistic fingerprinting for WiFi-based indoor positioning with mobile devices. *Pervasive and Mobile Computing*, *31*, 107–123.

Cakan, E., Şahin, A., Nakip, M., & Rodoplu, V. (2021). Multi-layer perceptron decomposition architecture for mobile IoT indoor positioning. In *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)* (pp. 253–257).

Cheng, L., Chang, H., Wang, K., & Wu, Z. (2020). Real time indoor positioning system for smart grid based on UWB and Artificial Intelligence techniques. In *2020 IEEE conference on Technologies for Sustainability (SusTech)* (pp. 1–7).

Chon, Y., Talipov, E., Shin, H., & Cha, H. (2011). Mobility prediction-based smartphone energy optimization for everyday location monitoring. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems* (pp. 82–95).

Constandache, I., Gaonkar, S., Sayler, M., Choudhury, R. R., & Cox, L. (2009). Enloc: Energy-efficient localization for mobile phones. In *IEEE INFOCOM 2009* (pp. 2716–2720).

Duque Domingo, J., Cerrada, C., Valero, E., & Cerrada, J. A. (2017). An improved indoor positioning system using RGB-D cameras and wireless networks for use in complex environments. *Sensors*, *17*(10), 2391.

Ericsson. (Jun. 2021).
Ericsson Mobility Report. (https://www.ericsson.com/en/mobility-report)

Fang, S., Islam, T., Munir, S., & Nirjon, S. (2020). EyeFi: Fast human identification through vision and WiFi-based trajectory matching. In *IEEE International*

Conference on Distributed Computing in Sensor Systems (DCOSS).

Fantana, N. L., Riedel, T., Schlick, J., Ferber, S., Hupp, J., Miles, S., ... Svensson, S. (2013). Iot applications—value creation for industry. *Internet of Things: Converging technologies for smart environments and integrated ecosystems*, 153.

García, E., Poudereux, P., Hernández, Á., Ureña, J., & Gualda, D. (2015). A robust uwb indoor positioning system for highly complex environments. In *2015 IEEE International Conference on Industrial Technology (ICIT)* (pp. 3386–3391).

González, J. L. S., Morillo, L. M. S., Álvarez-García, J. A., Ros, F. E. D. S., & Ruiz, A. R. J. (2019). Energy-Efficient Indoor Localization WiFi-Fingerprint System: An experimental study. *IEEE Access*, *7*, 162664–162682.

Gu, Y., Lo, A., & Niemegeers, I. (2009). A survey of indoor positioning systems for wireless personal networks. *IEEE Communications surveys & tutorials*, *11*(1), 13–32.

Gu, Y., & Ren, F. (2015). Energy-efficient indoor localization of smart hand-held devices using Bluetooth. *IEEE Access*, *3*, 1450–1461.

Haykin, S. S., & Haykin, S. S. (2001). *Kalman filtering and neural networks* (Vol. 284). Wiley Online Library.

Huang, Y., Bi, H., Li, Z., Mao, T., & Wang, Z. (2019). Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 6272–6281).

Kárník, J., & Streit, J. (2016). Summary of available indoor location techniques. *IFAC-PapersOnLine*, *49*(25), 311–317.

Kasmi, Z., Norrdine, A., & Blankenbach, J. (2015). Towards a decentralized magnetic indoor positioning system. *Sensors*, *15*(12), 30319–30339.

Kjærgaard, M. B., Langdal, J., Godsk, T., & Toftkjær, T. (2009). Entracked: energy-efficient robust position tracking for mobile devices. In *Proceedings of the 7th international conference on Mobile systems, applications, and services* (pp. 221–234).

Kuxdorf-Alkirata, N., Spathmann, O., Koch, O., & Bruckmann, D. (2018). Improved energy efficiency of indoor positioning systems by adaptive sampling and smart post-processing of sensor data. In *2018 16th IEEE International New Circuits and Systems Conference (NEWCAS)* (pp. 225–228).

Kwak, M., Park, Y., Kim, J., Han, J., & Kwon, T. (2018). An energy-efficient and lightweight indoor localization system for Internet-of-Things (IoT) environments.

Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, *2*(1), 1–28.

Liu, H., Darabi, H., Banerjee, P., & Liu, J. (2007). Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *37*(6), 1067–1080.

Liu, H., Xu, Y., Wu, X., Lv, X., Zhang, D., & Zhong, G. (2019). Big data forecasting model of indoor positions for mobile robot navigation based on apache spark platform. In *2019 ieee 4th international conference on cloud computing and big data analysis (icccbda)* (pp. 378–382).

Liu, X., Zhan, Y., & Cen, J. (2018). An energy-efficient crowd-sourcing-based indoor automatic localization system. *IEEE Sensors Journal*, *18*(14), 6009–6022.

Ma, Y., Zhu, X., Zhang, S., Yang, R., Wang, W., & Manocha, D. (2019). Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, pp. 6120–6127).

Mainetti, L., Patrono, L., & Sergi, I. (2014). A survey on indoor positioning systems. In *2014 22nd international conference on software, telecommunications and computer networks (softcom)* (pp. 111–120).

Mantini, P., & Shah, S. K. (2016). Multiple people tracking using contextual trajectory forecasting. In *2016 IEEE Symposium on Technologies for Homeland Security (HST)* (pp. 1–6).

Mukhopadhyay, S. C., Tyagi, S. K. S., Suryadevara, N. K., Piuri, V., Scotti, F., & Zeadally, S. (2021). Artificial intelligence-based sensors for next generation IoT applications: a review. *IEEE Sensors Journal*, *21*(22), 24920–24932.

Mussina, A., & Aubakirov, S. (2018). Rssi based Bluetooth Low Energy indoor positioning. In *2018 IEEE 12th International Conference on Application of Information and Communication Technologies (AICT)* (pp. 1–4).

Nessa, A., Adhikari, B., Hussain, F., & Fernando, X. N. (2020). A survey of machine learning for indoor positioning. *IEEE Access*, *8*, 214945–214965.

Nguyen, D. C., Cheng, P., Ding, M., Lopez-Perez, D., Pathirana, P. N., Li, J., ... Poor, H. V. (2020). Enabling AI in future wireless networks: a data life cycle perspective. *IEEE Communications Surveys & Tutorials*, *23*(1), 553–595.

Niculescu, D., & Nath, B. (2003). Ad hoc positioning system (APS) using AOA. In *Ieee infocom 2003. twenty-second annual joint conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)* (Vol. 3, pp. 1734–1743).

Noertjahyana, A., Wijayanto, I. A., & Andjarwirawan, J. (2017). Development of mobile indoor positioning system application using android and Bluetooth Low Energy with trilateration method. In *2017 international conference on soft computing, intelligent system and information technology (ICSIIT)* (pp. 185–189).

Pellegrini, S., Ess, A., Schindler, K., & Van Gool, L. (2009). You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th International Conference on Computer Vision* (pp. 261–268).

Piras, M., Marucco, G., & Charqane, K. (2010). Statistical analysis of different low cost gps receivers for indoor and outdoor positioning. In *IEEE/ION Position, Location and Navigation Symposium* (pp. 838–849).

Positioning Dataset. (2019, May). (https://zenodo.org/record/2647508#.YPBYOgzbIU)

Pritt, N. (2013). Indoor positioning with maximum likelihood classification of Wi-Fi signals. In *Sensors, 2013 ieee* (pp. 1–4).

Ramakrishnan, R., Gaur, L., & Singh, G. (2016). Feasibility and efficacy of ble beacon iot devices in inventory management at the shop floor. *International Journal of Electrical & Computer Engineering (2088-8708)*, *6*(5).

Saylam, A., Cikmazel, R. O., Kelesoglu, N., Nakip, M., & Rodoplu, V. (2021). Energy-Efficient Indoor Positioning for Mobile Internet of Things Based on Artificial Intelligence. In *Innovations in Intelligent Systems and Applications Conference* (pp. 1–6).

Saylam, A., Güzeliş, C., & Rodoplu, V. (n.d.). Machine Learning Enabled Sleep Time Estimation (MLE-STE) Architecture For Indoor Positioning In Energy-Efficient Mobile Internet Of Things (IoT) (submitted).

Saylam, A., Kelesoglu, N., Cikmazel, R. O., Nakip, M., & Rodoplu, V. (2021). Dynamic Positioning Interval Based On Reciprocal Forecasting Error (DPI-RFE) Algorithm for Energy-Efficient Mobile IoT Indoor Positioning. In *2021 International Conference on Computer, Information and Telecommunication Systems (CITS)* (pp. 1–5).

Shafer, I., & Chang, M. L. (2010). Movement detection for power-efficient smartphone WLAN localization. In *Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems* (pp. 81–90).

Shih, P.-L., Chiu, P.-J., Cheng, Y.-C., Lin, J.-Y., & Yi, C.-W. (2012). Energy-aware pedestrian trajectory system. In *2012 41st international conference on parallel processing workshops* (pp. 514–523).

Shin, D.-H., & Sung, T.-K. (2002). Comparisons of error characteristics between toa and tdoa positioning. *IEEE Transactions on Aerospace and Electronic Systems*, *38*(1), 307–311.

Smagulova, K., & James, A. P. (2019). A survey on lstm memristive neural network architectures and applications. *The European Physical Journal Special Topics*, *228*(10), 2313–2324.

Svalastog, M. S. (2007). *Indoor positioning-technologies, services and architectures* (Unpublished master's thesis).

Tahsien, S. M., Karimipour, H., & Spachos, P. (2020). Machine learning based solutions for security of Internet of Things (IoT): A survey. *Journal of Network and Computer Applications*, *161*, 102630.

Vu, T. H. N., Le, T. H., & Lee, Y. K. (2014). Forecasting moving object position based on temporal patterns. In *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)* (pp. 733–740).

Wang, C., Ma, L., Li, R., Durrani, T. S., & Zhang, H. (2019). Exploring trajectory prediction through machine learning methods. *IEEE Access*, *7*, 101441–101452.

Wang, Y., & Ho, K. (2016). TDOA positioning irrespective of source range. *IEEE Transactions on Signal Processing*, *65*(6), 1447–1460.

Xue, H., Huynh, D. Q., & Reynolds, M. (2018). SS-LSTM: A hierarchical LSTM model for pedestrian trajectory prediction. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 1186–1194).

Yin, Z., Wu, C., Yang, Z., & Liu, Y. (2017). Peer-to-peer indoor navigation using smartphones. *IEEE Journal on Selected Areas in Communications*, *35*(5), 1141–1153.

Yu, H. K., Oh, S. H., & Kim, J. G. (2020). AI based location tracking in WiFi indoor positioning application. In *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)* (pp. 199–202).

Yüksel, K. D., & Töreyin, B. U. (2018). A deep learning and RSSI based approach for indoor positioning. In *2018 26th Signal Processing and Communications Applications Conference (SIU)* (pp. 1–4).

Zeimpekis, V., Giaglis, G. M., & Lekakos, G. (2002). A taxonomy of indoor and outdoor positioning techniques for mobile location services. *ACM SIGecom Exchanges*, *3*(4), 19–27.

Zhang, L., Liu, J., & Jiang, H. (2012). Energy-efficient location tracking with smart-

phones for IoT. In *SENSORS, 2012 IEEE* (pp. 1–4).